Preparing for Exam 1

Exam 1 will be in class on Thursday, 2 March.

It will cover material from Classes 1–9, Problem Sets 1–4 (including the provided comments), Quizzes 1–4, and Chapter 1–5 of the TCS Book. Nearly everything on the exam will have been covered in at least three of these places (e.g., in class, on a problem set, and in the textbook; or in multiple classes, the textbook, and a quiz).

As a reminder from the syllabus, you may prepare a one-page (letter-size, two-sided) reference sheet for use during the exam, but all other resources are forbidden (no internet, textbook, other humans, magnification instruments, etc.). We expect that students will benefit from thinking about what to put on your reference sheet in preparing for the exam, and you may work with anyone you want (including other students in the class) to prepare your reference sheets together.

The problems below should give you an idea what to expect on the Exam — problems 1–9 are essentially an example of a full Exam 1 (from the Fall 2019 course, but with a few edits to some questions and whitespace removed), so similar in length to what you should expect for the exam on March 2. If you want to see the original exam that was given to students in 2019, you are welcome to look at that also: https://uvatoc.github.io/f19/exam1comments/. This exam does not cover some of the topics we have covered for this Exam, so we have also provided some additional practice problems after the practice exam.

We will post solutions and comments for some of these problems soon (and are happy to answer any questions you have about them), but encourage you to first try them on your own, then in discussion with other students if possible, before looking at the solutions (hopefully to verify your own answer is correct, and to see other approaches). We emphasize, though, that you are not expected to solve all of these problems and there is no submission expected for these. The problems are provided to give you problems to practice and check your understanding in preparing for the exam. If you are able to solve these problems, you should be confident that you'll be able to do well on the exam.

Practice Exam 1

Boolean Circuits

For these questions, we assume the following logical functions with their standard meanings:

NOT(a): *NOT*$(0) = 1$, *NOT*$(1) = 1$.
OR(a, b): *OR*$(0, 0) = 0$, otherwise *OR*$(a, b) = 1$.
AND(a, b): *AND*$(1, 1) = 1$, otherwise *AND*$(a, b) = 0$.

1. Give a simple description (which could be just the name of a well known function) of the function defined by the code below:

```
def MYSTERY(a, b):
   v1 = NAND(a, b)
   return NAND(v1, v1)
```

Answer: From this table we can see that the function returns 1 if both a and b are 1, 1s so it is the familiar AND function.

a	b	v1	*output*
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Another way to see this is that $NAND(x, x) = NOT(x)$, so $NOT(NAND(a, b)) = NOT(NOT(AND(a, b)))$, which simplifies to $AND(a, b)$.

2. Define *XOR* using only *NOT*, *OR* and *AND*, where *XOR*$(0, 0) =$ *XOR*$(1, 1) = 0$ and *XOR*$(0, 1) =$ *XOR*$(1, 0) = 1$.

Answer: There are many ways to do this. Here are two:

```
def XOR(a, b):
   both1 = AND(a, b)
   both0 = NOT(OR(a, b))
   both_same = OR(both1, both0))
   return NOT(both_same)
```

```
def XOR(a, b):
   dif_or_0s = NOT(AND(a, b))
   dif_or_1s = OR(a, b)
   both_dif = AND(both1, both0))
   return both_dif
```

Countability

For these problems, you may use any results that were proven in class or on a problem set in your proof (without needing to prove them).

3. Prove that the set of all fish in the sea is *countable*. (For purposes of this question, you can assume the "sea" in question is the Mediterranean Sea, and *fish* has its conventional meaning.)

Answer: This problem should remind you of the question on PS1 asking about the set of Python programs that can run on your laptop. A finite set is countable. The number of fish in the Mediterranean Sea must be finite (for lots of obvious reasons that don't need to be stated, but if you are not convinced we know that the volume of the sea is finite, and each fish has non-zero volume, so it can contain at most a finite number of fish), so must be countable.

