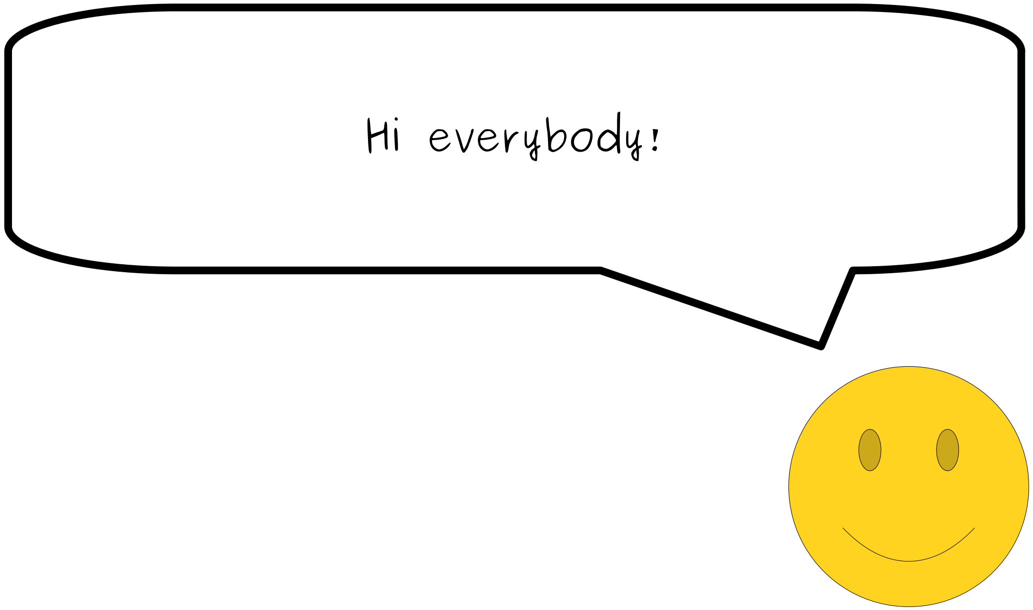
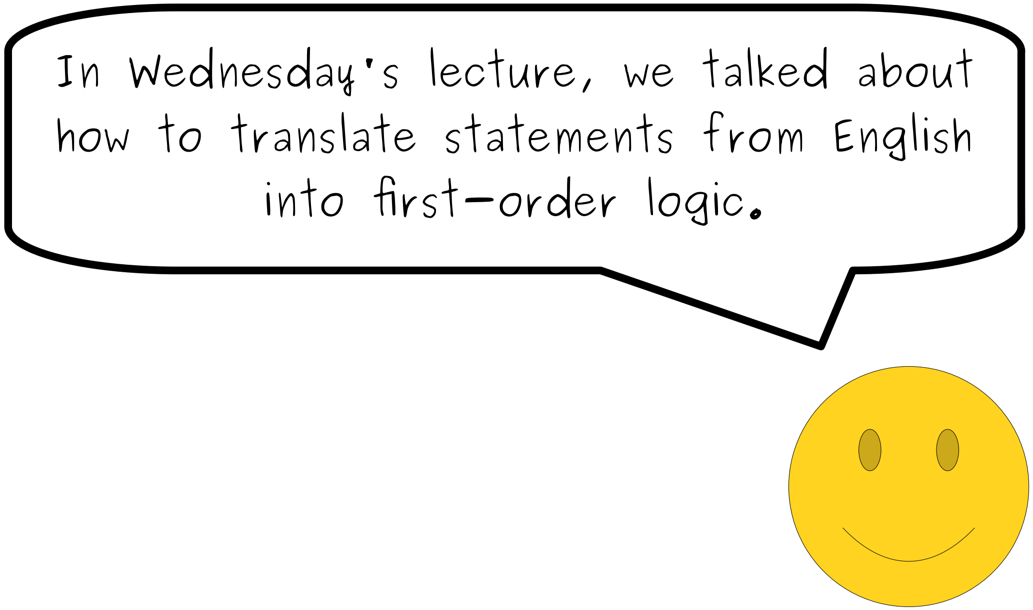
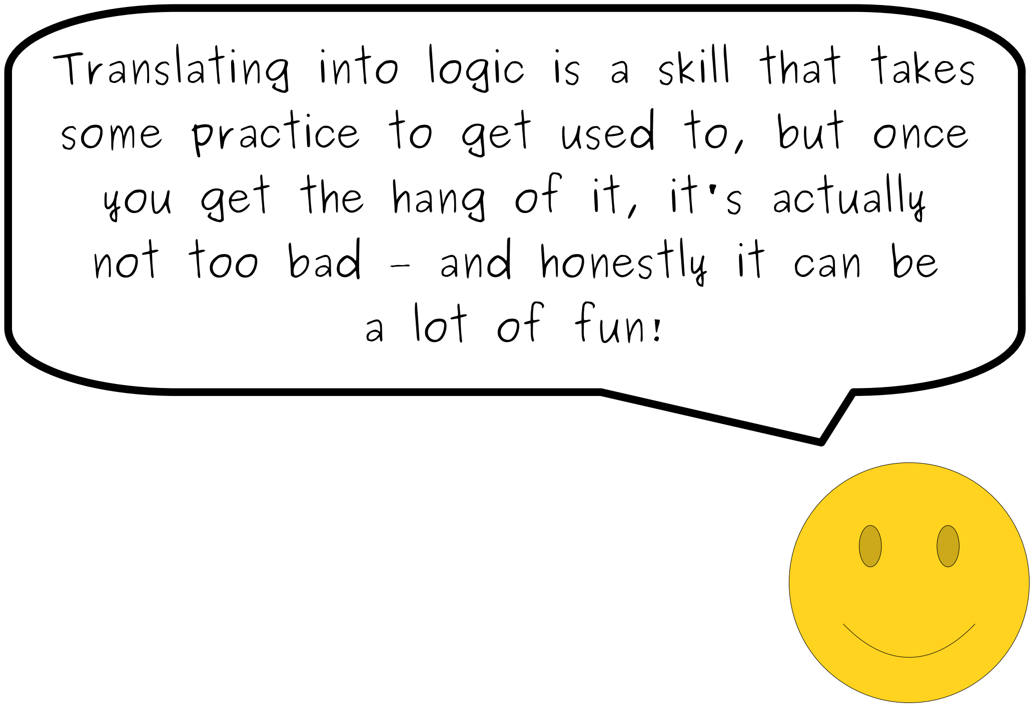
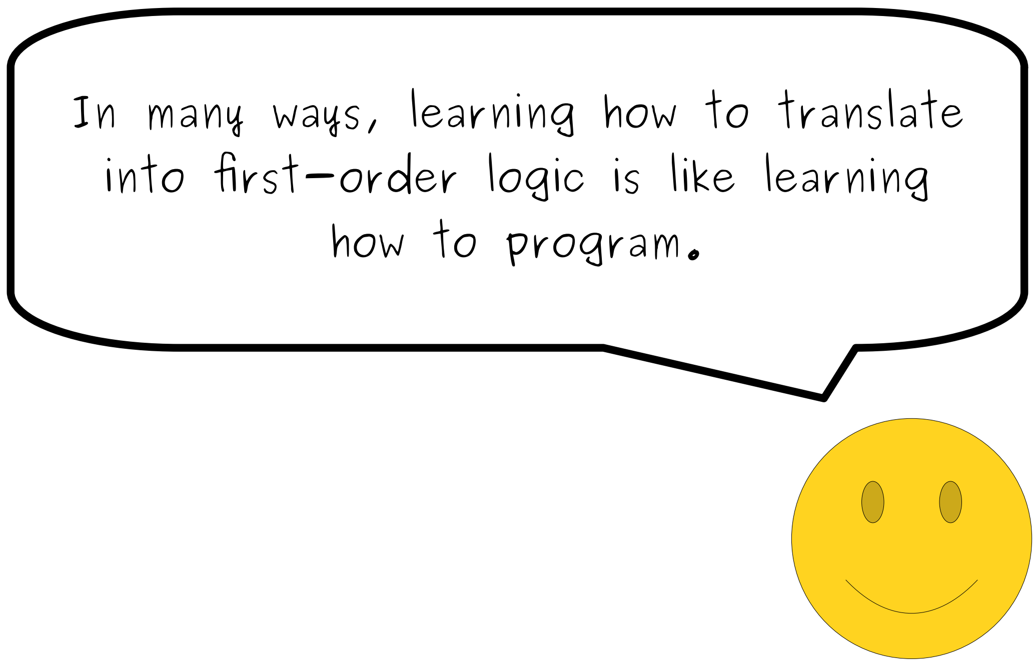
Guide to First-Order Logic Translations


Hi everybody!


In Wednesday's lecture, we talked about how to translate statements from English into first-order logic.

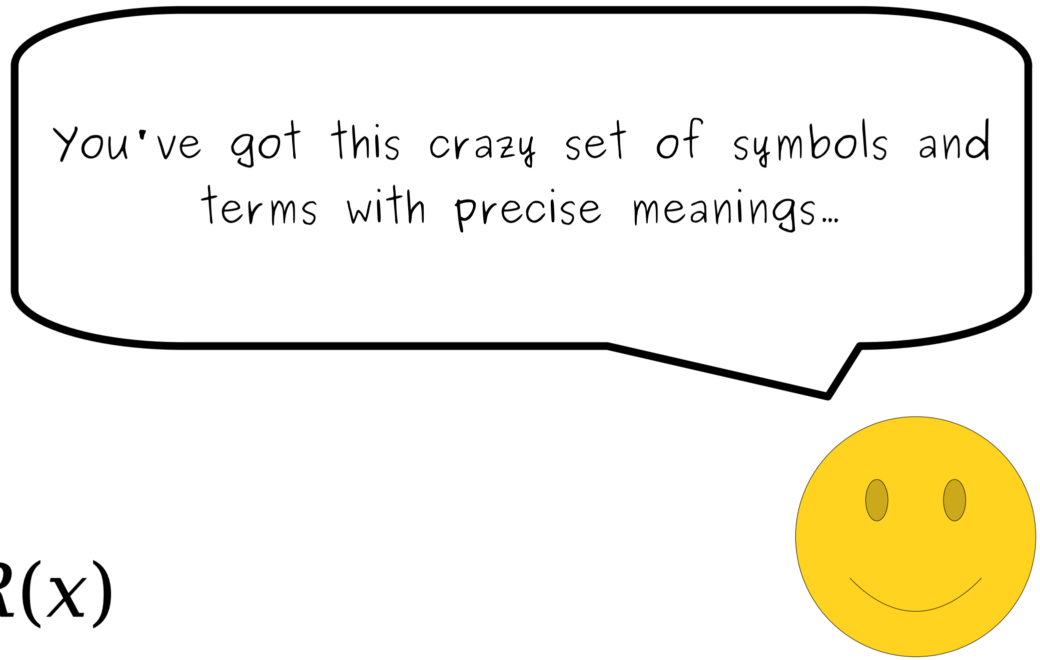

Translating into logic is a skill that takes some practice to get used to, but once you get the hang of it, it's actually not too bad – and honestly it can be a lot of fun!


In many ways, learning how to translate into first-order logic is like learning how to program.

$P(x)$

$\wedge$

$\exists$



$\vee$

$Q(x, y)$

$R(x)$

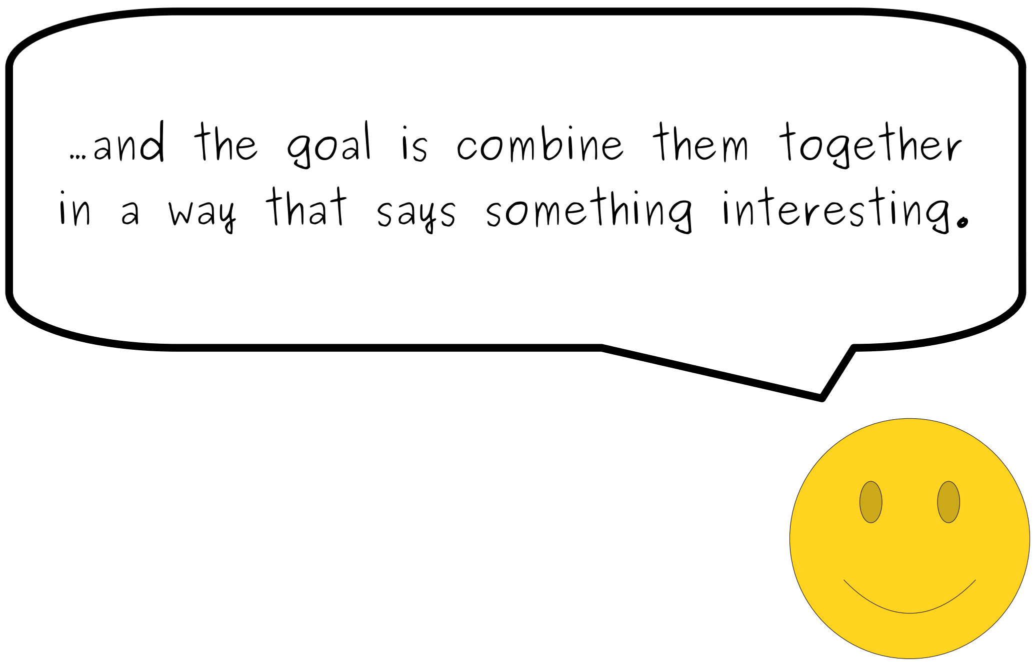
$\rightarrow$

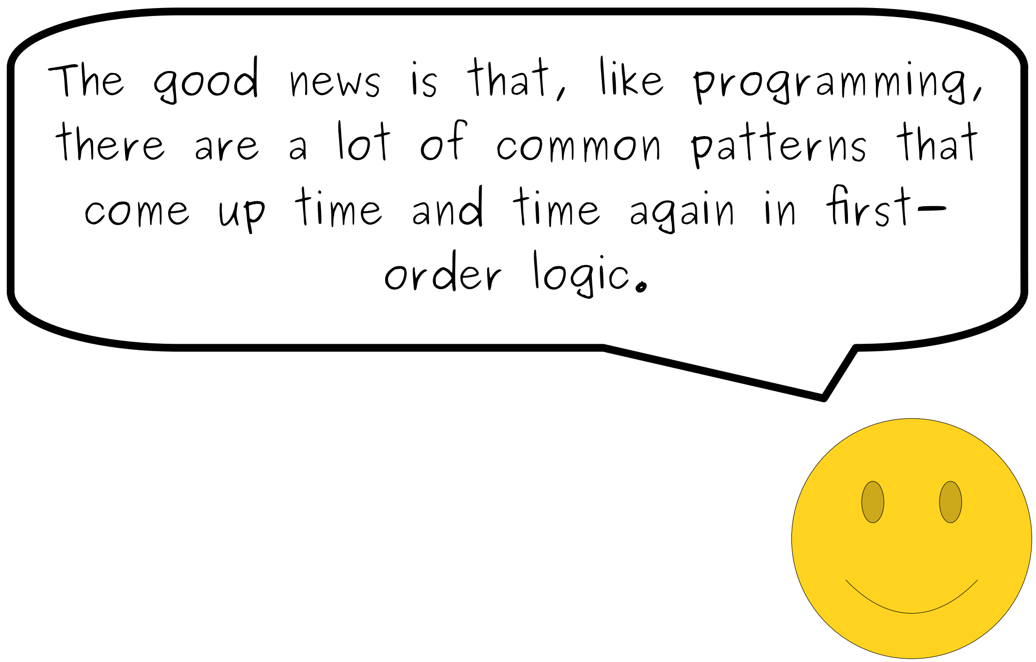
$\forall$

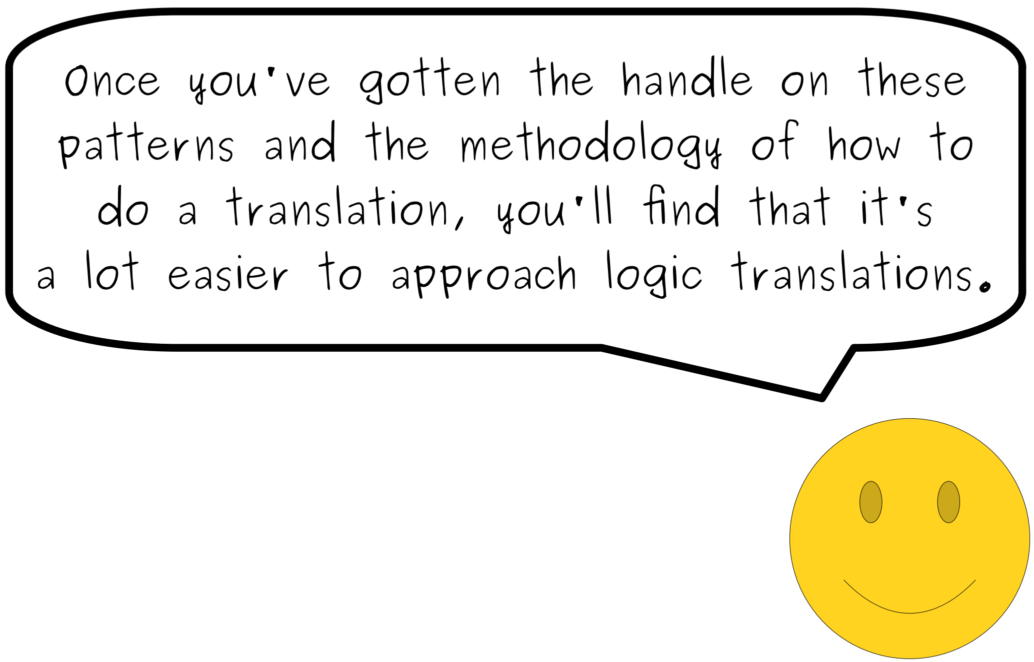
$S(y)$

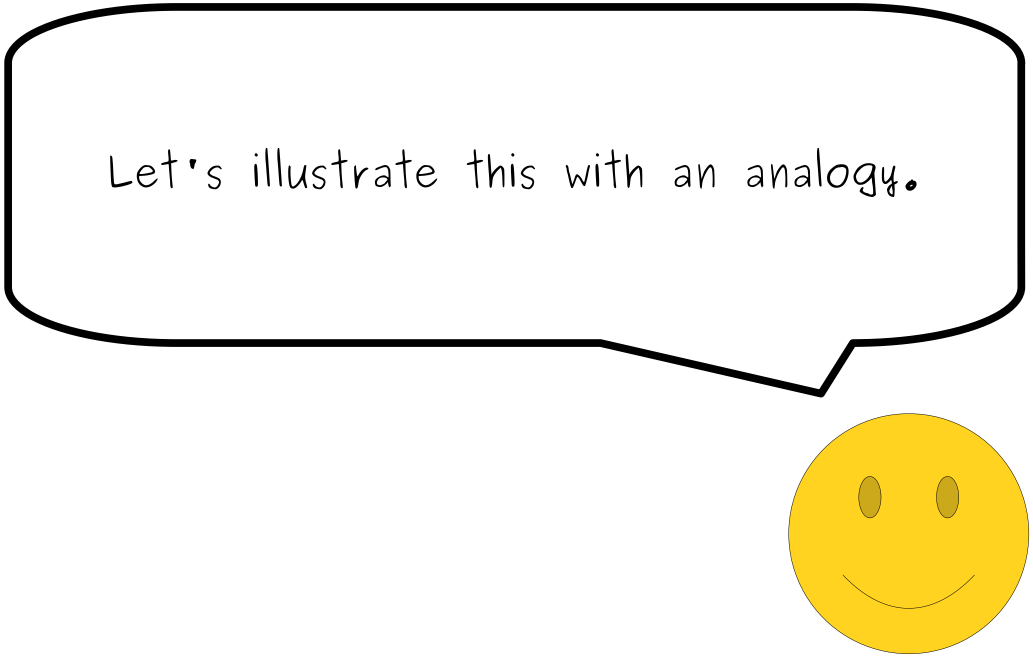
$=$

$$
\begin{aligned}
&\forall x.\ (P(x) \lor R(x) \to \\
&\qquad \exists y.\ (S(y) \land Q(x, y)) \\
&)
\end{aligned}
$$

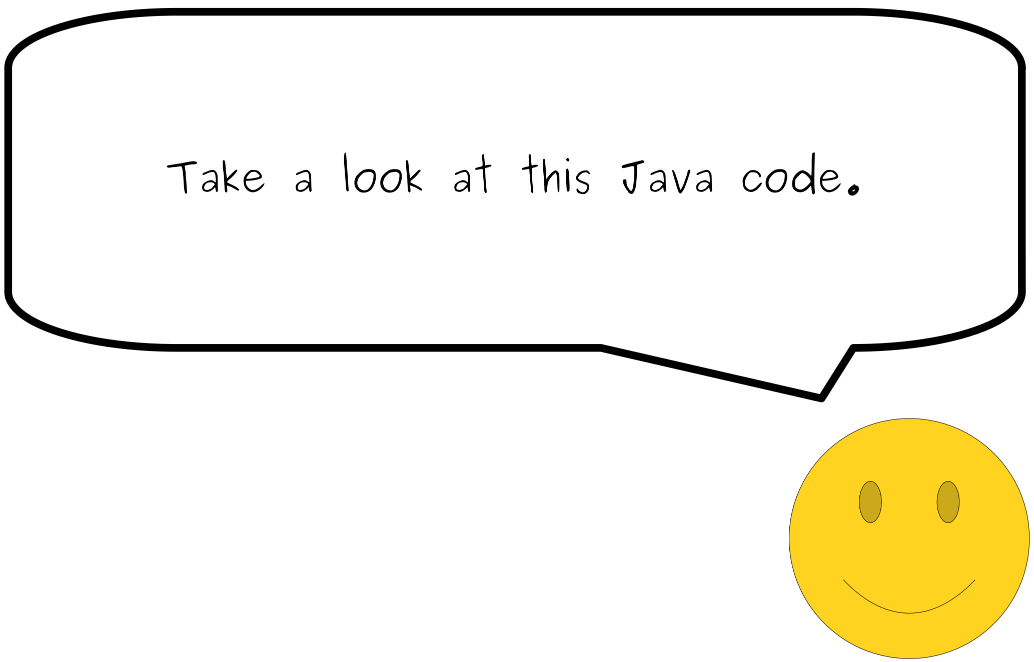



The good news is that, like programming, there are a lot of common patterns that come up time and time again in first-order logic.

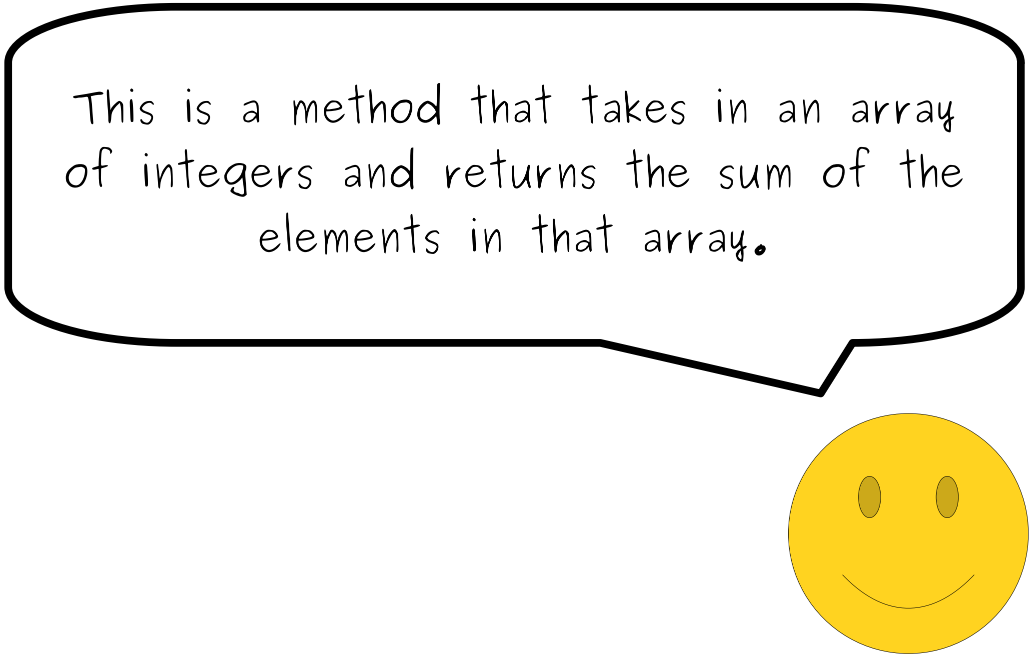

Once you've gotten the handle on these patterns and the methodology of how to do a translation, you'll find that it's a lot easier to approach logic translations.


Let's illustrate this with an analogy.

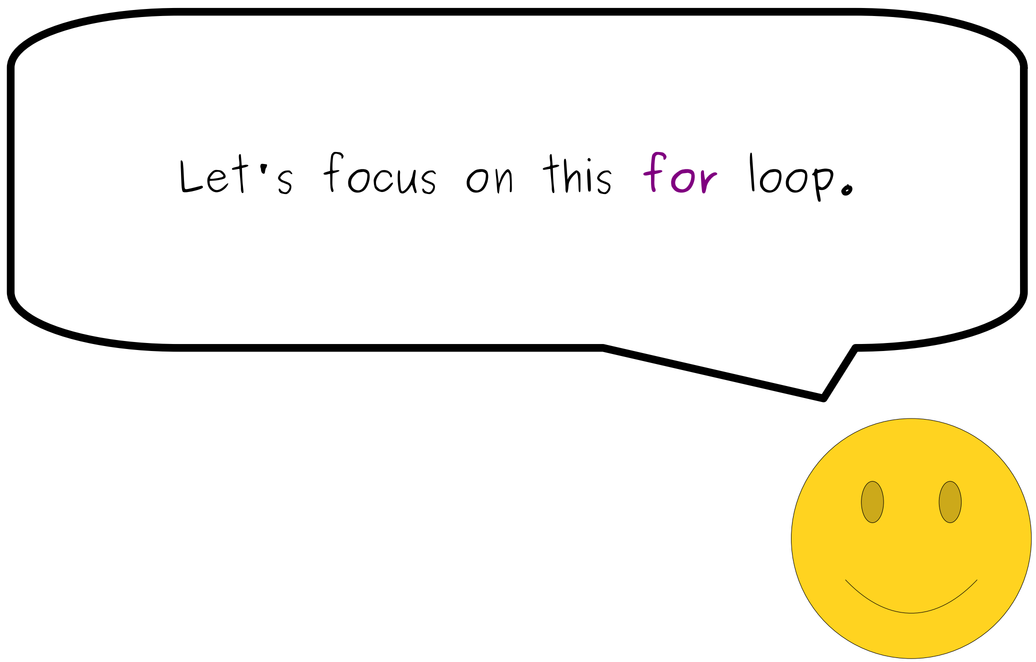
```
int sumOf(vector<int> elems) {
   int result = 0;
   for (int i = 0; i < elems.size(); i++) {
      result += elems[i];
   }
   return result;
}
```



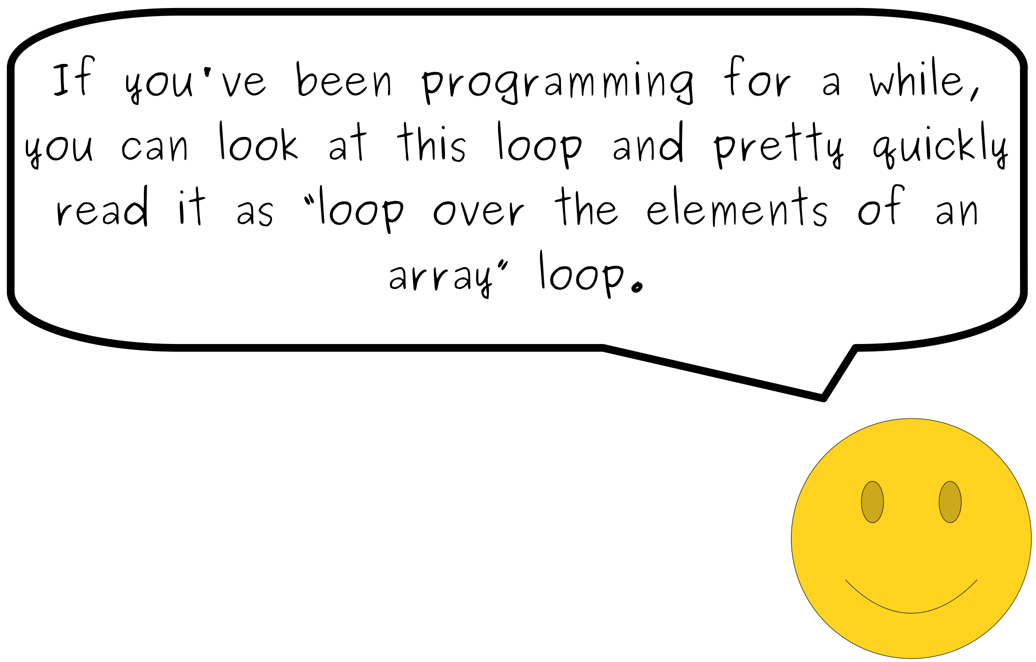
```
int sumOf(vector<int> elems) {
   int result = 0;
   for (int i = 0; i < elems.size(); i++) {
      result += elems[i];
   }
   return result;
}
```



