Dependent Types and Theorem Proving: Proving is programming in disguise

Wojciech Kołowski

May 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*.

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` – "not b"
- `b && c` – "b and c"
- `b || c` – "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

- A proposition asserts that something is the case, irrespectively of whether this really is the case or not.
- Math example: "4 is a prime number."
- Software example: "For each input string x, if x is not malformed, my program produces as output an array of length at most 10."
- Hardware example: "This circuit implements addition of 16 bit integers."
- Real world example: "It's raining or I like trains."
- **Beware! Formal logic is not very good for reasoning about the real world!**

Propositional constants and connectives

- Propositions (usual letters: P, Q, R) are defined as follows:
- $\top$ – the true proposition.
- $\perp$ – the false proposition.
- $P, Q, R, \ldots$ – 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" or "if P then 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.
- In classical logic, **we think of propositions as being either true or false.**
- Therefore, classical logic is the logic in which truth values are the booleans.
- The truth value of a propositional variable is determined by a **valuation** $v : \mathrm{Var} \to \mathrm{Bool}$.
- If $v(P) = \mathtt{true}$, then P is considered to be true.
- Otherwise it's considered false.

Semantics of classical logic 1/2

- Given a valuation $v : \mathrm{Var} \to \mathrm{Bool}$, the truth value of a proposition can be determined with a recursive function $[\![-]\!] : \mathrm{Prop} \to \mathrm{Bool}$.
- $[\![\top]\!] = \texttt{true}$
- $[\![\bot]\!] = \texttt{false}$
- $[\![P]\!] = v(P)$, where P is a variable.
- $[\![\neg P]\!] = \texttt{not}\ [\![P]\!]$
- $[\![P \vee Q]\!] = [\![P]\!]\ \texttt{||}\ [\![Q]\!]$
- $[\![P \wedge Q]\!] = [\![P]\!]\ \texttt{\&\&}\ [\![Q]\!]$
- $[\![P \implies Q]\!] = (\texttt{not}\ [\![P]\!])\ \texttt{||}\ [\![Q]\!]$
- $[\![P \iff Q]\!] = [\![P]\!]\ \texttt{==}\ [\![Q]\!]$

Semantics of classical logic 2/2

- P is satisfiable when $[\![P]\!] = \texttt{true}$ for some valuation.
- P is falsifiable when $[\![P]\!] = \texttt{false}$ for some valuation.
- P is a tautology when $[\![P]\!] = \texttt{true}$ for all valuations.

Example

- Example: the proposition $P \implies Q$ is satisfiable (for $v(P) = \texttt{true}, v(Q) = \texttt{true}$).
- It is also falsifiable (for $v(P) = \texttt{true}, v(Q) = \texttt{false}$).
- Therefore, it is not a tautology.
- Example: the proposition $P \wedge Q \implies Q \wedge P$ is a tautology.
- We have $[\![P \wedge Q \implies Q \wedge P]\!] =$ $(\texttt{not}\ (v(P)\ \texttt{\&\&}\ v(Q)))\ \texttt{||}\ (v(P)\ \texttt{\&\&}\ v(Q))$.
- For any values of $v(P)$ and $v(Q)$ we always get $\texttt{true}$.

The rest

- We can check whether a proposition is a tautology by trying all possible valuations, but there are exponentially many of them.
- We can do better by defining a proof system with some axioms and inference rules, which would allow us to **prove** that a proposition is a tautology without trying all valuations.
- We won't do that because **classical logic is not the right logical system for proving programs correct**.
- We will use constructive logic instead.

Constructive logic 1/2

- In constructive logic, **propositions ARE NOT either true or false**.
- In constructive logic we usually think about propositions **in terms of their proofs.**
- In everyday language and also in mathematics as it is usually practiced, a "proof" means an argument by which one human demonstrates the truth of a statement to another human.
- In constructive logic, a proof is a formal object which **certifies that the given proposition has been proven**, in which case we say that the propositions holds.
- Meaning of propositions is determined by how we can prove them and how we can use their proofs to prove other propositions.

Constructive logic

- If we have a proof of P, we may think of it as "true" (although we shouldn't think in terms of true and false).
- If we have a proof of $\neg P$, we may think that P is "false".
- If we have neither proof, we don't know anything about P.

Propositions are types, proofs are programs

- If t is a proof of A, which we write as $t : A$, then we consider the proposition A to be true.
- Otherwise, we don't know anything about A.