

Logic for Computer Science

<http://www.cse.iitd.ac.in/~sak/courses/ilcs/2012-13.index.html>

S. Arun-Kumar

*Department of Computer Science and Engineering
I. I. T. Delhi, Hauz Khas, New Delhi 110 016.*

April 16, 2015

Contents

0 Background Preliminaries	25
0.1 Motivation for the Study of Logic	25
0.2 Sets	29
0.3 Relations and Functions	34
0.4 Operations on Binary Relations	44
0.5 Ordering Relations	48
0.6 Partial Orders	50
0.7 Infinite Sets: Countability and Uncountability	51
0.8 Induction Principles	70
0.9 Mathematical Induction	70
0.10 Complete Induction	76
0.11 Structural Induction	86
0.12 Simultaneous Induction	103
1 Lecture 1: Introduction	111
2 Lecture 2: Propositional Logic Syntax	126

2.1	Propositions in Natural Languages	134
2.2	Translation of Natural Language Statements	141
2.3	Associativity and Precedence of Operators	150
2.4	Rooted Trees	162
3	Lecture 3: Semantics of Propositional Logic	169
4	Lecture 4: Logical and Algebraic Concepts	182
5	Lecture 5: Identities and Normal Forms	193
6	Lecture 6: Tautology Checking	210
7	Lecture 7: Propositional Unsatisfiability	250
7.1	Space Complexity of Propositional Resolution.	261
7.2	Time Complexity of Propositional Resolution	262
8	Lecture 8: Analytic Tableaux	271
9	Lecture 9: Consistency & Completeness	282

10 Lecture 10: The Compactness Theorem	296
11 Lecture 11: Maximally Consistent Sets	306
12 Lecture 12: Formal Theories	322
13 Lecture 13: Proof Theory: Hilbert-style	337
14 Lecture 14: Derived Rules	352
15 Lecture 15: The Hilbert System: Soundness	375
16 Lecture 16: The Hilbert System: Completeness	386
17 Lecture 17: Introduction to Predicate Logic	398
18 Lecture 18: The Semantics of Predicate Logic	421
19 Lecture 19: Substitutions	437
20 Lecture 20: Models, Satisfiability and Validity	452
20.1 Some Model Theory	469

21 Lecture 21: Structures and Substructures	472
22 Lecture 22: Predicate Logic: Proof Theory	489
23 Lecture 23: Predicate Logic: Proof Theory (Contd.)	511
24 Lecture 24: Existential Quantification	525
25 Lecture 25: Normal Forms	542
26 Lecture 26: Skolemization	563
27 Lecture 27: Substitutions and Instantiations	585
28 Lecture 28: Unification	597
29 Lecture 29: Resolution in FOL	630
30 Lecture 30: More on Resolution in FOL	643
31 Lecture 31: Resolution: Soundness and Completeness	655
32 Lecture 32: Resolution and Tableaux	674

33 Lecture 33: Completeness of Tableaux Method	683
34 Lecture 34: Completeness of the Hilbert System	692
34.1 Model-theoretic and Proof-theoretic Consistency	695
35 Lecture 35: First-Order Theories	706
36 Lecture 36: Towards Logic Programming	724
37 Lecture 37: Verification of Imperative Programs	740
38 Lecture 38: Verification of WHILE Programs	752
39 Lecture 39: Concluding Remarks	770
40 References	782

List of Slides

- Lecture 1: Introduction

1. What is Logic?
2. Reasoning, Truth and Validity
3. Examples
4. Objectivity in Logic
5. Formal Logic
6. Formal Logic: Applications
7. Form and Content
8. Facets of Mathematical Logic
9. Logic and Computer Science

- Lecture 2: Propositional Logic Syntax

1. Truth and Falsehood: 1
2. Truth and Falsehood: 2
3. Extending the Boolean Algebra
4. Table of Truth & Falsehood
5. Sums & Products
6. Propositional Logic: Syntax
7. Propositional Logic: Syntax - 2
8. Natural Language equivalents
9. Some Remarks

10. Associativity and Precedence

11. Syntactic Identity

12. Abstract Syntax Trees

13. Subformulae

14. Atoms in a Formula

15. Degree of a Formula

16. Size of a Formula

17. Height of a Formula

- Lecture 3: Semantics of Propositional Logic

1. Semantics of Propositional Logic: 1

2. Semantics of Propositional Logic: 2

3. Models and Satisfiability

4. Example: Abstract Syntax trees

5. Tautology, Contradiction, Contingent

- Lecture 4: Logical and Algebraic Concepts

1. Logical Consequence: 1

2. Logical Consequence: 2

3. Other Theorems

4. Logical Implication

5. Implication & Equivalence

6. Logical Equivalence as a Congruence

- Lecture 5: Identities and Normal Forms

1. Adequacy
2. Adequacy: Examples
3. Functional Completeness
4. Duality
5. Principle of Duality
6. Negation Normal Forms: 1
7. Negation Normal Forms: 2
8. Conjunctive Normal Forms
9. CNF

- Lecture 6: Tautology Checking

1. Arguments
2. Arguments: 2
3. Validity & Falsification
4. Translation into propositional Logic
5. Atoms in Argument
6. The Representation
7. Propositional Rendering
8. The Strategy
9. Checking Tautology
10. Computing the CNF
11. Falsifying CNF

- Lecture 7: Propositional Unsatisfiability

1. Tautology Checking
2. CNFs: Set of Sets of Literals
3. Propositional Resolution
4. Clean-up
5. The Resolution Method
6. The Algorithm
7. Resolution Examples: Biconditional
8. Resolution Examples: Exclusive-Or
9. Resolution Refutation: 1
10. Resolution Refutation: 2
11. Resolvent as Logical Consequence
12. Logical Consequence by Refutation

- Lecture 8: Analytic Tableaux

1. Against Resolution
2. The Analytic Tableau Method
3. Basic Tableaux Facts
4. Tableaux Rules
5. Structure of the Rules
6. Tableaux
7. Slim Tableaux

- Lecture 9: Consistency & Completeness

- 1. Tableaux Rules: Restructuring
- 2. Tableaux Rules: 2
- 3. Tableau Proofs
- 4. Consistency
- 5. Unsatisfiability
- 6. Hintikka Sets
- 7. Hintikka's Lemma
- 8. Tableaux and Hintikka sets
- 9. Completeness

- Lecture 10: The Compactness Theorem

- 1. Satisfiability of Infinite Sets
- 2. The Compactness Theorem
- 3. Inconsistency
- 4. Consequences of Compactness

- Lecture 11: Maximally Consistent Sets

- 1. Consistent Sets
- 2. Properties of Finite Character: 1
- 3. Properties of Finite Character: 2
- 4. Properties of Finite Character: 3
- 5. Maximally Consistent Sets
- 6. Lindenbaum's Theorem

- Lecture 12: Formal Theories

1. Introduction to Reasoning
2. Proof Systems: 1
3. Requirements of Proof Systems
4. Proof Systems: Desiderata
5. Formal Theories
6. Formal Language
7. Axioms and Inference Rules
8. Axiomatic Theories
9. Syntax and Decidability
10. A Hilbert-style Proof System
11. Rule Patterns

- Lecture 13: Proof Theory: Hilbert-style

1. More About Formal Theories
2. An Example Proof
3. Formal Proofs
4. Provability and Formal Proofs
5. The Deduction Theorem
6. About Formal Proofs

- Lecture 14: Derived Rules

1. Simplifying Proofs

- 2. Derived Rules
 - 3. The Sequent Form
 - 4. Proof trees in sequent form
 - 5. Transitivity of Conditional
 - 6. Derived Double Negation Rules
 - 7. Derived Operators
 - 8. Rules for Derived Operators
- Lecture 15: The Hilbert System: Soundness
 - 1. Formal Theory: Issues
 - 2. Formal Theory: Incompleteness
 - 3. Soundness of Formal Theories
 - 4. Soundness of the Hilbert System
 - 5. Soundness of the Hilbert System
 - Lecture 16: The Hilbert System: Completeness
 - 1. Towards Completeness
 - 2. Towards Truth-tables
 - 3. The Truth-table Lemma
 - 4. The Completeness Theorem
 - Lecture 17: Introduction to Predicate Logic
 - 1. Predicate Logic: Introduction-1
 - 2. Predicate Logic: Introduction-2

- 3. Predicate Logic: Introduction-3
- 4. Predicate Logic: Symbols
- 5. Predicate Logic: Signatures
- 6. Predicate Logic: Syntax of Terms
- 7. Predicate Logic: Syntax of Formulae
- 8. Precedence Conventions
- 9. Predicates: Abstract Syntax Trees
- 10. Subterms
- 11. Variables in a Term
- 12. Bound Variables And Scope
- 13. Bound Variables And Scope: Example
- 14. Scope Trees
- 15. Free Variables
- 16. Bound Variables
- 17. Closure

- Lecture 18: The Semantics of Predicate Logic

- 1. Structures
- 2. Notes on Structures
- 3. Infix Convention
- 4. Expansions and Reducts
- 5. Valuations and Interpretations
- 6. Evaluating Terms

- 7. Coincidence Lemma for Terms
- 8. Variants
- 9. Variant Notation
- 10. Semantics of Formulae
- 11. Notes on the Semantics

- Lecture 19: Substitutions

- 1. Coincidence Lemma for Formulae
- 2. Substitutions
- 3. Instantiation of Terms
- 4. The Substitution Lemma for Terms
- 5. Admissibility
- 6. Instantiations of Formulae
- 7. The Substitution Lemma for Formulae

- Lecture 20: Models, Satisfiability and Validity

- 1. Satisfiability
- 2. Models and Consistency
- 3. Examples of Models:1
- 4. Examples of Models:2
- 5. Examples of Models:3
- 6. Logical Consequence
- 7. Validity
- 8. Validity of Sets of Formulae

9. Negations of Semantical Concepts

- Lecture 21: Structures and Substructures

1. Satisfiability and Expansions
2. Distinguishability
3. Evaluations under Different Structures
4. Isomorphic Structures
5. The Isomorphism Lemma
6. Substructures
7. Substructure Examples
8. Quantifier-free Formulae
9. Lemma on Quantifier-free Formulae
10. Universal and Existential Formulae
11. The Substructure Lemma

- Lecture 22: Predicate Logic: Proof Theory

1. Proof Theory: First-Order Logic
2. Proof Rules: Hilbert-Style
3. The Mortality of Socrates
4. The Mortality of the Greeks
5. Faulty Proof:2
6. A Correct Proof
7. The Sequent Forms
8. The Case of Equality

- 9. Semantics of Equality
- 10. Axioms for Equality
- 11. Symmetry and Transitivity
- 12. Symmetry of Equality
- 13. Transitivity of Equality
- Lecture 23: Predicate Logic: Proof Theory (Contd.)
 - 1. Alpha Conversion
 - 2. The Deduction Theorem for Predicate Calculus
 - 3. Useful Corollaries
 - 4. Soundness of Predicate Calculus
 - 5. Soundness of The Hilbert System
- Lecture 24: Existential Quantification
 - 1. Existential Generalisation
 - 2. Existential Elimination
 - 3. Remarks on Existential Elimination
 - 4. Restrictions on Existential Elimination
 - 5. Equivalence of Proofs
- Lecture 25: Normal Forms
 - 1. Natural Deduction: 6
 - 2. Moving Quantifiers
 - 3. Quantifier Movement

- 4. More on Quantifier Movement
 - 5. Prenex Normal Forms
 - 6. The Prenex Normal Form Theorem
 - 7. Prenex Conjunctive Normal Form
 - 8. The Herbrand Algebra
 - 9. Terms in a Herbrand Algebra
 - 10. Herbrand Interpretations
 - 11. Herbrand Models
 - 12. Ground Quantifier-free Formulae
- Lecture 26: Skolemization

- 1. Skolemization
- 2. Skolem Normal Forms
- 3. SCNF
- 4. Ground Instance
- 5. Herbrand's Theorem
- 6. The Herbrand Tree of Interpretations
- 7. Compactness of Sets of Ground Formulae
- 8. Compactness of Closed Formulae
- 9. The Löwenheim-Skolem Theorem

- Lecture 27: Substitutions and Instantiations
- 1. Substitutions Revisited
 - 2. Some Simple Facts

- 3. Ground Substitutions
 - 4. Composition of Substitutions
 - 5. Substitutions: A Monoid
- Lecture 28: Unification
 - 1. Unifiability
 - 2. Unification Examples:1
 - 3. Unification Examples:2
 - 4. Generality of Unifiers
 - 5. Generality: Facts
 - 6. Most General Unifiers
 - 7. More on Positions
 - 8. Disagreement Set
 - 9. Example: Disagreement 1
 - 10. Example: Occurs Check
 - 11. Example: Disagreement 3
 - 12. Example: Disagreement 4
 - 13. Disagreement and Unifiability
 - 14. The Unification Theorem
 - Lecture 29: Resolution in FOL
 - 1. Recapitulation
 - 2. SCNFs and Models
 - 3. SCNFs and Unsatisfiability

