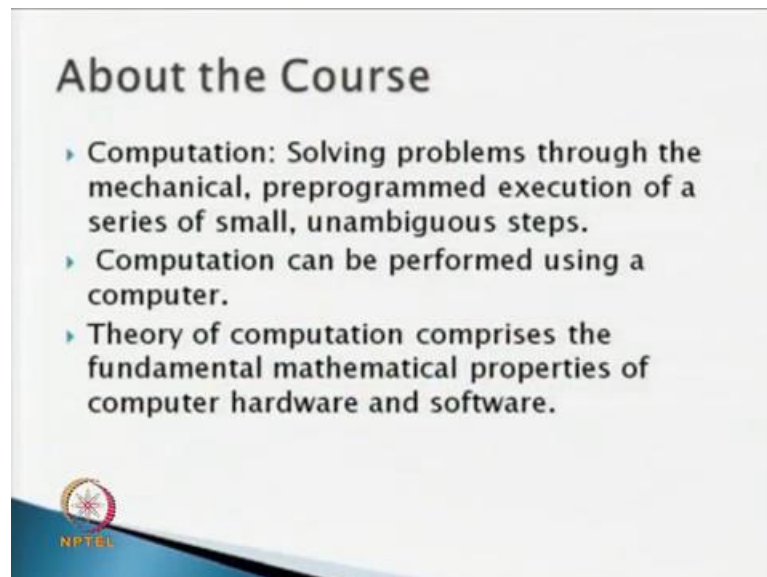


Formal Languages and Automata Theory
Prof. Diganta Goswami
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 1
Languages and Finite Representation
Lecture - 1
Introduction

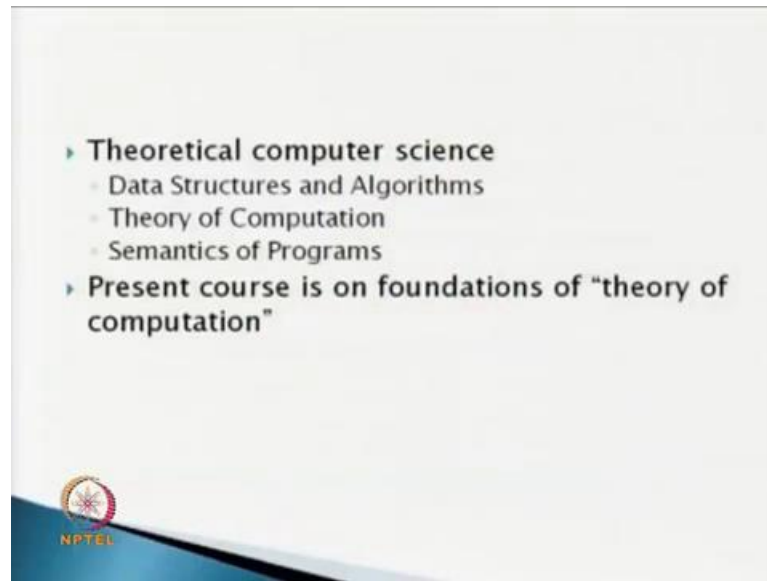
This course this which formal languages and automata theory. In this lecture interruptible lecture simply discuss what this motivation behind this course. The content of this course how will dealing this course, what are the books what are the basic recommends and so on? So, this course will be taken ((Refer Time: 00:45)) I am Diganta Goswami the faculty of computer science department. And my colleagues K V Krishna from mathematics ((Refer Time: 00:58)) IIT Guwahati. Shall briefly discuss course, the contents; what are the books, how the topics will ((Refer Time: 01:08)) in syllabus.

