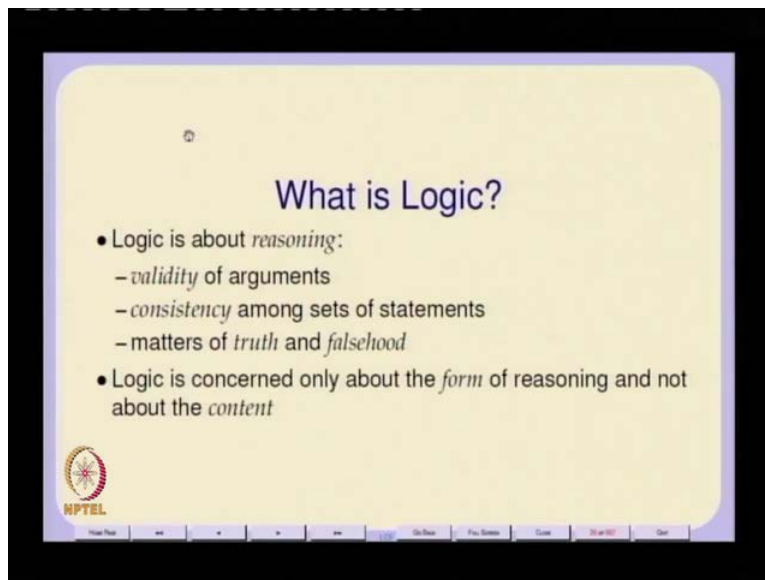


Logic for CS
Prof. Dr. S. Arun Kumar
Department of Computer Science
Indian Institute of Technology, Delhi

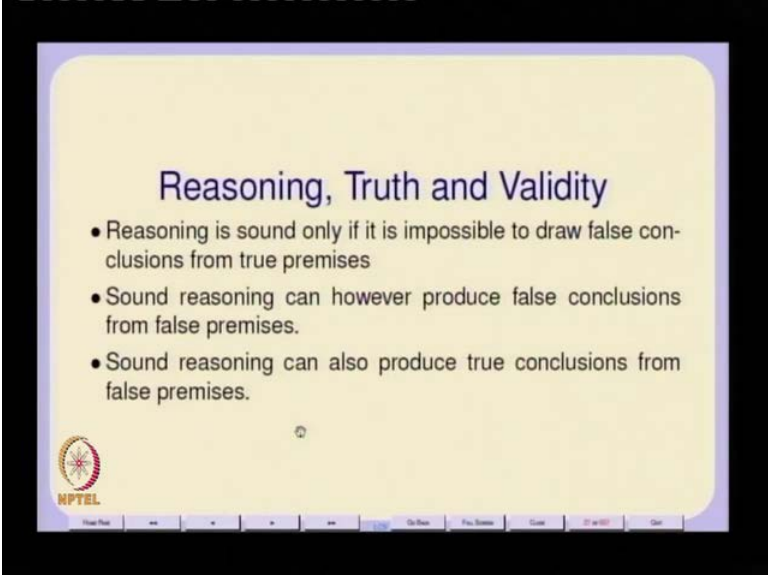
Lecture - 39
References

(Refer Slide Time: 00:43)



So, welcome to the last lecture, on logic for computer science. So, we have let us recapitulate, what all we have done? Well there was introduction in which we actually looked at the notions of what is logic the notion of validity of arguments we made it clear that logic is concerned only about the form of reasoning.

(Refer Slide Time: 00:56)



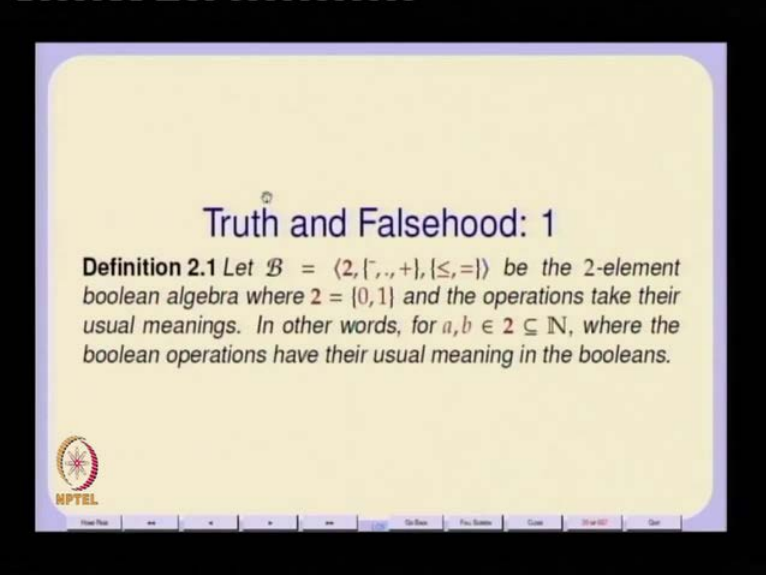
Reasoning, Truth and Validity

- Reasoning is sound only if it is impossible to draw false conclusions from true premises
- Sound reasoning can however produce false conclusions from false premises.
- Sound reasoning can also produce true conclusions from false premises.

NPTEL

And, not really about the content and then we also spoke about Reasoning Truth and Validity. And, we showed how in we have gone through many examples where there were fallacious proofs and we have shown how you can sometimes derive false conclusions from true premises. And, on the other hand we can always even with Sound reasoning principles we can produce false conclusions from false premises. So, that is so in that sense logic really has to do with only finding logical consequence of logical consequences of premises that we as our assumptions and or our hypothesis. So, we speak and it is not really concerned with the actual truth of statements that one make about a any structure or about the universe or world in general. So, and in particular we focused on first-order logic by starting from propositional logic the proposition connectives propositional logic or also sentential logic.

(Refer Slide Time: 02:08)

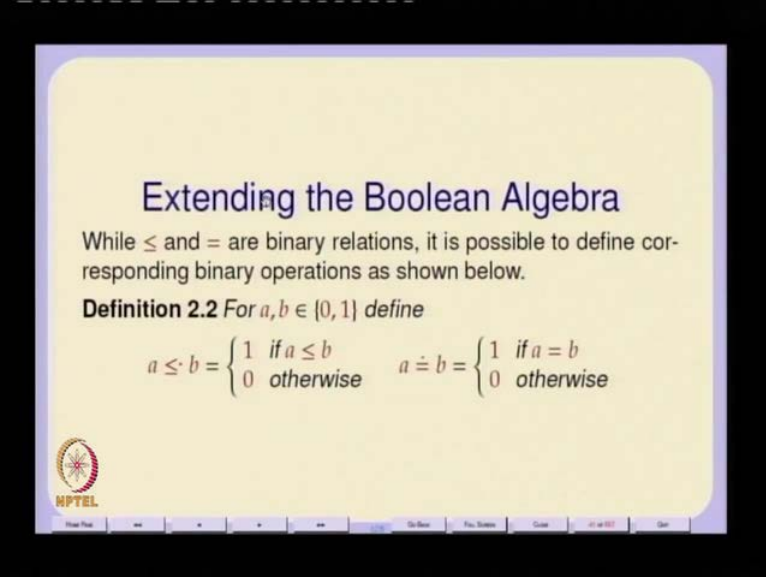


Truth and Falsehood: 1

Definition 2.1 Let $\mathcal{B} = \langle 2, \{\cdot, \cdot, +\}, \{\leq, =\} \rangle$ be the 2-element boolean algebra where $2 = \{0, 1\}$ and the operations take their usual meanings. In other words, for $a, b \in 2 \subseteq \mathbb{N}$, where the boolean operations have their usual meaning in the booleans.

NPTEL

(Refer Slide Time: 02:12)



Extending the Boolean Algebra

While \leq and $=$ are binary relations, it is possible to define corresponding binary operations as shown below.

Definition 2.2 For $a, b \in \{0, 1\}$ define

$$a \cdot b = \begin{cases} 1 & \text{if } a \leq b \\ 0 & \text{otherwise} \end{cases} \quad a \doteq b = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

NPTEL

We define the notion of Truth and falsehood we so in that sense it is a mathematical it is a mathematical approached logic. Where, we essentially studied a, logic also as a branch of mathematics rather than really as what was originally intended namely that logic should have been the foundation of mathematics.

(Refer Slide Time: 02:40)

Table of Truth & Falsehood

a	\bar{a}	$a \cdot b$	$a + b$	$a \oplus b$	$a \equiv b$
0	1	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	0	1	1	1	1

The slide also includes a small table for NOT:

a	\bar{a}
0	1
1	0

The slide features the NPTEL logo in the bottom left corner and a navigation bar at the bottom.

So, instead we just treated it as a branch of mathematics and then we tried it with Boolean algebra. And, we went through all these the propositional logic syntax the usual propositional connectives. And, then we also showed that there was some I mean so, this is this is also an algebraic treatment of logic derived from Boolean algebra. Essentially, propositional logic and we spoke about identities normal forms.

(Refer Slide Time: 03:17)

Validity & Falsification

The validity of such arguments involves showing that the conclusion is a **logical consequence** of the hypotheses that precede it. The following alternatives exist:

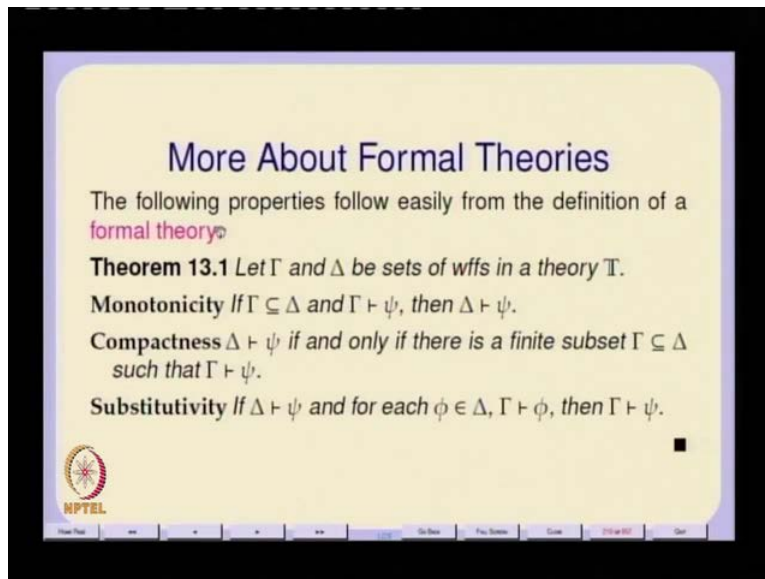
1. Using a truth table.
2. Using theorem 4.2
3. Using one of the parts of theorem 4.3.