- 4. Representing SCNFs
 - 5. Clauses: Terminology
 - 6. Clauses: Ground Instances
 - 7. Facts about Clauses
 - 8. Clauses: Models
 - 9. Clauses: Herbrand's Theorem
 - 10. Resolution in FOL
- Lecture 30: More on Resolution in FOL
 - 1. Standardizing Variables Apart
 - 2. Factoring
 - 3. Example: 1
 - 4. Example: 2
 - 5. Refutation
 - Lecture 31: Resolution: Soundness and Completeness
 - 1. Soundness of FOL Resolution
 - 2. Ground Clauses
 - 3. The Lifting Lemma
 - 4. Lifting Lemma: Figure
 - 5. Completeness of Resolution Refutation: 1
 - 6. Completeness of Resolution Refutation: 2
 - 7. Completeness of Resolution Refutation: 3

- Lecture 32: Resolution and Tableaux
 - 1. FOL: Tableaux
 - 2. FOL: Tableaux Rules
 - 3. FOL Tableaux: Example 1
 - 4. First-Order Tableaux
 - 5. FOL Tableaux: Example 2
- Lecture 33: Completeness of Tableaux Method
 - 1. First-order Hintikka Sets
 - 2. Hintikka's Lemma for FOL
 - 3. First-order tableaux and Hintikka sets
 - 4. Soundness of First-order Tableaux
 - 5. Completeness of First-order Tableaux
- Lecture 34: Completeness of the Hilbert System
 - 1. Deductive Consistency
 - 2. Models of Deductively Consistent Sets
 - 3. Deductive Completeness
 - 4. The Completeness Theorem
- Lecture 35: First-Order Theories
 - 1. (Simple) Directed Graphs
 - 2. (Simple) Undirected Graphs
 - 3. Irreflexive Partial Orderings

- 4. Irreflexive Linear Orderings
- 5. (Reflexive) Preorders
- 6. (Reflexive) Partial Orderings
- 7. (Reflexive) Linear Orderings
- 8. Equivalence Relations
- 9. Peano's Postulates
- 10. The Theory of The Naturals
- 11. Notes and Explanations
- 12. Finite Models of Arithmetic
- 13. A Non-standard Model of Arithmetic
- 14. Z-Chains

- Lecture 36: Towards Logic Programming

- 1. Reversing the Arrow
- 2. Horn Clauses
- 3. Goal clauses
- 4. Logic Programs
- 5. Sorting in Logic
- 6. Prolog: Selection Sort
- 7. Prolog: Merge Sort
- 8. Prolog: Quick Sort
- 9. Prolog: SEND+MORE=MONEY
- 10. Prolog: Naturals

- Lecture 37: Verification of Imperative Programs

1. The WHILE Programming Language
2. Programs As State Transformers
3. The Semantics of WHILE
4. Programs As Predicate Transformers
5. Correctness Assertions
6. Total Correctness of Programs
7. Examples: Factorial 1
8. Examples: Factorial 2

- Lecture 38: Verification of WHILE Programs

1. Proof Rule: Epsilon
2. Proof Rule: Assignment
3. Proof Rule: Composition
4. Proof Rule: The Conditional
5. Proof Rule: The While Loop
6. The Consequence Rule
7. Proof Rules for Partial Correctness
8. Example: Factorial 1
9. Towards Total Correctness
10. Termination and Total Correctness
11. Example: Factorial 2
12. Notes on Example: Factorial

13. Example: Factorial 2 Made Complete

14. An Open Problem: Collatz

- Lecture 39: Concluding Remarks

1. Summary
2. The Limitations of Predicate Logic
3. Sortedness
4. Many-Sorted Logic: Symbols
5. Many-Sorted Signatures
6. Many-Sorted Signature: Terms
7. Many-Sorted Predicate Logic
8. Reductions

0. Background Preliminaries

al-go-rism *n.* [ME algorsme<OFr.<Med.Lat. *algorismus*, after Muhammad ibn-Musa Al-Kharzimi (780-850?).] *The Arabic system of numeration:* DECIMAL SYSTEM.

al-go-rithm *n* [Var. of ALGORISM.] *Math. A mathematical rule or procedure for solving a problem.*

△word history: Algorithm originated as a variant spelling of algorism. The spelling was probably influenced by the word aruthmetic or its Greek source arithm, "number". With the development of sophisticated mechanical computing devices in the 20th century, however, algorithm was adopted as a convenient word for a recursive mathematical procedure, the computer's stock in trade. Algorithm has ceased to be used as a variant form of the older word.

Webster's II New Riverside University Dictionary 1984.

0.1. Motivation for the Study of Logic

In the early years of this century symbolic or formal logic became quite popular with philosophers and mathematicians because they were interested in the concept of what constitutes a correct proof

in mathematics. Over the centuries mathematicians had pronounced various mathematical proofs as correct which were later disproved by other mathematicians. The whole concept of logic then hinged upon what is a correct argument as opposed to a wrong (or faulty) one. This has been amply illustrated by the number of so-called proofs that have come up for Euclid's parallel postulate and for Fermat's last theorem. There have invariably been "bugs" (a term popularised by computer scientists for the faults in a program) which were often very hard to detect and it was necessary therefore to find infallible methods of proof. For centuries (dating back at least to Plato and Aristotle) no rigorous formulation was attempted to capture the notion of a correct argument which would guide the development of all mathematics.

The early logicians of the nineteenth and twentieth centuries hoped to establish formal logic as a foundation for mathematics, though that never really happened. But mathematics does rest on one firm foundation, namely set theory. But Set theory itself has been expressed in first order logic. What really needed to be answered were questions relating to the automation or mechanizability of proofs. These questions are very relevant and important for the development of present-day computer science and form the basis of many developments in automatic theorem proving. David Hilbert asked the important question, as to whether all mathematics, if reduced to statements of symbolic logic, can be derived by a machine. Can the act of constructing a proof be reduced to the manipulation of statements in symbolic logic? Logic enabled mathematicians to point out why an

alleged proof is wrong, or where in the proof, the reasoning has been faulty. A large part of the credit for this achievement must go to the fact that by symbolising arguments rather than writing them out in some natural language (which is fraught with ambiguity), checking the correctness of a proof becomes a much more viable task. Of course, trying to symbolise the whole of mathematics could be disastrous as then it would become quite impossible to even read and understand mathematics, since what is presented usually as a one page proof could run into several pages. But at least in principle it can be done.

Since the latter half of the twentieth century logic has been used in computer science for various purposes ranging from program specification and verification to theorem-proving. Initially its use was restricted to merely specifying programs and reasoning about their implementations. This is exemplified in the some fairly elegant research on the development of correct programs using first-order logic in such calculi such as the weakest-precondition calculus of Dijkstra. A method called Hoare Logic which combines first-order logic sentences and program phrases into a specification and reasoning mechanism is also quite useful in the development of small programs. Logic in this form has also been used to specify the meanings of some programming languages, notably Pascal.

The close link between logic as a formal system and computer-based theorem proving is proving to be very useful especially where there are a large number of cases (following certain patterns)

to be analysed and where quite often there are routine proof techniques available which are more easily and accurately performed by theorem-provers than by humans. The case of the four-colour theorem which until fairly recently remained a unproved conjecture is an instance of how human ingenuity and creativity may be used to divide up proof into a few thousand cases and where machines may be used to perform routine checks on the individual cases. Another use of computers in theorem-proving or model-checking is the verification of the design of large circuits before a chip is fabricated. Analysing circuits with a billion transistors in them is at best error-prone and at worst a drudgery that few humans would like to do. Such analysis and results are best performed by machines using theorem proving techniques or model-checking techniques.

A powerful programming paradigm called declarative programming has evolved since the late seventies and has found several applications in computer science and artificial intelligence. Most programmers using this logical paradigm use a language called Prolog which is an implemented form of logic¹. More recently computer scientists are working on a form of logic called constraint logic programming.

In the rest of this chapter we will discuss sets, relations, functions. Though most of these topics are covered in the high school curriculum this section also establishes the notational conventions that

¹actually a subset of logic called Horn-clause logic

will be used throughout. Even a confident reader may wish to browse this section to get familiar with the notation.

0.2. Sets

A *set* is a collection of *distinct* objects. The class of CS253 is a set. So is the group of all first year students at IITD. We will use the notation $\{a, b, c\}$ to denote the collection of the objects a , b and c . The elements in a set are not ordered in any fashion. Thus the set $\{a, b, c\}$ is the same as the set $\{b, a, c\}$. Further, repetitions of elements in a set do not change it in any way. Two sets are *equal* if they contain exactly the same elements. Hence the sets $\{a, b, c\}$, $\{a, b, c, a\}$, $\{b, a, c\}$, $\{c, b, a, c\}$ are all equal.

We can describe a set either by enumerating all the elements of the set or by stating the properties that uniquely characterize the elements of the set. Thus, the set of all even positive integers not larger than 10 can be described either as $S = \{2, 4, 6, 8, 10\}$ or, equivalently, as $S = \{x \mid x \text{ is an even positive integer not larger than } 10\}$

A set can have another set as one of its elements. For example, the set $A = \{\{a, b, c\}, d\}$ contains two elements $\{a, b, c\}$ and d ; and the first element is itself a set. We will use the notation $x \in S$ to

denote that x is an *element of* (or *belongs to*) the set S .

A set A is a *subset* of another set B , denoted as $A \subseteq B$, if $x \in B$ whenever $x \in A$.

An *empty set* is one which contains no elements and we will denote it with the symbol \emptyset . For example, let S be the set of all students who fail this course. S might turn out to be empty (hopefully; if everybody studies hard). By definition, the empty set \emptyset is a subset of all sets. We will also assume an *Universe of discourse* \mathbb{U} , and every set that we will consider is a subset of \mathbb{U} . Thus we have

1. $\emptyset \subseteq A$ for any set A
2. $A \subseteq \mathbb{U}$ for any set A

The *union* of two sets A and B , denoted $A \cup B$, is the set whose elements are exactly the elements of either A or B (or both). The *intersection* of two sets A and B , denoted $A \cap B$, is the set whose elements are exactly the elements that belong to *both* A and B . The *difference* of B from A , denoted $A - B$, is the set of all elements of A that do not belong to B . The *complement* of A , denoted $\sim A$ is the difference of A from the universe \mathbb{U} . Thus, we have

1. $A \cup B = \{x \mid (x \in A) \text{ or } (x \in B)\}$
2. $A \cap B = \{x \mid (x \in A) \text{ and } (x \in B)\}$

$$3. A - B = \{x \mid (x \in A) \text{ and } (x \notin B)\}$$

$$4. \sim A = \mathbb{U} - A$$

We also have the following named identities that hold for all sets A , B and C .

Basic properties of set union.

$$1. (A \cup B) \cup C = A \cup (B \cup C)$$
 Associativity

$$2. A \cup \phi = A$$
 Identity

$$3. A \cup \mathbb{U} = \mathbb{U}$$
 Zero

$$4. A \cup B = B \cup A$$
 Commutativity

$$5. A \cup A = A$$
 Idempotence

Basic properties of set intersection

$$1. (A \cap B) \cap C = A \cap (B \cap C)$$
 Associativity

$$2. A \cap \mathbb{U} = A$$
 Identity

$$3. A \cap \phi = \phi$$
 Zero

- | | |
|--------------------------|----------------------|
| 4. $A \cap B = B \cap A$ | <i>Commutativity</i> |
| 5. $A \cap A = A$ | <i>Idempotence</i> |

Other properties

- | | |
|---|---|
| 1. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ | <i>Distributivity of \cap over \cup</i> |
| 2. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | <i>Distributivity of \cup over \cap</i> |
| 3. $\sim (A \cup B) = \sim A \cap \sim B$ | <i>De Morgan's law $\sim \cup$</i> |
| 4. $\sim (A \cap B) = \sim A \cup \sim B$ | <i>De Morgan's law $\sim \cap$</i> |
| 5. $A \cap (\sim A \cup B) = A \cap B$ | <i>Absorption \cup</i> |
| 6. $A \cup (\sim A \cap B) = A \cup B$ | <i>Absorption \cap</i> |

The reader is encouraged to come up with properties of set difference and the complementation operations.

We will use the following notation to denote some standard sets:

The empty set: \emptyset

The Universe: \mathbb{U}

The set of Natural Numbers: $\mathbb{N} = \{0, 1, 2, \dots\}$. We will include 0 in the set of Natural numbers.
After all, it is quite natural to score a 0 in an examination!

The set of positive integers: $\mathbb{P} = \{1, 2, 3, \dots\}$

The two-element set: $\mathbb{2} = \{0, 1\}$. More generally for any natural number n we let $\mathbb{n} = \{0, 1, \dots, n-1\}$ the set of all naturals less than n . By convention \emptyset is the set of all naturals less than 0.

