Under consideration for publication in Math. Struct. in Comp. Science

Vulnerability Modelling with Functional Programming and Dependent Types

Cezar Ionescu, Potsdam Institute for Climate Impact Research, Germany

Received 13 October 2012

We present an interdisciplinary effort in the field of global environmental change, related to the understanding of the concept of "vulnerability". We have used functional programming to capture the generic aspects of the myriad of definitions of vulnerability, and have used the resulting formalisation to learn something new about vulnerability and to write some better software for vulnerability assessment. In the process, we have also found out something about formalisation in general, about the advantages and disadvantages of dependent types, and about the role of computing science in the larger intellectual landscape.

1. Introduction

"Translations can sometimes create a sense of *explanation*", says Goldblatt in his account of categorial logic (Goldblatt 2006). The act of translating from one language to another requires a much closer reading than usually practiced, and that is perhaps most true of translations in the language of mathematics. Writing a program, especially in a high-level declarative language such as Haskell or Agda, is a similar exercise, and it is the same resulting feeling of explanation which accounts for statements common in the software world, such as "you only understand it if you've implemented it". Moreover, the understanding achieved can be more easily shared with others by virtue of the fact that it is expressed in executable form. For the non-specialist, it is usually easier to play around with a program than with a theorem.

Perhaps the best-known example of using insights from functional programming to better understand a "real-world" domain is the work on financial contracts of Peyton-Jones and Eber described in (Peyton Jones *et al.* 2000; Peyton Jones and Eber 2003). Our work here aims to achieve similar results in the interdisciplinary world of global environmental change, by focusing on the concept of vulnerability and the task of vulnerability assessment in a computational context.

2. Vulnerability

In the past decade, the concept of "vulnerability" has played an important role in global environmental change (which includes fields such as climate change, development studies, food security, natural hazard studies, and so on). Vulnerability studies carried out within these fields have often been successful in alerting policymakers to precarious

situations. The importance of the concept in the particular field of climate change is described, for example, as follows (Lemmen and Warren 2004):

> ... Studies based primarily on the output of climate models tend to be characterised by results with a high degree of uncertainty and large ranges, making it difficult to estimate levels of risk. In addition, the complexity of the climate, ecological, social and economic systems that researchers are modelling means that the validity of scenario results will inevitably be subject to ongoing criticism. ... Such criticisms should not be interpreted as questioning the value of scenarios; indeed, there is no other tool for projecting future conditions. What they do, however, is emphasise the need for a strong foundation upon which scenarios can be applied, a foundation that provides a basis for managing risk despite uncertainties associated with future climate changes. This foundation lies in the concept of vulnerability.

Unfortunately, this foundation has problems of its own. The definitions of vulnerability differ across the fields mentioned, and even within any one field there seem to be a bewildering multitude of definitions to choose from (Thywissen, for example, summarises thirty-five definitions of vulnerability in (Thywissen 2006)!).
This is not the place to describe the detailed work of analysing these definitions: for that, we refer the reader to (Wolf 2008; Wolf 2010; Ionescu 2009a; Ionescu 2009b). However, we do want to give a couple of representative examples, in order to give a hint of the tantalising similarity between the many definitions and between the various uses of "vulnerability". We want to suggest that this is the same sort of similarity which emerges many times when writing computer programs: one finds oneself using the same computational patterns over and over again, with small yet important differences. Inventing the proper concepts and tools for describing these patterns as instances of the same structure has been one of the main driving forces of computer science, and therefore, the computer scientist should be in a good position to attempt to understand "the essence" of vulnerability.
On to the definitions. We start with the ordinary language definition, because it is the source of the more technical usage we will encounter later (besides being easier to understand for readers outside the fields of global environmental change).

2.1. *The Oxford English Dictionary definition*

The latest edition of the Oxford English Dictionary gives the following definition for "vulnerable" (OED 2005):

vulnerable (adj.):

1 exposed to the possibility of being attacked or harmed, either physically or emotionally: *we were in a vulnerable position* | *small fish are* vulnerable to *predators*
2 Bridge (of a partnership) liable to higher penalties, either by convention or through having won one game towards a rubber.