If the argument is not valid, then a falsifying assignment needs to be also given.

The slide features the NPTEL logo in the bottom left corner and a navigation bar at the bottom.

Then, we also gave an algorithm for checking tautologies. And, we used this tautology checking also to Validate or Falsify arguments. And, we showed that there were several techniques could use truth tables you could use various other logical consequence mechanisms we could also in. But, we could also use proof theoretic mechanisms and which is what we did in our Hilbert style proof system.

(Refer Slide Time: 03:45)



More About Formal Theories

The following properties follow easily from the definition of a **formal theory**.

Theorem 13.1 Let Γ and Δ be sets of wffs in a theory \mathbb{T} .

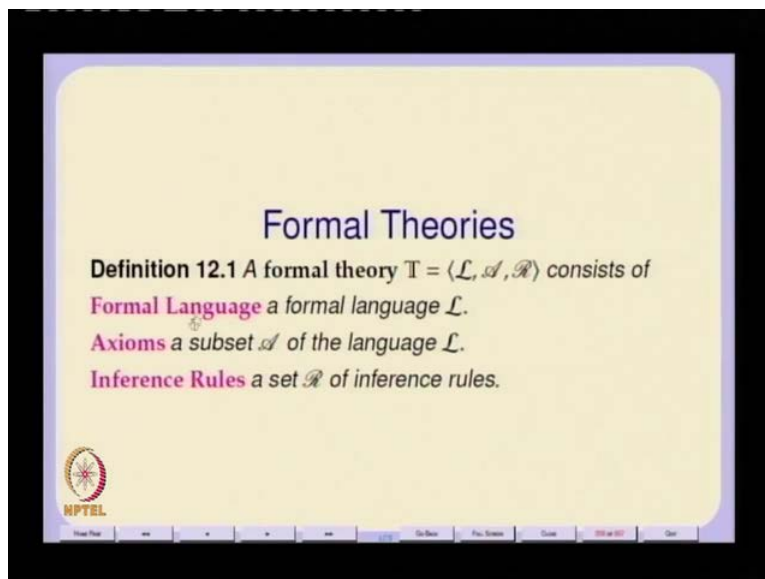
Monotonicity If $\Gamma \subseteq \Delta$ and $\Gamma \vdash \psi$, then $\Delta \vdash \psi$.

Compactness $\Delta \vdash \psi$ if and only if there is a finite subset $\Gamma \subseteq \Delta$ such that $\Gamma \vdash \psi$.

Substitutivity If $\Delta \vdash \psi$ and for each $\phi \in \Delta$, $\Gamma \vdash \phi$, then $\Gamma \vdash \psi$.

NPTEL

(Refer Slide Time: 03:54)



Formal Theories

Definition 12.1 A formal theory $\mathbb{T} = \langle \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ consists of

Formal Language a formal language \mathcal{L} .

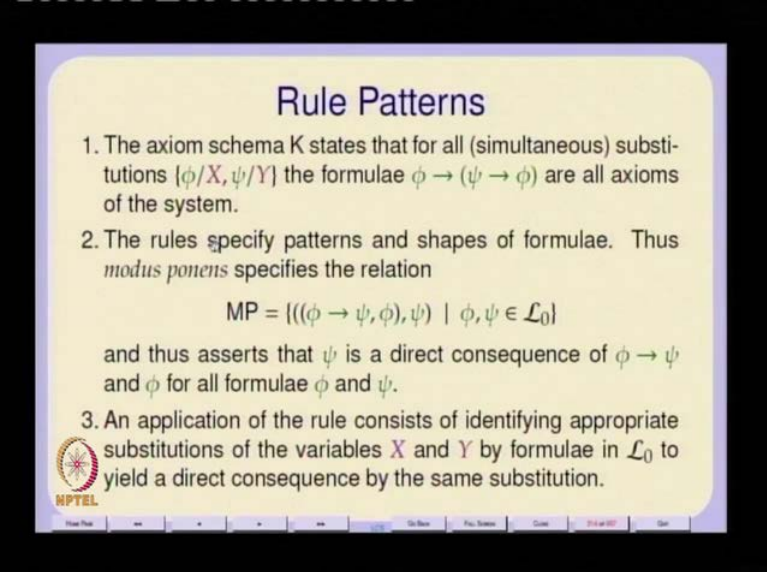
Axioms a subset \mathcal{A} of the language \mathcal{L} .

Inference Rules a set \mathcal{R} of inference rules.

NPTEL

And, from the Hilbert style proof system we came up with the notion of formal theories. Especially the Monotonicity property the Compactness property Substitutivities soundness of a Formal Theory these are all axioms of the subset of the language we gave a set of inference rules of these various forms we spoke about axiomatic theories and decidability.

(Refer Slide Time: 04:09)



Rule Patterns

1. The axiom schema K states that for all (simultaneous) substitutions $\{\phi/X, \psi/Y\}$ the formulae $\phi \rightarrow (\psi \rightarrow \phi)$ are all axioms of the system.
2. The rules specify patterns and shapes of formulae. Thus *modus ponens* specifies the relation
$$\text{MP} = \{((\phi \rightarrow \psi), \phi), \psi) \mid \phi, \psi \in \mathcal{L}_0\}$$
and thus asserts that ψ is a direct consequence of $\phi \rightarrow \psi$ and ϕ for all formulae ϕ and ψ .
3. An application of the rule consists of identifying appropriate substitutions of the variables X and Y by formulae in \mathcal{L}_0 to yield a direct consequence by the same substitution.

NPTEL


And, we actually went through this Rule Patterns and which are which as you can see a completely syntactic. And, therefore they are not really concerned about underlying content of the sentences about which we are reasoning. Then, we also showed so it is an important properties about formal theories and about the Hilbert style proof system. We, showed for convenience we could derive a large number of new rules we could use them.

(Refer Slide Time: 04:38)

Derived Rules

- By **substitutivity** (theorem 13.1) we may simplify our proofs by incorporating theorems and meta-theorems as *derived rules* of our proof system.
- These **rules** may be presented in sequent form.
- The proof of the reflexivity may be rendered in sequent form by simply pre-pending each node in the tree with "⊢".
- The Deduction Theorem and its converse may be rendered in sequent form as a derived rule.
- **Reflexivity** may be expressed in sequent form as a derived rule.

These derived rules may be directly invoked in later proofs.



We, could essentially plug in the proofs and therefore the substitutivity of rules and we showed convenient way of introducing assumptions and discharging assumptions through the deduction theorem which, was and its and its converse so we showed that we could move between conclusions between proof conclusions and assumptions that, we make.

(Refer Slide Time: 05:00)

The Sequent Form

Let Γ be a sequence of formulae.

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

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


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

MP. $\frac{\Gamma \vdash X \quad \Gamma \vdash X \rightarrow Y}{\Gamma \vdash Y}$

\rightarrow . $\frac{}{\Gamma \vdash X \rightarrow X}$

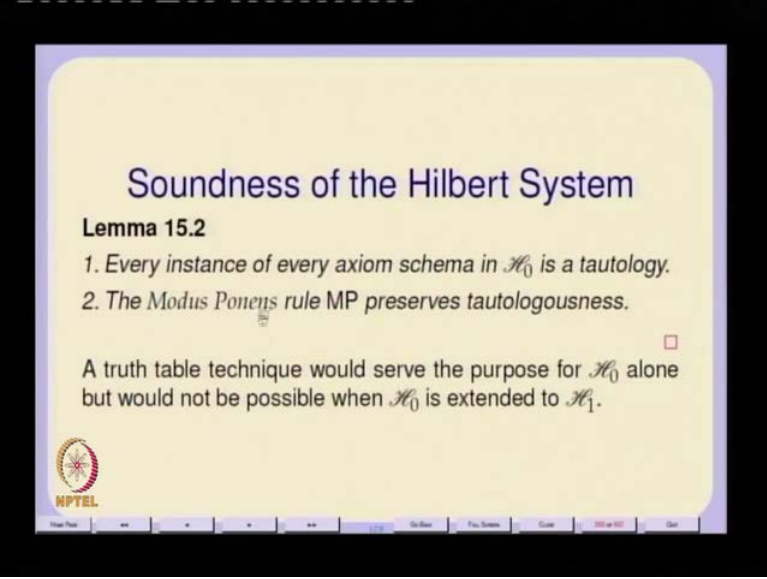
DT \Leftarrow . $\frac{\Gamma \vdash X \rightarrow Y}{\Gamma, X \vdash Y}$

DT \Rightarrow . $\frac{\Gamma, X \vdash Y}{\Gamma \vdash X \rightarrow Y}$



And, which is in fact the deduction theorem actually tells us something about the normal reasoning practices that, we normally have that we normally employ when dealing with proofs in mathematics.

