Lab 11: First-order logic with types

Functional Programming (ITI0212)

This week we saw how to encode first-order logic in Lean, including logical implication, conjunction, disjunction, negation, and existential and universal quantifiers.

Task 1

Write proofs of the following propositions:

- example : (P \land Q) \leftrightarrow (Q \land P) (conjunction is commutative)
- example : (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P) (contraposition).
- example (p : $\alpha \rightarrow$ Prop) : (\exists x, p x) $\rightarrow \neg$ (\forall x, \neg p x)

Hint: to access the components of an existential proposition h, you can write

let \langle a, b \rangle := h in your function, or you can write have instead of let.

Task 2

In classical logic, if a proposition is not false, then it is true, that is, $\neg \neg P \rightarrow P$ is true for all propositions P. This corresponds to the principle of "proof by contradiction" whereby if we assume $\neg P$ and reach a contradiction, we are allowed to conclude P.

This does not hold for a general proposition in Lean, because the logic of Lean is constructive by default. Thus to prove P we really need to construct an element of P and there is no general way of doing this from an element of $\neg \neg$ P.

This is a bit like how, in everyday life, "it's not *not* going to rain" doesn't mean it is going to rain, or "he's not *lying*..." doesn't imply that the truth was told.

On the other hand, $\neg \neg P \rightarrow P$ is not *false* in Lean, because for *some* propositions it does hold. Prove that,

- $\neg\neg$ False \rightarrow False,
- $\neg\neg$ True \rightarrow True.
- $\neg\neg$ isEven 1 \rightarrow isEven 1

Hint: prove \neg is Even 1 first, and give the proof a name (using def or theorem instead of example), so that you can re-use it.

Task 3

In classical logic, for every proposition P, the proposition $P \lor \neg P$ is true. That is, for every proposition P, either P or its negation holds. This is called the *law of the* excluded middle or tertium non datur.

Again, in the constructive logic of Lean, this law does not hold in general. $P \lor \neg P$ is true in Lean just when we can construct a proof of P or a proof of $\neg P$. But some propositions are open questions, for which we cannot (yet) provide either! Constructive logic eschews the Platonic realm in which every proposition has a definite truth value even when that value is not known to mortals (or the computer).

Prove that if excluded middle holds then double negation elimination holds. That is, prove $(P \lor \neg P) \rightarrow (\neg \neg P \rightarrow P)$.