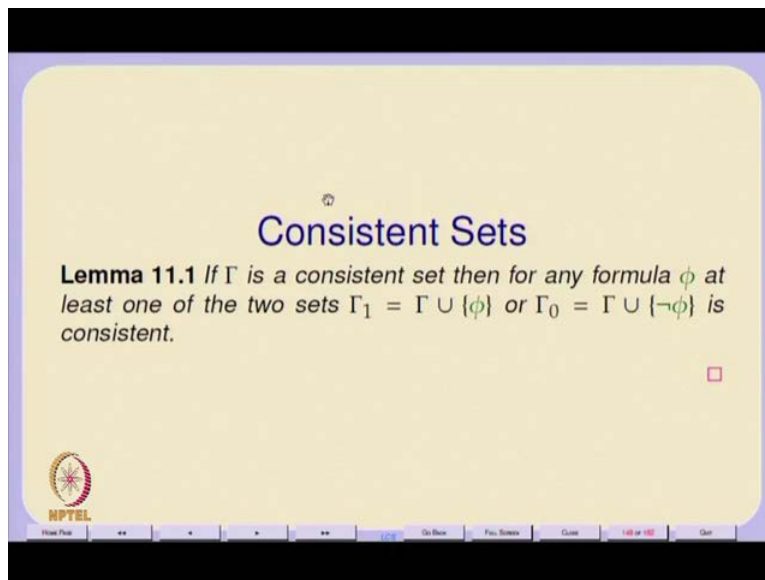


Logic for CS
Prof. Dr. S. Arun Kumar
Department of Computer Science
Indian Institute of Technology, Delhi

Lecture - 12
Formal Theories

(Refer Slide Time: 00:57)



Consistent Sets

Lemma 11.1 *If Γ is a consistent set then for any formula ϕ at least one of the two sets $\Gamma_1 = \Gamma \cup \{\phi\}$ or $\Gamma_0 = \Gamma \cup \{\neg\phi\}$ is consistent.*

□

NPTEL

So, I wanted to start formal theories today. But, since there was some confusion last time about maximal consistent sets I thought I will and since there was some errors in the slides. I thought it would be best just go through it once and then come to formal theories. So, I will quickly go through it so you have so we have the notion of maximally consistent sets. Basically, what we are saying is given any satisfiable set of formulae. How many more elements from the language of propositional logic can you add? While, keeping it consistent all the time. So, it turns out that for you if you can keep on adding any formula as long as you do not add both it and its negation.

(Refer Slide Time: 01:25)

Proof of lemma 11.1

Proof: Suppose Γ is consistent but both Γ_0 and Γ_1 are inconsistent. Then by compactness and by definition 10.5 there must be consistent finite subsets $\Delta_0, \Delta_1 \subseteq_f \Gamma$ such that $\Gamma'_0 = \Delta_0 \cup \{\neg\phi\}$ and $\Gamma'_1 = \Delta_1 \cup \{\phi\}$ are both inconsistent. Let $\Delta_{01} = \Delta_0 \cup \Delta_1$. By facts 10.6.1 both $\Delta_{01} \cup \{\neg\phi\}$ and $\Delta_{01} \cup \{\phi\}$ are inconsistent and hence unsatisfiable whereas $\Delta_{01} \subseteq_f \Gamma$ is consistent. Hence there is a truth assignment τ which satisfies Δ_{01} , and such that

$$\mathcal{T}[\phi]_\tau = 0 = \mathcal{T}[\neg\phi]_\tau,$$

which is impossible. \square

So, only one of them can be in a consistent set and the proof of that of course realize on the semantics in a proof of contradiction.

(Refer Slide Time: 01:25)

Properties of Finite Character: 1

Definition 11.2 A property p of sets is called a **property of finite character** if for any set S , S has the property p iff every finite subset of S has the property p .

Notation: $S \models p$ denotes the statement " S has property p ".

And, then we define this general notion called a property of finite character. And this is much it goes outside it comes from mathematics and it is really because nothing to do with logic. But, it is something that can be used in mathematical logic. Because satisfiability or consistency of a set

of formulae is itself a property of finite character. And, that we get from the compactness theorem a property of sets is called property of finite character. If, for any set S has that property p if and only if every finite subset S also has a property p . So, our compactness essentially says that S set γ is consistent if, and only if every finite subset of γ is also consistent so that is a corollary of the compactness theorem. And so, therefore the notion of consistency itself is a property consistency of sets of formulae is a property of finite character in the universe of all propositional formulae.

(Refer Slide Time: 02:46)

Properties of Finite Character: 2

Example 11.3

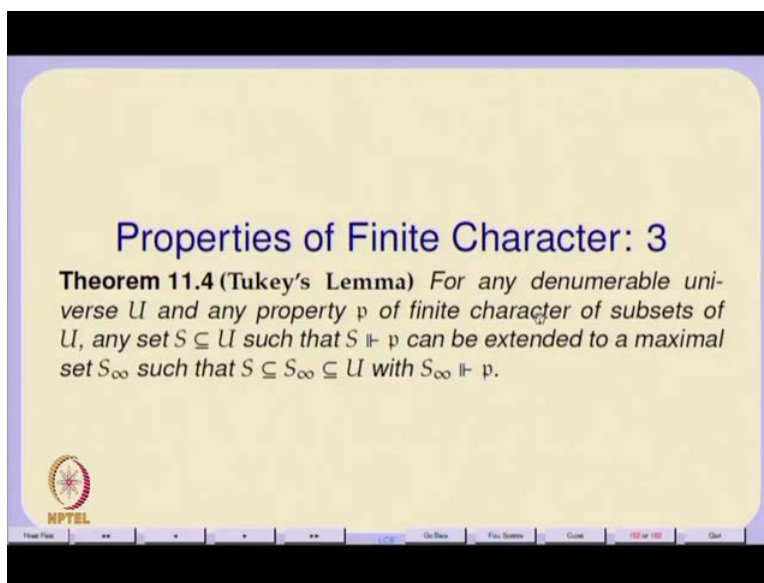
1. The property of a partially ordered set being **totally ordered** is a property of finite character. That is, if $\langle P, \leq \rangle$ is a partially ordered set, then P is totally ordered (i.e. for every $a, b \in P$, $a \leq b$ or $b \leq a$) iff every finite subset of P is totally ordered.
2. However the property of a totally ordered set $\langle T, \leq \rangle$ being **well-ordered** is not a property of finite character since every finite subset of T is well-ordered, but T itself may not be well-ordered (e.g. take the set of integers \mathbb{Z} under the usual \leq relation).
3. By the corollary 10.4 to the compactness theorem, consistency/satisfiability is a property of finite character.