(Refer Slide Time: 05:39)



Soundness of the Hilbert System

Lemma 15.2

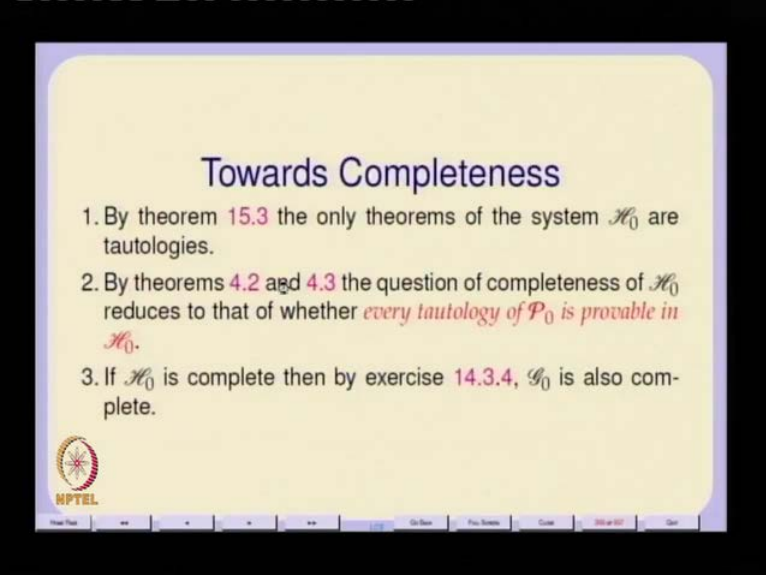
1. Every instance of every axiom schema in \mathcal{H}_0 is a tautology.
2. The Modus Ponens rule MP preserves tautologousness. \square

A truth table technique would serve the purpose for \mathcal{H}_0 alone but would not be possible when \mathcal{H}_0 is extended to \mathcal{H}_1 .

NPTEL

And, moving from the Hilbert style proof system we have we have also showed it soundness and completeness. We, moved on to those Soundness of the Hilbert style System we showed that every instance of every axiom in the Hilbert style in the system \mathcal{H} naught which, is the Hilbert style proof system for propositional logic is a tautology. And, the modus ponens rule preserves tautologousness in particular it is also preserves tautologousness. And, therefore the notion of deriving conclusions of fresh facts from a set of assumptions which are held to be true uses a modus ponens rule.

(Refer Slide Time: 06:25)



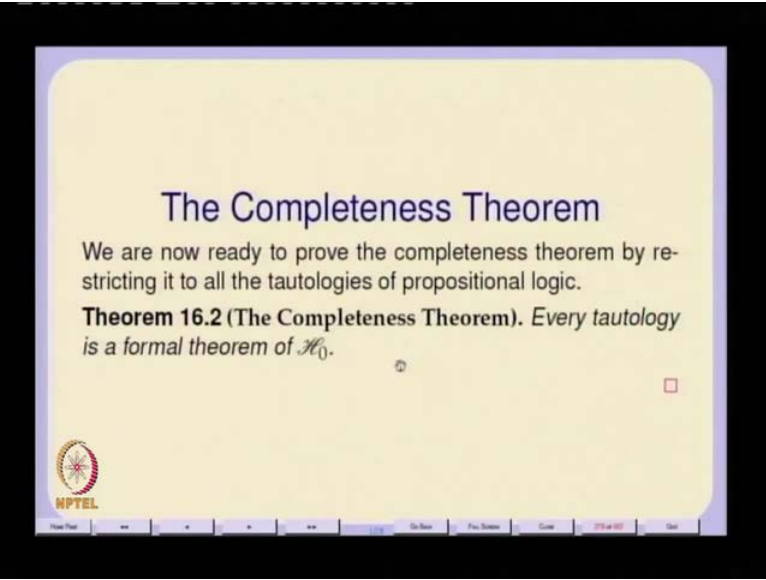
Towards Completeness

1. By theorem 15.3 the only theorems of the system \mathcal{H}_0 are tautologies.
2. By theorems 4.2 and 4.3 the question of completeness of \mathcal{H}_0 reduces to that of whether *every tautology of \mathcal{P}_0 is provable in \mathcal{H}_0 .*
3. If \mathcal{H}_0 is complete then by exercise 14.3.4, \mathcal{G}_0 is also complete.

NPTEL

And, we showed that this system is also complete in the sense that every tautology in propositional logic.

(Refer Slide Time: 06:29)



The Completeness Theorem

We are now ready to prove the completeness theorem by restricting it to all the tautologies of propositional logic.

Theorem 16.2 (The Completeness Theorem). *Every tautology is a formal theorem of \mathcal{H}_0 .* ◻

NPTEL

The Completeness I am just say is that every tautology can be derived as a formal theorem of the Hilbert style proof system.

(Refer Slide Time: 06:47)

The Truth-table Lemma

Lemma 16.1 Let ϕ be a formula with $\text{atoms}(\phi) \subseteq \{p_1, \dots, p_k\}$. For each truth assignment τ ,

$$p_1^*, \dots, p_k^* \vdash \phi^*$$

where for each i , $1 \leq i \leq k$,

$$p_i^* \equiv \begin{cases} p_i & \text{if } \tau(p_i) = 1 \\ \neg p_i & \text{otherwise} \end{cases}$$

and

$$\phi^* \equiv \begin{cases} \phi & \text{if } \mathcal{T}[\phi]_{\tau} = 1 \\ \neg \phi & \text{otherwise} \end{cases}$$

The slide includes an NPTEL logo in the bottom left corner and a small square symbol in the bottom right corner. A navigation bar is visible at the very bottom of the slide.

And, in particular what we did was its relationship to this to the semantics of propositional logic which, is our language was by this actually simulating the Truth Table and we what we showed is that by this lemma. Is, that every Truth Table can actually we have simulated as a logical proof in the Hilbert style system going on from the Hilbert style proof system.

(Refer Slide Time: 07:12)

The Analytic Tableau Method

1. Like resolution, the tableau method also checks for the unsatisfiability of a set of formulae.
2. Like resolution, each step of the method *preserves* satisfiability.
3. Unlike resolution
 - (a) There are no transformations of mammoth formulae into a normal form (esp. the use of distributivity which can increase sizes of formulae).
 - (b) It works with the list of formulae $[\phi_1, \dots, \phi_n, \neg\psi]$ directly by breaking up the formulae and building a tree called the tableau
 - (c) It relies on the symmetry between truth and falsehood at a semantic level to check for satisfiability or unsatisfiability.

The slide includes an NPTEL logo in the bottom left corner and a small square symbol in the bottom right corner. A navigation bar is visible at the very bottom of the slide.

We, have also looked at Analytic Tabular which gave a convenient analytic tabular and resolution method which, gave convenient methods of actually proof which can be automated and the tabular.

(Refer Slide Time: 07:23)

Tableaux Rules	
	$\neg\neg. \frac{\neg\neg\phi}{\phi}$
$\wedge. \frac{\phi \wedge \psi}{\phi}$ ψ	$\neg\wedge. \frac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi}$
$\vee. \frac{\phi \vee \psi}{\phi \mid \psi}$	$\neg\vee. \frac{\neg(\phi \vee \psi)}{\neg\phi}$ $\neg\psi$
$\rightarrow. \frac{\phi \rightarrow \psi}{\neg\phi \mid \psi}$	$\neg\rightarrow. \frac{\neg(\phi \rightarrow \psi)}{\phi}$ $\neg\psi$
$\leftrightarrow. \frac{\phi \leftrightarrow \psi}{\phi \wedge \psi \mid \neg\phi \wedge \neg\psi}$	$\neg\leftrightarrow. \frac{\neg(\phi \leftrightarrow \psi)}{\phi \wedge \neg\psi \mid \neg\phi \wedge \psi}$

And, Genson natural deduction system share this common feature that day actually breakup the formula into essentially sub formulae depending upon the kind of operators and the root. So, in that sense in propositional logic the tabulo rule are such that every tabulo would actually be finite. And, we showed that the tabulo was the tabulo system was also complete moreover by being structurally inductive by the rules being structurally inductive. We, also showed that it is possible to automate this for a propositional theorem prover. So, we could do tautology checking validity checking and so on. And, all those problems for propositional logic I actually decidable algorithmically decidable. So, we showed how the tabulo rules can be used to come up with tabulo proofs, we showed the consistency, we showed how unsatisfiability can be proven through closed tabulos we also had a the notion of Hintika sets.

(Refer Slide Time: 08:42)

Satisfiability of Infinite Sets

From corollaries 9.4 and 9.5 we have

Corollary 10.1 A finite set Γ is unsatisfiable iff there is a closed tableau rooted at Γ .

■

Corollary 10.2 If a finite set Γ is satisfiable then every nonempty subset of Γ is satisfiable too.

■

Question 1. Suppose Γ were a denumerable (countably infinite) set. Under what conditions is Γ satisfiable?

Question 2. Suppose every subset of a denumerable set Γ is satisfiable. Then is Γ necessarily satisfiable?

Question 3. Suppose that only all finite subsets of a denumerable set Γ are satisfiable. Then is Γ satisfiable?

NPTEL

And, we showed using Hintikka sets that tableau method is also complete. We also then had the compactness theorem which, showed that in order to prove unsatisfiability of infinite sets it is sufficient to find a finite subset which is unsatisfiable. So, these were this is a property of this is what is known as a property of finite character which, the compactness gives us and which unsatisfiability also satisfies. Then, moving on to first-order logic, we actually defined this we went through essentially the same processes. But, now it is more complicated because the underlying model of truth table which is finite which does not hold any more I mean. And, we are looking at essentially modeling structures expressed mostly algebraically. And, we are looking at using syntactical methods to prove facts about complicated algebraic structures given their and classes of structures which share a common signature. So, the notion of signature the notion of substitution and therefore an extension of the Hilbert style proof system for first-order logic. Before that, the semantics of the first-order logic therefore gets considerably the syntax and semantics of first-order logic become considerably more complex you can see that.

(Refer Slide Time: 10:19)

Predicate Logic: Symbols

- V: a countably infinite collection of **variable** symbols;
 $x, y, z, \dots \in V$.
- F: a countably infinite collection of **function** symbols;
 $f, g, h, \dots \in F$.
- A: a countably infinite collection of **atomic predicate** symbols;
 $p, q, r, \dots \in A$.
- Grouping symbols: $(,), [,]$.
- All of the above sets are pairwise disjoint.

NPTEL

(Refer Slide Time: 10:29)

Predicate Logic: Signatures

Definition 17.2 A **signature** (or more accurately **1-sorted signature**) Σ is a denumerable (finite or countably infinite) collection of strings of the form

$$f : s^m \rightarrow s, m \geq 0$$

or

$$p : s^n, n \geq 0$$

such that there is at most one string for each $f \in F$ and each $p \in A$. m and n are respectively the **arity** of f and p .