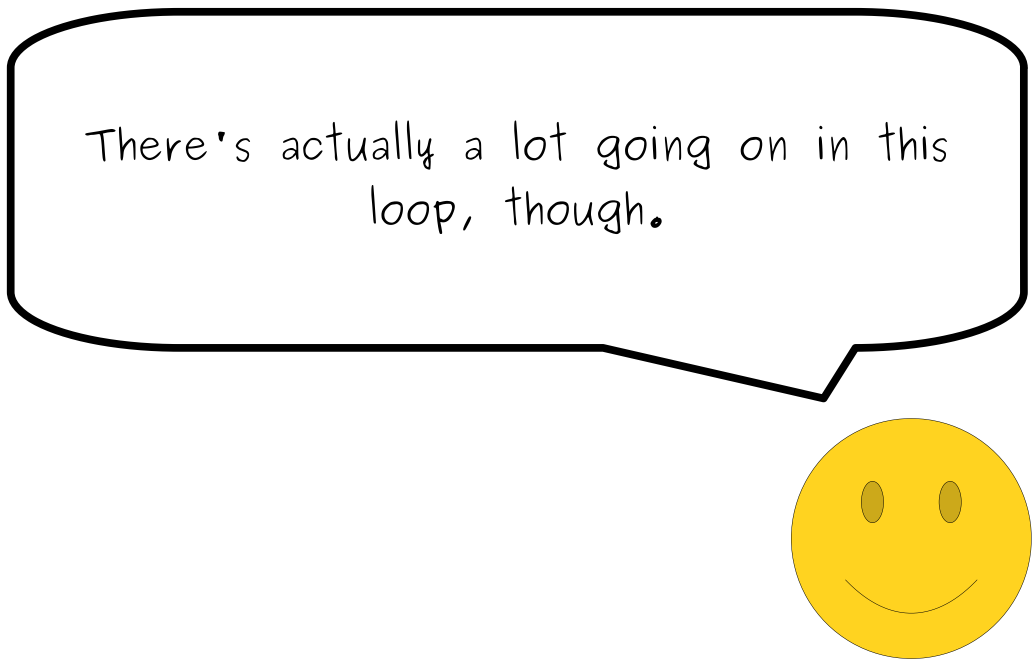
```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```



```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```



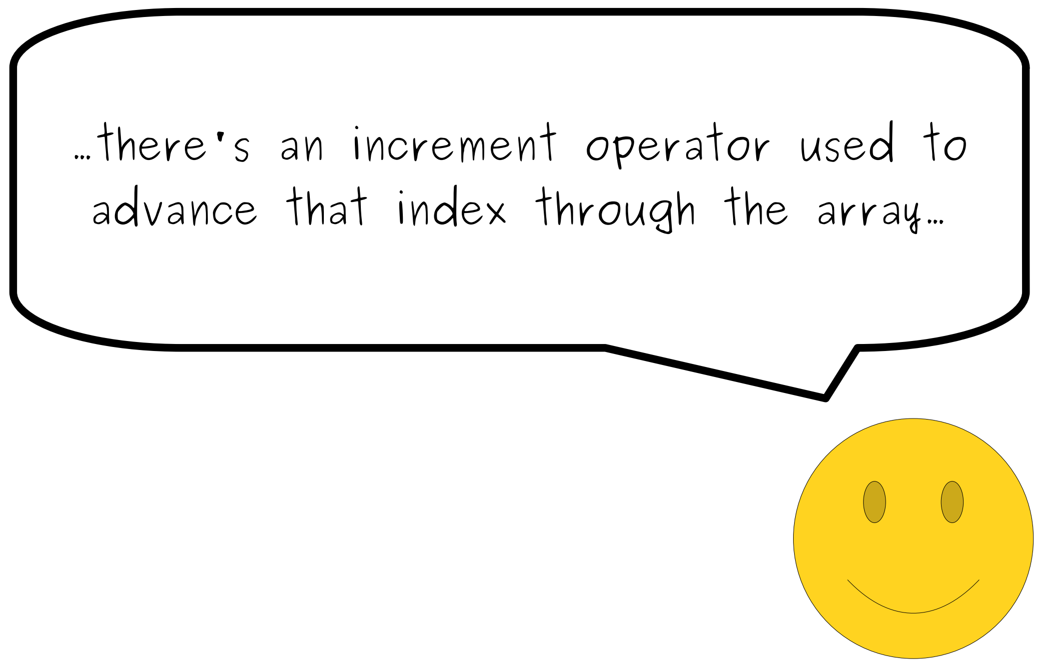
```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```



```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

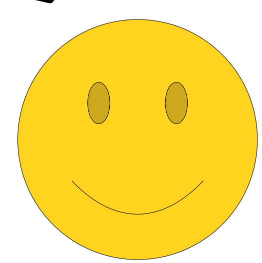
There's a variable declaration here that makes a new variable that tracks an index...

```cpp
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

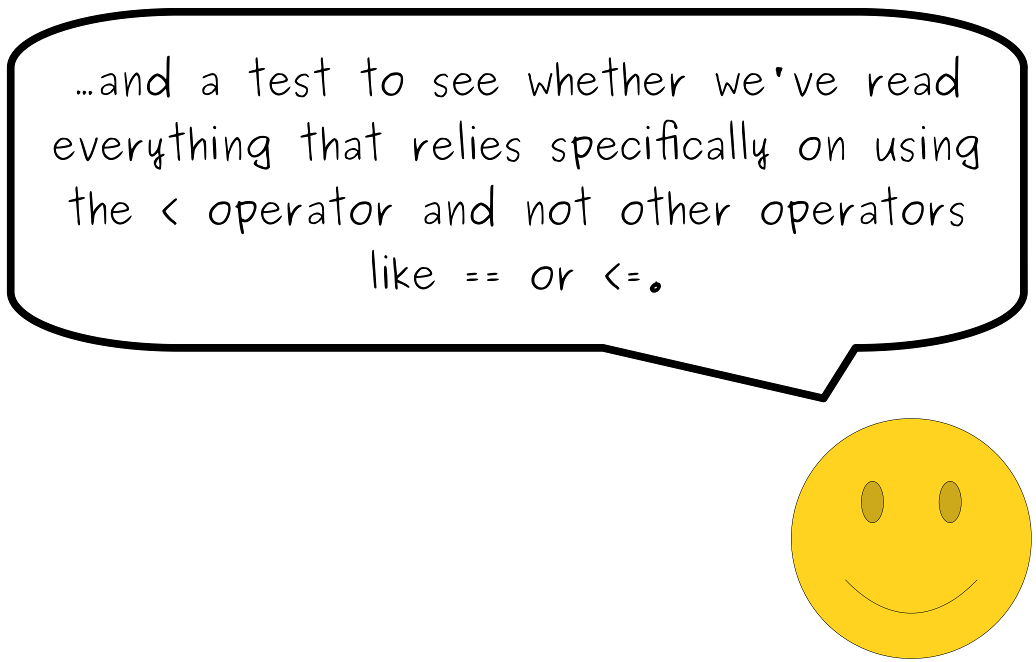


```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

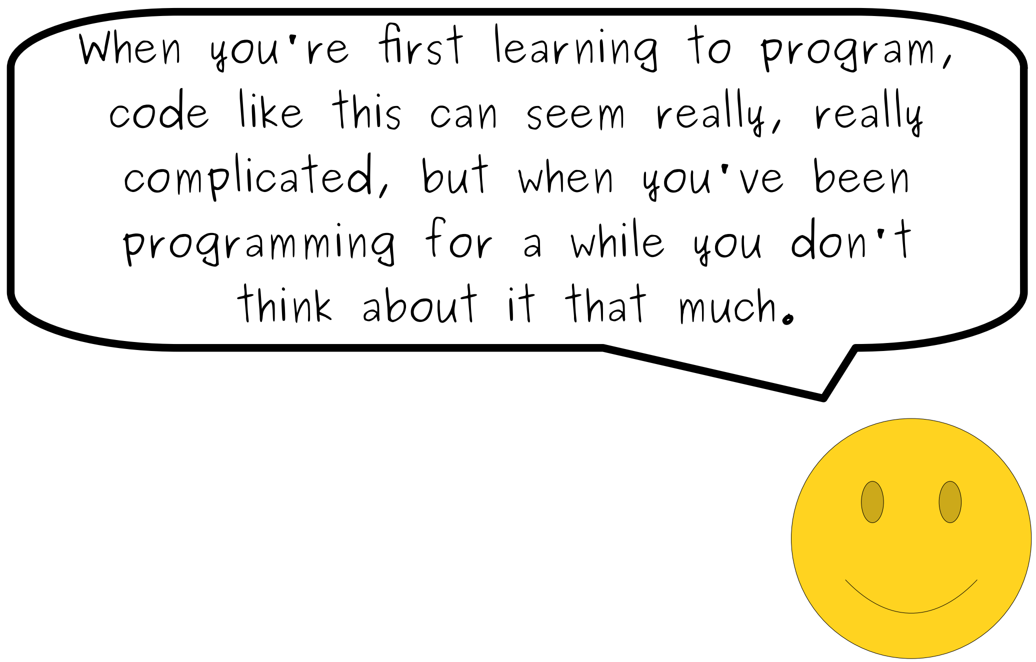
…a selection statement that picks out a single array element by using the variable we declared in the loop…



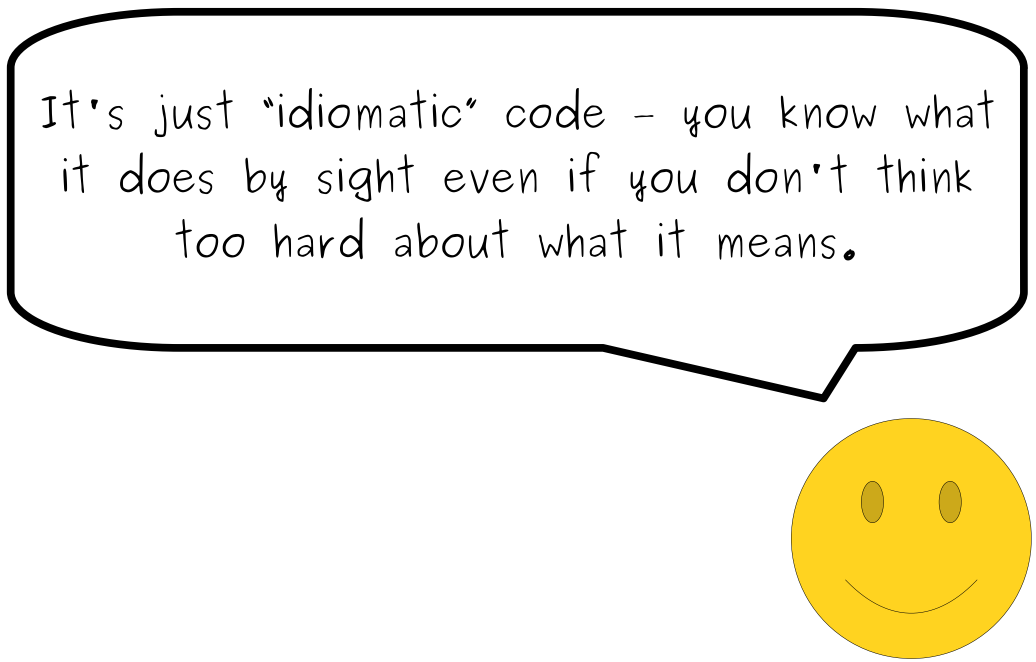
```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

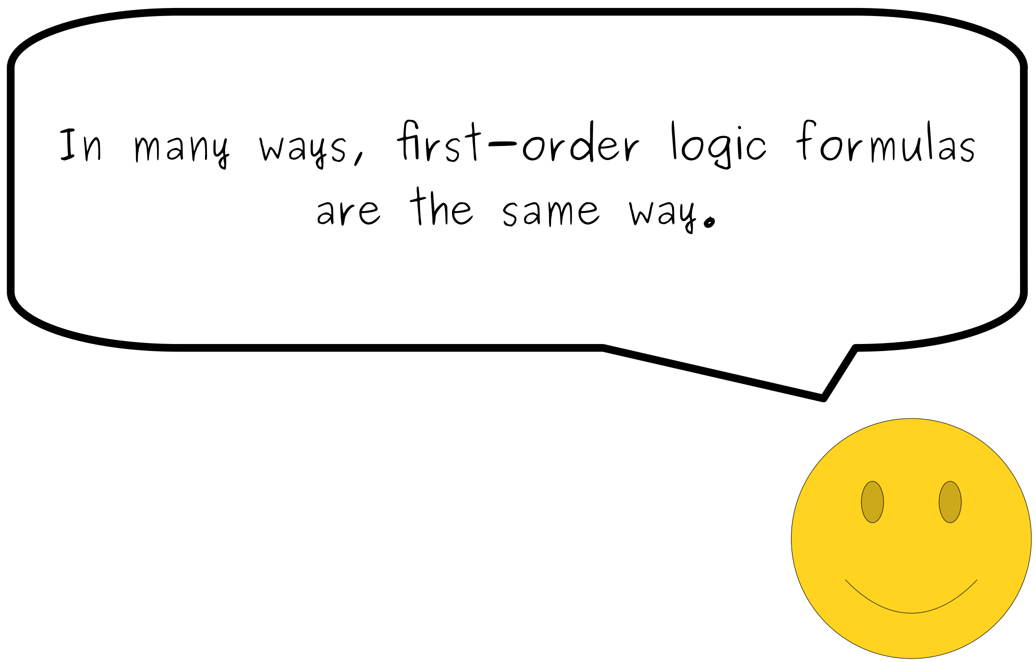