Vulnerability according to the OED is thus the condition of being vulnerable, and "vulnerable" is an adjective, a property that is predicated of something. This something is, in the context of the definition, the entity exposed to the possibility of harm. In the first example sentence, it is the position, in the second example sentence it is the small fish, and in the context of the game of Bridge, it is the partnership. The first of these is

somewhat surprising: one would expect "vulnerable" to be predicated of the subject of the sentence, "we", rather than of the position. However, this is an example of a transferred epithet (a *hypallage*), and the sentence can be interpreted as "we were in a position in which we were vulnerable".
The second example sentence introduces an *adjective complement*: the idea of vulnerability *to* something. The small fish are exposed to the possibility of being harmed or attacked *by the predators*. Here, "vulnerable" becomes a binary predicate, since it is relative not just to the entity exposed to the possibility of harm, but also to the cause of that harm. This is the typical usage of vulnerable in the context of Climate Change studies: "vulnerable *to* climatic change".
As a final remark, we note the emphasis on potentiality. The entity that is said to be vulnerable is not "exposed to harm", but "exposed to *the possibility* of harm".

2.2. *Vulnerability in the context of poverty analysis*

In development studies, the term "vulnerability" has gained prominence after being used in the 2000/1 World Development Report, where it was defined as "a measure of resilience against a shock – the likelihood that a shock will result in a decline in well-being" (The World Bank 2001). Vulnerability is here no longer a boolean predicate, but a measure which in general is going to take values other than *True* or *False*.
We can interpret this definition as a specialisation of the ordinary language usage, in the following way. The general idea of "being attacked or harmed" is replaced by the more specialised "suffers a decline in well-being". The "likelihood of . . . " refines perhaps the idea of "exposure to the possibility of . . . ", suggesting that zero likelihood represents impossibility, and that harm might be more or less possible.
Many other conceptualisations have been proposed in order to "operationalise" the World Bank definition, or to account for aspects that were felt lacking, such as taking into account the magnitude of the decline in well-being, not just the likelihood of that decline. In (Calvo and Dercon 2005), Calvo and Dercon propose the following definition of vulnerability *to poverty* as a synthesis of these various efforts:

> vulnerability is the magnitude of the threat of future poverty

where we have

1 The "magnitude of the threat" combines the likelihood of suffering poverty in the future, as well as the severity of the poverty in that case.
2 Vulnerability is "an ex-ante statement about future poverty", that is, it is a statement about an uncertain future.
3 This definition is meant to apply only to a particular situation: "[. . .] we are referring to vulnerability *to poverty*. Individuals face several other threats such as illness, or crime, or loneliness. Yet we focus on poverty in particular, as this was also the focus other authors arguably had in mind when using the term 'vulnerability'. We thus understand expressions such as 'vulnerability to an epidemic' as a shortcut to 'vulnerability to poverty due to an epidemic'. "

The last remark is interesting in the light of the second example sentence in the Oxford Dictionary: "small fish are vulnerable to predators". If we take some notion of "harm" that befalls the fish as analogous to poverty, then Calvo and Dercon seem to suggest

that this sentence is an abbreviation of "small fish are vulnerable to harm due to predators". In the Oxford Dictionary, "vulnerable to" was relative to the factors that induced the potential harm, here, "vulnerable to" is relative to the potential harm induced by the given factors.

This asymmetry between the usage of "vulnerability to ..." between climate change and development studies leads sometimes to charged debates, which are hard to understand for outsiders. The computer scientist involved in interdisciplinary research can perhaps see the resemblance to some disputes in his own field (see, for example, the entry under "holy wars" in the latest edition of Eric Raymond's Jargon file) and approach them with a certain degree of sympathy.

Calvo and Dercon propose a set of requirements on a measure of vulnerability, which further elucidate their definition:

1 Vulnerability measures a set of outcomes across possible states of the world. It is assumed that there are finitely many states of the world, and each state is assigned a probability that it will occur.
2 Poverty is defined in terms of a threshold, which has the same value in all states of the world.
3 The states of the world in which the outcomes are above the threshold do not enter in the vulnerability measurement (this is called the "axiom of focus").
4 Monotonicity requirements: the likelier the outcomes below the threshold, and the greater their distance to the threshold, the greater the vulnerability.

A measure of vulnerability to poverty which satisfies these requirements has then the form:

$$V = sum\ [p\ i * v\ (x\ i)\ |\ i \leftarrow [1\,..\,n]]$$

where

$n :: \mathbb{N}$ -- the number of possible states of the world
$p :: \mathbb{N} \to [0,1]$ -- $p\ i$ is the probability of state i
$v :: \mathbb{R} \to \mathbb{R}$ -- a monotonically decreasing and convex function
$x\ i = (y\ i)\ /\ z$ -- relative distance to threshold of outcome i
$y :: \mathbb{N} \to \mathbb{R}$ -- $y\ i$ is the outcome in state i if below the
-- threshold, 0 otherwise
$z :: \mathbb{R}$ -- the threshold

This measure generalises many of those proposed in the literature on development studies.

