[1] Probabilistic programming is about programs where we don't care about the exact result, but about the probability of seeing particular results. It has applications in cryptography, and in writing secure and reliable systems, and in game playing. I'll explain soon, but I have to start with a quick review of basic probability. [2] A probability is a real number between 0 and 1 inclusive. I'd like to mention that that's a completely arbitrary decision that someone made long ago. They could have decided that it's a number from minus 1 to plus 1, or from minus infinity to plus infinity, and then we would have a very different theory. I'll stick with the standard, but I want you to know that it was just a choice someone made. [3] So the probabilities are those r in real such that 0 is less than or equal to r, which is less than or equal to 1. Probability 1 represents something that's certainly true, and probability 0 represents something that's certainly false. So for this lecture only, I make [4] 2 axioms that say true and 1 are the same, and false and 0 are the same. Numbers between, represent something that might be true or might be false. For example, a half represents something that is equally likely to be true or false. Now we can write all binary operations arithmetically. [5] We can write not x as 1 minus x. We can write x and y as x times y. And we can write x or y as x minus x times y plus y. And similarly for any other binary operator.

[6] A distribution is an expression whose value is a probability, and if you sum up all values of the expression, you get 1. If the variables in the expression are real valued, we have to integrate, so I'll stick to integer and binary variables so we can just add. A distribution can be used to [7] tell the frequency of occurrence of the values of its variables. Or, it can be used [8] to say how well we know what to expect or predict for the values of variables. [9] A distribution says what the probability is for each state of the variables. [10] Here's an example. The expression 2 to the minus n is a distribution, where n is a variable in nat plus 1, the positive naturals, because, for all values of n, 2 to the minus n is a probability, and, if you add up 2 to the minus n for all values of n, you get 1. [11] 2 to the minus n says that n has value 3 with probability one-eighth, because 2 to the minus 3 is one-eighth. It also tells us the probability that n has any other value. [12] Here's a distribution with 2 variables, n and m, both positive naturals. Whatever values you give to n and m, 2 to the minus n minus m is a probability, and if you sum it for all values of n and m you get 1. [13] This distribution says that the probability that n has value 3 and m has value 1 is one-sixteenth. And similarly it tells you the probability of any other combination of values of n and m.

[14] The specification n primed equals n plus 1 tells us, among other things, that if n is 5, then n primed is 6. That means, if n is 5, then n primed is 6 [15] with probability 1, and it's anything else [16] with probability 0. This equation is either true or false, and that's 1 and 0, so [17] it's a probability. But there are many pairs of values for n and n primed that make this equation true or 1, so [18] the sum is infinite, and [19] it's not a distribution of n and n primed. But, for any particular value of n, [20] there's only one value of n primed that makes the equation true or 1, so [21] for each value of n, that specification is a distribution of n primed. And likewise [22] any implementable deterministic specification is a distribution of its final state.

[23] In our ordinary, non-probabilistic programming, we have defined these basic programming expressions. Now we [24] convert them into probability distributions like this. ok and the assignment haven't really changed; they've just been re-expressed arithmetically. Also the conditional composition and the sequential composition haven't changed when their operands are binary. But now they apply also to operands that are probabilities. [25] Here's an example. It says: if one-third, which sounds funny. It means: with probability one-third we assign x the value 0, and with the remaining probability, which is two-thirds, we assign x the value 1. So we might be assigning 0, but more likely we're assigning 1 to x. Let's apply [26] the definitions, supposing x is the only variable. What does this say? Well, we can evaluate it for various values of x prime. [27] When x prime is 0, we get this. 0 equals 0 is true [28], which is 1, and 0 equals 1 is false, which is 0, so the answer is [29]

one-third. And that's the probability that x prime is 0. [30] When x prime is 1, we get this. 1 equals 0 is false [31], which is 0, and 1 equals 1 is true, which is 1, so the answer is [32] two-thirds. And that's the probability that x prime is 1. [33] When x prime is 2, we get this. 2 equals 0 is false [34], which is 0, and 2 equals 1 is false, which is 0, so the answer is [35] 0. And that's the probability that x prime is 2. And the same for any other value.