The set of integers: $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$

The set of rational numbers: \mathbb{Q}

The set of real numbers: \mathbb{R}

The Boolean set: $\mathbb{B} = \{false, true\}$

The Powerset of a set A : $\mathbb{2}^A$ is the set of all subsets of the set A .

0.3. Relations and Functions

The **Cartesian product** of two sets A and B , denoted by $A \times B$, is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$. Thus,

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

Given another set C we may form the following different kinds of cartesian products (which are not at all the same!).

$$(A \times B) \times C = \{((a, b), c) \mid a \in A, b \in B \text{ and } c \in C\}$$

$$A \times (B \times C) = \{(a, (b, c)) \mid a \in A, b \in B \text{ and } c \in C\}$$

$$A \times B \times C = \{(a, b, c) \mid a \in A, b \in B \text{ and } c \in C\}$$

The last cartesian product gives the construction of tuples. Elements of the set $A_1 \times A_2 \times \cdots \times A_n$ for given sets A_1, A_2, \dots, A_n are called *ordered n-tuples*.

A^n is the set of all ordered n -tuples (a_1, a_2, \dots, a_n) such that $a_i \in A$ for all i . i.e.,

$$A^n = \underbrace{A \times A \times \cdots \times A}_{n \text{ times}}$$

A **binary relation** \mathcal{R} from A to B is a subset of $A \times B$. It is a characterization of the intuitive notion that some of the elements of A are related to some of the elements of B . We also use the *infix* notation $a\mathcal{R}b$ to mean $(a, b) \in \mathcal{R}$. When A and B are the same set, we say \mathcal{R} is a binary relation *on* A . Familiar binary relations from \mathbb{N} to \mathbb{N} are $=, \neq, <, \leq, >, \geq$. Thus the elements of the set $\{(0, 0), (0, 1), (0, 2), \dots, (1, 1), (1, 2), \dots\}$ are all members of the relation \leq which is a subset of $\mathbb{N} \times \mathbb{N}$.

In general, an n -ary *relation* among the sets A_1, A_2, \dots, A_n is a subset of the set $A_1 \times A_2 \times \cdots \times A_n$.

Definition 0.1 Let $\mathcal{R} \subseteq A \times B$ be a binary relation from A to B . Then

1. For any set $A' \subseteq A$ the **image** of A' under \mathcal{R} is the set defined by

$$\mathcal{R}(A') = \{b \in B \mid a\mathcal{R}b \text{ for some } a \in A'\}$$

2. For every subset $B' \subseteq B$ the **pre-image** of B' under \mathcal{R} is the set defined by

$$\mathcal{R}^{-1}(B') = \{a \in A \mid a\mathcal{R}b \text{ for some } b \in B'\}$$

3. \mathcal{R} is **onto** (or **surjective**) with respect to A and B if $\mathcal{R}(A) = B$.
4. \mathcal{R} is **total** with respect to A and B if $\mathcal{R}^{-1}(B) = A$.
5. \mathcal{R} is **one-to-one** (or **injective**) with respect to A and B if for every $b \in B$ there is at most one $a \in A$ such that $(a, b) \in \mathcal{R}$.
6. \mathcal{R} is a **partial function** from A to B , usually denoted $\mathcal{R} : A \rightharpoonup B$, if for every $a \in A$ there is at most one $b \in B$ such that $(a, b) \in \mathcal{R}$. \mathcal{R} is a **total function** from A to B , usually denoted $\mathcal{R} : A \longrightarrow B$ if \mathcal{R} is a partial function from A to B and is total. Notice that every total function is also a partial function. A is called the **domain** and B the **co-domain**. The **range** of the function is $\mathcal{R}(A)$.
7. \mathcal{R} is a **one-to-one correspondence** (or **bijection**) if it is an injective and surjective total function.

Notes.

1. Given a (partial or total) function $f : A \rightharpoonup B$, the binary relation it corresponds to is called the *graph* of the function and $\text{graph}(f) = \{(a, b) \in A \times B \mid f(a) = b\}$.
2. A binary relation $\mathcal{R} \subseteq A \times B$ may also be thought of as a total function $\mathcal{R} : \mathcal{P}(A) \longrightarrow \mathcal{P}(B)$. Likewise \mathcal{R}^{-1} the converse of the relation \mathcal{R} , may be thought of as a total function $\mathcal{R}^{-1} : \mathcal{P}(B) \longrightarrow \mathcal{P}(A)$ (c.f.

parts 1 and 2 of definition 0.1 where relation symbol has been “overloaded”).

3. Similarly every partial function $f : A \rightharpoonup B$ may be “overloaded” to mean the *total* function $f : 2^A \longrightarrow 2^B$, which yields the image of A' for each $A' \subseteq A$. Likewise even though the converse (see part 2 of definition 0.6) of $\text{graph}(f)$ may not be a function, the total inverse function $f^{-1} : 2^B \longrightarrow 2^A$ is well defined and for each $B' \subseteq B$, yields the pre-image of B' .

Notation. Let f be a total function from set A to set B . Then

- $f : A \xrightarrow{1-1} B$ will denote that f is injective,
- $f : A \xrightarrow[\text{onto}]{} B$ will denote that f is surjective, and
- $f : A \xrightarrow[\text{onto}]^{1-1} B$ will denote that f is bijective,

Example 0.2 The following are some examples of familiar binary relations along with their properties.

1. The \leq relation on \mathbb{N} is a relation from \mathbb{N} to \mathbb{N} which is total and onto. That is, both the image and pre-image of \leq under \mathbb{N} are \mathbb{N} itself. What are image and the pre-image respectively of the relation $<?$

2. The binary relation which associates key sequences from a computer keyboard with their respective 8-bit ASCII codes is an example of a relation which is total and injective.
3. The binary relation which associates 7-bit ASCII codes with their corresponding ASCII characters is a bijection.

The figures 1, 2, 3, 4 and 5 respectively illustrate the concepts of partial, injective, surjective, bijective and inverse of a bijective function on finite sets. The directed arrows go from elements in the domain to their images in the codomain.

We may equivalently define partial and total functions as follows.

Definition 0.3 A **function** (or a **total function**) f from A to B is a binary relation $f \subseteq A \times B$ such that for every element $a \in A$ there is a unique element $b \in B$ so that $(a, b) \in f$ (usually denoted $f(a) = b$ and sometimes $f : a \mapsto b$). We will use the notation $R : A \rightarrow B$ to denote a function R from A to B . The set A is called the **domain** of the function R and the set B is called the **co-domain** of the function R . The **range** of a function $R : A \rightarrow B$ is the set $\{b \in B \mid \text{for some } a \in A, R(a) = b\}$. A **partial function** f from A to B , denoted $f : A \rightharpoonup B$ is a total function from some subset of A to the set B . Clearly every total function is also a partial function.

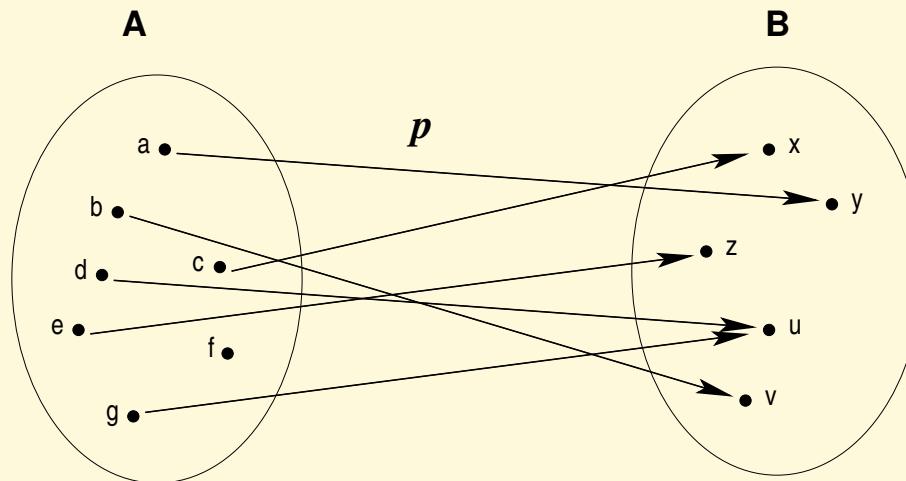


Figure 1: A partial function (*Why is it partial?*)

The word “function” unless otherwise specified is taken to mean a “total function”. Some familiar examples of partial and total functions are

1. $+$ and \times (addition and multiplication) on the natural numbers are total functions of the type $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$
2. $-$ (subtraction) on the natural numbers is a partial function of the type $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$.
3. div and mod are total functions of the type $f : \mathbb{N} \times \mathbb{P} \rightarrow \mathbb{N}$. If $a = q * b + r$ such that $0 \leq r < b$

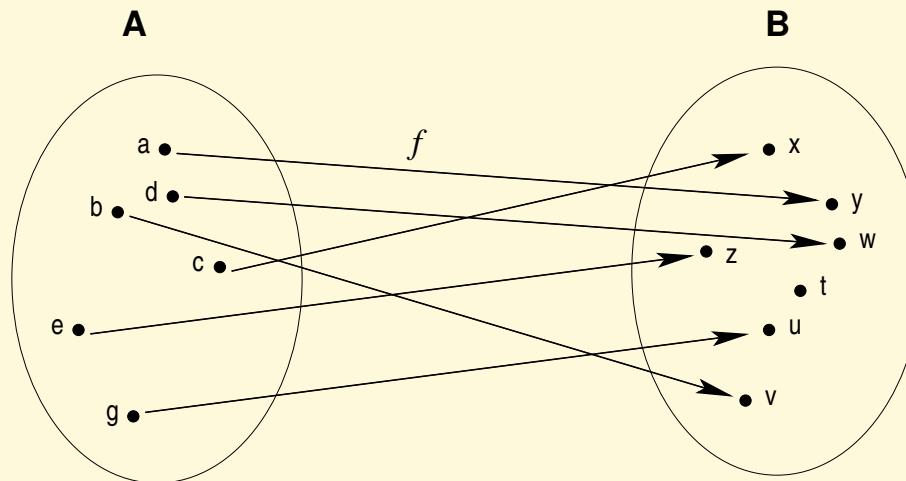


Figure 2: An injective function (*Why is it injective?*)

and $a, b, q, r \in \mathbb{N}$ then the functions div and mod are defined as $\text{div}(a, b) = q$ and $\text{mod}(a, b) = r$. We will often write these binary functions as $a * b$, $a \text{ div } b$, $a \text{ mod } b$ etc. Note that div and mod are also partial functions of the type $f : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$.

4. The binary relations $=$, \neq , $<$, \leq , $>$, \geq may also be thought of as functions of the type $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$ where $\mathbb{B} = \{\text{false}, \text{true}\}$.

Definition 0.4 Given a set A , a **finite sequence** of length $n \geq 0$ of elements from A , denoted \vec{a} , is

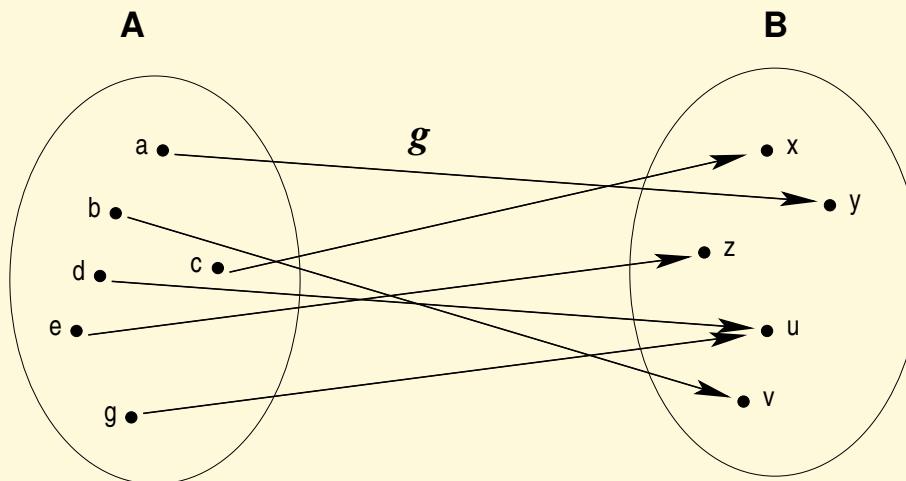


Figure 3: A surjective function (*Why is it surjective?*)

a (total) function of the type $\vec{a} : \{1, 2, \dots, n\} \rightarrow A$. We normally denote such a sequence of length n by $[a_1, a_2, \dots, a_n]$. Alternatively, \vec{a} may be regarded as a total function from $\{0, \dots, n - 1\}$ to A and may be denoted by $[a_0, a_1, \dots, a_{n-1}]$. The empty sequence, denoted $[]$, is also such a function $[] : \emptyset \rightarrow A$ and denotes a sequence of length 0.

