

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

Boolean “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`”

What is a logic

- Boolean logic is not an example of what logicians call a “logic”, in the sense that it is not a “logical system”, but merely a type with some unary and binary functions on it.
- A logic usually consists of:
- A definition of what **propositions** we’re dealing with.
- A **semantics**, which tells us what these propositions mean.
- A **proof system**, which tells us which propositions can be proven and disproven.
- A **soundness theorem** which states that propositions proven true using the proof system are semantically true.
- Optionally, there may also be a **completeness theorem** which states that all semantically true propositions can be proven.

Propositions

- Propositions (ϕ, ψ) are defined as follows:
- \top – the true proposition.
- \perp – the false proposition.
- P, Q, R, \dots – propositional variables.
- $\neg P$ – negation, read “not P ”.
- $P \vee Q$ – disjunction, read “ P or Q ”.
- $P \wedge Q$ – conjunction, read “ P and Q ”.
- $P \implies Q$ – implication, read “ P implies Q ”.
- $P \iff Q$ – logical equivalence, read “ P if and only if Q ”.

Classical logic

- Classical logic is the most widely known/taught/used logical system in the world.
- We think of propositions as being either true or false.
- Therefore, classical logic is the logic in which **the truth values are booleans**.
- The truth value of propositional variables is determined by a **valuation** $v : \text{Var} \rightarrow \text{Bool}$.
- If $v(P) = \text{true}$, then P is considered to be true.
- Otherwise it's considered false.

Semantics of classical logic

- Given a valuation $v : \text{Var} \rightarrow \text{Bool}$, the truth value of a proposition can be determined with a recursive function $\llbracket - \rrbracket : \text{Prop} \rightarrow \text{Bool}$:
- $\llbracket \top \rrbracket = \text{true}$
- $\llbracket \perp \rrbracket = \text{false}$
- $\llbracket P \rrbracket = v(P)$, where P is a variable.
- $\llbracket \neg \phi \rrbracket = \text{not } \llbracket \phi \rrbracket$
- $\llbracket \phi \vee \psi \rrbracket = \llbracket \phi \rrbracket \text{ || } \llbracket \psi \rrbracket$
- $\llbracket \phi \wedge \psi \rrbracket = \llbracket \phi \rrbracket \text{ \&\& } \llbracket \psi \rrbracket$
- $\llbracket \phi \implies \psi \rrbracket = (\text{not } \llbracket \phi \rrbracket) \text{ || } \llbracket \psi \rrbracket$
- $\llbracket \phi \iff \psi \rrbracket = \llbracket \phi \rrbracket \text{ == } \llbracket \psi \rrbracket$

Constructive logic

- Constructive logic is a logical system different from classical logic.
- It will serve us to prove correctness of our programs.
- We **DO NOT** think of propositions as being either true or false.
- Instead, we think in terms of **truth certificates**. If we have a certificate of truth for the proposition ψ , then ψ is true. Otherwise, we don't know anything about ψ .

