# **Section 7.1 First Order Predicate Calculus**

A proposition is a statement that is either true or false. You can abbreviate statements to make WFFs. For example, “*Let a mean ‘Kermit the frog is green’*.”

Meanwhile, a predicate is a function or method that “returns” true or false (boolean). For example, “*Let g(x) mean ‘x is green’.*” By itself, it doesn’t have a truth value, so you can create a proposition by using a constant (specific things in the world).

* Let k = Kermit the frog
* g(k) is a proposition
* a = g(k)

**More Predicate Examples**

* Let g(x) mean “x is green”
* Let f(x, y) mean “x + y == 4”
* Let p(x) mean “x is prime”
* Let b(x) mean “x has a banner ID”

Let *i* = 3, *j* = 6, and *k* = Kermit the frog. You can write: g(k) ∧ ¬ b(k) ∧ p(i) ∧ ¬ p(j) ∧ ¬ f(i, j)

Predicates contain a variable (as seen in the parentheses of the bullet points), which are argument(s) or parameter(s) of a predicate that can take on a lot of different values.

**Domain**

A domain is like the universe of the problem. For example,

* Domain: Rowan students
* Predicate: b(x) means x has a banner ID
* Statement: For all x in our domain, b(x) [aka, all Rowan students have a banner ID]

You can make similar statements for “it is not the case” and “there exists at least one” x.

**Notation**

“For all” = ∀ “It is not the case” = ¬ ∀ “There exists” = ∃ “Not the same” = n “x==y” = eq(x,y)

## **Examples**

For example, let b be the “has brown hair” predicate. The statement, “*at least one person has brown hair*”, could be represented by ∃z b(z), where the domain (z) are people.

With the same predicate, if you want to say the statement “*more than one person has brown hair*”, it could be represented as ∃x ∃y (b(x) ∧ b(y) ∧ n(x,y)), which can translate this as there exists a person x and a person y that both have brown hair and persons x and y are not the same person.

With the same domain, let g be the “has green hair” predicate. The statement “*no one has green hair”* can be written two ways: ¬ (∃y g(y)) or ∀x(¬ g(x)).

Let p(x,y) be the “x is the parent of y” predicate. The statement “*everyone has a parent*” can be written as ∀x ∃y (p(y,x)), which translates to, for all people (x), there exists a parent (y) where y is the parent of x.

For the statement “*everyone has only one parent*”, include the same clause as before and add another one + an implication. ∀x ∃y (p(y,x)) ∧ ∀x((p(y, x) ∧ (z,x) ⟶ eq(z, y).

With the same domain and predicate, the statement “*not everyone is a parent”* can be written as “It is not the case that all people (z), there is somebody that exists, where z is the parent of x”, or ¬ ∀x ∃x(p(z, x)) and “There exists a person (a) such that for all other people b, a is NOT the parent of b”, or ∃a ∀b(¬ p(a, b)).

Or, the statement “*some people are parents*” can be written as

∃x ∃y ∃c ∃q(p(x, c) ∧ p(y, q) ∧ ¬ eq(x, y). This means that there exists x, y, c, and q where x is the parent of c AND y is the parent of q AND x and y are NOT the same person.

Or, the statement “*no one is their own parent*” can be written as ¬ ∃x(p(x, x)), or there does not exist an x where x is the parent of x. Another option is ∀x(¬ p(x, x)), or for all x, there is no such case that x is the parent of x.

Or, “*there is no one person who is everyone’s parent*” can be written as ¬ ∃y, ∀x(p(y, x)), meaning that there does not exist y such that for all x, y is the parent of x.

When no domain is specified, you can define a domain. For example, if you were given the statement, “*not all birds can fly*”, you can say that, “Let b(x) be x is a bird and let f(x) be x can fly”, so you can explain, ∃x(b(x) ∧ ¬ f(x)), which means that there exists an x where x is a bird and x cannot fly.

“*Some birds can’t fly*”, so you could use the same WFF as before: ∃x(b(x) ∧ ¬ f(x)). However, the problem specifies “birds”, so you have to do multiple birds:

∃x, ∃y (b(x) ∧ b(y) ∧ eq(x, y) ∧ ¬ f(x) ∧ ¬ f(y)).

Another no specified domain, but the predicates and constants are the following:



The statement “*all days are sunny*” could be ∀x(d(x) ⟶ s(x)). (If it’s a day, it’s sunny.)

*Some days are not rainy* = there’s at least one day that isn’t rainy = ∃x(d(x) ∧ ¬ r(x)).

*Every day that is not sunny is not rainy.* ∀x(d(x) ∧ ¬ s(x) ⟶ ¬ r(x)).

*Some days are sunny and rainy* = ∃x(d(q) ∧ s(x) ∧ r(q)).

*No days are sunny and rainy =* ¬∃x(d(q) ∧ s(x) ∧ r(q)).

*It rained both Monday and Tuesday =* r(m) ∧ r(t).

*Monday was sunny; therefore, every day will be sunny.* = s(m) ⟶ ∀x(s(x)).

**You don’t need to know the meaning to work with predicates.**

Let x, y, z be variables.

Let a, b, c be constants.

Let p, q, r be predicate letters.

Problem: Suppose I tell you that p(a) ∧ p(b) ∧ p(c). What can you conclude if x ∈ {a, b, c}?

Notice that the question part of the problem is specifying the domain!

If you know the predicate is true for a, b, and c and the domain involves a, b, and c, then you can conclude that everything in your domain, the predicate is true. Thus, you can conclude that ∀x(p(x)) and ∃x(p(x)).

Problem: How about x ∈ {a, b, c, d, e, f}?

Well, you can say that at least one of them is true with ∃x(p(x)) and not all domains are true for the predicate: ¬ (∀x(p(x))).

Problem: Suppose I tell you that p(a) ∨ p(b) ∨ p(c). What can you conclude if x ∈ {a, b, c}?

Since this is an or, you can conclude that one or more of the domain keeps the predicate true. Thus, ∃x(p(x)).

Problem: Suppose I tell you that p(a) ∨ (p(b) ∨ p(c)). What can you conclude if x ∈ {a, b, c}?

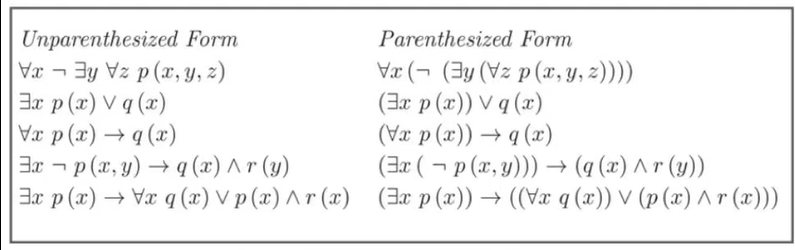
You can still conclude that there is at least one that is true: ∃x(p(x)).

Problem: Suppose I tell you that p(a, b) ∧ p(b, b) ∧ p(c, b). What can you conclude if x ∈ {a, b, c}?

Firstly, you can say ∀x(p(x, b)). This is because no matter what domain you choose, it will always be true with b. It’s true for anything in the domain. Thus, you can also say ∃x(p(x, b)).

**In the textbook, they save space by saying,** *“Whenever you see an a, b, or c, you can assume that they are constants.”*

## **Rules of Precedence**

**

Same as propositional logic, plus if any quantifiers or the negation symbol appear next to each other, then the rightmost symbol is grouped with the smallest WFF to its right.

The **scope** of ∃x in (∃x W) is W. Similarly, the scope of (∀x W) is W. In the absence of parentheses, the scope of a quantifier is the smallest WFF immediately to its right.

Examples:

