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 of P is a certificate that P holds, i.e. a formal object which **certifies that P has been proven.**,
- Meaning of propositions is determined by how we can prove them and how we can use them to prove other propositions.

Constructive logic 2/2

- You shouldn't think about propositions as being either "true" or "false", but if you can't help yourself, then:
- If we have a proof of P, we may think of it as "true".
- 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.

Types vs propositions

- There's a strange parallel going on between propositions and types.
- Types are, obviously, not either true or false – they are inhabited by programs.
- A program t of type A is something that, after performing some computations, returns an element of type A.
- The meaning of a type A is determined by how we can write programs of type A and how we can use programs of type A to write other programs.

Propositions are types, proofs are programs

- This “strange parallel” is not a coincidence. There are no coincidences in mathematics!
- It is most often referred to as the Curry-Howard correspondence, after two out of many people who discovered it.
- But it is better presented as a set of slogans:
- **Propositions are types.**
- **Proofs are programs.**
-