Here s is merely a symbol signifying a **sort** of elements. A generalization to many-sorted algebras would require the use of as many such symbols s_1, \dots, s_m as there are sorts.

NPTEL

We, had to define a separate set of symbols countably infinite collection of variable symbols function symbols. Then, notion of a, signature the term and actually the sort of each element signature which is given by a sort S by sort symbol S. And, which you can think of loosely speaking as, being related to the notion of types in programming languages. So, essentially a single sorted signature is or a 1-sorted signature is what we consider throughout. And, it refer to basically a simple algebraic system consisting of a single set. And, all elements essentially

belong to purportedly or putatively belong to this set. And, this set might be and since we are looking at classes of structures we did not specify this set explicitly. But, instead just uses syntactic symbol called S for a sort. So, then as we saw the semantics of this semantics of predicate logic therefore also gets considerably more complex.

(Refer Slide Time: 11:39)

Structures

Given a signature Σ , a Σ -structure or Σ -algebra \mathbf{A} consists of

- a non-empty set $A = |A|$ called the **domain** (or **carrier** or **universe**) of \mathbf{A} ,
- a function $f_{\mathbf{A}} : A^m \rightarrow A$ for each m -ary function symbol $f \in \Sigma$ (including symbols for each constant)
- a relation $p_{\mathbf{A}} \subseteq A^n$ for each n -ary atomic predicate symbol $p \in \Sigma$ and
- (for completeness) a truth value $p_{\mathbf{A}} \in 2 = \{0, 1\}$ for each (0-ary) atomic proposition $p \in \Sigma$.

When the Σ -algebra is understood or is the only structure under consideration, we omit the subscript \mathbf{A} from the functions and relations.

NPTEL

(Refer Slide Time: 11:47)

Notes on Structures

1. The domain has to be non-empty.
2. All functions are *total*.
3. One way to deal with *partial* functions (e.g. division on natural numbers) of arity $m > 1$ is to treat them as $(m+1)$ -ary relations and define predicate symbols to represent them.

NPTEL

Since, we have to deal with Structures which satisfies certain signature properties. And, essentially first-order logic deals with properties of structures actually it turns out that it deals with properties of classes of structure which share common signatures. So, in that sense first-order logic is a parameterized over the signature. And, we are essentially looking at the axiom of such structures.

(Refer Slide Time: 12:01)

Infix Convention

If in particular structures, functions or relations are normally written in infix form, then we use the infix form in the logical language too.

Example 18.1 If $\mathbf{N} = \langle \mathbb{N}; +; < \rangle$ the set of natural numbers under the binary operation of addition (+) and the binary relation less-than (<) is the structure, then we write predicate formulae using the corresponding symbols in the language in infix form. For example, the formula

$$\forall x[\exists y[x < x + y]]$$

with the operation in infix form is more easily understood in place of the more pedantic

$$\forall x[\exists y[< (x, +(x, y))]]$$

The slide includes an NPTEL logo and a navigation bar at the bottom.

(Refer Slide Time: 12:14)

Valuations and Interpretations

To be able to define the truth or falsehood of a predicate with free variables, it is necessary to be able to first define a valuation for the variables.

Definition 18.3 Given a Σ -structure \mathbf{A} , a **valuation** is a function $v_{\mathbf{A}} : V \rightarrow |\mathbf{A}|$ which assigns to each variable a unique value in $|\mathbf{A}|$.

Definition 18.4 Given a Σ -structure \mathbf{A} and a valuation $v_{\mathbf{A}} : V \rightarrow |\mathbf{A}|$, $(\mathbf{A}, v_{\mathbf{A}})$ is called a Σ -**interpretation**.

The subscript " \mathbf{A} " is often omitted when the context makes clear the structure that is being referred to.

The slide includes an NPTEL logo and a navigation bar at the bottom.

So, we had various examples we also showed so we had the notion of interpretation of the symbols of the first-order language over a signature sigma. We, needed then notion of valuations. And, there from there we actually defined the notion of truth of a formula with free variables. So, where the truth might depend the truth of the formula might depend on the values assigned to the free variables. And, so therefore in addition to the interpretation you also required the valuation.

(Refer Slide Time: 12:40)

Evaluating Terms

Definition 18.5 Given Σ -interpretation (A, v) the value of a term t in the interpretation is defined by induction on the structure of the term.

$$\mathcal{V}_A[x]_v \stackrel{df}{=} v(x), \quad x \in V$$

$$\mathcal{V}_A[f(t_1, \dots, t_m)]_v \stackrel{df}{=} f_A(\mathcal{V}[t_1]_v, \dots, \mathcal{V}[t_m]_v), \quad f: s^m \rightarrow s \in \Sigma$$

Notational conventions.

1. If $Var(t) \subseteq \{x_1, \dots, x_m\}$, we sometimes write $t(x_1, \dots, x_m)$ to denote this fact.
2. The subscript "A" is often omitted when the context makes clear the structure that is being referred to.

NPTEL

(Refer Slide Time: 12:47)

Coincidence Lemma for Terms

The following lemma may be easily proved by induction on the structure of terms.

Lemma 18.6 Given two Σ -interpretations (A, v) and (A, v') , and a term t , if $v(x) = v'(x)$ for each $x \in Var(t)$, then $\mathcal{V}[t]_v = \mathcal{V}[t]_{v'}$ ■

Notational convention.

If $t(x_1, \dots, x_m)$, $v(x_1) = a_1, \dots, v(x_m) = a_m$ for $a_1, \dots, a_m \in |A|$ in some Σ -interpretation (A, v) , then we write $t_A(a_1, \dots, a_m)$ instead of $\mathcal{V}[t(x_1, \dots, x_m)]_v$, since only the values of the variables occurring in t are needed to evaluate it.

NPTEL

So, we had a formal semantics for the Evaluation of Terms firstly and then we gave a coincidence lemma which essentially since variables are in infinite set. We, had to show that we had this coincidence lemma which, essentially said that under two different interpretations or under two different valuations if the variables mentioned in a term of the same. And, the valuations for all those variables that finite set of variables is the same in both valuations. Then, on the values of the terms would also be there values of the terms in the two valuations would be same. So, we had the notion of variance and then we gave this semantics of first-order formulae in particular that of the two quantifiers which, are which these quantifiers is actually parameterized on a variable the notion. Therefore, that we brought the notion of bound variables and then free and free variables and substitutions. We, also showed that you can have a coincidence lemma for formulae you can replace valuation variants by substitutions. And, then what we did was we use we used this notions of we use the same notions. We, extended the notions that we had for propositional logic to first-order logic we define the notions of models and satisfiability and validity based on structures and substructures. And, then we extended the Hilbert proof system to first-order logic for example.

(Refer Slide Time: 14:21)

Proof Rules: Hilbert-Style

Definition 22.1 $\mathcal{H}_1(\Sigma)$, the Hilbert-style proof system for Predicate logic consists of

- The set $\mathcal{L}_1(\Sigma)$ generated from A and $\{\neg, \rightarrow, \forall\}$
- The three logical axiom schemas **K**, **S** and **N**,
- The two axiom schemas

$$\forall E. \frac{}{\forall x[X] \rightarrow [t/x]X}, t \equiv x \text{ or } [t/x] \text{ admissible in } X$$

$$\forall D. \frac{}{\forall x[X \rightarrow Y] \rightarrow (X \rightarrow \forall x[Y])}, x \notin FV(X)$$

- The *modus ponens (MP)* rule and

$$\forall I. \frac{[y/x]X}{\forall x[X]}, y \equiv x \text{ or } y \notin FV(X)$$

NPTEL

So, we had these extra rules essentially dealing with quantifiers we also had an peculiar we also had derived rules in this case. But, you could for example we had these in addition to the derived

rules we had we also had the notion of equality for which we gave some axioms. But, more importantly we showed that the rules for a existential quantification for example.

(Refer Slide Time: 15:00)

Existential Generalisation

Existential quantification is the derived operator defined by

$$\exists x[\phi] \stackrel{df}{=} \neg \forall x[\neg \phi]$$

We then have the rules

$$\exists I. \frac{\Gamma \vdash \{t/x\}\phi}{\Gamma \vdash \exists x[\phi]}, \{t/x\} \text{ admissible in } \phi$$

$$\exists E. \frac{\Gamma \vdash \exists x[\phi]}{\Gamma \vdash \{a/x\}\phi}, a \notin FV(\Gamma) \cup FV(\exists x[\phi]) \text{ is fresh}$$

$\exists I$ is a **derived rule**

However $\exists E$ is not a derived rule in the Hilbert-System.

The Existential Elimination rule the corresponding derived rules are not is not the existential introduction is a derived rule however existential elimination is not a derived rule.

(Refer Slide Time: 15:17)

Equivalence of Proofs

Theorem 24.3 (Existential-Elimination Elimination Theorem). *If $\Gamma \vdash \exists E \phi$ is a correct proof then $\Gamma \vdash \phi$ provided no constants introduced in the proof $\Gamma \vdash \exists E \phi$ occur in ϕ . i.e. if ϕ is provable from Γ by use of the $\exists E$ rule then ϕ is provable from Γ without making use of the $\exists E$ rule.*

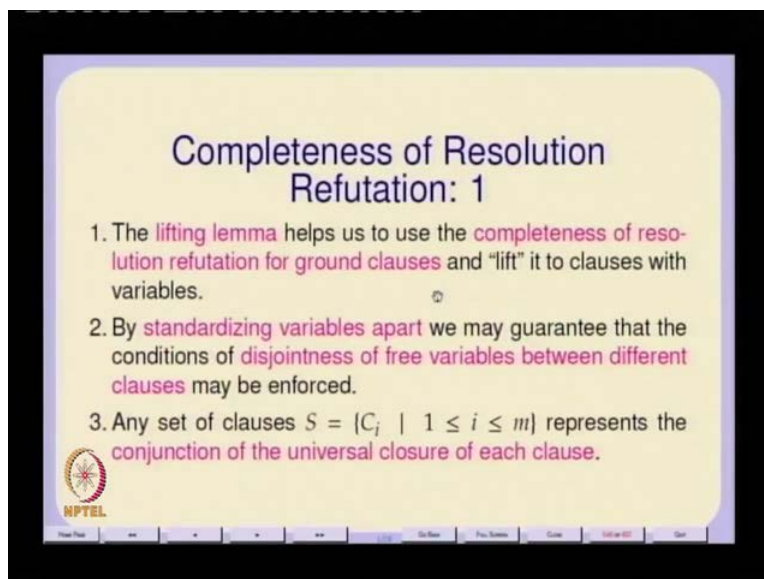
□

However this theorem is not applicable if ϕ does contain any of the constants introduced by the proof $\Gamma \vdash \exists E \phi$.

