Logic in Computer Science
Chapter 12

Antonina Kolokolova

February 7, 2014

How to describe a set? List elements, or list a property that elements have.

$S = \{John, Bob, Mary, George, Alex\}$. Another set is $S = \{2, 3, 4\}$, which is the same as $S = \{x \mid 2 \leq x \leq 4\}$. If a set is infinite, we cannot list its elements, so we have to list the property: $S = \{x \mid x \in \mathbb{N} \wedge x|2\}$ is the set of all even natural numbers

12.1 Operations on sets

Suppose S_1 and S_2 are two sets (with the universe U). Then the following are set operations:

$$\begin{aligned}
\text{Union:} \quad & S_1 \cup S_2 = \{x|x \in S_1 \vee x \in S_2\} \\
\text{Intersection:} \quad & S_1 \cap S_2 = \{x|x \in S_1 \wedge x \in S_2\} \\
\text{Difference:} \quad & S_1 - S_2 = \{x|x \in S_1 \wedge x \notin S_2\} \\
\text{Complement:} \quad & \bar{S_1} = \{x|x \notin S_1\}
\end{aligned}$$

Note that the complement of an empty set is the universe, and the complement of the universe is the empty set.

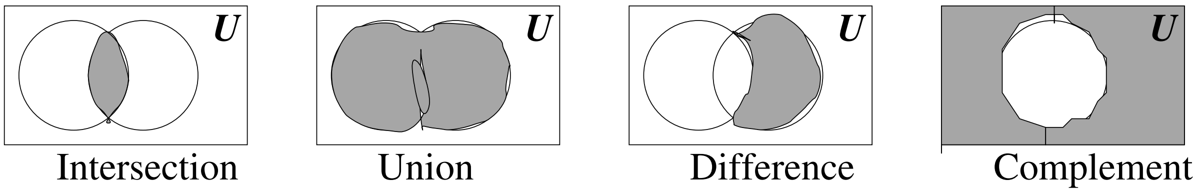


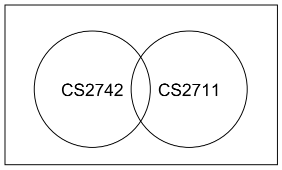
Figure 1: Set operations

Two sets are equal of they are subsets of each other: $A = B$ is defined to be $\forall x(x \in A \iff x \in B)$, or, alternatively, iff $A \subseteq B \wedge B \subseteq A$.

One way of representing sets is Venn diagrams. Here, each set is drawn as a circle, intersecting of the sets have common elements. Venn diagrams are useful for visualizing sets containing common elements.

Example 1. Suppose 15 students take CS2742 and 10 students take CS2711. Provided that 3 students take both, how many are in either one of these two courses?

To solve the problem, draw a Venn diagram (two intersecting circles) for the sets of students in CS2742 and CS2711, respectively. Put the number of students taking both (3) in the intersection. Now, put the remaining number of students in the corresponding sets outside of the intersection: 15-3=12 in the CS2742 set, and 10-3=7 in the CS2711 set.



Now, the total number of students, that is students that are in either of these two courses, is the sum of these three numbers: $|A \cup B| = (|A| - |A \cap B|) + (|B| - |A \cap B|) + |A \cap B| = 12 + 7 + 3 = 22$. Equivalently, $|A \cup B| = |A| + |B| - |A \cap B| = 15 + 10 - 3$ (since we subtracted $|A \cap B|$ twice and added once).

In general, this type of argument is called *the rule of inclusion/exclusion* and it can be generalized for an arbitrary number of sets. Using the notation $|S|$ to denote the number of elements in the set,

$$\begin{aligned} |A \cup B| &= |A| + |B| - |A \cap B| \\ |A \cup B \cup C| &= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| \end{aligned}$$

Later in the course we will develop techniques that will allow us to prove a general inclusion/exclusion rule.

Puzzle 1. What is wrong with the following argument?
"Nothing is better than an eternal bliss. A hamburger is better than nothing. Therefore,a hamburger is better than eternal bliss".

