Mathematical Logic

Part Two

Recap from Last Time

Recap So Far

- A ***propositional variable*** is a variable that is either true or false.
- The ***propositional connectives*** are
  - Negation: $\neg p$
  - Conjunction: $p \wedge q$
  - Disjunction: $p \vee q$
  - Implication: $p \rightarrow q$
  - Biconditional: $p \leftrightarrow q$
  - True: $\top$
  - False: $\bot$

Logical Equivalence

- Two propositional formulas φ and ψ are called ***equivalent*** if they have the same truth tables.
- We denote this by writing $\varphi \equiv \psi$.
- Some examples:
  - $\neg(p \wedge q) \equiv \neg p \vee \neg q$
  - $\neg(p \vee q) \equiv \neg p \wedge \neg q$
  - $\neg p \vee q \equiv p \to q$
  - $p \wedge \neg q \equiv \neg(p \to q)$

New Stuff!

Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \rightarrow x \geq 8 \vee y \geq 8$$

Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x < 8 \wedge y < 8 \rightarrow x + y \neq 16$$

"If $x < 8$ and $y < 8$, then $x + y \neq 16$"

Theorem: If $x + y = 16$, then $x \geq 8$ or $y \geq 8$.

Proof: By contrapositive. We prove that if $x < 8$ and $y < 8$, then $x + y \neq 16$. To see this, note that

$$\begin{aligned} x + y &< 8 + y \\ &< 8 + 8 \\ &= 16 \end{aligned}$$

This means that $x + y < 16$, so $x + y \neq 16$, which is what we needed to show. ■

Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \rightarrow x \geq 8 \vee y \geq 8$$

Why All This Matters

- Suppose we want to prove the following statement:

"If $x + y = 16$, then $x \geq 8$ or $y \geq 8$"

$$x + y = 16 \wedge x < 8 \wedge y < 8$$

"$x + y = 16$, but $x < 8$ and $y < 8$."

Theorem: If $x + y = 16$, then $x \geq 8$ or $y \geq 8$.

Proof: Assume for the sake of contradiction that $x + y = 16$, but $x < 8$ and $y < 8$. Then

$$\begin{aligned} x + y &< 8 + y \\ &< 8 + 8 \\ &= 16 \end{aligned}$$

So $x + y < 16$, contradicting that $x + y = 16$. We have reached a contradiction, so our assumption must have been wrong. Therefore if $x + y = 16$, then $x \geq 8$ or $y \geq 8$. ■

Why This Matters

- Propositional logic is a tool for reasoning about how various statements affect one another.
- To better understand how to prove a result, it often helps to translate what you're trying to prove into propositional logic first.
- That said, propositional logic isn't expressive enough to capture all statements. For that, we need something more powerful.