NPTEL

So, what we had was this so other sort of examples as I said. One is given the universe of partially ordered sets which of the sets are totally ordered. So, this property of being totally ordered is a property of finite character. And basically, within the universe of partially ordered sets a certain set is a total order if, and only if all its subsets are total orders all its finite subsets are total orders. However, as I said the property of if you take the universe of totally ordered sets the question of, Whether a set is well ordered is not something that the property of finite character? So, the well ordering property is not a finite character because of the fact that an infinite set may not be well ordered. But, all its finite subsets and infinite total totally ordered sets may not be well ordered. But, all its finite subsets may be well ordered because they will have a minimum elements.

So, by well ordering we just mean that, there should be no infinite descending chain. And, other kind of properties of finite character that you can encounter in mathematics. As I, said with questions like, tiling of an infinite plane k colorability of an infinite graph. So, k colorability is the property of finite character and basically there is a theorem in on the theory of infinite graphs. Which says that an infinite graph is k colorable if, and only if all finite sub graphs of it are k colorable. So, and similarly there are things about tiling so there is a fairly general purpose lemma called Zorn's lemma. Which, allows us to actually extend for properties of finite character. A set having a property of finite character to some maximal set still possessing that property. So, Tukey's lemma essentially says that actually Tukey's lemma is more general it is not restricted to denumerable universes. But, since in this course will be mostly restricted to denumerable universes. Because we are restricted to languages and languages are always denumerable.

(Refer Slide Time: 05:16)



For any denumerable universe and any property p of finite character of subsets of U any set S subset of U . Such that S has the property p can be extended to a maximal set S_∞ . Such that, S_∞ also has a property p where, p is a property of finite character.

(Refer Slide Time: 05:38)

Proof of Tukey's Lemma

Proof: Let $S \subseteq U$ be a set with $S \models p$. Since U is denumerable its elements can be enumerated in some order

$$a_1, a_2, a_3, \dots \quad (4)$$

Starting with $S = S_0$ consider the sets $S_{i+1}, i \geq 0$


$$S_{i+1} = \begin{cases} S_i \cup \{a_{i+1}\} & \text{if } S_i \cup \{a_{i+1}\} \models p \\ S_i & \text{otherwise} \end{cases}$$

Clearly we have the infinite chain

$$S = S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots$$

such that for each $S_i, S_i \models p$. Let $S_\infty = \bigcup_{i \geq 0} S_i$.

Claim. $S_\infty \models p$.



NPTEL

And essentially I had made some mistakes with inducues. But, now I hope that I, have corrected all the inducues. So, essentially what we are saying is since our universe is going to be is a denumerable set we just we can enumerate its elements in some order a_1, a_2, a_3 etcetera. And, you we can consider an infinite chain of sets S starting with S naught being equal to the given set S . Assume that S has the property p and we can construct S_i plus 1 as consisting of S_i and a_i plus 1. If, this S_i union a_i plus 1 has the property p otherwise S_i plus 1 is just the same as S_i . And, therefore with the assumption that S_i has a property p by construction. Since, we started off with S naught which has a property p all subsequent S_i 's will have the property p all subsequent S_i 's are also finite. So, our infinite set S infinity is just the union of this chain.

(Refer Slide Time: 05:38)

⊢ Let $T \subseteq_f S_\infty$, then $T \subseteq_f S_i$ for some $i \geq 0$. Since $S_i \models p$, p is a property of finite character and $T \subseteq_f S_i$, $T \models p$. Hence every finite subset of S_∞ has property p . Therefore since p is a property of finite character, $S_\infty \models p$. \dashv

Claim. S_∞ is maximal.

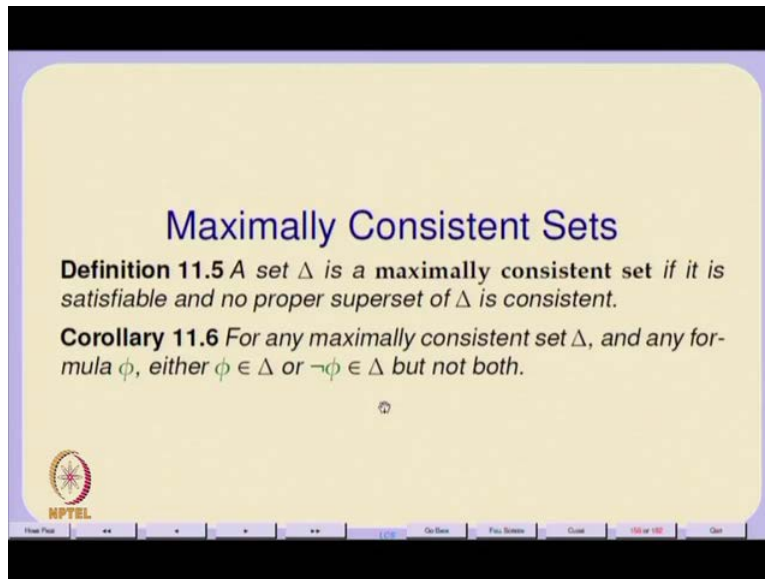
⊢ Suppose there exists an element $a \in U$ such that $S_\infty \cup \{a\} \models p$. Since p is a property of finite character, for every $T \subseteq_f S_\infty \cup \{a\}$, $T \models p$. In particular, for each $i \geq 0$, $S_i \subseteq_f S_\infty \cup \{a\}$ and $S_i \models p$. Further $a = a_{i+1}$ for some $i \geq 0$ in the enumeration (4). Hence $S_i \cup \{a_{i+1}\} \models p$ since $S_i \cup \{a_{i+1}\} \subseteq_f S_\infty \cup \{a\}$ and p is a property of finite character. But then $S_{i+1} = S_i \cup \{a_{i+1}\} \subseteq S_\infty$ and hence $S_\infty = S_\infty \cup \{a\}$. \dashv

And, what you can prove is that S_∞ has the property p . And since, p is the property of finite character it is enough to show that every finite subset of S_∞ has this property p . So, we assume any finite subset T and because of our enumerations and so on and so forth. This T must be contained in some S_i . For some i greater than or equal to C think of this just T is our finite set it consists of elements a_j . Where, j is the index in the naturals take the maximum index this finite since T is of finite set there must be a maximum index let us say m . Now, this T must be the subset of S_m for example. And since, S_m would satisfy p and p is a property of finite character T would also satisfy the property p because, T is a finite subset of S_m . And so, for a any arbitrary finite subset of S_∞ we, essentially show that T has property p .

And therefore and since, p is a property of finite character S_∞ also has the property. And, the fact that S_∞ is maximal is very easy. Assume you have some element a , which can be added to S_∞ so, you have $S_\infty \cup \{a\}$. Now, this so this $S_\infty \cup \{a\}$ is an infinite set having the property p , p is a property of finite set character. Therefore, every subset of $S_\infty \cup \{a\}$ also has the property p . Moreover since, this element a occurs somewhere in the enumeration, it must have some index $i + 1$ let us say. So, then what it means is that, since every subset of $S_\infty \cup \{a\}$ has this property p in particular $S_i \cup \{a_{i+1}\}$ would have this property p . But, $S_i \cup \{a_{i+1}\}$ has this property p is just S_{i+1} . And,

therefore S_∞ is the same as $S_\infty \cup \{a\}$. And therefore, you cannot extend S_∞ anymore and therefore it's maximal.