(Refer Slide Time: 01:22)



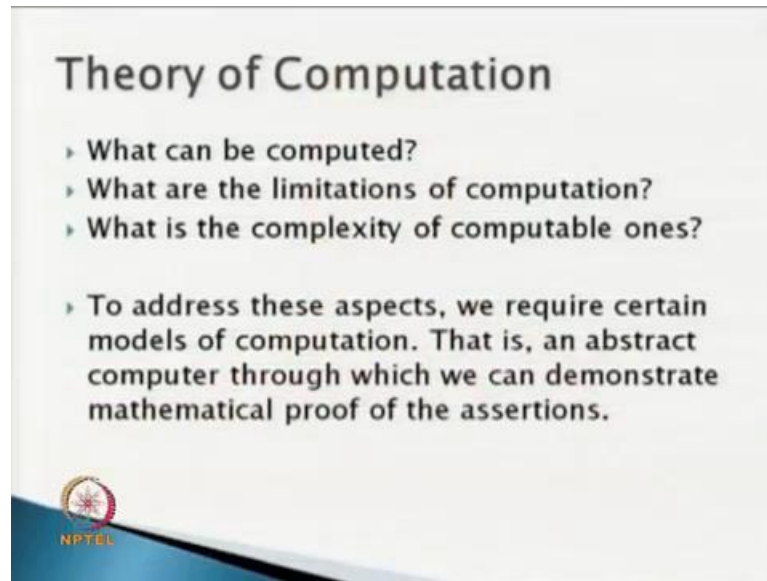
And what are the prerequisites for the discuss; which we all know that computation is basically solving problems through the mechanical preprogrammed ((Refer Time: 01:26)) of a finite number unambiguous steps. Computation can be performed using a computer that is known to us. And theory of computation comprises the fundamental mathematical properties of computer hardware and software.

(Refer Slide Time: 01:47)



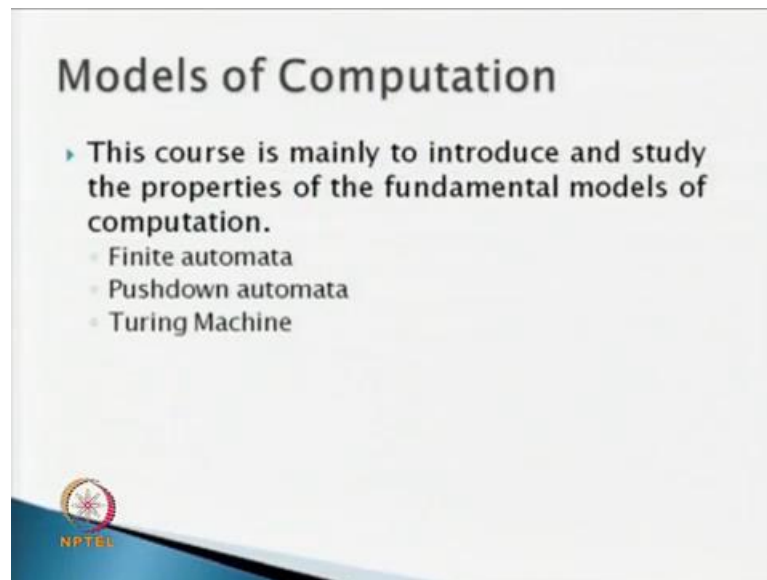
And, theoretical computer science and basically comprises few components for says data structures and algorithms. For solving any problem we to right algorithm that means you have to follow the finite number of steps which are unambiguous steps which when followed in sum order; it also particular problem; while executing the algorithm we need to store, access data, many ((Refer Time: 02:15)) and so on. And for that purpose of course some ((Refer Time: 02:20)) is called data structures; for example, takes q and so on. So, in data structures and algorithms discuss all those concepts various objects and algorithms; then to implement those algorithms we need to write in sum programming languages. So, therefore semantic programs comes into the picture and then theory competition already we have say that various ((Refer Time: 02:56)) both hardware and software. And this present course is foundation course for theory of competition.