It is very common in computer science to distinguish between the notion of a sequence and that of a string or a word.

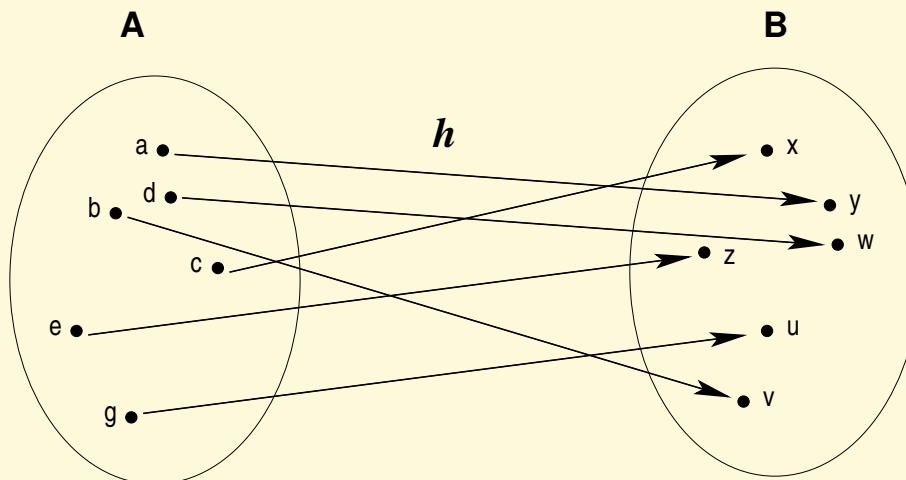


Figure 4: An bijective function (*Why is it bijective?*)

Definition 0.5 An **alphabet** is a finite set of symbols also called **letters**. Any finite sequence of letters from an alphabet is called a **string** or a **word**. A string of length $n \in \mathbb{N}$ is usually written $a_1a_2\dots a_n$ or “ $a_1a_2\dots a_n$ ”, where each $a_i \in A$, $1 \leq i \leq n$. The unique empty string (of length 0) is usually denoted ε and the operation of juxtaposing two strings s and t to form a new string is called **(con)catenation**.

It is quite clear that there exists a simple bijection from the set A^n (which is the set of all n-tuples of elements from the set A) and the set of all sequences of length n of elements from A . We will often

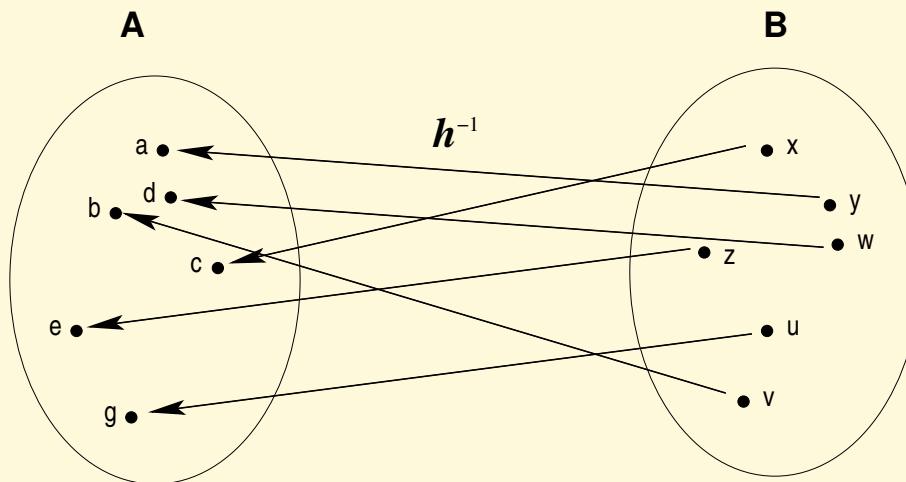


Figure 5: The inverse of the bijective function in Fig 4(*Is it bijective?*)

identify the two as being the same set even though they are actually different by definition². The set of all finite sequences of elements from A is denoted A^* , where

$$A^* = \bigcup_{n \geq 0} A^n$$

²In a programming language like ML, the difference is evident from the notation and the constructor operations for tuples and lists

The set of all *non-empty* sequences of elements from A is denoted A^+ and is defined as

$$A^+ = \bigcup_{n>0} A^n$$

An **infinite** sequence of elements from A is a total function from \mathbb{P} to A . The set of all such infinite sequences is denoted A^ω .

0.4. Operations on Binary Relations

Definition 0.6

1. Given a set A , the **identity relation** over A , denoted \mathcal{I}_A , is the set $\{(a, a) \mid a \in A\}$.
2. Given a binary relation \mathcal{R} from A to B , the **converse** of \mathcal{R} , denoted \mathcal{R}^{-1} is the relation from B to A defined as $\mathcal{R}^{-1} = \{(b, a) \mid (a, b) \in \mathcal{R}\}$.
3. Given binary relations $\mathcal{R} \subseteq A \times B$ and $\mathcal{S} \subseteq B \times C$, the **composition** of \mathcal{R} with \mathcal{S} is denoted $\mathcal{R}; \mathcal{S}$ and defined as $\mathcal{R}; \mathcal{S} = \{(a, c) \mid a \mathcal{R} b \text{ and } b \mathcal{S} c, \text{ for some } b \in B\}$.

Note that unlike in the case of functions (where for any function $f : A \rightarrow B$ its inverse $f^{-1} : B \rightarrow A$ may not always be defined), the converse of a relation is always defined. Given functions

(whether partial or total) $f : A \rightharpoonup B$ and $g : B \rightharpoonup C$, their composition is the function $g \circ f : A \rightharpoonup C$ defined simply as the relational composition $\text{graph}(f); \text{graph}(g)$. Hence $(g \circ f)(a) = g(f(a))$.

The following is an important theorem with various applications in section 0.7.

Theorem 0.7 (Schroeder-Bernstein Theorem) *Let A and B be sets and let $f : A \xrightarrow{\text{1-1}} B$ and $g : B \xrightarrow{\text{1-1}} A$ be injective functions. Then there exists a bijection between A and B .*

Proof: Since f and g are both injective (1-1), they are both total functions, but their inverses may not be total. By injectivity, for any $a \in A$, $f(a) = b$ implies that b cannot be the image under f of any other member of A . Likewise for any $b \in B$, $g(b) \in A$ and for every other $b' \in B$ we have $g(b') \neq g(b)$. Hence $f^{-1} : B \rightharpoonup A$ and $g^{-1} : A \rightharpoonup B$ are both partial functions.

For any $a_0 \in A$ we define the origin of a_0 as a_0 itself if $g^{-1}(a_0)$ is undefined i.e. if a_0 is not the image of any $b \in B$ under g . (Likewise for any $b_0 \in B$, the origin of b_0 is b_0 itself if $f^{-1}(b_0)$ is undefined). Otherwise $g^{-1}(a_0) = b_1$ for a unique $b_1 \in B$. Now consider the maximal (possibly infinite) sequence

of elements

$$\begin{array}{lll} a_0 & \in A, \\ g^{-1}(a_0) & = b_1 & \in B, \\ f^{-1}(b_1) & = a_2 & \in A, \\ g^{-1}(a_2) & = b_3 & \in B, \\ \vdots & \vdots & \vdots, \\ f^{-1}(b_{2k-1}) & = a_{2k} & \in A, \\ g^{-1}(a_{2k}) & = b_{2k+1} & \in B, \\ \vdots & \vdots & \vdots \end{array}$$

such that for each $k > 0$, $a_{2k} = f^{-1}(b_{2k-1})$ and $b_{2k+1} = g^{-1}(b_{2k})$. We then have the following cases for each $a_0 \in A$.

- *Case A_A . a_{2m} is the origin of a_0 for some $m \geq 0$.* That is, the sequence $a_0, b_1, a_2, b_3, \dots, a_{2m}$ is finite and $g^{-1}(a_{2m})$ is undefined. In this case a_{2m} is the origin of a_0 and $a_0 \in A_A$.
- *Case A_B . b_{2m+1} is the origin of a_0 for some $m \geq 0$.* That is, the sequence $a_0, b_1, a_2, b_3, \dots, a_{2m}, b_{2m+1}$ is finite and $f^{-1}(b_{2m+1})$ is undefined. Then b_{2m+1} is the origin of a_0 and $a_0 \in A_B$.
- *Case A_U . The origin of a_0 is undefined.* That is, the sequence $a_0, b_1, a_2, b_3, \dots, a_{2m}, b_{2m+1}, \dots$ is infinite. Then $a_0 \in A_U$.

Hence A may be partitioned into three (mutually disjoint) sets A_A, A_B, A_U depending upon the origins of the elements of A . (Analogously, B may be partitioned into B_A, B_B and B_U).

Now we may define the total function $h : A \longrightarrow B$ such that

$$h(a) = \begin{cases} f(a) & \text{if } a \in A_A \cup A_U \\ g^{-1}(a) & \text{if } a \in A_B \end{cases}$$

Claim. $h : A \xrightarrow[\text{onto}]{l\text{-}l} B$ i.e. h is a bijection from A to B .

⊓ The proof of the claim is easy and is left to the interested reader. In fact, we may show that the following hold

$$\begin{aligned} h(A_U) &= B_U , \quad h^{-1}(B_U) = A_U \\ h(A_A) &= B_A , \quad h^{-1}(B_A) = A_B \\ h(A_B) &= B_B , \quad h^{-1}(B_B) = A_B \end{aligned}$$

⊣

QED

■

0.5. Ordering Relations

We may define the n -fold composition of a relation \mathcal{R} on a set A by induction as follows

$$\mathcal{R}^0 = \mathcal{I}_A$$

$$\mathcal{R}^{n+1} = \mathcal{R}^n; R$$

We may combine these n -fold compositions to yield the **reflexive-transitive closure** of \mathcal{R} , denoted \mathcal{R}^* , as the relation

$$\mathcal{R}^* = \bigcup_{n \geq 0} \mathcal{R}^n$$

Sometimes it is also useful to consider merely the **transitive closure** \mathcal{R}^+ of \mathcal{R} which is defined as

$$\mathcal{R}^+ = \bigcup_{n > 0} \mathcal{R}^n$$

Definition 0.8 A binary relation \mathcal{R} on a set A is

1. **reflexive** if and only if $\mathcal{I}_A \subseteq \mathcal{R}$;
2. **irreflexive** if and only if $\mathcal{I}_A \cap \mathcal{R} = \emptyset$;

3. **symmetric** if and only if $\mathcal{R} = \mathcal{R}^{-1}$;
4. **asymmetric** if and only if $\mathcal{R} \cap \mathcal{R}^{-1} = \emptyset$;
5. **antisymmetric** if and only if $(a, b), (b, a) \in \mathcal{R}$ implies $a = b$.
6. **transitive** if and only if for all $a, b, c \in A$, $(a, b), (b, c) \in \mathcal{R}$ implies $(a, c) \in \mathcal{R}$.
7. **connected** if and only if for all $a, b \in A$, if $a \neq b$ then $a\mathcal{R}b$ or $b\mathcal{R}a$.

Given any relation \mathcal{R} on a set A , it is easy to see that \mathcal{R}^* is both reflexive and transitive.

Example 0.9

1. The edge relation on an undirected graph is an example of a symmetric relation.
2. In any directed acyclic graph the edge relation is asymmetric.
3. Consider the reachability relation on a directed graph defined as: A pair of vertices (A, B) is in the reachability relation, if either $A = B$ or there exists a vertex C such that both (A, C) and (C, B) are in the reachability relation. The reachability relation is the reflexive transitive closure of the edge relation.

4. The reachability relation on directed graphs is also an example of a relation that need not be either symmetric or asymmetric. The relation need not be antisymmetric either.

0.6. Partial Orders

Definition 0.10 A binary relation \mathcal{R} on a set A is

1. a **preorder** if it is reflexive and transitive;
2. a **strict preorder** if it is irreflexive and transitive;
3. a **partial order** if is an antisymmetric preorder;
4. a **strict partial order** if it is irreflexive, asymmetric and transitive;
5. a **linear order**³ if it is a connected partial order;
6. a **strict linear order** if it is connected, irreflexive and transitive;
7. an **equivalence** if it is reflexive, symmetric and transitive.

³also called **total order**

0.7. Infinite Sets: Countability and Uncountability

Definition 0.11 A set A is finite if it can be placed in bijection with a set $\mathbb{n} = \{0, \dots, n - 1\}$ for some $n \in \mathbb{N}$.

The above definition embodies the usual notion of counting. In particular note that the empty set \emptyset is finite since it can be placed in bijection with itself.

Definition 0.12 A set A is called infinite if there exists a bijection between A and some proper subset of itself.

This definition begs the question, “If a set is not infinite, then is it necessarily finite?”. It turns out that indeed it is. Further it is also true that if a set is not finite then it can be placed in bijection with a proper subset of itself. But rigorous proofs of these statements are beyond the scope of this course and hence we shall not pursue them.