```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

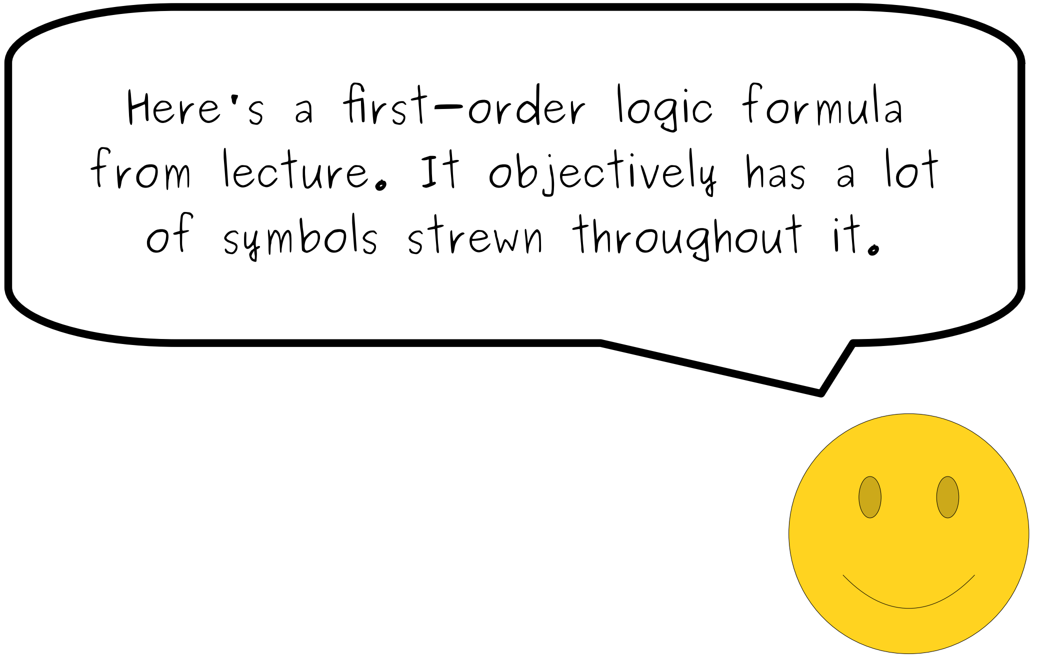


```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

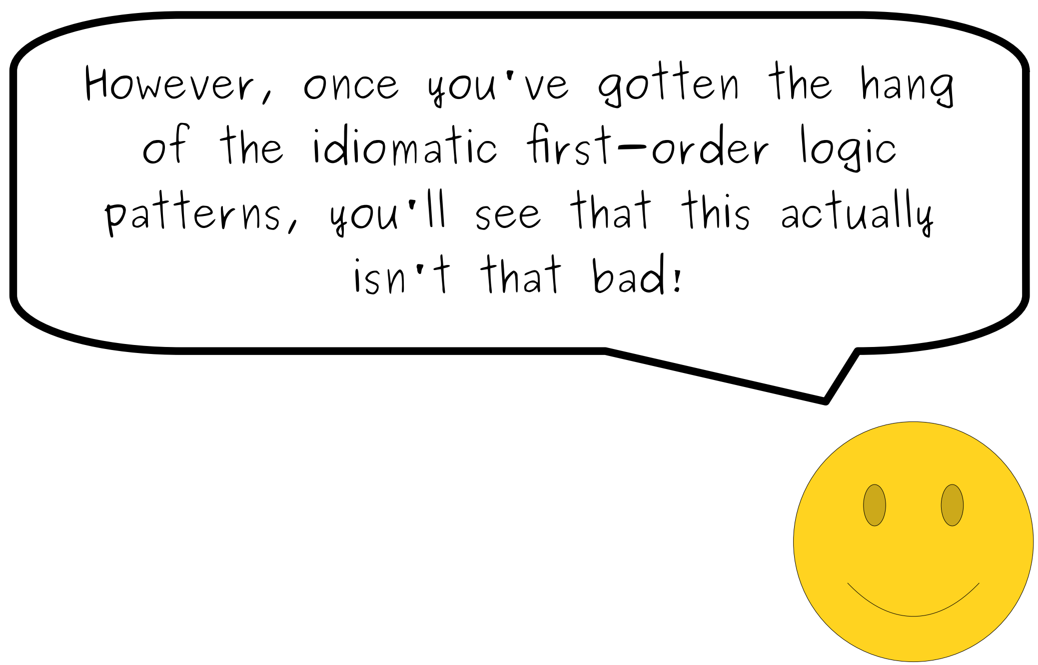



In many ways, first-order logic formulas are the same way.

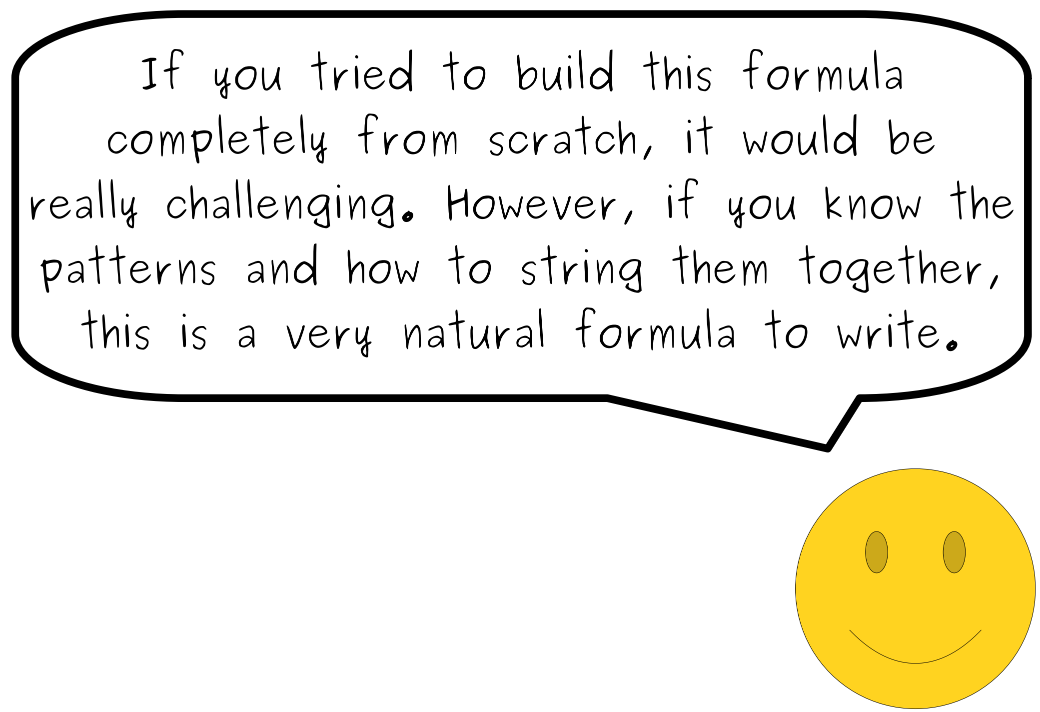
$$
\begin{aligned}
&\forall p.\ (Person(p) \to \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\qquad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$



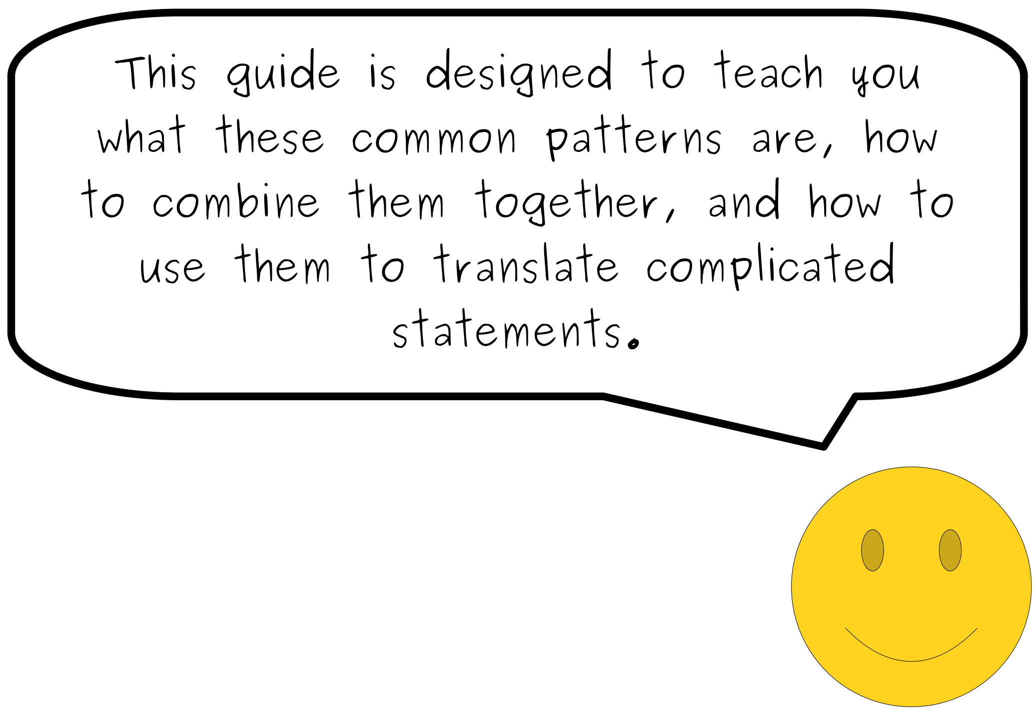
$$
\begin{aligned}
&\forall p.\ (Person(p) \to \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\qquad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$



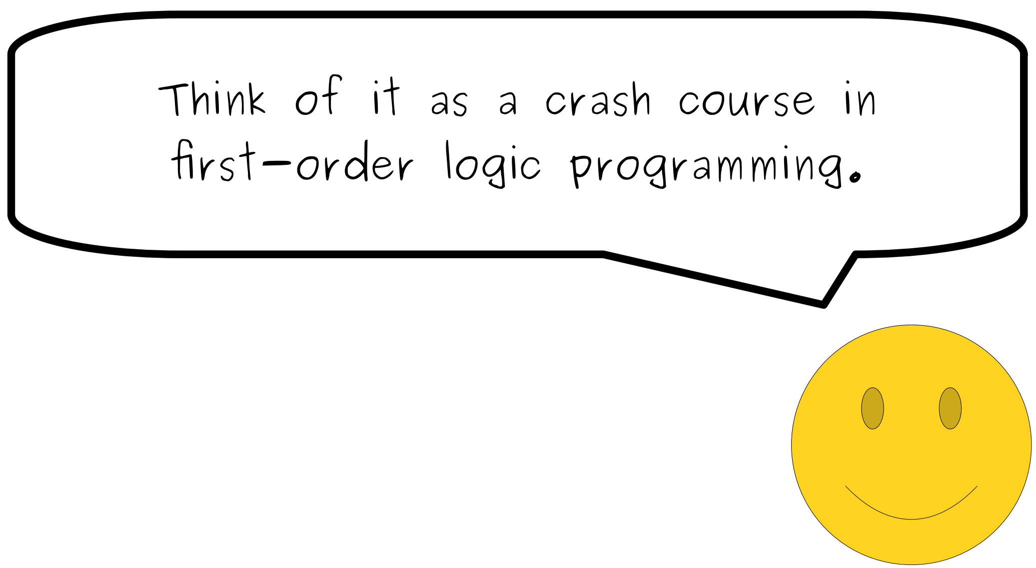
$$
\begin{aligned}
&\forall p.\ (Person(p) \rightarrow \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\quad\quad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$



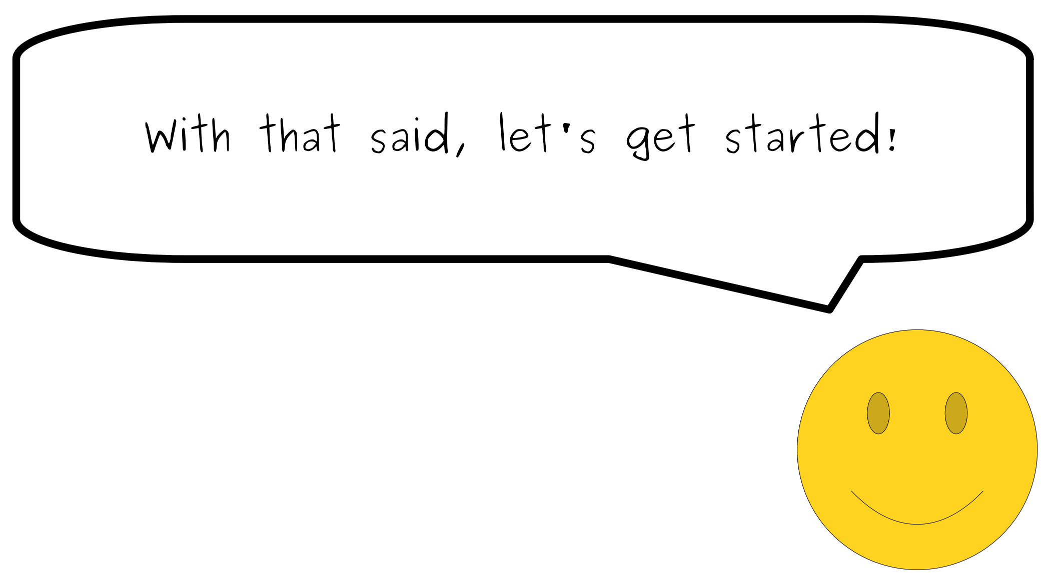
$$
\begin{aligned}
&\forall p.\ (Person(p) \to \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\qquad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$

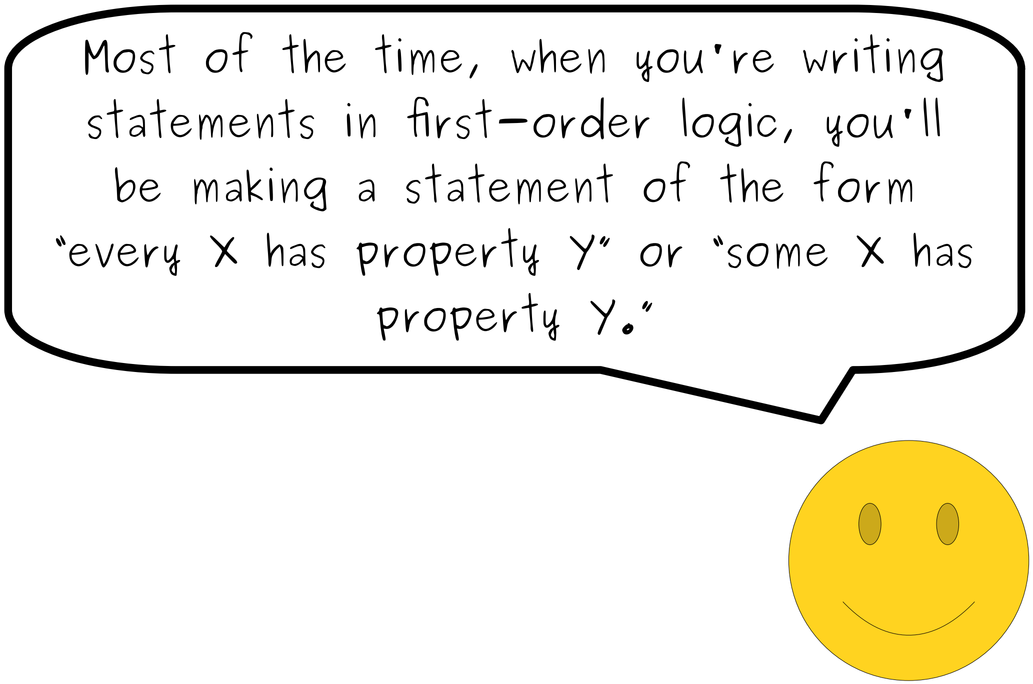


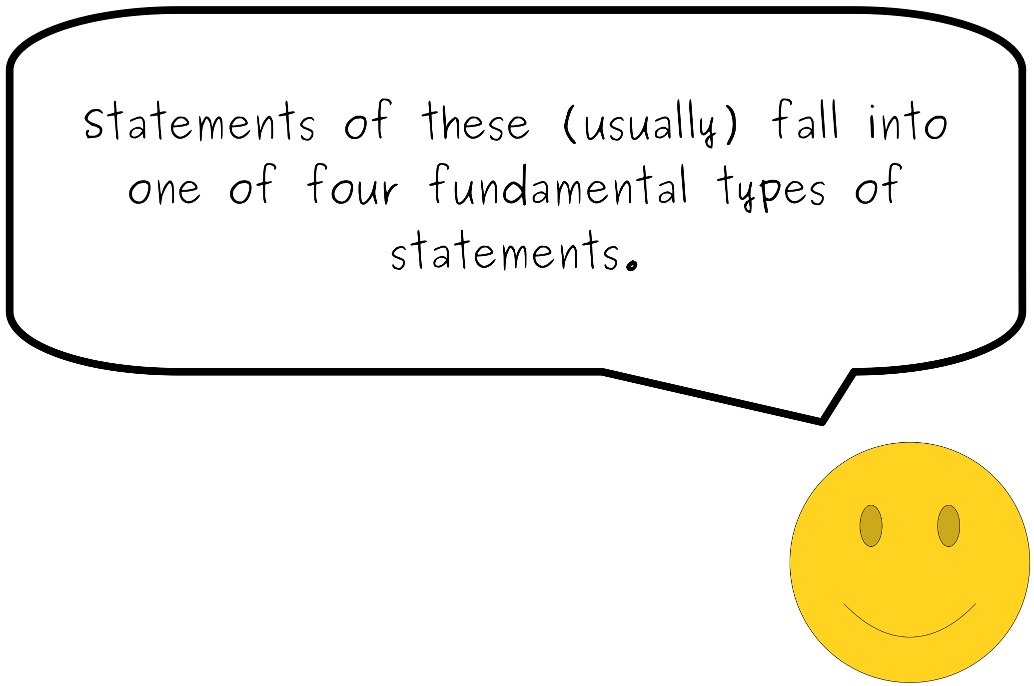
$$
\begin{aligned}
&\forall p.\ (Person(p) \to \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\qquad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$



$$
\begin{aligned}
&\forall p.\ (Person(p) \to \\
&\quad \exists q.\ (Person(q) \land p \neq q \land \\
&\qquad Loves(p, q) \\
&\quad) \\
&)
\end{aligned}
$$




Most of the time, when you're writing statements in first-order logic, you'll be making a statement of the form "every X has property Y" or "some X has property Y."


Statements of these (usually) fall into one of four fundamental types of statements.

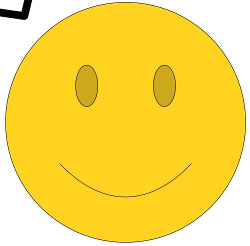
"All *P*s are *Q*s."

"Some *P*s are *Q*s."

"No *P*s are *Q*s."

"Some *P*s aren't *Q*s."

These four classes of statements are called **Aristotelian Forms**, since they were first described by Aristotle in his work "Prior Analytics" … though you don't need to know that unless you want to show off at cocktail parties. ^_^



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

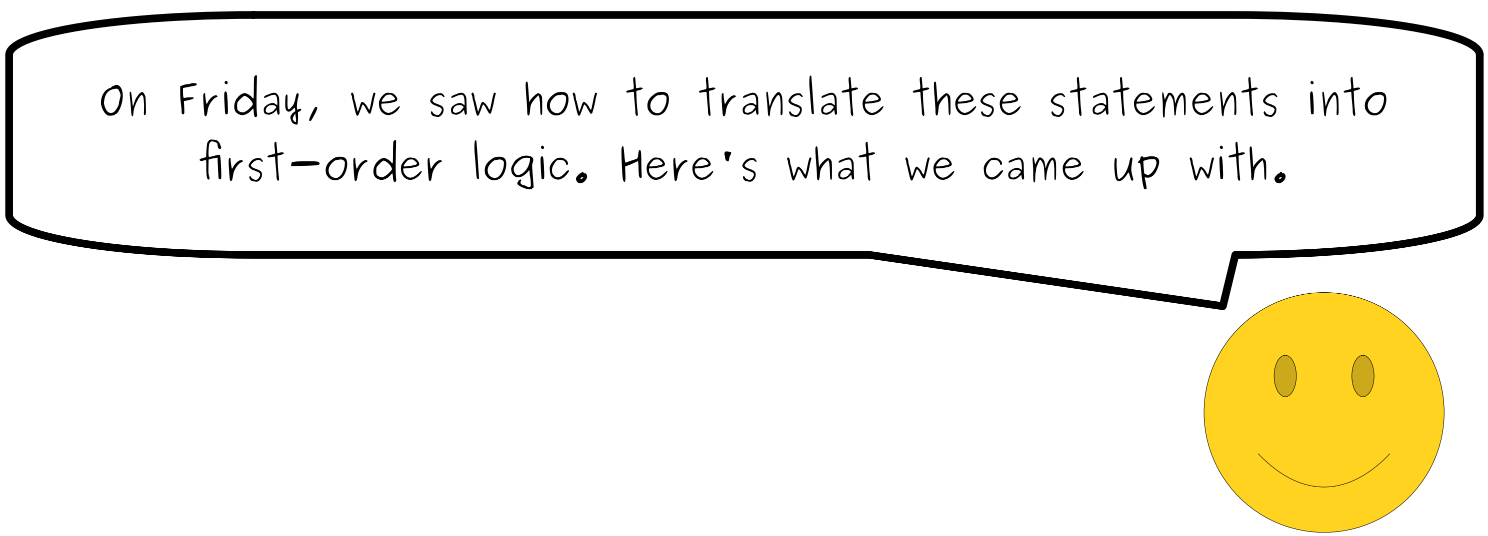
$\exists x.\ (P(x) \land Q(x))$

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

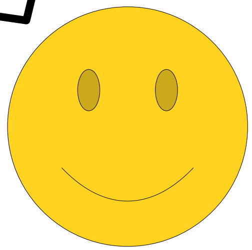
"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

In lecture we spent time talking about why ∀ gets paired with → and why ∃ gets paired with ∧. We already talked in lecture about why this is, so we're not going to review it here. After all, our goal is to see how to use these patterns, not how to derive them.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

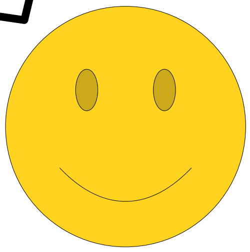
"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

However, you absolutely should memorize these patterns. They're like the "loop over an array" for loop pattern in Python, C, or C++ – they come up frequently and you ultimately want to get to the point where you can easily read and write them as a unit.



"All Ps are Qs."

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

"Some Ps are Qs."

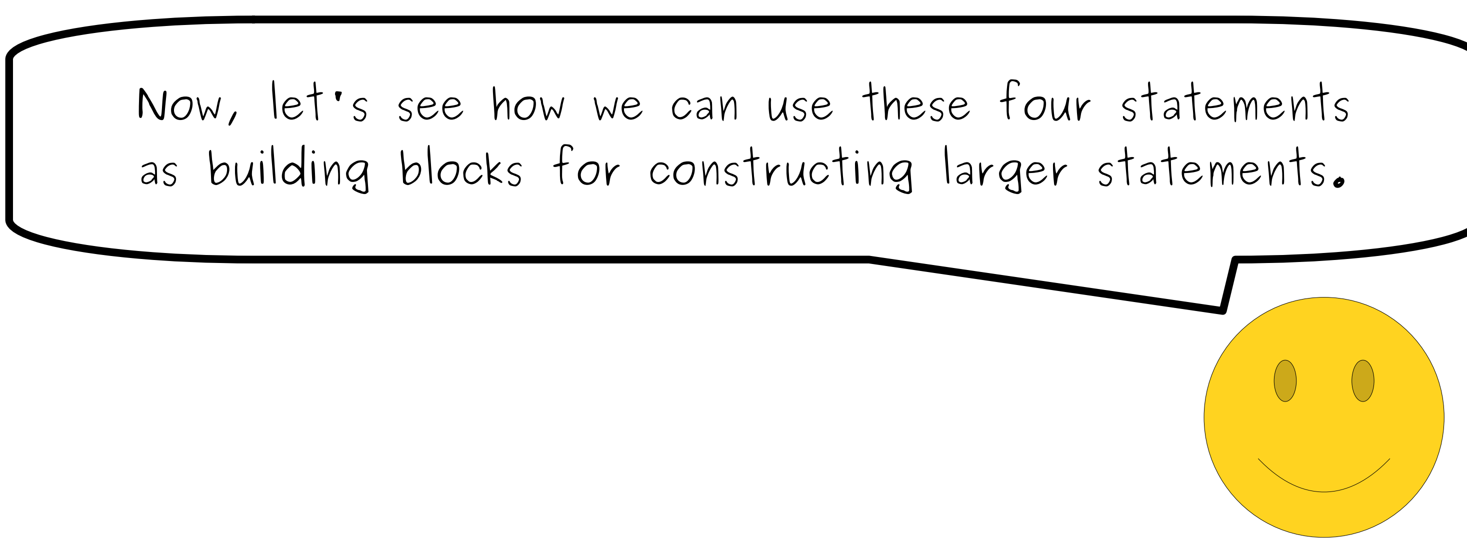
$$\exists x.\ (P(x) \land Q(x))$$

"No Ps are Qs."

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

"Some Ps aren't Qs."

$$\exists x.\ (P(x) \land \neg Q(x))$$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

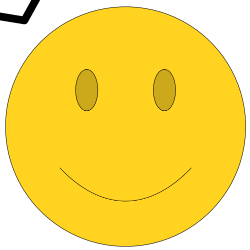
"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Imagine that we have these predicates available to us to use…



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

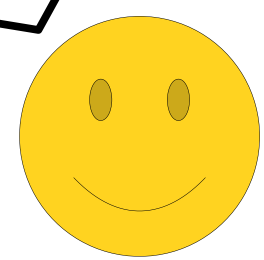
$\exists x.\ (P(x) \land \neg Q(x))$

Every orange cat is fluffy.

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

…and that we want to translate this statement into first-order logic.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

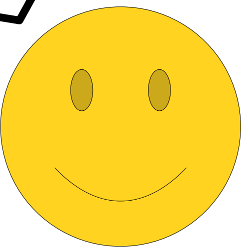
$\exists x. (P(x) \land \neg Q(x))$

Every orange cat is fluffy.

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Let's see how we can use these formulas to help out our translation.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

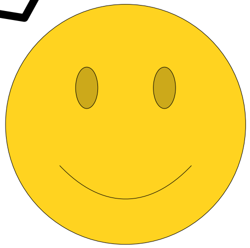
$$\exists x.\ (P(x) \land \neg Q(x))$$

Every orange cat is fluffy.

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

First, what kind of statement is this?



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

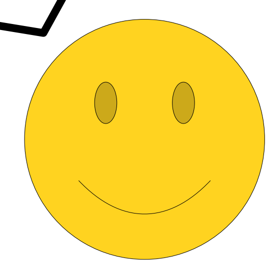
$\exists x.\ (P(x) \land \neg Q(x))$

Every orange cat is fluffy.

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

It seems to look a lot like this one – we're saying that all objects of one kind (orange cats) are also of another kind (fluffy).



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

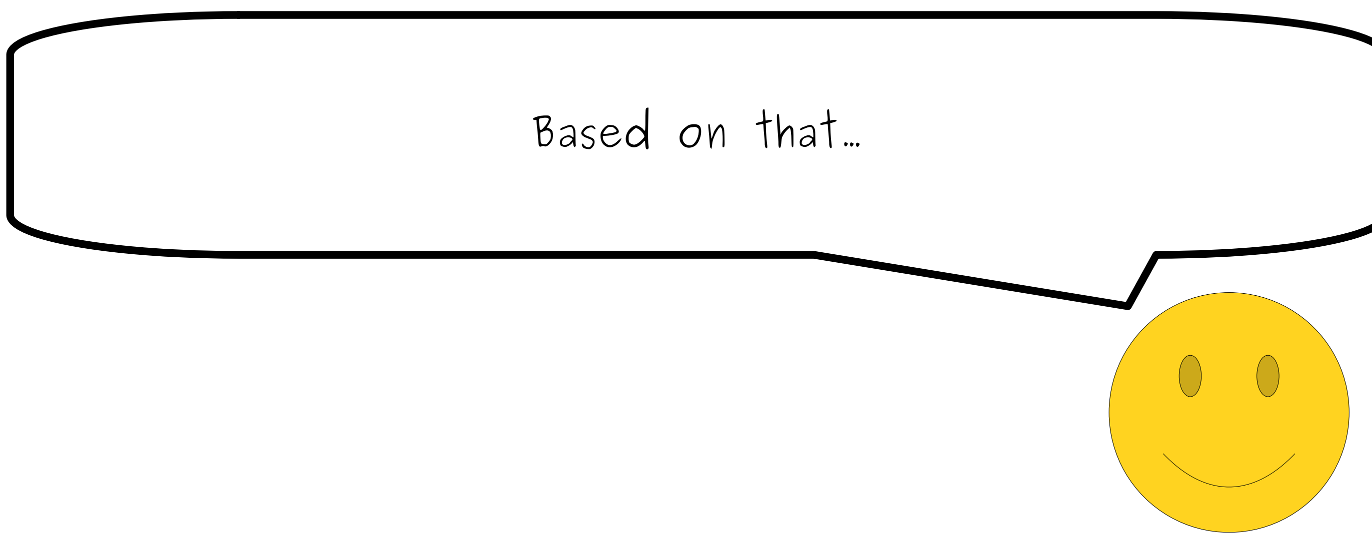
"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

Every orange cat is fluffy.

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

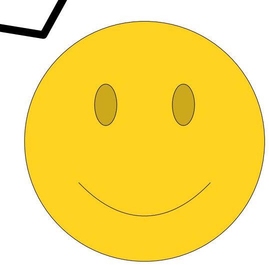
$\exists x.\ (P(x) \land \neg Q(x))$

∀x. (*x is an orange cat* → *x is fluffy*)

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

...we can start adding in a bit of structure to our first-order logic formula.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

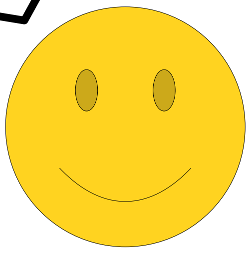
$$\exists x. (P(x) \land \neg Q(x))$$

$$\forall x. (\textit{x is an orange cat} \to \textit{x is fluffy})$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

From here, our goal is to keep replacing the remaining English statements in the formula with something in first-order logic that says the same thing.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

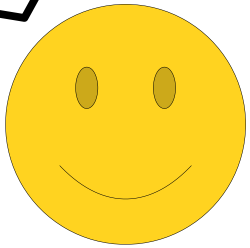
$\exists x. (P(x) \land \neg Q(x))$

∀x. (*x is an orange cat* → $\boxed{\textit{x is fluffy})}$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

For example, this part of the formula is easy to translate…



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

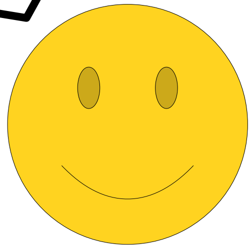
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ ($*x is an orange cat* $\to$ $\boxed{\textit{x is fluffy}}$$)$

Available Predicates:

Orange(x)
Cat(x)
$\boxed{\textit{Fluffy(x)}}$

…because we have a predicate that directly expresses this idea!



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

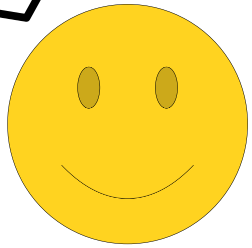
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (\textit{x is an orange cat} \to Fluffy(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

So let's go and snap that predicate in there. Progress!



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

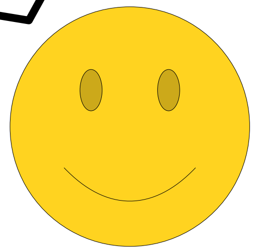
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (x \textit{ is an orange cat} \to \mathit{Fluffy}(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

So what about the rest of the formula? How do we express the idea that x is an orange cat?



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

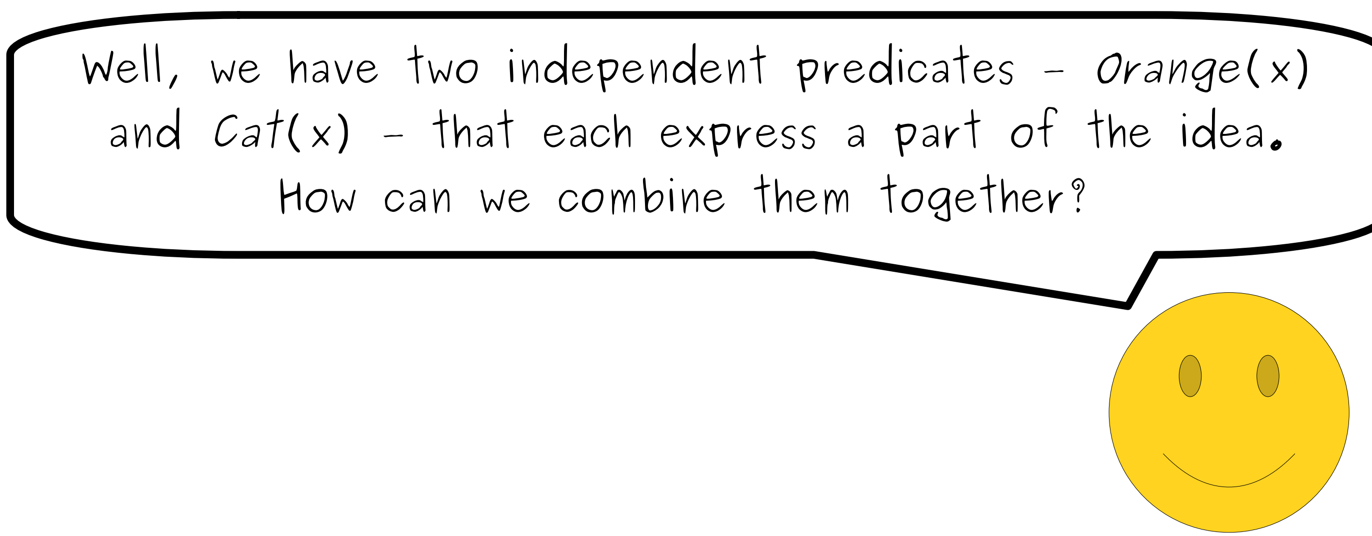
“Some *P*s aren't *Q*s.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (\textit{x is an orange cat} \to \mathit{Fluffy}(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

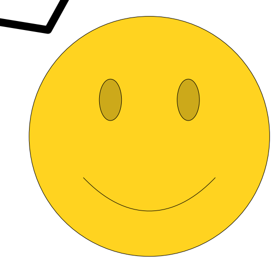
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (\textit{x is an orange cat} \to \mathit{Fluffy}(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Let's begin by seeing how <u>not</u> to do this.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

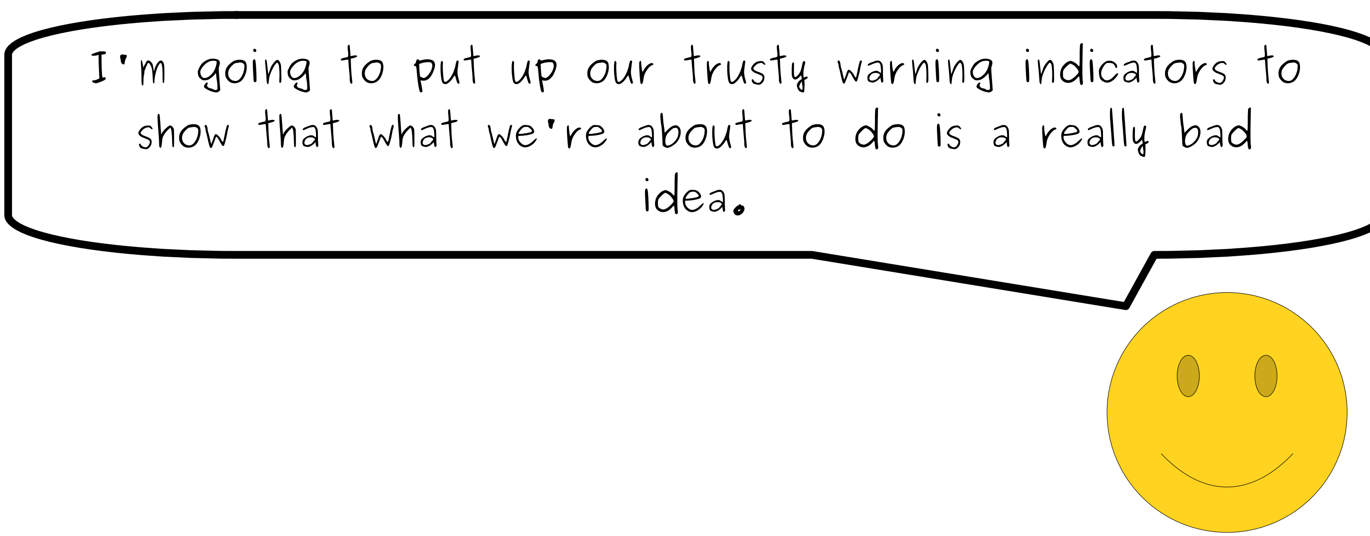
"Some Ps aren't Qs."

$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (\textit{x is an orange cat} \to \mathit{Fluffy}(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

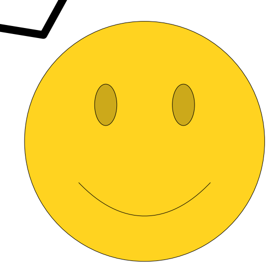
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(Cat(x)) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Here's something common we see people do that doesn't work,



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

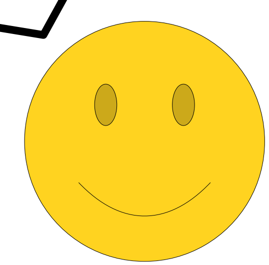
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(Cat(x)) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

This superficially looks like it works correctly – it seems like it's saying that x is a cat that's orange.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

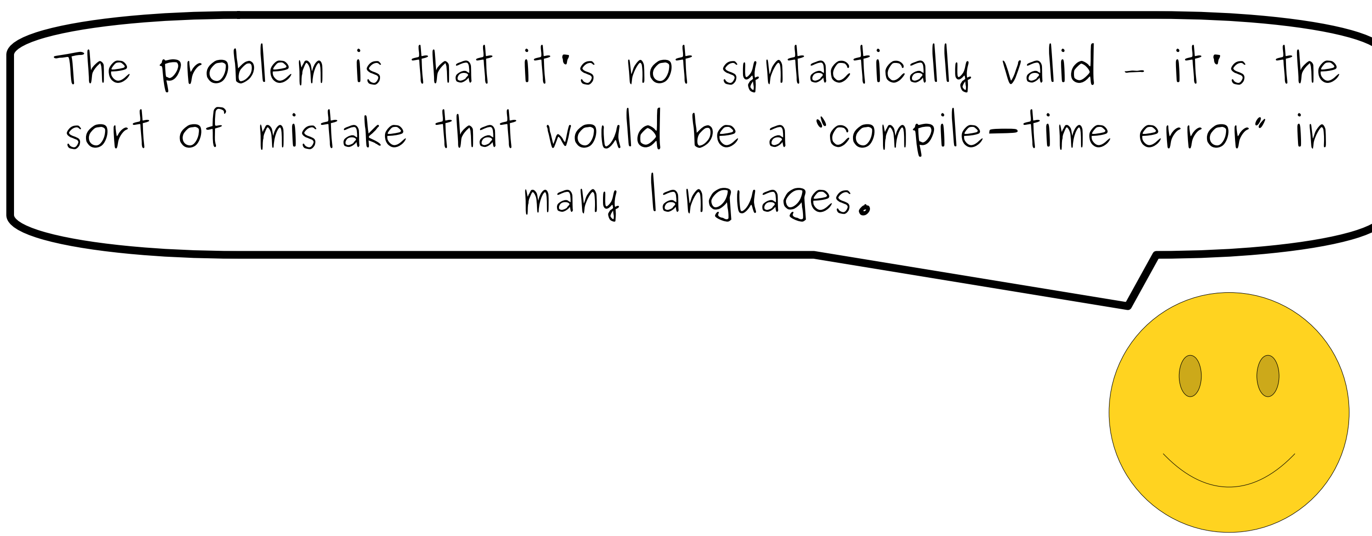
"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

$$\forall x.\ (Orange(Cat(x)) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

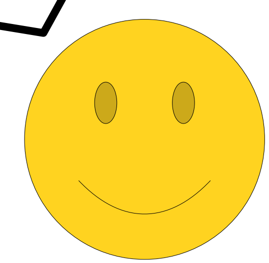
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(Cat(x)) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

The reason this doesn't work is that *Orange* and *Cat* are predicates – they take in objects and produce either true or false.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

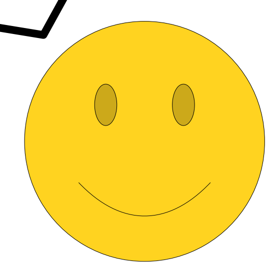
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(\underbrace{Cat(x)}_{\texttt{bool}}) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

This means that the statement *Cat*(x) evaluates to either "true" or "false." Intuitively, it takes in an object and returns a boolean.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

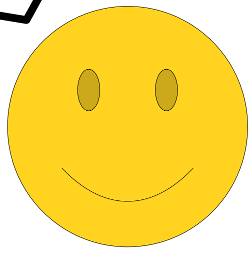
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(\underbrace{Cat(x)}_{\texttt{bool}}) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

The problem is that *Orange* expects that it will take in an object and return a boolean – but it's not being provided an object as input!



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

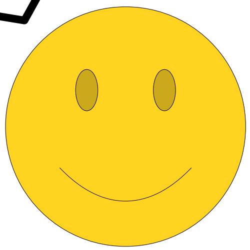
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(\underbrace{Cat(x)}_{\texttt{bool}}) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

This is the first-order logic equivalent of a type error.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

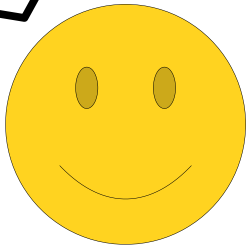
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(\underbrace{Cat(x)}_{\texttt{bool}}) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

So even though this might at first glance seem right, it's not actually legal… so we're going to have to find some other way of expressing this idea!



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

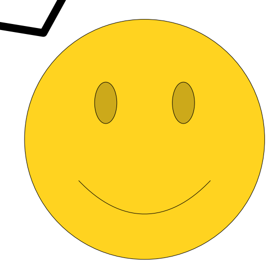
$\exists x. (P(x) \land \neg Q(x))$

$\forall x. (\textit{x is an orange cat} \to \mathit{Fluffy}(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Let's revert back to what we had before.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

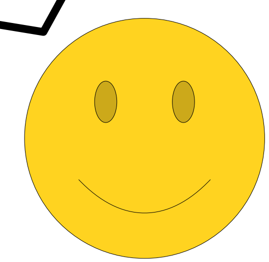
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (x \textit{ is an orange cat} \to \mathit{Fluffy}(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

We're trying to express the idea that x is an orange cat.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

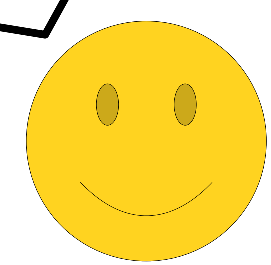
$\exists x. (P(x) \land \neg Q(x))$

$\forall x.$ (*x is orange and x is a cat* $\to Fluffy(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

If you think about it, that's the same as saying that x is an orange and that x is a cat.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

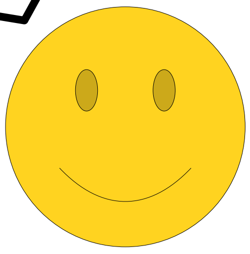
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.$ (*x is orange and x is a cat* $\to$ *Fluffy*(x))

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

This is something that's a lot easier to translate into first-order logic.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

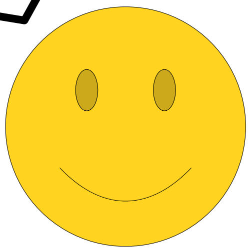
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (x \textit{ is orange} \land x \textit{ is a cat} \to \mathit{Fluffy}(x))$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

The “and,” for example, just becomes a ∧ connective.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

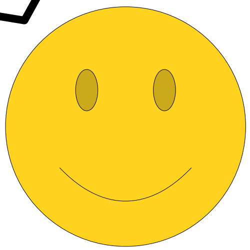
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(x) \land Cat(x) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

And, given the predicates we have available, we can translate the left and right halves of that expression directly into first-order logic.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

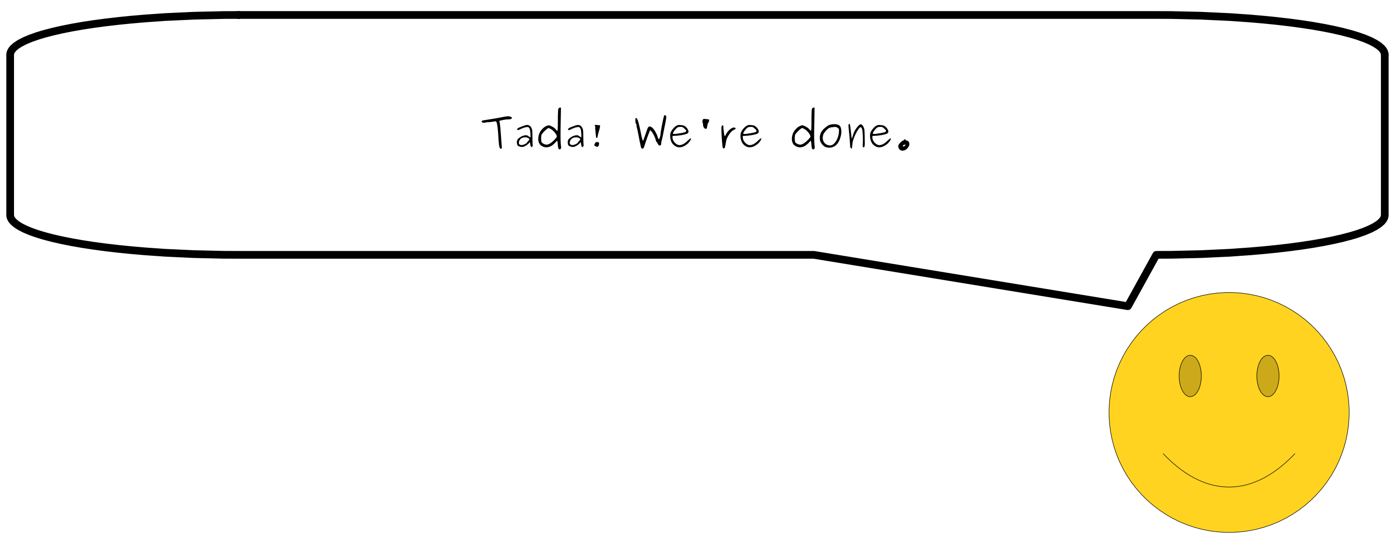
"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\forall x.\ (Orange(x) \land Cat(x) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

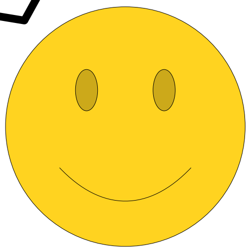
$\exists x.\ (P(x) \land \neg Q(x))$

$$\forall x.\ (Orange(x) \land Cat(x) \to Fluffy(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)

Although this wasn't a particularly complicated example, especially compared to what we did in class the other day, I do think it's helpful to see where it comes from, since we walked through it step-by-step.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

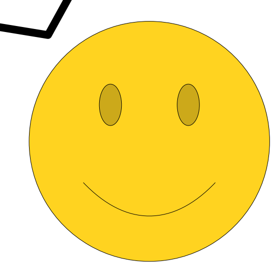
$$\forall x.\ (P(x) \to \neg Q(x))$$

"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

Available Predicates:

Orange(x)
Cat(x)
Fluffy(x)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

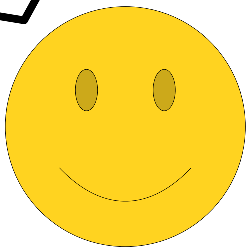
"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

Available Predicates:

Corgi(x)
Person(x)
Loves(x, y)

Let's change our available set of predicates so that we can talk about whether something's a corgi, whether something's a person, and whether one thing x loves another thing y.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

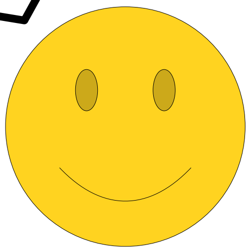
$\exists x. (P(x) \land \neg Q(x))$

There's a corgi that loves everyone.

Available Predicates:

Corgi(x)
Person(x)
Loves(x, y)

With these predicates, let's see how to translate this statement into first-order logic.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

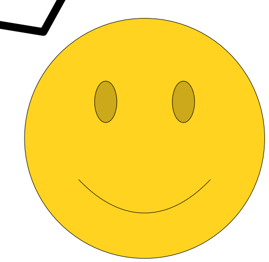
$\exists x. (P(x) \land \neg Q(x))$

There's a corgi that loves everyone.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Again, we can start off by asking what kind of statement this is. What exactly is it that we're talking about here?



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

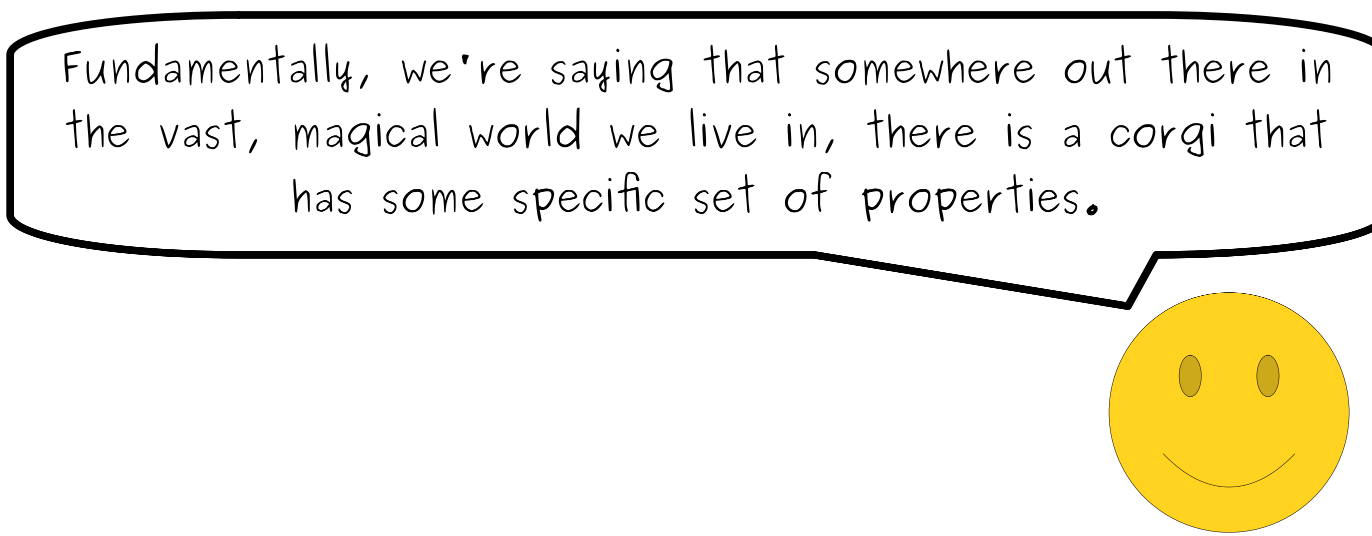
“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$

There's a corgi that loves everyone.

Available Predicates:

Corgi(x)
Person(x)
Loves(x, y)



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

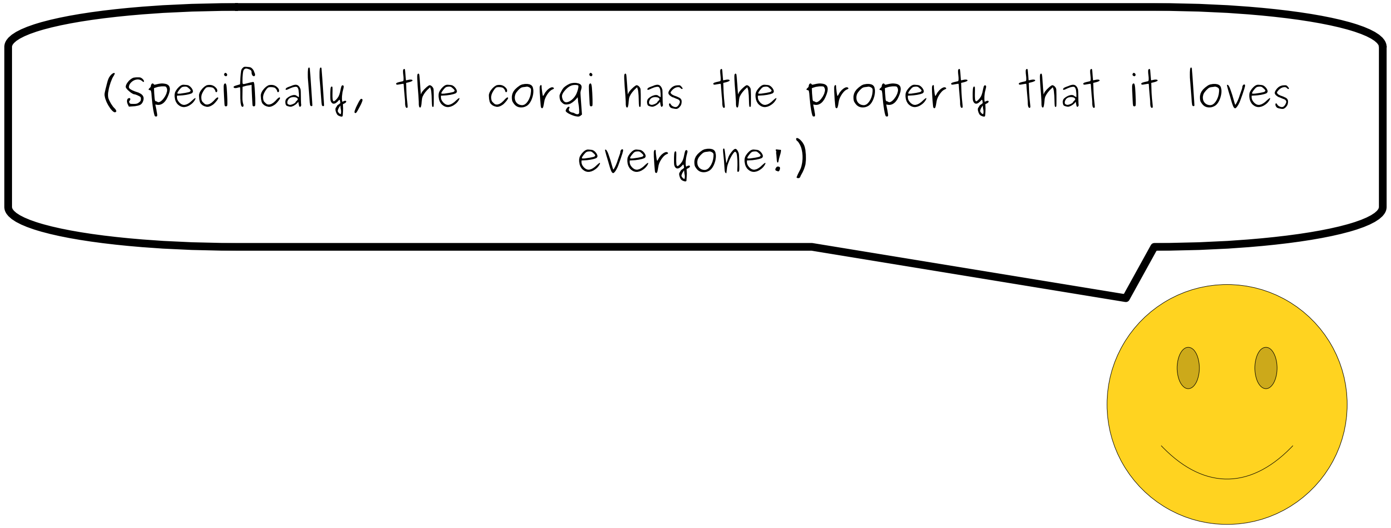
"Some Ps aren't Qs."

$$\exists x.\ (P(x) \land \neg Q(x))$$

There's a corgi that loves everyone.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

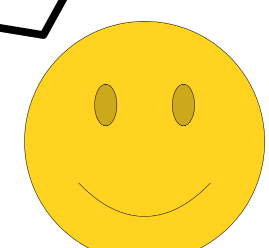
$\exists x.\ (P(x) \land \neg Q(x))$

There's a corgi that loves everyone.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

That statement looks a lot like this one over here – we're saying that some corgis happen to love everyone.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

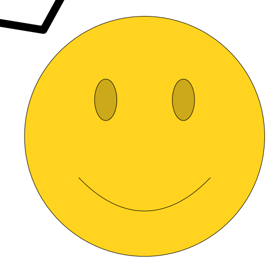
$$\exists x.\ (P(x) \land \neg Q(x))$$

$\exists x.$ (*x is a corgi* $\land$ *x loves everyone*)

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

We'll partially translate our statement by using that general pattern.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

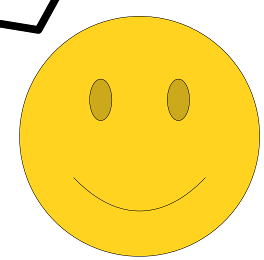
$\exists x. (P(x) \land \neg Q(x))$

$\exists x.$ (*x is a corgi* $\land$ *x loves everyone*)

Available Predicates:

Corgi(x)
Person(x)
Loves(x, y)

As before, we'll continue to make incremental progress translating bits and pieces of this formula until we arrive at the final result.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

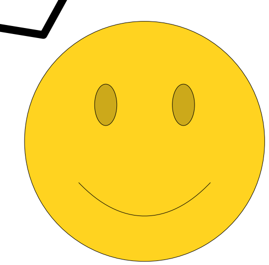
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land$ *x loves everyone*$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

For example, we can directly express the idea that x is a corgi, so let's go do that.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

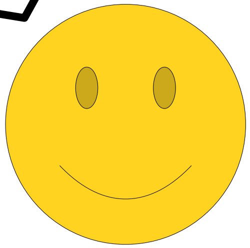
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land$ *x loves everyone*$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Now, we have to think about how to translate the statement "x loves everyone."



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

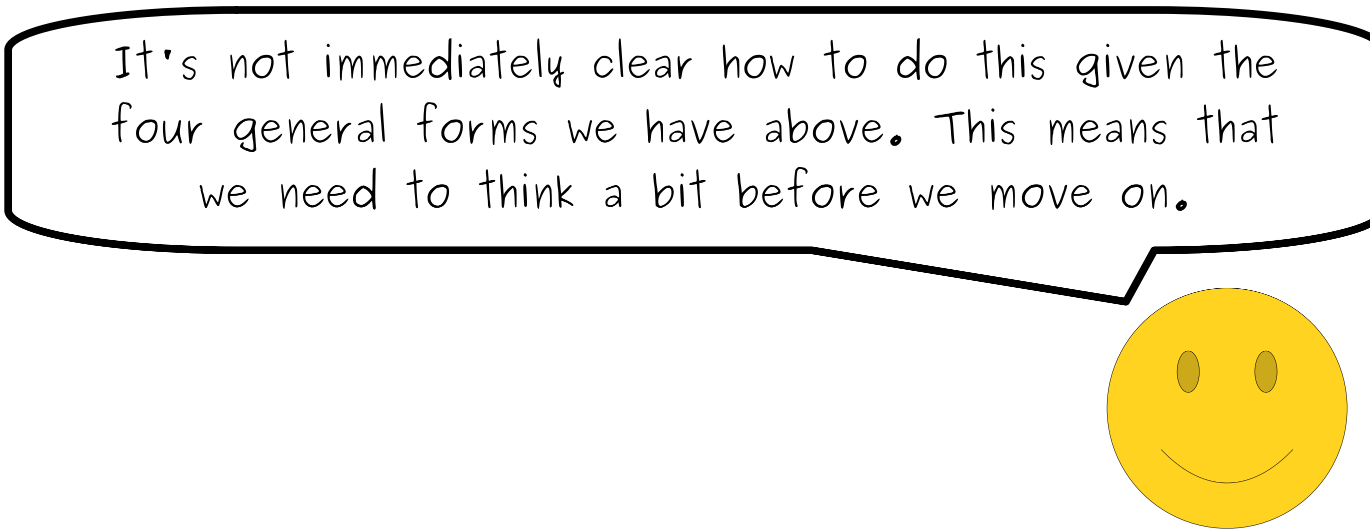
“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (\mathit{Corgi}(x) \land$ *x loves everyone*$)$

Available Predicates:

$\mathit{Corgi}(x)$
$\mathit{Person}(x)$
$\mathit{Loves}(x, y)$



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

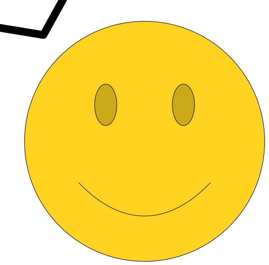
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land$ *x loves everyone*$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

When translating statements like these, it sometimes helps to introduce variables representing names for things.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

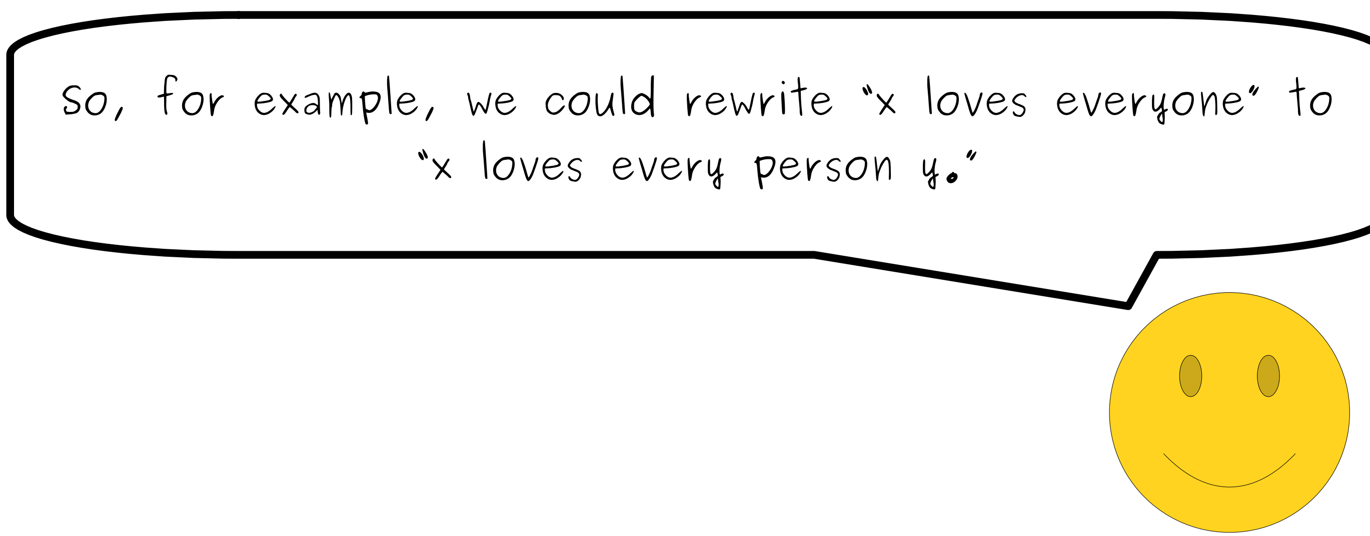
“Some Ps aren't Qs.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\exists x.\ (Corgi(x) \land \text{x loves every person y})$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

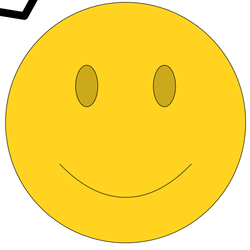
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land$ *x loves every person y*$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

This is suggesting that we're probably going to want to use one of the templates on the left, since this statement says something about every person y.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

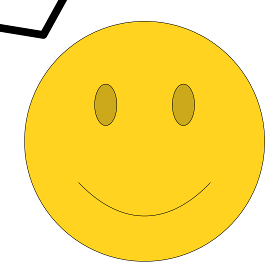
$\exists x. (P(x) \land \neg Q(x))$

$\exists x. (Corgi(x) \land \textit{x loves every person y})$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

To see exactly how this matches, we might want to rewrite this blue part to focus more on what we're saying about y.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

