Nested Quantifiers Essentials

A student-focused guide to understanding and applying nested quantifiers in logical statements and proofs. This cheatsheet emphasizes clear interpretation, truth evaluation strategies, and common pitfalls to help solidify your grasp of predicate logic.

Foundations of Quantifiers

QUANTIFIER SYMBOLS & BASICS

`∀` (Universal Quantifier)	**Meaning**: "For all", "For every", "For any". **Truth Condition**: True if the predicate holds for *every* element in the domain. **Example**: `∀x P(x)` - "For every x, P(x) is true." **Tip**: To prove `∀x P(x)`, you must show `P(x)` is true for *every single* `x`. To disprove, find just *one* counterexample.
`∃` (Existential Quantifier)	**Meaning**: "There exists", "For some", "At least one". **Truth Condition**: True if the predicate holds for *at least one* element in the domain. **Example**: `∃x P(x)` - "There exists an x such that P(x) is true." **Tip**: To prove `∃x P(x)`, you only need to find *one* example. To disprove, you must show `P(x)` is false for *every* `x`.
Domain of Discourse	The set of all possible values for the variables in a quantified statement. **Crucial for truth evaluation**. **Example**: `∀x (x > 0)` • If domain is integers `Z`, it's false (e.g., x=-1). • If domain is positive integers `Z+`, it's true.
Free vs. Bound Variables	`Bound`: A variable is bound if it falls within the scope of a quantifier (e.g., `x` in `∀x P(x)`). `Free`: A variable is free if it is not bound by any quantifier. **Example**: In `P(x, y) ∧ ∃y Q(y, z)`, `x` and `z` are free, while the first `y` is free and the second `y` is bound.
COMMON MISTAKE	Forgetting the domain of discourse. The truth value of a quantified statement *always* depends on its domain. **Strategy Tip**: Always state your domain of discourse clearly before evaluating or constructing proofs.
SCOPE OF QUANTIFIERS	A quantifier's scope extends to the smallest subformula that immediately follows it. Parentheses `()` are used to explicitly define scope. **Example**: `∀x (P(x) → Q(x))` means "For all x, if P(x) then Q(x)." `∀x P(x) → Q(x)` means "(For all x, P(x)) implies Q(x)." (Q(x) is outside the scope of ∀x and x here is a free variable in Q(x)).

NESTED QUANTIFIER STRUCTURES

Order Matters! The sequence of quantifiers drastically changes the meaning of a statement, especially when mixing `∀` and `∃`.
`∀x ∃y P(x, y)`
Plain English: "For every x, there exists a y such that P(x, y) is true."   **Interpretation**: For each `x` you pick, you can find *some* `y` (which might depend on `x`) that makes `P(x, y)` true. Each `x` gets its own `y`.   **Real-world Example**: Let `P(x, y)` be "x loves y". Domain is people. "Everybody loves somebody." (Each person has *someone* they love, but not necessarily the same person for everyone).
`∃y ∀x P(x, y)`
Plain English: "There exists a y such that for every x, P(x, y) is true."   **Interpretation**: There is a *single*, specific `y` that makes `P(x, y)` true for *all* `x`. This `y` is independent of `x`.   **Real-world Example**: Let `P(x, y)` be "x loves y". "There is somebody whom everyone loves." (One specific person is loved by *everyone*).
`∀x ∀y P(x, y)` (or `∀y ∀x P(x, y)`)
Plain English: "For all x and for all y, P(x, y) is true."   **Interpretation**: The predicate `P(x, y)` holds for *every possible pair* of `x` and `y` from the domain.   **Order**: Order does **not** matter for two universal quantifiers.
`∃x ∃y P(x, y)` (or `∃y ∃x P(x, y)`)
Plain English: "There exists an x and there exists a y such that P(x, y) is true."   **Interpretation**: There is *at least one pair* (`x`, `y`) for which `P(x, y)` holds.   **Order**: Order does **not** matter for two existential quantifiers.
STRATEGY TIP: Dependency
Think of nested quantifiers as a game: • If `∀` comes first, the `y` you pick can depend on the `x` chosen by your opponent.   • If `∃` comes first, *you* pick a `y` first, and it must work for *all* `x` chosen by your opponent later.

Evaluating & Negating Quantifiers

TRUTH VALUE ANALYSIS

To determine the truth value of a nested quantified statement over a finite domain, systematically check all possibilities.

Example: `∀x ∃y (x + y = 0)` over domain `D = {-1, 0, 1}`.

1. **For x = -1**: Is there a `y` in `D` such that `-1 + y = 0`? Yes, `y = 1`. (True for x=-1)
2. **For x = 0**: Is there a `y` in `D` such that `0 + y = 0`? Yes, `y = 0`. (True for x=0)
3. **For x = 1**: Is there a `y` in `D` such that `1 + y = 0`? Yes, `y = -1`. (True for x=1)

Since for *every* `x`, we found a `y`, the statement `∀x ∃y (x + y = 0)` is **True**.

Example: `∃y ∀x (x + y = 0)` over domain `D = {-1, 0, 1}`.

1. **Try y = -1**: Is it true for *all* `x` that `x + (-1) = 0`? No, if `x = 1`, `1 + (-1) = 0` (True), but if `x = 0`, `0 + (-1) ≠ 0` (False). So, `y = -1` doesn't work for all `x`.
2. **Try y = 0**: Is it true for *all* `x` that `x + 0 = 0`? No, if `x = 1`, `1 + 0 ≠ 0` (False). So, `y = 0` doesn't work for all `x`.
3. **Try y = 1**: Is it true for *all* `x` that `x + 1 = 0`? No, if `x = 0`, `0 + 1 ≠ 0` (False). So, `y = 1` doesn't work for all `x`.