Example 0.13 We give appropriate bijections to show that various sets are infinite. In each case, note that the codomain of the bijection is a proper subset of the domain.

1. The set \mathbb{N} of natural numbers is infinite because we can define the 1-1 correspondence p :

$$\mathbb{N} \xrightarrow[\text{onto}]{1-1} \mathbb{P}, \text{with } p(m) \stackrel{df}{=} m + 1.$$

2. The set E of even natural numbers is infinite because we have the bijection $e : E \xrightarrow[\text{onto}]{1-1} F$ where F is the set of all multiples of 4.
3. The set of odd natural numbers is infinite. (Why?)
4. The set \mathbb{Z} of integers is infinite because we have the following bijection $z : \mathbb{Z} \xrightarrow[\text{onto}]{1-1} \mathbb{N}$ by which the negative integers have unique images among the odd numbers and the non-negative integers have unique images among the even numbers. More specifically,

$$z(m) = \begin{cases} 2m & \text{if } m \in \mathbb{N} \\ -2m - 1 & \text{otherwise} \end{cases}$$

Example 0.14 The set \mathbb{R} of reals is infinite. We outline the proof by considering the nonempty open interval $(a, b) = \{p \mid a < p < b\}$ and use figure 6 as a guide to understand the mapping.

Take any line-segment \overline{AB} of length $b - a \neq 0$ and “bend” it into the semi-circle $\widehat{A'B'}$ and place it tangent to the x -axis at the point $(0, 0)$ (as shown in the figure). The bijection between the points on the semi-circle and the real numbers p , $a < p < b$ is “obvious”. This semicircle has a radius

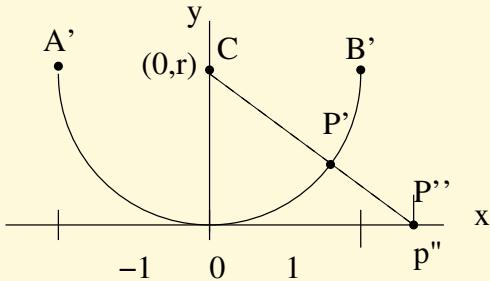


Figure 6: Bijection between the arc $A'B'$ and the real line

$r = \frac{b-a}{\pi}$. The centre C of this semi-circle is then located at the point $(0, r)$ on the 2-dimensional plane.

Consider an arbitrary point P' on the semi-circle, which corresponds to a real number p , $a < p < b$. The ray $\overrightarrow{CP'}$ intersects the x -axis at some point P'' which has the coordinates $(p'', 0)$. Since $A' \neq P' \neq B'$, the ray cannot be parallel to the x -axis). Similarly from every point P'' on the x -axis there exists a unique point P' on the semi-circle such that C , P' and P'' are collinear. Each point P' such that $A' \neq P' \neq B'$ on this semi-circle corresponds exactly to a unique real number p in the open interval (a, b) and vice-versa. Hence there exists a 1-1 correspondence between the points on the semicircle (excluding the end-points of the semi-circle) and those on the x -axis. Let p'' be the x -coordinate of the point P'' . Since the composition of bijections is a bijection (see exercises),

we may compose all these bijections to obtain a 1-1 correspondence between each p in the interval (a, b) and the real numbers.

Definition 0.15 *An infinite set is said to be **countable** (or **countably infinite**) if it can be placed in bijection with the set \mathbb{P} . Otherwise, it is said to be **uncountable**.*

The above definition essentially says that a countably infinite set may be enumerated by selecting a unique “first element”, a unique “second” element and so on. Countability of an infinite set therefore implies that for any positive integer n , it should be possible to obtain the unique designated n -th element from the set and also for any element in the set, it should be possible to obtain its position in the enumeration.

Fact 0.16 *The following are easy to prove.*

1. *An infinite set A is countable if and only if there is a bijection between A and \mathbb{N} .*
2. *Every infinite subset of \mathbb{N} is countable.*
3. *If A is a finite set and B is a countable set, then $A \cup B$ is countable.*
4. *If A and B are countable sets, then $A \cup B$ is also countable.*

Theorem 0.17 \mathbb{N}^2 is a countable set.

Proof:

We show that \mathbb{N}^2 is countably infinite by devising a way to order the elements of \mathbb{N}^2 which guarantees that there is indeed a 1-1 correspondence. For instance, an obvious ordering such as

(0, 0)	(0, 1)	(0, 2)	(0, 3)	...
(1, 0)	(1, 1)	(1, 2)	(1, 3)	...
(2, 0)	(2, 1)	(2, 2)	(2, 3)	...
:	

is not a 1-1 correspondence because we cannot answer the following questions with (unique) answers.

1. What is the n -th element in the ordering?
2. What is the position in the ordering of the pair (a, b) for arbitrary naturals a and b ?

So it is necessary to construct a more rigorous and ingenious device to ensure a bijection. So we consider the ordering implicitly defined in figure 7. By traversing the blue rays $\overrightarrow{D_0}$, $\overrightarrow{D_1}$, $\overrightarrow{D_2}$, ... in

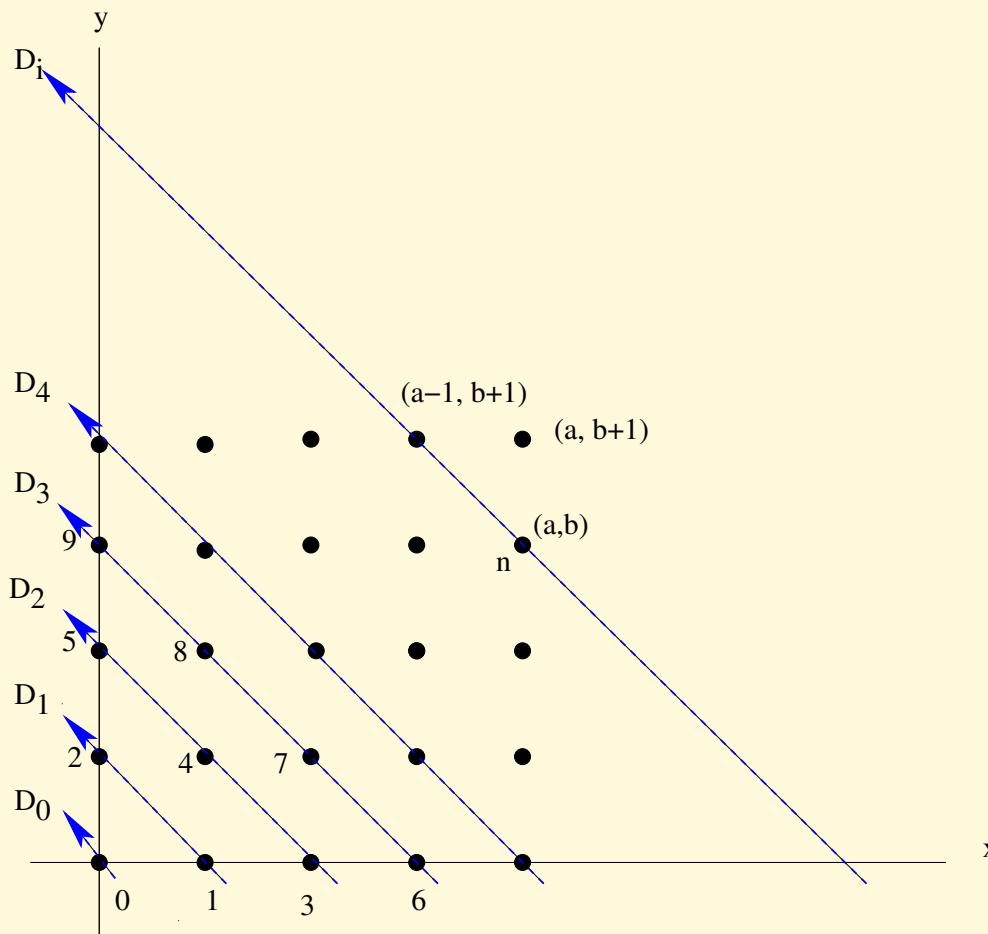


Figure 7: Counting “lattice-points” on the “diagonals”

order, we get an obvious ordering on the elements of \mathbb{N}^2 . However it should be possible to give unique answers to the above questions.

Claim $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ defined by $f(a, b) = \frac{(a + b)(a + b + 1) + 2b}{2}$ is the required bijection.

Proof outline: The function f defines essentially the traversal of the rays $\overrightarrow{D_0}, \overrightarrow{D_1}, \overrightarrow{D_2}, \dots$ in order as we shall prove. It is easy to verify that $\overrightarrow{D_0}$ contains only the pair $(0, 0)$ and $f(0, 0) = 0$. Now consider any pair $(a, b) \neq (0, 0)$. If (a, b) lies on the ray $\overrightarrow{D_i}$, then it is clear that $i = a + b$. Now consider all the pairs that lie on the rays $\overrightarrow{D_0}, \overrightarrow{D_1}, \dots, \overrightarrow{D_{i-1}}$ ⁴

The number of such pairs is given by the “triangular number”

$$i + (i - 1) + (i - 2) + \dots + 1 = \frac{i(i + 1)}{2}$$

Since we started counting from 0 this number is also the value of the lattice point $(i, 0)$ under the function f . This brings us to the starting point of the ray D_i and after crossing b lattice points along

⁴Under the usual (x, y) coordinate system, these are all the *lattice points* on and inside the right triangle defined by the three points $(i - 1, 0), (0, 0)$ and $(0, i - 1)$. A *lattice point* in the (x, y) -plane is point whose x - and y - coordinates are both integers.

the ray D_i we arrive at the point (a, b) . Hence

$$\begin{aligned} f(a, b) &= \frac{i(i+1)}{2} + b \\ &= \frac{(a+b)(a+b+1) + 2b}{2} \end{aligned}$$

We leave it as an exercise to the reader to define the inverse of this function. (*Hint: Use “triangular numbers”!*) QED ■

Theorem 0.18 *The countable union of countable sets is countable, i.e. given a family $\mathcal{A} = \{A_i \mid A_i \text{ is countable}, i \in \mathbb{N}\}$ of countable sets, their union $A_\infty = \bigcup_{i \in \mathbb{N}} A_i$ is also countable.*

Proof: For simplicity we assume that the sets are all pairwise disjoint i.e. $A_i \cap A_j = \emptyset$ for each $i \neq j$. Hence for each element $a \in A_\infty$, there exists a unique $i \in \mathbb{N}$ such that $a \in A_i$. This implies there exists a bijection $h : A_\infty \xrightarrow[\text{onto}]{1-1} \{(i, a) \mid a \in A_i, i \in \mathbb{N}\}$. Since each A_i is countable, there exists a bijection $f_i : A_i \xrightarrow[\text{onto}]{1-1} \mathbb{N}$ for each $i \in \mathbb{N}$. Define the bijection $g : A_\infty \xrightarrow[\text{onto}]{1-1} \mathbb{N}^2$ such that $g(a) = (i, f_i(a))$. By theorem 0.17 it follows that A_∞ is countable. QED ■

Example 0.19 *Let the language \mathcal{M}_0 of minimal logic be “generated” by the following process from*

a countably infinite set of “atoms” \mathbb{A} , such that \mathbb{A} does not contain any of the symbols “ \neg ”, “ \rightarrow ”, “(” and “)”).

1. $\mathbb{A} \subseteq \mathcal{M}_0$,
2. If μ and ν are any two elements of \mathcal{M}_0 then $(\neg\mu)$ and $(\mu \rightarrow \nu)$ also belong to \mathcal{M}_0 , and
3. No string other than those obtained by a finite number of applications of the above rules belongs to \mathcal{M}_0 .

We prove that the \mathcal{M}_0 is countably infinite.

Solution There are at least two possible proofs. The first one simply encodes of formulas into unique natural numbers. The second uses induction on the structure of formulas and the fact that a countable union of countable sets yields a countable set. We postpone the second proof to the chapter on induction. So here goes!

Proof: Since \mathbb{A} is countably infinite, there exists a 1 – 1 correspondence $\text{ord} : \mathbb{A} \rightarrow \mathbb{P}$ which uniquely enumerates the atoms in some order. This function may be extended to a function ord' which includes the symbols “ \neg ”, “(”, “)”, “ \rightarrow ”, such that $\text{ord}'(\neg) = 1$, $\text{ord}'(()) = 2$, $\text{ord}'(()) = 3$, $\text{ord}'(\rightarrow) = 4$, and $\text{ord}'(A) = \text{ord}(A) + 4$, for every $A \in \mathbb{A}$. Let $\text{Syms} =$

$\mathbb{A} \cup \{\neg, (\cdot, \cdot), \rightarrow\}$. Clearly $\text{ord}' : \text{Syms} \rightarrow \mathbb{P}$ is also a 1 – 1 correspondence. Hence there also exist inverse functions ord^{-1} and ord'^{-1} which for any positive integer identify a unique symbol from the domains of the two functions respectively.