(Refer Slide Time: 05:38)



Maximally Consistent Sets

Definition 11.5 A set Δ is a **maximally consistent set** if it is satisfiable and no proper superset of Δ is consistent.

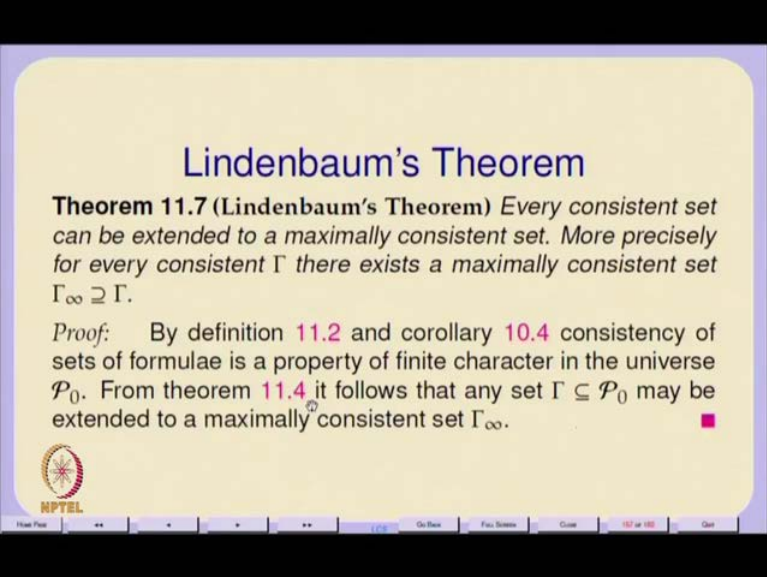
Corollary 11.6 For any maximally consistent set Δ , and any formula ϕ , either $\phi \in \Delta$ or $\neg\phi \in \Delta$ but not both.

NPTEL

Slide navigation controls: Home, First, Previous, Next, Last, Refresh, Search, Close, 108 of 108, Quit

So that, is what we did and what the lemma essentially says is that given any consistent set Γ and since satisfiability is a property of finite character. Essentially any set Γ can be extended to a maximally consistent set Δ such that Γ is a subset of Δ . Any consistent set Γ can be extended to a maximally consistent set Δ such that Γ is a subset of Δ . And as a corollary it is easy to see that for any formula ϕ either ϕ or its negation should be in this maximally consistent set.

(Refer Slide Time: 10:21)



Lindenbaum's Theorem

Theorem 11.7 (Lindenbaum's Theorem) *Every consistent set can be extended to a maximally consistent set. More precisely for every consistent Γ there exists a maximally consistent set $\Gamma_\infty \supseteq \Gamma$.*

Proof: By definition 11.2 and corollary 10.4 consistency of sets of formulae is a property of finite character in the universe \mathcal{P}_0 . From theorem 11.4 it follows that any set $\Gamma \subseteq \mathcal{P}_0$ may be extended to a maximally consistent set Γ_∞ . ■

NPTEL

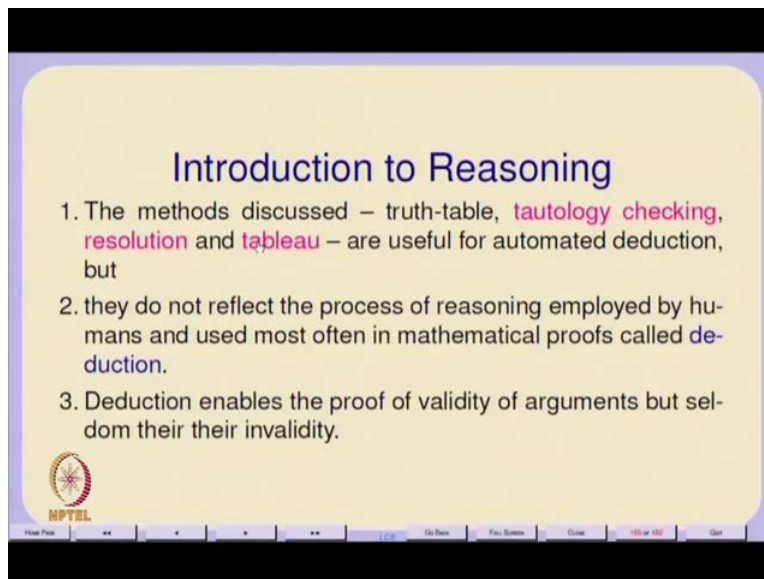
So, Lindenbaum's Theorem then follows from this fairly trivially. But, in case you want an Abnesio proof of Lindenbaum's Theorem that is also here. Which I have corrected so you should be able to look at it and convince yourself. That it is you get a gamma infinity which is that maximal and consistent all. So then let us actually formally start today's lecture. So, one of the so what we are actually been doing so far. Is really delving into semantics of the of proposition logic we have essentially been working with these semantics of proposition logic.

But if, you look at logic and its evolution it essentially started off to formalize mathematics and, branches of mathematics. Where, the notion of a mathematical reality exists only in the abstract. So, one of the things that all our methods have done is, like we our tautology checker or truth table or automated reduction they are all a tabular methods for example. So, they are all very useful for automated reduction. But, they never capture the flavor of the notion of reasoning which was what logic was originally made to do. To capture the notion of mathematical reasoning in such a way that you know it sort of it captures our own notion of reasoning of in mathematics in fairly abstract. And, domains which means that our semantics itself or of the of the models that we are interested in the abstract domains. That, we are interested in do not actually exist except through some language description. So, when we are talking about, the mathematics of a certain domain that domain is only in the abstract. And, it is described by some

language by our language. In fact usually we are using our natural language to describe all of mathematics.

The point is that we need to be able to reason entirely in that language. And, still capture this semantics of that mathematical domain. Without, actually having that mathematical domain directly available to us I mean that is what mathematical reasoning is all about.

(Refer Slide Time: 13:04)



The slide is titled "Introduction to Reasoning" and contains three numbered points. The first point states that methods like truth-table, tautology checking, resolution, and tableau are useful for automated deduction but do not reflect human reasoning. The second point notes that human reasoning is used in mathematical proofs called deduction. The third point states that deduction enables the proof of validity of arguments but seldom their invalidity. The slide also features the NPTEL logo and a navigation bar at the bottom.

Introduction to Reasoning

1. The methods discussed – truth-table, **tautology checking**, **resolution** and **tableau** – are useful for automated deduction, but
2. they do not reflect the process of reasoning employed by humans and used most often in mathematical proofs called **deduction**.
3. Deduction enables the proof of validity of arguments but seldom their their invalidity.

