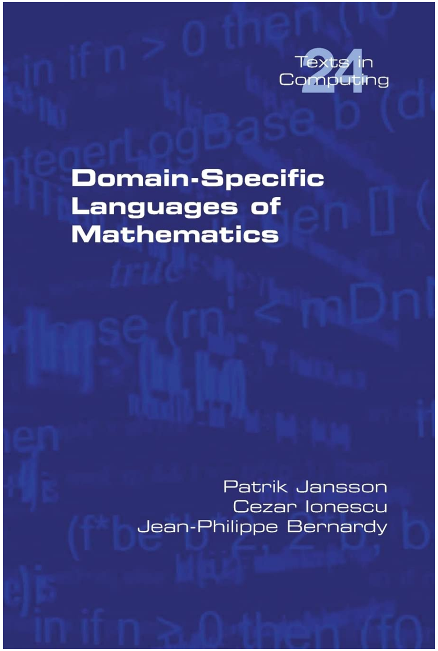
# Domain-Specific Languages of Mathematics

Patrik Jansson

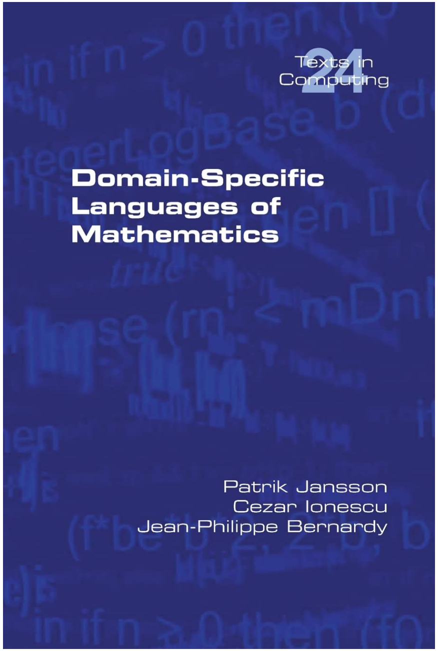
Functional Programming unit, Chalmers University of Technology

2023-08-30

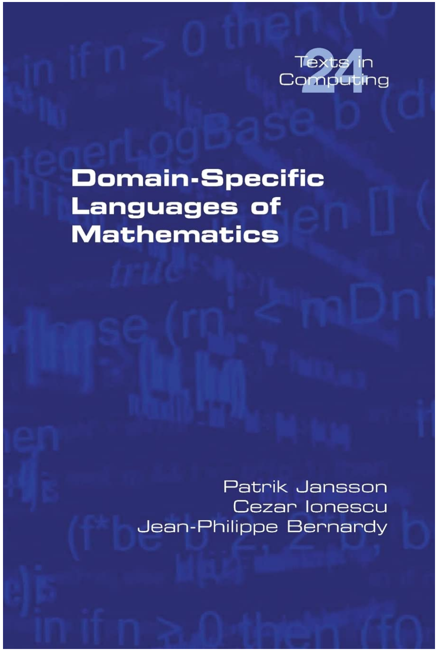
- Computer scientist, Haskell hacker, catalyst of research ideas, likes to connect the big picture with formal details, software & language technology advocate.
- Have worked on proofs in Agda, Climate Impact Research, Parametricity for Dependent Types, Testing, Parsing, polytypic programming, and **Domain-Specific Languages of Mathematics**



- The book (on the left) is course literature for a 7.5hec course at Chalmers and UGOT.
- The main idea behind the course is to encourage the students to approach mathematical domains from a functional programming perspective.



- The book (on the left) is course literature for a 7.5hec course at Chalmers and UGOT.
- The main idea behind the course is to encourage the students to approach mathematical domains from a functional programming perspective.
- Students will learn about the language Haskell; identify the main functions and types involved; introduce calculational proofs; pay attention to the syntax of mathematical expressions; and, finally, to organize the resulting functions and types in domain-specific languages.



- The book (on the left) is course literature for a 7.5hec course at Chalmers and UGOT.
- The main idea behind the course is to encourage the students to approach mathematical domains from a functional programming perspective.
- Students will learn about the language Haskell; identify the main functions and types involved; introduce calculational proofs; pay attention to the syntax of mathematical expressions; and, finally, to organize the resulting functions and types in domain-specific languages.
- The minicourse today and tomorrow gives a teaser: some functional programming in Haskell and a few examples of lectures from the course.

## What is a Domain-Specific Language (DSL)

A DSL has four components:

- Surface syntax (mostly ignored in this course): a set of strings defined by a grammar
- Abstract syntax: usually a recursive Haskell datatype of syntax trees (*AbsSyn*)
- Semantic type: a type *Sem* of values (meanings) for the syntax
- Semantics: a function from $\mathit{eval} : \mathit{AbsSyn} \to \mathit{Sem}$

Examples:

- date expressions (“2023-08-30”, “last wed. in Aug.”, “today”, ...)
- excel-sheet formula (“SUM(A1:A9)”, “RIGHT(LEFT(C7,4),2)”, ...)
- first-order logic ($P \Rightarrow Q$, $\forall\, \epsilon > 0.\ \ \exists\, \delta > 0.\ \ \epsilon > \delta$, ...)

## Mini-course Overview

- Haskell — a short introduction to “Type-Driven Development (TDD)”
  - Typed functional programming
  - Function types, Polymorphism, function composition
  - Pair types, TDD, . . .
  - Sum types, *Either*, *Maybe*, . . .

## Mini-course Overview

- Haskell — a short introduction to “Type-Driven Development (TDD)”
  - Typed functional programming
  - Function types, Polymorphism, function composition
  - Pair types, TDD, . . .
  - Sum types, *Either*, *Maybe*, . . .
- Text-book examples:
  - Complex numbers (from text to code, semantics, syntax trees)
  - Scoping, limit of function, derivative in terms of limits, typing
  - Lagrangian: give type and formal predicate from a text-book quote

## Mini-course Overview

- Haskell — a short introduction to “Type-Driven Development (TDD)”
  - Typed functional programming
  - Function types, Polymorphism, function composition
  - Pair types, TDD, . . .
  - Sum types, *Either*, *Maybe*, . . .
- Text-book examples:
  - Complex numbers (from text to code, semantics, syntax trees)
  - Scoping, limit of function, derivative in terms of limits, typing
  - Lagrangian: give type and formal predicate from a text-book quote
- Interaction welcome: ask questions at any time and I will try to explain.

## Haskell introduction

Live coding

`HaskellIntro.lhs`