Functional Hybrid Modelling towards mathematical physics

Work in progress report

Dmitrii Legatiuk
DFG Research Training Group 1462, Bauhaus-Universität Weimar, Germany
Weimar
dmitrii.legatiuk@uni-weimar.de

Henrik Nilsson
Functional Programming Laboratory, School of Computer Science, University of Nottingham, UK
Henrik.Nilsson@nottingham.ac.uk

ABSTRACT

Modelling languages have become an indispensable to practising engineers. They offer modelling at a high level of abstraction backed by features such as automatic simulation and even derivation of production code. However, partly because of the offered automation, modelling languages are limited to specific application areas: to our knowledge, no modelling language supports mathematical physics modelling in its full generality. Yet, when developing large, coupled, multiphysics models, there is a clear need for such an overarching language to ensure the coherence of the model as a whole, even if submodels ultimately are realised in modelling languages targeting specific domains or are pre-existing. In prior work, it was demonstrated how treating models as abstract objects in category theory offers one way to ensure coherence of key aspects for composite models. Type theory offers complementary approaches. This paper presents a first step towards a language supporting abstract modelling in mathematical physics with the aim of ensuring coherence of coupled multiphysics models early in the design process. To that end, following the approach of Functional Hybrid Modelling (FHM), we discuss how a language supporting quite general modelling equations can be realised as an embedding in Haskell. The appeal of the proposed approach is that only very few core concepts are needed, which greatly simplifies the semantics. The appeal of an embedded realisation as such is that much of the language infrastructure comes for free.

CCS CONCEPTS

• **Theory of computation → Type theory**; • **Computing methodologies → Modeling methodologies**; *Representation of mathematical functions*; *Simulation languages*; Continuous models;

KEYWORDS

Modelling, Haskell, Functional Hybrid Modelling, abstraction

ACM Reference Format:
Dmitrii Legatiuk and Henrik Nilsson. 2017. Functional Hybrid Modelling towards mathematical physics. In *Proceedings of 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Munich, Germany, December 2017 (EOOLT 2017),* 4 pages.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The solution of any engineering problem starts with a modelling process. Conceptually, there are two phases: we can call them the abstract and the concrete modelling phase. The abstract phase is concerned with capturing all relevant aspects of the problem as mathematical equations. The concrete phase is concerned with turning these equations into artefacts, such as programs, that can then be used to study chosen aspects of the problem, often (but not exclusively) through simulation. The abstract model should be as general as possible, subject to decisions about what aspects are of interest to model, so as to not unduly constrain the concretisation phase. Indeed, modern problems of engineering are typically coupled problems in that a number of concrete models need to be used together. For example, it is common that no one method suffices for simulating all parts and aspects of a system. Or there may be existing models system parts or aspects that have to be used. A choice among different kinds of models is typically involved. Creation of a realistic model thus requires careful thinking about the modelling *process*, making the modelling process itself a fundamental part of engineering practice that currently is not always addressed properly.

In recent years, dedicated modelling languages have become popular among practising engineers as a more expedient alternative to main-stream programming languages. Some modelling languages are specific to particular application domains. They can thus offer modelling at a high level of abstraction, allowing, for example, simulation code to be derived automatically. Their scope is, however, highly limited. Other modelling languages, such as Simulink [1], aim to be domain-agnostic. But they are geared towards the concrete phase in that they are rather prescriptive about the simulation process. Yet other languages, such as Modelica [2], support the abstract modelling phase more directly and take care of the subsequent concrete phase more or less automatically. But the price for this is again limited applicability in that only specific kinds of equations are supported for the abstract modelling, such as Differential-Algebraic Equations in the case of Modelica. That means that many problems in the wider setting of mathematical physics, such as boundary value problems of elasticity theory, fluid mechanics, electrodynamics etc., cannot be modelled adequately as partial differential equations are necessary in these cases.

In some cases it is possible to work around such limitations within existing languages, but the consequence is invariably that the model becomes "contaminated" by concrete aspects, i.e. operational detail, making it less general. There is thus a need for language support for the abstract modelling phase in its full generality that currently is largely unmet. Of course, we cannot hope for the same level of automation in such a setting. But, by capturing appropriate static properties, through a type system for systems modelling, we can check for consistency ensuring that the modelling process has been carried out appropriately. This would catch many system modelling errors early, at the abstract level. Thus, our primary goal is to support the modelling process on the abstract level to ensure that the construction of a multiphysics model does not contradict to basic physical laws and assumptions before preparing a simulation code.

