Domain-Specific Languages of Mathematics
Course codes: DAT326 / DIT982

Patrik Jansson

2023-08-22

Contact	Patrik Jansson, 0729852033.
Results	Announced within 15 workdays
Exam check	(by appointment via email)
Aids	One textbook of your choice (Domain-Specific Languages of Mathematics, or Beta - Mathematics Handbook, or Rudin, or Adams and Essex, or . . .). No printouts, no lecture notes, no notebooks, etc.
Grades	To pass you need **a minimum of 5p on each question (1 to 4)** and also reach these grade limits: 3: >=48p, 4: >=65p, 5: >=83p, max: 100p

Remember to write legibly. Good luck!

For reference: the learning outcomes. Some are tested by the hand-ins, some by the written exam.

- Knowledge and understanding
  - design and implement a DSL (Domain-Specific Language) for a new domain
  - organize areas of mathematics in DSL terms
  - explain main concepts of elementary real and complex analysis, algebra, and linear algebra
- Skills and abilities
  - develop adequate notation for mathematical concepts
  - perform calculational proofs
  - use power series for solving differential equations
  - use Laplace transforms for solving differential equations
- Judgement and approach
  - discuss and compare different software implementations of mathematical concepts

1. [25pts] Algebraic structure: a DSL for semirings.

   A semiring is a set R equipped with two binary operations $+$ and $\cdot$, called addition and multiplication, such that:

   - $(R, +, 0)$ is a commutative monoid with identity element 0:
     $(a + b) + c = a + (b + c)$
     $0 + a = a + 0 = a$
     $a + b = b + a$
   - $(R, \cdot, 1)$ is a monoid with identity element 1:
     $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
     $1 \cdot a = a \cdot 1 = a$
   - Multiplication left- and right-distributes over $(R, +, 0)$:
     $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
     $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
     $a \cdot 0 = 0 \cdot a = 0$

   (a) Define a type class *SemiRing* that corresponds to the semiring structure.

   (b) Define a datatype *SR v* for the language of semiring expressions (with variables of type v) and define a *SemiRing* instance for it. (These are expressions formed from applying the semiring operations to the appropriate number of arguments, e.g., all the left hand sides and right hand sides of the above equations.)

   (c) Find two other instances of the *SemiRing* class.

   (d) Gice a type signature for, and define, a general evaluator for *SR v* expressions on the basis of an assignment function.

   (e) Specialise the evaluator to the two *SemiRing* instances defined in (1c). Take three semiring expressions of type *SR String*, give the appropriate assignments and compute the results of evaluating, in each case, the three expressions.

   Each question carries 5pts.

2. [25p] **Laplace**

   Consider the following differential equation:

   $$2f + 2f' + f'' = 0, \quad f(0) = -2, \quad f'(0) = 2$$

   (a) [10p] Solve the equation assuming that f can be expressed by a power series *fs*, that is, use *integ* and the differential equation to express the relation between *fs*, *fs'*, *fs''*. What are the first four coefficients of *fs*? Explain how you compute them.

   (b) [15p] Solve the equation using the Laplace transform. You should need this formula (note that α can be a complex number) and the rules for linearity + derivative:

   $$\mathcal{L}\,(\lambda t.\, e^{\alpha * t})\, s = 1/(s - \alpha)$$

   Show that your solution does indeed satisfy the three requirements.

3. [20p] **Derivatives and** *FunExp*

Consider the domain of simple functions of one variable. The syntax is given by *FunExp*, the semantic type by S, and the semantics by the homomorphism *eval*:

$$
\begin{array}{l}
\textbf{data } \mathit{FunExp} = C\ \mathbb{R} \mid X \mid \mathit{Add}\ \mathit{FunExp}\ \mathit{FunExp} \mid \mathit{Mul}\ \mathit{FunExp}\ \mathit{FunExp} \\
\textbf{type } F = \mathit{FunExp} \\
\textbf{type } S = \mathbb{R} \to \mathbb{R}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{eval} :: F & \to & S \\
\mathit{eval}\ (C\ c) & = & \mathit{cS}\ c \\
\mathit{eval}\ X & = & \mathit{xS} \\
\mathit{eval}\ (\mathit{Add}\ e_1\ e_2) & = & \mathit{addS}\ (\mathit{eval}\ e_1)\ (\mathit{eval}\ e_2) \\
\mathit{eval}\ (\mathit{Mul}\ e_1\ e_2) & = & \mathit{mulS}\ (\mathit{eval}\ e_1)\ (\mathit{eval}\ e_2)
\end{array}
$$