Now consider any string⁵ belonging to Syms^* . It is possible to assign a unique positive integer to this string by using powers of primes. Let $p_1 = 2, p_2 = 3, \dots, p_i, \dots$ be the infinite list of primes in increasing order. Let the function $\text{encode} : \text{Syms}^* \rightarrow \mathbb{P}$ be defined by induction on the lengths of the strings in Syms^* , as follows. Assume $s \in \text{Syms}^*$, $a \in \text{Syms}$ and “” denotes the empty string.

$$\begin{aligned}\text{encode}(\text{""}) &= 1 \\ \text{encode}(sa) &= \text{encode}(s) \times p_m^{\text{ord}'(a)}\end{aligned}$$

where s is a string of length $m - 1$ for $m \geq 1$.

It is now obvious from the unique prime-factorization of positive integers that every string in Syms^* has a unique positive integer as its “encoding” and from any positive integer it is possible to get the unique string that it represents. Hence Syms^* is a countably infinite set. Since the language of minimal logic is a subset of the Syms^* it cannot be an uncountable. Hence there are only two possibilities: either it is finite or it is countably infinite.

Claim. The language of minimal logic is not finite.

⁵This includes even arbitrary strings which are not part of the language. For example, you may have strings such as “) \neg (”.

Proof of claim. Suppose the language were finite. Then there exists a formula ϕ in the language such that $\text{encode}(\phi)$ is the maximum possible positive integer. This $\phi \in \text{Syms}^*$ and hence is a string of the form $a_1 \dots a_m$ where each $a_i \in \text{Syms}$. Clearly

$$\text{encode}(\phi) = \prod_{i=1}^m p_i^{\text{ord}'(a_i)}$$

. Now consider the longer formula $\psi = (\neg\phi)$. It is easy to show that

$$\text{encode}(\psi) = 2^{\text{ord}'("(")} \times 3^{\text{ord}'("\neg")} \times \prod_{i=1}^m p_{i+2}^{\text{ord}'(a_i)} \times p_{m+3}^{\text{ord}'(")")}$$

and $\text{encode}(\psi) > \text{encode}(\phi)$ contradicting the assumption of the claim.

Hence the language is countably infinite. QED

Not all infinite sets that can be constructed are countable. In other words even among infinite sets there are some sets that are “more infinite than others”. The following theorem and the form of its proof was first given by Georg Cantor and has been used to prove several results in logic, mathematics and computer science.

Theorem 0.20 (Cantor's diagonalization). *The powerset of \mathbb{N} (i.e. $2^{\mathbb{N}}$, the set of all subsets of \mathbb{N}) is an uncountable set.*

Proof: Firstly, it should be clear that $2^{\mathbb{N}}$ is not a finite set, since it can be placed in bijection with $2^{\mathbb{P}}$ which is a proper subset of $2^{\mathbb{N}}$.

Consider any subset $A \subseteq \mathbb{N}$. We may represent this set as an infinite sequence σ_A composed of 0's and 1's such that $\sigma_A(i) = 1$ if $i \in A$, otherwise $\sigma_A(i) = 0$. Let $\Sigma = \{\sigma \mid \text{for each } i \in \mathbb{N}, \sigma(i) \in \{0, 1\}\}$ be the set of all such sequences. It is easy to show that there exists a bijection $g : 2^{\mathbb{N}} \xrightarrow[\text{onto}]{1-1} \Sigma$ such that $g(A) = \sigma_A$, for each $A \subseteq \mathbb{N}$. Clearly, therefore $2^{\mathbb{N}}$ is countable if and only if Σ is countable. Hence, if there exists a bijection $f : \Sigma \xrightarrow[\text{onto}]{1-1} \mathbb{N}$, then $g \circ f$ is the required bijection from $2^{\mathbb{N}}$ to \mathbb{N} . On the other hand, if there is no bijection f then $2^{\mathbb{N}}$ is uncountable if and only if Σ is uncountable. We make the following claim which we prove by Cantor's diagonalization.

Claim. The set Σ is uncountable.

We prove the claim as follows. Suppose Σ is countable then there exists a bijection $h : \mathbb{N} \xrightarrow[\text{onto}]{1-1} \Sigma$. In fact let $h(i) = \sigma_i \in \Sigma$, for each $i \in \mathbb{N}$. Now consider the sequence ρ constructed in such a manner

that for each $i \in \mathbb{N}$, $\rho(i) \neq \sigma_i(i)$. In other words,

$$\rho(i) = \begin{cases} 0 & \text{if } \sigma_i(i) = 1 \\ 1 & \text{if } \sigma_i(i) = 0 \end{cases}$$

Since ρ is an infinite sequence of 0's and 1's, $\rho \in \Sigma$. But from the above construction it follows that since ρ is different from every sequence in Σ it cannot be a member of Σ , leading to a contradiction. Hence the assumption that the bijection h exists is wrong. Hence the assumption that Σ is countable must be wrong. QED ■

It is possible to generalize the above theorem and the proof to all powersets as follows.

Theorem 0.21 (The Powerset theorem). *There is no 1-1 correspondence between a set and its powerset.*

Proof: Let A be any set and let $\mathcal{P}(A)$ be its powerset. Assume that $g : A \rightarrow \mathcal{P}(A)$ is a 1-1 correspondence between A and $\mathcal{P}(A)$. This implies for every $a \in A$, $g(a) \subseteq A$ is uniquely determined and further for each $B \subseteq A$, $g^{-1}(B)$ exists and is uniquely determined.

For any $a \in A$, a is called an *interior* member if $a \in g(a)$ and otherwise a is an *exterior* member. Consider the set

$$X = \{x \in A \mid x \notin g(x)\}$$

which consists of exactly the exterior members of A . Since g is a 1-1 correspondence, there exists a unique $x \in A$ such that $X = g(x)$. Note that X could be the empty set.

x is either an interior member or an exterior member. If x is an interior member then $x \in g(x) = X$ which contradicts the assumption that X contains only exterior members. If x is an exterior member then $x \notin g(x) = X$. But then since x is an exterior member $x \in X$, which is a contradiction. Hence the assumption that there exists a 1-1 correspondence g between A and \mathcal{P}^A must be false. QED ■

Example 0.22 We show using the Schroeder-Bernstein theorem 0.7 that there exists a bijection between the sets $\mathcal{P}^{\mathbb{P}}$ and the real closed-open interval $[0, 1)$. We construct two injective mappings $f : \mathcal{P}^{\mathbb{P}} \xrightarrow{\text{i-i}} [0, 1)$ and $g : [0, 1) \xrightarrow{\text{i-i}} \mathcal{P}^{\mathbb{P}}$ as follows: For any $A \subseteq \mathbb{P}$ let $f(A) = 0.d_1d_2d_3\dots$ such that $d_i = 1$ if $i \in A$ and $d_i = 2$ otherwise. Clearly for every A there exists a unique image in $[0, 1)$ and no two distinct subsets of \mathbb{P} would have identical images. Hence f is injective.

To define g we consider only normal binary representations of real numbers. That is, we consider only binary representations which do not have an infinite sequence of trailing 1s, since any num-

ber of the form $0.b_1b_2\dots b_{i-1}0\bar{1}$ equals the real number $0.b_1b_2\dots b_{i-1}1\bar{0}$ which is normal. Every real number in $[1, 0)$ has a unique normal representation. Now consider the function defined by $g(0.b_1b_2b_3\dots) = \{i \in P \mid b_i = 1\}$. g is clearly a well-defined function and it is injective as well. Hence they are both uncountable sets.

Exercise 0.1

1. Find the fallacy in the proof of the following purported theorem.

Theorem: If $x = y$ then $2 = 1$.

Proof:

1.	$x = y$	<i>Given</i>
2.	$x^2 = xy$	<i>Multiply both sides by x</i>
3.	$x^2 - y^2 = xy - y^2$	<i>Subtract y^2 from both sides</i>
4.	$(x + y)(x - y) = y(x - y)$	<i>Factorize</i>
5.	$x + y = y$	<i>Cancel out $(x - y)$</i>
6.	$2y = y$	<i>Substitute x for y, by equation 1.</i>
7.	$2 = 1$	<i>Divide both sides by y</i>

QED

2. Prove that if $A \subseteq B$ then $\mathcal{P}^A \subseteq \mathcal{P}^B$.

3. Prove that for any binary relations \mathcal{R} and \mathcal{S} on a set A ,

- (a) $(\mathcal{R}^{-1})^{-1} = \mathcal{R}$
- (b) $(\mathcal{R} \cap \mathcal{S})^{-1} = \mathcal{R}^{-1} \cap \mathcal{S}^{-1}$
- (c) $(\mathcal{R} \cup \mathcal{S})^{-1} = \mathcal{R}^{-1} \cup \mathcal{S}^{-1}$
- (d) $(\mathcal{R} - \mathcal{S})^{-1} = \mathcal{R}^{-1} - \mathcal{S}^{-1}$

4. Prove that the composition operation on relations is associative. Give an example of the composition of relations to show that relational composition is not commutative.

5. Prove that the composition of bijections is a bijection. That is, prove that for any bijective total functions $f : A \xrightarrow[\text{onto}]{1-1} B$ and $g : B \xrightarrow[\text{onto}]{1-1} C$, their composition is the function $g \circ f : A \xrightarrow[\text{onto}]{1-1} C$.

6. Is the composition of injective functions also injective? Is the composition of surjective functions also surjective? Prove or disprove the two statements.

7. Prove that the inverse of a bijective function is also a bijective function.

8. Prove that for any binary relations \mathcal{R} , \mathcal{R}' from A to B and \mathcal{S} , \mathcal{S}' from B to C , if $\mathcal{R} \subseteq \mathcal{R}'$ and $\mathcal{S} \subseteq \mathcal{S}'$ then $\mathcal{R}; \mathcal{S} \subseteq \mathcal{R}'; \mathcal{S}'$

9. Prove or disprove⁶ that relational composition satisfies the following distributive laws for relations, where $\mathcal{R} \subseteq A \times B$ and $\mathcal{S}, \mathcal{T} \subseteq B \times C$.

- (a) $\mathcal{R}; (\mathcal{S} \cup \mathcal{T}) = (\mathcal{R}; \mathcal{S}) \cup (\mathcal{R}; \mathcal{T})$
- (b) $\mathcal{R}; (\mathcal{S} \cap \mathcal{T}) = (\mathcal{R}; \mathcal{S}) \cap (\mathcal{R}; \mathcal{T})$
- (c) $\mathcal{R}; (\mathcal{S} - \mathcal{T}) = (\mathcal{R}; \mathcal{S}) - (\mathcal{R}; \mathcal{T})$

10. Prove that for $\mathcal{R} \subseteq A \times B$ and $\mathcal{S} \subseteq B \times C$, $(\mathcal{R}; \mathcal{S})^{-1} = (\mathcal{S}^{-1}); (\mathcal{R}^{-1})$.

11. Show that a relation \mathcal{R} on a set A is

- (a) antisymmetric if and only if $\mathcal{R} \cap \mathcal{R}^{-1} \subseteq \mathcal{I}_A$
- (b) transitive if and only if $\mathcal{R}; \mathcal{R} \subseteq \mathcal{R}$
- (c) connected if and only if $(A \times A) - \mathcal{I}_A \subseteq \mathcal{R} \cup \mathcal{R}^{-1}$

12. Consider any reflexive relation \mathcal{R} on a set A . Does it necessarily follow that A is not asymmetric? If \mathcal{R} is asymmetric does it necessarily follow that it is irreflexive?

13. Prove that

- (a) \mathbb{N}^n , for any $n > 0$ is a countably infinite set,

⁶that is, find an example of appropriate relations which actually violate the equality

(b) If $\{A_i | i \geq 0\}$ is a countable collection of pair-wise disjoint sets (i.e. $A_i \cap A_j = \emptyset$ for all $i \neq j$) then $A = \bigcup_{i \geq 0} A_i$ is also a countable set.

(c) \mathbb{N}^* the set of all finite sequences of natural numbers is countable.

14. Prove that

(a) \mathbb{N}^ω the set of all infinite sequences of natural numbers is uncountable,

(b) the set of all binary relations on a countably infinite set is an uncountable set,

(c) the set of all total functions from \mathbb{N} to \mathbb{N} is uncountable.

15. Prove that there exists a bijection between the set $2^{\mathbb{N}}$ and the open interval $(0, 1)$ of real numbers.

What can you conclude about the cardinality of the set $2^{\mathbb{N}}$ in relation to the set \mathbb{R} ?

16. Prove that the composition operation on relations is associative. Give an example of the composition of relations to show that relational composition is not commutative in general.

17. Consider any reflexive relation \mathcal{R} on a set A . Does it necessarily follow that \mathcal{R} is not asymmetric? If \mathcal{R} is asymmetric does it necessarily follow that it is irreflexive?