4. Prove that the set of the even natural numbers (i.e., $\{0, 2, 4, 6, \ldots\}$) is *countably infinite*.

Answer: One way to show this is to find a bijection between $\mathbb{N}$ and the even numbers. A simple one is $g(x) = 2x$. This maps $0 \leftrightarrow 0, 1 \leftrightarrow 2, 2 \leftrightarrow 4, 3 \leftrightarrow 6, \ldots$. Each natural number maps to a unique even since there are no two $a, b \in \mathbb{N}$ such that $2a = 2b$ where $a \neq b$.

A harder approach (in this case, but usually a good idea if you are stuck finding a bijection) is to show *countable* and *infinite* separately. The evens are countable since they are a subset of $\mathbb{N}$, which we already know is countable and a any subset of a countable set is countable.

 Creative Commons BY-NC 4.0

We can show the evens are infinite since there is no maximal even number. Towards a contradiction, assume there is some maximal even number k. We can create a greater even number, $k + 2$. Thus, we have a contradiction and there is no maximal even number.

5. Prove that the set of all directed graphs (as defined below) is *uncountable*.

> **Definition 1 (Directed Graph)** A *directed graph* $G = (V, E)$ consists of a (possibly infinite) set V of vertices and a (possibly infinite) set E of edges. Every edge is an ordered pair of two distinct elements of V.

Answer: In Problem Set 2 we proved that the set of all undirected graphs is uncountable. Since each undirected graph can be mapped to a directed graph (e.g., just make all the edges in the direction from lower $\rightarrow$ higher), we know there are at least as many directed graphs as there are undirected graphs. Since we already know the undirected graphs are uncountable, this shows the directed graphs are also uncountable.

Proofs with Definitions

Here we define the Counting Numbers, similarly to the definition of Natural Numbers you have seen in class:

> **Definition 2 (Counting Numbers)** We define the *Counting Numbers* as:
>
> 1. **1** is a Counting Number.
> 2. If n is a Counting Number, **S**(n) is a Counting Number.

6. Prove that the cardinality of the set of all *Counting Numbers* (as defined above) is *countably infinite*.

Answer: Similarly to problem 4 (in this exam), we can prove a set is countably infinite either by showing a bijection with $\mathbb{N}$, or by separately showing the set is infinite and the set is countable.

Approach 1: Bijection

We can map our Counting Numbers to $\mathbb{N}$ by simply counting the number of **S**s in the number. Accounting to the base case of the definition, **1** is a Counting Number. This has 0 **S**s in it, so maps to the natural number 0. The recursive case produces a new Counting Number by adding one **S** to a previous one, and this is the only way to create new Counting Numbers. So, we can map any Counting Number to a $\mathbb{N}$ by the following bijection: $0 \leftrightarrow \mathbf{1}, 1 \leftrightarrow \mathbf{S}(\mathbf{1}), 2 \leftrightarrow \mathbf{S}(\mathbf{S}(\mathbf{1})), \ldots$.

Approach 2: Infinite and Countable

The Counting Numbers are *infinite* since the recursive rule can always produce a new Counting Number. We can prove by contradiction that there is no last Counting Number, since if there were some last Counting Number x, we can produce a new one **S**(x).

The Counting Numbers are *countable* since we can count the number of applications of the recursive rule needed to produce any counting number x, starting from the base **1**.

 Creative Commons BY-NC 4.0

Computing Models

Recall from class and the book that two computing models are *equivalent* in power if every computation that can be defined using the first model can also be defined using the second model, and every computation that can be defined using the second model can also be defined using the first model.

7. Prove that Boolean circuits using the gateset $\{MAJ, NOT, ZERO\}$ is equivalent to Boolean circuits using the gateset $\{MAJ, NOT, ONE\}$. (*ZERO* is the constant gate that outputs `0` for any input, and ONE is the constant gate that outputs `1` for any input. MAJ is the majority of three inputs gate you are familiar with from PS3.)