3. The IPCC definition of vulnerability

Within the climate change research community, the most influential definition of "vulnerability" is given by the Intergovernmental Panel of Climate Change in its assessment reports. The most recent of them, the Fourth Assessment Report (Parry *et al.* 2007), contains the following:

> *vulnerability*: the degree to which a system is susceptible to and unable to cope with, adverse effects of climate change, including climate variability and extremes. Vulnerability is a function of the character, magnitude and rate of climate variation to which a system is exposed, its sensitivity, and its adaptive capacity.

As in the previous section, vulnerability is not a boolean predicate, but one admitting degrees: "the degree to which ...". We can interpret the IPCC definition as a specialisation of the OED one: there is a notion of "harm", phrased as the occurrence of "adverse effects" with which the system is "unable to cope". The potentiality aspect is expressed by "susceptible" (versus, for example, "affected"). The role of climate change ("including climate variability and extremes") is that of an adjective complement, similar to the "predators" to which the small fish were said to be vulnerable. Here, the notion is of vulnerability *to climate change*. This is in contrast to the vulnerability *to poverty* defined in the previous section: climate change is the cause of harm here, poverty was the harmful effect there.
On the whole, there is an intention of making the context of vulnerability statements more precise than in the previous two sections. This is also apparent from the evolution of this definition, compare for example an earlier version (in the Second Assessment Report (Watson *et al.* 1995)): "vulnerability defines the extent to which climate change may damage or harm a system".
Along the same lines, the second sentence makes explicit some of the determinants of vulnerability. Besides a characterisation of the climate change factors that vulnerability is considered relative to, one should also take into account the "sensitivity" and the "adaptive capacity", defined as follows.

> *sensitivity*: the degree to which a system is affected, either adversely or beneficially, by climate variability or change. The effect may be direct (e.g., a change in crop yield in response to a change in the mean, range or variability of temperature) or indirect (e.g., damages caused by an increase in the frequency of coastal flooding due to sea-level rise).

> *adaptive capacity*: the ability of a system to adjust to climate change (including climate variability and extremes) to moderate potential damages, to take advantage of opportunities, or to cope with the consequences.

4. The structure of vulnerability

Common to all the definitions and uses of vulnerability we have seen is (at least) the idea of potential negative outcomes, formulated by the Oxford Dictionary as "exposed to the possibility of being harmed". A first difficulty in modelling comes from having to deal with the concept of "possibility". There are several alternatives: for example, one could try to formulate vulnerability within standard modal logic with possibility.
Instead, we decided to consider a temporal view of possibility: something is possible if it could happen in the future. This interpretation is consistent with most vulnerability assessments: computer programs, scenarios, or some form of statistical analysis are used to determine the possible future evolutions, or characteristics of these future evolutions.
What are these evolutions of? A natural answer is "of the world", or, to be more precise,

"of the state of the world" (this is also the terminology of Calvo and Dercon above). Accordingly, let us assume that we are given a set *State* of "states of the world", or "states of affairs". An evolution might be represented by a list of states, or by a function from some time set into states, and so on. For the moment, let us just assume that *Evolution* is the type of such evolutions. We might then model possibility by a function which tells us, given a state, which are the possible future evolutions of the world:

$$possible :: State \to [Evolution]$$

However, this is not very generic. In many cases, we do not have just a list of possible evolutions, but also some more information: for example, we might know the probabilities of the evolutions, or their likelihood might be expressed by fuzzy degrees of membership, or they might be ranked according to their plausibility, etc.
A convenient way to accommodate all these cases is to assume that the possible future evolutions form a functorial structure:

$$possible :: Functor\ f \Rightarrow State \to f\ Evolution$$

Given a state, we now have a structure of possible evolutions. We are interested in the potential harm, or damage, or adverse effects. Usually, if the evolution is known, it is easy to ascertain if and to what degree harm has befallen the entity under consideration. If we know, for example, how the deal has played out, we know whether the partnership has overbid and how much they have lost as a result. If we look at a complete evolution of some coastal area over fifty years, we can tell how many people have lost their livelihoods (or, indeed, their lives) because of sea-level rise. This suggests that the notion of harm can be modelled by a function which acts on evolutions. "Harm" has negative connotations, there might be more or less of it along a trajectory: more is worse. Thus, we want the target of the function to be some sort of (possibly partial) preorder. Assuming a suitably defined typeclass *Preorder* we have

$$harm :: Preorder\ v \Rightarrow Evolution \to v$$

So now, given a state, we can obtain a structure that represents the possible harm values that might befall the entity we are interested in by applying $fmap\ harm \cdot possible$. This is not yet vulnerability, which is not "possibility of harm" or "susceptibility to adverse effects", but rather a measure of this possible harm or susceptibility: a boolean one, perhaps, as in "exposed to the possibility of harm", or a more nuanced one, such as "degree to which a system is susceptible to adverse effects". In any case, the values of this measure can, like the values of harm, be compared, so it is natural to require that they too form a preorder:

$$measure :: Preorder\ w \Rightarrow f\ v \to w$$

Putting it all together, we obtain the structure of vulnerability:

$$vulnerability = measure \cdot fmap\ harm \cdot possible$$

Before we go any further, let us remark that it is only at this point that one can really enter into an interdisciplinary dialogue. If we squint away the *fmap* and the function composition symbols, this looks very close to a natural language definition: "vulnerability is a measure of possible harm". For many a social scientist, mathematics has to do with "numbers", and in any case is supposed to take the form of cryptic strings of cuneiforms of which one can make sense only after abandoning any shred of

intuition about the human world, and any trace of human sensibility one might possess. Therefore, this kind of analysis comes as a surprise and is somewhat of a door-opener. It is easy to see that the definitions we have seen so far can be interpreted in this structure. A boolean reading of "exposed to the possibility of harm" might have:

$$
\begin{aligned}
&\mathit{possible} \;::\; \mathit{State} \to [\mathit{Evolution}]\\
&\mathit{harm} \;::\; \mathit{Evolution} \to \mathit{Bool}\\
&\mathit{measure} \;::\; [\mathit{Bool}] \to \mathit{Bool}\\
&\mathit{measure} = \mathit{or}
\end{aligned}
$$

As another example, Calvo and Dercon's vulnerability to poverty has (using the notation in 2.2 where n and p are defined):

$$
\begin{aligned}
&\textbf{type}\ \mathit{State} = \mathit{Int} \quad \text{-- better would be the set 0, 1, ..., n-1}\\
&\textbf{type}\ \mathit{Evolution} = \mathit{State} \quad \text{-- one-step evolution, with initial step fixed}\\
&\textbf{newtype}\ \mathit{SimpleProb}\ a = \mathit{SP}\ [(a, \mathit{Float})] \quad \text{-- simple probability distribution}\\
&\mathit{possible}\ \_ = \mathit{SP}\ [(i, p\ i) \mid i \leftarrow [0\,..\,n-1]]\\
&\mathit{harm} \quad = v \cdot x\\
&\mathit{measure} = \mathit{sum}
\end{aligned}
$$

And so on: Wolf (Wolf 2010) has shown how to read dozens of definitions found in the literature in terms of this simple structure. Nevertheless, this is not the entire story of modelling vulnerability, it is just a reasonable start. Among the missing elements is a more detailed view of evolutions, which we have left unspecified. It would seem reasonable to define them in terms of states, e.g. as functorial structures of states:

$$\textbf{newtype}\ \mathit{Evolution}\ g = \mathit{Functor}\ g \Rightarrow E\ (g\ \mathit{State})$$

making *possible* a coalgebra with carrier *State*. Coalgebras are extensively applied in the study of dynamical systems and modal logics, and indeed the intuition behind the *possible* function is that of a dynamical system (which describes possible future evolutions). This point of view is further developed in greater detail in (Ionescu 2009a), see also (Lincke *et al.* 2009).

5. Applications

Now that we have teased out the basic structure of vulnerability, it is time to see what are the kinds of things that we can do with it. In this section, we present four applications of the model: we use it to explore the informal usage, to make the technical usage more precise and more correct, to help compare vulnerability assessments and reuse their results, and to ensure the correctness of software components for vulnerability assessment.

5.1. *New perspectives on the informal usage of vulnerability*

As a very simple example, consider the translation we made above of the OED definition. We considered *possible* to be a list valued function, and *harm* a boolean one: either there is some harm, or there isn't. Vulnerability came out as a predicate: $\mathit{vulnerability} :: \mathit{State} \to \mathit{Bool}$. The extension of this predicate is represented by the states in which the entity under consideration might come to harm (or be attacked) in a

possible future evolution. What we have here, therefore, is a mathematical analogue of the transferred epithet we found in the first example given by the OED: "we were in a vulnerable position", which seemed to assign the exposure to harm to the "position", instead of to "us".
For a less trivial example, consider the difference between using "vulnerable to ⟨*negative consequence*⟩", as is used in development studies ("vulnerable *to poverty*"), versus "vulnerable to ⟨*cause*⟩", as preferred by the climate change community ("vulnerable *to climate change*").
The OED example sentence, "small fish are *vulnerable to* predators", is similar to the second kind of usage. To translate it in our structure, the most natural approach is to refine the boolean harm assessment above to take into account only the situations in which the predators have influenced the evolution:

$$\begin{aligned} &\mathit{predators} :: \mathit{Evolution} \to \mathit{Bool} \\ &\mathit{wounded} :: \mathit{Evolution} \to \mathit{Bool} \\ \\ &\mathit{harm} :: \mathit{Evolution} \to \mathit{Bool} \\ &\mathit{harm} = \mathit{wounded} \wedge \mathit{predators} \end{aligned}$$