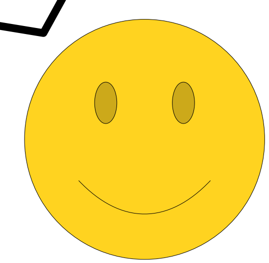
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\exists x.\ (\mathit{Corgi}(x) \land \textit{every person y is loved by x})$$

Available Predicates:

$\mathit{Corgi}(x)$
$\mathit{Person}(x)$
$\mathit{Loves}(x, y)$

When I was learning how to write, I remember being told that the passive voice should not be used. But sometimes, like in this case, it's actually helpful for exposing the structure of what's going on – every person y is loved by x.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

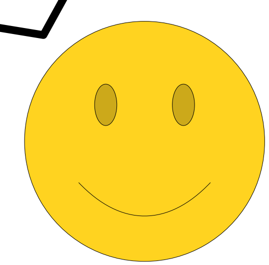
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land \textit{every person y is loved by x})$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

If we write things this way, it becomes a bit clearer that this statement matches this first general pattern. Let's go and apply it!



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

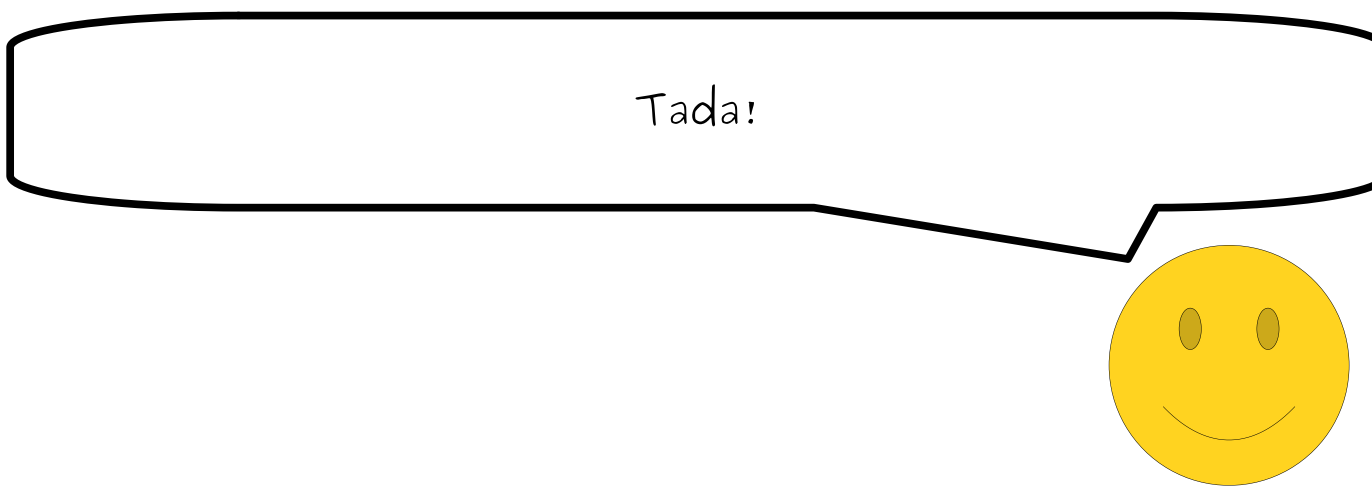
“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x)\ \land$
$\quad \forall y.\ ($*y is a person* $\to$ *y is loved by x*$)$
$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

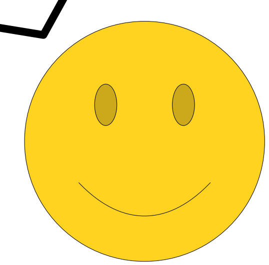
$\exists x.\ (P(x) \land \neg Q(x))$

$$
\begin{array}{l}
\exists x.\ (Corgi(x)\ \land \\
\qquad \forall y.\ (\text{y is a person} \to \text{y is loved by x}) \\
)
\end{array}
$$

You'll notice that I've written this part of the formula on the next line and indented it. It's extremely useful to structure the formula this way – it shows what's nested inside of what and clarifies the scope of the variables involved. While it's not strictly required that you do this in your own translations, we highly recommend it!

Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

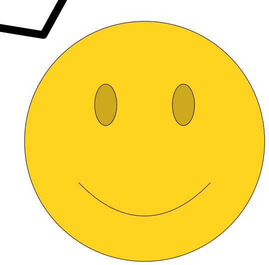
$\exists x.\ (P(x) \land \neg Q(x))$

$\exists x.\ (Corgi(x) \land$
$\quad \forall y.\ ($*y is a person* $\to$ *y is loved by x*$)$
$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Now that we're here, we can do the finishing touches of translating this statement by replacing these blue parts with predicates!



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

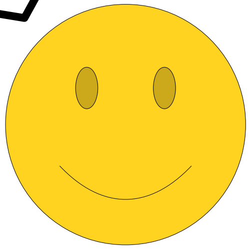
$\exists x.\ (P(x) \land \neg Q(x))$

$$
\begin{aligned}
&\exists x.\ (Corgi(x)\ \land \\
&\qquad \forall y.\ (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

That gives this, our final statement.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

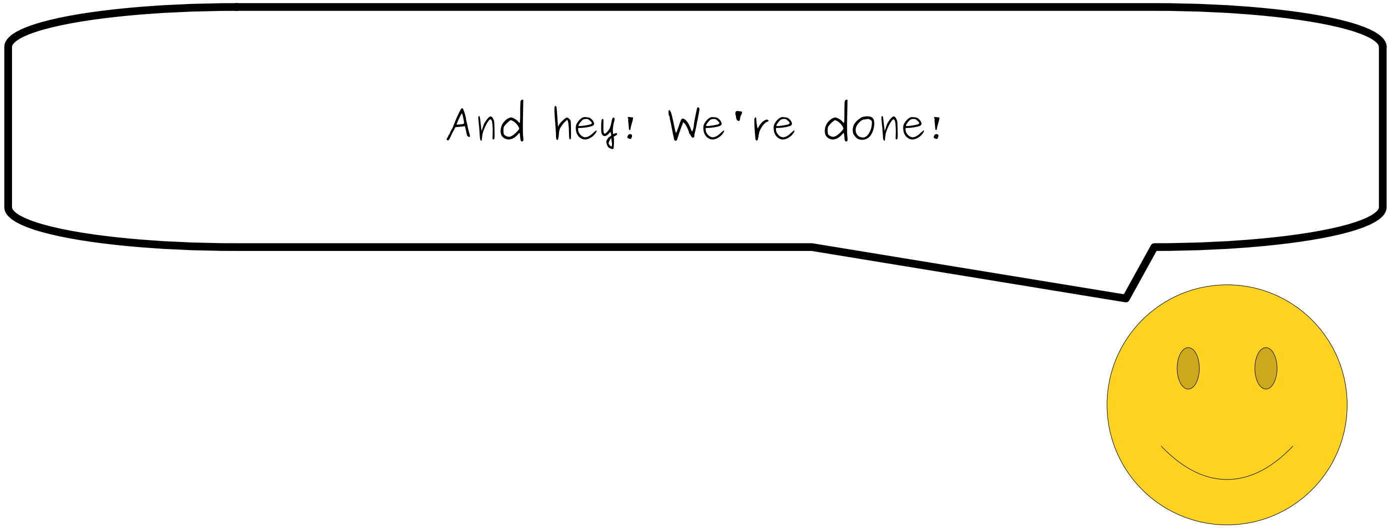
"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\exists x.\ (Corgi(x) \land \\ \quad \forall y.\ (Person(y) \to Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

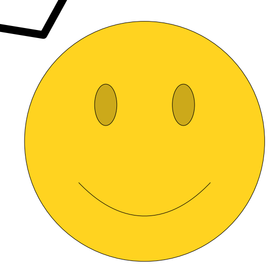
$\exists x. (P(x) \land \neg Q(x))$

$$
\begin{aligned}
&\exists x. (Corgi(x) \land \\
&\qquad \forall y. (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Before we move on, let's pause and look at the formula that we came up with.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

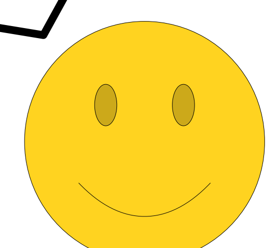
$\exists x.\ (P(x) \land \neg Q(x))$

$$
\begin{aligned}
&\exists x.\ (\mathit{Corgi}(x) \land \\
&\qquad \forall y.\ (\mathit{Person}(y) \to \mathit{Loves}(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Corgi}(x)$
$\mathit{Person}(x)$
$\mathit{Loves}(x, y)$

Just as we can use the above patterns to translate the original statement into logic, we can use those same patterns to translate this *out of* logic and back into English (or any language of your choice, really!)



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

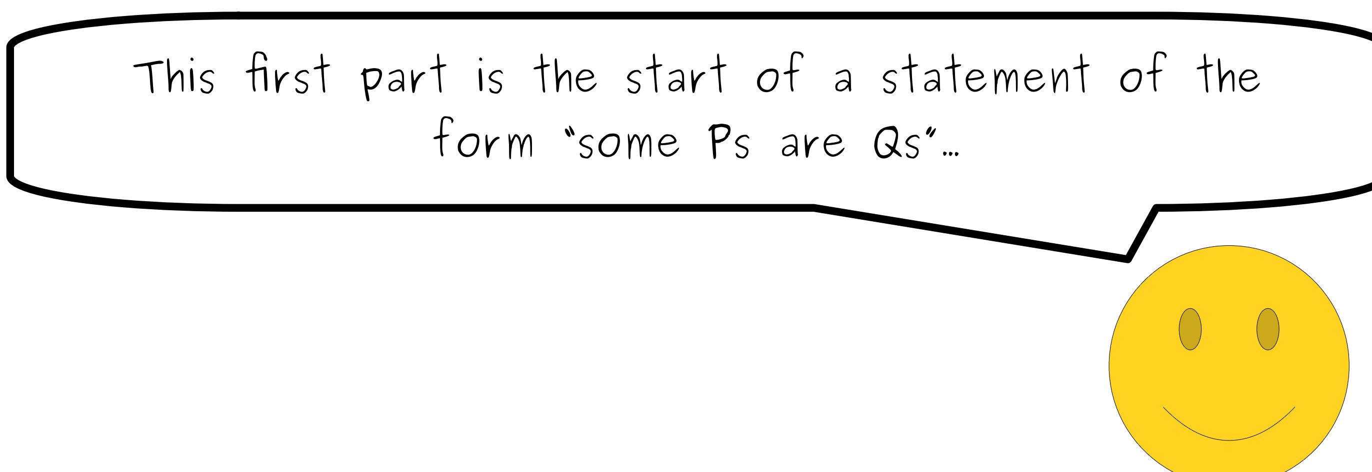
“Some *P*s aren't *Q*s.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\begin{aligned}&\exists x.\ (Corgi(x) \land\\&\qquad \forall y.\ (Person(y) \to Loves(x, y))\\&)\end{aligned}$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\exists x.\ (Corgi(x) \land$$
$$\quad \forall y.\ (Person(y) \to Loves(x, y))$$
$$)$$

There is a corgi...

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x. (P(x) \land \neg Q(x))$

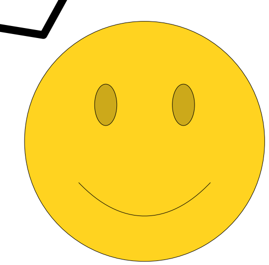
$$
\begin{array}{l}
\exists x. (Corgi(x) \land \\
\qquad \forall y. (Person(y) \to Loves(x, y)) \\
)
\end{array}
$$

There is a corgi...

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

This part of the statement starts off a statement of the form "all Ps are Qs"...



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

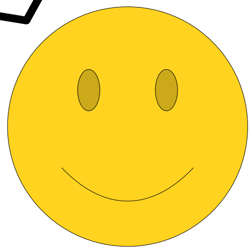
$$
\begin{aligned}
&\exists x.\ (Corgi(x) \land \\
&\quad \forall y.\ (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

There is a corgi
that every person…

Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)

…so we can continue our translation like this.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x. (P(x) \land \neg Q(x))$

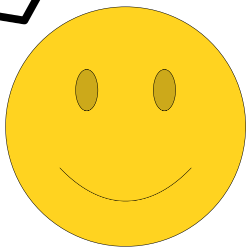
$$
\begin{array}{l}
\exists x. (Corgi(x) \land \\
\quad \forall y. (Person(y) \to Loves(x, y)) \\
)
\end{array}
$$

There is a corgi
that every person
is loved by.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

The last bit is a predicate, so we can just read it off.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

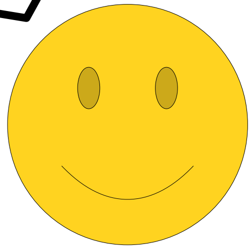
$$
\begin{aligned}
&\exists x.\ (Corgi(x) \land \\
&\qquad \forall y.\ (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

There is a corgi
that every person
is loved by.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

We now have a (grammatically awkward) but correct translation of our logic statement back into English.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

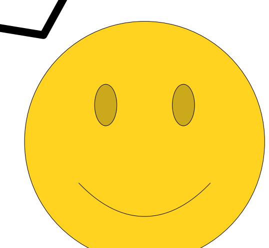
$$
\begin{array}{l}
\exists x.\ (Corgi(x) \land \\
\qquad \forall y.\ (Person(y) \to Loves(x, y)) \\
)
\end{array}
$$

"There is a corgi that loves everyone."

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

With a bit of English rewriting, we can get back to our original statement. Nifty! Looks like we got it right!



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

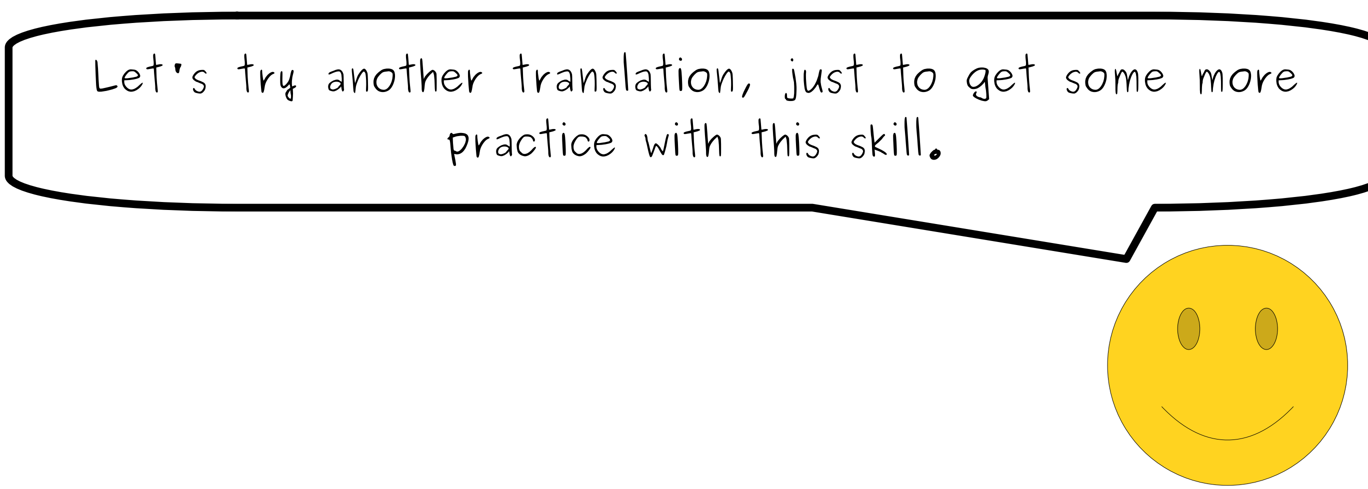
$$\forall x.\ (P(x) \to \neg Q(x))$$

“Some *P*s aren't *Q*s.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

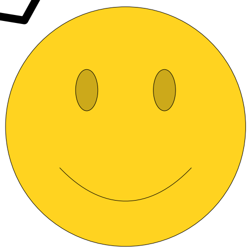
$$\exists x.\ (P(x) \land \neg Q(x))$$

Everybody loves at least one corgi.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

How might we translate this statement?



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

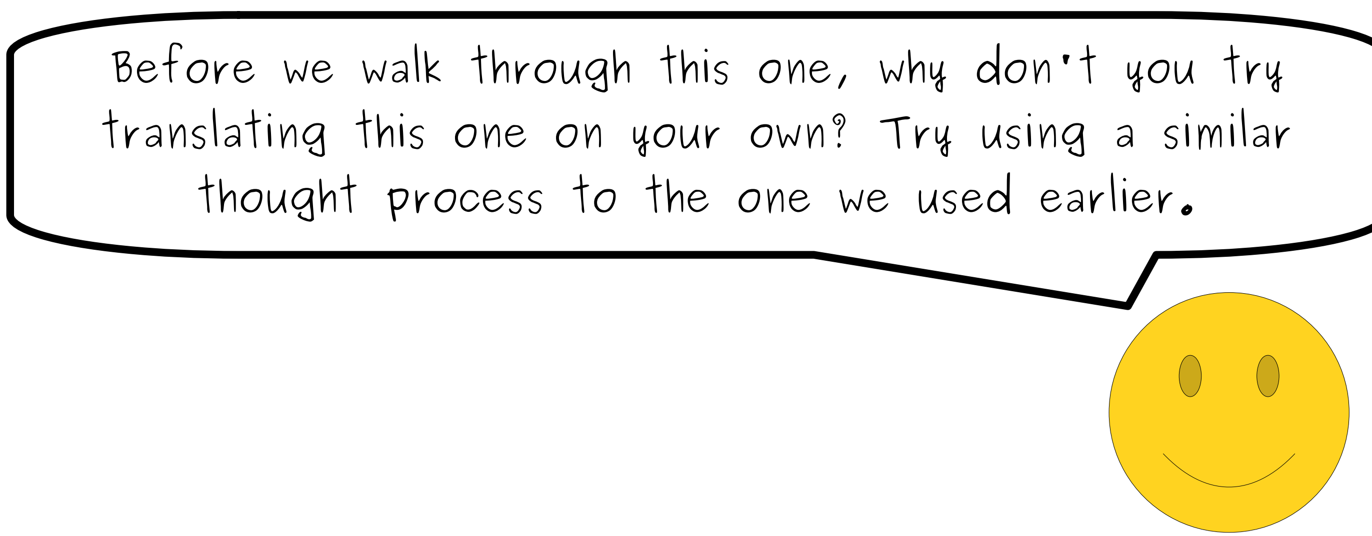
“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$

Everybody loves at least one corgi.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

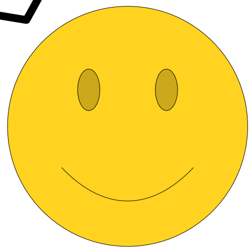
$\exists x.\ (P(x) \land \neg Q(x))$

Everybody loves at least one corgi.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Did you actually try this? Because if you didn't, you really should. Like, seriously.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

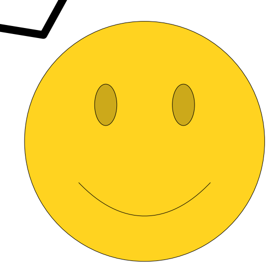
$\exists x. \ (P(x) \land \neg Q(x))$

Everybody loves at least one corgi.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

So you translated the statement on your own? Great!
Let's do this one together.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

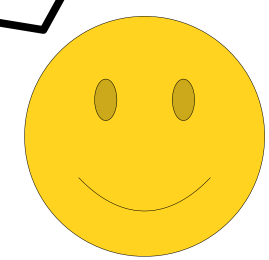
$$\exists x.\ (P(x) \land \neg Q(x))$$

Everybody loves at least one corgi.

Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)

First, we need to start off by thinking about what exactly this statement says.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

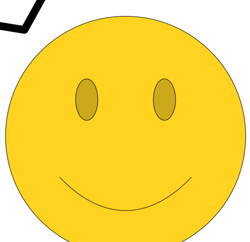
$\exists x.\ (P(x) \land \neg Q(x))$

Everybody loves at least one corgi.

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

This says "if you pick any person, you'll find that there's some corgi that they like."



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

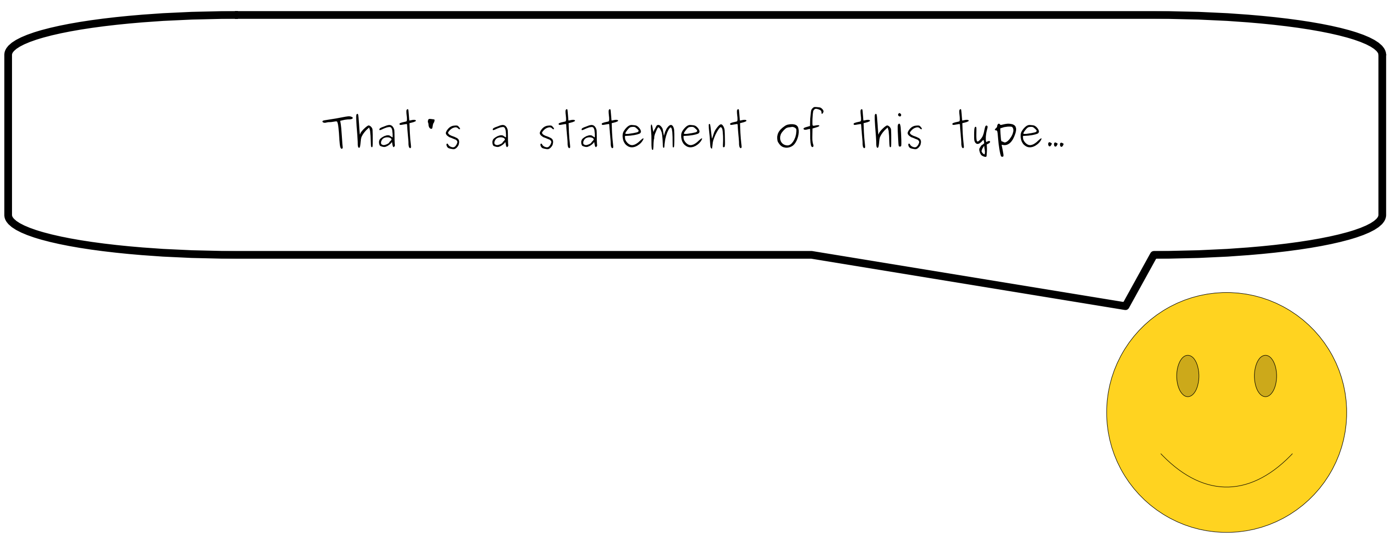
“Some *P*s aren't *Q*s.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

Everybody loves at least one corgi.

Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)



“All Ps are Qs.”

$\boxed{\forall x.\ (P(x) \to Q(x))}$

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

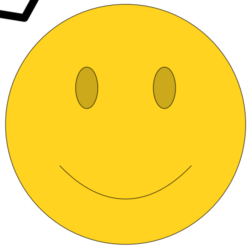
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.$ (*x is a person* → *x loves at least one corgi*)

Available Predicates:

Corgi(x)
Person(x)
Loves(x, y)

…so we can make some initial progress like this.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

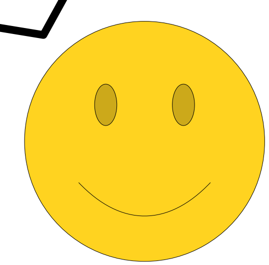
$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (Person(x) \to$ *x loves at least one corgi*$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

From here, we can translate the “x is a person” part directly into first-order logic.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (Person(x) \to$

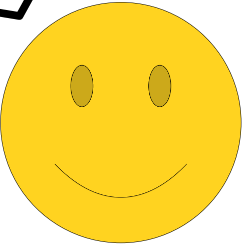
x loves at least one corgi

$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Now, we have to figure out how to translate that last part.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

$$\exists x.\ (P(x) \land \neg Q(x))$$

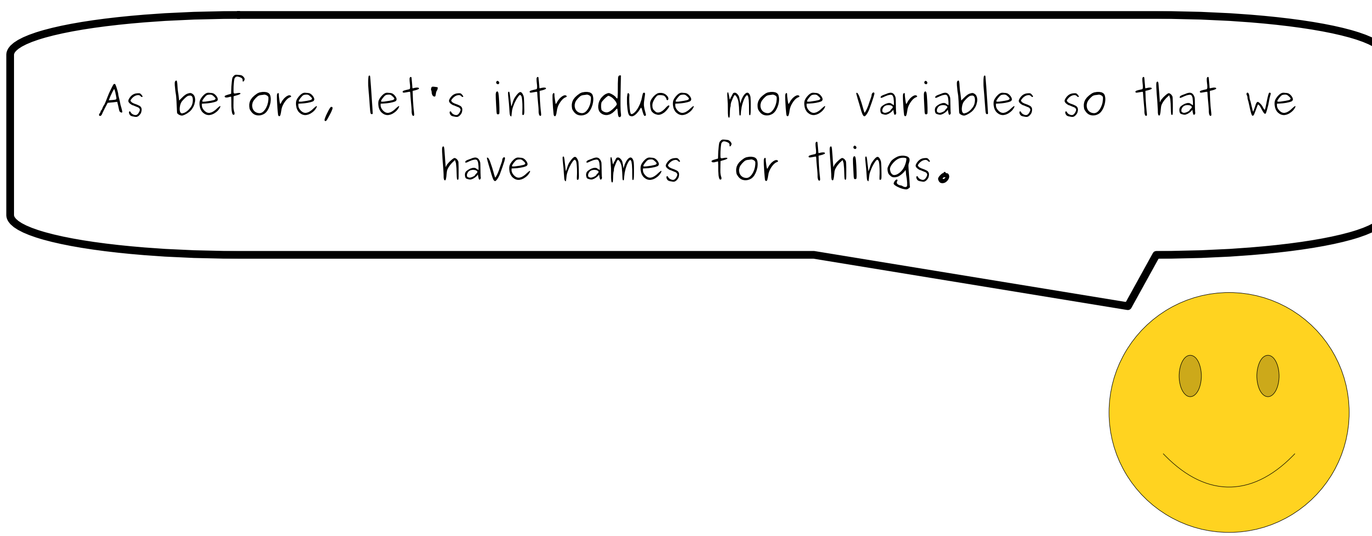
$\forall x.\ (Person(x) \to$

x loves at least one corgi y

$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

$\exists x. (P(x) \land \neg Q(x))$

$\forall x. (Person(x) \to$

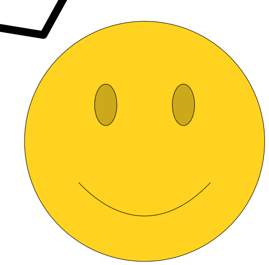
there is a corgi y that is loved by x

$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

And, as before, let's fiddle around with the verb structure to make clearer what kind of statement this is.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

$\exists x. (P(x) \land \neg Q(x))$

$\forall x. (Person(x) \to$

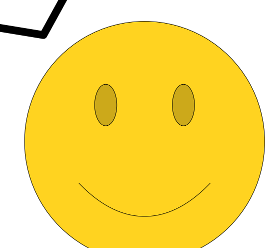
there is a corgi y that is loved by x

$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

From here it's (hopefully) a bit clearer that this is a "some P's are Q's" statement – some corgis happen to be loved by person x.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

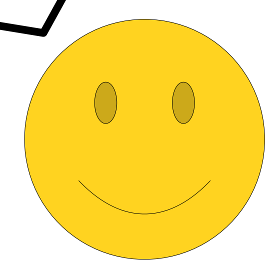
$$\exists x.\ (P(x) \land \neg Q(x))$$

$\forall x.\ (Person(x) \to$
$\quad \exists y.\ (y \text{ is a corgi} \land y \text{ is loved by } x)$
$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

We can make more progress on our translation by using that template.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

$\forall x.\ (Person(x) \to$

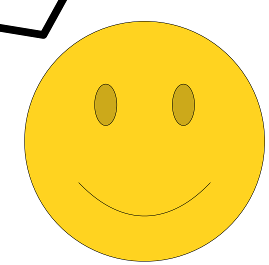
$\quad \exists y.\ ($*y is a corgi* $\land$ *y is loved by x*$)$

$)$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

At this point we just need to put in the finishing touches and rewrite the blue parts using predicates…



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

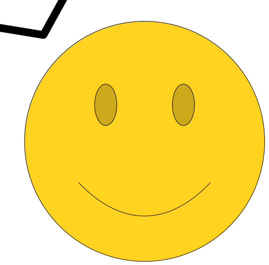
$\exists x.\ (P(x) \land \neg Q(x))$

$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\qquad \exists y.\ (Corgi(y) \land Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

…like this! Tada! We're done.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

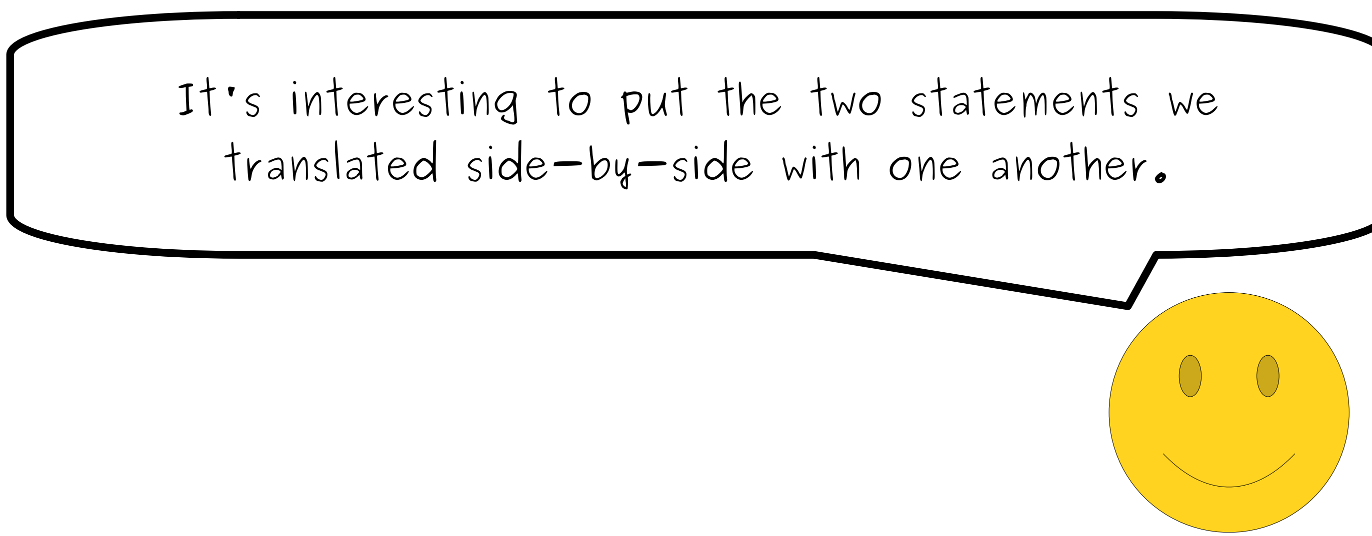
$\exists x. (P(x) \land \neg Q(x))$

$$
\begin{array}{l}
\exists x. (Corgi(x) \land \\
\quad \forall y. (Person(y) \to Loves(x, y)) \\
)
\end{array}
$$

$$
\begin{array}{l}
\forall x. (Person(x) \to \\
\quad \exists y. (Corgi(y) \land Loves(x, y)) \\
)
\end{array}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

$\exists x. (P(x) \land \neg Q(x))$

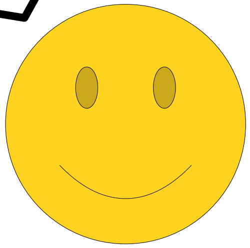
$$
\begin{aligned}
&\exists x. (Corgi(x) \land \\
&\quad \forall y. (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x. (Person(x) \to \\
&\quad \exists y. (Corgi(y) \land Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

These statements have a lot of similarities, though they're clearly different in a number of ways.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

$$\exists x. \ (P(x) \land \neg Q(x))$$

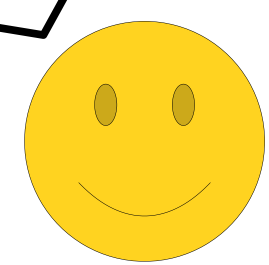
$$\exists x. \ (Corgi(x) \land \\ \quad \forall y. \ (Person(y) \to Loves(x, y)) \\ )$$

$$\forall x. \ (Person(x) \to \\ \quad \exists y. \ (Corgi(y) \land Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

One major difference between these two is the order in which the quantifiers appear. The first has them in the order ∃∀, and the second has them in the order ∀∃.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

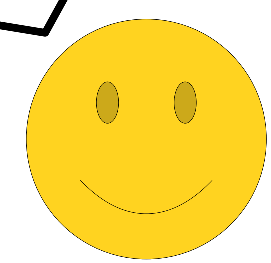
$$
\begin{aligned}
&\exists x.\ (Corgi(x) \land \\
&\quad \forall y.\ (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Something I'd really like to stress is that, when we did these translations, we didn't just magically "guess" that we needed those particular quantifiers and that they would be in these orders.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

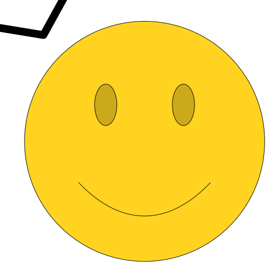
$$
\begin{array}{l}
\exists x.\ (Corgi(x)\ \land \\
\quad \forall y.\ (Person(y) \to Loves(x, y)) \\
)
\end{array}
$$

$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\
)
\end{array}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Instead, we started off with the original statement and incrementally translated it top-down, only adding in the quantifiers when we needed them.



"All Ps are Qs."

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

"Some Ps are Qs."

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

"No Ps are Qs."

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

"Some Ps aren't Qs."

$$\exists x. (P(x) \land \neg Q(x))$$

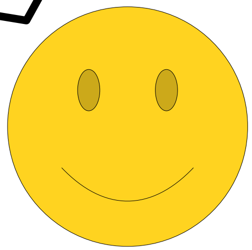
$$\exists x. (Corgi(x) \land \\ \quad \forall y. (Person(y) \to Loves(x, y)) \\ )$$

$$\forall x. (Person(x) \to \\ \quad \exists y. (Corgi(y) \land Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

One of the biggest mistakes we see people make when learning first-order logic for the first time is trying to write the whole statement in a single go, adding in quantifiers somewhat randomly to try to get things to work.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$\exists x.\ (P(x) \land \neg Q(x))$

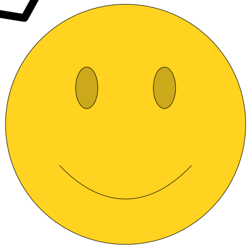
$$
\begin{aligned}
&\exists x.\ (Corgi(x) \land \\
&\qquad \forall y.\ (Person(y) \to Loves(x, y)) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\qquad \exists y.\ (Corgi(y) \land Loves(x, y)) \\
&)
\end{aligned}
$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Don't do that! It's really, really hard to get right on a first try.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

$\exists x.\ (P(x) \land \neg Q(x))$

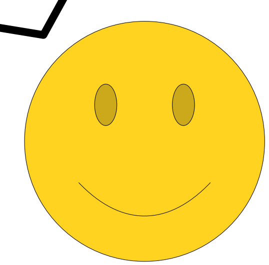
$$\exists x.\ (Corgi(x) \land \\ \quad \forall y.\ (Person(y) \to Loves(x, y)) \\ )$$

$$\forall x.\ (Person(x) \to \\ \quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Instead, use the approach we outlined here. Work slowly, going one step at a time, and only adding in quantifiers when you need them.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

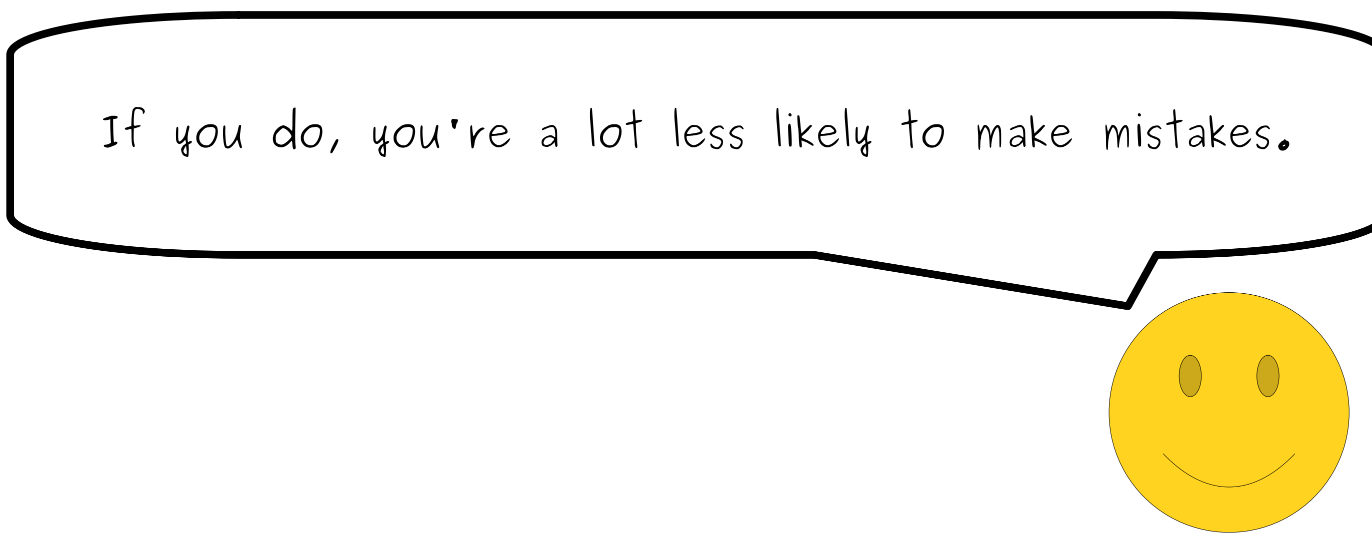
$$\exists x.\ (P(x) \land \neg Q(x))$$

$$\exists x.\ (Corgi(x) \land \\ \quad \forall y.\ (Person(y) \to Loves(x, y)) \\ )$$

$$\forall x.\ (Person(x) \to \\ \quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

"Some *P*s aren't *Q*s."

$$\exists x.\ (P(x) \land \neg Q(x))$$

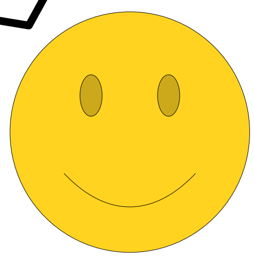
$$\begin{array}{l} \exists x.\ (Corgi(x) \land \\ \quad \forall y.\ (Person(y) \to Loves(x, y)) \\ ) \end{array}$$

$$\begin{array}{l} \forall x.\ (Person(x) \to \\ \quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\ ) \end{array}$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

Going back to our programming analogy, you can write a lot of similar programs that all use if statements and for loops.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

$$\exists x.\ (P(x) \land \neg Q(x))$$

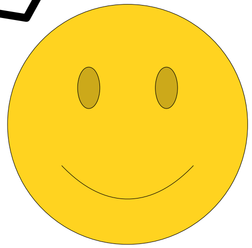
$$\exists x.\ (Corgi(x) \land \\ \quad \forall y.\ (Person(y) \to Loves(x, y)) \\ )$$

$$\forall x.\ (Person(x) \to \\ \quad \exists y.\ (Corgi(y) \land Loves(x, y)) \\ )$$

Available Predicates:

$Corgi(x)$
$Person(x)$
$Loves(x, y)$

However, you rarely write programs by just throwing a bunch of loops and if statements randomly and hoping that it'll work – because chances are, it won't.



“All Ps are Qs.”

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

“Some Ps are Qs.”

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

“No Ps are Qs.”

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

“Some Ps aren't Qs.”

$\exists x. (P(x) \land \neg Q(x))$

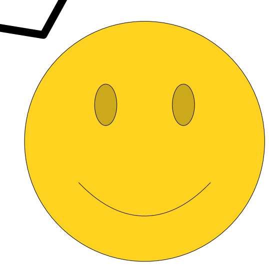
$$\begin{array}{l} \exists x. \ (Corgi(x) \land \\ \quad \forall y. \ (Person(y) \to Loves(x, y)) \\ ) \end{array}$$

$$\begin{array}{l} \forall x. \ (Person(x) \to \\ \quad \exists y. \ (Corgi(y) \land Loves(x, y)) \\ ) \end{array}$$

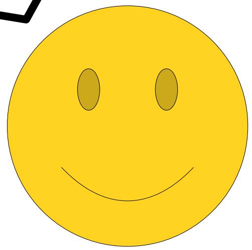
Available Predicates:

Corgi(*x*)
Person(*x*)
Loves(*x*, *y*)

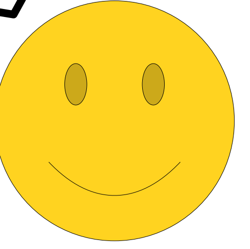
Instead, you work from the outside in – add in a loop when you need it, and if you need to nest an if statement, then you add it when you need it.



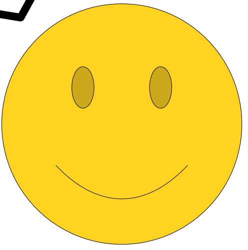
So at this point we've gotten some practice with the fundamentals of translation. Pretty much everything else we'll be doing is just more advanced applications of these concepts.



To give you a better sense of how these concepts scale up to more complicated examples, let's walk through some more complex statements and how to translate them. Along the way, you'll see a bunch of nifty tricks and insights that will help you out going forward.

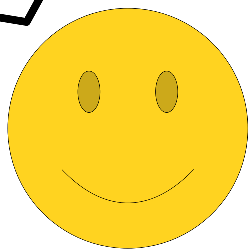


Let's start off by seeing how to talk about pairs of things.

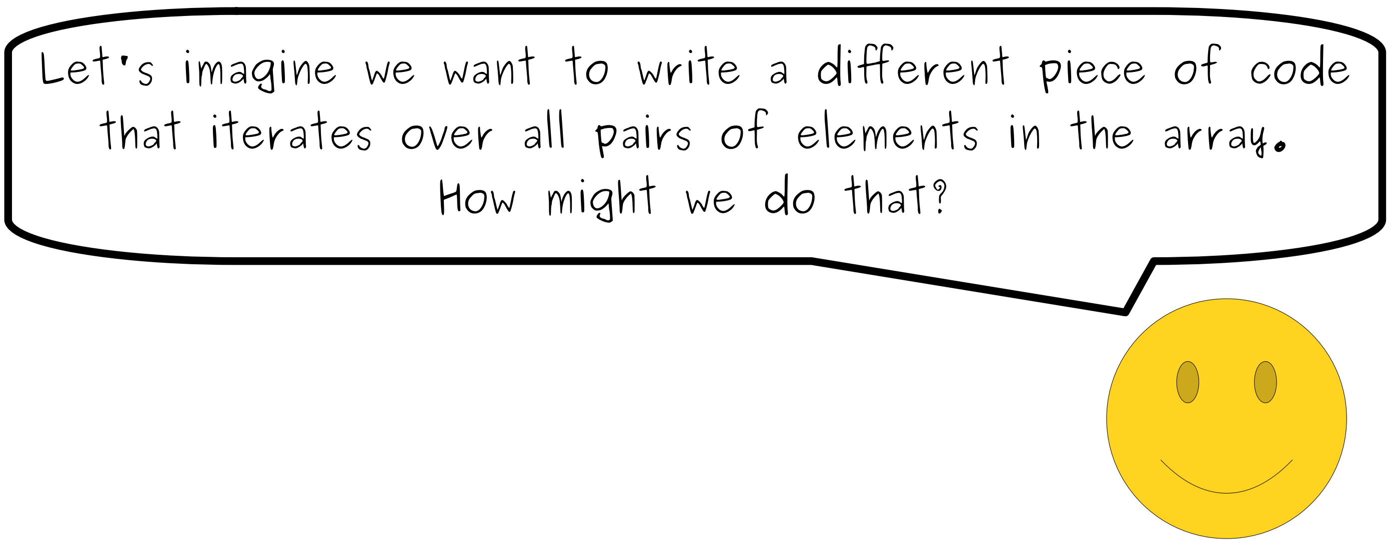


