CS 70 Discrete Mathematics and Probability Theory
Fall 2020 HW 9

1 Unprogrammable Programs

Prove whether the programs described below can exist or not.

(a) A program $P(F,x,y)$ that returns true if the program F outputs y when given x as input (i.e. $F(x) = y$) and false otherwise.

(b) A program P that takes two programs F and G as arguments, and returns true if F and G halt on the same set of inputs (or false otherwise).

Solution:

(a) P cannot exist, for otherwise we could solve the halting problem:

```
def Halt(F, x):
  def Q(x):
    F(x)
    return 0
  return P(Q, x, 0)
```

`Halt` defines a subroutine `Q` that first simulates F and then returns 0, that is `Q`(x) returns 0 if $F(x)$ halts, and nothing otherwise. Knowing the output of `P`(F,x,0) thus tells us whether F(x) halts or not.

(b) We solve the halting problem once more:

```
def Halt(F, x):
  def Q(y):
    loop
  def R(y):
    If y = x:
      F(x)
    Else:
      loop
  return not P(Q, R)
```

`Q` is a subroutine that loops forever on all inputs. `R` is a subroutine that loops forever on every input except x, and runs $F(x)$ on input x when handed x as an argument. Knowing if `Q` and `R`

halt on the same inputs is thus tantamount to knowing whether F halts on x (since that is the only case in which they could possibly differ). Thus, if $P(Q,R)$ returns "True", then we know they behave the same on all inputs and F must not halt on x, so we return `not P(Q,R)`.

2 Computations on Programs

(a) Is it possible to write a program that takes a natural number n as input, and finds the shortest arithmetic formula which computes n? For the purpose of this question, a formula is a sequence consisting of some valid combination of (decimal) digits, standard binary operators ($+$, $\times$, the "^" operator that raises to a power), and parentheses. We define the length of a formula as the number of characters in the formula. Specifically, each operator, decimal digit, or parentheses counts as one character.

(*Hint:* Think about whether it's possible to enumerate the set of possible arithmetic formulas. How would you know when to stop?)

(b) Now say you wish to write a program that, given a natural number input n, finds another program (e.g. in Java or C) which prints out n. The discovered program should have the minimum execution-time-plus-length of all the programs that print n. Execution time is measured by the number of CPU instructions executed, while "length" is the number of characters in the source code. Can this be done?

(*Hint:* Is it possible to tell whether a program halts on a given input within t steps? What can you say about the execution-time-plus-length of the program if you know that it does not halt within t steps?)

Solution:

(a) Yes it is possible to write such a program.

We already know one way to write a formula for n, which is to just write the number n (with no operators). Let the length of this formula in characters be l. In order to find the *shortest* formula we simply need to search among formulae that have length at most l.

Since there are a finite number of formulas of length at most l, we can write a program that iterates over all of them. For example, if we treat each character as a byte or an 8-bit number, the whole formula becomes a binary integer of length at most $8l$, so we can simply iterate over all binary numbers up to 2^{8l} and for each one check if it is a valid formula.

For each formula that we encounter we can compute its value in finite time (since there are no loop/control structures in formula). Therefore we can check whether it computes n, and then among those that do compute n we find the smallest one.

(b) Yes. Again it is possible to write such a program.

As before, given a number n, there is one program that we know can definitely write n, which is the program that prints the digits of n one by one. Let the length plus running time of this

program be l. We only need to check programs that have a length of at most l and a running time of at most l, since otherwise their running time plus length would be bigger than l.

Similar to the previous part, we can iterate over all programs of length at most l (by treating each one as a large binary integer and checking each one's validity by e.g. compiling it). For each such program, we then run it for at most l steps. If it takes more time, we stop executing it and go to the next program, otherwise in at most l steps we see its output and we can check whether it is equal to n or not.

Now among all programs that have length at most l and execute for at most l steps and print n we find the one that has the shortest length plus execution time.

3 Kolmogorov Complexity

Compressing a bit string x of length n can be interpreted as the task of creating a program of fewer than n bits that returns x. The Kolmogorov complexity of a string $K(x)$ is the length of an optimally-compressed copy of x; that is, $K(x)$ is the length of shortest program that returns x.

(a) Explain why the notion of the "smallest positive integer that cannot be defined in under 280 characters" is paradoxical.

(b) Prove that for any length n, there is at least one string of bits that cannot be compressed to less than n bits.