[36] Now let's make the example a little more complicated. After the first line, x is either 0 or 1. If x equals 0 then with probability a half add 2 and with probability a half add 3, and if x isn't 0 then with probability a quarter add 4 and with probability 3 quarters add 5. If you want to know how that comes out, just [37] apply the definitions I gave you for assignment and if and dot. We [38] already did the first line, and the [39] rest is similar. Then [40] do the sum, or take my word, and it comes out like this. In the end, x is 2 with probability one-sixth, and x is 3 with probability one-sixth, and x is 5 with probability one-sixth, and x is 6 with probability one-half, and x is anything else with probability 0.

[41] The next topic is average value. Let P be any distribution of final values, and let e be any numeric expression. Then after distribution P, the average value of e is P dot e. [42] Here's an example. Let variable n vary over nat plus 1 according to distribution 2 to the power minus n. Then the average value of n squared is, well, you write down the distribution with a prime on n, and then dot n squared. Now you just [43] apply the definition of dot. I'll leave you to figure out how to do the sum, and just tell you that [44] the answer is 6.

[45] Here's our previous programming example. After that distribution, what's the average value of x? Just [46] follow it with a dot and x. And [47] we already simplified the distribution to this, so it's this followed by dot x. Now [48] use the definition of dot, and [49] do the sum. And this time the sum is easy. When x double-prime is 2, we get a sixth, plus 0, plus 0, plus 0, times 2. When x double-prime is 3 we get 0 plus a sixth plus 0 plus 0 times 3. When x double-prime is 5 we get 0 plus 0 plus a sixth plus 0 times 5. When x double-prime is 6 we get 0 plus 0 plus 0 plus a half times 6. And when x double-prime is anything else we just get 0. And that works out to [50] 4 and 2 thirds. So that's the average value of x according to that probability distribution. [51]

[52] Here's a similar formula. After distribution P, the probability that binary expression b is true is P dot b. The reason the two formulas are so similar, in fact they're the same, is that [53] probability is just the average value of a binary expression. So computing a probability is just computing an average. [54] Here's that same programming distribution we've been using, and then we ask, what's the probability that x is greater than 3? To find out, just put dot x greater than 3 after it. [55] Here's the calculation. I won't take time to read through it, but it comes out 2 thirds. And you can figure out the probability of anything else the same way.

In many programming languages there's a [56] random number function. I'll write rand n for a function that returns a random natural number from 0 to n. Well, it may be called a function, but it's not, and the use of functional notation for something that isn't a function leads to problems. [57] Is rand n equal to rand n? Probably not. [58] Is rand n plus rand n equal to twice rand n? Probably not. So we have to get rid of the misleading notation. [59] rand n has value r with probability r is in 0 to n divided by n. For a value of r that is in 0 to n, the probability is 1 over n. For a value of r that is outside 0 to n, the probability is 0 over n. So [60] replace rand n with an integer variable r that has that distribution. Or, if you prefer, [61] replace rand n with a variable whose domain is 0 to n and whose distribution is 1 over n. [62] Here's an example. x gets rand 2 followed by x gets x plus rand 3. First we have to [63] replace the two rands with variables, let's say r and s, with the right distributions. Now we can safely use mathematics. So [64] rewrite the second assignment inside the parentheses and then use the substitution law. Now [65] do the sum, and you get all this, which can be simplified to [66] this, which says x is finally 0 with probability 1

sixth, 1 with probability 1 third, 2 with probability 1 third, and 3 with probability 1 sixth. Actually, in this example, we didn't really need to invent new variables r and s. We could have used the variable x that we already have. [67] So let's start again. [68] Replace rand by using variable x prime. And then [69] use the formula for sequential composition. [70] Do the sum, which takes some time but isn't hard, and you [71] get the same answer as before.