```
int sumOf(vector<int> elems) {
    int result = 0;
    for (int i = 0; i < elems.size(); i++) {
        result += elems[i];
    }
    return result;
}
```

Earlier, we talked about this Java code for iterating over all the elements of an array.

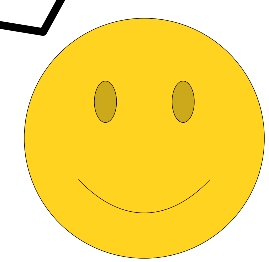


```
int sumOf(vector<int> elems) {
   int result = 0;
   for (int i = 0; i < elems.size(); i++) {
      result += elems[i];
   }
   return result;
}
```



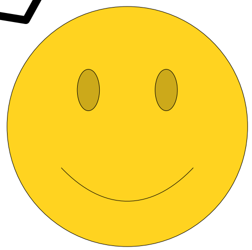
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Here's one possible option using the venerable "double-for-loop" pattern that you've probably gotten to know and love.

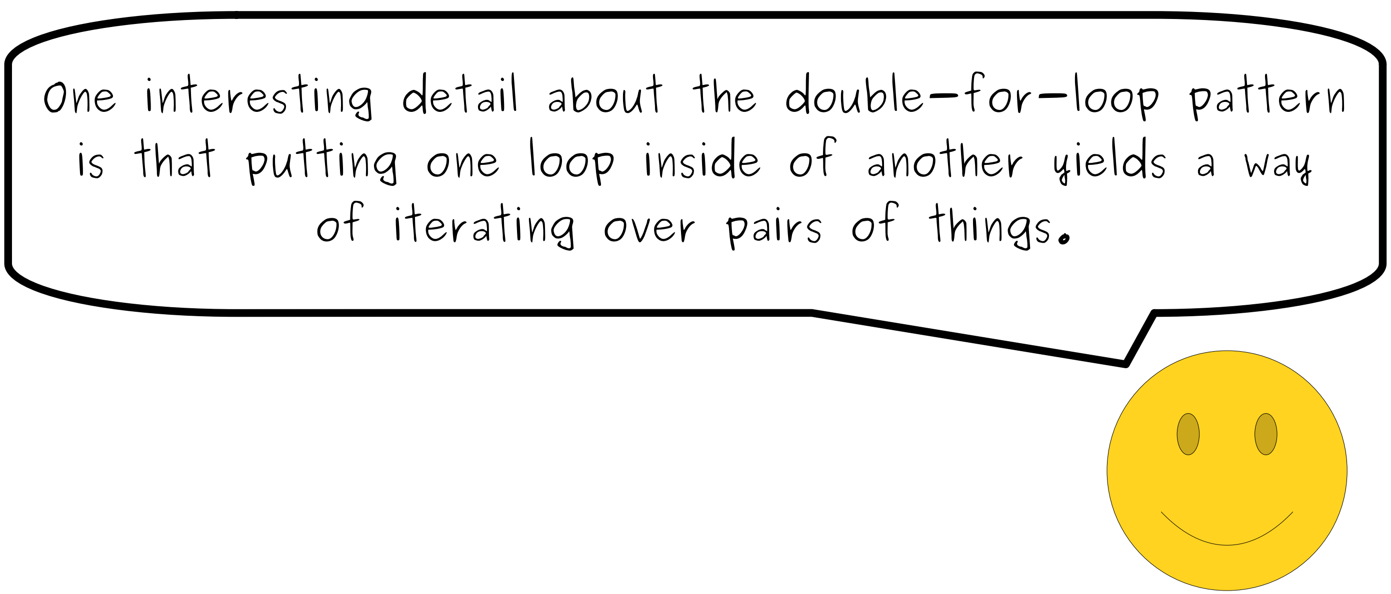


```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

As with the regular "loop over the elements of an array" loop, the double-for-loop is a programming idiom. Once you've seen it enough times, you just know what it means and don't have to think too much about it.

