Projects Summer Term 2024

Supervisors: Alexander Dinges, Cass Alexandru, Michael Youssef, Sebastian Schloßer

Kick-off: 26.04.2024

Organization

Schedule

- Kick-off: Friday, 26.04.2024, 13:45 - 15:15, room 36-265
- Final presentations: 26.07.2024, 13:45 - 20:00, room 36-265
  Mandatory attendance!
- Regular meetings with your supervisor

Project Teams

- Five topics to choose from
- Two teams per topic (working independently)
- Four or five students per team (collaborating)

$\rightarrow$ Capacity of 50 students

Agenda

- We present you the topics
- You (physically) go to the supervisor whose topic you would like to work on
- There you form teams and exchange details with team members & supervisor

Projects on ExClaim

Supervisor: Sebastian Schloßer

Projects on ExClaim

Background

About ExClaim

EXCLAIM

- Exercise management
  - Students
  - Groups (same exercise session)
  - Teams (collaborate on homework)
  - Homework submission
  - Homework grading
  - Automatic group assignment
- Exam management
- Automatic test execution

Remote Test Executor (RTE)

- Docker
- Consistent environment
- No risks for tutors
- Fast feedback (for students and as guidance while grading)
- Can execute test cases without providing the source code to students

History (1)

- Softech Achievement Tracking System (STATS)
  - Manage students, groups, teams, exams
  - No submission (on paper or via e-mail)
  - No automatic group assignment
- Optimus: Automatic group assignment
- ExClaim & RTE: For submission & testing

Three separate systems!

- Users management only in stats
- Linking data via student id

History (2)

- Integrate all features in one system
- Optimize database layout
- Use code-generation for type-safe database access
- Simplify building the system

Projects on ExClaim

First Project: New Frontend

New Frontend

- Old: Thymeleaf template engine
  - Backend produces rendered HTML pages
  - Interaction: Javascript code snippets interacting with page
- New: Vue Framework
  - Single Page Application
  - Page rendered by browser (Javascript engine)
  - TypeScript source code
  - Communication with backend via type-safe JSON objects

Current status

- Co-existence of both systems
- Can move one page after another
- Overall setup completed
  - Building must be simple!
  - Easy deployment (GitLab CI)
  - Type-safety: database $\leftrightarrow$ backend $\leftrightarrow$ frontend
- Some pages (Login, Homepage) already moved

Your Task

- Move remaining pages to new frontend
- Thereby design a nice user interface

Technologies to Learn / Know

- Git
- Backend
  - Java
  - Spring Boot
  - jOOQ (Code Generator for type-safe database access)
- Frontend
  - TypeScript / Javascript
  - Vue
  - HTML, CSS

Projects on ExClaim

Second Project: Online IDE

Motivation

- Initial project setup can be complicated
  - Overwhelming for first-year students
  - Support effort during first few weeks

Vision

- Readily setup IDE in the browser
  - No local installation or configuration required
  - Configured with all required/recommended plugins
- No template downloading $\rightarrow$ directly start editing
- No upload required $\rightarrow$ directly work on copy on the server

Your Task

- Try out `code-server`[1] (Visual Studio Code in the browser)
- Find a way to integrate it into ExClaim
- Rework how ExClaim stores uploaded files
  $\rightarrow$ Use Git?

[1] `https://github.com/coder/code-server`

Technologies to Learn / Know

- Git
- `code-server`
  - Linux as operating system!
  - Javascript / Node.js
  - Probably Docker
- ExClaim backend
  - Java
  - Spring Boot

Project: Search tool for Agda

Supervisor: Alexander Dinges

Your task

- CLI search tool for an Agda repository
- Find function signatures/propositions/types together with their names given an (incomplete) part of the type
- Cope with variables
- Reasonably fast
- Good ranking
- Readable code

Technologies to learn/know

- Git
- A tiny bit of Agda
- You can use (almost) any programming language you like

Developing a web app using Haskell

Supervisor: Michael Youssef

Why Haskell?

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

- Haskell is a pure language
- No unintended side effects
- ADTs
- Modularity
- Laziness
- Referential transparency

Why Haskell?

R P TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

- Haskell is a pure language
- No unintended side effects
- ADTs
- Modularity
- Laziness
- Referential transparency

- No null-pointer exceptions
- No IO exceptions
- Elegant approach towards handling IO
- No unhandled cases

Challenges/What you will learn?