Let's [72] do something that's a bit more fun. We'll play blackjack. Actually, this is a simplified blackjack just to make it shorter, but all the necessary ingredients are here for the full game. In this version, you are dealt a card from a deck; its value is in the range 1 to 13 inclusive. You may stop with just one card, or have a second card if you want. Your object is to get a total as near as possible to 14 , but not over 14 . You can use any strategy you like, and the one we'll use right now is to take a second card if the first is under 7 . So [73] here's the deal. x gets rand of 13 plus 1. That's a number from 1 to 13 inclusive. Then, if x is less than 7, x gets x plus another rand of 13 plus 1. Otherwise leave x alone. The first thing we have to do is [74] get rid of rand. And the [75] next thing is to use the definitions of sequential composition and if. The details don't matter here, so just take my word that you get this, and [76] it ends up like this. It's a complicated expression that tells us the distribution of x prime. Does it say that it's a good strategy to take a second card if the first one is under 7? Let's find out. [77] Let's play the game with 2 different strategies simultaneously, and see which one wins. Player x will play the under n strategy, and player y will play the under n plus 1 strategy, and we'll see who wins. [78] Here are the two cards, c and d. Both players will use the same cards to see which strategy is best. [79] Here's what x does. If the first card c is less than n, then x gets the sum of the 2 cards c plus d else x gets just the first card c. And [80] y says if the first card is less than n plus 1 then y gets both cards else y gets the first card. Next, I'm putting the [81] condition that x wins, and we'll see how probable that is. x wins if x has more than y, but not over 14, or if x isn't over 14 and y is over 14. Now we calculate. First [82] replace rand. I'll use the variables we already have, so that's c prime is in 1 to 14, and so is d prime, and we multiply by the distribution for each of them, which is 1 over 13. Now it's just math, and if you'll trust me, I'll just [83] show the answer, which is n minus 1 over 169. That's the probability that x wins. We can make the same kind of calculation to find [84] the probability that y wins, and the probability of a tie. If we try a few values for n, we find that [85] under 8 is the best strategy. The real game of blackjack is more complicated than this one, but the same kind of calculation can tell you the best strategy.

That program didn't have any loop in it. I want to do an example with a loop. [86] How about this one. If you repeatedly throw a pair of dice until they are equal, how long does it take? [87] Here's the program. The first die is thrown and its value is assigned to variable u. The second die is thrown and assigned to v. If u equals v then stop, else the clock goes tick and we go round again. What we need for R is the distribution of final times, t prime. The first throw could be any number from 1 to 6. For the second throw, there's a 1 sixth chance it will be the same, and a 5 sixths chance it will be different. So it's 5 sixths for each unequal time, and then 1 sixth for the final time. So [88] I make the hypothesis that t prime has the distribution 5 sixths to the power t prime minus t times 1 sixth . The factor t prime greater than or equal to t is there just to give 0 probability that time will go backwards. I have to prove this hypothesis in the same way we prove any refinement. [89] And I'll start with the [90] right side. u and v are assigned, then if u equals v, then ok, but I'm interested only in the time, I don't care what happens to u and v in the end, so I've written t prime equals t. Else increase t, and there's the hypothesis. In this proof step, I'm going to do two things. I'm going to get rid of rand from the top line, and in the bottom line I'm going to use the substitution law [91]. The top line now says u prime is in 1 to 7, and v and t are unchanged. Then v prime is in 1 to 7 and u and t are unchanged. The bottom line ends with t prime greater than or equal to t plus 1, and, or times, 5 sixths to the power t

prime minus t minus 1, divided by 6. Next, in the top line, I'm getting rid of the sequential composition. [92] It now says u prime and v prime are both in 1 to 7 and t is unchanged, and that's divided by 36. Next I want to do the remaining sequential composition and the **if**, both at the same time, and some simplification. [93] Here's what I get. The summation and the first multiplication are due to the sequential composition. The plus sign and its two operands are due to the **if**. I guess I shouldn't have made such a big step, but I wanted to fit it all on one page. Now we have to do the [94] sum. There are 6 occasions in the sum when t prime is equal to t, namely when u double prime equals v double prime, and 30 occasions when t prime is greater than or equal to t plus 1, namely when u double prime and v double prime are unequal. We can combine [95] them, and we get back the original hypothesis. And that completes the proof.

In textbooks on probability you see carefully reasoned arguments about why something has a certain probability distribution. No matter how careful the arguments are, the result of informal argument is just a hypothesis, and it needs proof. But never do you see a formal proof. Sometimes the columns of popular scientific magazines contain disagreements about probabilities. The problem may have to do with a sequence of activities, with choices to be made. And the best way to formalize that is with a program. And the only way to prove a distribution, and end the arguments, is with a combination of probability theory and programming theory. Probabilistic programming. Now you can do that. [96] By the way, if you want to know [97] the average number of times you need to throw the dice before they come up equal, just follow the distribution with t, and you get [98] t plus 5. So on average, it takes 5 additional throws, after the first one, to get equal dice.