3.1 Introduction

Motivation

- Every person drinks water.
- Amir is a person.
- Therefore, Amir drinks water.

In propositional logic, the best we can do to express this is:

- A = Every person drinks water.
- B = Amir is a person.
- C = Amir drinks water.

The top-level proposition is $A \wedge B \rightarrow C$.
To prove this proposition, we need information about A, B, and C.
We need to be able to reason about:

- objects
- groups of objects
- attributes of objects
- relationships between objects

Motivation

We need to be able to reason about:

• **individual objects**	Waterloo, Toronto
• **groups of objects**	cities, vehicles
• **attributes/properties** of objects	"Waterloo is boring"
• **relations** between objects	"Waterloo is west of Toronto"
• **all objects**	"All cities are boring"
• **some object(s)**	"Some city is exciting"

Predicate logic adds: **types**, **functions** and **quantifiers** to capture these concepts.

> **Note:** A predicate is a function whose return type is proposition.

Think of typed predicate logic as a programming language for reasoning about objects.

3.1.1 Types

Do the following both make sense?

- $drinks(Bill, tea)$
- $drinks(Vancouver, helium)$

Programming languages use types to distinguish between different kinds of objects, which helps catch errors.
We can do the same thing in logic.

Rules for Types

1. Each symbol, object, function, *etc.* has exactly one type within an individual formula or proof.
   - In one formula, 1 may be an integer and in another formula, 1 may be a natural number.
   - Within a single formula, 1 **may not** be both a natural number an integer.
   - If you need both naturals and integers (or reals and integers) in a single proof, use predicates for the smaller types.
2. Each type must be non-empty.
   - This is a technical requirement that is needed so that things we think should be true are in fact true in the logic.

     For example:

     - If P is true for all elements in type Ty, then there is an element of type Ty for which P is true.
     - $(\forall(x : Ty),\ P(x)) \rightarrow (\exists(x : Ty),\ P(x))$
     - If Ty could be empty, then this statement is false.

3.1.3 Predicates

A **predicate** describes:

- an attribute (property) of an object
- a relationship between two or more objects

Example 2.2:

English	Logic
"Bill" is a person	$Bill : person$
"child" is a predicate about a person	$child : person \rightarrow Prop$
"Bill is a child"	$child\ Bill$
"apples" are a food	$apples : food$
"eats" is a predicate about a person and a food	$eats : person \rightarrow food \rightarrow Prop$
"Bill eats apples"	$eats\ Bill\ apples$

Quantifiers

A quantifier is always followed by a variable with a type and then a predicate

Name	Symbol	Description
Universal quantification	$\forall (x : Ty),\ P\ x$	$P\ x$ is true for all objects of type Ty ("every", "all", "for all")
Existential quantification	$\exists (x : Ty),\ P\ x$	$P\ x$ is true for at least one object of type Ty ("some", "there exists")

The formula: $\exists (x : Ty),\ P\ x$ is read as:
"There exists x of type Ty, such that $P\ x$".

Bound and Free Variables

- A variable is **bound** to the closest quantifier to the left of its name whose scope it is within. All variable occurrences that are bound to the same quantifier represent the same object.
- An occurrence of a variable is bound if it falls within the scope of a quantifier for that variable.
- An occurrence of a variable is **free** if it does not fall within the scope of a quantifier for that variable.
- A wff is **closed** if it contains no free variables.
- We will be working with closed wff.
- Whether a variable is free or bound depends on the occurrence of the variable in the formula, not on its name. The same variable name can be free or bound.

VALUE OF FREE VAR COMES FROM ENVIRONMENT, JUST LIKE IN PROPOSITIONAL LOGIC.