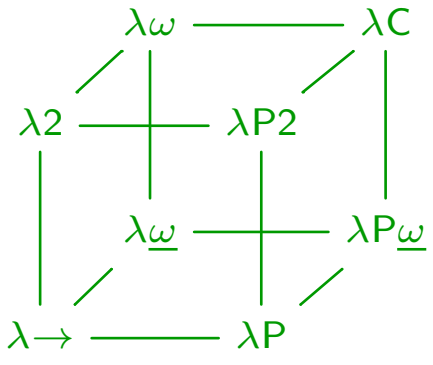
introduction & lambda calculus

Freek Wiedijk

Type Theory & Coq
2024–2025
Radboud University Nijmegen

September 6, 2024



organization

coordinates

https://www.cs.ru.nl/~freek/courses/tt-2024/

+

Brightspace

teachers:

- Freek Wiedijk
  freek@cs.ru.nl
- Herman Geuvers
  herman@cs.ru.nl
- Niels van der Weide
  n.vanderweide@cs.ru.nl
- Robbert Krebbers
  robbert@cs.ru.nl

teaching assistant:

- Luko van der Maas
  luko.vandermaas2@ru.nl

structure of the course

first half:

- five lectures on the type theory of Coq, by Freek (Fridays)
- three lectures on metatheory, by Herman (Fridays)
- Coq practicum (Thursdays)
  $\longrightarrow$ required, not graded
- two hour written exam
  $\longrightarrow$ one third of the final grade

second half:

- student presentations (Mondays & Fridays)
  45 minutes, in pairs
  $\longrightarrow$ one third of the final grade
- Coq project
  $\longrightarrow$ one third of the final grade

materials

- Femke van Raamsdonk, VU Amsterdam
  Logical Verification Course Notes, 2008
  - course notes
  - slides
  - Coq practicum files
- Herman Geuvers
  Introduction to Type Theory, 2008
  - summer school lecture notes
  - slides
  - some exercises
- reading list papers
- some supporting documents
  - Jules Jacobs: Coq cheat sheet
  - examples of induction/recursion principles
- many old exams, all with answers

prerequisites

course is self-contained, but...

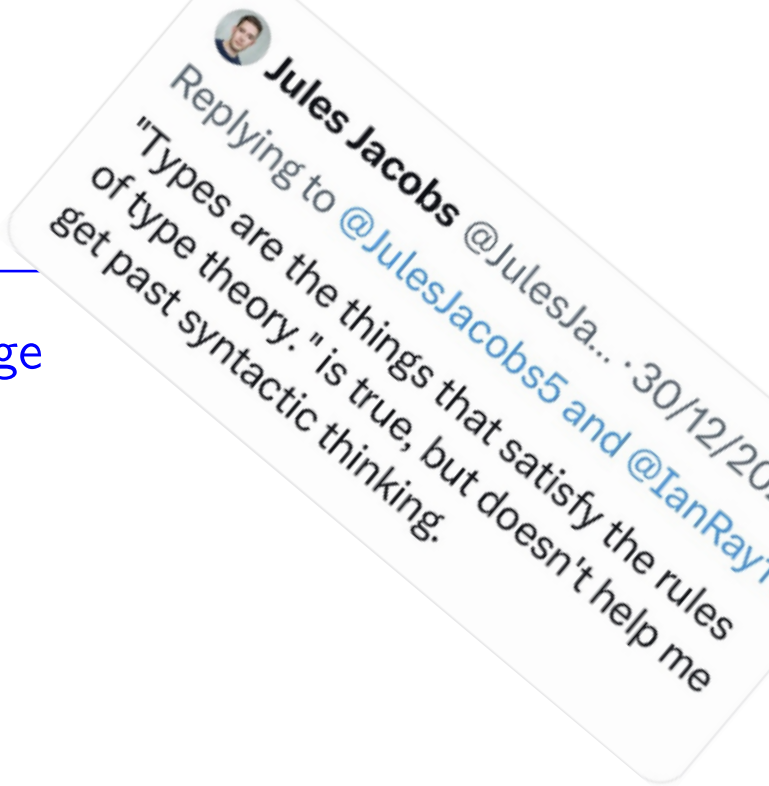
we will presuppose some basic familiarity with:

- context-free grammars
  NWI-IPC002 Languages and Automata
- mathematical logic: natural deduction
  NWI-IPI004 Logic and Applications
- functional programming
  NWI-IBC040 Functional Programming
- lambda calculus
  NWI-IBC025 Semantics and Rewriting

as well as some mathematical maturity

introduction

what is a type?



- an attribute of expressions in a language

```
int i;
float pi = 3.14;
i = 2 * pi;
```

- something like a set

$$\text{int} = \{-2^{31}, -2^{31}+1, \ldots, -1, 0, 1, \ldots 2^{31}-1\}$$
$$\text{nat} = \{0, 1, 2, 3, \ldots\}$$

but: types do not overlap
the 0 of nat is different from the 0 of int

also: an object has a type
a type has a kind
... but there it stops