(Refer Slide Time: 03:07)



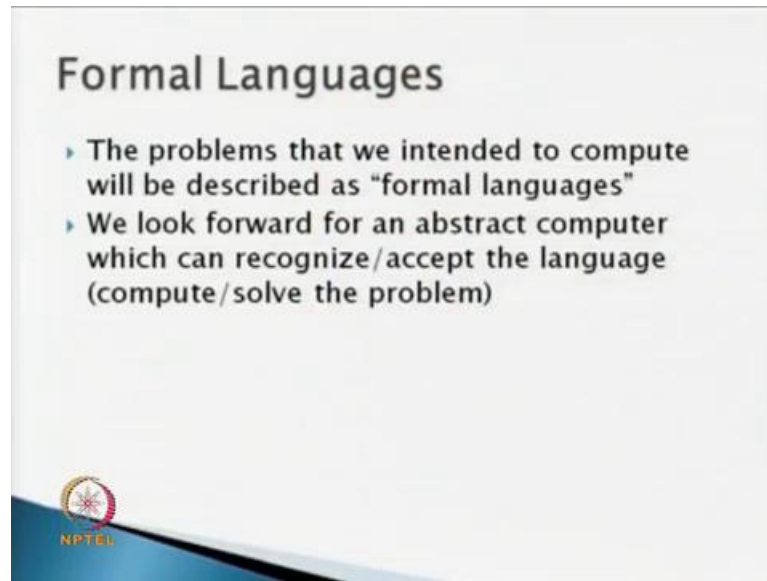
So, what is there in theory computation? So, mainly what are we discussed here is that what can be computed; we have large number of problems in the world, in the universe ((Refer Time; 03:18)) finite. So, can I compute or can I solve all those problems that is what we have to see include a computation. So, if can you compute why we can compute or what are limitations of computation that will have to discussed here. Then, we that is can compute or solved a problem how difficult it is ((Refer Time: 03:44)) complexity ((Refer Time: 03:45)) that also we discussed in this course. But obviously to address all those aspects we require ((Refer Time: 03:54)) more than sub competition. So, ((Refer Time: 03:59)) abstract computer through which we can demonstrate the mathematical proof of the assertions we make about all those. Now, this course mainly to introduce some starting the property of, the fundamental models of computation.

(Refer Slide Time: 04:19)



Since, we have say that we need various kind of abstract machines or abstract devices and the different kinds of abstract devices, abstract computers or models. So, ((Refer Time: 04:29)) basically the different kinds of models of computation and the properties. For example, will have here very simple models of computation that is finite automata; then we have pushdown automata, then turning machine which is most power full is all most models of computation. We will introduce those concepts, those machines and study of various properties of those motors. Now, the problems that we want to compute which those abstract devices the models of computation; we discussed as formal languages.

(Refer Slide Time: 05:14)



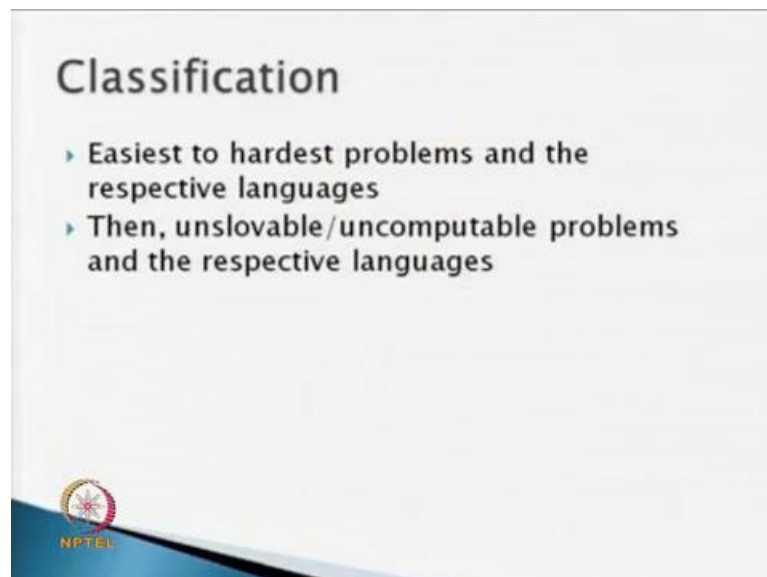
Formal Languages

- ▶ The problems that we intended to compute will be described as “formal languages”
- ▶ We look forward for an abstract computer which can recognize/accept the language (compute/solve the problem)

NPTEL

Therefore, we need to look forward for an abstract computer which can recognize or expect those languages; when an abstract computer or those model can accept language we sat that it can solve the problem. Because whenever it accept the language means it has accept solve that problem whatever motion.