(c) Say you have a program K that outputs the Kolmogorov complexity of any input string. Under the assumption that you can use such a program K as a subroutine, design another program P that takes an integer n as input, and outputs the length-n binary string with the highest Kolmogorov complexity. If there is more than one string with the highest complexity, output the one that comes first alphabetically.

(d) Let's say you compile the program P you just wrote and get an m bit executable, for some $m \in \mathbb{N}$ (i.e. the program P can be represented in m bits). Prove that the program P (and consequently the program K) cannot exist.

(*Hint*: Consider what happens when P is given a very large input n.)

Solution:

(a) Since there are only a finite number of characters then there are only a finite number of positive integers that can be defined in under 280 characters. Therefore there must be positive integers that are not definable in 280 characters and by the well-ordering principle there is a smallest member of that set. However the statement "the smallest positive integer not definable in under 280 characters" defines the smallest such an integer using only 67 characters (including spaces). Hence, we have a paradox (called the Berry Paradox).

(b) The number of strings of length n is 2^n. The number of strings shorter than length n is $\sum_{i=0}^{n-1} 2^i$. We know that sum is equal to $2^n - 1$ (remember how binary works). Therefore the cardinality of the set of strings shorter than n is smaller than the cardinality of strings of length n. Therefore there must be strings of length n that cannot be compressed to shorter strings.

(c) We write such a program as follows:

```
def P(n):
  complex_string = "0" * n
  for j in range(1, 2 ** n):
    # some fancy Python to convert j into binary
    bit_string = "0:b".format(j)
    # length should now be n characters
    bit_string = (n - len(bit_string)) * "0" + bit_string
    if K(bit_string) > K(complex_string):
      complex_string = bit_string
   return complex_string
```

(d) We know that for every value of n there must be an incompressible string. Such an incompressible string would have a Kolmogorov complexity greater than or equal to its actual length. Therefore our program P must return an incompressible string. However, suppose we choose size n_k such that $n_k \gg m$. Our program $P(n_k)$ will output a string x of length n_k that is not compressible meaning $K(x) \geq n_k$. However we have designed a program that outputs x using fewer bits than n_k. This is a contradiction. Therefore K cannot exist.

4 Five Up

Say you toss a coin five times, and record the outcomes. For the three questions below, you can assume that order matters in the outcome, and that the probability of heads is some p in $0 < p < 1$, but *not* that the coin is fair ($p = 0.5$).

(a) What is the size of the sample space, $|\Omega|$?

(b) How many elements of Ω have exactly three heads?

(c) How many elements of Ω have three or more heads?
(*Hint:* Argue by symmetry.)

For the next three questions, you can assume that the coin is fair (i.e. heads comes up with $p = 0.5$, and tails otherwise).

(d) What is the probability that you will observe the sequence HHHTT? What about HHHHT?

(e) What is the chance of observing at least one head?

(f) What about the chance of observing three or more heads?

For the final three questions, you can instead assume the coin is biased so that it comes up heads with probability $p = \frac{2}{3}$.

(g) What is the chance of observing the outcome HHHTT? What about HHHHT?

(h) What about the chance of at least one head?

(i) What about the chance of ≥ 3 heads?

Solution:

(a) Since for each coin toss, we can have either heads or tails, we have 2^5 total possible outcomes.

(b) Since we know that we have exactly 3 heads, what distinguishes the outcomes is at which point these heads occurred. There are 5 possible places for the heads to occur, and we need to choose 3 of them, giving us the following result: $\binom{5}{3}$.

(c) We can use the same approach from part (b), but since we are asking for 3 or more, we need to consider the cases of exactly 4 heads, and exactly 5 heads as well. This gives us the result as: $\binom{5}{3} + \binom{5}{4} + \binom{5}{5} = 16$.
To see why the number is exactly half of the total number of outcomes, denote the set of outcomes that has 3 or more heads as A. If we flip over every coin in each outcome in set A, we get all the outcomes that has 2 or less head. Denote the new set as A'. Then we know that A and A' have the same size and they together cover the whole sample space. Therefore, $|A| = |A'|$ and $|A| + |A'| = 2^5$, which gives $|A| = 2^5/2$.

(d) Since each coin toss is an independent event, the probability of each of the coin tosses is $\frac{1}{2}$ making the probability of this outcome $\frac{1}{2^5}$. This holds for both cases since both heads and tails have the same probability.