Answer: Since the only difference between the two gatesets if the first one includes $ZERO$ (but not ONE and the second one includes ONE but not $ZERO$, we can show they are equivalent by showing how to produce ONE using the first gateset, and produce $ZERO$ using the second gateset:

($\Longrightarrow$) $ONE = NOT(ZERO)$

($\Longleftarrow$) $ZERO = NOT(ONE)$

8. Prove that Boolean circuits using the gateset $\{MAJ, NOT\}$ is **not** equivalent to Boolean circuits using the gateset $\{NAND, XOR\}$.

Answer: We know from PS3 and class that $\{MAJ, NOT\}$ is *not universal*, but since $\{NAND, XOR\}$ includes $NAND$ it clearly is universal. They cannot be equivalent since what it means to be not universal is that it cannot compute some function that a universal gate set can compute.

9. Prove that *AON-STRAIGHTLINE*, straightline programs composed of *AND*, *OR*, and *NOT* operations is equivalent to *MOP-STRAIGHTLINE*, straightline programs composed of the plus, multiply, and constant one operations defined by:

```
def PLUS(a, b):
   return (a + b) % 2


def MULT(a, b):
   return (a * b) % 2


def ONE(a, b):
   return 1
```

The `%` operator is modulo (remainder after division). So, for example `(0 + 1) % 2 = 1`, `(1 + 1) % 2 = 0`, and `(1 * 1) % 2 = 1`.

Answer: First, we rewrite the `PLUS`, `MULT`, and `ONE` operations using logic gates:

`PLUS(a, b) = AND(OR(a,b), NOT(AND(a, b))`	`PLUS` is `XOR`
`MULT(a, b) = AND(a, b)`	`MULT` is `AND`
`ONE(a, b) = OR(OR(a, b), NOT(OR(a, b)))`	Always outputs **1** ($a \vee NOT(a)$)

Since we already know *AON-STRAIGHTLINE* is universal, we could just argue that *MOP-STRAIGHTLINE* is a regular Boolean circuit so can't be more powerful than universal. Alterantively, we could show how to define *AON-STRAIGHTLINE* using *MOP-STRAIGHTLINE*:

 Creative Commons BY-NC 4.0

`AND(a, b) = MULT(a, b)`	`AND` is `MULT`
`NOT(a) = PLUS(a, ONE(a, a))`	`XOR(a, 1) = NOT(a)`
`OR(a, b) = PLUS(PLUS(a, b), MULT(a, b))`	Lots of alternatives

Additional Practice Problems

Below are some additional practice problems that we think will be helpful to you in preparing for the Exam. They are collected and adapted from various sources, mostly previous exams used in our courses, so should give you a good idea of other types of problems to expect on the exam.

Countable, Uncountable, Unknown

10. For each set described below, indicate whether its cardinality is *Countable*, *Uncountable*, or *Unknown* (not determined by the question if it is countable or uncountable). Circle one option and give a proof of your answer.

(a) The set of all grades that students will get on this exam.

Countable *Uncountable* *Unknown*

Answer: *Countable*. Proof: It is finite, and all finite sets are countable.

(b) The set of NAND circuits that compute XOR.

Countable *Uncountable* *Unknown*

Answer: *Countable* Proof: This is a subset of the set of all NAND circuits. We set of all NAND circuits is countable since the way we defined Boolean circuits the number of gates is finite. We could prove this by showing a way to map all NAND circuits to a unique natural number. (Note that with a different definition of circuits where the number of gates is countably infinite, this would be a much trickier question.)

(c) The set of of all Boolean functions that cannot be computed using a Boolean circuit with 2^{3120} or fewer NAND gates.

Countable *Uncountable* *Unknown*

Answer: *Countable* or *Uncountable*. Assuming by "Boolean functions" we mean the finite Boolean functions from $0,1^n \rightarrow 0,1$ where $n \in \mathbb{N}$, this is the union of a countably infinite number of finite sets (for each number of inputs, $n \in \mathbb{N}$ there are a finite number of functions), which (as argued in more detail in problem 18 below) we know is countable.

