Beginning with Google, it can translate one language to another. Converting one language to another is pretty much challenging for humans; learning one language gets very difficult for us. However, Google can translate up to many languages. On the other hand, we can search for any information on google, getting instant information. This process requires very high power from the hardware system; otherwise, it will be impossible to perform this operation. Thus can we say that computers have no limitations? The answer is no. Computers do have some set of limitations.
For example, during the middle ages, scientists or physicists wanted to create a type of machine in which no energy would be required, or in other words, the machine will infinitely keep on operating; however, this was impossible to create because of "the laws of physics" which states that energy cannot be created or destroyed, but it can change to other forms. Similarly, computers have laws that forbid/restrict them from operating in a way. This law is called "the laws of computation." Laws of physics tell us the limitations and possible boundaries within which nature operates; meanwhile, laws of computation tell us about the limitations of computers and sets the boundary within which the computer can work.

Here automata theory is the study and analysis of machines. It also studies the computational Limitation machines by which the problem can be solved. One example is the gumball machine. In this machine, suppose the user can get one gumball by rotating a knob attached to the machine for at least 15$. Keeping in mind that only $5 and $10 can be used. Thus, the following actions can occur 5$,10$, and R(the release state, which means the user tries to rotate the knob). This working mechanism can be represented/designed as something called a state diagram which illustrates all the possible actions that take place. Suppose the initial state initially is at 0$ the following actions can be performed from the 0$ state. And those are, the user can add 5$, if this is added, then we say the current state is at 5$ and so or 10$ or R. If R is pressed, then the state does not change. Similarly, for 5$ and 10$, the user can perform the three actions, which are 5$,10%, and R. The last state will contain above or equal 15$. And even in the last state, we can add 5$,10$ and R. If 5 and 10 are added, then the state remains at the last state; thus, the value keeps on increasing, but the user can only get one gumball; however, if R is added then the state changes to the first initial state which was at 0$ state. This is how we compute a state diagram for the gumball machine.

Furthermore, another example of such a machine is the slot machine. Using the automata approach, the previous machine followed a sequence of steps performed one after the other. Here, if we use the automata theory, we will find that it is impossible to find a state in which a coin will pop out due to many possible combinations of symbols; thus, we can say that not all machines can be designed out without first knowing about the limitations and the machines. This is slightly similar to the translator of google, which concludes that not everything is possible, but we can overcome the difficulties by using different approaches.

BASIC:
SET
Sets are the collection of similar and dissimilar elements. The members of the sets are called elements. A set is represented by {x,y,z} the two-second brackets enclose the set and the comma-separated each element. For example, a collection of positive integers in a set as such {23,8,0} in which there are three elements. Some symbols are used to represent some information, such as ϵ, which means that an element is a member of a set, and ∉ means that an element is not a member of that set. An example of the above notation is that,
8 ϵ {23,8,0} which is true and 7 ∉ {28,8,0} which is also true.

A subset (⊆) means that a set is a subset of another set. In other words, a member of one set is present in the other set too. A proper subset(⊂) says that if an element is present in both the sets and the size of B is less than A, then only it is a proper subset.

Usually, when there is a repeated element, we only consider it one element when we list them out; however, there can be many same elements in the multiset. On the other hand, an empty set(φ) is a set in which there are no elements present, and the size of that set is 0. There are some set operations which are union(U), intersection(⋂), differences(-), and compliment(X'). When we write AUB, then an element x can be either in set A or B suppose, A ={1,2,3} B= {4,5,6}
So AUB = {1,2,3,4,5,6}. Meanwhile, for A⋂B, an element x must be present in both set A and B. For example, A⋂B = {φ} it is empty as no elements are in common. If we talk about the difference between A and B (A-B), then we say that all the elements of A minus the common elements with B(intersection between A and B). So, in short, we can say that A-B = A-(A⋂B). Similarly, compliment means that only the elements of that one set are being complimented but not the intersection or common elements with the other set. Suppose A' represents all the elements of A but not the A⋂B. Venn diagram is the diagram used to represent the sets.

SEQUENCE

Sequence or Tuples is the collection of elements in order. Unlike sets, the sequence is represented by the first bracket, for example, (3,4,5). In sequence, this (3,4,5) is **NOT** equal to (4,3,5) as the order matters here. Some examples of sequence are 2D - (x,y) is not equal to (y,x),3D- (3,4,5) is **NOT** equal to (4,3,5) and vector- [1,2,3] is not equal to [1,3,2]. Other names given are tuples(for finite elements) and k tuple(here, k can be any positive integer). Power set is the set of sets; in other words, it combines sets containing the null set. An example of a power set, A = {1,2,3}
So power set will contain the following elements : {{o},{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}}
So we the power set will have 2^|A| number of elements. A cross product or cartesian product of set A and B is the set of all ordered pairs(x,y) in which x belongs to A and y belongs to B. An

example of cartesian product is: A={1,2,3} B={4,5} thus AxB = {(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)} for BxA then x belongs to B and y belongs to A and the same process is applied. Some properties must be kept in our minds which are:

1. commutative (AUB = B U A and A ∩ B = B ∩ A)
2. Associative property: (A U B) U C = A U (B U C) also (A ∩ B) ∩ C = A ∩ (B ∩ C)
3. Distributive property: A U (B ∩ C) = (A U B) ∩ (A U C) also A ∩ (B U C) = (A ∩ B) U (A ∩ C)