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

- Type classes
- Practical usages of functors, applicatives and monads
- Monad transformers
- Type families
- Template Haskell
- Debugging with lazy evaluation

Challenges/What you will learn?

R P TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

- Type classes
- Practical usages of functors, applicatives and monads
- Monad transformers
- Type families
- Template Haskell
- Debugging with lazy evaluation
- Network IO
- Database storage
- Caching
- Writing a fully fledged application in Haskell

Why should you care?

R P TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

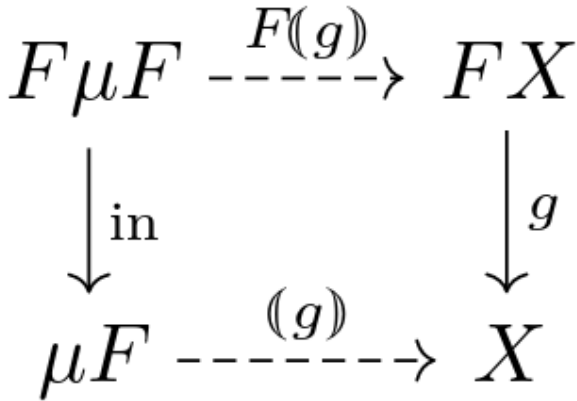
- ECTS….
- Learn some practical applications of the stuff you learned in FP
- Knowledge you gain is transferable to other programming language paradigms

Editor für 3D kommutierende Diagramme

Supervisor: Cass Alexandru

Kontext

- kommutierende Diagramme sind ein Standard Beweiswerkzeug in der Kategorientheorie
- Beispieldiagramm:



Bestehende Tools

- https://q.uiver.app/
- https://tikzcd.yichuanshen.de/

Problem Statement

- Oft sind Diagramme eigentlich in 3D zu Hause!

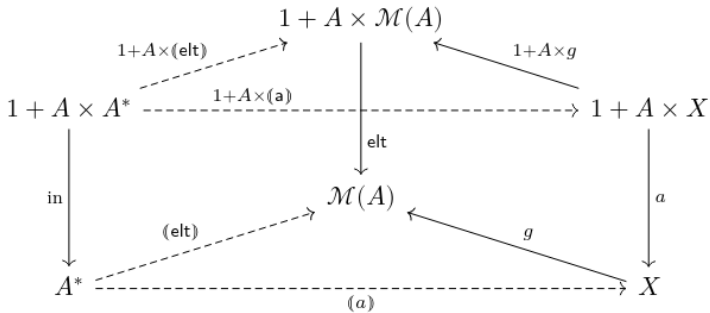

$1 + A \times \mathcal{M}(A)$
$1+A\times(\mathsf{elt})$
$1+A\times g$
$1 + A \times A^*$
$1+A\times(\mathsf{a})$
$1 + A \times X$
elt
in
$\mathcal{M}(A)$
a
(elt)
g
A^*
X
(a)