If by "Boolean functions" we mean functions where the input can be infinite, $0,1^\infty \rightarrow 0,1$, then the set is *Uncountable*. We know the full set of such functions is uncountable (can be easily mapped to the infinite bitstrings), and only a finite number of functions can be computed with fewer than 2^{3102} NAND gates. Removing a finite number of elements from an uncountable set (or even a countably infinite number of elements) leaves us with a set that is still uncountable.

 Creative Commons BY-NC 4.0

Induction

11. Define the function $\texttt{ALT}_n : \{0,1\}^{2n} \to \{0,1\}$ such that for a string $w \in \{0,1\}^{2n}$ we say that $\texttt{ALT}_n(w) = 1$ provided $w \in (01)^*$. We could compute $\texttt{ALT}_1$ using the following straightline program:

```
def ALT1(x1,x2):
    diff = XOR(x1,x2)
    return AND(x2, diff)
```

We could then implement $\texttt{ALT}_n$ as follows:

```
def ALTn(x1,x2,...,x2n):
    diff = XOR(x1,x2)
    first = AND(x2,diff)
    rest = ALTn(x3,...,x2n)
    return AND(first, rest)
```

Suppose we have similarly implemented $\texttt{ALT}_{n-1}$, $\texttt{ALT}_{n-2}$, etc., (and all other dependent subroutines).

Show that the number of `NAND` gates needed to represent a circuit for $\texttt{ALT}_n$ is no more than $10n$ gates (hint: `XOR` requires 4 `NAND` gates and `AND` requires 3 `NAND` gates).

Comments. We won't provide a written solution to this (no one asked about it), but you should all be able to construct a solid proof by induction. Four main things to remember:

1. First, state what you are proving clearly. (This is true for all proofs!)
2. Then, state the induction hypothesis, $P(n)$ where n is an input natural number, and your goal is to prove $\forall n \in \mathbb{N} : P(n)$ to prove the theorem from the first step. (For some problems you will have to think carefully if you need $\forall n \in \mathbb{N}$ or some subset of $\mathbb{N}$ or some other countable set.
3. Prove the base case: $P(0)$.
4. Prove the inductive case: $P(n) \implies P(n+1)$. You should think carefully about which values n this should apply for. Also, in some cases it may be easier to prove $P(n-1) \implies P(n)$.

Asymptotics

12. Let $f(n) = 8n^{4.5}$ and $g(n) = 5n^5$, which of the following are true? Support your answer to each part with a convincing argument.

(a) $f \in O(g)$

Answer: True. Since the degree of $g(n)$ is 5, and the degree of $f(n)$ is 4.5, we know f grows asymptotically slower than g, so $f \in O(g)$.

(b) $f \in \Omega(g)$

Answer: False.

(c) $f \in \Theta(g)$

Answer: False. Since $f \notin \Omega(g)$ it cannot be in $\Theta(g)$ which is the intersection of $O(g)$ and $\Omega(g)$.

Creative Commons BY-NC 4.0 Mohammad Mahmoody and David Evans

Relation Properties

13. Considered the relation, $\leq$ (less than or equal to, with the standard meaning), with the domain set, $\mathbb{N}$ and codomain set $\mathbb{N}$. Which of these properties does the $\leq$ relation have: function, total, injective, surjective, bijective?

Answer: Note we could consider $\leq$ as a function from an ordered pair of natural numbers to a Boolean, but here we consider it as a relation where there is an edge from each domain element to all the codomain elements it is less than or equal to (this wasn't clearly stated in the question, so there are lots of other ways you could have reasonably interpreted the $\leq$ relation.

So, there is an edge from $a \in \mathbb{N}$ to $b \in \mathbb{N}$ if and only if $a \leq b$. Since for a given $a \in \mathbb{N}$, there can be more than one $b \in \mathbb{N}$ such that $a \leq b$ this means there can be multiple edges out of a domain element, so it is not a function. It is total since there is at least one edge out of every element of $\mathbb{N}$ (note that $<$ would not be total, since 0 is not less than any element of the codomain).