(e) We will use the complementary event, which is the event of getting no heads. The probability of getting no heads is the probability of getting all tails. This event has a probability of $\frac{1}{2^5}$ by a similar argument to the previous part. Since we are asking for the probability of getting at least one heads, our final result is: $1 - \frac{1}{2^5}$.

(f) Since each outcome in this probability space is equally likely, we can divide the number of outcomes where there are 3 or more heads by the total number of outcomes to give us: $\frac{\binom{5}{3}+\binom{5}{4}+\binom{5}{5}}{2^5}$

(g) By using the same idea of independence we get for HHHTT: $\frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{2^3}{3^5}$
For HHHHT, we get:
$\frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{2^4}{3^5}$

(h) Similar to the unbiased case, we will first find the probability of the complement event, which is having no heads. The probability of this is $\frac{1}{3^5}$, which makes our final result $1 - \frac{1}{3^5}$

(i) In this case, since we are working in a nonuniform probability space (getting 4 heads and 3 heads don't have the same probability), we need to separately consider the events with different numbers of heads to find our result. This will get us:

$$\binom{5}{3}\frac{2^3}{3^5}+\binom{5}{4}\frac{2^4}{3^5}+\binom{5}{5}\frac{2^5}{3^5}$$

5 Ball-and-Bin Counting Problems

Say you have 5 bins, and randomly throw 7 balls into them.

1. What is the probability that the first bin has precisely 3 balls in it?
2. What is the probability that the third bin has at least 3 balls in it?
3. What is the probability that at least one of the bins has precisely 3 balls in it?

Solution:

1. First, choose 3 balls out of 7 to put in the first bin. Each of these 3 balls has $Pr = 1/5$ to be in the first bin, and each of the rest $7-3=4$ balls has $Pr = 4/5$ to NOT be in the firts bin. Thus, the answer is

$$\binom{7}{3}\left(\frac{1}{5}\right)^3\left(\frac{4}{5}\right)^4.$$

2. Extend part (a) results into the sum of exactly $3/4/5/6/7$ balls

$$\sum_{k=3}^{7}\binom{7}{k}\left(\frac{1}{5}\right)^k\left(\frac{4}{5}\right)^{7-k}.$$

3. Use inclusion-exclusion. Let A_i be the event that bin i has exactly 3 balls. Then $\sum_{i=1}^{5}\mathbb{P}[A_i] = 5\binom{7}{3}(1/5)^3(4/5)^4$. We have to subtract the events $A_i \cap A_j$, of which there are $\binom{5}{2}$. We have $\mathbb{P}[A_i \cap A_j] = 7!/(3!)^2(1/5)^6(3/5)$.

   The reasoning behind $\frac{7!}{3!3!}$ is because: we want the 7 balls to be split in exactly 3 balls in A_i, 3 balls in A_j, and last ball wherever else. First assume that all 7 balls are lined up, and A_i takes first three balls, A_j takes the next three balls, so you have 7! ways to order all the balls initially. Then, reduce the over-counted parts since the order of the 3 balls in A_i doesn't matter (divide by 3! since there are this many ways to order the 3 balls), and similarly, the order of the 3 balls in A_j doesn't matter either. Thus, you have $\frac{7!}{3!3!}$.

   Therefore our answer is

$$5\binom{7}{3}\left(\frac{1}{5}\right)^3\left(\frac{4}{5}\right)^4-\binom{5}{2}\frac{7!}{3!3!}\left(\frac{1}{5}\right)^6\frac{3}{5}.$$

6 Monty Hall's Revenge

Due to a quirk of the television studio's recruitment process, Monty Hall has ended up drawing all the contestants for his game show from among the ranks of former CS70 students. Unfortunately for Monty, the former students' amazing probability skills have made his cars-and-goats gimmick unprofitable for the studio. Monty decides to up the stakes by asking his contestants to generalise to three new situations with a variable number of doors, goats, and cars:

(a) There are n doors for some $n > 2$. One has a car behind it, and the remaining $n-1$ have goats. As in the ordinary Monty Hall problem, Monty will reveal one door with a goat behind it after you make your first selection. How would switching affect the odds that you select the car?

(Hint: Think about the size of the sample space for the experiment where you *always* switch. How many of those outcomes are favorable?)