NPTEL

Home Page ** * * ** Go Back First Screen Close 108 of 102 Quit

So, the methods that we discuss directly went to the truth table. Which, in that sense that mathematical domain was a concrete domain already available to us. And, we all are propositional logic did was described just that domain basically. So, all our theorems are essentially of that domain which lets say is already available as a concrete domain. But if I were to start with a new branch of mathematics on a, totally abstract domain which does not have any realization in our so, called reality. Then, I have to describe it in some language, and all the statements about it are also in some language.

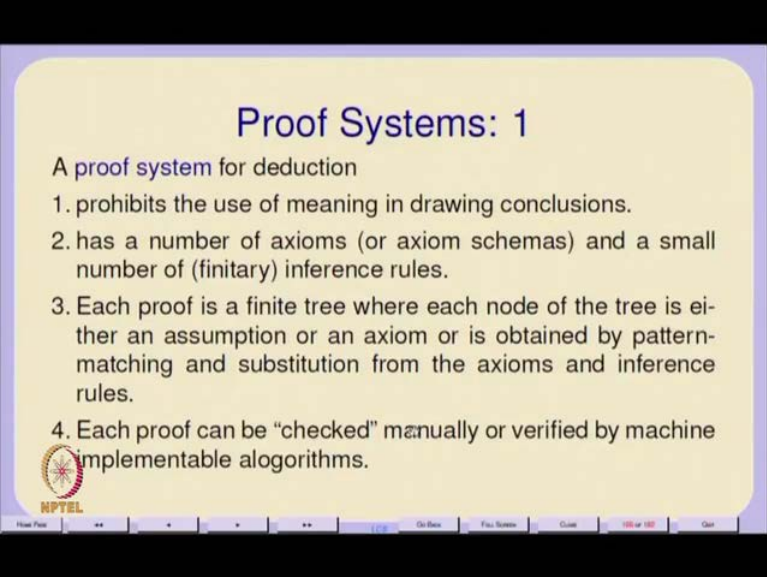
And, the actual domain itself is only the structure of it or the models which satisfy it are only available in the abstract. And, they are not available in any concrete form and yet our reasoning processes is are somehow mathematically consistent. So, we want a mathematically consistent theory. So, you take let us say take something like infinite dimension vector spaces or some such things the theory of infinite dimensional vector spaces. Then, were will go for a concrete cement

exist that which the only thing available is your language to describe infinite dimension vector spaces. And, all your reasoning has to be occur in that in that same language. You have to ensure that your reasoning is sort of consistent there are no circularities in your reasoning. There is a presentation which is logical and in some stepwise fashion. So, that there are no contradictions also and that is the duty of the formulization of those things is essentially what logics came up for. Simply because, even in a even in fairly simple domains like Euclidian geometry the problems are circularity and contradictions were there. And, so the necessity of being able to make what is what would call a logically consistent presentation was as important as being able to provide the intuition for the proofs of theros. So, that means now there are two different things happening here. One is mathematical intuition about, that domain can have fantastic flashes of insight. But, the other thing which is not necessarily directly related to the intuition is the presentation of that intuition in some logically consistent manner without contradictions without circularities and so. So, that presentation is what you expect mathematics to you expect logic to achieve. So, logic is all about reasoning about those presentations. And in that, sense since what you sine you do not have the abstract mathematical domain directly with you except that you describe it in some language.

So, the only thing you have really is the syntax of that language. So, the actual domain itself is described syntactically and what you want in your logic is to be able to do that. And, come out with a formal notion of the process of deduction the process of proof. So, the process of proof of course is again different from the process of intuiting about the properties. So, if I have an intuition about certain properties of that about a, objects in that domain. That does not necessarily mean that I can present a consistent proof. And that presentation is a, different ball game all together and that presentation is what we are expecting to do through logic. So, we expect to present essentially mathematical theories and, formulize the notion of reasoning in a consistent fashion through our notion of formal proofs. So, what we are going to do now is so what we did so far was essentially the semantics of propositional logic. So, now let us assume that we have understood the semantics of propositional logic. And now, what we want to know is, How to build theories? And describe them in some in some formal language. So, what we want to do is, we want to bring in the notion of a formal theory. And we want to formalize the notion of reasoning in that theory. Which is not like the tabular methods the resolution method or any such thing.

But, reasoning the way we do logical deductions. So, there is a process of deduction from certain abstract principles like inspired. Let us, say the axioms given by Euclid from Euclidian geometry or presentations of any mathematical domain like say group. The presentation of groups is just in terms of the 3 axioms of a group. That basically, the semi group axioms identity axiom and inverse axiom. So, that so essentially those 3 axioms say that, well whatever everything that satisfies these 3 axioms is a group. And therefore whatever, is derivable from those axioms are the theory of groups is the theory of groups. And, they will be available and they will be applicable to any group that you might chose. And throughout all that you are delving in is you do not actually necessarily have a group available in any concrete form. The group is only an abstract they are only abstract models of groups available to you. What you have is, only a language of presentation and a language of reasoning and, we have to formulize them. So, essentially so these so they do not so the methods that we have discussed so far. While, they are good for automated deduction do not reflect the process of reasoning that we usually employ. And that is the process of reasoning is called deduction. And, deduction enables but of course there is there is one thing deduction enables to prove the validity of arguments. So, essentially what we are talking about, is validity of arguments again. But without of actually a, semantic model completely described in syntax. So, but the deduction enables proof of validity but seldom enables proof of their invalidity I mean their invalidity. The invalidity construction is still by constructing a model counter example and so, we have to see what to do about that.

(Refer Slide Time: 20:08)



Proof Systems: 1

A **proof system** for deduction

1. prohibits the use of meaning in drawing conclusions.
2. has a number of axioms (or axiom schemas) and a small number of (finitary) inference rules.
3. Each proof is a finite tree where each node of the tree is either an assumption or an axiom or is obtained by pattern-matching and substitution from the axioms and inference rules.
4. Each proof can be "checked" manually or verified by machine implementable algorithms.

NPTEL

Home Page ** * ** ** On Back First Screen Close 100 of 100 Clear

So, proof so will talk about Proof Systems. So, all mathematical domains essentially have a proof system. So, a Proof System for deduction prohibits the use of meaning in drawing conclusions firstly. Because, we are talking about it as a language of communication and that is only that is the only language we have. Even the model is described in this language or communication even if you have a model in our mind it is only describe in the language. And so it is a sentence of that language which actually are the preminent objects are in some sense the only concrete objects. There is a similar analogy with programming you know it is possible to say that algorithms Tukey's are abstract objects they do not actually exists. The only concrete objects that exist are programs. So, if you look at in the case of mathematical reasoning also we essentially going through the similar process. The actual models that we are trying to describe are abstract objects so only concrete things that we have now are really the sentences we use to describe those abstract objects. So, the analogy between and proof is really like a program.