Injective means ≤ 1 edge into every codomain element. It is not injective — for example, the codomain element 2 has incoming edges from three domain elements (0, 1, and 2). It is surjective (≥ 1 edge into every codomain element), since every element of the codomain has at least one domain element that is $\leq$ it. It is not bijective, since to be bijective it must be both injective and surjective, but it is not injective.

14. Set Cardinality

a. Assume $R : A \rightarrow B$ is an *total injective* function between A and B. What must be true about the relationship between $|A|$ and $|B|$?

   Answer: Since R is a total function, there is exactly one edge out of each domain element. To be injective (≤ 1 in), there cannot be multiple edges into any codomain elements (but there can be codomain elements with 0 incoming edges). So, every element of the domain is connected to a different element in the codomain. This means $|A| \leq |B|$.

b. Assume $R : A \rightarrow B$ is an *total surjective* function between A and B. What must be true about the relationship between $|A|$ and $|B|$?

   Answer: Surjective means ≥ 1 in, so $|A| \geq |B|$.

c. Assume $R : A \rightarrow B$ is a (not necessarily total) *surjective* function between A and B. What must be true about the relationship between $|A|$ and $|B|$?

   Answer: $|A| \geq |B|$. It doesn't matter that R is not necessarily total, since if there are additional elements in A that have no out edges, that just means it is "more bigger". It does matter that R is a function — otherwise, all the out edges (to cover all of B) could be coming from a single domain element.

Countable and Uncountable Infinities

15. Prove that the integers, i.e., $\ldots, -2, -1, 0, 1, 2, \ldots$, are countably infinite.

Answer: We can construct a bijection between the integers and the natural numbers, by alternating between the positive and negative integers: $0, -1, 1, -2, 2, \$ldots$. Since this shows the cardinality of the integers is the same as the cardinality of the naturals, which is countably infinite, this completes the proof.

16. Prove that the number of total injective functions between $\mathbb{N}$ and $\mathbb{N}$ is *uncountable*.

 Creative Commons BY-NC 4.0

Answer: As stated originally (with *countable* instead of uncountable), this one was impossible since the cardinality of the set is actually uncountable.

A good problem solving strategy when a question seem hard to answer (and indeed, this question is quite difficult!) is to first start with an easier, but related question. (At least in this class, and it nearly all "real world" contexts, you are much better off answering a question by saying that you can't figure out how to solve the original question, but here's a variation on it that you can answer, than seeing a clearly bogus and deceptive answer to the original question.) So, first let's answer an easier variation of this question where instead of counting total, injective functions, we count all the binary relations.

A binary relation is a subset of all possible edges between elements in the domain and codomain: $R \subseteq \mathbb{N} \times \mathbb{N}$. The cardinality of $\mathbb{N} \times \mathbb{N}$ is the same as the cardinality of $\mathbb{N}$, it is countably infinite. To see this, draw the product set in a grid and consider a bijection like, $0 \longleftrightarrow (0,0); 1 \longleftrightarrow (0,1); 2 \longleftrightarrow (1,0); 3 \longleftrightarrow (2,0); 4 \longleftrightarrow (1,1); 5 \longleftrightarrow (0,2); 6 \longleftrightarrow (0,3); 7 \longleftrightarrow (1,2); 8 \longleftrightarrow (2,1); 9 \longleftrightarrow (3,0); \cdots$.

But, we can define a relation for any subset of the set $\mathbb{N} \times \mathbb{N}$. The set of all subsets of a set is its powerset, so we have the powerset of a countably infinite set which is uncountable. (Unlike in graphs, the labels matter here, since we are mapping between actual elements of the domain and codomain, so it is easy to see that all of these relations are different.)

