

Dependent Types and Theorem Proving: Proving is programming in disguise

Wojciech Kołowski

March 2021

Plan of lectures

- Lecture 1: Programming with dependent types.
- **Lecture 2: Proving theorems with dependent types.**
- Lecture 3: Differences between programming and proving.
- Lecture 4: Examples of bigger programs and longer proofs.
- Lecture 5: A deeper dive into F^* .

- 1 Introduction: boolean logic and classical logic
- 2 Constructive propositional logic: you already know it
 - Propositional logic
 - Propositions are types, proofs are programs
 - Function types are implications
 - Sum is disjunction
 - Product is conjunction
 - Unit is True
 - Falsity and negation
- 3 Higher-order logic: you already know it
 - Predicates and relations
 - Universal quantifier is the dependent function type
 - Existential quantifier is the dependent pair type
- 4 Induction is recursion
- 5 Inductive predicates and relations
 - Undecidability and generative thinking
 - Proof relevance
- 6 Equality
 - Definition and convertibility
 - Proposition of equality

The booleans and their logic

- Being a programmer, you are good friends with the booleans, aren't you?
- There are two booleans, `true` and `false`.
- We can combine booleans `b` and `c` with the usual boolean functions:
- `not b` – negation, pronounced “not `b`”
- `b && c` – conjunction, pronounced “`b` and `c`”
- `b || c` – disjunction, pronounced “`b` or `c`”
- We can also define less commonly used boolean functions:
- `b ==> c = not b || c` – implication, pronounced “if `b` then `c`”
- `b <=> c = b ==> c && c ==> b` – logical equivalence, pronounced “`b` if and only if `c`”

Classical logic