However, what we manage to show is that you can every proof by existential elimination which uses the existential elimination rule can also be transformed into a proof which, does not employ the existential elimination rule. In, this process we also required and reduction theorem we showed that there are restrictions on the application of the deduction theorem. So, essentially the deduction theorem applies mainly for sentences. So, the quantification over free variables and therefore generalizing from free variables or instantiating free variables is a certain issue. And, one has to be careful about generalizing on for example existential variables by universal quantifier. So, that further we general we have also the proof procedure the other proof procedures for propositional logic were extended to resolution for example and tabulos.

So, we did both we also showed that are Hilbert style proof system was sound and complete the tabulo method was sound and complete again we extended the notion of hintika sets first-order tabulo and hintika sets. And, we showed that is a both sound and complete and we proved the soundness and completeness of the Hilbert style proof system. We, also proved the soundness and completeness of resolution. What, we showed in particular was that resolution was refutation complete. So, which means it is not necessarily clear that you might be able to derive all logical consequences but what is necessary what is true definitely is that your you can always refuse the negation of a logical consequence. And, prove it so in that sense resolution was refutation complete.

(Refer Slide Time: 17:21)



The slide is titled "Completeness of Resolution Refutation: 1" and contains three numbered points. The text is as follows:

Completeness of Resolution Refutation: 1

1. The **lifting lemma** helps us to use the **completeness of resolution refutation for ground clauses** and "lift" it to clauses with variables.
2. By **standardizing variables apart** we may guarantee that the conditions of **disjointness of free variables between different clauses** may be enforced.
3. Any set of clauses $S = \{C_i \mid 1 \leq i \leq m\}$ represents the **conjunction of the universal closure of each clause**.

The slide also features the NPTEL logo in the bottom left corner and a navigation bar at the bottom.

So, this is what this is what we so therefore this gives us directly methods for automating the notion of theorem proving a automating the notion of proof. Even from first-order logic either through resolution or through tabulo a, the Hilbert style system does not necessarily permit itself to be automated. So, easily on the other hand resolution and tabulo is due and with that we have essentially. We, showed some these we also gave some applications of these first-order theories first of course is that logic programming is essentially something that is based on resolution. A, particular implementation of logic programming is prolog. But a and we looked at some prolog code 2 there some prolog code that here for example. And, we showed and there are however there are certain pragmatic reason by prolog is really different from logic programming. First, of all prolog does not it treats negation as a failure of proof rather than negation is an operator.

(Refer Slide Time: 18:38)

```

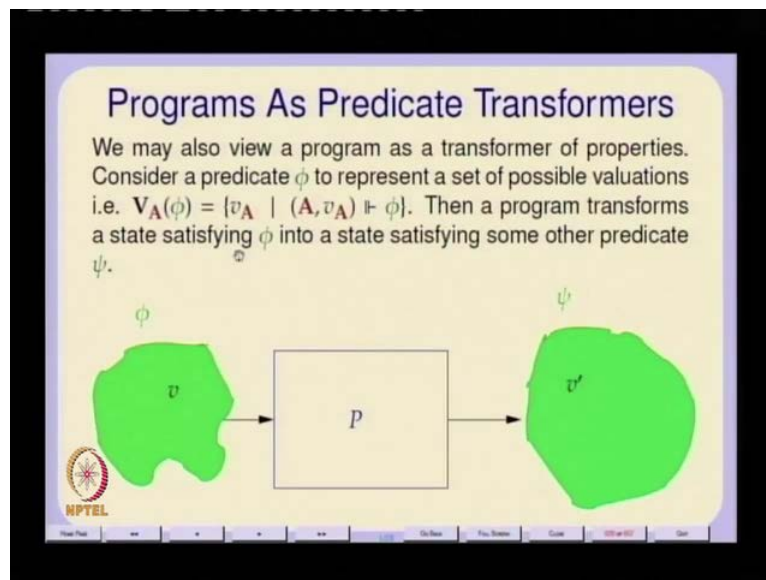
Prolog: Naturals

isnf(z).
isnf(s(X)) :- isnf(X).
rewrite(z,z).
rewrite(s(X), s(X)) :- isnf(X).
rewrite(s(X), s(Y)) :- isnf(Y), rewrite(X, Y).
rewrite(a(z, Y), Y) :- isnf(Y).
rewrite(a(Y, z), Y) :- isnf(Y).
rewrite(a(s(X), Y), s(Z)) :- rewrite(a(X,Y), Z).
rewrite(a(X, s(Y)), s(Z)) :- rewrite(a(X,Y), Z).
rewrite(a(X,Y), Z) :- rewrite(X,U), rewrite(Y,V), rewrite(a(U,V), Z).
even(z).
even(s(s(X))) :- even(X).
/*The next line is necessary so that it doesn't go
into an infinite loop*/
even(s(X)) :- not(X=z).
even(X) :- rewrite(X, Y), even(Y).
od(X) :- not even(X). % negation as failure
rewrite(a(a(s(z), s(s(z))), a(s(z), s(s(z))))), X).
NPTR :- s(s(s(s(s(s(z)))))) */

```

So, that is one thing and more over prolog the all prolog implementations which follow what is known as a war and abstract machine model actually tend to unsound because of certain problems substitutions and unification most general unifiers. Then, we looked at other applications for example. We, also looked at the verification of imperative programs the notion of a predicate as notion of a programming language or programming construct as a not just as a state transformer but as a, but as a predicate transformer.

(Refer Slide Time: 19:24)



And, that is what we looked at so this is like generalizing the notion of meaning from single states two sets of states and such. So, states refer essentially signify a property and therefore a predicate. And, so we actually extended this notion we came up with these notions of correctness assertions and total correctness assertions. And, we have given some examples all the verification we also at the at the end. So, we had these rules the notion of a looping variant the notion of a well-ordered set or a bound function which, basically traverses down a well-ordered a well-ordering in order to prove termination of these programs. Finally, we ended with an open problem on the Collatz problem.

(Refer Slide Time: 20:18)

An Open Problem: Collatz

Consider the following specification where Σ includes the *div* and *mod* functions on positive integers.

$$\mathbb{Z} \models [x > 0] \text{ Collatz } [x = 1]$$

where

$$\text{Collatz} \stackrel{df}{=} \{x > 1? [x \text{ mod } 2 = 0? x := x \text{ div } 2 : x := 3 * x + 1]\}$$

NPTEL logo and navigation controls are visible at the bottom of the slide.

So, what would like to do now after this rap is to look at why first-order logic why what about higher order logics. What, about second-order logic the other questions are what about we looked always at 1-sorted signatures can we look at many sorted signatures in what happens the first-order logic of many sorted logic signatures. For, example what happens to higher order logic these are some questions that a go beyond the scope of this course but will just touch on them.

(Refer Slide Time: 20:55)

Summary

1. A mathematical treatment of the essentials of reasoning
2. A rigorous treatment of First-order logic
3. Applications in logic and computer science
 - (a) some elementary theorem proving
 - (b) logic programming
 - (c) program verification
4. Some illustrations of the power of first-order logic
5. Some illustrations of the lack of distinguishability.

NPTEL logo and navigation controls are visible at the bottom of the slide.

So, here is a simple Summary of what we have done a mathematical treatment of the essentials of reasoning a rigorous treatment of first-order logic applications. Basically, to theorem proving logic programming and program verification. We, also showed some gave some illustration of the power of first-order logic. For, example we required just two axioms for equality to axiomatize all properties of equality. And, we showed basically that in many cases you can come up with first-order theories which, are sufficiently powerful. But, we also showed that there are first-order logic has certain severe constraints. For, example there is a lack of distinguishability one of the things we did.

(Refer Slide Time: 21:37)

A Non-standard Model of Arithmetic

Consider the model $N_S = \langle \mathbb{N}, \Sigma_S \rangle$ of the axioms of **number theory**.

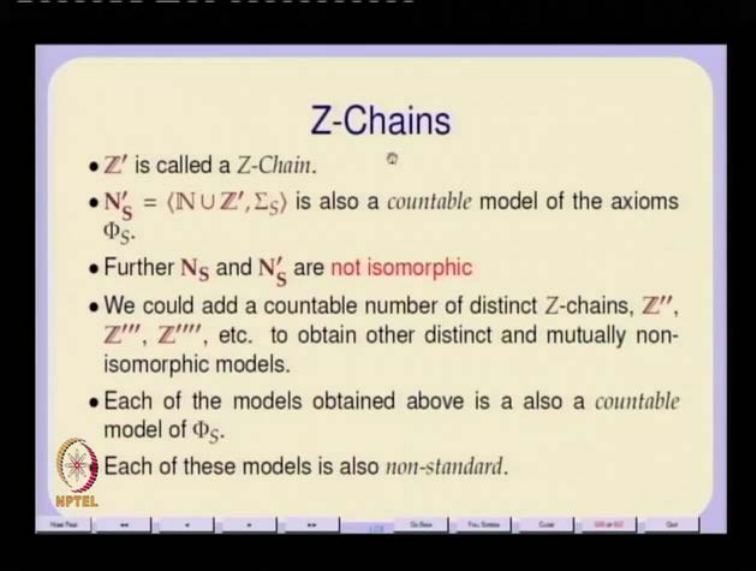
- We add a new element $0' \neq 0$.
- This implies adding an infinite number of new elements $0^{(+1)^n}$ one for each $n > 0$. For simplicity let us call these elements $1', 2', 3', \dots$. Each of these new elements is different from every element in \mathbb{N} .
- Since $0' \neq 0$, it must have a "predecessor" say $-1'$ which again leads to the addition of all the elements $-2', -3', -3', \dots$ each of which is distinct and different from all other elements. Let us call this set of elements Z' .