(Refer Slide Time: 05:50)



Classification

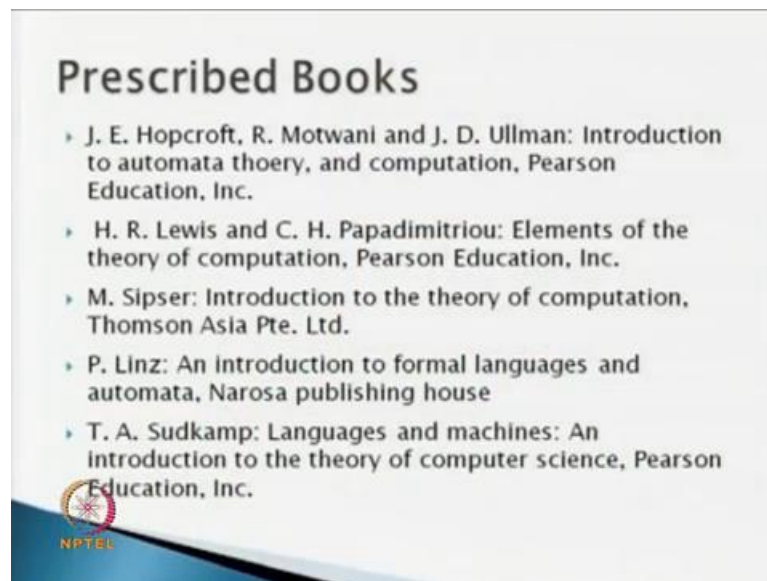
- ▶ Easiest to hardest problems and the respective languages
- ▶ Then, unsolvable/uncomputable problems and the respective languages

NPTEL

Now, whenever we say that a problem gets solved there will be some easy problem and there will be some hard problem ((Refer Time: 05:54)) classify problem to get this based on their complexity or hardness; easiest to hardest problem. And then since

problems are given irrespective languages what are the corresponding languages to each of those. Then, we have some problems which cannot be solved at all by any computing devices; that means so that no computer device to solve a particular problem; that means unsolvable or un computable problems. And then what is the corresponding languages for those language. So, since we have said that they are various models of computation and gives the problems as formal languages as input to the computer devices. Therefore, corresponding to its automata class of languages.

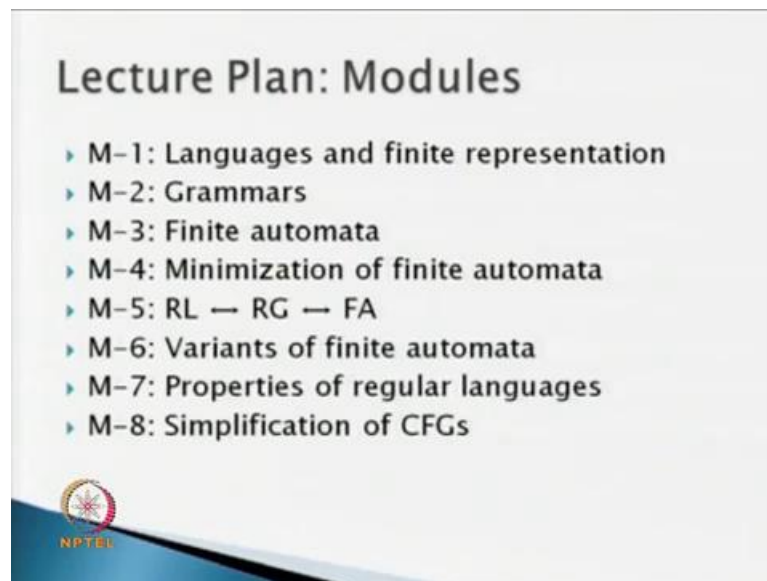
(Refer Slide Time: 06:54)