So, but that analogy we may or may not have the time to get into but anyway. Let us look at it so a proof system essentially relies on a certain language. And it prohibits the use of meaning in drawing conclusions it has a number of axioms. So, our description of a mathematical model inspired by Euclid is essentially use have a collection of starting axioms. And or axiom schemas and a small number of finitely expressible inference rules. So, these inference rules will allow us to reason about this abstract mathematical domain. These axioms and these inference rules will

actually allow us to reason about reason and prove properties about these objects in whatever mathematical abstract mathematical domain we might be interested in describing. There is another way the is of course a way of looking at it that is that. This whole theory itself is like an abstract mathematical domain capable of being described in a meta language but, that is a different matter let us not get into at a moment. So, we want proofs to be non-circular so in fact it would be great if proofs are all trees so, at least directed as a click graphs. So, dependencies are expressed by a proof but expressed in such of in a cycle free manner.

So, that is one thing and the other thing the other important thing is even if this is not we want. Since, you use a logical language to communicate ideas about objects belonging to some possibly abstract mathematical domain. Which may or may not have a concrete existence then, what you are saying is at least. Whoever, you are communicating the properties to and the proofs too should be able to manually verify then, there is something consistent in your theory. That your theory is somewhat consistent that your proofs are actually valid proofs. They actually provide valid logical consequences think of it this way. You do not have any concrete representation of that of those abstract mathematical logics.

And, neither the person who you are communicating with neither that person has any concrete representation. The only thing that you have available is this language of communication. And therefore he should also be convinced then, the properties you express of this domain are consistent. And all the properties that you derive of this domain have also been done have been derived in some consistent non-circular fashion. So, this will lead us to something known as proof theory but at the moment let us look at, formal theories.

(Refer Slide Time: 25:12)

Requirements of Proof Systems

Syntactic. Proof systems are purely syntactic and no use is made of semantics in any proof.

Finitary. All axioms, axiom-schemas and rules of inference must be expressible in a finitary manner.

Decidability. The correctness of any application of a rule of inference must be machine-verifiable.

Soundness. The system must allow the deduction of only valid conclusions from the assumptions.

Completeness. The system must allow all valid truths to be deduced.

The semantics may be used to prove only the soundness and completeness of the proof system.

Navigation bar: Home, First, Previous, Next, Last, On Slide, Full Screen, Close, 12 of 100, Clear

So, there is another thing about this the in fact you know if this Tukey's this last point here says that each proof can be checked manually or verified by machine implementable algorithms. So, this is a requirement which comes actually from the original notion of computation. As, essentially machine computation as essentially simulating behavior of a human computer. Of a human computer who does not have intelligence but has all a finite collection of rules at his disposal. So, you can think of this human computer as a child of let us say primary school or some such thing and doing computation on pieces of papers. So, in fact the original model of computation during machine was actually inspired essentially by this idea of a human computer who does not exercise intelligence.

So, Turing had a very clear idea of excluding intelligence from it. And, so what he saying so therefore proofs can be checked by algorithms. So, there should be even if my machines are not intelligent enough to come out with new theorems. And, new proofs at least my the proofs that I specify should be mechanically decidable. By this by a machine as to whether, the proofs are correct or not whether there is circularity or not whether it is consistent or not. So, the proofs should be checkable definitely manually but at even by machine that's a stronger constrain here. Many logic books do not emphasize that but since, we are doing logic for computer science I think it is correctly emphasize that. So, the requirements of a proof system that we are looking at firstly is that so we are talking about, just a language.

So, it means completely syntactic there is no notion there is no place of semantics here everything has to be syntactically represented and syntactically proven. And because, of the this let us look at the third point decidability. The correctness of any application which involves a proof if it has to manually checked then, you have to be able to check that your deduction process is correct. Each step of your deduction process is correct and that should be machine checkable if that has to be machine checkable by an algorithm. And, that algorithm always has to terminate. Then what it means is that, essentially even if you use an infinite collection of axioms even if you use an infinite collection of rules There has to be an algorithm which can actually.

So, that has to be a finitary expression of those axioms and those rules. In such a fashion that there is a that an algorithm can apply those rules. And, check that your deductions each step of your deduction is in fact correct. So, the finitariness actually comes from the decidability. So, all we are saying is we are not saying that you should exclude anything in it any infinite collections of objects. All we are saying is that you have to it should be possible to express collections of infinitary objects in some finitary fashion. Through the means of this language because, every sentence is the of this language that we are going to use only a finite string symbol or in more computer science terms it just a finite abstraction syntaxry. So, each sentence is a finite abstract syntaxry so, it should be finitely expressive. There the only time when you might want to actually use the semantics is to justify the soundness or the completeness of your system. Then, at that time there is no alternative except two actually be able to create models. And, show that you know you are your mathematical theory sort of consistent and maybe you will have a complete theory.


(Refer Slide Time: 30:22)

Proof Systems: Desiderata

There are two **conflicting** desirable properties of proof systems.

Minimality. Inspired by Euclid and the controversy over the parallel postulate. *Is there a minimal set of axioms and inference rules from which all truths and only truths may be deduced?*

Naturalness. *Is there a natural intuitive set of axioms and rules from which all truths and only truths may be deduced?*



NPTEL

Home Page | ** | * | ** | On Home | For Students | Contact | 108 of 182 | Close

So, That is when so this is how we are going to look so focus is really on proof systems. And the notion of deduction. So, now when you look at proof systems and they are desirable properties there are two kinds of desirable properties about proof systems. Which, are not necessarily in agreement with each other. One which is inspired by essentially Hilbert axiomatization Euclidean geometry. And, actually something that we encounter in all our books and when is Minimality that means I, have to be able to expound the entire theory with a minimal number of axioms and inference rules. And I mean this comes from the controversy over the parallel postulate. Which, raised over several 100 years the question is, Whether the parallel postulate can be derived from the other postulates of Euclid? So, in that sense was Euclid's axiomatization of Euclidean geometry minimal or not. So, that was the major problem that, David Hilbert try to address. So, he like minimal systems but actually a lot of us we use minimal systems even in computer science.

Because, we use for example as I said this is a specification of groups by 3 axioms is like the minimal set of axioms four groups. So, all are presentations all same terms of minimal systems so, minimality as it uses. The other thing is Naturalness I, mean is there some is there any intuitive way of doing things you know a much more natural way of doing things. And so we look at, proof systems of both kinds. So, the first thing to do is to look at minimal proof system for propositional logic.

(Refer Slide Time: 32:16)

Formal Theories

Definition 12.1 A formal theory consists of

- Formal Language** a formal language \mathcal{L} .
- Axioms** a subset of the language \mathcal{L}
- Inference Rules** a set of inference rules

NPTEL

188 of 188

So, will define a formal theory a Formal Theory consist of therefore I, mean. So now, we are only interested in theories the actual what there meant to capture the actual models of this theories we are not interested. We are, just interested in the formal consistency in the construction of theories. So, a formal theory consist of a formal language this is, going to be the language of description about, possibly abstract object in an abstract mathematical domain a collection of axioms.

