PROFESSOR: We've mentioned the P equals NP question a number of times now as the most important question in theoretical computer science, and we've said that one way to formulate it is exactly to ask whether there's an efficient that is polynomial-time procedure to test whether or not a formula in propositional logic is satisfiable.

Now, why is that such an important problem? We're not just logicians and we want to know whether or not some formula is satisfiable. How did it take on this enormous importance and apply to so many fields? And illustrating how you could use a satisfiability tester to factor efficiently is a good hint about why it is that all sorts of things reduce to SAT and why it, in fact, is such a centrally important problem.

So let's suppose that we have a satisfiability tester and use it to find how to factor a number n. Now, the observation begins with how you use a SAT solver is that you can begin by writing or observing that it's easy enough to design a digital circuit that multiplies, that does arithmetic multiplications. In other words, it's got some number of bits reserved for an input x, a k bits, and another k bits for an input y, and it's got 2k output lines that produce the digits of x times y. You might need one extra digit, but never mind that.

So this is a multiplier circuit takes an x, a k bit x in and a k bit y in and it spits out the product, which is another 2k bit number, and this is not a terribly big circuit. The naive way to design it would use a number of gates and a number of wires that was about quadratic in the number k. It's easy enough to design one of these things where the size is literally bounded by 5 times k squared, maybe plus a constant.

And so this definitely a small polynomial. Given the number of bits that I'm working with, it's easy enough to build this multiplier circuit.

Now, suppose that I have a way to test satisfiability of circuits. How am I going use this multiplier circuit to factor? Well, the first thing I'm going to do is let's suppose the number that I'm factoring is n and is the product of two primes, p and q. Those are the kinds of n's that we've been using in RSA, and let me also observe that it's very easy to design an n tester-- that is, a little digital circuit that has 2k input lines and produces on its one output line precisely when the input is the binary representation of n.

So let's attach this equality tester that does nothing but ask whether it's being fed the digits of

n as input and it produces an output, 1 for n and 0 if the input pattern is and the digital representation, the binary representation of anything other than n. That's another trivial circuit to build.

So we put those two together, and now watch what happens. I'm going to take the circuit and set the first of the input bits to 0, and then I'm going to ask the SAT solver the following question-- is there a way to set the remaining input bits other than 0? So I've set the first one to 0. What about these other bits? The SAT solver can tell me whether or not it's possible to get a 1 out of this circuit with the 0 there fixed.

So let's ask the SAT solver what happens, and the SAT solver says, hey, yes, there is a way to fill in the remaining digits and get an output 1. Well, what does that tell me? Well, it tells me that there is a factor that starts with 0, so let's fix the 0 based on the fact that it's possible for me to fill in the remaining digits with the bits of factors x and y that equal n.

Let's try to set the second input bit to 0 and see what happens. Well, we'll ask the SAT tester, is it possible now to fill in the remaining digits to get the two numbers x and y that multiply and produce n and therefore output 1? And the SAT tester says, no, this is an unsatisfiable circuit. You can't get a 1 out of it any more. That tells me that I have to set the second bit to 1 in order to have a factor of n where the x and y will multiply together to be n.

All right, fine. Go to the third bit, ask whether or not 0 works. The SAT tester says, let's say, yes. So then I could fix 0. I now know the first all three bits of x. And of course, I go on and in 2k SAT tests, I know exactly what p and q are, and I have, in fact, found the factors p and q.

So that wraps that one up. That's how you use a SAT tester. You just do the SAT test 2k times and you factored this 2k bit number. And of course, if the SAT test is polynomial in k, then doing it 2k times just is also polynomial in k with one degree higher.

Now, the satisfiability problem, as we formulated, was a problem about formulas that as you wrote out a propositional formula and asked whether or not it was satisfiable, and I'm instead asking about satisfiability of binary circuits. But in fact, as we did in some early exercises, you can describe a binary circuit by assigning a fresh variable to every wire in the circuit and then writing a little formula around each gate which explains how the input wires to that gate are related to the output wire of that gate. And that little formula explains that wiring of that gate, and you take the "and" of all those formulas and you have a formula that is describing the

structure of the circuitry, and in fact the formula is satisfiable if and only if the circuit can produce an output 1.

So we really have by assuming that I could test satisfiability of formulas, I can therefore test satisfiability of circuits, and therefore I can factor. So that's the simple trick to find a propositional formula that's equisatisfiable to the circuit-- if the circuit produces output 1 if and only if this formula of about the same size as the circuit is satisfiable. And that's the last piece that I needed in order to completely reduce the factoring to the satisfiability problem, and you could see that this is actually a general method that will enable you to reduce most any kind of one-way function to a few SAT tests.