12.2 Boolean algebras

Before continuing, let us look at a simple example of sets:

Example 2 (Intervals on a real line)**.** Let $(-1,0]$ and $[0,1)$ be two intervals on a real line. $(-1,0] \cup [0,1) = (-1,1)$, $(-1,0] \cap [0,1) = \{0\}$, $(-1,0] - [0,1) = (-1,0)$.

Definition 1. *Two sets are disjoint if they have no common elements. Sets $A_1 \dots A_n$ form a partition of a set A if sets are pairwise disjoint (that is, $\forall i,j\ A_i \cap A_j = \emptyset$ and their union forms the whole set A.*

Note that the rule of inclusion/exclusion simplifies greatly when sets are disjoint: in this case, the sum of the sizes of the disjoint sets is exactly the size of their union. This is used in probability theory.

Some subset relations:

1) $A \cap B \subseteq A$
2) $A \subseteq A \cup B$
3) Transitivity: if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.

To prove that $A \subseteq B$ show that any element $x \in A$ is also $\in B$. Proof style: "suppose x is in A. Now show that $x \in B$." Now use logic of the definitions.

Now can prove properties of sets such as DeMorgan, by using "suppose x is in the left side... show that x is in the right side".

To prove something does not hold, find a counterexample.

Example 3. Show that it is not true that for all A, B, C, $(A - B) \cup (B - C) = A - C$. To prove this, find a counterexample, that is, sets A, B, C for which this does not hold. Let $A = \{1,2\}$, $B = \{2\}$ and $C = \{1\}$. Then $A - C = \{2\}$, $A - B = \{1\}$, $B - C = \{2\}$ and the union is $\{1,2\}$. Alternatively, think of an element in the LHS that is not in $A - C$: in this case, such an element is some element not in A.

As you have noticed, the algebra of sets is very similar to the algebra of propositions. This is because they are all examples of Boolean algebras.

Definition 2. *A Boolean algebra is a set B together with two operations, generally denoted $+$ and $\cdot$, such that for all a and b in B both $a + b$ and $a \cdot b$ are in B and the following properties hold:*

- *Commutative laws: $a + b = b + a$ and $a \cdot b = b \cdot a$.*
- *Associative laws: $(a + b) + c = a + (b + c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*

- *Distributive laws:* $(a+b)\cdot c = a\cdot c + b\cdot c$ *and* $a\cdot b + c = (a+c)\cdot(b+c)$ *(recall that the second one does not hold for the normal arithmetic* $+$ *and* $\cdot$*).*
- *Identity laws:* $a+0=a$ *and* $a\cdot 1 = a$
- *Complement laws: for each* a *there exists an element called negation of* a *and denoted* $\bar{a}$ *such that* $a+\bar{a}=1$, $a\cdot\bar{a}=0$.

In the case of propositional logic, 0 is F, 1 is T and there are no other elements, so it is sufficient to say that $\bar{T}=F$ and $\bar{F}=T$ (in that setting, $\neg$ is used for complementation). In set theory, 0 and 1 are $\emptyset$ and the universe U, respectively, and negation of every set is its complement. Now, properties of Boolean algebras such as De Morgan's law can be derived from these axioms.

Example 4 (Idempotent identity)**.** Show that $a+a=a$.

$$\begin{aligned}
a &= a+0 && \text{because 0 is the identity for } + \\
&= a+(a\cdot\bar{a}) && \text{by the complement law for } \cdot \\
&= (a+a)\cdot(a+\bar{a}) && \text{by the distributive law} \\
&= (a+a)\cdot 1 && \text{by the complement law for } + \\
&= a+a && \text{because 1 is the identity for } +
\end{aligned}$$

Now, we obtain that the Idempotent identity holds for propositional logic and for set theory, since both are examples of Boolean algebras.

Chapter 13

13.1 Building new sets: power set, Cartesian product, relations.

A *power set* of a set A, denoted 2^A, is a set of all subsets of A. For example, if $A = \{1, 2, 3\}$ then $2^A = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}\}$.