(Refer Slide Time: 32:58)

Formal Language

1. An alphabet $\Sigma = X \cup \Omega \cup \{(\,,\,)\}$ consisting of a set X of *variables* a set Ω of *connectives* each with a pre-defined arity and grouping symbols.
2. \mathcal{L} is defined inductively on Σ .
3. The **well-formed formulas** or **wffs** of \mathcal{L} are defined inductively on the alphabet.
4. Membership of strings (from the alphabet) in \mathcal{L} is **decidable** i.e. there exists an algorithm to decide whether a given string is a well-formed formula

NPTEL

Formal Theories

178 of 188

So, this Formal Language of course has an alphabet and grouping symbols and so on so forth. And there is a collection X of variables a set of operators essentially we are going to take an algebraic way of most things. And, grouping symbols like brackets and so on if you like. And, what at every point we want things to be machine checkable that is important thing. If not machine generatable at least machine checkable machine verifiable. So, which means the question of so this language has to be inductively defined I mean you can not just create an uncountable set of randomly created sentences. If it has to be machine checkable it has to be inductively defined. So, which means the typically what we want a finite abstract syntax trees for each sentence in the language.

And, inductively defined essentially means that we should be able to apply principles like the principle structural induction on these abstract syntaxtries in order to come out with properties of the sentences of the language. And the question of whether a certain string so the this is of course a this is a typical non-computer scientist. We are looking at, the question of you have this grouping symbols and so on the question of, Whether certain string belongs to the language is should be decidable? There should be an algorithm to decide it. But, essentially what we are saying is if, you say that we have to be are language is essentially consist of and this sentence each sentence of our language is essentially an abstract syntaxtry induct. Which is inductively defined by this language I essentially we are talking about a term algebra. So, you can think of this that your formal language is essentially a, term algebra.

That is a easiest way so then the membership and so on those kind of questions are automatically decidable there are simple algorithm to decide. Basically, you have to do tri-traversal and check their airities are maintained that is it, there is nothing else to be done.

(Refer Slide Time: 35:23)

Axioms A **decidable** subset of \mathcal{L} .

Inference Rules A finite set of rules.

1. Each rule R of arity $m \geq 0$ is a **decidable relation** $R \subseteq \mathcal{L}^m \times \mathcal{L}$ i.e. there exists an algorithm which for any $\phi_1, \dots, \phi_m, \psi$ can determine whether $((\phi_1, \dots, \phi_m), \psi) \in R$
2. For each $((\phi_1, \dots, \phi_m), \psi) \in R$, ϕ_1, \dots, ϕ_m are called the **premises** and ψ a **direct consequence** by virtue of R .
3. Each such rule is presented in the form
$$R. \frac{X_1 \cdots X_m}{Y}$$
 where the variables X_1, \dots, X_m, Y are the "shapes" of the formulae allowed by the rule.

4. If $m = 0$, R is called an **axiom schema**

NPTEL Formal Theories 11 of 26

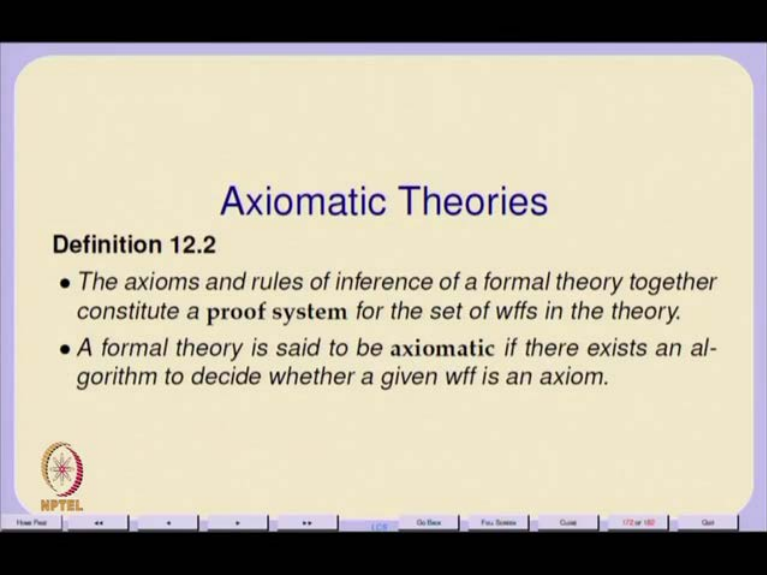
So, membership is decidable that that it is enough to know that membership is decidable axiom. So, some subset of this language are going to be taken as the axioms of our theory. So, this again has to be decidable subset. And, what do I mean by decidable subset? I mean that having chosen collection L prime of L as the axioms. There should be an algorithm which can tell you for any given sentence in the language whether it is an axiom or not. So, what it means is from this language L you cannot randomly pick sentences and call the axioms. So, there has to be some structure to those sentences. So, that an algorithm can work within finite time given an input sentence within finite time it can give you a yes or no answer is to the question is this sentence an axiom or not so, it has to be a decidable. So, there is a decidable subset of L which is their rules which are the axioms. And, then the inference rules there should be of I , mean again our decidability condition means that we cannot have an infinite collections of rules. A, rules have to be some of have to have the finitary property of expressability.

So, that there is an algorithm which decides whether I, have applied a rule correctly or not. So, each rule is just a decidable relations of membership in this relation is decidable by an algorithm. Which can clearly give a yes or no answer has this rule been applied to obtain this sentence ψ from these sentences ϕ_1 to ϕ_m . That, question has to have a definite answer by an algorithm it has to have a definite yes or no answer. So, there is so this R should be a decidable relation consisting of let us say R is an m ary relation for some m greater than or equal to 0.

So, m could be 0 also which means then, this relation R becomes a unary relation. Which, also means that it what is known as an axiom schema see the fourth point here. But it is an axiom schema but it still remains decidable everything remains decidable therefore, manually and machine verifiable. So, for each such ordered pair belonging to, this relation R ϕ_1 to ϕ_m are called the premises and ψ is called a direct consequence of ϕ_1 to ϕ_m . And, usually because of these decidability and other conditions this r therefore could be an infinite relation I mean it could have a cardinality. Which is $L^m \times L$ if, L is an infinite language $L^m \times L$ is also an infinite language countably infinite. So, R could be an, infinite subset of $L^m \times L$ so which means that and if it has to be decidable. Then essentially R would usually be presented in the form of pattern. So, these does it look like purple color? This purple color is essentially express patterns. These are patterns very much like the patterns you see in a higher order functional programming language they are structural patterns about this, sentences. So, we are not talking about string matching we are talking about pattern matching.