$S = \langle \mathbb{N} \cup Z', \Sigma_S \rangle$ is a model of Φ_S and is said to be **non-standard**.

So, the Non-standard Model of Arithmetic that we proposed allowed for the axioms of number theory to have models which, are not isomorphic to the standard model of arithmetic as we, standard model of the natural numbers which, successor as we are used to it and it the fact that it is that there are models which are not isomorphic to the standard model indicates that the power of distinguishability that first-order logic has is perhaps not enough. And, it does not necessarily capture exactly all the properties that one would like which characterize a unique model. So, in essentially in this non-standard models what we showed is that, in addition to the existing a standard natural numbers. One could actually, have a whole copies of classes of integers on both sides and they would not satisfies the same standard axioms of number theory. And, this means that the first-order theory of numbers with the axioms that we have got a does not have that fine

discriminating power Which, can exclude all those copies of integers what were called a Z chains.

(Refer Slide Time: 23:15)



Z-Chains

- Z' is called a *Z-Chain*.
- $N'_S = \langle \mathbb{N} \cup Z', \Sigma_S \rangle$ is also a *countable* model of the axioms Φ_S .
- Further N_S and N'_S are **not isomorphic**
- We could add a countable number of distinct Z-chains, Z'' , Z''' , Z'''' , etc. to obtain other distinct and mutually non-isomorphic models.
- Each of the models obtained above is also a *countable* model of Φ_S .
- Each of these models is also *non-standard*.

NPTEL

So, it could not actually eliminate them I mean. So, that that fineness of expressiveness of the axioms is something that absent in first-order logic especially the first-order theory of numbers. And, therefore there is there is an obvious question of how does one proceed. Other thing is that, a for example the first-order theory of numbers with of piano arithmetic. For, example includes an property of the principle of mathematical induction which, in the first-order case is extremely restricted. Whereas, piano is originally intended principle was for all possible properties of all possible subsets of numbers which might only be characterized by which might be parameterized by as by a some single integer or natural number.

(Refer Slide Time: 24:29)

The Limitations of Predicate Logic

1. The property of well-orderings viz.
Every subset of a well-ordered set has a least element cannot even be expressed in FOL. It requires the power of second-order logic
2. Transitive closures are not expressible
3. Isomorphic models cannot be distinguished
4. The existence of non-standard models implies that not all non-isomorphic models may be distinguished either.
5. The validity problem is undecidable.

So, let us look at certain extensions one is so this so this one possibility is one thing is look, at this property of well-orderings by the way this should be for every non-empty subset. Every non-empty subset of a well-ordered set has a least element. Now, this cannot even be expressed in FOL because it requires a power of second-order logic.

(Refer Slide Time: 24:49)

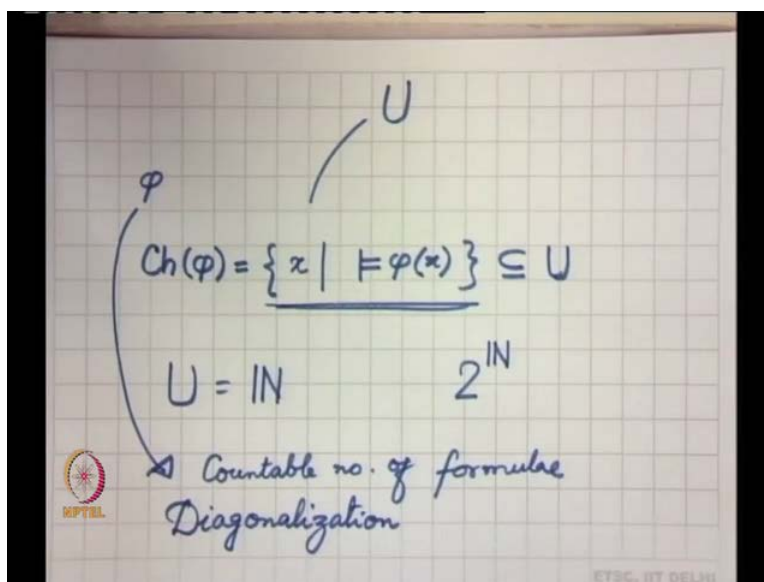
S, \leq

$\varphi \neq T \subseteq S$
└ least element.

$\forall x [\exists z [z \in X] \rightarrow \forall y [y \in X \rightarrow z \leq y]]$

Because, we are essentially saying that one should be able to given a, set S the well-ordering principle along with the first-order predicate less than or equal to. What, we are saying is that the well-ordering principle essentially says that we are it is not just about individuals of S. But, subsets of S so, if I have some s any non-empty subset of S. Then, this non-empty subset of S has a least element. So, how would one express this one would essentially express this by saying for all X. Such, that there exist some z such that z belongs to X. So, this indicates that X is a non-empty set. For, all X such that there exist a z such that z belongs to S. This, indicates that for all y the there exist an X this is a small x such that x belongs to X. And, for all y such that y belongs to X. We, have that x is less than or equal to y. Now, here it is important to see that this X this capital X throughout actually represents subsets of the domain of this course. And, this is not something that we allowed in first-order logic. So, even a very basic principle like well-ordering cannot be expressed in first-order logic. And, it request at least the power of second-order logic. And, in second-order logic essentially what we are allowing is quantification over [subs/subset] sets or quantification over first-order properties. If, you were to think of a subset of S you can think of it as a unary predicate representing a property. So, while is true that, every unary property has a characteristic set.

(Refer Slide Time: 27:51)



So, for any property phi so let us say phi is the first-order property. Then, I can think of a characteristic set of phi has being defined as the set of all elements x. Such, that phi of x is true

this given a universe of discourse U this set is a subset of U . However, a not all subsets of U may be characterized by properties. However, in order in second-order logic every property every first-order property is characterized by subset. So, what we mean by the second-order property is that now, we should be able to allow a quantification over subsets or over or even firstly over properties definitely but more importantly we would require quantification over subsets. So, one of the reasons I am saying that while every first-order property has a characteristic set subset of the universe of discourse not every subset of the universe of discourse can be characterized by a first-order property. Because, if you look if the universe of discourse say the natural numbers then the number of subsets of natural numbers is uncountable. However, our language of first-order properties has only a at most Countable number of formulae. So, therefore while every property every unary property is characterized by a subset of \mathbb{N} among the natural numbers not every subset of natural numbers can be expressed as a unary property. And, a simple diagonalization argument would show that this is this is intact true that not every property of the natural numbers is expressible in first-order logic.

(Refer Slide Time: 30:24)

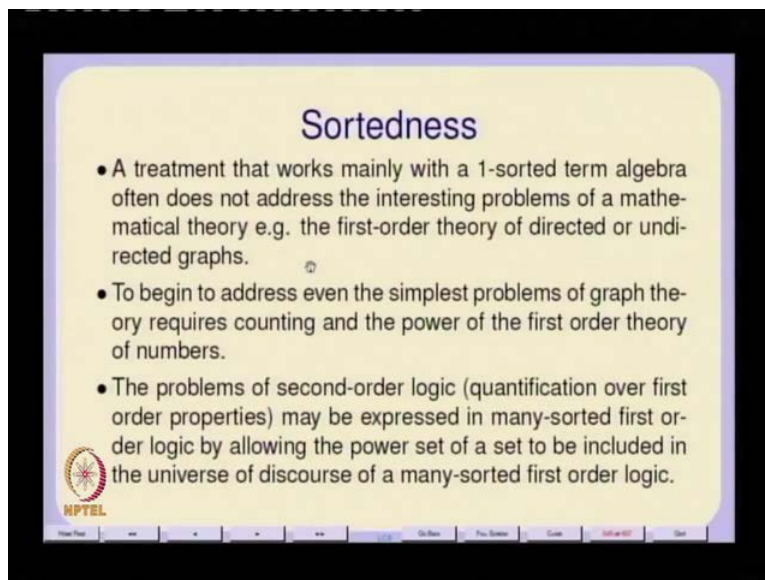
The Limitations of Predicate Logic

1. The property of well-orderings viz.
Every subset of a well-ordered set has a least element cannot even be expressed in FOL. It requires the power of second-order logic
2. Transitive closures are not expressible
3. Isomorphic models cannot be distinguished
4. The existence of non-standard models implies that not all non-isomorphic models may be distinguished either.
5. The validity problem is undecidable.

NPTEL

The, other thing for example transitive closures are not expressible when this means that for example the first-order theory of directed graphs can have cannot even characterized some of the most elementary properties we see in graph in graph theory books.

(Refer Slide Time: 30:51)



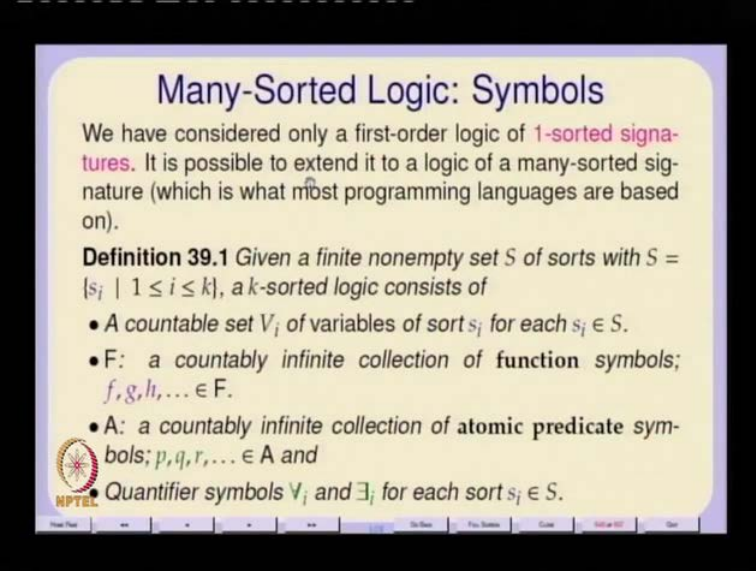
The slide is titled "Sortedness" and contains three bullet points. The first bullet point states that a treatment based on a 1-sorted term algebra often does not address interesting problems of mathematical theory, such as the first-order theory of directed or undirected graphs. The second bullet point notes that to address even the simplest problems of graph theory, one needs counting and the power of first-order theory of numbers. The third bullet point explains that second-order logic (quantification over first-order properties) can be expressed in many-sorted first-order logic by including the power set of a set in the universe of discourse. The NPTEL logo is visible in the bottom left corner of the slide.