Let $|A|$ denote the number of elements of A (also called *cardinality*, especially when talking about infinite sets.) The size of the power set, as notation suggests, is $2^{|A|}$.

Theorem 1. *Let A be a finite set. Then the cardinality of 2^A is $2^{|A|}$.*

Proof. Suppose A has n elements. Now, every subset S of A can be represented by a binary string of length n, which would have a 1 in the positions corresponding to an element in S, and a 0 in places corresponding to elements not in S. For example, if $A = \{1, 2, 3\}$ as above, then $S\{1, 3\}$ is represented by a string 101, and $\emptyset$ is represented by a string 000. Now, the number of binary strings of length n is 2^n. Therefore, the number of possible subsets of A (and thus the elements of 2^A) is also 2^n. □

What if A is infinite? Still the size of the powerset (called *cardinality* in this context) will be larger. In one of the upcoming lectures we will talk about a technique called Diagonalization, due to Cantor, that can be used to show this.

Another useful notation is the Cartesian product, which will allow us to talk about ordered tuples of elements (pairs, triples, etc). A *Cartesian product* of sets $A_1 \dots A_n$, denoted $A_1 \times \cdots \times A_n$ is a set of ordered tuples $< a_1, a_2, ...a_n >$ such that $a_1 \in A_1 \wedge a_2 \in A_2 \wedge \cdots \wedge a_n \in A_n$. Note that an *ordered tuple* (a, b) is not the same as a set $\{a, b\}$: here the order of elements matters, so the tuple $< 1, 2 >$ is not the same as the tuple $< 2, 1 >$. For two sets, their Cartesian product is $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$.

For example, a Cartesian product of sets $\{3, 4\}$ and $\{1, 2, 3\}$ is the set of pairs $\{(3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (4, 3)\}$. Note that the pair $(4, 3)$ is in the set, but the pair $(3, 4)$ is not, because 4 is not an element of $\{1, 2, 3\}$.

Definition 1. *A relation on n variables $R(x_1, \dots, x_n)$ is a subset of the Cartesian product of domains of $x_1, \dots, x_n$.*

You often hear an expression "relational databases". Indeed, a standard way to describe a database is as a set of relations, where each parameter corresponds to a field in a database, and each tuple (element) to an item in the database. For example, a student database could have a relation $StudentInfo(name, number, address)$ which could be a subset of $Strings \times \mathbb{N} \times Strings$. That is, every item in this relation will consist of three elements, denoting the name, student number and address of a specific student. A database usually contains several different relations.

13.2 Relations

Recall that a relation on n variables is just a subset of a Cartesian products of n sets which are domains of the variables. In this respect, a binary relation on a set A is a just a subset of $A \times A$.

For example, $R(x, y) \subseteq \mathcal{Z} \times \mathcal{Z}$ such that $R(x, y)$ if and only if $x \leq y$ is a binary relation. In this case, we often write directly $x \leq y$, rather than writing $R(x, y)$. The relation $x = y$ is also a binary relation, which can be defined for many different domains such as integers, reals, strings and so on.

Another common binary relation is congruence $\mod n$ for a natural number n: in that case, $R(x, y)$, written as $x \equiv y \mod n$, if $\exists z \in \mathcal{Z}$ such that $x - y = zn$. This is the same as saying that x and y have the same remainder from division by n. For every n there is a different $\mod n$ relation.

Yet another binary relation is $Parent(x, y)$, which contains all pairs x, y such that x is a parent of y. At this point you may ask what is the difference between a predicate $Parent(x, y)$ and a binary relation $Parent(x, y)$: the predicate is true on the pairs (x, y) that belong to the set which is the binary relation.