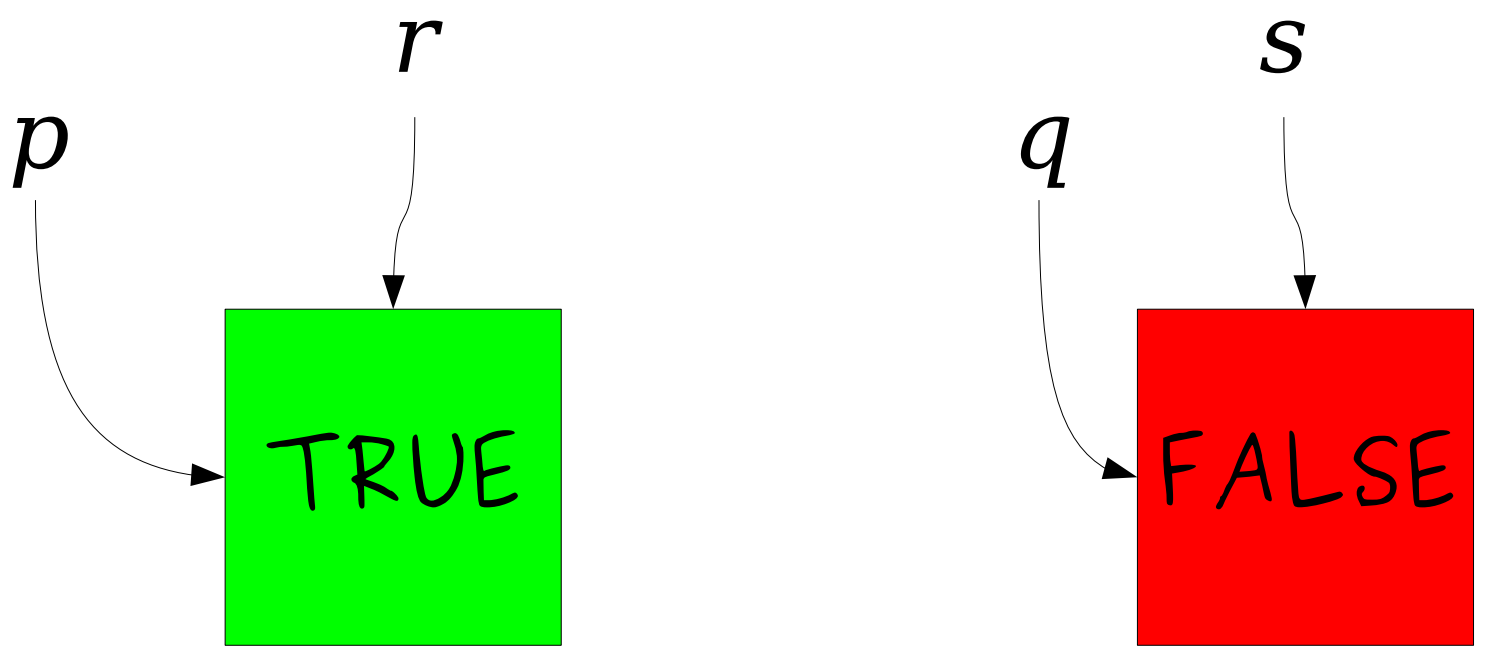
First-Order Logic

What is First-Order Logic?

- ***First-order logic*** is a logical system for reasoning about properties of objects.
- Augments the logical connectives from propositional logic with
  - ***predicates*** that describe properties of objects, and