So, moreover so one possibility is to actually tackle the notion of Sortedness so that is one thing. So, all are treatment was based on a 1-sorted term algebra and it often does not address the interesting problems of a mathematical theory. For, example and the first-order theory of directed or undirected graphs in any graph theory book. You, would be able you would want to count for example a, you would want to make statements like the directed graph is. Such, that every the sum of all in degrees of the graph is equal to the sum of all out degrees of the graphs so, that is so that includes counting. And, in the first-order theory of graphs there is no notion of counting. So, one possibility is to actually import also the theory of numbers and counting into the first-order theory of graphs. And, then try to axiomatize and try to derive properties of let us say directed or undirected graphs. So, this first step towards doing this is to actually dispense with a 1-sorted term algebra.

And, allow for a mini-sorted term algebra secondly if you look at this second-order formulation. This, is also something if you can think of instead of looking it as a structure of a set S with just the less than or equal to. If you also thought of this S union the power set of S along with the less than or equal to and may be some other properties regarding sets. Then, we could we could actually begin to think in terms of replacing this X this capital X by an individual here. And, so and individual could here denote either a single individual element from S or a single individual subset of S . And, of course then what we need to do is we need to be able to distinguish between

these two kinds of individuals. And, that is something that would be part of the sortedness. So, in particular second-order logic could then be transformed into a, 2-sorted first-order logic.

(Refer Slide Time: 33:17)



Many-Sorted Logic: Symbols

We have considered only a first-order logic of **1-sorted signatures**. It is possible to extend it to a logic of a many-sorted signature (which is what most programming languages are based on).

Definition 39.1 Given a finite nonempty set S of sorts with $S = \{s_i \mid 1 \leq i \leq k\}$, a k -sorted logic consists of

- A countable set V_i of variables of sort s_i for each $s_i \in S$.
- F : a countably infinite collection of function symbols; $f, g, h, \dots \in F$.
- A : a countably infinite collection of atomic predicate symbols; $p, q, r, \dots \in A$ and
- Quantifier symbols \forall_i and \exists_i for each sort $s_i \in S$.

NPTEL

And, that is what will see will look at it. We, look at this problem now but first is formalize the notion of Many-Sorted Logic. So, we had the notion of a 1-sorted logic Many-Sorted signature. Now, let us look at an many sorted signature this is important because for example most of our programming language is or and the and the composable programs that one can write in our programming in our general purpose programming languages. Or all essentially terms from a many-sorted algebra. Because, of the fact that first we are our programming language is he abstractly allow us to have many different kinds of elements integers, Boolean's, Real's, Characters strings. And, then functions defined between them a relations between them and so on and so forth. So, in that sense our programming language starts with its signature which, is already many-sorted. There, are at least 4 or 5 different kinds of basic elements basic sets from which our programs on which are program terms are defined.

So, we will call these different kinds as sorts. So, we have so assume given a finite non-empty set S of sorts with S equal to you know S_i $1 \leq i \leq k$. So, this symbol s here which I used for 1-sorted signatures as just a symbol. Now, gets replaced by a collection of such symbols one for each sort one for each kind of element that we are talking

about, our set of variables also they get slotted into various slots. So, we might think of the set of variables V as being divided up into countable disjoint subsets of variables where, each variable has a predefined sort associated with it and of course we need always a countable set for each sort. And, so will assume that our set of variables V is actually this disjoint union of variables V_i . Where, for each sort s_i in S . And, then we have a countably infinite collection of function symbols as we had here. And, that now what we allow a, function symbols to do is something that you will see later. It, allow us to take terms from different kinds of sorts. And, give you to terms of yet another kind of sort for example is possible to take terms from the integers two terms from integers and give you a Boolean. They, are less than relation can also be treated as a function for example if, you allow Boolean's as a sort. And, then of course an countable countably infinite collection of atomic predicate symbols as usual. And, then we because of this fact that we need to slot variables appropriately.

(Refer Slide Time: 37:10)

Many-Sorted Signatures

Definition 39.2 Given a finite nonempty set S of sorts with $S = \{s_i \mid 1 \leq i \leq k\}$, a S -sorted signature Σ consists of a set of strings of the form

- $f : s_{i_1}, s_{i_2}, \dots, s_{i_m} \rightarrow s_{i_0}, m \geq 0$
for each $f \in F$,
- $p : s_{i_1}, s_{i_2}, \dots, s_{i_n}, n \geq 0$
such that there is at most one string for each $f \in F$ and each $p \in A$.

A binary equality relation $=_i : s_i^2$ for some of the sorts $s_i \in S$.

NPTEL

We, also allow for a quantifier symbols one for each sort. So, this so you were to look at this so, I have put a subscript i here I have put a subscript i here. Essentially, to say that there is we also have for different kinds of elements for different kinds of variables we have different quantifiers. Then, a so a many sorted signature is something in which you have your functions f , which might take. Let us, say which might be an memory function taking elements from m possible different sorts and giving you $m + 1$ sort which, is also different. Your predicates likewise

could take could be parameterized on n different a sorts. And, still and therefore they allow you to explicate relationships between the various sorts. Then, of course for each sort it might be necessary to also have a binary equality relation equalities rather specialize as I told you before. Because, very often we might be talking of two putatively different elements by giving them different variable names x and y . And, then we might have to prove that they are really the same element in which case equality becomes a, necessary predicate binary predicate for each sort. So, will have this equality relation subscripted by i to denote an equality relation on the sort s_i .

(Refer Slide Time: 38:34)

Many-Sorted Signature: Terms

Definition 39.3 Given a S -sorted signature Σ , the set $T(\Sigma) = \bigcup_{1 \leq i \leq k} T_i(\Sigma)$ of Σ -terms is the disjoint union of the sets of terms $T_i(\Sigma)$ defined inductively such that every term is assigned a sort from S .

$$s, t, u ::= x_i \in V_i \mid f(t_1, \dots, t_m)$$

where $f : s_{i_1}, s_{i_2}, \dots, s_{i_m} \rightarrow s_{i_0} \in \Sigma$,

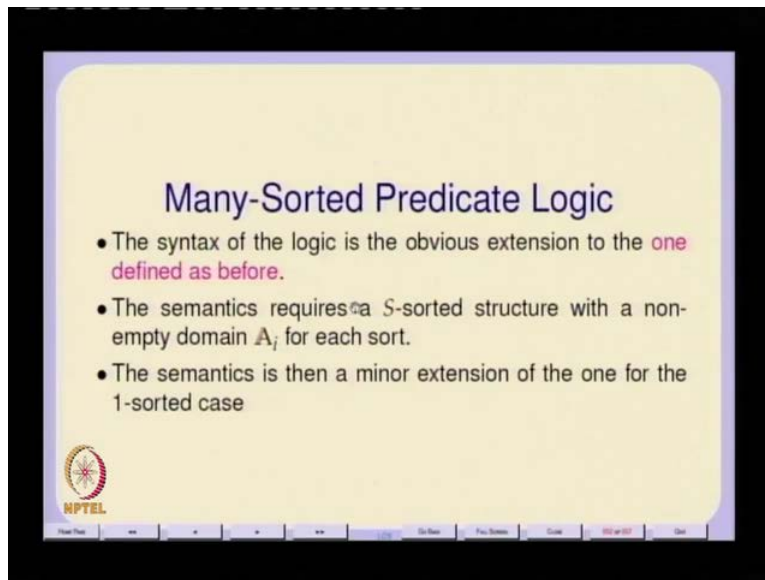
- each variable $x_i \in V_i$ is a term in $T_i(\Sigma)$, a fact usually denoted $x_i : s_i$.
- $f(t_1, \dots, t_m) \in T_{i_0}(\Sigma)$, a fact usually denoted $f(t_1, \dots, t_m) : s_{i_0}$.

NPTEL

Now, our terms are essentially like this the now, so this set of terms T sigma is the disjoint union of all possible terms which are been classified into sorts. So, every term in our term algebra now has a sort associated with it. And, that sort is one of the sorts taken from the set s . So, for example each variable x_i belonging to the variable set V_i is of sort s_i . Now, if I had a function f which was of type s_{i_1}, s_{i_2} to s_{i_m} going to s_{i_0} belonging to the signature. And, if I had terms t_1 to t_m which were respectively of these sorts which, were pre which have been defined as being of type s_i of being of sort $s_{i_1} s_{i_2} s_{i_m}$ respectively. Then, the net result of applying this function on these terms is to give you a term which belongs to the sort s_{i_0} . So, that is what we are saying here each variable x_i is a term in t sigma a fact usually denoted by this. So, this is our standard this is like our type declarations in a, programming languages. And, in the case of $f t_1$ to t_m it belongs to the sort s_{i_0} . So, therefore it belongs to the term of sets, the set of terms t_{i_0} of sigma.

And, once all terms have been classified in sorts the set of all terms is just this huge union of terms for each sort. You, just sort of agglomerate all the sorts of all the terms of different kinds into a single set and that you call the set of terms.

(Refer Slide Time: 40:42)



Many-Sorted Predicate Logic

- The syntax of the logic is the obvious extension to the **one defined as before.**
- The semantics requires a S -sorted structure with a non-empty domain A_i for each sort.
- The semantics is then a minor extension of the one for the 1-sorted case

NPTEL

So, there are Many-Sorted Predicate Logic then just has a syntax of the logic which is exactly as before and which we have seen before and so I am going to it a the semantics. Now, requires a S -sorted signature with a non-empty domain A_i for each sort s_i . Then, we just extend the semantics in a in a simple way similarly it becomes a little tedious to write it but, otherwise the extension is a fairly obvious. And, a this semantics is then a minor extension of essentially the 1-sorted case.