For this course, we have large number of the books available both Indian and foreign publication. So, but there are some classical books. For example, the book by Hopcroft, Motwani and Ullman. So, this one a classical book that we can refer to; whenever we have since we have define books we will see the different books discusses or introduces or ((Refer Time: 07:35)) problem in different ways. Say some books will discuss this automata first; various models of computation. Then, formal languages every book of course will first introduce you the basic concepts like ((Refer Time: 07:54)) languages. And then various times of language will be discussed and then automata discussed. So, some first discussed automata; then introduces grammars which a mechanism to generate languages. For example, in these books how Motwani and Ullman first discussed this automata; then introduces grammar and so on. Then, another book that book by lives and ((Refer Time: 08:29)) Lomita ((Refer Time: 08:30)) is very classical book. Then, ((Refer Time: 08:44)) introduction of theory of computation and good

book; when we have book by peter limes which discusses lot of numbers of examples; may be useful ((Refer Time: 09:00)). Then, book by ((Refer Time: 09:03)) languages and machines and gets theory of computer science. Now, this book basically first discussed this grammars, languages and then automata. So, whenever you have any doubt may be you can refer to any one of those books and get clarification.

(Refer Slide Time: 09:30)



Now, we will see that what we were going to cover the various contents of this course; what are the content and what are the flow of topics basically that is what we need to want to discussed here. So, this is course into 15 modules and each module each compression of many lectures. So, module 1 contains many say 2 lectures. So, initially we give the basic concepts; what is the alphabet, strings and then formal defining languages? And then we go for finite ((Refer Time: 10:14)) and that what mean by finite ((Refer Time: 10:16)) is that since we have say that we need to use computer to compute; and problems are given in terms of even formal languages. But you know that languages may be finite or infinite. In case of finite is fine you can give you a input to the computer finite ((Refer Time: 10:35)). But in case of infinite language we need to represent that language using some kind of finite representation; and then give to an input computer. So, therefore for every language we need to have some kind of finite representation; so what is the finite representation and how will represent any language using finite amount of information; that is what we discussed over here. So, we will use some kind of tool called say regular ((Refer Time: 11:03)) which can be used to

represent any infinite language is infinite ((Refer Time: 11:14)). But it is not case the very language can be written by using any regular explanation; they remain languages for which is not possible the regular explanation. Then, will go for grammars in module 2; which is again divided into 2 to 3 lectures. So, grammar is basically again ((Refer Time: 11:46)) tool which can generate languages. So, it is also finite ((Refer Time: 11:51)) for languages.

So, here first we will discussed on a kind of grammar called context free grammar; so in this grammar we have rules to generate the strings in the languages. Now, we can impose some restrictions and the rules of the grammars or we can more generate grammar; based on this ((Refer Time: 12:19)) we have different kinds of grammar which can generate different languages. So, firstly discussed the context free grammars and then will see how we can derived strings in a languages? So, one is derived is string in a language there is other ways to which as technique of derivation 3 ((Refer Time: 12:49)) which is also useful in ((Refer Time: 12:53)) a programming language. Then, we will impose from resection the conduct the grammar and we will see that they grammar called regular grammar; which is actually equivalent to the regular expression. That means, the way we representing that the class of language that is generated by regular grammars equivalent to the class of languages represent by regular expressions.

After that in module 3 we discussed we come to say automata; first we discussed the simply kind of automata which is called finite automata. So, all this automata basically abstract computing devices, models of computation and then compute or accept different kinds of languages; that mean we can solve different languages. So, finite automata is simplest of all; that is way computing power will be limited. Then, introduce ((Refer Time: 14:09)) infinite automata by the default find automata is determined mistake; non determine the important concept is course assign we introduce the concept of ((Refer Time: 14:20)).

And then we have another model is called ((Refer Time: 14:23)) simply is an f. But we can so that the deterministic automata and non defendable automata both here equivalent; they will we shown will be its module itself. Then, in module 4 we come to minimized of automata; what is done here is that since we have a suppose we have a language and we can construct find the automata through except that language; we can

construct in many different ways; in some cases the automata will number of ((Refer Time: 15:15)) in another case number of cases will be more but both are equivalent in the sense that accept the same language.