  - ***functions*** that map objects to one another,
  - ***quantifiers*** that allow us to reason about multiple objects simultaneously.

The Universe of Propositional Logic

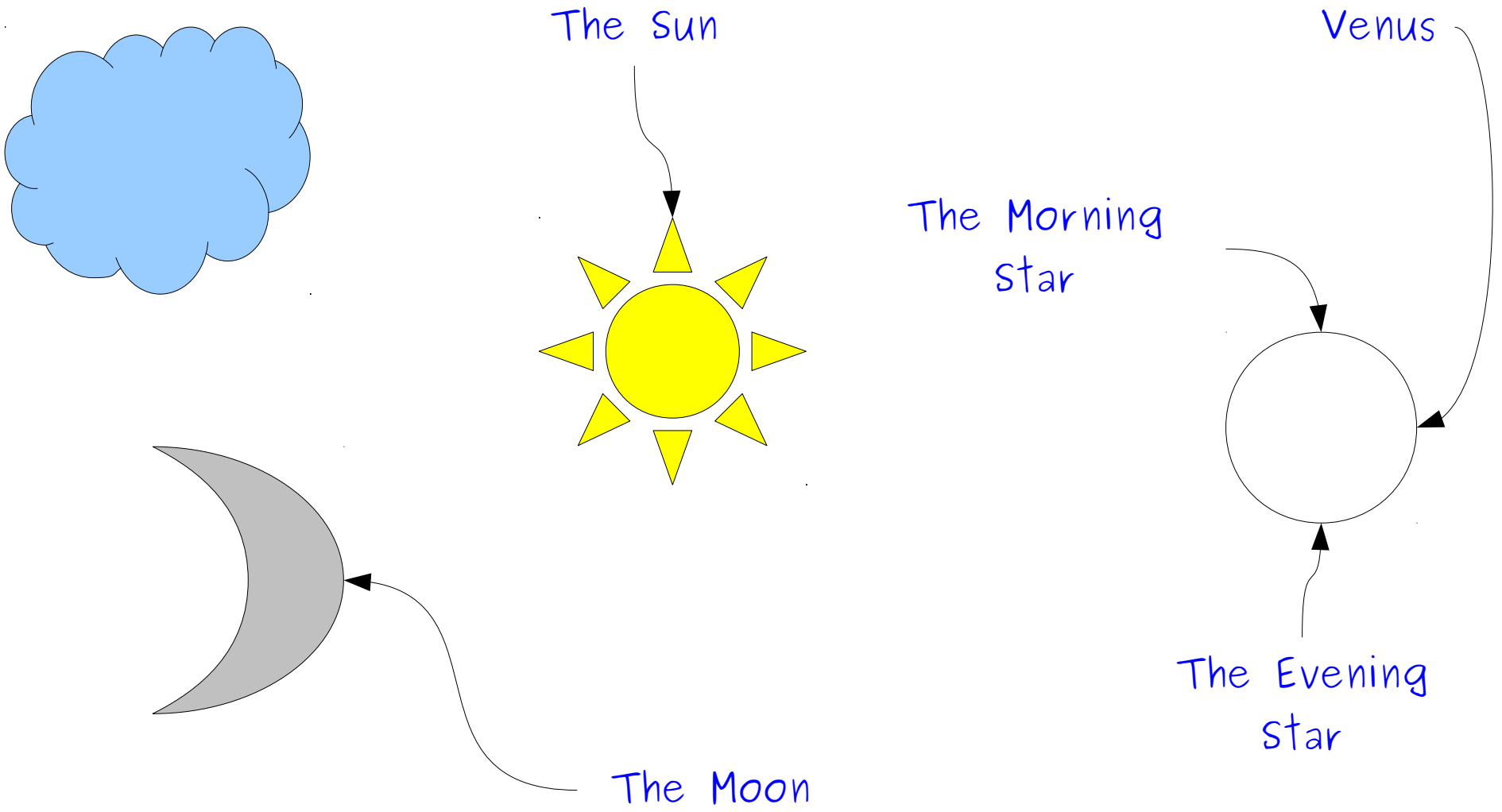
$$p \wedge q \rightarrow \neg r \vee \neg s$$