If you got this part, you are well prepared for the exam (which won't have any questions as hard as the one asked here on it). To answer the (corrected) original question, we need to think carefully about how to construct the total injective functions.

Let's divide the codomain into two sets, both of which are countably infinite. An easy way to do this is to split it into even and odd numbers: $\mathbb{N} = EVENS \cup ODDS$ where $EVENS = \{0, 2, 4, \ldots\}$ and $ODDS = \{1, 3, 5, \ldots\}$. Since $EVENS$ is countably infinite, there is a total, injective function between $\mathbb{N}$ and $EVENS$.

Now, we will show how to make a different total, injective function between $\mathbb{N}$ and each set in $\forall X \in \text{pow}(ODDS) \cup EVENS$. Since the cardinality of $\text{pow}(ODDS)$ is uncountable, if we can show a way to construct a different total, injective function for each element of X, we have showing that the cardinality of the set of all total, injective, functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable.

17. Prove that the number of different chess positions is countable. (A chess position is defined by the locations of pieces on an 8×8 board, where each square on the board can be either empty, or contain a piece from $\{\text{Pawn}, \text{Knight}, \text{Bishop}, \text{Castle}, \text{Queen}, \text{King}\}$ of one of two possible colors.)

Answer: The number of positions is finite, so it is countable.

18. Prove that number of Ziggy Pig ice cream dishes is countable. A Ziggy Pig ice cream can contain any number of scoops ($scoops \in \mathbb{N}$), and each scoop can be of any flavor, where distinct flavors are identified by $v \in \mathbb{N}$.

Note added: This problem has been updated — the original version of the question asked for a proof that it was "uncountable", which is not the case! You should consider an alternate version of the question where the number of scoops could be infinite, and then it would indeed be uncountable, but as stated the number of scoops is unlimited but finite since $scoops \in \mathbb{N}$.

Answer: If the number of scoops were infinite, we can map $pow(\mathbb{N})$ to the Ziggy Pig ice cream dishes, since each subset of the flavors is a different dish. This proves that it is uncountable. (If you are unfamiliar with the Ziggy Pig, I can only excuse your cultural gap because of your youth, but please aim to correct this travesty by watching "Bill and Ted's Excellent Adventure" over spring break!)

But, the way the question is stated the number of scoops is finite — $scoops \in \mathbb{N}$ means the number of scoops is a particular natural number, so although it is unbounded, it is finite. Then, the total number of dishes is the union of the number of dishes for each number of scoops: $D_0 \cup D_1 \cup D_2 \cup \cdots \cup D_i \cup \cdots$. This is a countable number of

 Creative Commons BY-NC 4.0

sets, since there is one set for each natural number. The cardinality of the set D_i is v^i (selecting one of v flavors for each scoop, this assumes the order of the scoops matters, but if not it will be less than this), which is finite. So, the full set is a union of a countably infinite number of finite sets. This is countably infinite. One way to see this is to just count the sets in order - start counting the elements of D_0, then instead of starting again at 0, just continue to the next natural number and count the elements of D_1, and so on. Note that here the component sets are finite, but the same argument works even if the component sets are countably infinite if we interleave the elements of the sets (count the first element of D_0; then the second element of D_0 and the first element of D_1; then the third element of D_0, the second element of D_1, the third element of D_2; this continues, eventually covering all elements of all the sets showing that the union of all the sets is countable.

19. Prove that all fish who have fully eaten Ziggy Pig ice creams (as describe in the previous problem) with an infinite number of scoops are Coho Salmon.

Answer: This is vacuously true, and contradicts the previous question where the number of scoops was finite. Since the amount of sugar in the universe is finite, there exist no Ziggy Pig ice creams with an infinite number of scoops, and the set of all fish who have eaten ice creams with an infinite number of scoops is empty. Any property is true about all elements in an empty set, since there are none of them.

Induction Practice

20. Prove by induction that every natural number less than 2^{k+1} can be written as $a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \cdots + a_k \cdot 2^k$ where all the a_i values are either 0 or 1.

