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

Patrik Jansson

2025-08-26

| | |
|---|---|
| **Contact** | Patrik Jansson, 072 985 2033. |
| **Results** | Announced within 15 working days |
| **Exam check (granskning)** | Thu 2025-09-04, 12.15-12.45, EDIT-6452 |
| **Permitted 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. |
| **Grading** | 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: lattice**

   A **lattice** is a set $L$ together with two operations $\vee$ and $\wedge$ (usually pronounced "sup" for supremum and "inf" for infimum) such that

   - $\vee$ and $\wedge$ are associative:
     $$\forall\ x, y, z \in L.\ \ (x \vee y) \vee z = x \vee (y \vee z)$$
     $$\forall\ x, y, z \in L.\ \ (x \wedge y) \wedge z = x \wedge (y \wedge z)$$
   - $\vee$ and $\wedge$ are commutative:
     $$\forall\ x, y \in L.\ \ x \vee y = y \vee x$$
     $$\forall\ x, y \in L.\ \ x \wedge y = y \wedge x$$
   - $\vee$ and $\wedge$ satisfy the **absorption laws**:
     $$\forall\ x, y \in L.\ \ x \vee (x \wedge y) = x$$
     $$\forall\ x, y \in L.\ \ x \wedge (x \vee y) = x$$

   (a) Define a type class *Lattice* that corresponds to the lattice structure.

   (b) Define a datatype *Lat v* for the language of lattice expressions. These are expressions built from variables of type $v$ using the lattice operations. Define a *Lattice* instance for the expression type.

   (c) Find and implement two other instances of the *Lattice* class. Ensure that your instances satisfy the lattice laws.

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

   (e) Specialise the evaluator to the two *Lattice* instances defined in (c). Take three lattice expressions of type *Lat 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:

$$f''(t) - 3\sqrt{2} * f'(t) + 4 * f(t) = 0, \quad f(0) = 2, \quad f'(0) = 3\sqrt{2}$$

(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 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 $\alpha$ 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 satisfies the differential equation (all three equalities).

3. [20p] **Typing maths:** Action

Consider the following slightly edited quote from Wikipedia Action (physics):

> **Action**
> Most commonly, the term 'action' is used for a functional $\mathcal{S}$ which takes a function of time as input and returns a scalar. In classical mechanics, the input function is the evolution $\mathbf{q}(t)$ of the system between two times $t_1$ and $t_2$, where $\mathbf{q}$ represents the generalized coordinates. The action $\mathcal{S}[\mathbf{q}(t)]$ is defined as the integral of the Lagrangian $L$ for an input evolution between the two times:
> $\mathcal{S}[\mathbf{q}(t)] = \int_{t_1}^{t_2} L[\mathbf{q}(t), \dot{\mathbf{q}}(t), t]\ dt$

To be more specific, let's state that

$\mathbf{q}(t) = (\omega * t, 1)$
$L[(\alpha, r), (v_\alpha, v_r), t] = A * (v_r^2 + (r * v_\alpha)^2) - B\ /\ r$

where $\omega$, $A$, $B$ are real-valued constants.

(a) [8p] Give types for the symbols $\mathcal{S}$, $\mathbf{q}$, $\dot{\mathbf{q}}$, $L$, $t$.

(b) [7p] Suggest other notation and restate the definition of the action $\mathcal{S}$. Be careful to explain where variables are bound and replace the dot (for time derivative) with suitable uses of $D : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$. You may use Haskell-style notation for functions and types.

(c) [5p] Consider the following quote from the same page:

> **Minimization of action integral**
> [...] Classical mechanics postulates that the path actually followed by a physical system is that for which the action is minimized [...]

Write a formal definition (in first order logic) of a predicate *MinAction* (*path*) that captures the quote.