Originating from Functional Reactive Programming [3], Functional Hybrid Modelling (FHM) [4, 5] is a first step in the development of a Haskell-based modelling language with a robust type system allowing us to detect modelling errors more effectively compared to the type systems in the traditional modelling languages. In essence, FHM enriches Haskell with a notion of first-class equations for modelling of physical systems. Haskell provides the required abstraction mechanisms and means for constructing systems by composing equation system fragments. While, in terms of language design and modelling aims, FHM is similar to Modelica and thus suffers the same limitations, the design of FHM can serve as a blueprint for a language for modelling systems in the abstract, demonstrating the usefulness of Haskell for this kind of modelling language research and design.

The extension of the FHM towards mathematical physics requires at first a deeper understanding of the modelling process. Particularly, an analysis of possible sources of modelling errors must be performed carefully with the idea to develop tools to detect the modelling errors at the abstract level. For that purpose, it is necessary at first to formalise the modelling process, otherwise a developed methodology for modelling error detection would be an application-specific. An attempt to work with the models in engineering on a more abstract level has been proposed in [6], where graph theory has been utilised. In this approach, models are considered as vertices of a graph and couplings between the models as edges. However, the focus of these works was not related to the evaluation of the modelling process, rather than to the evaluation of models in the concrete modelling phase based on uncertainty and sensitivity analyses. As an alternative, a modelling framework based on category theory has been proposed in [7]. In this framework, mathematical models are treated as abstract objects in categories, and a coupling of models is described by functorial mappings between these categories. By help of such an abstract modelling approach the strategy to check a consistency of the modelling process has been presented. The consistency in this case is a tool for checking if the modelling process has been done appropriately, i.e. a tool to detect modelling errors. Such an abstract modelling framework provides a necessary understanding of the modelling process in general. The categorical approach presented in [7] is a starting point for the extension of the FHM to more sophisticated mathematical models, serving as the semantics that would underpin such a language.

In this short work in progress report we present first ideas for the extension of the FHM to problems of mathematical physics. Especially, the aim of the paper is to show how equations of mathematical physics can be transferred to the FHM-like syntax, which is a basis for the further extension of the FHM. We further discuss first steps in the development of a type system covering models of mathematical physics with a particular focus on underling advantages offered by the use of FHM together with ideas for further Haskell implementation. The type system should serve the purpose of identification of modelling errors and provide a sufficient expressive power to describe a coupling of different models for building a coupled multiphysics model. This is an overall goal, however in this short paper we present a simple academic example illustrating issues which have to be addressed while extending the FHM.

2 MATHEMATICAL MODELLING AND TYPES

A huge variety of models appears in practice in mathematical physics. For example, they may be described using ordinary and partial differential equations, integral equations, stochastic equations etc. Models for coupled multiphysics problems may involve many different kinds of equations. However, for starting the discussion on how to realise an FHM-like language supporting such equations, we start with a simple one-dimensional model described using a single partial differential equation. Such a simple model suffices for presenting the general ideas and discussing central issues.

2.1 Example Model

Let us consider the following partial differential equation describing transverse vibrations of a solid beam in the framework of Rayleigh beam theory:

$$\rho F \frac{\partial^2 u}{\partial t^2} + EI_y \frac{\partial^4 u}{\partial x^4} - \rho I_y \frac{\partial^4 u}{\partial x^2 \partial t^2} = q(x,t). \quad (1)$$

Here, ρ is the material density, F is the area of cross section, E is the Young modulus, I_y is the moment of inertia, $q(x,t)$ is an applied external load and $u(x,t)$ is the displacement of points of the beam.

Equation (1) is particularly interesting from a modelling language point of view, since multi-domain modelling languages, such as Simulink and Modelica, tend to lack support for partial differential equations. Sometimes this can be addressed by converting a partial differential equation into a system of ordinary differential equations by discretising spatial or temporal variables. However, for equations involving mixed derivatives, as in our example, this approach, in general, is not applicable, and even where applicable, such transformations are cumbersome and obscure the model. To support mathematical physics modelling in general, intrinsic support for key equation types appearing in engineering practice are needed, and then in particular partial differential equations.