18. Prove that for any relation \mathcal{R} on a set A ,

(a) $\mathcal{S} = \mathcal{R}^* \cup (\mathcal{R}^*)^{-1}$ and $\mathcal{T} = (\mathcal{R} \cup \mathcal{R}^{-1})^*$ are both equivalence relations.

- (b) Prove or disprove: $\mathcal{S} = \mathcal{T}$.
19. Given any preorder \mathcal{R} on a set A , prove that the kernel of the preorder defined as $\mathcal{R} \cap \mathcal{R}^{-1}$ is an equivalence relation.
20. Consider any preorder \mathcal{R} on a set A . We give a construction of another relation as follows. For each $a \in A$, let $[a]_{\mathcal{R}}$ be the set defined as $[a]_{\mathcal{R}} = \{b \in A \mid a\mathcal{R}b \text{ and } b\mathcal{R}a\}$. Now consider the set $B = \{[a]_{\mathcal{R}} \mid a \in A\}$. Let \mathcal{S} be a relation on B such that for every $a, b \in A$, $[a]_{\mathcal{R}}\mathcal{S}[b]_{\mathcal{R}}$ if and only if $a\mathcal{R}b$. Prove that \mathcal{S} is a partial order on the set B .
21. For any two sets A and B , $A \preceq B$ if there exists an injective function $f : A \xrightarrow{1-1} B$.
- (a) Prove that \preceq is a preorder on any collection of sets.
- (b) Prove that any bijection between sets defines an equivalence relation on the collection of sets.

0.8. Induction Principles

Theorem: All natural numbers are equal.

Proof: Given a pair of natural numbers a and b , we prove they are equal by performing complete induction on the maximum of a and b (denoted $\max(a, b)$).

Basis. For all natural numbers less than or equal to 0, the claim holds.

Induction hypothesis. For any a and b such that $\max(a, b) \leq k$, for some natural $k \geq 0$, $a = b$.

Induction step. Let a and b be naturals such that $\max(a, b) = k + 1$. It follows that $\max(a - 1, b - 1) = k$. By the induction hypothesis $a - 1 = b - 1$. Adding 1 on both sides we get $a = b$ QED.

Fortune cookie on Linux

0.9. Mathematical Induction

Anyone who has had a good background in school mathematics must be familiar with two uses of induction.

1. definition of functions and relations by mathematical induction, and

2. proofs by the principle of mathematical induction.

Example 0.23 We present below some familiar examples of definitions by mathematical induction.

1. The factorial function on natural numbers is defined as follows.

Basis. $0! = 1$

Induction step. $(n + 1)! = n! \times (n + 1)$

2. The n -th power (where n is a natural number) of a real number x is often defined as

Basis. $x^1 = x$

Induction step. $x^{n+1} = x^n \times x$

3. For binary relations R, S on A we define their composition (denoted $R; S$) as follows.

$$R; S = \{(a, c) \mid \text{for some } b \in A, (a, b) \in R \text{ and } (b, c) \in S\}$$

We may extend this binary relational composition to an n -fold composition of a single relation R as follows.

Basis. $R^1 = R$

Induction step. $R^{n+1} = R; R^n$

Similarly the principle of mathematical induction is the means by which we have often *proved* (as opposed to *defining*) properties about numbers, or statements involving the natural numbers. The principle may be stated as follows.

Principle of Mathematical Induction – Version 1

A property P holds for all natural numbers provided

Basis. *P holds for 0, and*

Induction step. *For arbitrarily chosen $n > 0$,*

P holds for $n - 1$ implies P holds for n .

The underlined portion, called the **induction hypothesis**, is an assumption that is necessary for the conclusion to be proved. Intuitively, the principle captures the fact that in order to prove any statement involving natural numbers, it suffices to do it in two steps. The first step is the basis and

needs to be proved. The proof of the induction step essentially tells us that the reasoning involved in proving the statement for all other natural numbers is the same. Hence instead of an infinitary proof (one for each natural number) we have a compact finitary proof which exploits the similarity of the proofs for all the naturals except the basis.

Example 0.24 *We prove that all natural numbers of the form $n^3 + 2n$ are divisible by 3.*

Proof:

Basis. For $n = 0$, we have $n^3 + 2n = 0$ which is divisible by 3.

Induction step. Assume for an arbitrarily chosen $n \geq 0$, $n^3 + 2n$ is divisible by 3. Now consider $(n + 1)^3 + 2(n + 1)$. We have

$$\begin{aligned}(n + 1)^3 + 2(n + 1) &= (n^3 + 3n^2 + 3n + 1) + (2n + 2) \\ &= (n^3 + 2n) + 3(n^2 + n + 1)\end{aligned}$$

which clearly is divisible by 3.

QED

Several versions of this principle exist. We state some of the most important ones. In such cases, the underlined portion is the induction hypothesis. For example it is not necessary to consider 0 (or even 1) as the basis step. Any integer k could be considered the basis, as long as the property is to be proved for all $n \geq k$.

Principle of Mathematical Induction – Version 2

A property P holds for all natural numbers $n \geq k$ for some natural number k , provided

Basis. P holds for k , and

Induction step. For arbitrarily chosen $n > k$,

P holds for $n - 1$ implies P holds for n .

Such a version seems very useful when the property to be proved is either not true or is undefined for all naturals less than k . The following example illustrates this.

Example 0.25 Every positive integer $n \geq 8$ is expressible as $n = 3i + 5j$ where $i, j \geq 0$.

Proof: .

Basis. For $n = 8$, we have $n = 3 + 5$, i.e. $i = j = 1$.

Induction step. Assuming for an arbitrary $n \geq 8$, $n = 3i + 5j$ for some naturals i and j , consider $n + 1$. If $j = 0$ then clearly $i \geq 3$ and we may write $n + 1$ as $3(i - 3) + 5(j + 2)$. Otherwise $n + 1 = 3(i + 2) + 5(j - 1)$.

QED

However it is not necessary to have this new version of the Principle of mathematical induction at all as the following reworking of the previous example shows.

Example 0.26 The property of the previous example could be equivalently reworded as follows.

“For every natural number n , $n + 8$ is expressible as $n + 8 = 3i + 5j$ where $i, j \geq 0$ ”.

Proof: .

Basis. For $n = 0$, we have $n + 8 = 8 = 3 + 5$, i.e. $i = j = 1$.

Induction step. Assuming for an arbitrary $n \geq 0$, $n + 8 = 3i + 5j$ for some naturals i and j , consider $n + 1$. If $j = 0$ then clearly $i \geq 3$ and we may write $(n + 1) + 8$ as $3(i - 3) + 5(j + 2)$. Otherwise $(n + 1) + 8 = 3(i + 2) + 5(j - 1)$.

QED

In general any property P that holds for all naturals greater than or equal to some given k may be transformed equivalently into a property Q , which reads exactly like P except that all occurrences of “ n ” in P are systematically replaced by “ $n + k$ ”. We may then prove the property Q using the first version of the principle.

What we have stated above informally is, in fact a proof outline of the following theorem.

Theorem 0.27 *The two principles of mathematical induction are equivalent. In other words, every application of PMI - version 1 may be transformed into an application of PMI – version 2 and vice-versa.*

In the sequel we will assume that the principle of mathematical induction always refers to the first version.

0.10. Complete Induction

Often in inductive definitions and proofs it seems necessary to work with an inductive hypothesis that includes not just the predecessor of a natural number, but some or all of their predecessors as

well.

Example 0.28 *The definition of the following sequence is a case of precisely such a definition where the function $F(n)$ is defined for all naturals as follows.*

Basis. $F(0) = 0$

Induction step

$$F(n+1) = \begin{cases} 1 & \text{if } n = 0 \\ F(n) + F(n-1) & \text{otherwise} \end{cases}$$

This is the famous Fibonacci⁷ sequence.

One of the properties of the Fibonacci sequence is that the sequence converges to the “golden ratio”⁸. For any inductive proof of properties of the Fibonacci numbers, we would clearly need to assume that the property holds for the two preceding numbers in the sequence.

In the following, we present a principle that assumes a stronger induction hypothesis. And hence the principle itself seems “weaker” than the previous versions.

⁷named after Leonardo of Fibonacci.

⁸one of the solutions of the equation $x^2 = x + 1$. It was considered an aesthetically pleasing aspect ratio for buildings in ancient Greek architecture.

Principle of Complete Induction (PCI)

A property P holds for all natural numbers provided

Basis. P holds for 0.

Induction step. For an arbitrary $n > 0$

P holds for every m , $0 \leq m < n$ implies P holds for n

Example 0.29 Let $F(0) = 0$, $F(1) = 1$, $F(2) = 1$, \dots , $F(n+1) = F(n) + F(n-1)$, \dots be the Fibonacci sequence. Let ϕ be the “golden ratio” $(1 + \sqrt{5})/2$. We now show that the property $F(n+1) \leq \phi^n$ holds for all n .

Proof: By the principle of complete induction on n .

Basis. For $n = 0$, we have $F(1) = \phi^0 = 1$.

Induction step. Assuming the property holds for all m , $0 \leq m \leq n - 1$, for an arbitrarily chosen

$n > 0$, we need to prove that $F(n + 1) \leq \phi^n$.

$$\begin{aligned} F(n + 1) &= F(n) + F(n - 1) \\ &\leq \phi^{n-1} + \phi^{n-2} && \text{by the induction hypothesis} \\ &= \phi^{n-2}(\phi + 1) \\ &= \phi^n && \text{since } \phi^2 = \phi + 1 \end{aligned}$$

QED

Note that the feature distinguishing the principle of mathematical induction from that of complete induction is the induction hypothesis. It appears to be much stronger in the latter case. However, in the following example we again prove the property in example 0.29 but this time we use the principle of mathematical induction instead.

Example 0.30 Let $P(n)$ denote the property

$$“F(n + 1) \leq \phi^n.”$$

Rather than prove the original statement “For all n , $P(n)$ ” we instead consider the property $Q(n)$ which we define as

“For every m , $0 \leq m \leq n$, $\mathbf{P}(m)$.”

and prove the statement “For all n , $\mathbf{Q}(n)$ ”. This property can now be proved by mathematical induction as follows. The reader is encouraged to study the following proof carefully.

Proof: By the principle of mathematical induction on n .

Basis. For $n = 0$, we have $F(1) = \phi^0 = 1$.

Induction step. Assuming the property $\mathbf{Q}(n - 1)$, holds for an arbitrarily chosen $n > 0$, we need to prove the property \mathbf{Q} for n . But for this it suffices to prove the property \mathbf{P} for n , since $\mathbf{Q}(n)$ is equivalent to the conjunction of $\mathbf{Q}(n - 1)$ and $\mathbf{P}(n)$. Hence we prove the property $\mathbf{P}(n)$.

$$\begin{aligned} F(n + 1) &= F(n) + F(n - 1) \\ &\leq \phi^{n-1} + \phi^{n-2} \quad \text{by the induction hypothesis} \\ &= \phi^{n-2}(\phi + 1) \\ &= \phi^n \quad \text{since } \phi^2 = \phi + 1 \end{aligned}$$

QED

The above example shows quite clearly that the induction hypothesis used in any application of complete induction though seemingly stronger, can also lead to the proof of seemingly stronger

properties. But in fact, in the end the proofs are almost identical. These proofs lead us then naturally into the next theorem.

Theorem 0.31 *The two principles of mathematical induction are equivalent. In other words, every application of PMI may be transformed into an application of PCI and vice-versa.*

Proof: We need to prove the following two claims.

1. *Any proof of a property using the principle of mathematical induction, is also a proof of the same property using the principle of complete induction.* This is so because the only possible change in the nature of two proofs could be because they use different induction hypotheses. Since the proof by mathematical induction uses a fairly weak assumption which is sufficient to prove the property, strengthening it in any way does not need to change the rest of the proof of the induction step.
2. *For every proof of a property using the principle of complete induction, there exists a corresponding proof of the same property using the principle of mathematical induction.* To prove this claim we resort to the same trick employed in example 0.29. We merely replace each occurrence of the original property in the form $P(n)$ by $Q(n)$, where the property Q is defined as

“For every m , $0 \leq m \leq n$, $\mathbf{P}(m)$.”

Since $\mathbf{Q}(0)$ is the same as $\mathbf{P}(0)$ there is no other change in the basis step of the proof. In the original proof by complete induction the induction hypothesis would have read

For arbitrarily chosen $n > 0$, for all m , $0 \leq m \leq n - 1$, $\mathbf{P}(m)$

whereas in the new proof by mathematical induction the induction hypothesis would read

For arbitrarily chosen $n > 0$, $\mathbf{Q}(n - 1)$

Clearly the two induction hypotheses are logically equivalent. Hence the rest of the proof of the induction step would suffer no other change. The basis step and the induction step would together constitute a proof by *mathematical* induction of the property \mathbf{Q} for all naturals n . Since $\mathbf{Q}(n)$ logically implies $\mathbf{P}(n)$ it follows that the proof of property \mathbf{P} for all naturals has been done by *mathematical* induction.