Answer: For any induction proof, we should start by carefully defining the induction predicate. In this case, it follows directly from the proposition, except we swap k with n (this is just to have the induction predicate take n as its parameter, which is conventional):

$P(n)$ = every natural number less than 2^{n+1} can be written as

$$a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \cdots + a_n \cdot 2^n$$

where all the a_i values are either 0 or 1.

We want to prove this for all $n \in \mathbb{N}$.

Base Case: $n = 0$.

Since $2^{0+1} = 2$, we need to show that 0 and 1 can both be written as $a_0 \cdot 2^0$: $0 = 0 \cdot 2^0$ ($a_0 = 0$) and $1 = 1 \cdot 2^0$ ($a_0 = 1$).

Inductive Case: $P(n) \implies P(n+1)$.

From $P(n)$, we know all numbers less than 2^{n+1} can be written as

$$a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \cdots + a_n \cdot 2^n$$

where all the a_i values are either 0 or 1.

To prove $P(n+1)$ we need to show that all numbers less than $2^{(n+1)+1} = 2^{n+2}$ can be written as

$$a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \cdots + a_n \cdot 2^n + \cdot a_{n+1} \cdot 2^{n+1}$$

where all the a_i values are either 0 or 1.

 Creative Commons BY-NC 4.0

In comparing to what $P(n)$ means, to show $P(n+1)$ we need to cover the numbers from 0 to $2^{n+2}-1$, and we have an extra term $a_{n+1} \cdot 2^{n+1}$. If we set $a_{n+1} = 0$, we cover all the numbers from 0 to $2^{n+1}-1$ because this is the same as $P(n)$.

Each number from 2^{n+1} to $2^{n+2}-1$ can be written as $m + 2^{n+1}$ where $m \in \{0, \ldots, 2^{n+1}-1\}$, that is, the numbers covered by $P(n)$ using the terms up to $a_n \cdot 2^n$. So, we cover all the new numbers by setting $a_{n+1} = 1$, and all the old numbers by setting $a_{n+1} = 0$.

21. Prove by induction that every finite non-empty subset of the natural numbers contains a *greatest* element, where an element $x \in S$ is defined as the *greatest* element if $\forall z \in S - \{x\}.\ x > z$.

Answer: Note that this property may seem trivial, but it is actually quite subtle, and is only true because we limited it to *finite* sets. For example, $\mathbb{N}$ does not have a greatest element (but all subsets of $\mathbb{N}$ do have a least element, which is the well ordering principle).

To prove it, we do induction on the set of the sets. Since we are only dealing with non-empty sets, we are proving the predicate for all elements in $\mathbb{N} - \{0\}$:

$$P(n) = \text{a set of natural numbers of size } n \text{ has a greatest element}$$

Base case: $P(1)$. Every set of size 1 can be written as $\{x\}$ where $x \in \mathbb{N}$. This set has a greatest element, namely x.

Inductive case: $P(n) \implies P(n+1)$.

Every set, T, of size $n+1$ can be written as $T = S \cup \{z\}$ where $|S| = n$ and $z \notin S$ for some $z \in \mathbb{N}$. By $P(n)$, we know S has some greatest element $g \in S$.

We have two cases to consider:

1. $z > g$: The greatest element of T is z.
2. $z < g$: The greatest element of T is g. Since T includes every element of S, we also know $g \in T$.

We know $z \neq g$ since $z \notin S$ and $g \in S$.

This covers all possibilities, and in both cases we have a greatest element in T.

22. In class, we argued that a "good" Boolean circuit always eventually evaluates to a value using the definition of circuit evaluation. Prove that a Boolean circuit where there is a cycle on a path between an input and an output will never produce a value for that output.

Answer: We gave an informal solution in class. The main idea is to show that a gate's output is undefined until all of its inputs are defined, but if there is a cycle, one of its inputs depends on its output, so will never be defined.

 Creative Commons BY-NC 4.0