Note that for a complete formulation of an engineering problem, boundary and initial conditions must be added to (1). For our purposes here, however, it is enough to consider the model given only by equation (1).

2.2 Preliminaries

Solutions of problems of mathematical physics typically have a clear physical interpretation. It is thus attractive to start our construction by taking the physical background of a model into account. For example, for the Rayleigh model we might introduce the polymorphic type of displacements as `S` α = `(Time, Coordinate)` $\rightarrow \alpha$, where `S` α is the type of a displacement whose instantaneous value is of type α. This approach has the obvious advantage that the physical background of a problem is directly reflected in the types. However, it also implies that each problem of mathematical physics requires, in general, a unique type set of types, which is significantly limits reuse.

Instead, we propose to work directly with functions and operators used in mathematical models to describe a given physical phenomenon, forgetting the physical background for now. The idea is to introduce a set of independently typed primitives that may be combined into models of arbitrary complexity. For example, inspection of the Rayleigh model reveals that it is constructed from basic operations such as partial differentiation, multiplication, and addition. The connection to the physical reality can be re-established by a type system that keeps track of physical dimensions. See for example [8], [9], or languages like F#.

Note that complexity here is understood as a model complexity in the sense of [7], not related to the notion of computational complexity. In our case, complexity is understood, roughly, as a measure of how many different physical phenomena are involved in a problem. See [7] for the details.

2.3 Functional Embedding

The original work on FHM [4] introduced the idea of embedding differential equations in a functional language through first class notions of functions and relations on *signals*: *signal functions* and *signal relations*. The term "signal" was a natural choice as FHM is designed for initial value problems represented by DAEs. The situation in mathematical physics is more general as models may depend on several variables, or in the case of a static problem, be time-independent. So perhaps a different term should be chosen for the unknowns in this new setting. However, we will stick with *signal* for the unknowns for now, on the understanding that signals may be *temporal*, *spatial*, *spatio-temporal*, or even *constant*. Thus, a signal is (in general) a function of several variables used and (part of) a solution to the equations defining the model. We note that keeping track of different *kinds* of signals offer another opportunity to enforce model consistency, but we leave that as future work.

Let $\{\xi_i\}_{i=0}^{n}$ be a finite-dimensional set of variables with the convention that ξ_0 corresponds to the time variable, ξ_i, $i = 1, \ldots, n$ correspond to the spatial variables used in a model, and n denotes the total number of dimensions. The model (1) can now be written as follows:

$$\rho F \frac{\partial^2 u}{\partial \xi_0^2} + EI_y \frac{\partial^4 u}{\partial \xi_1^4} - \rho I_y \frac{\partial^4 u}{\partial \xi_1^2 \partial \xi_0^2} = q(x, t). \tag{2}$$

Following [4], we introduce the polymorphic type of signals

$$\mathsf{S}\,\alpha = (T_0, T_1, \ldots, T_n) \rightarrow \alpha,$$

where T_i, $i = 0, 1, \ldots, n$ are the types of variables $\{\xi_i\}_{i=0}^{n}$ and α is the type of values "carried" by the signal; i.e., its value at a specific point in time and space. The concrete type α depends on the model under consideration: if we work with scalar-valued quantities, then α is a base type; if we have vector-valued quantities, α is a product type. For example, a two-dimensional heat conduction problem would lead to a temperature field of the form $\theta\colon \mathbb{R}_+ \times \mathbb{R}^2 \rightarrow \mathbb{R}$ providing values of a temperature at each point of two-dimensional body, while solution of plane elasticity problems leads to a vector-valued displacement field of the form $\mathbf{u}\colon \mathbb{R}_+ \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$ providing components of horizontal and vertical displacements of every point of a rigid body.

Signals exist only implicitly (or there would be no need to solve a problem), defined by stated relations among them. Thus, following [4], we introduce the type

$$\mathsf{SR}\,\alpha$$

for a relation on a signal of type $\mathsf{S}\ \alpha$. To describe given models as relations on signals, we need to specify refined types for the basic operations such as differentiation, integration, etc.