Since no single `y` worked for *all* `x`, the statement `∃y ∀x (x + y = 0)` is **False**.

STRATEGY TIP: Verification vs. Counterexample

- To verify `∀x P(x)`, you need a proof that works for all `x`.
- To verify `∃x P(x)`, you need one concrete example.
- To disprove `∀x P(x)`, you need one counterexample.
- To disprove `∃x P(x)`, you need a proof that works for all `x` showing `P(x)` is always false.

NEGATING NESTED QUANTIFIERS

De Morgan's Laws for Quantifiers	`¬∀x P(x)` is logically equivalent to `∃x ¬P(x)` `¬∃x P(x)` is logically equivalent to `∀x ¬P(x)`
Rule for Negation	When negating a quantified statement, move the negation `¬` inward past each quantifier, flipping the quantifier type (`∀` to `∃`, `∃` to `∀`) as you pass it. Finally, negate the predicate.
Original Statement	Negated Statement
`¬∀x (P(x))`	`∃x (¬P(x))`
`¬∃x (P(x))`	`∀x (¬P(x))`
`¬∀x ∃y P(x, y)`	1. Move `¬` past `∀x`: `∃x ¬∃y P(x, y)` 2. Move `¬` past `∃y`: `∃x ∀y ¬P(x, y)` **English Example**: `¬("Everybody loves somebody")` `"Somebody loves nobody."` (i.e., there exists a person x such that for all people y, x does not love y).
`¬∃x ∀y P(x, y)`	1. Move `¬` past `∃x`: `∀x ¬∀y P(x, y)` 2. Move `¬` past `∀y`: `∀x ∃y ¬P(x, y)` **English Example**: `¬("There is somebody whom everyone loves")` `"For every person, there is someone they don't love."` (i.e., for every person x, there exists a person y such that x does not love y).
COMMON MISTAKE	Negating only the predicate and not flipping the quantifiers. Remember to flip *each* quantifier as the negation passes it.

Quantifiers in Proofs

APPLICATION IN PROOFS

To Prove `∀x P(x)`	**Strategy**: Universal Generalization. Let `x` be an *arbitrary* (but specific) element from the domain. Show that `P(x)` holds for this arbitrary `x`. **Example**: Prove `∀x (x^2 ≥ 0)` for `x ∈ R`. Let `x` be any real number. We know `x^2` is always non-negative. Thus, `x^2 ≥ 0`.
To Prove `∃x P(x)`	**Strategy**: Existential Generalization (Constructive Proof). Find or construct a *specific* element `x` in the domain for which `P(x)` is true. **Example**: Prove `∃x (x^2 = 4)` for `x ∈ Z`. We can pick `x = 2`. `2^2 = 4`. Done.
To Disprove `∀x P(x)` (i.e., Prove `∃x ¬P(x)`)	**Strategy**: Find a Counterexample. Provide *one specific* element `x` in the domain for which `P(x)` is false (or `¬P(x)` is true). **Example**: Disprove `∀x (x > 5)` for `x ∈ Z`. Let `x = 3`. `3 > 5` is false. Counterexample found.
To Disprove `∃x P(x)` (i.e., Prove `∀x ¬P(x)`)	**Strategy**: Show `¬P(x)` for all `x`. Use Universal Generalization on `¬P(x)`, or proof by contradiction. **Example**: Disprove `∃x (x^2 < 0)` for `x ∈ R`. Let `x` be any real number. We know `x^2` is always non-negative, so `x^2 < 0` is always false. Thus `∀x (x^2 ≥ 0)`.
Proving `∀x ∃y P(x, y)`	**Strategy**: For an *arbitrary* `x` from the domain, you must show how to find or construct a `y` (which typically depends on `x`) such that `P(x, y)` holds. **Example**: Prove `∀x ∈ R ∃y ∈ R (x + y = 0)`. Let `x` be an arbitrary real number. We need to find a `y` such that `x + y = 0`. Let `y = -x`. Since `x ∈ R`, `y = -x ∈ R`. So, for any `x`, we can find such a `y`.
Proving `∃y ∀x P(x, y)`	**Strategy**: You must find or construct a *specific* `y` from the domain, and then prove that for *every* `x` in the domain, `P(x, y)` holds for *that fixed* `y`. **Example**: Prove `∃y ∈ R ∀x ∈ R (x + y = x)`. We need to find a `y` that works for all `x`. Let `y = 0`. Now, for any arbitrary real number `x`, `x + 0 = x` is true. Thus, such a `y` exists.

Constructive vs. Non-Constructive Existence Proofs	**Constructive**: You explicitly provide an example of the element whose existence is claimed (e.g., finding a `y` for $\exists y$ `P(y)`). **Non-Constructive**: You prove the element must exist without actually finding it (e.g., using proof by contradiction, or showing that if it didn't exist, a contradiction would arise).
COMMON MISTAKE & STRATEGY TIP	A common mistake is confusing the order or dependency. Always remember: if $\exists y$ is inside $\forall x$, `y` can depend on `x` . If $\exists y$ is outside $\forall x$, `y` must be a single, fixed value for all `x` . **Strategy Tip**: To approach proofs with nested quantifiers, unpack them layer by layer, starting with the outermost quantifier. The strategy for handling the inner quantifier will often depend on the specific element picked by the outer quantifier.