Propositional Logic

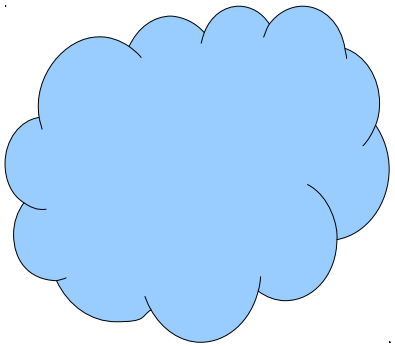
- In propositional logic, each variable represents a ***proposition***, which is either true or false.
- We can directly apply connectives to propositions:
  - $p \rightarrow q$
  - $\neg p \wedge q$
- The truth of a statement can be determined by plugging in the truth values for the input propositions and computing the result.
- We can see all possible truth values for a statement by checking all possible truth assignments to its variables.

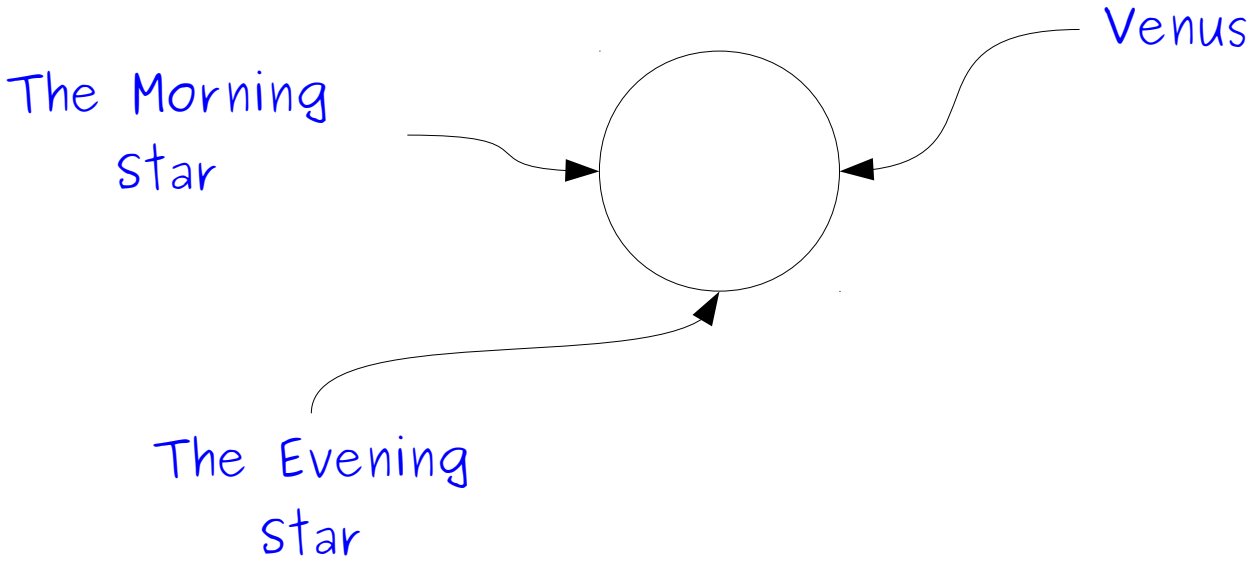
The Universe of First-Order Logic



First-Order Logic

- In first-order logic, each variable refers to some object in a set called the ***domain of discourse***.
- Some objects may have multiple names.
- Some objects may have no name at all.





Propositional vs. First-Order Logic

- Because propositional variables are either true or false, we can directly apply connectives to them.

$$p \rightarrow q \qquad \neg p \leftrightarrow q \wedge r$$

- Because first-order variables refer to arbitrary objects, it does not make sense to apply connectives to them.

$$\mathit{Venus} \rightarrow \mathit{Sun} \qquad 137 \leftrightarrow \neg 42$$

- ***This is not C!***

Reasoning about Objects

- To reason about objects, first-order logic uses ***predicates***.
- Examples:
  - *ExtremelyCute*(*Quokka*)
  - *DeadlockEachOther*(*House*, *Senate*)
- Predicates can take any number of arguments, but each predicate has a fixed number of arguments (called its ***arity***).
  - The arity and meaning of each predicate are typically specified in advance.
- Applying a predicate to arguments produces a proposition, which is either true or false.

First-Order Sentences

- Sentences in first-order logic can be constructed from predicates applied to objects:

$$LikesToEat(V, M) \wedge Near(V, M) \rightarrow WillEat(V, M)$$

$$Cute(t) \rightarrow Dikdik(t) \vee Kitty(t) \vee Puppy(t)$$

$$x < 8 \rightarrow x < 137$$

The notation **x < 8** is just a shorthand for something like ***LessThan*(x, 8)**. Binary predicates in math are often written like this, but symbols like **<** are not a part of first-order logic.

Equality

- First-order logic is equipped with a special predicate **=** that says whether two objects are equal to one another.
- Equality is a part of first-order logic, just as → and ¬ are.
- Examples:

$$MorningStar = EveningStar$$

$$TomMarvoloRiddle = LordVoldemort$$

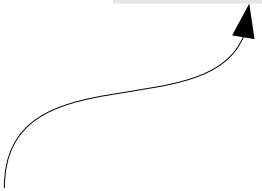
- Equality can only be applied to ***objects***; to see if ***propositions*** are equal, use ↔.

For notational simplicity, define $\neq$ as

$$x \neq y \quad \equiv \quad \neg(x = y)$$

Expanding First-Order Logic

$$(x < 8 \wedge y < 8) \rightarrow (x + y < 16)$$



Why is this allowed?

Functions

- First-order logic allows ***functions*** that return objects associated with other objects.
- Examples:

$$x + y$$

$$LengthOf(path)$$

$$MedianOf(x, y, z)$$

- As with predicates, functions can take in any number of arguments, but each function has a fixed arity.
  - As with predicates, the arity and interpretation of functions are specified in advance.
- Functions evaluate to ***objects***, not ***propositions***.
- There is no syntactic way to distinguish functions and predicates; you'll have to look at how they're used.

How would we translate the statement

"For any natural number n,
n is even if and only if n^2 is even"

into first-order logic?

Quantifiers

- The biggest change from propositional logic to first-order logic is the use of ***quantifiers***.
- A ***quantifier*** is a statement that expresses that some property is true for some or all choices that could be made.
- Useful for statements like “for every action, there is an equal and opposite reaction.”

"For any natural number n,
n is even iff n^2 is even"

$$\forall n.\ (n \in \mathbb{N} \rightarrow (Even(n) \leftrightarrow Even(n^2)))$$

∀ is the ***universal quantifier***
and says "for any choice of n,
the following is true."

The Universal Quantifier

- A statement of the form **$\forall x.\ \psi$** asserts that for ***every*** choice of x in our domain, ψ is true.
- Examples:

$$\forall v.\ (Puppy(v) \to Cute(v))$$

$$\forall n.\ (n \in \mathbb{N} \to (Even(n) \leftrightarrow \neg Odd(n)))$$

$$Tallest(SK) \to$$
$$\quad \forall x.\ (SK \neq x \to ShorterThan(x, SK))$$

Some muggles are intelligent.

$$\exists m.\ (Muggle(m) \wedge Intelligent(m))$$

∃ is the ***existential quantifier*** and says "for some choice of m, the following is true."

The Existential Quantifier

- A statement of the form **$\exists x.\ \psi$** asserts that for ***some*** choice of x in our domain, ψ is true.
- Examples:

  $\exists x.\ (\mathit{Even}(x) \wedge \mathit{Prime}(x))$

  $\exists x.\ (\mathit{TallerThan}(x, \mathit{me}) \wedge \mathit{LighterThan}(x, \mathit{me}))$

  $(\exists x.\ \mathit{Appreciates}(x, \mathit{me})) \rightarrow \mathit{Happy}(\mathit{me})$

Operator Precedence (Again)

- When writing out a formula in first-order logic, the quantifiers ∀ and ∃ have precedence just below ¬.
- Thus

$$\forall x.\ P(x) \lor R(x) \to Q(x)$$

is interpreted as the (malformed) statement

$$((\forall \mathbf{x}.\ P(\mathbf{x})) \lor R(\mathbf{x})) \to Q(\mathbf{x})$$

rather than the (intended, valid) statement

$$\forall x.\ (P(\mathbf{x}) \lor R(\mathbf{x}) \to Q(\mathbf{x}))$$

Time-Out for Announcements!

Problem Set Logistics

- Problem Set Two checkpoint was due at 12:50PM today. We'll try to get it back to you with feedback by Wednesday.
- The rest of Problem Set Two is due on Friday.
  - ***Start early!***
  - Ask questions in office hours, over email, or on Piazza!
- The TAs are humming along on Problem Set One. They'll have everything graded by Wednesday.

Problem Set Solutions

- We'll release hardcopies of the problem set solutions (and solution sets in general) in lecture.
- If you can't make it to lecture or forget to pick up a copy, you can pick up the solution sets from Gates.
- I'll show you where right now!


STANFORD ENGINEERING
VENTURE FUND LABORATORIES
WILLIAM GATES
COMPUTER SCIENCE


CS106 STAFF
CS106A
CS106B
Lecturer TA
Section Leaders
CSynII Coordinators
Lecturer
EE282
Spring 14/15
Handouts
Christos Kozyrakis
CS190
Spring 14/15
Handouts
John Ousterhout


CS103

Your Questions

“What is your favorite proof?”

I don't think I have a single “favorite” proof, though there are a bunch that I really like. My favorite from CS103 are the proof of Cantor's theorem and some of the impossibility results from later on. I'll point them out when we get there.

“Besides being interesting in its own right and good for cocktail parties, how does the material we learn in 103 come in handy later on?”

Finite automata, regular expressions, and grammars (in a few weeks) show up in all sorts of applications and are definitely useful to know about. The impossibility and hardness results we'll cover are good to know when on the job. However, the main benefit is being comfortable reasoning mathematically.

“Why do you teach instead of working in industry?”

“Why didn't you go on to get a PhD in CS theory; you seem to love the subject matter. Or is it something you might still do in the future?”

A few reasons:

1. High impact
2. High job satisfaction
3. Meet amazing people
4. Intellectual interest
5. Flexibility

“For applying to tech jobs, oftentimes a candidate could preferably include a list of projects he or she has done. How does one get started on building a profile of side projects? What types of projects make a well-rounded profile?”

Don't feel obligated to have side projects! Keep in mind that companies are looking at a lot of signals and this is just one of them.

If you want to do something fun on the side, go right ahead! Try to pick something you're actually interested in.