Consider the Rayleigh model as formulated in (2). The partial derivatives used in the model need to be typed. One possibility is to introduce an explicit typing for partial derivatives w.r.t. each individual variable. This is not very convenient even in the case of a small number of variables. Instead, we leverage that signals are indexed on $\{\xi_i\}_{i=0}^{n}$. Partial differentiation w.r.t. individuals variables can be then be written as a general operator

$$D_i^n := \frac{\partial^n}{\partial \xi_i^n}, \tag{3}$$

where i is the index of a the variable w.r.t. which to differentiate and n is the order of the derivative. The operator has to satisfy the standard laws:

$$D_i^n D_i^m = D_i^m D_i^n = D_i^{n+m}, \quad D_i^n D_j^m = D_j^m D_i^n, \quad D_i^0 = I, \tag{4}$$

with I denoting the identity operator. These rules have to be reflected in the type system.

The type of the operator D_i^n is

$$D :: (\mathtt{Nat}, \mathtt{Nat}) \rightarrow \mathsf{SR}\,(\alpha, \alpha)$$

where the variable index and the derivative order are restricted to the type `Nat` of natural numbers ($\mathbb{N} = \{0, 1, 2, \ldots\}$) as we are not considering fractional differential equations at present. Note that the result of applying a D operator on a signal is a new signal. The D operators can thus be seen as binary relations on two signals, as reflected by their type.

The next step is to introduce notation for defining relations on signals. Following [4], we adopt notation inspired by λ-abstraction, giving the following notation for a signal relation:

$$\textbf{sigrel} \quad \textit{pattern} \quad \textbf{where} \quad \textit{equations}$$

The *pattern* introduces signal variables that are bound to the value of the corresponding signal at each point in time and space. Thus, for a given signal variable p of a given type t, i.e. $p :: t$, we have

$$\textbf{sigrel} \quad p \quad \textbf{where} \ldots :: \mathsf{SR}\,t$$

Together with the differentiation operator, the notation for defining relations is enough to denote simple partial differential equations. However, for a generalisation of the class of possible applications we need to introduce further general operators, such as the Laplace operator, divergence, gradient and curl. For the case of the Laplace operator we can directly use its factorisation by several operators of type (3), and therefore, it is, in fact, already inside the proposed framework. For the remaining three operators it is necessary to include additional type restrictions, since divergence and curl require vector-valued functions as arguments, while gradient works with scalar-valued arguments. Thus, some modelling errors can be filtered out already by simple type-checking of arguments of these operators.

Finally, to describe models we need to introduce two kinds of equations:

$$e_1 = e_2, \qquad sr \diamond e_3,$$

where e_i, $i = 1, 2, 3$ are expressions that are allowed to introduce new variables and sr is an expression denoting signal relation. As usual, we require equations to be well-typed. Particularly, if $e_i :: t_i$, $i = 1, 2, 3$, then the only possibility is that $t_1 = t_2$ and $sr :: \mathsf{SR}\, t_3$.

The first kind of equation requires the values of the two expressions to be equal at all points in space and time. For the Rayleigh model (2) it means precisely that the expression on the r.h.s. of the equation must have the type resulting from application of several basic operations on the l.h.s. of the equation. Otherwise the model would be inconsistent.

The second kind of equation allows us to work with arbitrary relations on signals. In this case, the symbol $\diamond$ can be understood as a *relation application* resulting in a constraint that must hold at all points in space and time. Moreover, the first kind of equation is just a special case of the second kind, as equality is a subset of general relations between two signals.

The differentiation operator D can now be written as follows

$$D(i)(n) \diamond (f, g), \quad i = 0, 1, \dots, N,$$

with N denoting the total number of variables, and where we assume that signals $f(\xi_i)$ and $g(\xi_i)$ are related by the differential equation

$$f(\xi_i) = \frac{\partial^n g(\xi_i)}{\partial \xi_i^n}.$$

In concrete syntax, we can adapt a notation for the operator D which is closer to the classical mathematical notations

$$\mathrm{D}(i)(n)\, g = f.$$

The meaning of this equation is exactly the same as in the first version.

Finally, we illustrate the proposed language by using it for the example of the Rayleigh model:

$$\begin{array}{lll} \mathit{Rayleigh} & :: & \mathsf{SR}\,(\mathbb{R}, \mathbb{R}) \\ \mathit{Rayleigh} & = & \mathbf{sigrel} \quad (u, q) \quad \mathbf{where} \\ & & \quad q \;\; = \;\; \rho \cdot F \cdot D(0)(2)u + E \cdot Iy \cdot D(1)(4)u \\ & & \qquad\qquad -\rho \cdot Iy \cdot (D(1)(2) \circ D(0)(2))\, u \end{array}$$

where $D(i)(n) \circ D(j)(m)$ denotes the composition of two operators D obeying rules (4), the material constants ρ, F, E, Iy are understood as signals which are constant at all points in space and time and the equality relation represent the principle meaning of any mathematical physics model given by an equation.