We are now computing the resulting harm across an evolution by combining the impacts (some of the little fish have been wounded) with the presence of the predators.
This is a general pattern: the computation of harm results from assessing the some impacts along those evolutions which are influenced by some factors of interest (such as climate change or predators). Both the choice of which impacts to assess, and of which evolutions are considered relevant, are subjective: for example, health studies disregard the potential monetary losses of the tobacco industry, emphasising instead the loss of life caused by smoking (while at the same time largely ignoring climate change).
Similarly, in the context of development studies the focus is on evolutions which end up with a population subject to poverty, no matter what the cause might be, while the climate change community is interested in those evolutions influenced by anthropogenic global warming, no matter what the impacts might be. This distinction is probably what causes the different emphasis on the components of the *harm* function. Moreover, as we have seen, there is a certain symmetry in the combination between impacts and factors of interest, which makes it particularly easy to slip from one form of expression to the other. Expressing this in a simple mathematical form has sometimes helped us forestall what might have developed in a heated debate, and given both sides an appreciation of the other's position.

5.2. *Correctness of vulnerability assessments*

Until now, we have just looked at very general structure, a composition of three functions of very general types. Except for the preorder requirements on the targets of the *harm* and *measure* functions, we have not constrained the structure in any way which would make it "for vulnerability". This is about to change.
Let us look again at the *measure* function, fixing the type variables:

$$\mathit{measure} :: F\ V \to W$$

This function is supposed to act on structures of values representing possible harm, and

returns an assessment of this possible harm. It seems natural to require that if we increase the harm values in a structure, the measure should return higher values: the more possible harm, the higher the vulnerability (all relations are considered non-strict). This leads us to formulate the following definition:

The monotonicity condition for vulnerability measures:
For all increasing functions $inc :: V \to V$ (i.e. for all $v :: V$ we have $inc\ v \sqsupseteq v$) and for all $v :: F\ V$, we have:

$$measure\ (fmap\ inc\ v) \sqsupseteq measure\ v$$

We have formalised the idea of increasing the harm values in a structure by mapping an increasing function on the structure, making use of the functoriality of F (in fact, it is difficult to imagine how one would do without that assumption).
Most measures we encounter in practice satisfy this common-sense condition, but not all. Some vulnerability assessments use a form of "democratic choice" measure: take the most frequent occurrence, the most common result, or the likeliest harm value. This kind of measure does not satisfy the monotonicity condition. For example, take
$F = SimpleProb,\ V = \mathbb{N}$, and define the likeliest harm value measure:

$$\begin{aligned}
&measure :: SimpleProb\ \mathbb{N} \to \mathbb{N}\\
&measure\ (SP\ ns) = snd\ (maximum\ (map\ swap\ ns))\\
&\quad \textbf{where}\\
&\quad swap\ (x, p) = (p, x)
\end{aligned}$$

The *swap* is necessary because in Haskell the order on pairs is the lexicographical ordering. Take

$$\begin{aligned}
ns &= SP\ [(10, 0.4), (0, 0.3), (1, 0.3)]\\
inc &:: \mathbb{N} \to \mathbb{N}\\
inc\ 0 &= 1\\
inc\ (n+1) &= n+1
\end{aligned}$$

The function *inc* is obviously non-decreasing, but
$measure\ xs = 10 > measure\ (fmap\ inc\ xs) = 1$. Thus, the measure "likeliest harm or impact" fails the monotonicity condition and cannot be used as a vulnerability measure. The monotonicity condition is about the only thing that the various groups of scientists actually performing vulnerability assessments could unhesitatingly sign off on. As such, it has come as a big surprise that it was violated in some cases. Since it is simple to explain and understand, this is is perhaps the best example of something quite useful coming out of a formalisation effort.

5.3. *Assistance in comparing vulnerability assessments*

In computational vulnerability assessments, the possible evolutions are given by computer models. In climate change, these usually take the form of a list of trajectories, each associated to a different scenario of political measures, technological change, or resource exploitation. The functor in the type of *possible* is in this case the (non-empty) list functor *List1*. In development studies, the models are generally stochastic, and each evolution has a probability of realisation. Therefore, the functor that gives the structure of possible evolutions is *SimpleProb*. Often we have combinations of such possibilities:

lists of stochastic evolutions (leading to *List1* (*SimpleProb Evolution*), or probability distributions over possible groups of scenarios (*SimpleProb* (*List1 Evolution*)).