There are several major types of binary relations. In this case, it is often more convenient to view a binary relation $R(x, y)$ as "taking" a x "into" y. Binary relations can be:

- Reflexive: $\forall x \in A\ R(x, x)$.
  For example, $x = y, x \leq y, x \equiv y \mod n$ are reflexive, but $Parent9(x, y)$ and $x < y$ are not.
- Symmetric: $\forall x, y \in A\ R(x, y) \to R(y, x)$.
  Antisymmetric: $\forall x, y \in A\ R(x, y) \wedge R(y, x) \to x = y$.
  For example, $x = y, x \equiv y \mod n$ are symmetric, $x \leq y$, $x < y$ and $Parent(x, y)$ are antisymmetric, and the relation $Likes(x, y)$ defined on the domain of people is neither symmetric nor antisymmetric. Note that symmetric and antisymmetric are not complements: an easy way to see that is by noticing that both have a universal quantifier in the definition, whereas a complement of a universal quantifier is existential

quantifier. Another way of checking it is to find a relation such as $Likes(x, y)$ which is neither symmetric nor antisymmetric.

- Transitive: $\forall x, y, z \in A \; R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
  For example, relation $x = y$ is transitive, because if $x = y$ and $y = z$ then $x = z$ as well. Same can be said about $x \leq y$ and $x < y$: if x is smaller than y, and y is smaller than z, then x is definitely smaller than z.

  Note that $Parent(x, y)$ relation is not transitive: if x is a parent of y, and y is a parent of z, then x is not a parent of z, it is a grandparent. But often we do want to express the fact that one person is related to another by a chain of Parent(x, y) relationships: that is, it is a grandparent, or great-grandparent, or great-great-grandparent and so on. For that, we can define a relation $Ancestor(x, y)$, which would contain all pairs (x, y) related by a chain of $Parent()$ relations. Such a relation is called *a transitive closure.*

  Definition 2. *For a relation $R(x, y)$, its transitive closure contains all pairs x, y such that there is a sequence $z_1, \ldots, z_n$ with $R(x, z_1)$, $R(z_n, y)$ and $\forall 1 \leq i < n \; R(z_i, z_{i+1})$.*

  For example, if John is a grandparent of Jill via Mary who is the daughter of John and the mother of Jill, then x=John, y=Jill and $n = 1$, so there is just one z_1=Mary such that Parent(John,Mary) and Parent(Mary, Jill). Thus, Ancestor(John, Mary) holds.

  A way to describe the Ancestor relation is by the following recursive definition (of the Ancestor predicate): $Ancestor(x, y) = Parent(x, y) \vee \exists z Parent(x, z) \wedge Ancestor(z, y)$.

  However, this definition is not a first-order formula – it mentions the relation $Ancestor(x, y)$ which is being defined. Moreover, as we have seen with the Flight example in the predicate logic, transitive closure cannot be defined by a first-order formula.

A relation that is reflexive, symmetric and transitive is called an *equivalence* relation. Equality and congruence mod 2 are equivalences.So is equivalence of digital circuits computing the same function. Any equivalence relation breaks up the set of all objects on which it is defined into equivalence classes: in such a class all objects are equivalent to each other, but not to any object outside of the class. For example, $x = y \mod 5$ breaks all natural numbers into 5 equivalence classes: 1) numbers that are divisible by 5, 2) numbers that have a remainder 1 from division by 5, 3) ones with remainder 2, 4) with remainder 3, 4) with remainder 4. In the first class, there are numbers 0,5,10,15, 20.., in the third – 2,7,12,17,22,... It is possible to extend this definition to all integers: -1 will be in the same class as 4, -2 in the same class as 3, -5 as 5 and so on. For the names of the equivalence classes we can pick any element of them, but it is convenient to choose $0, 1, 2, 3, 4$, and in general, we represent equivalence classes $\mod n$ as $0, 1, \ldots, n - 1$.

There is a special term to denote relations which are reflexive, transitive and antisymmetric: they are called *partial order* relations (e.g., subset relation). A total order is a subclass of partial orders with an additional property that any two elements are related: that is, for any x, y either R(x,y) or R(y,x). E.g.: $\leq$ on numbers.

