The technique of derived variables comes up in analyzing state machines. So let's just take a quick look at it together. So a derived variable is simply a function on the states of a state machine that assigns some value to the states. So it's just that kind of a function mapping.

If the values happen to be, say, the non-negative integers, it's called non-negative integer value, but it could be real value, complex value, and even take on other kinds of odd kinds of values, not necessarily numerical. No pun there-- not odd numbers, but unusual values.

So let's look at the example of the robot on the grid. The states were pairs of non-negative integers giving the coordinates of where the robot was. And one of the derived variables that we found was real useful was the sum value, sigma, of a state, which is defined to be x plus y. And this would be a non-negative integer valued derived variable.

So the word "derived" comes because we're making it up. It's not part of the specification of the state machine or part of the program that defines the state machine. So in the robot example, the actual states were composed of the two coordinates x and y, but the derived variable that we made up was their sum of signal. Another useful derived variable for that robot example was the parity of sigma, whether or not the number was even or odd. So sigma is a 0, 1 valued variable, which takes the value 0 if the sum is even and 1 if the sum is odd.

So in the case of fast exponentiation, we looked at the actual variable z, which was part of the invariant and a crucial part of the program. And what we noticed about z was that z was a strictly decreasing and natural number valued variable. As a matter of fact, we noticed that it halved at each step. But its values were non-negative integers, and it's strictly decreasing at every step.

So that implies by the Well Ordering Principle that it will take a minimum value. And what we know about the minimum value of a strictly decreasing variable is that the algorithm is stuck, because once z has reached its minimum value, if the machine took another step, then it would get smaller. So it means that the algorithm has to terminate.

So this gives you a general methodology for proving termination-- find a non-negative integer valued strictly decreasing variable guarantees the program stops. As a matter of fact, we can say sometimes how long it will take for the program to stop. As we saw with fast exponentiation, it took not z, which was the obvious bound, but in fact, log of z, because z not only went down at every step, it got halved at every step. So in general, the concept of a strictly decreasing variable is one-- as shown here-- that at every step of the state machine, at each transition, it gets strictly smaller.

A related idea is a weakly decreasing variable. These are not necessarily useful for proving termination, but they are often useful, as you'll see as we progress through the term-- examples where it helps you analyze the behavior of the algorithm. So a weakly decreasing variable is one which goes down or stays constant. It never gets larger.

So if we looked at the example of sigma, the sum of the coordinates, that's up and down all over the place. It's neither increasing nor decreasing. The other extreme is the parity variable pi, which was the 0 or 1 according to whether or not the sum of the coordinates was even or odd. And pi is a constant, and that means that it's both weakly increasing and weakly decreasing in the degenerate sense that weakly increasing is allowed to stay the same. In fact, something is weakly increasing and weakly decreasing if and only if it's a constant.

By the way, we used to call weakly decreasing variables "non-increasing," which is the standard terminology in the field. In calculus, you talk about non-increasing functions. And we just found that it caused a lot of confusion, because you have to remember that non-increasing is not the same as not increasing. So there's an example of a function that is not increasing, but it's certainly not non-increasing. And if that didn't register, I'll let you think about it.

By the way, this method of proving termination by finding a strictly decreasing natural number valued variable generalizes straightforwardly to a variable which takes on values from a well-ordered set of real numbers. Remember, a well-ordered set of real numbers, one of the definitions of it is that it's a set of numbers where it's impossible to find an infinite decreasing sequence of values-- w0 less than w1 less than w2 less than w1 going on forever. If that can't happen, then the set is called well ordered.

Of course, the non-negative integers are the most obvious basic case, but there are a bunch of others described in the notes. And in general, the termination principle is that if you can find a strictly decreasing variable of derived variable whose values always come from a well-ordered set, that also is a way to prove termination. That's going to guarantee termination for the same reason that the variable will have to take a minimum value. That's the other definition of well ordered. And when it does, the machine can't move anymore.