Figure 1: pie

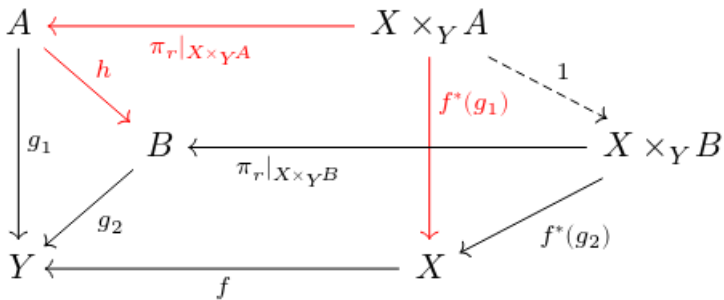
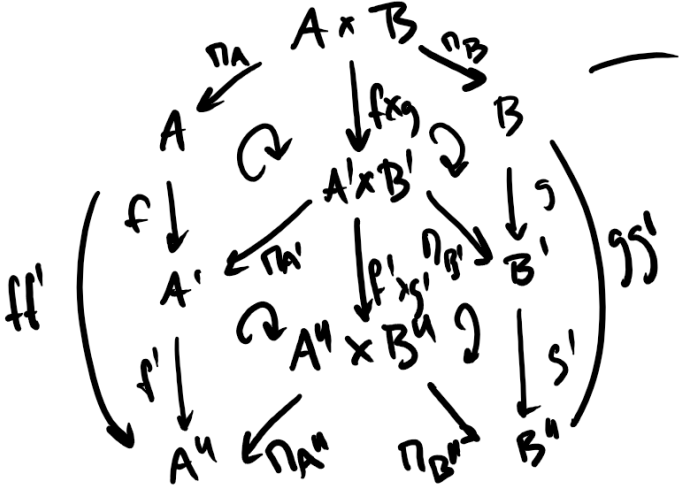
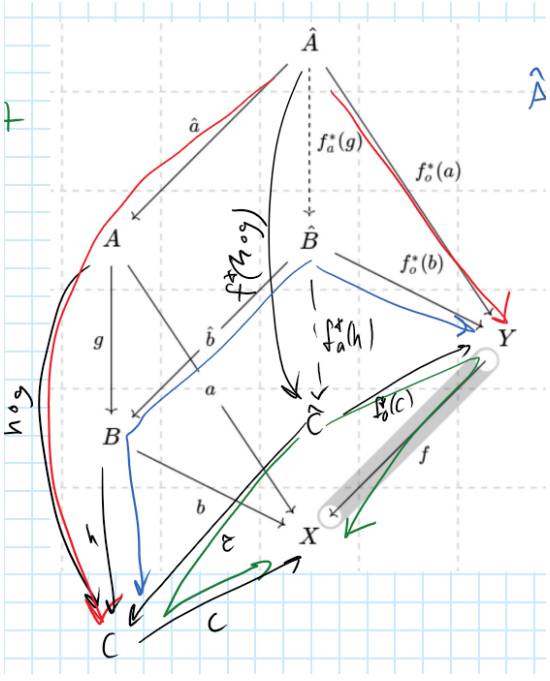


Figure 2: triangular prism


A × B
πA
πB
A
B
f×g
A'×B'
f
g
A'
B'
πA'
πB'
ff'
gg'
f'×g'
A''×B''
f'
g'
A''
B''
πA''
πB''


$\hat{A}$
$\hat{a}$
$f_a^*(g)$
$f_o^*(a)$
A
$\hat{B}$
$f_o^*(b)$
g
$\hat{b}$
Y
a
B
C
f
b
X
C
C

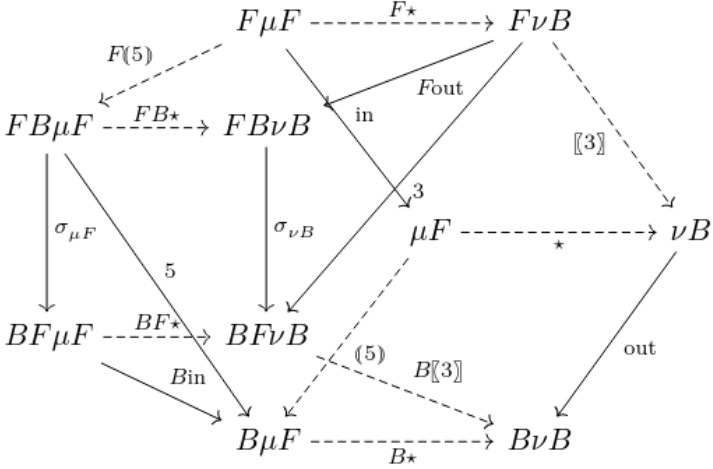


Figure 5: star

MVP (Minimum Viable Product)

- Objekte
- Pfeile (am liebsten verschiedene Arten – gestrichen, doppelt), mit TeX Pfeilspitzen
- Pfeilbeschriftung
- Labels mit LaTeX typesetting (oder Imitate wie KaTeX, MathJax)
- click-and-drag umpositionierung der Objekte
- Verschiedene 3d Objekte mit Vertizes, an denen nodes plaziert werden können
  - Minimum: Vorauswahl an Standard Objekten: Prismata mit konfigurierbarer Anzahl Ebenen
- Sowohl Labels als auch Pfeile verhalten sich als 2D sprites die immer zur Kamera gedreht bleiben
- Export von verschiedenen 2D-Projektionen des Diagramms
- Export als Vektorgrafik
- möglicherweise als Plug-in für bestehende 3D-Software (blender – hat auch svg Renderer)

Nice-to-haves

- Web app
- client side
- Wechsel zwischen canvas view (Bearbeitung des Objekts auf dem das Diagramm “lebt”) und diagram view
- TikZ / pgf code export
- Pfeile zwischen Pfeilen (2-cells)
- Animierte Folge von Diagrammen / gefärbten Pfaden in einem Diagramm (svg Animation)