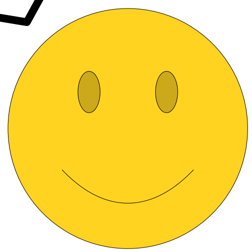


```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```



```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Turns out, we can adapt this idea to work in first-order logic as well!

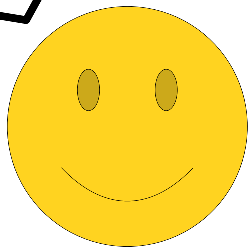


```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Let's imagine that we have these two predicates, one of which says something is a pancake, and one of which says that two things taste similar.



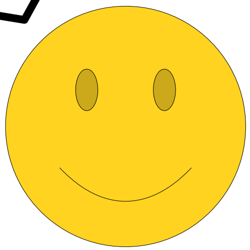
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any two pancakes taste similar

Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

How might we translate this statement into first-order logic?



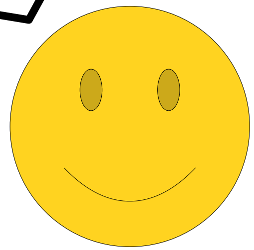
```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any two pancakes taste similar

Available Predicates:

Pancake(x)
TasteSimilar(x, y)

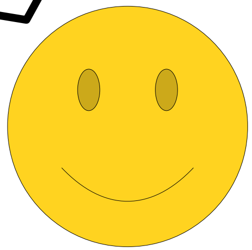
This statement is different from our earlier one because it talks about any possible pair of objects rather than any possible individual object.



```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any two pancakes taste similar

The good news is that we can translate it in a way that bears a strong resemblance to the above Java code with a double for loop.



Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

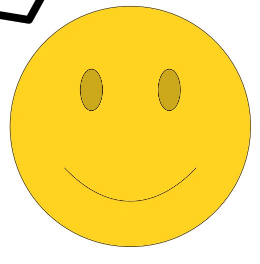
```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any two pancakes taste similar

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Specifically, we'll proceed as follows.



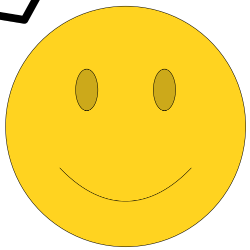
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any two pancakes x and y taste similar

Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

First, let's introduce some new variables into our English so that we have names for things.



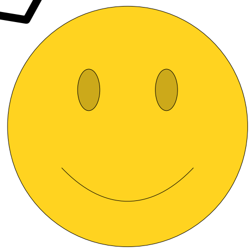
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any pancake x tastes similar to any pancake y

Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

We can then rejigger the English statement so that it looks like this. After all, this means the same thing as what we started with.



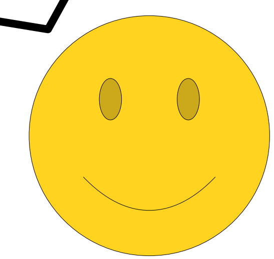
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any pancake x tastes similar to any pancake y

Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

Now, we can think back to our Aristotelean form templates that we just got really familiar with and see how to apply them here.



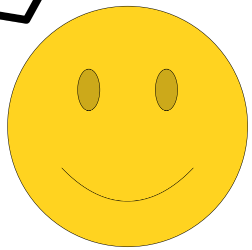
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

Any pancake x tastes similar to any pancake y

Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

Since this statement says something to the effect of "any pancake x has some special property…"



```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$\forall x.\ (Pancake(x) \to$

$\quad$ *x tastes similar to any pancake y*

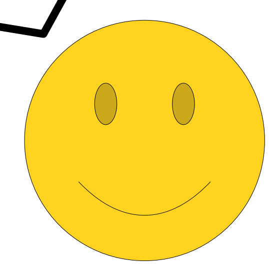
$)$

Available Predicates:

$Pancake(x)$

$TasteSimilar(x, y)$

… we can begin translating it into logic like this.



```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$\forall x.\ (Pancake(x) \rightarrow$

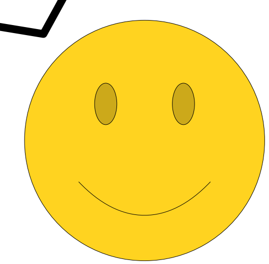
x tastes similar to any pancake y

$)$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Now, let's look at that middle portion and see if we can translate it as well.



```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$\forall x.\ (Pancake(x) \rightarrow$

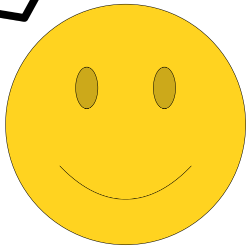
any pancake y tastes similar to x

$)$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Reordering the statement gives us this to work with, which exposes a bit more structure.



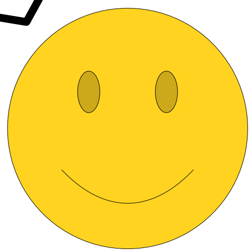
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\qquad \forall y.\ (Pancake(y) \to \\
&\qquad\qquad \textit{x tastes similar to y} \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

We can then rewrite it like this.



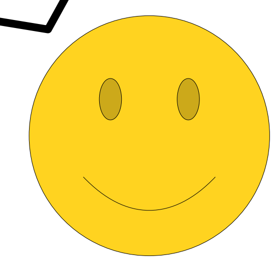
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\quad \forall y.\ (Pancake(y) \to \\
&\quad\quad \textit{x tastes similar to y} \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

As a final step, we'll translate that innermost portion.



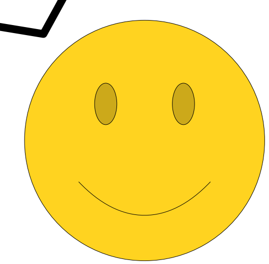
```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\quad\quad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

Tada! We're done.



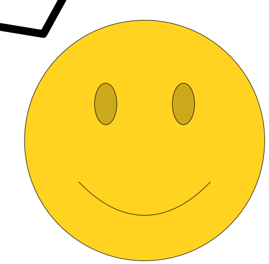
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\quad\quad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

We now have a statement that says that any two pancakes taste similar. (We can debate whether this is true or not in a separate guide.)



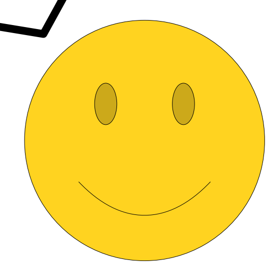
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\qquad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

Hopefully, you can notice that there's a bit of a parallel to the Java double for loop given above.



