MATH 113: 3/12 WORKSHEET

Truth-functional logic has limited expressiveness. For many important applications of logic, most notably mathematics, we need more. *First-order logic* provides this this extra expressiveness.

We keep the prior grammar of logical connectives: $\wedge$, $\vee$, $\neg$, $\to$, and $\leftrightarrow$. We expand the grammar by adding these new elements.

- **Names and variables for objects.** We use lowercase letters from earlier in the alphabet for names and lowercase letters from the end of the alphabet for variables. For example, a, n, or c_7 would all be names, while x or y would be variables.
- **Predicates.** We use capital letters for predicates—things that may be true of one or more objects. For example, we might use $E(x)$ to stand for "x is an even number".
- **Quantifiers.** We add two quantifiers which let us talk about how many objects satisfy such and such. The *universal quantifier* $\forall$ ("for all" or "every") says that all objects satisfy such and such. The *existential quantifier* $\exists$ ("there exists" or "some") says that some object satisfies such and such. For example, $\forall x\ E(x)$ says that every object is an even number while $\exists x\ E(x)$ says that some object is an even number.

While these are the main new concepts to understand first-order logic, there's others we need.

Domains.

If you're talking about physical reality, there are no unicorns so $\exists x\ U(x)$ would be false, where $U(x)$ means "x is a unicorn". But maybe we're instead talking about your Dungeons and Dragons game, where there are unicorns. In general, when using first-order logic we care about the *domain* we are talking about.

A *domain* is a collection of objects (usually assumed at least one object). We pick a domain to have only the objects of interest to us. For example, if you're doing mathematics then your domain wouldn't include humans or chairs. Each name picks out exactly one object in the domain, but an object can have one name, many, or none at all.

Sometimes the domain is left implicit. But it is good practice to be explicit about what your domain is.

Recall: "every A is B" is expressed as $\forall x\ A(x) \to B(x)$ and "some A is B" is expressed as $\exists x\ A(x) \wedge B(x)$.

Example. Suppose we are talking about numbers, so our domain is the collection of real numbers like 0 or -2 or π. What do we do if we want to talk just about integers (whole numbers)? For example, "every integer is either even or odd" is true, but "every real number is either even or odd" is not, because π is neither.

The solution is we can introduce a predicate to cut down on our domain. Let's have $\mathbb{Z}(x)$ mean "x is an integer" ($\mathbb{Z}$ because that's the usual symbol for the set of integers—Z from the German word "Zahlen"). Then we can express "every integer is either even or odd" as:

$$\forall x\ [\mathbb{Z}(x) \to (E(x) \lor O(x))].$$

(1) How would you express the "there is an integer which is not even"?
(2) How would you express "there is a non-integer which is neither even nor odd"?
(3) How would you express "every odd number is an integer"?

Identity.

A special predicate is the *identity* predicate, saying that x and y are identical—the same object. Borrowing from mathematics we write this as $x = y$. Unlike other predicates where the same letter could be reinterpreted to refer to a different concept, with the identity predicate we only allow this interpretation. That is, $x = y$ always means "x and y are the same object".

One use of the identity predicate is to say that different names give the same object. For instance, suppose j and w are both names to refer to the object Julia Williams. Then $j = w$ expresses that fact.

We write $x \neq y$ as an abbreviation for $\neg(x = y)$.

Counting.

Equipped with the identity predicate now we can count. For example, here is how we express that our domain only has one object:

$$\exists x\ \forall y\ y = x.$$

Similarly, suppose $U(x)$ means "x is a unicorn" and we want to express that there is only one unicorn in our domain. We could do this by writing:

$$\exists x\ [U(x) \land \forall y\ (U(y) \to y = x)].$$

We could express that our domain has at least two objects by writing

$$\exists x\ \exists y\ x \neq y.$$

(1) How do you express that the domain has exactly two objects?
(2) How do you express that there are at least two unicorns?
(3) How do you express that there are exactly two unicorns?
(4) How do you express that the domain has at least three objects? Exactly three objects?
(5) How do you express that the domain has at most two objects?
(6) Can you generalize the pattern and explain how to express the domain has at least n objects? Exactly n objects?