The structure that holds the possible evolutions is the most volatile element of a vulnerability assessment. The choice of the *harm* or *measure* functions are largely the result of political decisions, and changing them requires a discussion between many parties, but the type of the functor used depends only on the model that is used to compute the evolutions, and that is largely the choice of a few scientists. Indeed, politicians expect the scientists to use several models, probably because of lack of confidence in any single model, combined with a certain faith in a democratic process.

The problem is that, unfortunately, the vulnerability measure also depends on the selected functor. Therefore, the question is, assuming we change the underlying model type from *F1* (say, the list functor) to *F2* (for example *SimpleProb*), can we reuse the vulnerability measure defined on *F1* to obtain a "compatible" one on *F2*?

A couple of examples show that there is some hope for this. For instance, there is a strong intuition that averaging a list of numbers should be compatible with taking the expected value of a probability distribution, or that taking the maximum of a list should be compared to taking the maximal value with non-zero probability. By the same token, taking the maximum is not compatible with the expected value, nor is the minimum compatible with the maximum, and so on. Several other such examples are put forward and analysed in (Ionescu 2009a), but these should already suffice to get the idea: we seem to have strong intuitions in cases where there is a natural transformation from one structure to the other. This suggests the following definition:

Take two vulnerability measures $m_1 :: F1\ V \to W$ and $m_2 :: F2\ V \to W$. Consider a natural transformation $\tau :: F1\ a \to F2\ a$. Then, we say that m_1 and m_2 are compatible with respect to τ if, for any $v1, v2 :: F1\ V$, we have

$$m_1\ v1 \sqsubseteq m_1\ v2 \Rightarrow m_2\ (\tau\ v1) \sqsubseteq m_2\ (\tau\ v2)$$

This is a natural kind of monotonicity requirement. The interesting thing about it is that it can provide the sort of reuse we needed:

Proposition 1. Compatible vulnerability measure. With the notation above, given a vulnerability measure m_2 and a natural transformation τ, take

$$m_1 = m_2 \cdot \tau$$

Then m_1 is a vulnerability measure compatible with m_2 with respect to τ.

The compatibility condition is trivially fulfilled. To verify that m_1 is indeed a vulnerability measure, take an arbitrary increasing function $inc :: V \to V$. Then, for any $v :: V$, we have:

$$
\begin{aligned}
& m_1\ (fmap\ inc\ v) \\
= \quad & \{\ \text{definition of } m_1\ \} \\
& m_2\ (\tau\ (fmap\ inc\ v)) \\
= \quad & \{\ \text{naturality of } \tau\ \} \\
& m_2\ (fmap\ inc\ (\tau\ v)) \\
\sqsupseteq \quad & \{\ m_2 \text{ vulnerability measure}\ \} \\
& m_2\ (\tau\ v) \\
= \quad & \{\ \text{definition of } m_1\ \} \\
& m_1\ v
\end{aligned}
$$

For example, the natural transformation

$$
\begin{aligned}
& probToList1 :: SimpleProb\ a \to List1\ a \\
& probToList1\ (SP\ aps) = map\ fst\ aps
\end{aligned}
$$

would give, from the maximum harm measure on lists, the maximum non-zero probability harm on simple probability distributions, while

$$
\begin{aligned}
& list1ToProb :: List1\ a \to SimpleProb \\
& list1ToProb\ as = SP\ [(a, p) \mid x \leftarrow xs]\ \textbf{where}\ p = 1.0\ /\ realToFrac\ (length\ xs)
\end{aligned}
$$

gives, from the expected value measure, the average value of a list measure.

5.4. *Software correctness: testing and proving*

The complete vulnerability model was developed in Haskell, and we have shown how to use the software components we developed in order to re-implement some of the computational vulnerability assessments in a generic fashion. The Haskell library was quite inefficient, and using it for any realistic assessment would have been a hopeless task, yet it served a useful proof of concept and prototyping role. In the meantime, the main components have been translated to more efficient C++ code (Lincke *et al.* 2009), making them also more accessible to the mainstream scientific programmers, who care a lot about performance and interoperability with existing tools, and rather less about functional programming.

