[1] A quantifier is just an operator that applies to a function. [2] The upside down A is called the universal quantifier, pronounced “for all”. It applies to functions that have a binary result; in other words, it applies to predicates. And the result of applying a universal quantifier to a predicate is binary. The result can be thought of as applying the predicate to all its domain elements, and then conjoining all those results together. [3] Here's an example. For all r in rat, r is greater than or equal to 0. In this example, the result is false. [4] The backwards E is called the existential quantifier, pronounced “there exists”. In my opinion, that's a very unfortunate pronunciation, because it has nothing to do with the existence of anything. Maybe it would be better to say for some. Like for all, it's an operator that applies to predicates, with a binary result. The result can be thought of as applying the predicate to all its domain elements, and then disjoining all those results together. [5] Here's an example. There exists n in nat such that n equals 0. Or, for some n in nat, n equals zero. In this example, the result is true. [6] The Greek capital sigma is the summation quantifier. It applies to functions that have a numeric result, and its result is numeric. The result can be thought of as applying the function to all its domain elements, and then adding all those results together. [7] Here's an example. The sum, as n varies over nat plus 1, of 1 over 2 to the n. So that's a half plus a quarter plus an eighth, and so on, and the result is 1. [8] The Greek capital P is the product quantifier. By the way, it is Pee, not Pie. Ask anyone who speaks Greek. It applies to functions that have a numeric result, and its result is numeric. The result can be thought of as applying the function to all its domain elements, and then multiplying all those results together. [9] Here's an example. The product, as n varies over nat plus 1, of 4 n squared over 4 n squared minus 1. And the result is pee by 2. [10] Next we have the maximum quantifier, which some people call the least upper bound. [11] An example is the maximum, as x varies over the rationals, of x times 4 minus x. And the result is 4. [12] And finally, the minimum quantifier, or greatest lower bound. [13] An example is the minimum, as n varies over the positive naturals, of 1 over n. And the result is 0.

[14] There are a couple of slight abbreviations we will use. The first is to stop writing the scope brackets when a quantifier is applied to a function. Here are two examples. I really don't like this abbreviation at all. The scope brackets are useful for showing the scope. But we're bowing to tradition here, and the tradition is to write the quantifier beside the variable. And the scope is everything to the right, up to a big equals or big implies sign, or the end of the expression, or an enclosing parenthesis. The other abbreviation [15] is to group the variables when there is a repeated quantification with the same domain. For all x and y in rat, something, really means for all x in rat, for all y in rat, something. For another example, the sum as n and m vary over 0 to 10 of n times m really means the sum as n varies over 0 to 10 of the sum as m varies over 0 to 10 of n times m.

[16] Here are the axioms defining these quantifiers. I've put them all on one page so you can see the pattern. The first axiom for each quantifier shows its effect on the null domain. The middle axiom for each quantifier has a one element domain, and the last axiom for each quantifier has a union domain. [17] For the null domain, the result is always the identity element for the operator that the quantifier is based on. [18] Most people don't have any trouble with this one. The sum of no numbers is 0. If you're adding things up, you initialize the variable to 0. [19] If you're multiplying things together, you have to initialize the variable to 1. So the product of no numbers is 1. [20] You probably agree that there isn't an element in the empty domain with property b. And you've probably written a search loop that began by initializing found to false. The disjunction of no binary values is false. But you might have trouble with [21] this one. All elements in the empty domain have property b, whatever b is. Well, if it helps, you can say that there aren't any elements in the empty domain that don't have property b. So they all have it. All zero of them. If that helps. Anyway, the conjunction of no binary values is true. [22] The maximum of no numbers is

minus infinity because minus infinity is the identity for maximum. [23] And the minimum of no numbers is infinity because infinity is the identity for minimum.

[24] For the one element domain, the result is always the result of applying the function to that one element. The sum of one number is that number. And so on. [25] For a union domain, that's where the operator that the quantifier is based on comes in. For all v in the union of A and B means for all v in A and for all v in B. There exists v in the union means there exists v in A or there exists v in B. [26] Sum and product are slightly different from for all and there exists. That's because conjunction and disjunction are idempotent, but sum and product aren't. To add up over the union of A and B, you have to add over A, and add over B, but then you might have added some things twice, so you have to subtract things over the intersection. And product is similar. Maximum and minimum are idempotent, so they're just like for all and there exists.

There's one last quantifier that we need. [27] The solution quantifier. It applies to a predicate and gives its solutions. If the predicate has an empty domain, [28] there are no solutions in that domain. If the predicate has a one element domain, that one element may or may not be a solution. [29] If it is, then the result is that element. If it isn't, then there are no solutions of the predicate in that domain. And when the domain is a union [30] the solutions are the union of the solutions for the parts of the union. [31] Here's an example. I pronounce it: those i in int such that i squared equals 4. And the result is [32] minus 2 and 2. It's the formal way of saying what variable we're solving for, and what domain we're looking in. We're solving an equation here, but an equation is just a special case of binary expression. This [33] next example solves an inequality. Those n in nat that are less than 3. And the result is [34] 0 1 and 2. These are bunch equations, and if we [35] apply set formation, we get set equations. And once again, just for the sake of allowing traditional mathematical notations, [36] we can leave out the solutions quantifier in this one context. I'm sure you've used this quantifier before, at least in this context, but maybe you didn't know you were using it. There are lots of laws about quantifiers. So have a look in the back of the textbook and see what they are.

[37] I want to end this lecture with a very important message. An expression talks about its nonlocal variables. This expression can be read as: there exists an n in nat such that x equals 2 times n. And that's true if and only if x is an even natural. So that's what it's saying. It's saying something about x. The n is there just to help say that. It can't be talking about n, because we could rename n to some other variable and get an equivalent expression. So an expression talks about its nonlocal variables.