(b) Again there are $n > 2$ doors, one with a car and $n-1$ with goats, but this time Monty will reveal $n-2$ doors with goats behind them instead of just one. How does switching affect the odds of winning in this modified scenario?

(c) Finally, imagine there are $k < n-1$ cars and $n-k$ goats behind the $n > 2$ doors. After you make your first pick, Monty will reveal $j < n-k$ doors with goats. What values of j,k maximize the relative improvement in your odds of winning if you choose to switch? (i.e. what j,k maximizes the ratio between your odds of winning when you switch, and your odds of winning when you do not switch?)

Solution:

Throughout the solution, we will refer to W as the event that the contestant wins, and $\mathbb{P}_S(W)$ and $\mathbb{P}_N(W)$ as the probabilities of this event happening if the contestant is (S)witching or (N)ot switching, respectively.

(a) $\mathbb{P}_N(W) = 1/n$ since only one out of n initial choices gets us the car. Under the switching strategy two things can happen: Either the first choice hits the car, and so switching (to any of the remaining $n-2$ doors) will inevitably get us the goat, or our first choice picks a goat, leaving one of the remaining $n-2$ doors with the car. This sequence of choices—first choosing from one of n doors, then switching to one of $n-2$ remaining doors—gives us a sample space of size $n(n-2)$. If we divide the number of favorable outcomes by the total number of outcomes, we get

$$\mathbb{P}_S(W) = \left(\underbrace{1}_{\text{first choice = car}} \cdot \underbrace{0}_{\text{second choice = car}} + \underbrace{(n-1)}_{\text{first choice = goat}} \cdot \underbrace{1}_{\text{second choice = car}} \right) / \underbrace{n(n-2)}_{\text{total \# of choices}}$$
$$= \frac{n-1}{n(n-2)} = \frac{1}{n} \cdot \frac{n-1}{n-2}$$

which is larger than $\mathbb{P}_N(W) = 1/n$ (ever so slightly so the larger n becomes, which demonstrates the intuitive fact that Monty's help gets decreasingly helpful the more doors there are), so switching doors is the better strategy.

(b) $\mathbb{P}_N(W) = 1/n$ remains unchanged. The same approach as in part (a) yields the same numerator as before. For the denominator, we need to figure out the size of the sample space for the experiment where we first pick a door at random, then switch. Again, there are n ways of making the first choice. Once Monty reveals $n-2$ other doors, though, there is only one remaining option for us to switch to. Thus the denominator is much smaller:

$$\begin{aligned}\mathbb{P}_S(W) &= \left(\underbrace{1}_{\text{first choice = car}} \cdot \underbrace{0}_{\text{second choice = car}} + \underbrace{(n-1)}_{\text{first choice = goat}} \cdot \underbrace{1}_{\text{second choice = car}} \right) / \underbrace{n \cdot 1}_{\text{total \# of choices}} \\ &= \frac{n-1}{n} = 1 - \frac{1}{n}\end{aligned}$$

so switching is again the better strategy.

(c) Now $\mathbb{P}_N(W) = k/n$ since k doors hide a car. Reasoning about sample spaces in the same way we did in part (b) gives us a way to compute the denominator of $\mathbb{P}_S(W)$. However, now the numerator (number of favorable outcomes in the case where we switch) changes too:

$$\begin{aligned}\mathbb{P}_S(W) &= \left(\underbrace{k}_{\text{first choice = car}} \cdot \underbrace{k-1}_{\text{second choice = car}} + \underbrace{(n-k)}_{\text{first choice = goat}} \cdot \underbrace{k}_{\text{second choice = car}} \right) / \underbrace{n(n-j-1)}_{\text{total \# of choices}} \\ &= \frac{k(n-1)}{n(n-j-1)} = \frac{k}{n} \cdot \frac{n-1}{n-j-1}.\end{aligned}$$

From here we see that $\mathbb{P}_S(W)/\mathbb{P}_N(W) = \frac{n-1}{n-j-1}$, which is maximal if $j = n-k-1$. In other words, if Monty reveals all but one goat (which he does in the original show where $n = 3, k = 1$ and $j = 1 = n-k-1$), then the contestant can increase their chances of winning by a factor of $\frac{n-1}{k}$ (which is a factor of 2 in the original show). In particular, the largest relative advantage of switching is achieved when $k = 1$.