One of the benefits of using a high-level programming language as a vehicle for formalisation is that we can easily transform the mathematical conditions, such as the monotonicity condition, into tests that have to be passed by the implementation. This is even more so when one has the benefit of a tool such as QuickCheck (Hughes 2000; Claessen and Hughes 2003). For example, this is the test for the monotonicity condition:

$$
\begin{aligned}
& test\_monotonicity\ measure\ geninc\ v = forAll\ geninc \\
& \quad (\lambda inc \to (measure\ (fmap\ inc\ v) \sqsupseteq measure\ v))
\end{aligned}
$$

We have to use a custom generator (*geninc*) which guarantees that the functions it generates are increasing. A naive attempt such as

$$
\begin{aligned}
& test\_monotonicity\ measure\ inc\ mv = increasing\ inc \Rightarrow \\
& \quad mv \sqsubseteq measure\ (fmap\ inc\ mv)
\end{aligned}
$$

is in most cases inadequate because arbitrarily generated functions are unlikely to be

increasing, and QuickCheck will stop with an inconclusive result once it reaches the maximum number of attempts for which it is configured.
Even with a custom generator, the coverage for large dimensional spaces can be quite poor. Unfortunately, many of the conditions imposed on the elements of our library are of this nature: they quantify over large spaces, and naive testing is likely inadequate. Moreover, it is not easy to formulate tests generically. We end up having to write a new custom generator for every change in the type of harm values, with all the disadvantages this brings: waste of time, introducing new sources of errors, and it's boring.
The strange thing is that, in all the cases we have seen, it is trivial to *prove* that the monotonicity condition is satisfied, or to show that it isn't. In fact, it is even trivial to do the proofs *formally*. The proofs are also more generic than the corresponding tests: for example, we can show that the maximum of non-empty lists measure fulfils the monotonicity condition no matter what the type V is, but we can hardly write a generator for increasing functions independent of their type.
This is a bit of a reversal of the common software engineering wisdom that testing is easier than proving, and a different argument for the latter than the Dijkstra dictum that "testing can only show the presence of bugs, never their absence". What we have found is that good testing is hard, and in many cases harder than proving the condition to be tested.
As a result of this, at this stage we have switched from using Haskell as the carrier of our formalisation, to Agda. This allows us to put all the specifications in the code: a preorder is represented by a record which includes the reflexivity and transitivity assumptions, and the monotonicity condition is expressed by

```
record VulnMeasure {f : Set → Set}   {v w : Set}
                   (F : Functor f)   (V : Preorder v)
                   (W : Preorder w) (measure : f v → w) : Set where
  field
    MonotonicityCondition  : ∀ {x inc}
      (I : Increasing V inc) → W ⊢ measure F : fmap inc x ⊒ measure x
```

which tells the whole story in the code (at least to those who know that the $\_ \vdash \_ \sqsupseteq \_$ function extracts and flips the comparison function from a preorder, and the $\_ : fmap \_$ function does the same to the *fmap* of the functor record).
On the other hand, this is perhaps too much information for our partners from the social sciences, who might still prefer the less precise Haskell version. Moreover, programming with dependent types is sometimes quite tricky, and reverting to Haskell for a less precise prototype is something we do quite often (that is the reason for adding "at this stage" in the first sentence of the previous paragraph: we might have been less successful if we had started out with Agda).
Finally, we have implemented several examples of vulnerability assessment in Haskell. Some of them were quite expensive computationally, despite their "toy program" nature. At present, Agda cannot compete on this front, so we still have to use other programming languages for efficiency reasons.

6. Conclusions and Perspectives

Here are some of the lessons we have learnt from formalising vulnerability:

1 formalisation has a lot in common with generic programming: in particular, the method of writing a high-level (pseudo-) functional program proved very fruitful
2 using a programming language which is syntactically close to the mathematical language makes it easy to implement the formalised concepts
3 dependent types are the best tool we found for expressing the kind of higher-order conditions that appeared in the formalisation, and for ensuring that the implementations satisfy these conditions
4 the existence of dependently typed programming languages does not make a language like Haskell obsolete: we can use it to make less precise, prototypical formalisations, and revert to it for more efficient computations.

The formalisation has given rise to a series of on-going activities: extending our library of vulnerability measures, compatible transformations, and proofs of monotonicity conditions; comparative analysis of various vulnerability assessments (Hinkel 2011); writing C++ software components for generic vulnerability modelling (Lincke *et al.* 2008; Lincke *et al.* 2009); or investigations of other kinds of mathematical structure that could be used in vulnerability assessments (Wolf 2010).
On the more conceptual side, we are currently working on formalisation and analysis of the various equilibria used in economical modelling (Pareto, Nash, Walrasian, correlated, etc.). We hope to contribute to the correctness and transparency of the economical models used for integrated assessments for policy advice. The difficulty here is that integrated assessments are supposed to combine the results of different disciplines: in particular, this is reflected in coupling economic models with models of climate change, land use, urban development scenarios, and more. If the assumptions of one model are not met by another, their coupling can lead at best to a crash, and at worst to seemingly plausible, but completely arbitrary results. Unfortunately, these assumptions are rarely specified, in fact, in many cases they are not even known. To tackle this problem requires, even in the simplest settings, an interdisciplinary effort. We hope that we have shown that computing science has more to offer to such an enterprise than computer support.