So, therefore for any language would like to asked is there a find automata with minimum numbers substance; yes that is what we can do. For that purpose we can be introduced a characterization for the class of language is x and y ; the regular expression. And then becomes so that we can construct a find automata which minimum numbers subsets. Then, we go for showing equivalence the various concepts like say regular language in which regular grammar and find automata; a class of language accept by regular expression is said to be regular languages that way. And so that the define automata and regular expressions they are equivalent; the class of language except by the expression is equivalent to the class of language accept by automata. Similarly, regular languages and regular grammar the equivalent regular grammar is class of regular languages so on. So, will equipment in the various models in module 5.

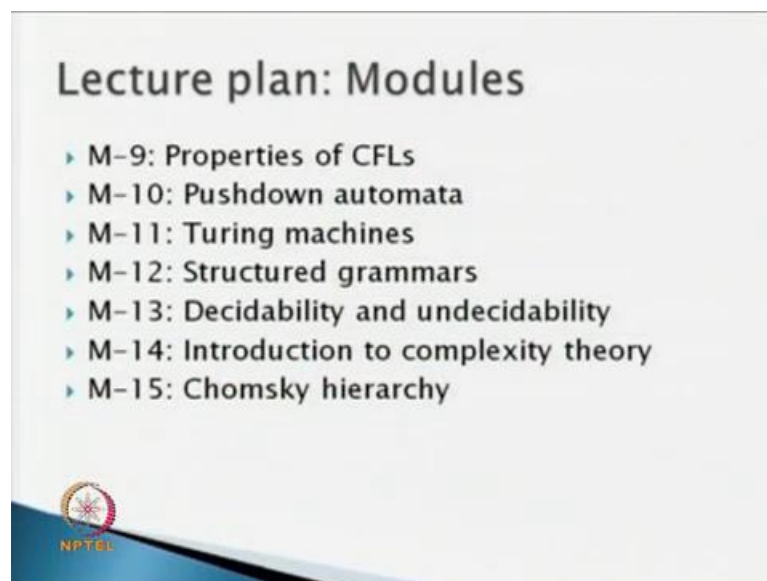
In module 6 we discuss some variation of find automata; so whenever normally will discussed find the automata we see that accepts a class of languages particular language. But find the automata can sometimes also produce some output; given some input it may be produced some outputs. So, in that context we will see move machine and ((Refer Time: 17:23)) which produced some output. And then we will also same and other relevant of find the automata is ((Refer Time: 17:37)) where we read in head can move ((Refer Time: 17:43)); in normal find the automata reading in head only move in one direction. But will case ((Refer Time: 17:49)) it can move back input. So, this various kinds of automata will be discussed in module 6. Module 7 we will discuss property of regular languages. So, what are the various properties of generic characterized of regular line thus; that will be discussed in this model 7.

Again, suppose as a languages were it is languages regular no; suppose it remains not regular then how can you so that this language is not regular ?Whatever we have tool do this he will discussed here in module 7 using sum kinds of ((Refer Time: 18:22)) languages. So, that a language is non-regular and then many ((Refer Time: 18:41)) for regular languages we will proved over here. Then, we will go for implication of concept of grammar we have already introduce in module 2. Then, in that grammar we

will have will many different kinds of rules. And so that some rules are not really necessary and you can simplify the grammar by elevating or simplifying sounding ((Refer Time: 19:11)). And still we have an equivalent language; that means given a grammar we can simplify that grammar and still this grammar we generate same languages as the previous one; by simplification we means reducing the number of rules that will having grammar and so ((Refer Time: 19:39)) simplification.

And, then we will discuss some extended forms which we useful some standard forms of grammar which are called normal forms; which are useful in proving many theorems. And how to given a grammar, how to arrive at standard form for the grammar; which generate that a same language this is the original one that will discuss in this module; there many different kinds of standard forms; we are not going to discussed all kinds if standard forms only a few forms can be discussed.

(Refer Slide Time: 20:24)



In module 9 we discuss we properties of CFLS similar to the properties of regular languages. The properties of CFLS means conduct free languages; the language is generate by consider the grammar ((Refer Time: 20:34)) properties and some another properties likes say given any language ((Refer Time: 20:42)) conducts languages. So, how to prove that we have some tools to proved that and we will discussed in this module. So, as if said earlier is that which its abstract devices or automata; we associate the class of languages. For examples for finite of automata the class of languages is the

regular line ((Refer Time: 21:14)). Similarly, for the corresponding to this ((Refer Time: 21:22)) language that the automata is the corresponding automata is whose down automata.

