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

- We shouldn't think about propositions as being either "true" or "false", but it's a deeply ingrained and hard to avoid way of thinking, so a translation:
- If we have a proof of P, we may think that P is "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.

Propositions vs types

- 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.**
- **Proving theorems is just writing programs.**
- ... and a few more, which we'll see shortly.

True is the unit type 1/2

- There's the unit type `unit`.
- It's sole element is `()`.
- We can't do anything useful with it.

True is the unit type 2/2

- There's the true proposition $\top$.
- It's sole proof is ().
- We can't conclude anything useful from it.

Conjunction is the product type 1/2

- If `a` and `b` are types, then `a * b` is also a type.
- Elements of `a * b` are pairs `(x, y)`, where `x : a` and `y : b`.
- If we have a pair `x : a * b`, then `fst x : a` and `snd x : b`.

Conjunction is the product type 2/2

- If P and Q are propositions, then $P \wedge Q$ is also a proposition.
- To prove $P \wedge Q$, we have to prove P and we have to prove Q, so...
- ...proofs of $P \wedge Q$ are of the form `(x, y)`, i.e. they are pairs where `x` is a proof of P and `y` is a proof of Q.
- If $P \wedge Q$ holds, then we can conclude that P holds and we can conclude that Q holds, so...
- ...if `x` is a proof of $P \wedge Q$, then `fst x` is a proof of P and `snd x` is a proof of Q.

Implication is the function type 1/2

- If `a` and `b` are types, then `a -> b` is also a type.
- Elements of `a -> b` are of the form `fun (x : a) -> e` – they are functions which take an input `x` of type `a` and return `e` of type `b` as output.
- If we have a function `f : a -> b` and an `x : a`, then we we can apply `f` to `x`, written `f x`, to get an element of type `b`.

Implication is the function type 2/2

- If P and Q are propositions, then $P \implies Q$ is also a proposition.
- To prove $P \implies Q$, we need to assume that P holds and then provve Q under this assumption, so...
- ... proofs of $P \implies Q$ are of the form `fun (p : P) -> q`, i.e. they are functions which take a proof of P as input and return a proof of Q as output.
- If $P \implies Q$ holds and P holds, we can conclude that Q holds, so...
- ... if `f` is a proof of $P \implies Q$ and `x` is a proof of P, then `f x` is a proof of Q.

Disjunction is discriminated union

- If P and Q are propositions, then $P \vee Q$ is also a proposition.
- To prove $P \vee Q$, we need either to prove P or to prove Q, so...
- ...proofs of $P \vee Q$ are of the form `inl p`, where `p` is a proof of P, or of the form `inr q`, where `q` is a proof of Q.
- If $P \vee Q$ holds and $P \implies R$ holds and $Q \implies R$ holds, we can conclude that R holds, so...
- ...if `x` is a proof of $P \vee Q$, then we can match on `x` and retrieve the proofs of P/Q and use them to prove R.