13.3 Functions

Recall that a relation on n variables $R(x_1, \ldots, x_n)$ is a subset of the Cartesian product of domains of $x_1, \ldots, x_n$. A *function* is a special kind of relation that has exactly value of x_n for any tuple of values of $x_1 \ldots x_{n-1}$. Usually we write $f(x_1 \ldots x_{n-1}) = x_n$ to mean that R is a function and $R(x_1, \ldots, x_{n-1}, x_n)$ holds.

So just as we defined numbers using sets, we now defined functins and relations on numbers (and not just numbers: the variables can be anything).

Example 1. $f(x) = Mother(x)$ is a function, so is $f(x) = x^2$, so is $f(x) = x/y$.

Definition 3. *We often write functions as $f : X \to Y$ (read as "function f from X to Y") meaning that the tuples of variables of f come from X, and that the output value of f comes from Y. We call X the* domain *of f, and $\{y | x \in X \wedge f(x) = y\}$ a* range *of f, or* image *of X under f. A set Y is called* codomain*; the range of f is a subset of the codomain.*

Domain and codomain can be different sets: e.g., function counting the number of 0's in a binary string $f : \{0,1\}^* \to \mathbb{N}$.

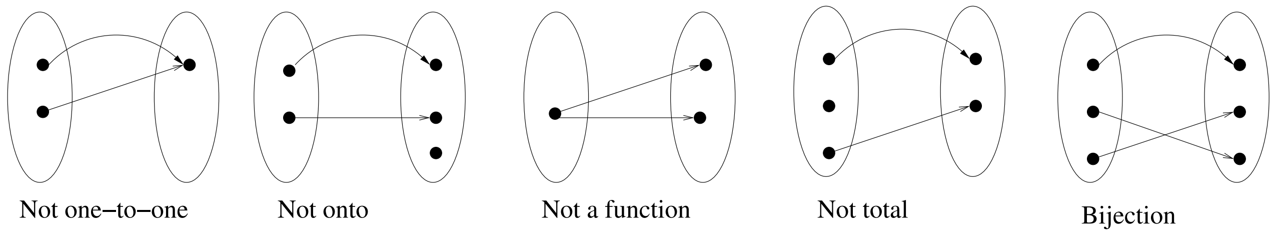
- Identity function: $f(x) = x$. Can be defined for any domain=codomain.
- Constant function: $f(x) = a$, where a does not change when x does. For example, $f : \mathbb{Z} \to \mathbb{Z}$, $f(x) = 0$.
- Arithmetic functions: logarithmic function $f(x, y) = \log_x y$, exponential $f(x, y) = x^y$, addition, multiplication, division, subtraction, etc.
- Boolean functions: a function from strings of 0s and 1s of length n (denoted $\{0,1\}^n$) to $\{0,1\}$.

A function is defined by a formula if there is a formula which is true exactly on tuples of inputs + output of the function. E.g., a function $F : \mathbb{N} \to \mathbb{N}$ $f(x) = x + 1$ can be defined by $y > x \wedge \forall z\ (z \leq x \vee z \geq y)$. Sometimes a function is not well defined on a certain domain: e.g., $\sqrt{x}$ is not well-defined when both the domain and the range are natural numbers.

Definition 4. *Let $f : X \to Y$ be a function. Then f is* one-to-one *(or* injective*) iff $\forall x, y \in X\ (f(x) = f(y) \to x = y)$. A function is* onto *(or* surjective*) if $\forall y \in Y \exists x \in X (f(x) = y)$. A function is* bijective *if it is both one-to-one and onto.*

To prove that two sets are the same size, give a bijection (or give two functions, one a surjection and one an injection).

To prove that a function is one-to-one show that $f(x) = f(y) \to x = y$.



Example 2. For example, $f(x) = 4x + 1$, $f(x) = f(y)$ so 4x+1=4y+1 so $x = y$. On the other hand, $f : \mathbb{Z} \to \mathbb{Z}$, $f(n) = n^2$ is not one-to-one: as a counterexample take $x = -1$ and $y = 1$. Then $x \neq y$, but $x^2 = y^2$.