So, to see how ((Refer Time: 21:30)) automata is different from finding the automata that means you can enhance by the automata by headings some more features in it. So, that it can accept different kinds of languages which nothing but the context free languages; we will also in the process that the class over the regular languages the proper subset of conversed languages. That means, whose down automata is more powerful in terms of accepting languages. And then we will prove here that who's on automata and context grammars their equality; we will so that if we have a presume automata you can construct and equivalent ((Refer Time: 22:33)) to it.

Similarly, if we have it context grammar I can contact equivalent automata from it. Then, in module 11 we discuss and I told you most powerful computing model that is ((Refer Time: 22:57)) machine. The ((Refer Time: 23:00)) is the most powerful computing devices in the sense that whatever it is machine can whatever a general purpose computer can do the present general purpose computing then at termination also give to the same thing. In the sense that is capable of computing whatever the general purpose computer can compute do. So, see how we can use a ((Refer Time: 23:30)) to do compute a function or different functions; then will use some modular approach in the sense that we can contract very complex ((Refer Time: 23:50)) theory eliminate from the complex functions using some simple theory machines which can compute some simple functions.

That means, you can combine ((Refer Time: 24:01)) to compute more complex functions. Then, also introduce to algorithms to ((Refer Time: 24:13)) and the different kinds of ((Refer Time: 24:19)) that means various of ((Refer Time: 24:21)) machines. For example, in normal ((Refer Time: 24:23)) of basic model we will have only one step there may be multiple step in ((Refer Time: 24:27)) there is a one variant the multi head ((Refer Time: 24:30)) two infinite step; normally we have single way infinite step when basic termination. But that may have two infinite as well we can introduced; again so that all this models are equivalent to the basic termination automata. In module 12 we will discuss structured grammars; which can compute any kinds of functions that computer by ((Refer Time: 25:04)) machines. That means, class of languages generate

by such grammars is equivalent to the class of language except by ((Refer Time: 25:12)) machine. So, we will introduce here grammatically compatible functions, structure grammars basically the most general form of grammar; then model starting we discussed ((Refer Time: 25:35)).

So, what can be ((Refer Time: 25:38)) and what cannot be decided? So, that many problems normally the problems related to languages their we have already introduced; many problems have say desirable. For example, given a finite automata and a string ((Refer Time: 26:03)) except device automata difference for this belongs to a language which is regular. So, this problem what are you this decidable and decidable; when so that this is decidable problem. So, see many other line problem can be related to languages shown to be in decidable. So, will discuss some problem which can be solved. And then we will discussed undecidability; that means they are many problems which cannot be decided using any computer devices. That means, for example saying ((Refer Time: 26:43)) machine consider any ((Refer Time: 26:44)) machine but we cannot compute or we can solve that particular problem.