$$
\begin{array}{lll}
\mathit{cS} & :: \mathbb{R} \to S; & \mathit{cS} = \mathit{const} \\
\mathit{xS} & :: S; & \mathit{xS} = \mathit{id} \\
\mathit{addS} & :: S \to S \to S; & \mathit{addS}\ f\ g = \lambda x \to f\ x + g\ x \\
\mathit{mulS} & :: S \to S \to S; & \mathit{mulS}\ f\ g = \lambda x \to f\ x * g\ x
\end{array}
$$

Your task is to implement another homomorphism: $\mathit{der2} : F \to (F, F)$ which (for all $\mathit{fe} : F$) should satisfy $\mathit{eval}\ (\mathit{fst}\ (\mathit{der2}\ \mathit{fe})) = \mathit{eval}\ \mathit{fe}$ and $\mathit{eval}\ (\mathit{snd}\ (\mathit{der2}\ \mathit{fe})) = D\ (\mathit{eval}\ \mathit{fe})$ where $D : S \to S$ is the derivative operator.

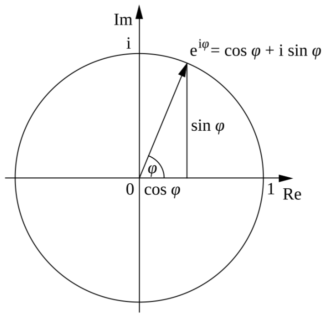
(a) [5p] Implement the top level (recursive) structure in the same style as *eval* so that it is clear that *der2* is also a homomorphism. In this part you can assume that *cder2*, *xder2*, *addder2*, and *mulder2* are available ("wishful thinking").

(b) [5p] Give the types and implementations of *cder2* and *xder2*.

(c) [10p] Give the types and implementations of *addder2* and *mulder2*.

4. [30p] **Euler and the unit circle**

The complex plane is the plane formed by the complex numbers, with a Cartesian coordinate system the real part along the x-axis, and the imaginary part along the y-axis. Euler's Formula, $\mathit{euler}\ \varphi = e^{i\varphi} = \cos\varphi + i * \sin\varphi$, establishes the fundamental relationship between the trigonometric functions and the complex exponential function. The unit circle



$$S^1 = \{a + i * b \mid a^2 + b^2 = 1, a \in \mathbb{R}, b \in \mathbb{R}\}$$

is the range of the function *euler*.

The elements of type S^1 form a *Group*:

$$
\begin{array}{l}
\textbf{instance } \mathit{Multiplicative}\ S^1 \textbf{ where } \mathit{one} = \mathit{oneS}^1; (*) = \mathit{mulS}^1 \\
\textbf{instance } \mathit{MulGroup}\ S^1 \textbf{ where } \mathit{recip} = \mathit{recipS}^1
\end{array}
$$

(a) [3p] Give, and explain, the types of φ, *euler* and i.

(b) [5p] Give the types of oneS^1, mulS^1, recipS^1.

(c) [5p] Implement oneS^1, mulS^1, recipS^1 in Haskell.

(d) [5p] State the property that a function $h : A \to B$ is a group homomorphism from $(A, \mathit{op}_A, \mathit{inv}_A, \mathit{unit}_A)$ to $(B, \mathit{op}_B, \mathit{inv}_B, \mathit{unit}_B)$ in first order logic.

(e) [12p] State and prove that *euler* is a group homomorphism. You may use the trigonometric "sum of angles" identities. **Note:** The learning outcome tested here is mainly "perform calculational proofs", thus you are expected to use equational reasoning and motivate your steps.