Domain-Specific Languages of Mathematics
Course codes: DAT326 / DIT983

Patrik Jansson

2025-03-21

Contact	Felix Cherubini, 072 252 1143, (Patrik Jansson, 072 985 2033).
Results	Announced within 15 workdays
Exam check	Tue 2025-04-08, 12.15-12.45 in EDIT 6128
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. [25p] **Algebraic structure: a DSL for vector spaces**

   A vector space over $\mathbb{R}$ is a set V together with a constant (or nullary) operation $0 : V$ (called "zero"), an operation $(+) : V \to V \to V$ (called "add"), an operation $(-) : V \to V$ (called "negate") and an operation $(\cdot) : \mathbb{R} \to V \to V$ (called "scale"), such that

   $$\begin{array}{lll}
   \forall\, v \in V. & v + 0 = 0 + v = v & \text{-- additive zero} \\
   \forall\, v_1, v_2, v_3 \in V. & (v_1 + v_2) + v_3 = v_1 + (v_2 + v_3) & \text{-- associative } (+) \\
   \forall\, v \in V. & v + (-v) = (-v) + v = 0 & \text{-- additive inverse} \\
   \forall\, v_1, v_2 \in V. & v_1 + v_2 = v_2 + v_1 & \text{-- commutative } (+) \\
   \forall\, x_1, x_2 \in \mathbb{R}, v \in V. & x_1 \cdot (x_2 \cdot v) = (x_1 * x_2) \cdot v & \text{-- repeated scaling} \\
   \forall\, v \in V. & 1 \cdot v = v & \text{-- unit scaling} \\
   \forall\, x \in \mathbb{R}, v_1, v_2 \in V. & x \cdot (v_1 + v_2) = x \cdot v_1 + x \cdot v_2 & \text{-- scaling left-distributive} \\
   \forall\, x_1, x_2 \in \mathbb{R}, v \in V. & (x_1 + x_2) \cdot v = x_1 \cdot v + x_2 \cdot v & \text{-- scaling right-distributive}
   \end{array}$$

   Remark: $(*)$ denotes the standard multiplication in $\mathbb{R}$.

   (a) Define a type class *Vector* that corresponds to the structure "vector space over $\mathbb{R}$".

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

   (c) Find and implement two other instances of the *Vector* class. Make sure the laws are satisfied.

   (d) Give a type signature for, and define, a general evaluator (on the basis of an assignment function) from *VecSyn a* expressions to any semantic type with a *Vector* instance.

   (e) Specialise the evaluator to the two *Vector* instances defined in (1c). Take three vector expressions (of type *VecSyn 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:

   $$f'' + 20f' + 99f = -99, \quad f(0) = 0, \quad f'(0) = -1$$

   (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 three 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. [25p] **Typing:** Lattice subgroups

Consider the following (rephrased) quote from Wikipedia:

> In geometry and group theory, a lattice in the real coordinate space $\mathbb{R}^n$ is a set of points L in this space with the properties that coordinate-wise addition or subtraction of two points in the lattice produces another lattice point, that there is a minimum distance d_1 such that any two lattice points a, b have at least distance d_1, and that every point x in $\mathbb{R}^n$ is within some maximum distance d_2 of a lattice point.

(a) [5p] Give types for L, d_1, a, b, d_2. Explain your reasoning and be careful to interpret "distance" and "two lattice points" in a way that makes the definition non-trivial.

(b) [5p] Define (in first-order logic) the predicate isLattice(U,n) which is true if and only if the set U is a lattice in $\mathbb{R}^n$. You may assume a function $d : \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}$ computing the distance of two points in $\mathbb{R}^n$.

(c) [5p] Prove $\neg$isLattice($\{(x, 0) \in \mathbb{R}^2 | x \in \mathbb{Z}\}, 2$).

(d) [5p] Let $a > 0$ be a positive real number. Prove isLattice($\{(x, y) \in \mathbb{R}^2 | ax, ay \in \mathbb{Z}\}, 2$).

(e) [5p] Find an example of a lattice in $\mathbb{R}^2$ which is not of the form given above. Explain your reasoning.

4. [25p] **Calculational proof:** Syntactic derivatives

Consider the following Haskell code for a DSL of 1-argument functions. The evaluator *eval* and the function der_2 are structural homomorphisms from the syntax type F. Your task is to implement some of the missing parts of the "deep copy and derivative" function der_2, and prove some properties about it.

```
data F = Zero | One | X | Sub F F | Mul F F | Div F F deriving (Eq, Show)
eval :: Field a => F -> (a -> a)          -- Equality labels below (for the proof)
eval Zero  _ = zero                       -- eval Zero
eval One   _ = one                        -- eval One
eval X     x = x                          -- eval X
eval (Sub fe ge) x = eval fe x - eval ge x   -- eval Sub
eval (Mul fe ge) x = eval fe x * eval ge x   -- eval Mul
eval (Div fe ge) x = eval fe x / eval ge x   -- eval Div
der2 :: F -> (F, F)
der2 Zero        = der2Zero                          -- der2 Zero
der2 One         = der2One                           -- der2 One
der2 X           = der2X                             -- der2 X
der2 (Sub fe ge) = der2Sub (der2 fe) (der2 ge)       -- der2 Sub
der2 (Mul fe ge) = der2Mul (der2 fe) (der2 ge)       -- der2 Mul
der2 (Div fe ge) = der2Div (der2 fe) (der2 ge)       -- der2 Div
```

Let the property $P(fe)$ be "Let fe_2 be the first, and fe' the second, component of the pair returned by $der_2\ fe$. Then $fe_2 == fe$ and $eval\ fe' == D\ (eval\ fe)$, where D is the differentiation operator." The specification of der_2 is then $\forall\ fe.\ \ P(fe)$.

(a) [5p] Give types for, and implement, *der2One*, *der2X*, and *der2Sub*.

(b) [5p] In an induction proof of correctness of der_2, one of the base cases is $P(X)$ (where X is a constructor in the datatype F). Prove this case using equational reasoning, carefully motivating each step as in "by def. *eval One*", "by def. $der_2\ X$", etc.

(c) [5p] Give the type for, and implement, *der2Div*.

(d) [10p] One of the inductive step cases is $\forall\ ge.\ \ \forall\ he.\ \ P(ge) \land P(he) \Rightarrow P(Div\ ge\ he)$. Prove this step using equational reasoning.