To prove that a function is onto, show that every element has a *preeimage*. To prove that it is not onto, show that there is an element in the codomain such that nothing maps into it.

Example 3. Consider again $f(x) = 4x+1$ over real numbers. There it is onto. Now consider it over integers. It is not onto integers.

Chapter 14

14.1 Cardinalities and diagonalization

It is easy to compare sizes (in set theory terminology, cardinalities) when sets are finite: whichever set has more elements, that set is larger. In a way, what we are doing in that comparison is creating a correspondence between the elements of the sets by matching elements of one set to elements of another one-by-one. If this correspondence is both one-to-one and onto, then our sets are of equal size. For example, to match $\{10, 20, 30\}$ with $\{a, b, c\}$ we can define a bijection $f(x)$ by $f(10) = a, f(20) = b, f(30) = c$. If sets have different sizes, then either f cannot be one-to-one (if the second set is smaller) or it is not onto (if the second set is large).

This is the idea behind the comparison of infinite sets. We declare two sets to have the same size (cardinality), if there is bijection (that is, a function which is both one-to-one and onto) from one set to the other. To extend this definition, we say that a (possibly infinite) set A has cardinality $|A| \leq |B|$ if there is a one-to-one function from A to B. Note that this gives us another way of proving that two sets have equal cardinality: give two one-to-one functions, one from A to B and another from B to A. Rephrasing pigeonhole principle: for sets such that $|A| < |B|$ there is no one-to-one function from B to A (no onto function from A to B).

But this definition gives us some strange consequences. With infinite sets, it is possible that a proper subset has the same cardinality as the whole set (although it is not possible for a subset to be larger than the whole: identity function gives the proof). Example: $\mathbb{N} \subset \mathbb{Q}$, $Even \subset \mathbb{N}$, $\mathbb{N} \subset \mathbb{N} \times \mathbb{N}$.

Definition 1. *We call sets that have the same cardinality as* $\mathbb{N}$ countable *sets. The sets that have larger cardinality such as* $\mathbb{R}$*, we call* uncountable*.*

Example 1. The set of positive rational numbers is countable.

Note that we already know an injective function from $\mathbb{N}$ to $\mathbb{Q}^+$: it is an identity function. So the interesting part is to give an injective function from $\mathbb{Q}^+$ to $\mathbb{N}$.

For that, look at the following table:

1/1	1/2	1/3	1/4	1/5	...
2/1	2/2	2/3	2/4	2/5	...
3/1	3/2	3/3	3/4	3/5	...
4/1	4/2	4/3	4/4	4/5	...
5/1	5/2	5/3	5/4	5/5	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Now, start counting the elements of the table as follows:

1: 1/1, 2: 1/2. 3: 2/1, 4: 1/3, 5: 2/2, 6: 3/1, 7: 1/4 and so on (that is, list all elements with the sum of numerator and denominator equal 2, then all with sum =3 and so on). Note that every rational number is represented in this table, multiple times, too (for example, 2/4 and 1/2). So we just constructed a one-to-one function from a set larger than $\mathbb{Q}^+$ to the set of natural numbers. Therefore, the cardinality of $\mathbb{Q}^+$ is at most the cardinality of $\mathbb{N}$. Since it is also at least $|\mathbb{N}|$, we conclude that $|\mathbb{Q}^+| = |\mathbb{N}|$.

This kind of argument is often used to show that a certain set is countable.

14.2 Diagonalization

At this point you may ask – is it true, then, that all infinite sets have the same size? What about a powerset of natural numbers $2^{\mathbb{N}}$? In this case, we can show that the resulting set is actually strictly larger than $\mathbb{N}$. For this, we will use a technique called *diagonalization*, due to Cantor, who used it to show that the set of real numbers is larger than the set of natural numbers.