4. [30p] **Polynomial function composition**

Consider the following code for polynomials represented as lists of coefficients:

$$
\begin{array}{l}
\textbf{type } P\ a = [a] \\
\textbf{type } S\ a = a \to a \\
\\
eval :: Num\ a \Rightarrow P\ a \to S\ a \\
eval\ [\,]\ \ \ \ \ \ x = 0 \\
eval\ (a : as)\ x = a + x * eval\ as\ x \\
\\
scaleP :: Num\ a \Rightarrow a \to P\ a \to P\ a \\
scaleP\ a = map\ (a*) \\
\\
addP :: Num\ a \Rightarrow P\ a \to P\ a \to P\ a \\
addP\ [\,]\ \ \ \ \ \ bs\ \ \ \ \ \ = bs \\
addP\ as\ \ \ \ \ \ [\,]\ \ \ \ \ \ = as \\
addP\ (a : as)\ (b : bs) = (a + b) : addP\ as\ bs \\
\\
mulP :: Num\ a \Rightarrow P\ a \to P\ a \to P\ a \\
mulP\ [\,]\ \ \ \ \ \ bs\ \ \ \ \ \ = [\,] \\
mulP\ as\ \ \ \ \ \ [\,]\ \ \ \ \ \ = [\,] \\
mulP\ (a : as)\ (b : bs) = (a * b) : addP\ (scaleP\ a\ bs)\ (mulP\ as\ (b : bs)) \\
\\
comP :: Num\ a \Rightarrow P\ a \to P\ a \to P\ a \\
comP\ [\,]\ \ \ \ \ \ bs = error\ \texttt{"TODO: your task 1"} \\
comP\ (a : as)\ [\,] = error\ \texttt{"TODO: your task 2"} \\
comP\ (a : as)\ bs = error\ \texttt{"TODO: your task 3"}
\end{array}
$$

As you can see, scaling, addition and multiplication of polynomials is already given. Your task is related to the operation *comP* which should implement function composition of polynomials represented as lists of coefficients. It is specified by $H_2(eval, comP, (\circ))$ where the predicate $H_2\cdot$ is defined by:

$$H_2(h, Op, op) = \forall\ x.\ \ \forall\ y.\ \ h\ (Op\ x\ y) == op\ (h\ x)\ (h\ y)$$

or, more concretely: $H_2(eval, comP, (\circ)) = \forall\ p.\ \ \forall\ q.\ \ eval\ (comP\ p\ q) == eval\ p \circ eval\ q$. **Important:** see the note below about equational reasoning.

(a) [6p] As a warm-up, compute the special cases $cs_1 = comP\ p\ q$ and $cs_2 = comP\ q\ p$ where $eval\ p\ x = 1 + x$ and $eval\ q\ x = x\hat{}2 - 1$. (Thus $p = [1, 1]$ and $q = [-1, 0, 1]$.)

(b) [4p] Use the specification to calculate task 1 above.

(c) [4p] Use the specification to calculate task 2 above.

(d) [7p] Use the specification to calculate $c = comP\ [a_0, a_1]\ [b_0, b_1]$

(e) [9p] Use the specification to calculate the general case (task 3 above). You may use $H_2(eval, addP, (+))$, $H_2(eval, mulP, (*))$, and the spec. of *comP* for shorter lists.

**Important:** The learning outcome tested here is mainly "perform calculational proofs", thus you are expected to use equational reasoning and motivate your steps. Here is an example of equational reasoning showing that $eval\ [a_0, a_1]\ x == a_0 + x * a_1$:

$$
\begin{array}{ll}
 & eval\ (a_0 : a_1 : [\,])\ x \\
= & \quad \text{-- def. of } eval\ (a : as) \\
 & a_0 + x * eval\ (a_1 : [\,])\ x \\
= & \quad \text{-- def. of } eval\ (a : as) \\
 & a_0 + x * (a_1 + x * eval\ [\,]\ x) \\
= & \quad \text{-- def. of } eval\ [\,] \\
 & a_0 + x * (a_1 + x * 0) \\
= & \quad \text{-- simplify} \\
 & a_0 + x * a_1
\end{array}
$$