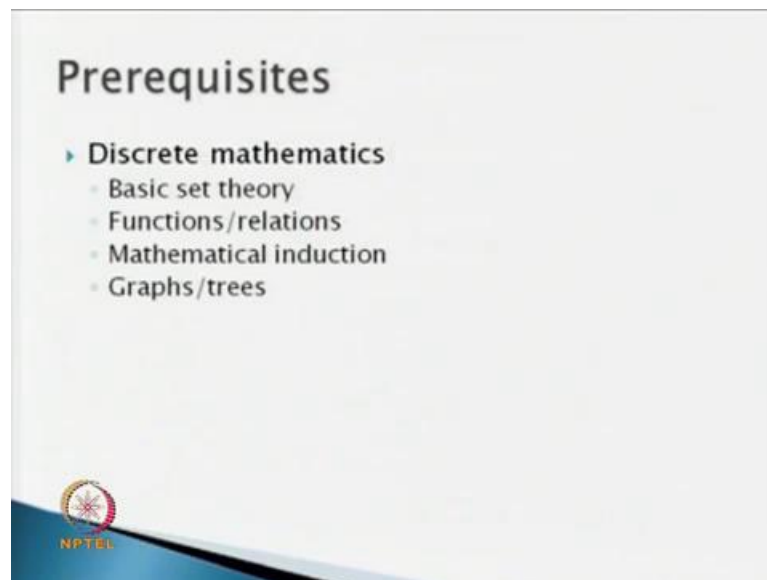
So, in this discuss we need to use some tools, some methods in which so that this problems are not decidable; it will used the concept of diagonalization and first show that ((Refer Time: 27:16)) machine is undesirable. And from there so that many other problems are also undecidable; we will also simply introduced and the problem is called P C P ((Refer Time: 27:30)) which can be used so that any problems it is languages are undecidable. So, once we have problems which are decidable we want know how difficult those problems are to solve. So, in module 14 we discussed that issue or introduced those issues; so discusses ((Refer Time: 27:56)); how complex problems are? Can you= classify the problems based on the complex level? So, they are of course large number of classic language but here will simply introduce only the basic classes, for example P, N P, N P complex.

Here, again we have used to many different tools; for example say reduction we want to so that if can solve one problem and we can also solved another problems. That means, they are having similar complexity level. So, here will so that one problem is N P and then using that using that assuming that problem a complete we can say that many other problem N P complex using this model reduction. First to show that one problem can be N P ((Refer Time: 29:11)) we use this hooks theorem we introduced this hooks

theorem and prove it and so that problem is N P ((Refer Time: 29:18)). And then use the method of ((Refer Time: 29:0)) many other problem are N P((Refer Time: 29:24)).

We give various different kinds N P compute problem by the method of reduction; so that all this problems and can be N P compute. And finally we conclude this lecture by giving a hierarchy of language classes. So, this is known as Chomsky hierarchy which named after a famous linguistic ((Refer Time: 29:50)). There are various languages for example ((Refer Time: 29:54)) of the formal languages conduct languages written this languages, recursion languages and recursion language ((Refer Time: 30:06)) basically accepts the class of languages to all recursively and precautionary languages. And in ((Refer Time: 30:14)) we also produces here and computing models that is called ((Refer Time: 30:20)) which accepts context sensitive languages. And corresponding grammar from we will context free context sensitive in the grammar. And here so that proper containment for example saying regular languages it properly to complex languages by profile can be context line is ((Refer Time: 30:43)) is recursion languages. And finally recursive languages ((Refer Time: 30:51)) recursively any would other languages. And this have key is known as hams key headed.

(Refer Slide Time: 31:21)



So, this all about the plans or flow of topics over in this course; but this course recurse basic concepts from prerequisites. For example, one will have the basic knowledge of set theory will set continuity of set; how would you find continuity of set such as

infinite sets we have so on. And this various operations from sets any other so on. Then, functions and relations what do you want to one function or bijection those things; for equivalent relation it can be class ((Refer Time: 31:59)) those concepts will be occur. Then, most of results or theorems that will have in this course will be proved. And many defined prove its can be used but most commonly using mathematical induction. So, were we have a ((Refer Time: 32:26)) inductive hypothesis; this will have some resistance, will have inductive hypothesis and will have inductive ((Refer Time: 32:33)); to so that prove the result there is kind of induction called structural inductance which will also be useful.

For example, we have that can be applied many structures which are recursively fine. For example, ((Refer Time: 32:56)) recursion definition; there we can apply the structure induction to prove binary ((Refer Time: 33:02)). Then, some concepts are graphs ((Refer Time: 33:09)) and because this ((Refer Time: 33:14)) we can moral it can be graph, it can be ((Refer Time: 33:17)) ((Refer Time: 33:32)) so on numbers of accurate will be levels of division and so on. And many other purpose ((Refer Time: 33:37)) for example, ((Refer Time: 33:39)) in a particular language can we consider a kind of tree structure which is called tree is called ((Refer Time: 33:48)). So, always concepts will required; ok once we have a basic concepts we start having following lectures very easily. So, let us all about the interaction of this discussed.