The idea of the proof is as follows. The proof proceeds by contradiction. Assume, for the sake of contradiction, that your set in question (i.e., $2^{\mathbb{N}}$ or $\mathbb{R}$) is countable, that is, there is function from the elements of this set to natural numbers. Make a table in which rows correspond to elements being enumerated in the order of that assumed enumeration (e.g, rows are real numbers represented as strings of digits or rows are (infinite) strings encoding the powerset of natural numbers). Now, construct a new element which belongs to the set and which is different from any row in the table.

For example, here is how the table looks for proving that the powerset of natural numbers is uncountable. Represent each subset S of natural numbers by an infinite string s with 0 in bit i when number i is not in the set S and 1 in bit i if $i \in S$. Now, list these strings as rows of the table, according to that alleged enumeration. The table would look similar to this:

1	**0**	0	1	1	0	1	1	0	1	....
2	1	**1**	1	1	1	0	0	1	1	....
3	1	0	**0**	0	0	1	1	1	1	....
4	1	1	0	**1**	1	0	0	1	1	....
5	0	0	1	1	**1**	1	1	0	0	....
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
-	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	....

Here, inside of the table is the alleged listing of all possible subsets of natural numbers, represented as strings. The string under the table differs from the first line in the table in the first bit, from the second line of the table in the second bit and so on. Since this string looks like the diagonal of the table inverted, the method is called diagonalization.

How do we show that this diagonal string that we constructed is not in the listing? Suppose it is, then it is in the table as a row number k for some k. But this is not possible, because our diagonal string differs from the string in row k in its k^{th} bit. Therefore, this string is not in the enumeration. Since this works for any possible enumeration of subsets of natural numbers (just construct a corresponding string for each enumeration), we conclude that $2^{\mathbb{N}}$ is not countable. However, it has the same size as the set of all real numbers (exercise: see how you can prove that). It makes sense to ask is there a set with cardinality strictly between the two, that is, some set A such that $|\mathbb{N}| < |A| < |\mathbb{R}|$. And the answer is... not only we don't know, but either way would not contradict the axioms of mathematics.

This is called the *Continuum Hypothesis*. It says that there is no set whose size is strictly between that of natural numbers and that of real numbers (that is, between $\mathbb{N}$ and $2^{\mathbb{N}}$. This hypothesis is not provable (or disprovable) from the current axioms of mathematics (of Zermelo-Fraenkel set theory ZFC).

Another interesting application of this method is to show that there are problems that cannot be solved by any (say, Java) program. Think about the simplest kind of problems, classification of inputs – or even simpler, determining whether a given input is a number belonging to a specified set. For example, such membership problem can be determining, given a natural number, whether that number is prime or whether it is a square of another number. As you are already noticing from the definition of the problem, here a "problem" (often called "language" in this setting) corresponds exactly to a subset of natural numbers discussed above. So as before the rows of our table will encode subsets of natural numbers, with each column corresponding to a number that can be given to a program as an input.

Now, we want to enumerate the rows of the table by Java programs, where a Java program associated with a row should correctly compute all values in this row (for example, if the row corresponds to 1s for all prime numbers, the Java program should correctly determine whether a given number is prime). Note that any Java program can be written as a finite binary string, and we already said that finite binary strings (with 1 in front) form a bijection with natural numbers. In particular, there are not more Java programs than natural numbers. Now, applying diagonalization exactly the same way as we did above, we obtain a language that differs from the language computed by i^{th} Java program on the i^{th} number. Therefore, this language is not computable.

At this point you may ask if there is a specific, concrete example of a language not computed by any Java program. Indeed there is, and to construct it, we will use intuition akin Russell's paradox $A = \{x \mid x \notin A\}$. This particular example is called the Halting Problem, and it corresponds to a very natural task of checking whether a given program goes into an infinite loop on a given input or eventually stops (halts). More precisely, let CheckHalt be an algorithm such that CheckHalt(M,x) prints "halts" if M terminates on input x, and "loops" if M does not terminate. Let $Diag(X) = \neg CheckHalt(X, X)$. Such a $Diag(X)$ gives a paradox (just think what $Diag$ would do if presented with its own code as an input).