Back to CS103!

Translating into First-Order Logic

Translating Into Logic

- First-order logic is an excellent tool for manipulating definitions and theorems to learn more about them.
- Applications:
  - Determining the negation of a complex statement.
  - Figuring out the contrapositive of a tricky implication.

Translating Into Logic

- ***Translating statements into first-order logic is a lot more difficult than it looks.***
- There are a lot of nuances that come up when translating into first-order logic.
- We'll cover examples of both good and bad translations into logic so that you can learn what to watch for.
- We'll also show lots of examples of translations so that you can see the process that goes into it.

Using the predicates

- $Puppy(p)$, which states that p is a puppy, and
- $Cute(x)$, which states that x is cute,

write a sentence in first-order logic that means “all puppies are cute.”

An Incorrect Translation

All puppies are cute!

$\forall x.\ (\mathit{Puppy}(x) \wedge \mathit{Cute}(x))$

This should work for *any* choice of x, including things that aren't puppies.

An Incorrect Translation

All puppies are cute!

~~$\forall x. (Puppy(x) \wedge Cute(x))$~~

This logical statement can be made false regardless of whether all puppies are cute. It's therefore not a faithful translation.

A Better Translation

All puppies are cute!

$\forall x.\ (Puppy(x) \rightarrow Cute(x))$

This should work for any choice of x, including things that aren't puppies.

“All P's are Q's”

translates as

$$\forall x.\ (P(x) \rightarrow Q(x))$$

Useful Intuition:

Universally-quantified statements are true unless there's a counterexample.

$$\forall x.\ (P(x) \to Q(x))$$

If x is a counterexample, it must have property P but not have property Q.

Using the predicates

- *Blobfish*(b), which states that b is a blobfish, and
- *Cute*(x), which states that x is cute,

write a sentence in first-order logic that means “some blobfish is cute.”

An Incorrect Translation

Some blobfish is cute.

$\exists x.\ (\mathit{Blobfish}(x) \rightarrow \mathit{Cute}(x))$

What happens if

1. The English statement is false, but
2. x refers to a puppy?

An Incorrect Translation

Some blobfish is cute.

$$\exists x.\ (\mathit{Blobfish}(x) \to \mathit{Cute}(x))$$

Although the English statement is false, this logical statement is true. It's therefore not a correct translation.

An Incorrect Translation

Some blobfish is cute.

$\exists x.\ (\mathit{Blobfish}(x) \wedge \mathit{Cute}(x))$

What happens if

1. The English statement is false, but
2. x refers to a puppy?

“Some *P* is a *Q*”

translates as

$$\exists x.\ (P(x) \wedge Q(x))$$

Useful Intuition:

Existentially-quantified statements are false unless there's a positive example.

$$\exists x.\ (P(x) \wedge Q(x))$$

If x is an example, it *must* have property P on top of property Q.

Good Pairings

- The ∀ quantifier *usually* is paired with →.
- The ∃ quantifier *usually* is paired with ∧.
- In the case of ∀, the → connective prevents the statement from being *false* when speaking about some object you don't care about.
- In the case of ∃, the ∧ connective prevents the statement from being *true* when speaking about some object you don't care about.

Using the predicates

- *Tall*(*t*), which states that *t* is tall;
- *Tree*(*t*), which states that *t* is a tree; and
- *Sequoia*(*s*), which states that *s* is a sequoia,

write a sentence in first-order logic that means “there's a tall tree that's a sequoia.”

Checking a Translation

There's a tall tree that's a sequoia.

$$\exists t.\ (\mathit{Tree}(t) \wedge (\mathit{Tall}(t) \rightarrow \mathit{Sequoia}(t)))$$

What if we pick t to be a short tree?

Checking a Translation

There's a tall tree that's a sequoia.

$$\exists t.\ (\mathit{Tree}(t) \wedge (\mathit{Tall}(t) \rightarrow \mathit{Sequoia}(t)))$$

This statement can be true even if no tall sequoias exist.

Checking a Translation

There's a tall tree that's a sequoia.

$\exists t.\ (\mathit{Tree}(t) \wedge \mathit{Tall}(t) \wedge \mathit{Sequoia}(t))$

Do you see why this statement doesn't have this problem?