For illustrative purposes, let us also show this example expressed directly using signal relations and application, which is the the underlying representation:

$$\begin{array}{lll} \mathit{Rayleigh} & :: & \mathsf{SR}\,(\mathbb{R}, \mathbb{R}) \\ \mathit{Rayleigh} & = & \mathbf{sigrel} \quad (u, q) \quad \mathbf{where} \\ & & \quad q \;\; = \;\; \rho \cdot F \cdot \mathit{d2dt2u} + E \cdot Iy \cdot \mathit{d4dx4u} \\ & & \qquad\qquad -\rho \cdot Iy \cdot \mathit{d4dx2dt2u} \\ & & \quad D(0)(2) \qquad\qquad \diamond \;\; (u, \mathit{d2dt2u}) \\ & & \quad D(1)(4) \qquad\qquad \diamond \;\; (u, \mathit{d4dx4u}) \\ & & \quad D(1)(2) \circ D(0)(2) \;\; \diamond \;\; (u, \mathit{d4dx2dt2u}) \end{array}$$

3 CONCLUSIONS AND FUTURE WORK

Real problems of mathematical physics require an extension of capabilities of the modern modelling languages. Particularly, the growing complexity of engineering problems requires a separation between an abstract modelling phase, where principal relations between different parts of models are considered, and a concrete modelling phase, where implementation issue are addressed. The aim of such a separation is to catch modelling errors as early as possible, particularly in the abstract phase before starting to to think about code derivation and choice of numerical techniques.

In this short work in progress report, taking advantage of the high-level of abstraction provided by FHM, we presented initial ideas for the extension of FHM towards problems of mathematical physics. Particularly, we have shown how general equations of mathematical physics can be expressed in an FHM-like syntax. Additionally, we outlined some aspects of a type system capturing important properties of partial differential equations, identified additional opportunities for consistency checking, and illustrated with a simple example of a model represented by a partial differential equation.

Future work includes further development of the type system for a formalised modelling process in mathematical physics along with Haskell implementation of the proposed constructions. Particularly, a deeper analysis of existing models in mathematical physics, particularly, analysis of a coupling of different models in multiphysics problems, from the point of view of type theory is necessary.

ACKNOWLEDGMENTS

This research is partially supported by the German Research Foundation (DFG) via the Research Training Group "Evaluation of Coupled Numerical Partial Models in Structural Engineering (GRK 1462)". The support offered by DFG is gratefully acknowledged.

REFERENCES

[1] *SIMULINK – User's guide*, MathWorks, Inc., 1992.
[2] *Modelica – a unified object-oriented language for systems modelling; Language specification version 3.3*, Modelica Association, 2012.
[3] H. Nilsson, A. Curtney, J. Peterson, *Functional reactive programming, continued*, Proceedings of the 2002 ACM SIGPLAN Haskell Workshop, pp. 51-64, 2002.
[4] H. Nilsson, J. Peterson, and P. Hudak, *Functional hybrid modeling*. Volume 2562 of Lecture Notes in Computer Science, Springer-Verlag, 2003.
[5] J. Capper, *Semantic methods for Functional Hybrid Modelling*, PhD thesis, 2014.
[6] H. Keitel, G. Karaki, T. Lahmer, S. Nikulla, V. Zabel, *Evaluation of coupled partial models in structural engineering using graph theory and sensitivity analysis*, Engineering Structures, 33, pp. 3726-3736, 2011.
[7] K. Gürlebeck, D. Hofmann, D. Legatiuk, *Categorical approach to modelling and to coupling of models*, Mathematical Methods in the Applied Sciences, Volume 40, Issue 3, pp. 523-534, 2017.
[8] J.J. Roche, *The mathematics of measurements: a critical history*, Springer Science & Business Media, 1998.
[9] A. Kennedy, *Dimension types*, Proceedings of the 5th European Symposium on Programming, 1994.