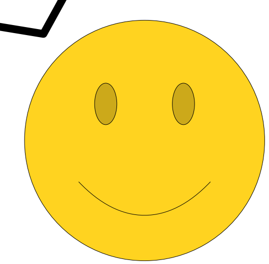
```
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\quad\quad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

If you think as quantifiers as a sort of "loop over everything" – which isn't that far from the truth – then the program and the formula both say "loop over one thing, then loop over another, then do something with the pair."



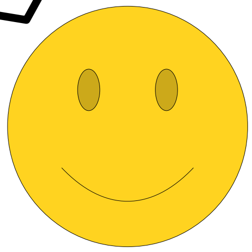
```cpp
void printPairsIn(vector<int> elems) {
    for (int i = 0; i < elems.size(); i++) {
        for (int j = 0; j < elems.size(); j++) {
            cout << elems[i] << ", " << elems[j] << endl;
        }
    }
}
```

$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\quad\quad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

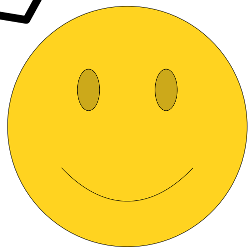
So if you ever need to write something where you're dealing with a pair of things, you now know how! You can just write two independent quantifiers like this.



Available Predicates:

Pancake(x)
TasteSimilar(x, y)

It turns out, though, that there's another way to express this concept that some people find a bit easier to wrap their head around. For completeness, let's quickly talk about this before moving on.

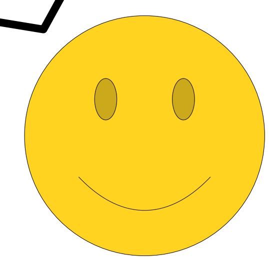


Any two pancakes taste similar

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Let's go back to our original statement.

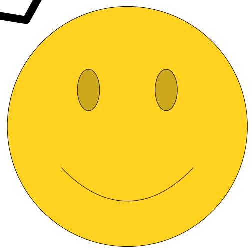


Any two pancakes x and y taste similar

Available Predicates:

Pancake(x)
TasteSimilar(x, y)

As before, let's add in some variables names so that we have ways of keeping our pancakes straight. (Ever gotten your pancakes confused? It's a horrible way to start off your day.)

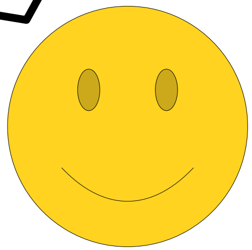


Any two pancakes x and y taste similar

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

The idea is that we know that, at this point, we're going to be reasoning about a pair of pancakes, and we're going to reason about them right now.

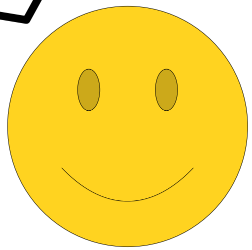


Any two pancakes x and y taste similar

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Therefore, rather than introducing two quantifiers at different points in time, we'll introduce both quantifiers at the same time...



$$\forall x.\ \forall y.\ (\textit{x and y are pancakes} \rightarrow$$
$$\quad \textit{x and y taste similar}$$
$$)$$

Available Predicates:

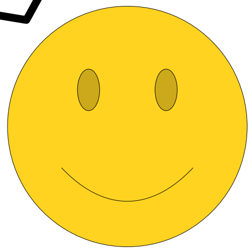
Pancake(x)
TasteSimilar(x, y)

∀x. ∀y. (*x and y are pancakes* →
x and y taste similar
)

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

Generally speaking, it is <u>not</u> a good idea to introduce quantifiers for variables all at once, but in the special case of working with pairs, it's perfectly safe.



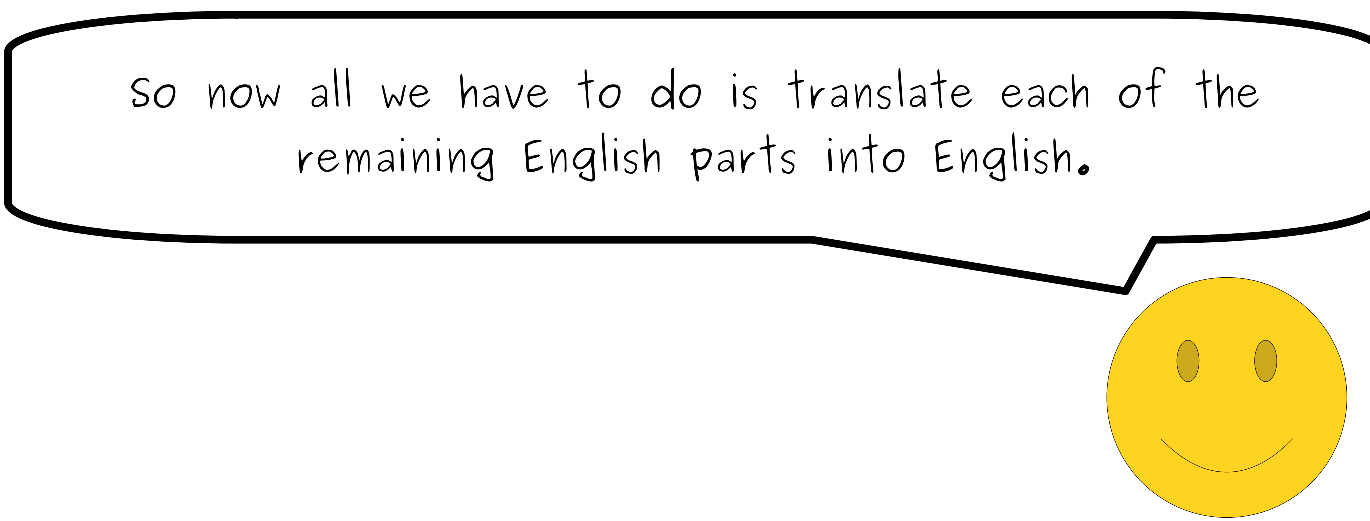
$\forall x.\ \forall y.$ (*x and y are pancakes* $\to$

x and y taste similar

)

Available Predicates:

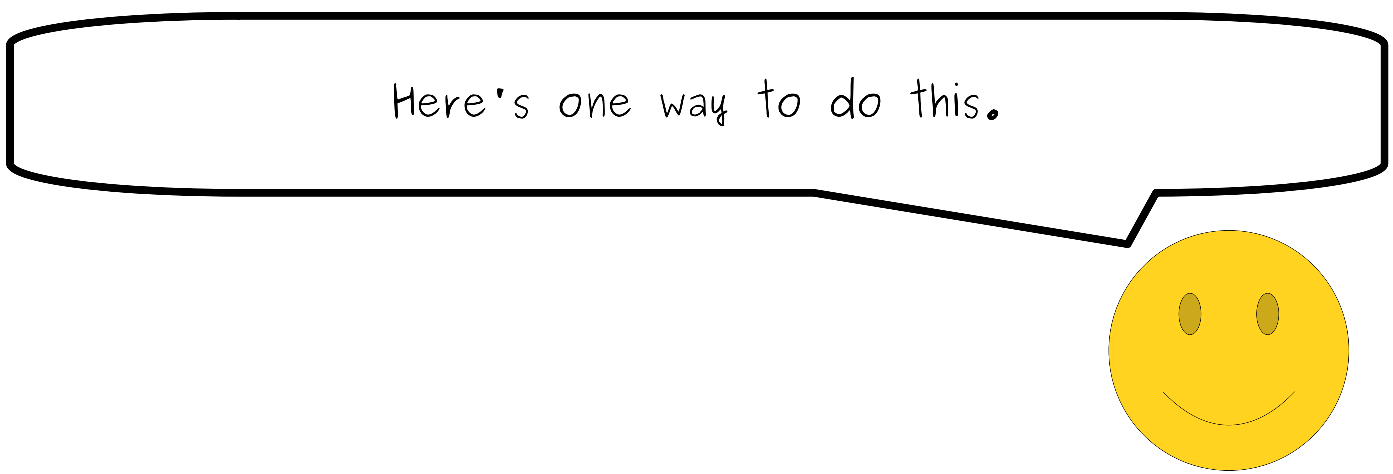
$Pancake(x)$
$TasteSimilar(x, y)$



$$
\begin{aligned}
&\forall x.\ \forall y.\ (\mathit{Pancake}(x) \land \mathit{Pancake}(y) \to \\
&\qquad \mathit{TasteSimilar}(x, y) \\
&)
\end{aligned}
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

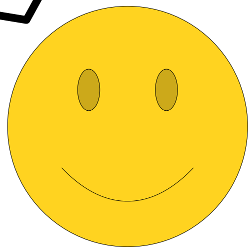


$$
\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to
\quad TasteSimilar(x, y)
)
$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

And we're done! This is a totally valid way to translate our original statement into first-order logic.



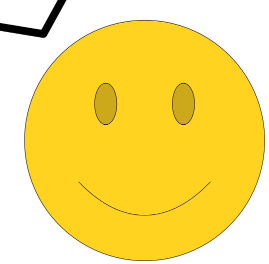
$$\forall x.\ (Pancake(x) \rightarrow \\ \quad \forall y.\ (Pancake(y) \rightarrow \\ \quad\quad TasteSimilar(x, y) \\ \quad) \\ )$$

$$\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \rightarrow \\ \quad TasteSimilar(x, y) \\ )$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

It's interesting, and useful, to put this second translation side-by-side with our original one.



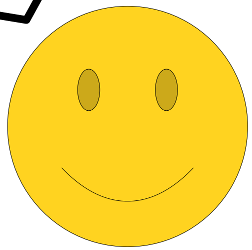
$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\quad \forall y.\ (Pancake(y) \to \\
&\quad\quad TasteSimilar(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to \\
&\quad TasteSimilar(x, y) \\
&)
\end{aligned}
$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

These statements look pretty different, but they say exactly the same thing. Both are perfectly correct.



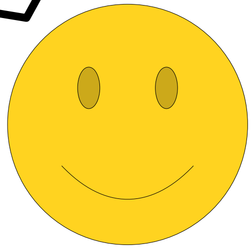
$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\quad \forall y.\ (Pancake(y) \to \\
&\quad\quad TasteSimilar(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to \\
&\quad TasteSimilar(x, y) \\
&)
\end{aligned}
$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

There's actually something pretty cool and pretty deep going on here.



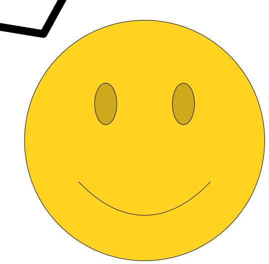
$$\forall x.\ (Pancake(x) \to \forall y.\ (Pancake(y) \to TasteSimilar(x, y)))$$

$$\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to TasteSimilar(x, y))$$

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$

For now, ignore the quantifiers. Just look at the predicates and how they relate.



$$
\forall x.\ (\mathit{Pancake}(x) \rightarrow \forall y.\ (\mathit{Pancake}(y) \rightarrow \mathit{TasteSimilar}(x, y)))
$$

$$
\forall x.\ \forall y.\ (\mathit{Pancake}(x) \land \mathit{Pancake}(y) \rightarrow \mathit{TasteSimilar}(x, y))
$$

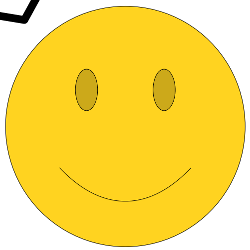
$$
A \rightarrow B \rightarrow C
$$

$$
A \land B \rightarrow C
$$

Available Predicates:

$\mathit{Pancake}(x)$
$\mathit{TasteSimilar}(x, y)$

Abstractly, here are the two propositional logic patterns used in the two statements.

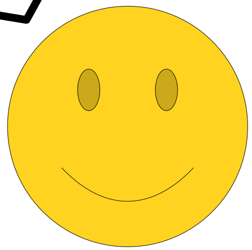


$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\quad \forall y.\ (Pancake(y) \to \\
&\quad\quad TasteSimilar(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to \\
&\quad TasteSimilar(x, y) \\
&)
\end{aligned}
$$

$A \to B \to C$ *is equivalent to* $A \land B \to C$

These statements are actually logically equivalent to one another. (If you've checked out the Guide to Negating Formulas, you'll see a cool way to derive this!)



Available Predicates:

Pancake(*x*)
TasteSimilar(*x*, *y*)

$$\forall x. \ (Pancake(x) \rightarrow \\ \quad \forall y. \ (Pancake(y) \rightarrow \\ \quad\quad TasteSimilar(x, y) \\ \quad) \\ )$$

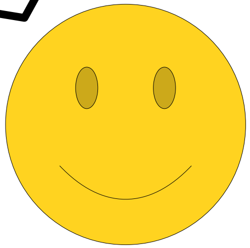
$$\forall x. \ \forall y. \ (Pancake(x) \land Pancake(y) \rightarrow \\ \quad TasteSimilar(x, y) \\ )$$

$$A \rightarrow B \rightarrow C \qquad \textit{is equivalent to} \qquad A \land B \rightarrow C$$

This pattern – changing a chain of implications into a single implication and a lot of ANDs and vice-versa – is sometimes called ***Currying*** and has applications in functional programming. (This is a total aside… you're not expected to know this.)

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$



$$
\begin{aligned}
&\forall x.\ (Pancake(x) \to \\
&\quad \forall y.\ (Pancake(y) \to \\
&\qquad TasteSimilar(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

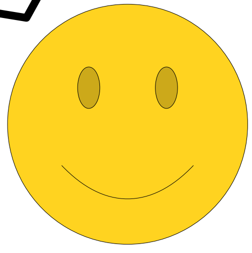
$$
\begin{aligned}
&\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \to \\
&\quad TasteSimilar(x, y) \\
&)
\end{aligned}
$$

$A \to B \to C$ *is equivalent to* $A \land B \to C$

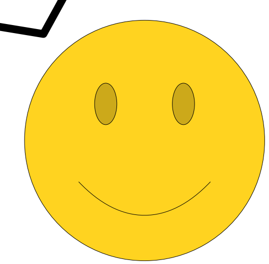
Ultimately, what's important is that you understand that both of these statements say exactly the same thing and that you end up comfortable working with both of them. Feel free to use whichever one you like more, but make sure you can quickly interpret both.

Available Predicates:

$Pancake(x)$
$TasteSimilar(x, y)$



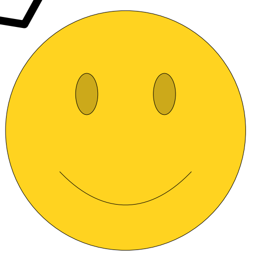
Let's do another example of where we might want to go and work with pairs.



Available Predicates:

Person(x)
Knows(x, y)

Let's switch our predicates from pancakes to people.

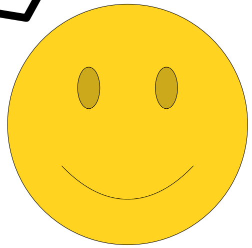


Everyone knows at least two people

Available Predicates:

Person(*x*)
Knows(*x*, *y*)

How might we translate this statement into first-order logic?

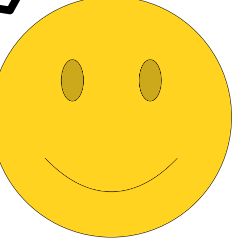


Everyone knows at least two people

Available Predicates:

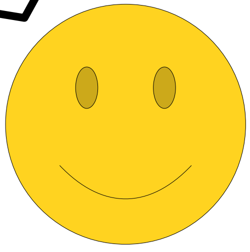
Person(x)
Knows(x, y)

Well, it seems like there's going to be a pair involved here somewhere, since there's something about "at least two people" here.



Everyone knows *at least two people*

However, that <u>does not</u> mean that we should immediately start writing out something about a pair of people. Remember – we should only introduce quantifiers when we immediately need them, and it's not clear that we need to start talking about these two people yet.



Available Predicates:

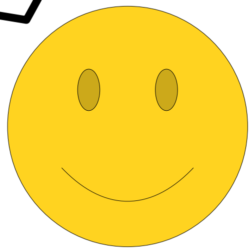
$Person(x)$
$Knows(x, y)$

Everyone knows at least two people

Available Predicates:

$Person(x)$
$Knows(x, y)$

Instead, let's look at the overall structure of this statement and see what it is that we're trying to say.

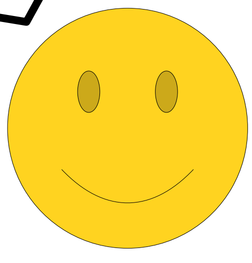


Every person x knows at least two people y and z

Available Predicates:

Person(x)
Knows(x, y)

As usual, let's start by introducing some variables so that we can keep track of who we're talking about.



$\forall x.\ (Person(x) \rightarrow$

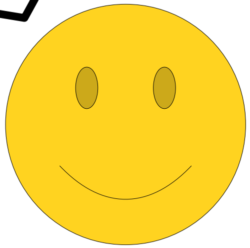
x knows at least two people y and z

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

We can then partially translate this statement using the techniques we've seen so far.



$\forall x. \ (Person(x) \rightarrow$

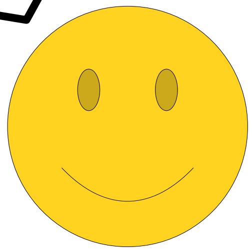
x knows at least two people y and z

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Now, we need to express the idea that x knows two people x and y.



$\forall x.\ (Person(x) \to$

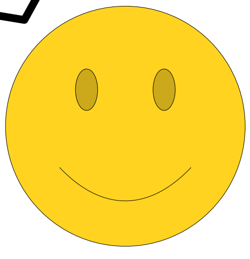
x knows at least two people y and z

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

There are a couple of ways to do it, and since we've got time, we'll do it in two different ways.

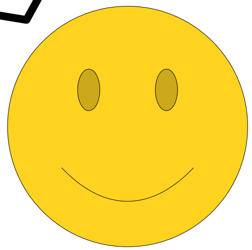


$\forall x.\ (Person(x) \rightarrow$

x knows at least two people y and z

$)$

Previously, we talked about working with pairs in a universally-quantified setting. Here, though, this particular pair is going to be existentially quantified, since we're saying that there exist two people with certain properties.



Available Predicates:

$Person(x)$
$Knows(x, y)$

$\forall x.\ (Person(x) \rightarrow$

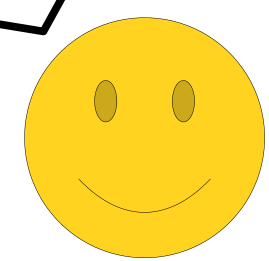
there are two people y and z that x knows

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

It might be easier to see that if we rewrite things like this.



$\forall x.\ (Person(x) \rightarrow$

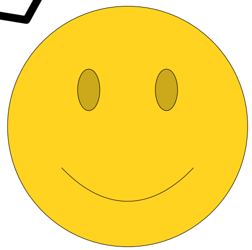
there are two people y and z that x knows

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Thinking back to our double for loop intuition, let's see if we can translate this statement by nesting some existential statements inside of one anothter.



$\forall x.\ (Person(x) \to$

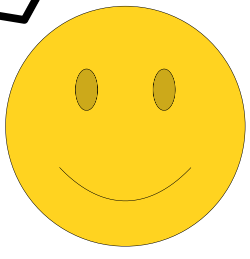
there is a person y that x knows and a different person z that x knows.

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Let's begin by rewriting the English like this.



$\forall x.\ (Person(x) \to$

$\quad \exists y.\ (Person(y) \land Knows(x, y) \land$

$\qquad$ *there is a different person z that x knows*

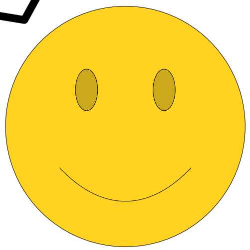
$\quad)$

$)$

Available Predicates:

$Person(x)$
$Knows(x, y)$

We can now make some progress translating this.

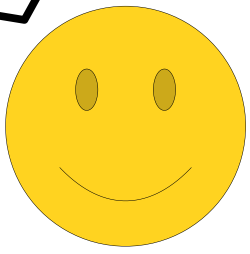


$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\qquad \text{there is a different person z that x knows} \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

We can then finish up the rest of this translation by translating this blue part in the middle. But that shouldn't be too bad!



$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land \\
\quad\quad\quad \textit{z is a different person from y} \\
\quad\quad) \\
\quad) \\
)
\end{array}
$$

Available Predicates:

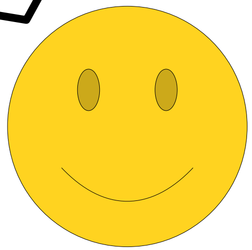
$Person(x)$
$Knows(x, y)$

$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land \\
\quad\quad\quad \textit{z is a different person from y} \\
\quad\quad) \\
\quad) \\
)
\end{array}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

The last step is to say that z and y aren't the same person.

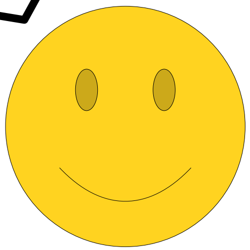


$$
\begin{array}{l}
\forall x.\ (Person(x) \rightarrow \\
\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land \\
\quad\quad\quad \textcolor{blue}{\textit{z is a different person from y}} \\
\quad\quad) \\
\quad) \\
)
\end{array}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

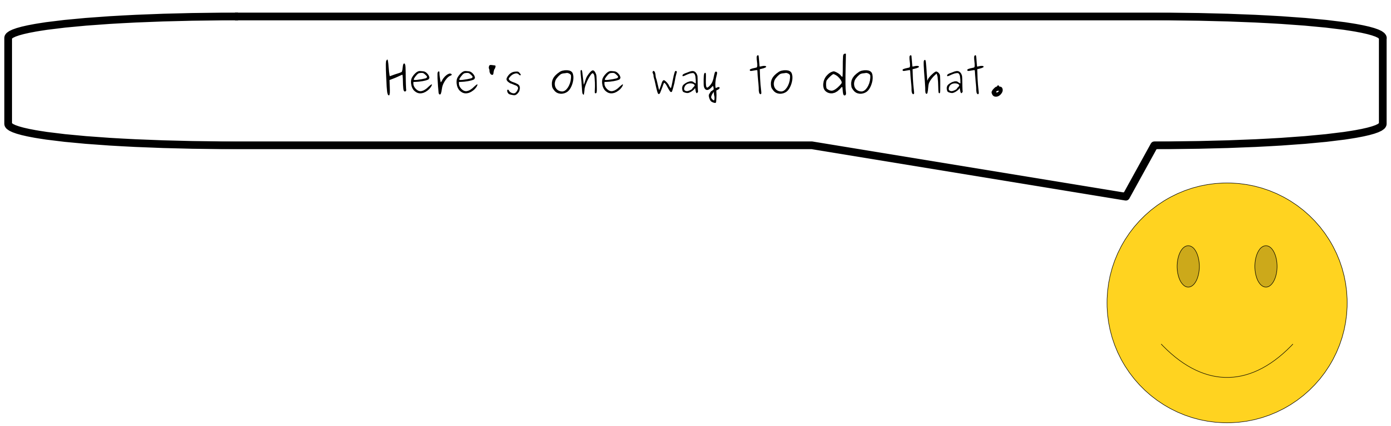
Even though we didn't explicitly list it in our list of predicates, remember that first-order logic has the equality predicate built into it, so we're always allowed to state that two things are the same or are different.



$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
\quad) \\
)
\end{array}
$$

Available Predicates:

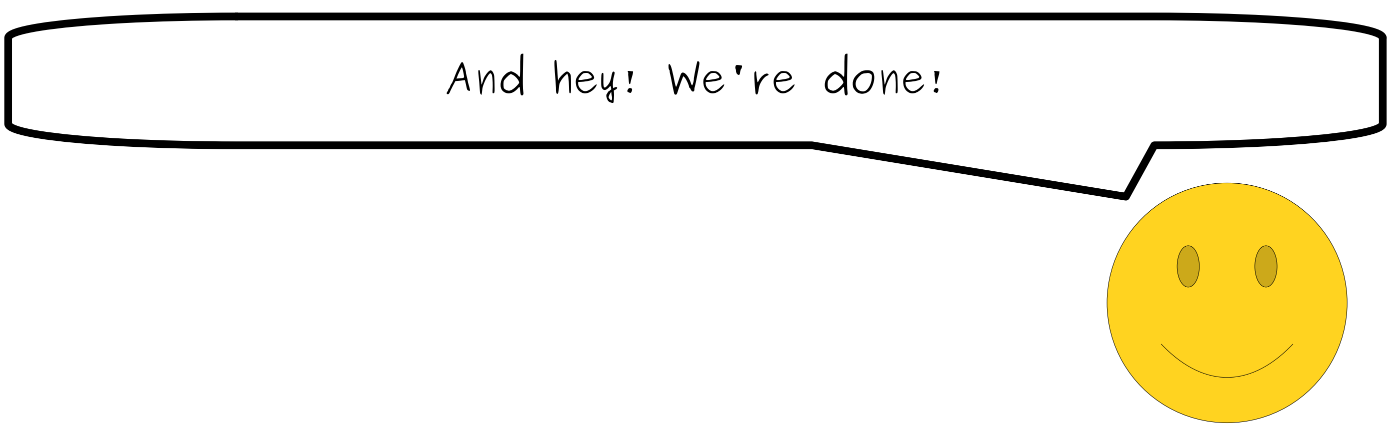
$Person(x)$
$Knows(x, y)$



$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\quad\quad\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\quad\quad) \\
&)
\end{aligned}
$$

Available Predicates:

Person(x)
Knows(x, y)

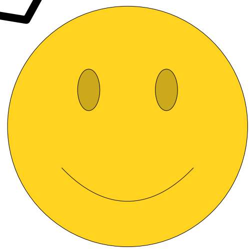


$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\qquad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Notice how we're using a pair of nested existential quantifiers to express the idea that there's a pair of people with specific properties.

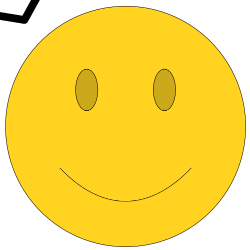


$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Hopefully, this seems familiar, since it's closely related to the analogous doubly-nested quantifiers we saw when talking about pairs of pancakes.

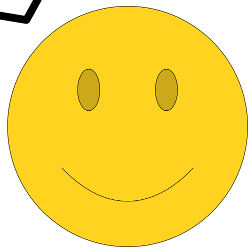


$$
\begin{aligned}
&\forall x.\ (Person(x) \rightarrow \\
&\qquad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\qquad\qquad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

Just as we could write "any pair of pancakes" in two ways, we can write "some pair of different people" in two ways.



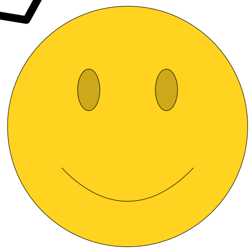
$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ \exists z.\ (Person(y) \land Person(z) \land z \neq y \land \\
&\quad\quad Knows(x, y) \land Knows(x, z) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Person(x)$
$Knows(x, y)$

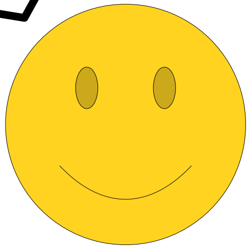
Here's the alternative approach. Here, we introduce the quantifiers for y and z at the same time, then constrain y and z with preconditions at the same time.



$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
&\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ (Person(x) \to \\
&\quad \exists y.\ \exists z.\ (Person(y) \land Person(z) \land z \neq y \land \\
&\quad\quad Knows(x, y) \land Knows(x, z) \\
&\quad) \\
&)
\end{aligned}
$$

These two approaches are completely equivalent, and both of them are correct. As with quantifying over pairs using ∀, it's a good idea to get comfortable with quantifying over pairs using ∃ with both of these approaches.



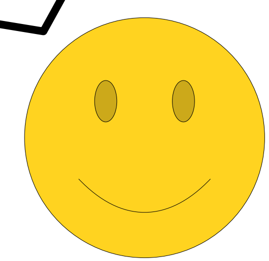
Available Predicates:

$Person(x)$
$Knows(x, y)$

$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ (Person(y) \land Knows(x, y) \land \\
\quad\quad \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y) \\
\quad) \\
)
\end{array}
$$

$$
\begin{array}{l}
\forall x.\ (Person(x) \to \\
\quad \exists y.\ \exists z.\ (Person(y) \land Person(z) \land z \neq y \land \\
\quad\quad Knows(x, y) \land Knows(x, z) \\
\quad) \\
)
\end{array}
$$

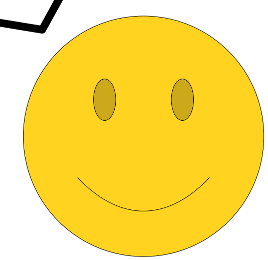
On Problem Set Two, you'll get to consider a variation on this problem: how would you express the idea that this person x knows exactly two people? That's a trickier proposition, but (hypothetically speaking) you may want to use this basic setup as a starting point.

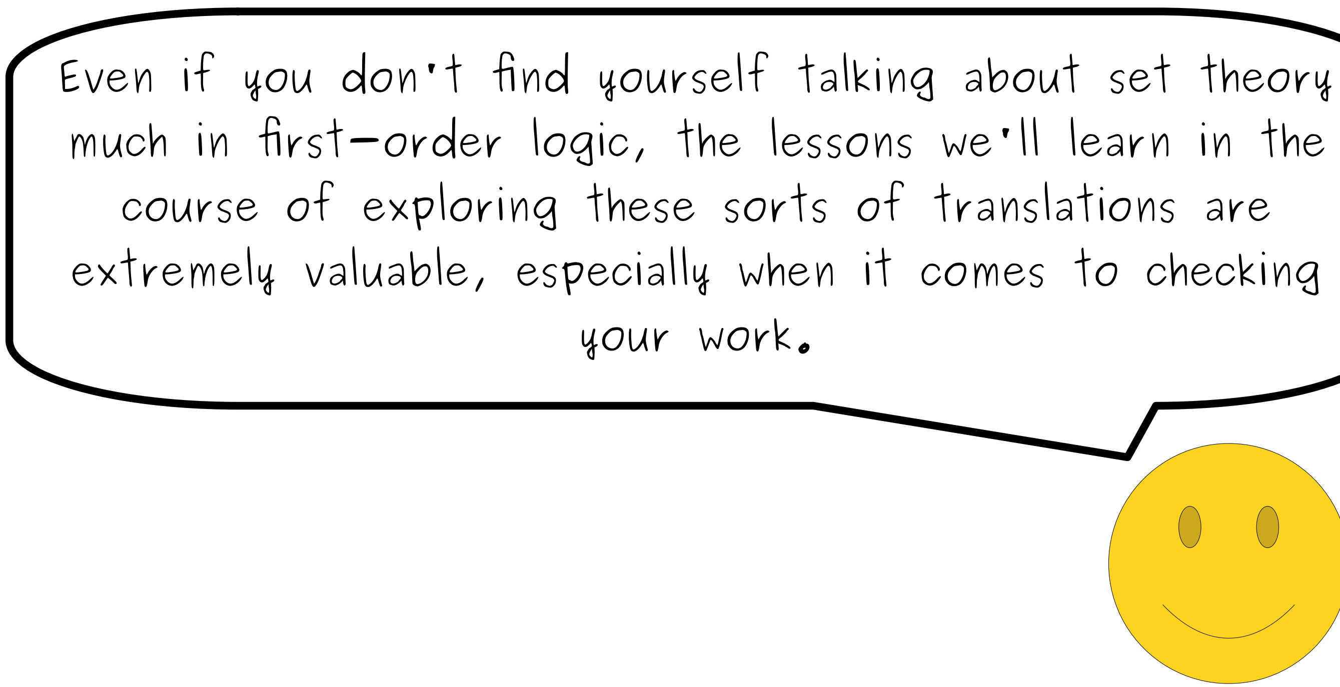


Available Predicates:

$Person(x)$
$Knows(x, y)$

There's one last topic I'd like to speak about in this guide, and that's what happens when you start talking about sets and set theory in first-order logic.

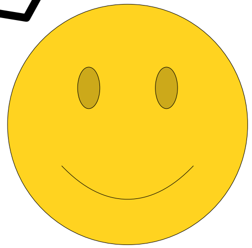



Even if you don't find yourself talking about set theory much in first-order logic, the lessons we'll learn in the course of exploring these sorts of translations are extremely valuable, especially when it comes to checking your work.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Let's imagine that we have the set of predicates over to the left. We can say that something is a set, that one thing is an element of something else, that something is an integer, and that something is negative.

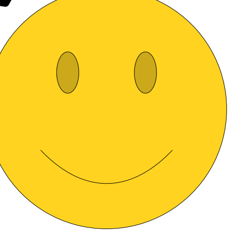


The set of all natural numbers exists

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

How might we translate this statement into first-order logic?



The set of all natural numbers exists

Available Predicates:

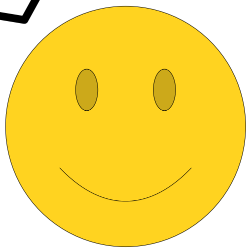
$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

The set of all natural numbers exists

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

First, the statement doesn't seem to look anything like the Aristotelian forms that we saw earlier. Instead, it just says that something exists.

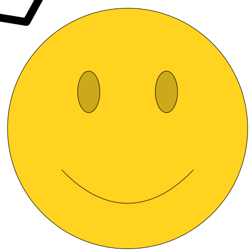


The set of all natural numbers exists

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Second, this statement refers to a specific thing – the set of all natural numbers – and so it's not exactly clear how we'd actually translate this into logic.

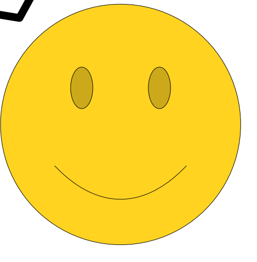


The set of all natural numbers exists

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

If you encounter a statement like this one, which asks you to show that something exists, it often helps to reframe the statement to translate in a different light.

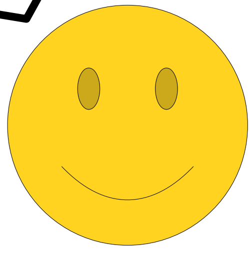


The set of all natural numbers exists

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Rather than saying "this specific thing exists…"



There is a set that is the set of all natural numbers

Available Predicates:

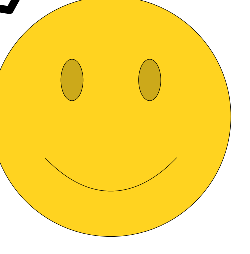
$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

There is a set that is the set of all natural numbers

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This looks a lot more like the forms that we saw earlier, so we can start to translate it into first-order logic using similar techniques.



$\exists S.\ (Set(S) \land$

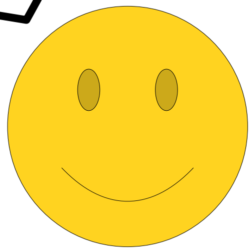
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Here's one way that we can get this translation started.



$\exists S.\ (Set(S) \land$

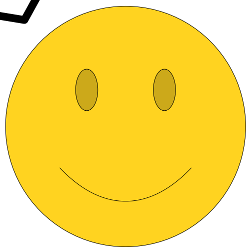
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So now we need to find a way to pin down the fact that S is the set of all natural numbers.



$\exists S.\ (Set(S) \land$

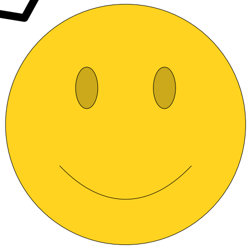
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

To do so, let's take a few minutes to think about how we might do that.

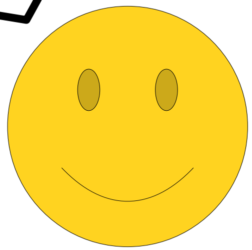


$\exists S.\ (Set(S)\ \land$

S is the set of all natural numbers

$)$

If we're going to say that S is the set of all natural numbers, we're probably going to need to find some way to talk about its elements. After all, sets are uniquely defined by their elements, so if we want to say that we have a set with a certain property, we can do so by saying that it has the right elements.



Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$\exists S.\ (Set(S) \land$

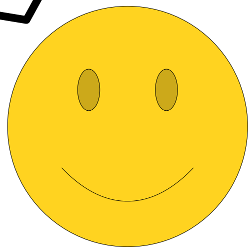
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We're not sure <u>how</u> we're going to do that, but at least we know to keep an eye out for that.



$\exists S.\ (Set(S) \land$

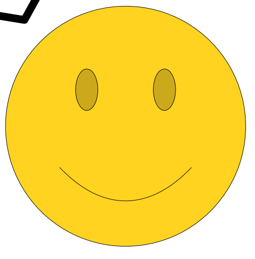
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Next, we need to find a way to say that something is a natural number.



$\exists S.\ (Set(S)\ \land$

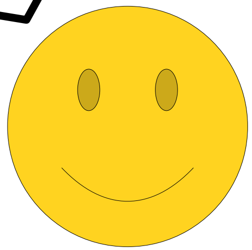
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We have the ability to say that something is an integer or that something is negative, and that might come in handy – the natural numbers are the integers that aren't negative!



$\exists S.\ (Set(S)\ \land$

S is the set of all natural numbers

$)$

Available Predicates:

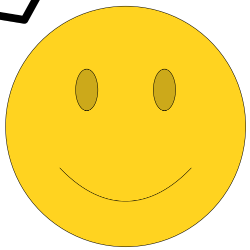
$Set(x)$

$x \in y$

$Integer(x)$

$Negative(x)$

So even if we have no idea where we're going right now, we at least know that (1) we want to say something about the elements of S, and (2) we're going to try to say something about how they're integers that aren't negative.



$\exists S.\ (Set(S)\ \land$

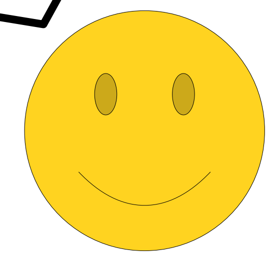
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Rather than just show you the final answer, let's see how *not* to do this.



$\exists S.\ (Set(S)\ \land$

S is the set of all natural numbers

$)$

Available Predicates:

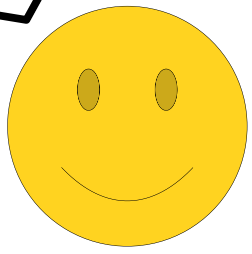
$Set(x)$

$x \in y$

$Integer(x)$

$Negative(x)$

As before, I'm going to put up the emergency warning flags indicating that we're doing something wrong here.



$\exists S.\ (Set(S)\ \land$

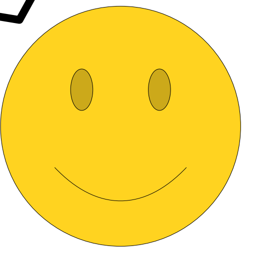
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Let's try an initial approach. What does it mean for S to be the set of all natural numbers?

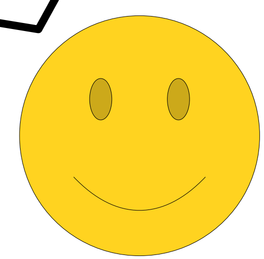


$\exists S. \ (Set(S) \land$

S contains all the natural numbers

$)$

Here's a reasonable – but incorrect – way of thinking about it. If you don't see why this is incorrect, don't worry! It's subtle, which is precisely why we're taking the time to go down this route.



Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$\exists S.\ (Set(S) \land$

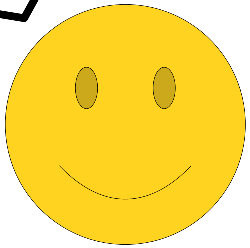
