Tutorial 9 – Predicates and relations

SUMMER SEMESTER 2026 FLORENT SCHAFFHAUSER

No handing-in for this one 🌴 .

Note: For the exercises below, I recommend writing the solution on paper first, as this will provide some training for the exam. It is still a good idea to use Rocq pseudo-code to do so, though! Then you can check your answers using Rocq 😅 .

EXERCISE 1. After reviewing the various definitions of even natural numbers that we saw on the lectures, solve the following problems.

a. Introduce an inductive predicate `IsOdd : nat → Prop`, characterising odd natural numbers.

b. Formalise the statement "*the sum of two odd numbers is even*".

c. Prove the previous statement.

d. Formalise the statement that "*no natural number is both even and odd*".

e. How would you prove the previous statement? Describe the structure of the proof, skipping intermediate results if necessary.

EXERCISE 2. Let `X` be a set and let `P, Q : X → Prop` be predicates on `X`. Recall that such predicates define subsets of `X` and that we can introduce the notation `x ∈ P` to denote the proposition `P x`.

a. What would be the formal definition of the predicate `P ∩ Q`, representing the intersection of the subsets `P` and `Q`?
Hint: Think of it as finding the right-hand side of the proposition `x ∈ P ∩ Q ↔ ?`.

b. Similarly, what would be the definition of the predicate `P ∪ Q`? What subset of `X` does it represent?

EXERCISE 3. Let `X` be a set and let `≤` be a pre-order on `X`.

a. What does it mean for a relation to be a pre-order and how is this formalised in a language such as Rocq?

b. Show that the relation `x ~ y := x ≤ y ∧ y ≤ x` is an equivalence relation.

Exercise 4. Let `≤ : nat → nat → Prop` be the (so-called *carrier* of the) pre-order on `nat` introduced in the lectures. Namely, `≤` is the relation defined inductively by the constructors:

- for all `n : nat`, `0 ≤ n`.
- for all `n m : nat`, the condition `n ≤ m → S n ≤ S m`.

a. Recall the formal definition of `≤` in Rocq.

b. Show that, for all `n m : nat`, the proposition `n ≤ m ∨ m ≤ n` holds (feel free to admit intermediary results if this helps).