So, if I so R is presented in the form of some m patterns X_1 to X_m . And, a pattern Y then it's a finitary presentation of a possibly infinite relation. So, the presentation of the rule in this form essentially. So, these X_1 to X_m these purple variables essentially defined their shapes of formulae what kinds of formula structures can you have. And, then you define having define the structures you can. Then, replace all the leaves which contain these purple variables by any formula in the language to get. But, as long as you replace the same variable by the same formula you have to get an element of this relation R . So, we will see that so r inference rules have of so one way of finitely of finitely presenting infinite relations is to use pattern matching.

(Refer Slide Time: 40:42)



The slide is titled "Axiomatic Theories" in a blue font. Below the title is "Definition 12.2" in bold. It contains two bullet points: "The axioms and rules of inference of a formal theory together constitute a **proof system** for the set of wffs in the theory." and "A formal theory is said to be **axiomatic** if there exists an algorithm to decide whether a given wff is an axiom." The slide also features the NPTEL logo in the bottom left corner and a navigation bar at the bottom with various icons and the text "172 of 183".

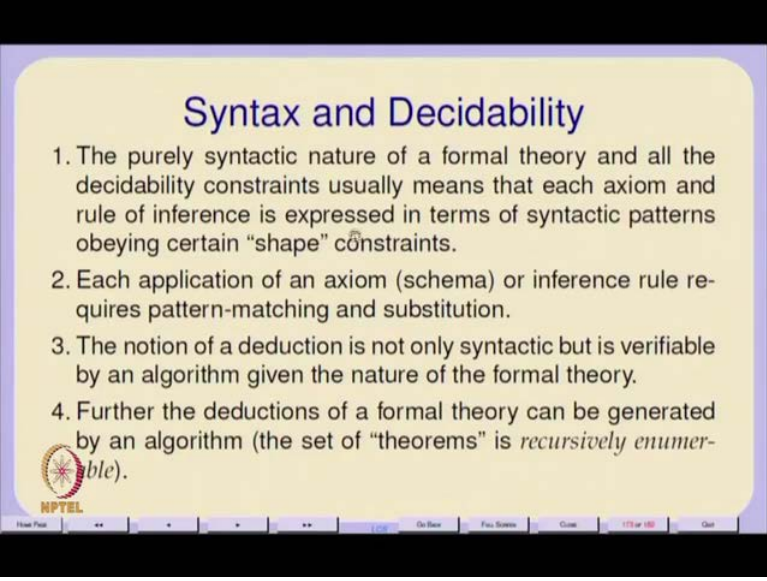
Axiomatic Theories

Definition 12.2

- The axioms and rules of inference of a formal theory together constitute a **proof system** for the set of wffs in the theory.
- A formal theory is said to be **axiomatic** if there exists an algorithm to decide whether a given wff is an axiom.

So, an Axiomatic theory so these are ingredients of a formal theory. And, essentially we will say that a formal theory is axiomatic. If, there exist an algorithm to decide whether a given well-formed formula is an axiom or not. So, that minimum of these concepts were defined before computer science was even thought of and so they expressed peculiarly. So, we made several distinctions but, I am not going to make those distinctions. So I am only interested in axiomatic theories everything that I am talking about should be decidable. And, we are also an enlightened age where pattern matching is something we are all familiar with. We, were done a huge amount of pattern matching in our programming. So, it is easy to understand these things in patterns.

(Refer Slide Time: 41:35)



Syntax and Decidability

1. The purely syntactic nature of a formal theory and all the decidability constraints usually means that each axiom and rule of inference is expressed in terms of syntactic patterns obeying certain "shape" constraints.
2. Each application of an axiom (schema) or inference rule requires pattern-matching and substitution.
3. The notion of a deduction is not only syntactic but is verifiable by an algorithm given the nature of the formal theory.
4. Further the deductions of a formal theory can be generated by an algorithm (the set of "theorems" is *recursively enumerable*).

NPTEL

Home Page | ** | * | ** | On Base | Full Screen | Close | 173 of 185 | Close

So, the Syntax and Decidability issues essentially dominate all this though it is not easy to make out from traditional logic book that this is what they mean. So, and of course a deduction has to be finite. Because any infinite deduction cannot be verified only finite prefixes of it can be verified. So, that so the other thing is that these deduction of a formal theory since everything is syntactic and pattern matching. These deduction of formal theory actually can be generated by a machine everything is syntactic. So, in fact these deductions or what might be called the set of formal theorems I , put theorems within quotes for specific reason. Because I will define the notion of a formal theorem which will look like different from what we normally understand is a theorem. This set of theorems is actually what is known as a recursively enumerable. And, what recursively enumerable means is just that there exist an algorithm. Which, not that it will always terminate take the they exist an algorithm but, which infinite time will generate the next element

So it is not really an algorithm in sense that it is not guarantee to terminate but it generates theorem. Let us talk about, generation of even sentences in our language of formally in a inductively defined language I can think of this language. Also, the sentences of this language also as being any enumerated by first enumerating all the leaf nodes in some one node abstract syntaxtries then maybe two or three node abstract syntaxtries and so on and so forth. But, if I have an infinite number of possible variables then of course it is never going to terminate. Even the generation of one node abstract syntax trees is never going to terminate. But, all we are

saying is that if there is an ordering of those on those variables if there is a total ordering on this variables then between any two variables is going to be finite amount of time.

Which, this algorithm will take to generate the next one. So, that is so for those of you have not done theory of computation the notion of recursive enumeration is that even if it is an infinite set it is possible to write a program. Which, is guaranteed to spend only a finite amount of time between the generation of subsequent elements of elements subsequent elements of the sequence. I mean take a simple program let us say for generation of all the primes. So, if you are going to generate all the primes. You, are not going to terminate the program you are going to put it in some infinite while loop or infinite recursion. But, if your algorithm is correct then the generation of the n plus 1th prime takes only a finite amount of time after you generated the n th prime. So, the primes therefore can be recursively enumerated in the sense that there is a program. Which, infinite time will generate the n 'th element of the sequence. And, will not take more than a finite time it is guaranteed and given the n 'th element it is guaranteed to produce the n plus 1th element infinite time. So, that is a notion of recursive enumerability loosely speaking.

And, essentially what we are saying is that these deductions. Therefore, the so in theory it is possible with just with just the notion of machine verifiability to have essentially a non-deterministic program. Which, uses all this methods to generate theorems of a formal theory. In such a fashion that it takes only a finite amount of time to generate. The next theorem see you can order the theorems some way you like 1step theorem, 2 step theorems and so on and so forth whatever. And, based on some ordering it is possible to generate all the theorems. And, that is so the theorems of a theory formal theory are recursively enumerable needless to say if your theory is consistent. Then, your theorems should be truths about that mathematical domain. This recursive enumerability only gives me and countably infinite collection of truths. The actual mathematical theory might have an uncountable number of truths. Take a mathematical theory relating to the real's which is already in an uncountable set. It is possible there are an uncountable number of truths which i cannot unravel in just countable amount of time. So, these has some issues with formal theories I mean this is these are some of the things that make mathematical still a viable profession I, mean we cannot leave it all to the machines.

