Domain Specific Languages of Mathematics
Course codes: DAT326 / DIT982

Patrik Jansson

2020-03-17

Contact	Patrik Jansson (x5415), Sólrún Einarsdóttir, Víctor López Juan
Results	Announced within 19 days
Exam check	2020-03-30 in EDIT 5468 at 12.10-12.40
Aids	"all examination aids are allowed" Special covid-19 rules
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 Quasigroups

   Consider the following mathematical definition (adapted from Quasigroup, Wikipedia):

   > A quasigroup $(Q, (\otimes), (//), (\backslash\backslash))$ is a set Q equipped with three binary operators (called multiplication, right- and left-division) satisfying the following identities for all x and y in Q:
   >
   > $y = (y \otimes x) // x$
   > $y = (y // x) \otimes x$
   > $y = x \backslash\backslash (x \otimes y)$
   > $y = x \otimes (x \backslash\backslash y)$

   (a) Define a type class *Quasi* that corresponds to the Quasigroup structure.

   (b) Define a datatype $Q\ v$ for the language of quasigroup expressions (with variables of type v) and define a *Quasi* instance for it. (These are expressions formed from applying the quasigroup 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 *Quasigroup* class. Make sure the laws are satisfied.

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

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

2. [20p] **Typing maths:** Action

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

> **Action**
> Most commonly, the term 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 $\mathcal{S}\left[\mathbf{q}(t)\right]$ is defined as the integral of the Lagrangian L for an input evolution between the two times:
> $\mathcal{S}\left[\mathbf{q}(t)\right] = \int_{t_1}^{t_2} L\left[\mathbf{q}(t), \dot{\mathbf{q}}(t), t\right] dt$

To be more specific, let's state that

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

where ω, 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 $D : (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$.

(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 of a predicate *MinAction* (*path*) that captures the quote.

3. [30p] **Polynomial function composition**

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

```
type P a = [a]
type S a = a -> a

eval :: Num a => P a -> S a
eval []       x = 0
eval (a : as) x = a + x * eval as x

scaleP :: Num a => a -> P a -> P a
scaleP a = map (a*)

addP :: Num a => P a -> P a -> P a
addP []       bs       = bs
addP as       []       = as
addP (a : as) (b : bs) = (a + b) : addP as bs

mulP :: Num a => P a -> P a -> P a
mulP []       bs       = []
mulP as       []       = []
mulP (a : as) (b : bs) = (a * b) : addP (scaleP a bs) (mulP as (b : bs))

comP :: Num a => P a -> P a -> P a
comP []       bs = error "TODO: your task 1"
comP (a : as) [] = error "TODO: your task 2"
comP (a : as) bs = error "TODO: your task 3"
```

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 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.

Note: 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 $eval\ [a_0, a_1]\ x == a_0 + x * a_1$:

```
  eval (a0 : a1 : []) x
=    -- def. of eval (a : as)
  a0 + x * eval (a1 : []) x
=    -- def. of eval (a : as)
  a0 + x * (a1 + x * eval [] x)
=    -- def. of eval []
  a0 + x * (a1 + x * 0)
=    -- simplify
  a0 + x * a1
```

4. [25p] **Laplace**

Consider the following coupled differential equations:

$$f'(x) = C * g(x), \quad g'(x) = f(x), \quad f(0) = C, \quad g(0) = 0$$

where $C = (n+1)\hat{}2$ and n is the last digit of your personal identity number. (For example, if 19890102-3286 would sit the exam, n would be 6 and $C = 49$.)

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

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

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

Show that your solutions do indeed satisfy the four requirements.