S contains all the natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Now, how might we translate this red statement into first-order logic?



$\exists S.\ (Set(S)\ \land$

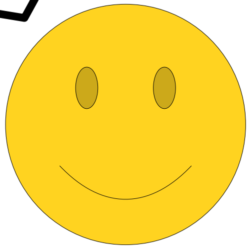
every natural number is an element of S

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Again, let's change up the ordering of the English to expose a bit more structure.

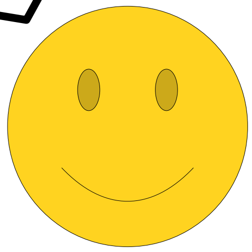


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (\textit{x is a natural number} \rightarrow \\
&\qquad \textit{x is an element of S} \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This matches one of our nice Aristotelian forms, so we can rewrite it like this.

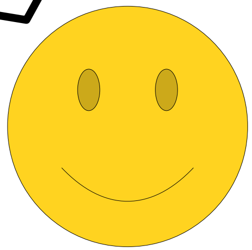


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (\text{x is a natural number} \to \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We can clean up the consequent of that implication (the part that's implied) using the predicates we have available.

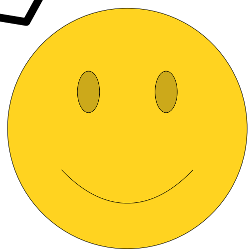


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (\textit{x is an integer and x isn't negative} \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

As for the antecedent – as we saw earlier, the natural numbers are the integers that aren't negative, so we can say something like this.

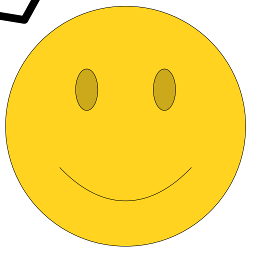


$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \to \\
\quad\quad x \in S \\
\quad) \\
)
\end{array}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We can then translate that into logic like this. Done! …ish

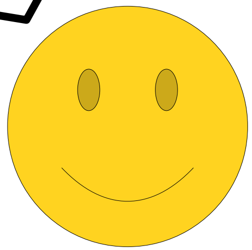


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So it seems like we're done, but we still have those big red warning signs everywhere. Why doesn't this work?

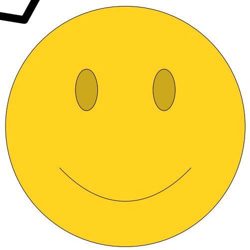


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \rightarrow \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Well, fundamentally, the way this statement works is by saying "there is some set S that is the set of all natural numbers."

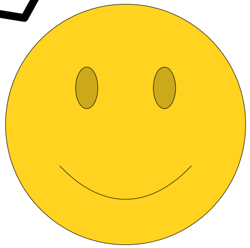


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \lnot Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Since this is an existentially-quantified statement, it's true if we can find a choice of S that makes it true.

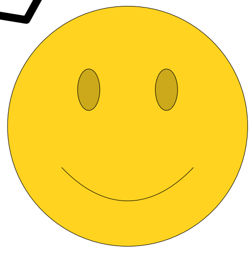


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We've tried to structure this statement with the intent that, specifically, the only choice of S that will work should be $\mathbb{N}$, the set of all natural numbers.

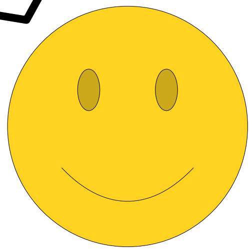


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land\\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to\\
&\qquad\qquad x \in S\\
&\qquad)\\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

If we can make this statement true <u>without</u> choosing S to be the set of all natural numbers, then we haven't actually stated that $\mathbb{N}$ exists.

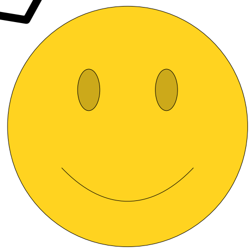


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Unfortunately, it is entirely possible to choose a set besides ℕ that makes this formula true.



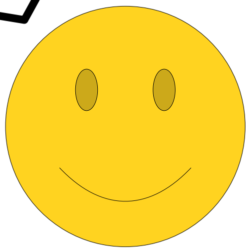
$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \rightarrow \\
\quad\quad x \in S \\
\quad) \\
)
\end{array}
$$

Choose $S = \mathbb{R}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Specifically, what if we choose S to be the set $\mathbb{R}$?



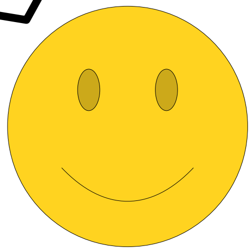
$$
\begin{aligned}
&\exists S.\ (\mathit{Set}(S)\ \land \\
&\quad \forall x.\ (\mathit{Integer}(x) \land \lnot \mathit{Negative}(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \mathbb{R}$.

Available Predicates:

$\mathit{Set}(x)$
$x \in y$
$\mathit{Integer}(x)$
$\mathit{Negative}(x)$

That means that S is definitely a set…



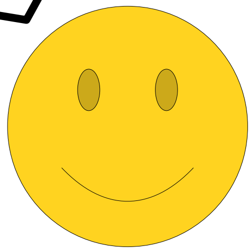
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \mathbb{R}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

…and this part of the formula is true: every nonnegative integer is contained in S.



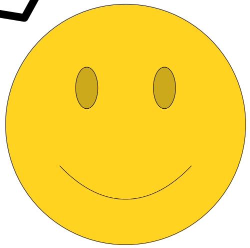
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Choose $S = \mathbb{R}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

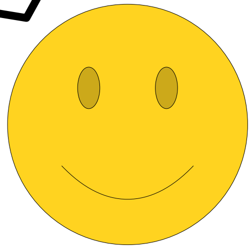
This means that the statement we've written doesn't say "the set of all natural numbers exists." It says "there is some set that contains all the natural numbers," which is similar, but not the same thing.



$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Choose $S = \mathbb{R}$.

Fundamentally, the issue with this translation is that we've put on a set of <u>minimum</u> requirements on S, not a set of <u>exact</u> requirements. As a result, it's possible to make this formula true with a choice of S that has some, but not all, of the properties of $\mathbb{N}$. We're going to need to rework the formula to correct that deficiency.



Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

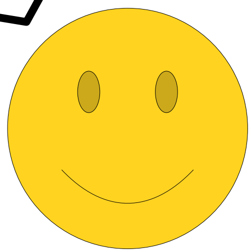
$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x)\ \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \mathbb{R}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

To do so, let's go back in time to the last point where everything was working correctly…



$\exists S. \ (Set(S) \land$

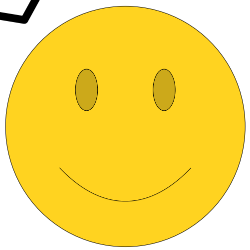
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

… which was this point here!



$$\exists S.\ (Set(S)\ \land$$

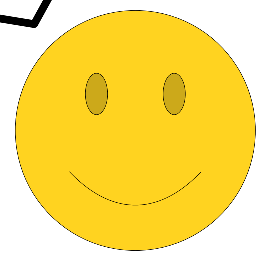
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Okay, so we know that just saying "S contains all the natural numbers" isn't going to work, because other sets besides ℝ can also contains all the natural numbers.



$\exists S.\ (Set(S) \land$

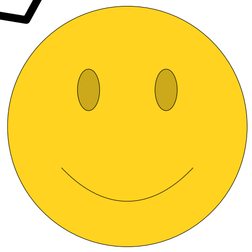
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So what other approaches can we take?



$\exists S.\ (Set(S)\ \land$

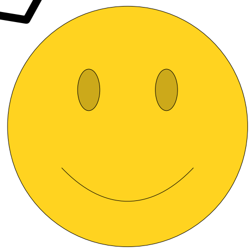
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

I'm going to show you another approach that doesn't work, which is a common strategy that we see students take after they realize that the previous approach is incorrect.



$\exists S.\ (Set(S)\ \land$

S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$

$x \in y$

$Integer(x)$

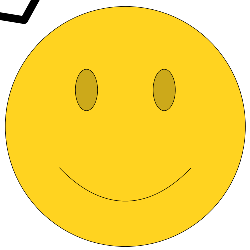
$Negative(x)$

$\exists S.\ (Set(S) \land$

S is the set of all natural numbers

$)$

Maybe we should think about this differently. The reason that we could get away with choosing ℝ for our set S was that our formula said "S has to have at least these elements." What if we try a different tactic and say that S has to have at most these elements?



Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$\exists S.\ (Set(S)\ \land$

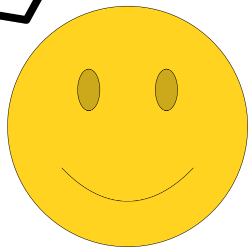
the only elements of S are natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

That is, what if we try replacing the previous blue statement with this red statement?



$\exists S.\ (Set(S)\ \land$

the only elements of S are natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$\exists S.\ (Set(S) \land$

the only elements of S are natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Given that it's different, let's see if we can translate this into first-order logic.

$\exists S.\ (Set(S)\ \wedge$

every element of S is a natural number

$)$

Available Predicates:

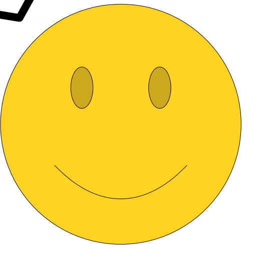
$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \to \\
&\qquad\qquad \textit{x is a natural number} \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This statement matches one of our forms, so let's go and translate it.

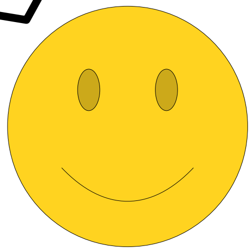


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\qquad \textit{x is a natural number} \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

And, since we've seen earlier how to express the idea that x is a natural number…

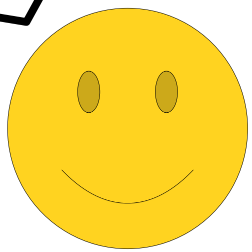


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

…we can complete our translation like this.

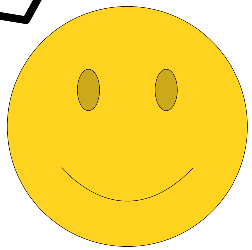


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So we're done! But is it correct?

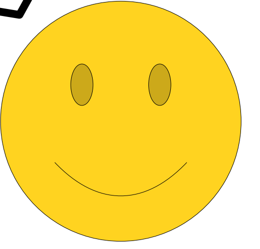


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

As before, we should check to make sure that the only way this statement can be made true is by picking S to be the set of all natural numbers. Is that really the case?



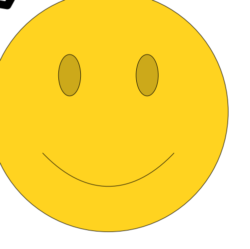
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \{137\}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Unfortunately, no. What if we pick this choice for S?



$$
\begin{aligned}
&\exists S.\ (\mathit{Set}(S)\ \land \\
&\qquad \forall x.\ (x \in S \rightarrow \\
&\qquad\qquad \mathit{Integer}(x) \land \neg \mathit{Negative}(x) \\
&\qquad) \\
&)
\end{aligned}
$$

Choose $S = \{137\}$.

Available Predicates:

$\mathit{Set}(x)$
$x \in y$
$\mathit{Integer}(x)$
$\mathit{Negative}(x)$

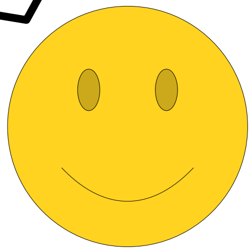
$$
\begin{array}{l}
\exists S.\ (Set(S) \land \\
\quad \forall x.\ (x \in S \rightarrow \\
\quad\quad Integer(x) \land \neg Negative(x) \\
\quad) \\
)
\end{array}
$$

$$Choose\ S = \{137\}.$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

… and this statement is true: every element of S is indeed a natural number.



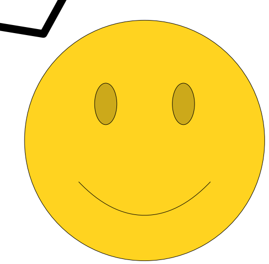
$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \{137\}$.

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

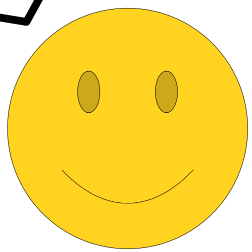
So our translation isn't correct – even if there is no set of all natural numbers, we can still make the formula true by picking some other set… in this case, any set that happens to only contain natural numbers.



$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Choose $S = \emptyset$

Interesting, we could have also chosen $S = \emptyset$ as a counterexample. Then this inner statement happens to be vacuously true because there are no elements of S to speak of!



Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

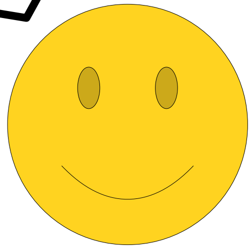
$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So here are our two attempted translations, each of which isn't correct.



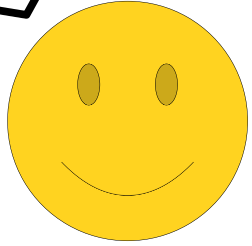
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Interestingly, although each of them is wrong, they're wrong in complementary ways.



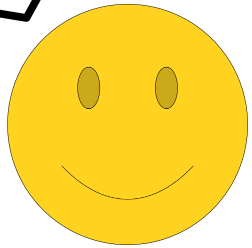
$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \rightarrow \\
\quad\quad x \in S \\
\quad) \\
)
\end{array}
$$

$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\quad \forall x.\ (x \in S \rightarrow \\
\quad\quad Integer(x)\ \land\ \neg Negative(x) \\
\quad) \\
)
\end{array}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Our first statement was wrong because it let us choose sets that had all the natural numbers, plus some other things that shouldn't be there.



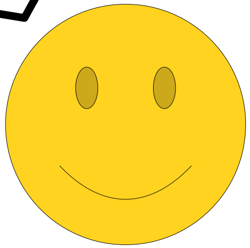
$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (Integer(x)\ \land\ \neg Negative(x) \to \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x)\ \land\ \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

However, notice that we can't pick an S that misses any natural numbers, because the inside says that all the natural numbers should be there.



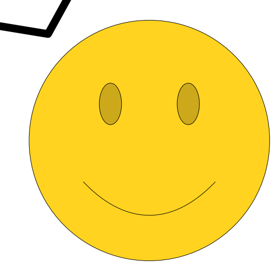
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This second statement was incorrect because it let us choose sets S with too few elements, since all it required was that elements that were present were natural numbers.



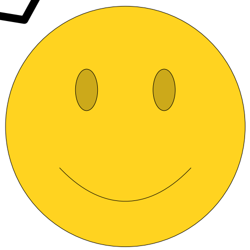
$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
\qquad\qquad x \in S \\
\qquad) \\
)
\end{array}
$$

$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\qquad \forall x.\ (x \in S \rightarrow \\
\qquad\qquad Integer(x) \land \neg Negative(x) \\
\qquad) \\
)
\end{array}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

However, note that this formula doesn't let us choose a set S that contains anything that's not a natural number, since it requires everything in S to be a natural number.



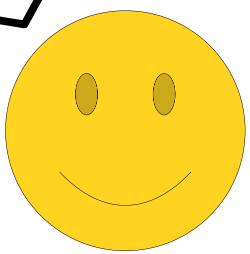
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \lnot Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \to \\
&\qquad Integer(x) \land \lnot Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

In a sense, you can think of our translations this way…



$\exists S.\ (Set(S) \land$

$\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to$

$\qquad x \in S$

$\quad)$

$)$

$(\mathbb{N} \subseteq S)$

$\exists S.\ (Set(S) \land$

$\quad \forall x.\ (x \in S \to$

$\qquad Integer(x) \land \neg Negative(x)$

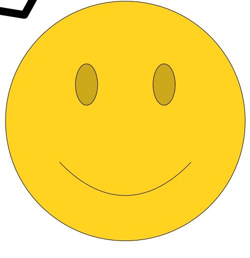
$\quad)$

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This first part says "$\mathbb{N} \subseteq S$," since it requires that every natural number be in S.



$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

$(\mathbb{N} \subseteq S)$

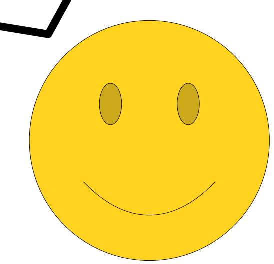
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

$(S \subseteq \mathbb{N})$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This second part says $S \subseteq \mathbb{N}$, since it requires that every element of S be a natural number.



$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

$$(\mathbb{N} \subseteq S)$$

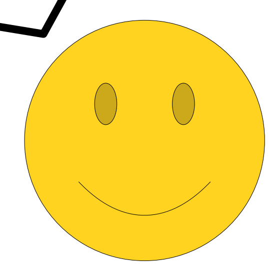
$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\qquad \forall x.\ (x \in S \to \\
&\qquad\qquad Integer(x) \land \neg Negative(x) \\
&\qquad) \\
&)
\end{aligned}
$$

$$(S \subseteq \mathbb{N})$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

In other words, each individual constraint doesn't guarantee that S has to be $\mathbb{N}$, but the two statements collectively would require that $S = \mathbb{N}$!



$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

$$(\mathbb{N} \subseteq S)$$

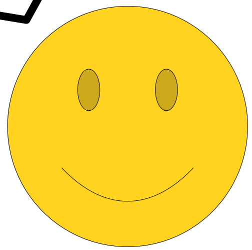
$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \to \\
&\qquad\qquad Integer(x) \land \neg Negative(x) \\
&\qquad) \\
&)
\end{aligned}
$$

$$(S \subseteq \mathbb{N})$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Let's wind back the clock and see if we can use this to our advantage.



$\exists S.\ (Set(S)\ \land$

S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$\exists S. \; (Set(S) \land$

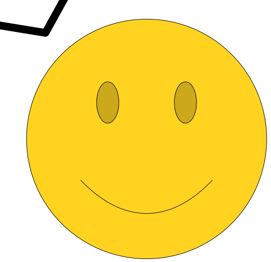
S is the set of all natural numbers

$)$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

The problem was that in the last two cases, we kept mistranslating this blue statement, which got us the wrong answer.

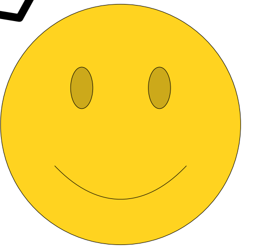


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad\quad S \subseteq \mathbb{N}\ \land \\
&\quad\quad \mathbb{N} \subseteq S \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So what if we translate it like this?



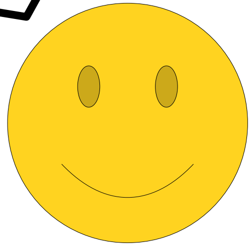
$$\exists S.\ (Set(S)\ \land$$
$$S \subseteq \mathbb{N}\ \land$$

$$\mathbb{N} \subseteq S$$
$$)$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

We can then snap in the two parts of the formulas that we built up earlier…



$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \rightarrow \\
&\qquad\qquad Integer(x) \land \neg Negative(x) \\
&\qquad)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

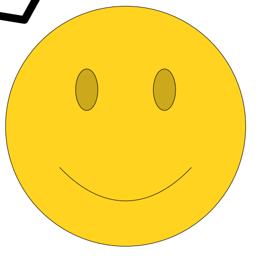
$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

And hey! This actually works!

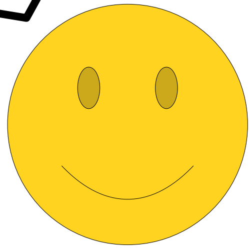


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \rightarrow \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \rightarrow \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

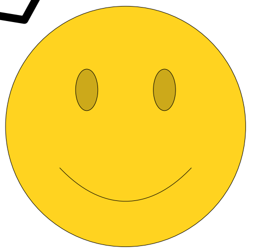
If we choose an S that contains something it shouldn't, this part will catch it…



$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \to \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

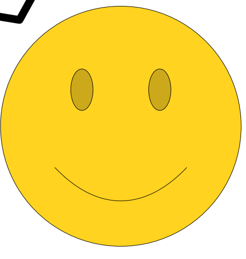


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\qquad \forall x.\ (x \in S \to \\
&\qquad\qquad Integer(x) \land \neg Negative(x) \\
&\qquad) \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So in that sense, we have a working formula!



$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \to \\
&\qquad\qquad Integer(x) \land \neg Negative(x) \\
&\qquad)\ \land \\
&\qquad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad\qquad x \in S \\
&\qquad) \\
&)
\end{aligned}
$$

Available Predicates:

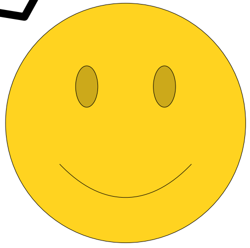
$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \to \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad)\ \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Look at these two implications.
Notice anything about them?

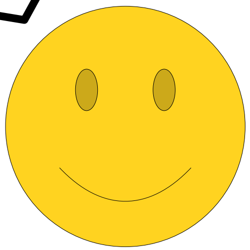


$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x) \land \neg Negative(x) \\
&\quad) \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\quad\quad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Except for the fact that the antecedent and the consequent have been swapped, they're the same!

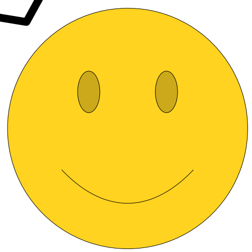


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \to \\
&\qquad Integer(x) \land \neg Negative(x) \\
&\quad)\ \land \\
&\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
&\qquad x \in S \\
&\quad) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

And hey… don't we have a special symbol to say that $A \to B$ and that $B \to A$?

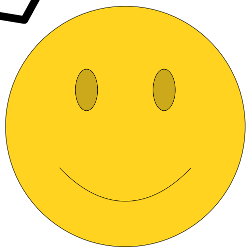


$$
\begin{array}{l}
\exists S.\ (Set(S) \land \\
\quad \forall x.\ (x \in S \to \\
\quad\quad Integer(x) \land \neg Negative(x) \\
\quad) \land \\
\quad \forall x.\ (Integer(x) \land \neg Negative(x) \to \\
\quad\quad x \in S \\
\quad) \\
)
\end{array}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

So as a final step, let's take this formula and rewrite it using the biconditional connective.

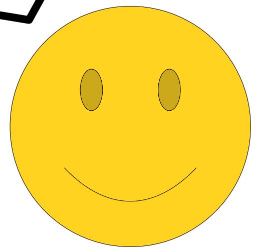


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \leftrightarrow Integer(x) \land \neg Negative(x)) \\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

That ends up looking like this.

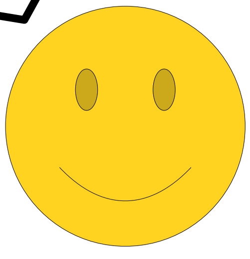


$$\exists S.\ (Set(S) \land \\ \quad \forall x.\ (x \in S \leftrightarrow Integer(x) \land \neg Negative(x)) \\ )$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

This single biconditional contains everything we need.

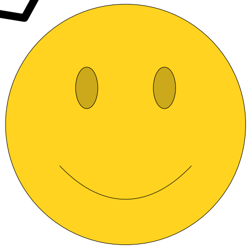


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land\\
&\qquad \forall x.\ (x \in S \leftrightarrow Integer(x)\ \land\ \neg Negative(x))\\
&)
\end{aligned}
$$

Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

In the forwards direction, it says "everything in S needs to be a natural number."



$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\qquad \forall x.\ (x \in S \leftrightarrow Integer(x)\ \land\ \neg Negative(x)) \\
&)
\end{aligned}
$$

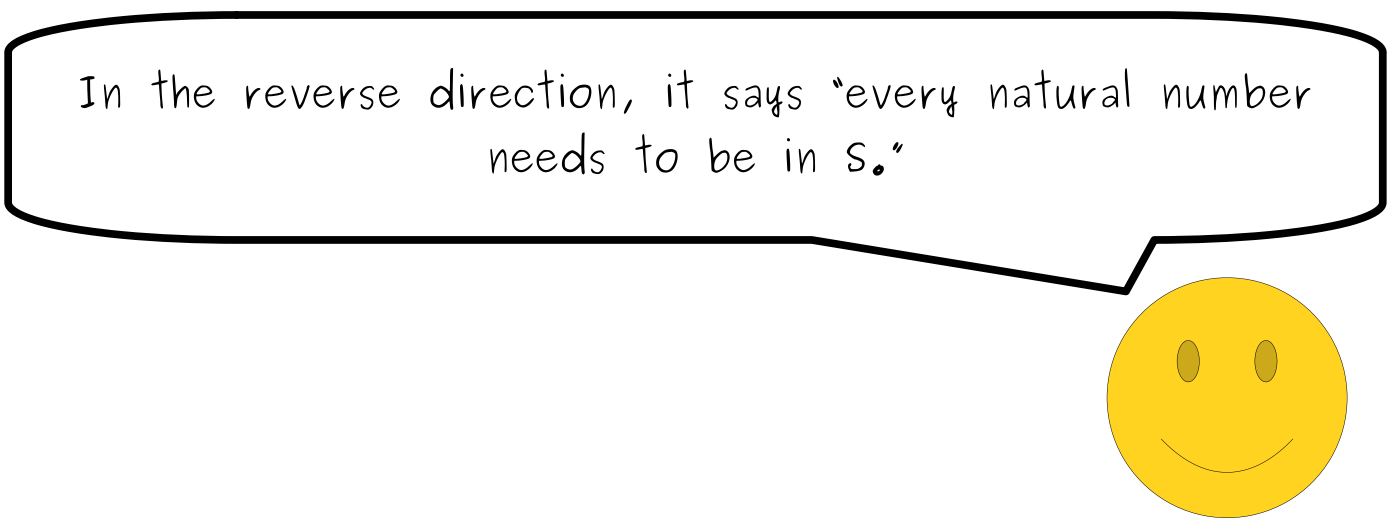
Available Predicates:

$Set(x)$

$x \in y$

$Integer(x)$

$Negative(x)$

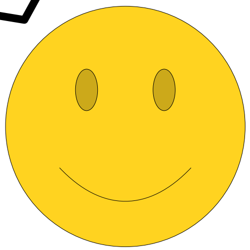


$$
\begin{array}{l}
\exists S.\ (Set(S)\ \land \\
\qquad \forall x.\ (x \in S \leftrightarrow Integer(x) \land \neg Negative(x)) \\
)
\end{array}
$$

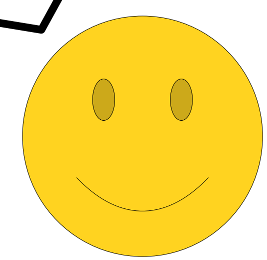
Available Predicates:

$Set(x)$
$x \in y$
$Integer(x)$
$Negative(x)$

Generally, if you're trying to write a statement in first-order logic that says that some set exists (which, hypothetically speaking, might happen sometime soon), you might find yourself using a biconditional to pin down the elements of the set. It's an easy way to say "the set contains precisely these elements."



Wow! We've covered a ton in this guide. Before we wrap up, let's briefly recap the major themes and ideas from what we've seen here.



“All *P*s are *Q*s.”

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

“Some *P*s are *Q*s.”

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

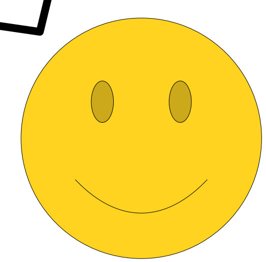
“No *P*s are *Q*s.”

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

“Some *P*s aren't *Q*s.”

$\exists x.\ (P(x) \land \neg Q(x))$

First, we saw these four basic statement building blocks. These are idiomatic expressions in first-order logic – the same way that a for loop over an array is idiomatic in most programming languages – and are extremely useful in assembling more complex statements.



"All *P*s are *Q*s."

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

"Some *P*s are *Q*s."

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

"No *P*s are *Q*s."

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

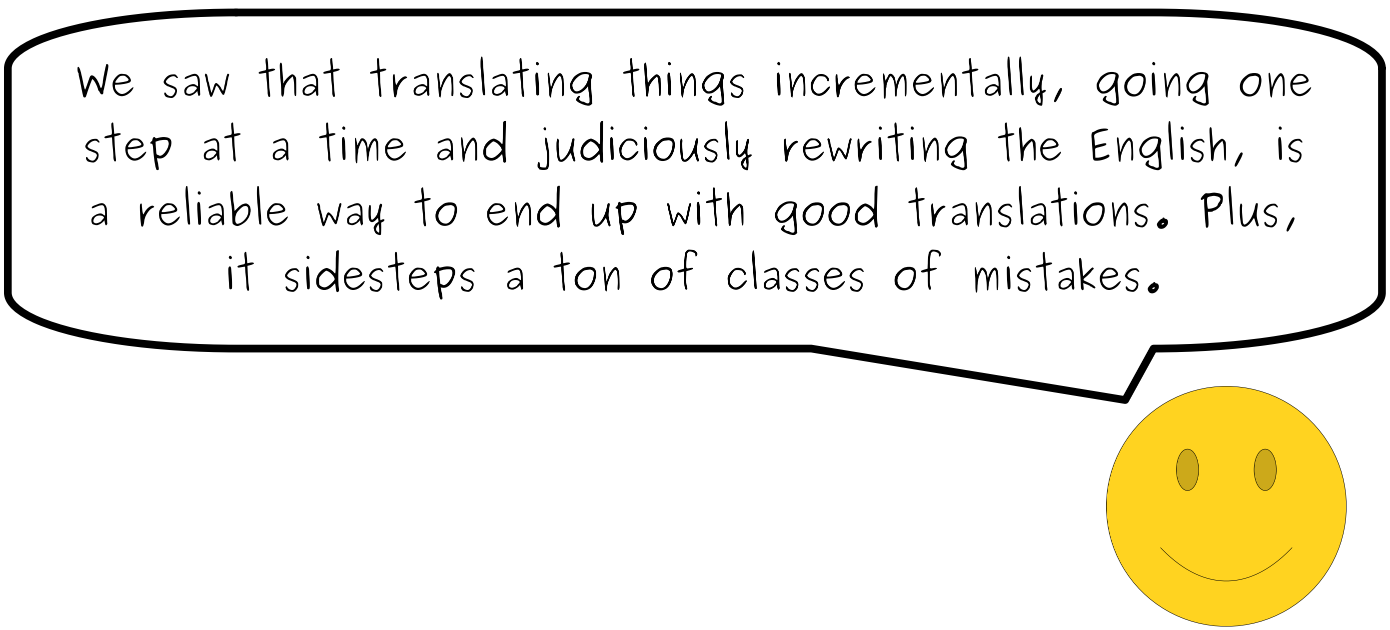
"Some *P*s aren't *Q*s."

$\exists x. \ (P(x) \land \neg Q(x))$

$\forall x. \ (Person(x) \to$

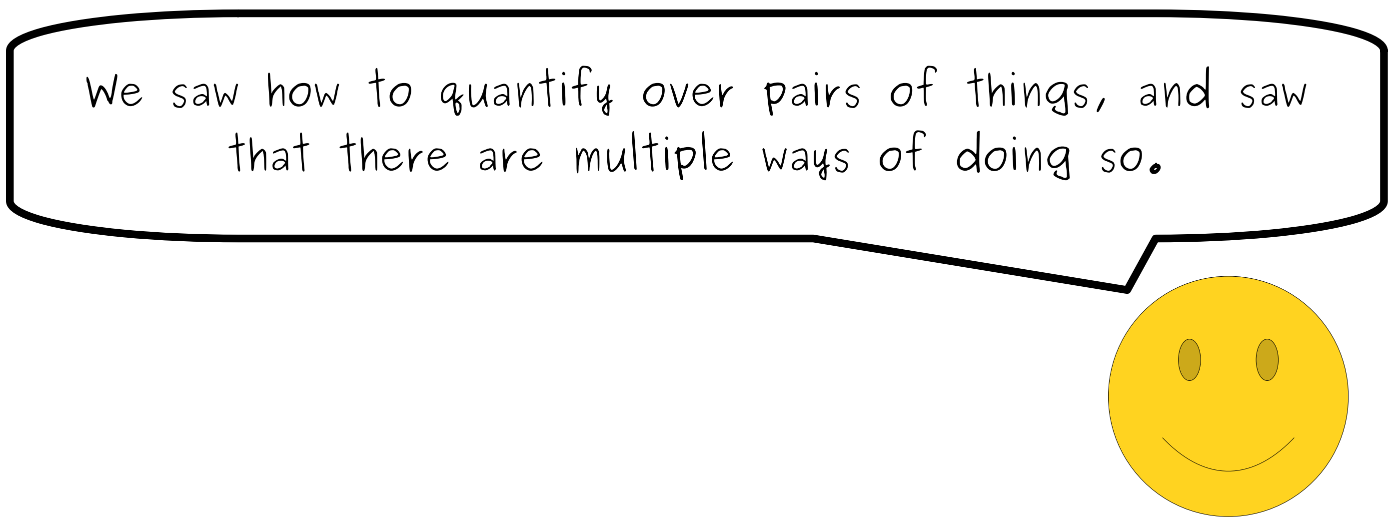
x loves at least one corgi y

$)$



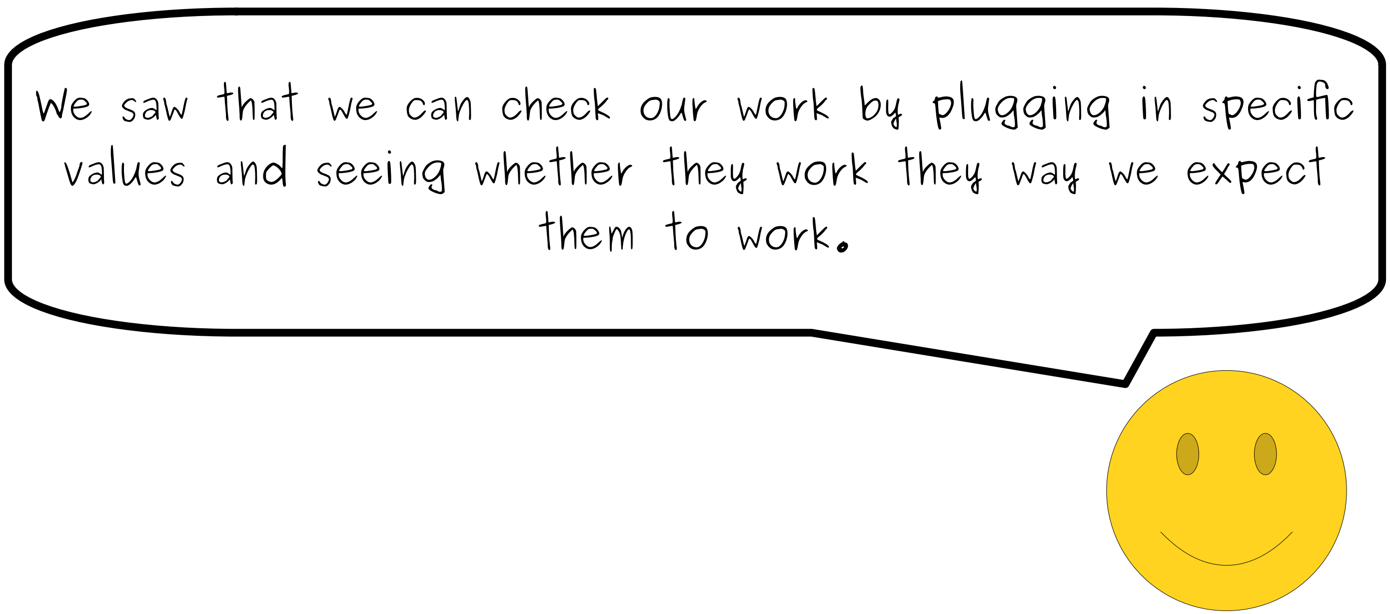
$$
\begin{aligned}
&\forall x.\ (\mathit{Pancake}(x) \to \\
&\quad \forall y.\ (\mathit{Pancake}(y) \to \\
&\quad\quad \mathit{TasteSimilar}(x, y) \\
&\quad) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\forall x.\ \forall y.\ (\mathit{Pancake}(x) \land \mathit{Pancake}(y) \to \\
&\quad \mathit{TasteSimilar}(x, y) \\
&)
\end{aligned}
$$

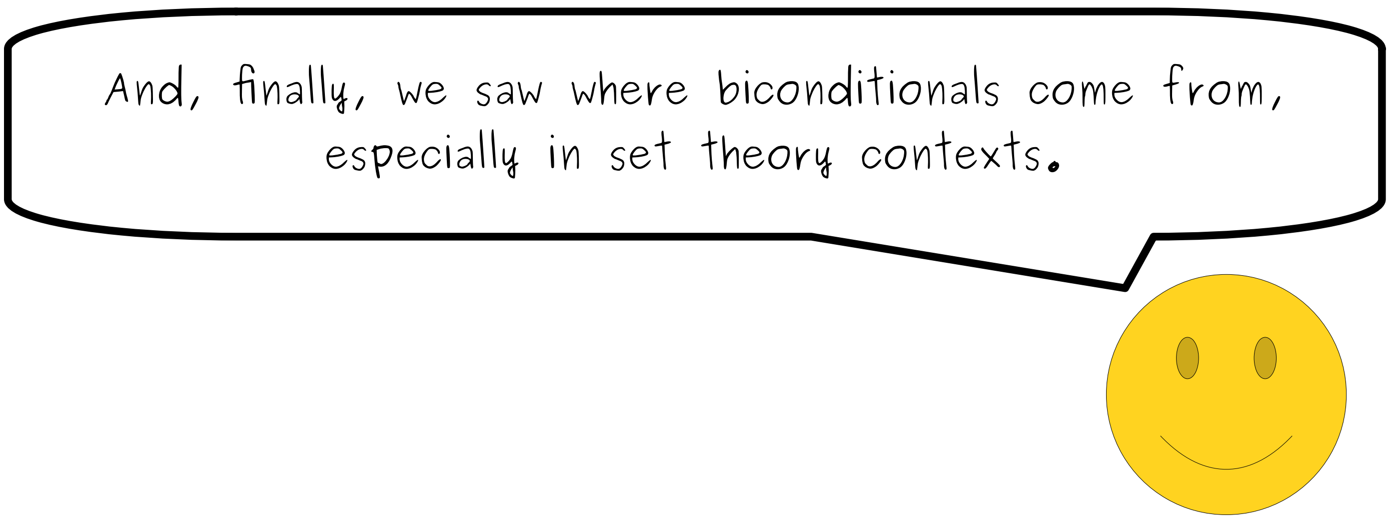


$$
\begin{aligned}
&\exists S.\ (Set(S)\ \land \\
&\quad \forall x.\ (x \in S \to \\
&\quad\quad Integer(x)\ \land\ \neg Negative(x) \\
&\quad) \\
&)
\end{aligned}
$$

$$\textit{Choose } S = \{137\}.$$



$$
\begin{aligned}
&\exists S.\ (Set(S) \land \\
&\qquad \forall x.\ (x \in S \leftrightarrow Integer(x) \land \neg Negative(x)) \\
&)
\end{aligned}
$$




Hope this helps!
Please feel free to ask questions if you have them.


Did you find this useful? If so, let us know! We can go and make more guides like these.