(Refer Slide Time: 47:35)

A Hilbert-style Proof System

Definition 12.3 \mathcal{H}_0 , the Hilbert-style proof system for Propositional logic consists of

- The set \mathcal{L}_0 generated from A and $\{\neg, \rightarrow\}$
- The following three axiom schemas

S.
$$\frac{}{(X \rightarrow (Y \rightarrow X)) \rightarrow ((X \rightarrow Y) \rightarrow (X \rightarrow Z))}$$

K.
$$\frac{}{X \rightarrow (Y \rightarrow X)}$$

N.
$$\frac{}{(\neg Y \rightarrow \neg X) \rightarrow ((\neg Y \rightarrow X) \rightarrow Y)}$$

- A single rule of inference *modus ponens*

MP.
$$\frac{X \rightarrow Y, X}{Y}$$

NPTEL

So, the first proof system was actually given by Hilbert and Acroman. And this is a presentation of this Hilbert proof system for propositional logic it is. So, first thing of course is it is minimal in several ways. One is the language itself is minimal it has only these two operators. But, we know that this two operators are functionally complete they form a functionally complete set they are adequate for all of propositional logic. So, all other operators can be defined in terms of these two some kinds of abbreviations. So, given that your actual language of description consist a only these two operators and the set of atoms the infinite set a of atoms. Then you have these three axiom schemas do not ask me, How Hilbert thought of it? To this day I have no clue how he came up with this. For there is an interesting way interesting thing to these axioms does anybody notice them. So Hilbert came up with these axioms and 100 years after he came up with these axioms there is something re-interesting about them. And, can any of you tell me what is interesting about them.

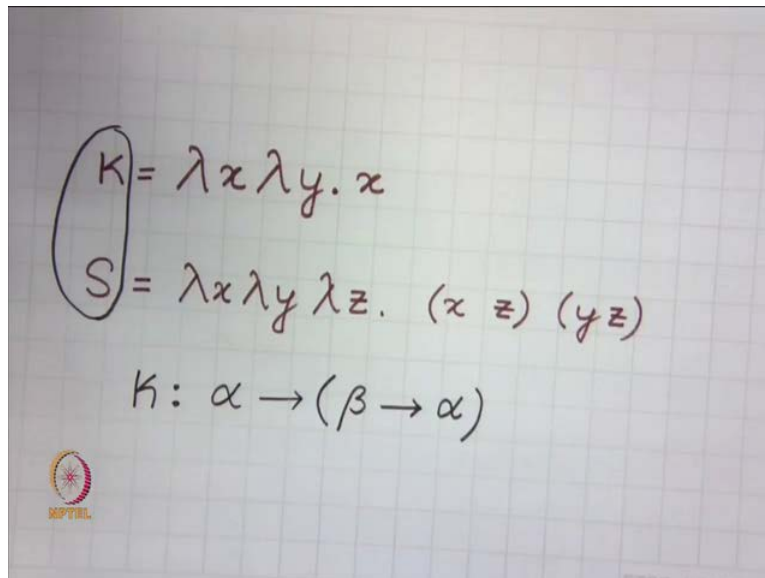
Student: Still do not know how he got?

We, I still do not know how he got them, I mean Hilbert died without he was too busy publishing too many papers in too many different areas and mathematics to tell us how he came up with them. But, this is actually it may not just be Hilbert it is called the Hilbert and Acroman proof system. But, this side but this, the whole style of coming up with a minimal set of axioms is

called a, Hilbert style proof system. Do, you see any interesting in this from the standard point of 20 first century about, these axioms? Does anybody know, Why I have called this axiom S?

This any can anybody guess, Why I have called this axiom k? Have you done the lambda calculus? Have you done the typed lambda calculus?

(Refer Slide Time: 50:08)



The image shows handwritten mathematical definitions for combinators K and S on a grid background. The definitions are:

$$K = \lambda x \lambda y. x$$
$$S = \lambda x \lambda y \lambda z. (x z) (y z)$$
$$K: \alpha \rightarrow (\beta \rightarrow \alpha)$$

In the bottom left corner of the grid, there is a small circular logo with a star and the text "NPTEL" below it.

Let us take this lambda term, what is this lambda term? You forgotten your programming language also, true or false it is actually true. But, that is only one particular way but there is something more fundamental about this term besides being true is called the combinatory K. What is the connection between this combinator K and this axiom K you done some. So, this is the combinator K and there is also a combinator S. And in fact, What is a connection between this S and this and this axiom S? No idea, Have you ever try to find? The take the lambda calculus of simple types or even polymorphic types it does not matter. What, is the type of K? It, takes x which maybe of type alpha. And, it produces a function lambda yx which is essentially a function of this type. Now, What is a connection between this K and this K? The types of this I mean. So, there is an isomorphism is actually on the types structure which is like propositional. And, only the thing about the lambda calculus by the way if, you must also learned that this K and S combinators are sufficient to describe all computable functions. So, every computable

function can be expressed in terms of these combinators K and S. So, an amazing result the interesting is that the types are actually like propositions and the K.

So, this K and S essentially can describe all positive so, this n is separate let us not talk about that. So, the K and S here essentially can give you all propositional tautologies. Which, do not have negation occurring in that it, is an amazing isomorphism between these two sets. But, even though amazing thing is this rule of inference modus ponens. What, is amazing about this?

(Refer Slide Time: 54:29)

$$K = \lambda x \lambda y. x$$

$$S = \lambda x \lambda y \lambda z. (x z) (y z)$$

$$K: \alpha \rightarrow (\beta \rightarrow \alpha)$$

$$\begin{array}{l} t: \alpha \rightarrow \beta \\ u: \alpha \\ \hline v: \beta \end{array}$$

You take any lambda term you take a lambda term t , which is of type $\alpha \rightarrow \beta$. And, you apply it to some term u which is of type α and what you get is an element of type β . So, you get an u element a new lambda term v which is of type β . And, this modus ponens exactly captures beta reduction. So, there is an amazing I mean it is these things were done independently though a, the lambda calculus itself. And, combinatory logic came up of few years after Hilbert's axiomatization. But, the connection is absolutely stunning and the fact that you can all types are just propositions is an, quite interesting thing. And, there is an isomorphism between this and modus ponens.

Which, is standard rule of inference if I, have proved that a something imply something else and if I am given antecedent then, I conclude the consequent. And, that is also beta reduction the type of beta reduction. So, this is all that you require and there is the, I will not go into this and it

connection with lambda calculus that's a little complicated. But, let us live with this so that is why I called these axioms S and K not many books not many logic books called them S and K. They call them α_1 α_2 I, mean is the ridiculous thing to do. But, and this modus ponens is actually beta reduction. So, there is an interesting connection between therefore programming languages and logic that is it.