Acknowledgements

Special thanks are due to Richard Klein for providing the initial impulse for this work, and for contributing to its development. I have profited greatly from discussions with Rupert Klein, Paul Flondor, and the members of the FAVAIA project and of the Cartesian Seminar at PIK. I am very grateful for the constructive comments and suggestions of the anonymous reviewers.

References

Calvo, C. and Dercon, S. (2005), *Measuring Individual Vulnerability*. University of Oxford, Department of Economics, Economics Series Working Papers. Available at `http://ideas.repec.org/p/oxf/wpaper/229.html`.

Claessen, K. and Hughes, J. (2003), Specification Based Testing with QuickCheck. In *The Fun of Programming*, Cornerstones of Computing, Palgrave, 17–40.

Goldblatt, R. (2006) *Topoi, The Categorial Analysis of Logic*. Dover Publications, Inc.

Hinkel, J (2011). Indicators of vulnerability and adaptive capacity: Towards a

clarification of the science-policy interface. *Global Environmental Change*, **21** (1), 198–208.

Hughes, J. (2000) *QuickCheck: An Automatic Testing Tool for Haskell*. User manual avaiable online at `http://www.cs.chalmers.se/~rjmh/QuickCheck/manual.html`

Ionescu, C. (2009) *Vulnerability Modeling and Monadic Dynamical Systems*. PhD thesis, Department of Mathematics and Informatics, Free University Berlin, Germany, 2009.

Ionescu, C. and Klein, R.J.T. and Hinkel, J. and Kavi Kumar K.V.S. and Klein, R (2009). Towards a formal framework of vulnerability to climate change. In *Environmental Modelling and Assessment*, **14** (1), 1–16, Springer Netherlands.

Lemmen, D. and Warren, F. (2004) *Climate Change Impacts and Adaptation: A Canadian Perspective*. Natural Resources Canada, 2004.

Lincke, D. and Ionescu, C. and Botta, N. (2008), *A generic library for earth system modelling based on monadic systems*. In proceedings of *Digital Earth Summit on Geoinformatics: Tools for Global Change Research*, Ehlers, M., Behncke, K. and Gerstengarbe, F. Editors, Wichmann.

Lincke, D. and Jansson, P. and Zalewski, M. and Ionescu, C. (2009), Generic Libraries in C++ with Concepts from High-Level Domain Descriptions in Haskell. A Domain-Specific Library for Computational Vulnerability Assessment. In *Domain-Specific Languages, Proceedings of the IFIP TC 2 Working Conference DSL 2009, Oxford, UK, July 15-17 2009*, Editor Taha, W.M.

Oxford Dictionary of English. Oxford University Press, Oxford, UK, Second edition (revised), 2005.

Parry, M. and Canziani, O. and Palutikof, J and van der Linden, P. and Hanson, C. (2007), *Climate Change 2007: Impacts, Adaptation and Vulnerability*. Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, Cambridge, UK.

Peyton Jones, S. and Eber, J.-M. and Seward, J., Composing contracts: an adventure in financial engineering. In *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, ACM, 2000.

Peyton Jones, S. and Eber, J.-M., How to write a financial contract. In *The Fun of Programming*, Cornerstones of Computing, Palgrave, 105–129.

Thywissen, K. (2006), *Components of Risk, A Comparative Glossary*. SOURCE - Studies Of the University: Research, Counsel, Education, UNU-EHS, 2.

Watson, R.T. and Zinyowera, M.C. and Moss, R.H. (1995), *Climate Change 1995: Impacts, Adaptations and Mitigation of Climate Change: Scientific-Technical Analyses*. Contribution of Working Group II to the Second Assessment of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, UK.

Wolf, S. and Lincke, D. and Hinkel, J. and Ionescu, C. and Bisaro, S. (2008), *Concept Clarification and Computational Tools – A Formal Framework of Vulnerability*. FAVAIA Working Paper 8, Potsdam Institute for Climate Impact Research, Available at `http://www.pik-potsdam.de/favaia/pubs/favaiaworkingpaper8.pdf`

Wolf, S. (2010) *From vulnerability formalization to finitely additive probability monads*. PhD thesis, Department of Mathematics and Informatics, Free University Berlin, Germany, 2010.

World Bank, The (2001) *World Development Report 2000/2001: Attacking Poverty*. Oxford University Press.