(Refer Slide Time: 41:28)

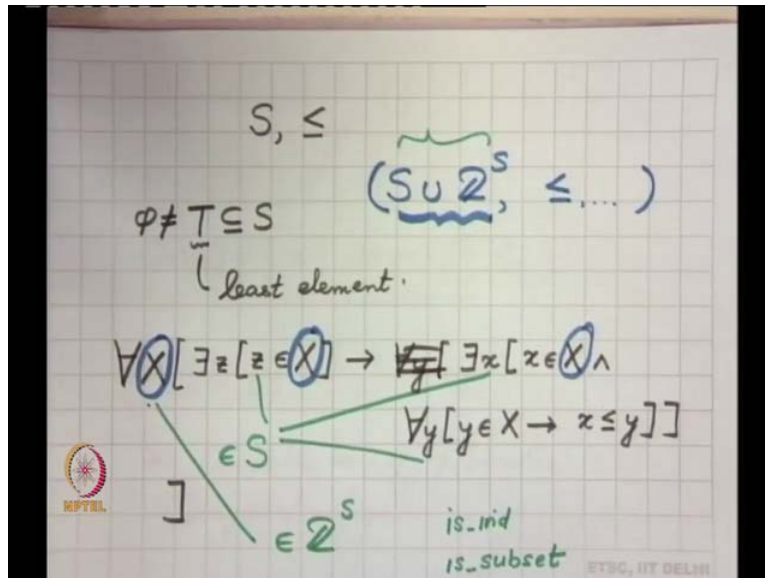
Reductions

- Second-Order logic may simply be considered a 2-sorted logic with predicates parameterised over both individuals and sets of individuals.
- An S-sorted Predicate Logic may be reduced to a 1-sorted Predicate logic by introducing a fresh set of unary-predicates is_{s_i} (one for each sort s_i) to denote membership in a sort and
- replace every quantified formula of the form $\forall_i x_i [\phi]$ by the 1-sorted formula $\forall x [is_{s_i}(x) \rightarrow \phi]$ and recursively for each quantifier,
- replace every quantified formula of the form $\exists_i x_i [\phi]$ by the 1-sorted formula $\exists x [is_{s_i}(x) \wedge \phi]$ and recursively for each quantifier,

NPTEL

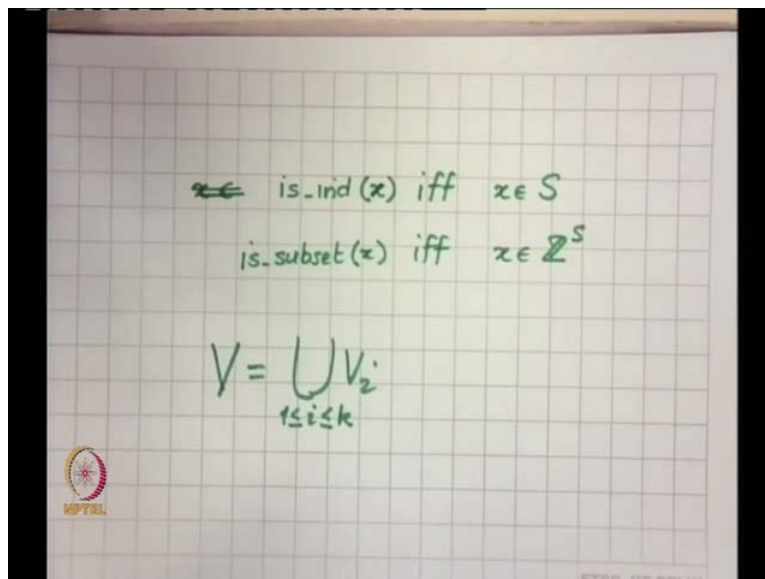
Once we have this many sorted predicate logic we can actually do one thing. Firstly a, second-order logic such is this which requires as you can see basically this notion of well-ordering. It, is a second-order predicate simply because we need to be able to have two different kinds of individuals one kind of individual ranges over this set over the set S of the universe of discourse. And, the other the capital X is actually range over two raise to S the power set of S. So, we can think of this as a many as a 2-sorted algebra. As, a 2-sorted structure with a once with a 1-sorted predicate with a first-order predicate less than or equal to. And, what we might do is we may just think of second-order logic. As, simply a 2-sorted logic with predicates parameterized over both individuals and set's of individuals. So, in that sense we have reduced second-order logic to a, 2-sorted first-order logic. This, actually a further Reduction possible any S sorted predicate logic may be reduced to a 1-sorted predicate logic. By, introducing a, fresh set of unary predicates and I am calling the set of unary predicates is is_i .

(Refer Slide Time: 43:15)



To, essentially what we are saying here is that, this individual's small z small x and small y belong to S. Whereas, this individual capital X belongs to two raise to S. So, in that sense we are dealing with two different kinds.

(Refer Slide Time: 43:59)



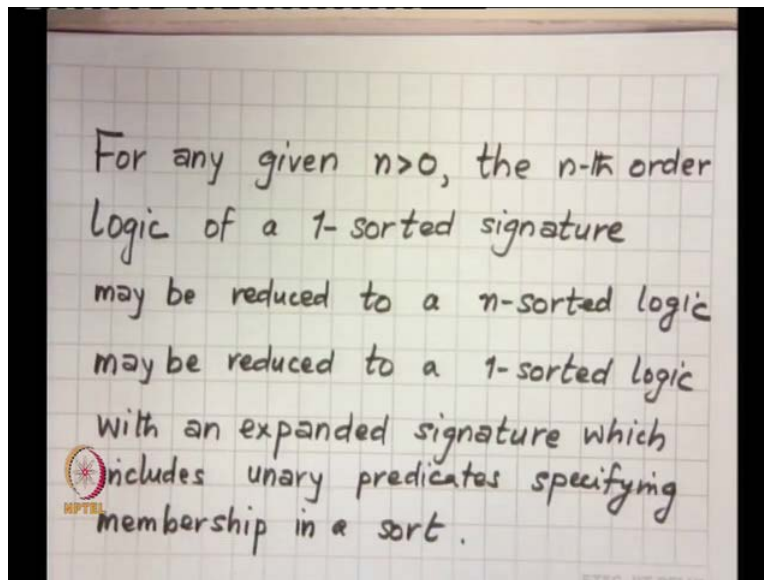
And, the membership or belonging to a certain kind can itself be regarded as a unary property which can be expressed by is individual is subset. For, example we can have two different we

can extend expand our signature to include these two first-order properties which, say that for any variable x is-ind of x is true if and only if x belongs to S . And, we might say that is-subset of x is true if and only if x belongs to two raise to S belongs to the power S . Then, when we do this then it is not anymore necessary to distinguish to syntactically have different symbols for the various variable various uses of variables. And, in fact we can conglomerate we can agglomerate all the variable sets V_i . So, we can just take V the set of variables to be just this V_i for all the sets $1 \leq i \leq k$. However, but what we need to do therefore is when we use quantification we need this implicit information to be used.

So, then every quantified formula for all x_i ϕ should be replaced by the 1-sorted formula for all x . If, x is s_i then ϕ where a now recursively you should we need to transform all the quantifiers inside ϕ also to the 1-sorted case. Similarly, for every quantified formula of the form there exist x_i ϕ we, replace it by the 1-sorted formula there exist x . Such that, is s_i x and ϕ where again in ϕ we replace all the quantifiers by this qualification. So, you can think of this s sort of bounded so this allows. So, this essentially is like a bounded quantification which specifies that type of each individual or membership in a sort it specifies it as a unary predicate. So, types that way this is very similar to the notion of types S sets in programming languages.

And, we are essentially using this membership as a unary property of an element. So, it is possible to prove that all models of an S sorted predicate logic have corresponding images in the 1-sorted predicate logic with this expanded signature. And, therefore it is possible to show that any second-order logic may simply be reduced to a first-order logic in this two step reductions. So, there is a 2-sorted logic with a , which is in turn reduced to a 1-sorted logic by introducing x by expanding the signature to include. These, membership predicates or property a type properties if, you like. This, kind of reduction of second-order logic to a, first-order logic of course does have a cost I mean certain theorems like the Lionines Skolem theorem no longer begin to hold. But, models can be exclusively constructed and in fact it is this it is this property that is used in logic programming to do all kinds of higher order programming also within the domain of a first-order resolution. But, of course this kind of reduction applies only to a, fixed finite n -th order logic.

(Refer Slide Time: 48:17)



So, for any given n the n -th order logic so for any given n the n -th order logic of a 1-sorted signature may be reduced to a, n -sorted logic and, which may in turn be reduced to a 1-sorted logic within expanded signature which, includes unary predicates specifying the specifying membership in the sort. And, of course correspondingly all the quantified formulae have to use this transformation which specifies the sort as a unary predicate. That is how one word take for any n . However, this is not true for all n in the sense that if, you look at higher order logic it transcends any fixed bound of n . And, therefore a higher order logic allows for arbitrary a sorting of arbitrary creation of subsets in indefinitely infinitely. So, whatever I am saying a refers to a fixed n . So, by an inductive process we can actually reduce any for any fixed n any n 's n -th order logic can be reduced to a first-order logic with an expanded signature which, specifies the hierarchy of types in the quantified in the quantified formulae. And, appropriately what it does is that every formula in that n -th order logic a goes through a transformation process which, specifies which actually transforms every quantified formula in this fashion. And, so and the otherwise the reasoning mechanisms for all these logics are more or less whatever is there in the first-order case certain decidability issues become important. So, it is not true that decidability by this reduction you can a, reduce all decidability also to first-order decidability. Because that has certain special problems associated with it. For, example so, we will not actually consider those things. But, you can see that in its essence all reasoning mechanisms as we looked at in our

first lecture we spoke about validity of arguments consistency matters of truth and false and reasoning. All those reasoning mechanisms essentially carry over from first-order logic to other n-th order logics. And, well that I would like to actually conclude this session and I hope you will be able to expand your knowledge further about higher order logics we using this set of lectures as a basis.