QED

The natural numbers are themselves defined as the smallest set \mathbb{N} such that $0 \in \mathbb{N}$ and whenever $n \in \mathbb{N}$, $n + 1$ also belongs to \mathbb{N} . Therefore we may state yet another version of PMI from which the other versions previously stated may be derived. The intuition behind this version is that a property

P may also be considered as defining a set $S = \{x \mid x \text{ satisfies property } P\}$. Therefore if a property P is true for all natural numbers the set defined by the property must be the set of natural numbers. This gives us the last version of the principle of mathematical induction.

Principle of Mathematical Induction – Version 0

A set $S = \mathbb{N}$ provided

Basis. $0 \in S$, and

Induction step. For arbitrarily chosen $n > 0$,

$n - 1 \in S$ implies $n \in S$.

We end this section with an example of the use of induction to prove that for any $n \in \mathbb{N}$, the set of all n -tuples of natural numbers is only countably infinite.

Example 0.32 Assume there exists a 1-1 correspondence $f_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$. Use this fact to prove by induction on n , that there exists a 1-1 correspondence $f_n : \mathbb{N}^n \rightarrow \mathbb{N}$, for all $n \geq 2$.

Solution. In general, to prove that a given function $F : A \rightarrow B$ is a 1-1 correspondence, we may prove it by contradiction. Then there are 2 cases to consider.

1. F is non-injective. Then there exist elements $a, a' \in A$, such that $a \neq a'$ and $F(a) = F(a')$.
2. F is non-surjective. Then there exists an element $b \in B$ such that $F(a) \neq b$, for any $a \in A$.

It is also easy to show that if $F : A \rightarrow B$ and $G : B \rightarrow C$ are both bijections then their composition $G \circ F : A \rightarrow C$ is also a bijection.

We now proceed to prove by induction on n .

Basis. For $n = 2$ it is given that f_2 is a bijection.

Induction step. Assume the induction hypothesis,

For some $n \geq 2$ there exists a bijection $f_n : \mathbb{N}^n \rightarrow \mathbb{N}$.

We need to prove that there exists a 1-1 correspondence (bijection) between \mathbb{N}^{n+1} and \mathbb{N} . We prove this by constructing a function $f_{n+1} : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$.

Let $g : \mathbb{N}^{n+1} \rightarrow (\mathbb{N}^n \times \mathbb{N})$ be the function defined by

$$g(x_1, \dots, x_n, x_{n+1}) = ((x_1, \dots, x_n), x_{n+1})$$

Claim: g is a 1-1 correspondence. The proof is trivial.

Let $h : \mathbb{N}^{n+1} \rightarrow (\mathbb{N} \times \mathbb{N})$ be defined by

$$h(x_1, \dots, x_n, x_{n+1}) = (f_n(x_1, \dots, x_n), x_{n+1})$$

Claim: h is a 1-1 correspondence. It can be proved from the fact that $f_n : \mathbb{N}^n \rightarrow \mathbb{N}$ is a bijection.

Since f_2 is also a bijection, it follows that the composition of h and f_2 , viz. $f_2 \circ h : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ is also a bijection. Hence f_{n+1} defined as $f_{n+1} \stackrel{df}{=} f_2 \circ h$, i.e.

$$f_{n+1}((x_1, \dots, x_n, x_{n+1})) = f_2(h((x_1, \dots, x_n, x_{n+1})))$$

is a bijection. ■

Note.

1. Many people assume automatically that $\mathbb{N}^{n+1} = \mathbb{N}^n \times \mathbb{N}$ or $\mathbb{N}^{n+1} = \mathbb{N} \times \mathbb{N}^n$. But while it is true

that there exists a bijection between \mathbb{N}^{n+1} and $\mathbb{N}^n \times \mathbb{N}$, they are not equal as sets. Hence we have defined the function g though it is not really necessary for the proof.

- Very often countability is assumed by people and they try to argue that since the sets are countable there should be a bijection. But it should be clear that establishing a bijection is necessary first to prove that the required sets are countable. In fact the aim of this problem is to construct a bijection to prove that the sets \mathbb{N}^n are all countable.*

0.11. Structural Induction

In many cases such as the syntactic definitions of programming languages, their semantics and the construction of “recursive” data types in languages such as ML and Java, it is helpful to consider a form of induction called structural induction. This form of induction enables us to prove fairly general properties about the datatypes so constructed and is a convenient tool for defining functions and proving properties about data types and programs.

Definition 0.33 *Let \mathbb{U} be a set called the **Universe**, B a nonempty subset of \mathbb{U} called the **basis**, and let K called the **constructor** set be a nonempty set of functions, such that each function $f \in K$ has associated with it a unique natural number $n \geq 0$ called its **arity** and denoted by $\alpha(f)$. If a function*

f has an arity of $n \geq 0$, then $f : \mathbb{U}^n \rightarrow \mathbb{U}$. Let \mathcal{X} be the family of subsets of \mathbb{U} such that each $X \in \mathcal{X}$ satisfies the following conditions.

Basis. $B \subseteq X$

Induction step. *if $f \in K$ is of arity $n \geq 0$, $a_1, \dots, a_n \in X$ then $f(a_1, \dots, a_n) \in X$*

A set A is said to be **inductively defined** from B , K , \mathbb{U} if it is the smallest set (under the subset ordering \subseteq) satisfying the above conditions i.e.

$$A = \bigcap_{X \in \mathcal{X}} X \quad (1)$$

The set A is also said to have been **generated** from the basis B and **the rules of generation** $f \in K$.

As in the other induction principles the underlined portion is the **induction hypothesis**. It may not be absolutely clear whether A defined as in (1) satisfies the two conditions of definition 0.33.

Lemma 0.34 $A \in \mathcal{X}$ where A and \mathcal{X} are as in definition 0.33.

Proof: We need to show that A satisfies the two conditions that each $X \in \mathcal{X}$ satisfies. It is easy to see that since $B \subseteq X$ for each $X \in \mathcal{X}$, we have $B \subseteq A$ and hence A does satisfy the

basis condition. As for the induction step, consider any $f \in K$ and elements $a_1, \dots, a_{\alpha(f)} \in A$. By equation (1) we have $a_1, \dots, a_{\alpha(f)} \in X$ for every $X \in \mathcal{X}$. Therefore $f(a_1, \dots, a_{\alpha(f)}) \in X$ for every $X \in \mathcal{X}$ which implies $f(a_1, \dots, a_{\alpha(f)}) \in A$. Hence $A \in \mathcal{X}$. QED ■

We may also think of A as the smallest set (under the subset ordering) which satisfies the set equation

$$X = B \cup \bigcup \{f(X^n) \mid f \in K, \alpha(f) = n \geq 0, X \subseteq \mathbb{U}\} \quad (2)$$

in the unknown X , where $f(X^n) = \{a \mid a = f(a_1, \dots, a_n), \text{ for some } (a_1, \dots, a_n) \in X^n\}$. We show in lemma 0.37 that such equations may be solved for their unique smallest solution.

Definition 0.35 Let \mathbb{U}, B, K be as in definition 0.33. Then a sequence $[a_1, \dots, a_m]$ of elements of \mathbb{U} is called a **construction sequence** for a_m if for all $i = 1, \dots, m$ either $a_i \in B$ or there exists a constructor $f \in K$, of arity $n > 0$, and $0 < i_1, \dots, i_n < i$ such that $f(a_{i_1}, \dots, a_{i_n}) = a_i$. a_i is said to **directly depend** on each of the elements a_{i_1}, \dots, a_{i_n} (denoted $a_{i_j} \prec^1 a_i$ for each $j \in \{1, \dots, n\}$). a_i **depends** on a_j , denoted $a_j \prec a_i$ if either $a_j \prec^1 a_i$ or there exists some i' such that $a_j \prec a_{i'}$ and $a_{i'} \prec^1 a_i$.

A contains exactly all those elements of \mathbb{U} which have a construction sequence. The basis along with the constructor functions are said to define the terms generated by the rules of construction of definition 0.33.

Example 0.36 Consider the following definition of a subclass of arithmetic expressions, called am-expressions generated only from natural numbers and the addition and multiplication operations. The rules may be expressed in English as follows.

Basis Every natural number is an am-expression.

Induction step

addition If e and e' are am-expressions then $\text{add}(e, e')$ is an am-expression.

multiplication If e and e' are am-expressions then $\text{mult}(e, e')$ is an am-expression.

Initiality Only strings that are obtained by a finite number of applications of the above rules are am-expressions (nothing else is an am-expression).

In the above definition of am-expressions \mathbb{N} is the basis, $K = \{\text{add}, \text{mult}\}$ is the set of constructors each of arity 2 and the universe \mathbb{U} consists of all possible finite sequences of symbols drawn from the natural numbers and applications of the constructors. The smallest set generated from finite sequences of applications of the basis and induction steps is the set of am-expressions involving only the naturals and the 2-ary constructors add and mult such that every application of a constructor has exactly two operands each of which in turn is either a natural number or constructed in a

similar fashion. “0”, “add(0, 1)”, “mult(add(1, 0), mult(1, 1))” are all am-expressions. On the other hand,

1. “add(0)” is not an am-expression since the arity of add is 2,
2. “0, 1” is not an am-expression since it is not a natural number (it is actually a sequence of two natural numbers),
3. “mult(∞ , 0)” is not an am-expression since ∞ is not a natural.

The am-expression “mult(add(1, 0), mult(1, 1))” has the following possible construction sequences.

1. [1, 0, add(1, 0), 1, 1, mult(1, 1), mult(add(1, 0), mult(1, 1))]
2. [1, 0, add(1, 0), mult(1, 1), mult(add(1, 0), mult(1, 1))] where replications of am-expressions have been omitted.
3. [1, 0, mult(1, 1), add(1, 0), mult(add(1, 0), mult(1, 1))] since it does not matter in which order the two operands of the last element in the sequence occur in the construction sequence as long as they precede the final am-expression.

A convenient shorthand notation called the *Backus-Naur Form (BNF)* is usually employed to express the rules of generation . For the set of am-expressions defined above the BNF is as follows.

$$e, e' ::= n \in \mathbb{N} \mid add(e, e') \mid mult(e, e')$$

It is possible to relate the notions of dependence in a construction sequence to the construction process by partially ordering the process of construction of elements in an inductively generated set as follows. Let B , K and \mathbb{U} be as in definition 0.33. Consider the infinite sequence of sets

$$[A_0 \subseteq A_1 \subseteq A_2 \subseteq A_3 \subseteq \dots] \quad (3)$$

defined by mathematical induction as $A_0 = B$ and for each $i \geq 0$, $A_{i+1} = A_i \cup \{f(a_1, \dots, a_{\alpha(f)}) \mid f \in K, a_1, \dots, a_{\alpha(f)} \in A_i\}$. Now consider the set

$$A_\infty = \bigcup_{i \geq 0} A_i \quad (4)$$

The following lemma shows that $A_\infty = A$ and is indeed the smallest solution to the equation (2). Moreover (3) gives a construction of the smallest solution by mathematical induction.

Lemma 0.37 *Let B , K and \mathbb{U} be as in definition 0.33 and A_∞ be as in equation (4). Then $A = A_\infty$.*

Proof: By definition (4) $A_i \subseteq A_\infty$ for each $i \in \mathbb{N}$

$$A_0 \subseteq A_1 \subseteq A_2 \subseteq A_3 \subseteq \cdots \subseteq A_\infty$$

We structure the proof in the form of two claims. Firstly we show that $A_\infty \in \mathcal{X}$ i.e. A_∞ is a set that satisfies the conditions in definition 0.33. Secondly, we prove that $A_\infty \subseteq A$. It follows then that $A = A_\infty$ since $A \subseteq X$ for each $X \in \mathcal{X}$.

Claim. $A_\infty \in \mathcal{X}$.

Proof of claim. Since $B = A_0 \subseteq A_\infty$, A_∞ does satisfy the basis condition. Consider any $f \in K$ and elements $a_1, \dots, a_{\alpha(f)} \in A_\infty$. By the definition of A_∞ , there exist indices $i_1, \dots, i_{\alpha(f)} \in \mathbb{N}$ such that $a_1 \in A_{i_1}, \dots, a_{\alpha(f)} \in A_{i_{\alpha(f)}}$. Let $i \geq \max(i_1, \dots, i_{\alpha(f)})$. It is clear from the definition of A_i that $a_1, \dots, a_{\alpha(f)} \in A_i$ and hence $f(a_1, \dots, a_{\alpha(f)}) \in A_{i+1} \subseteq A_\infty$. Hence A_∞ satisfies the second condition too and therefore $A_\infty \in \mathcal{X}$. \dashv

Claim. $A_\infty \subseteq A$.

Proof of claim. We prove by mathematical induction on indices i , that $A_i \subseteq A$ for every index i . The basis is trivial since we know that $A_0 = B \subseteq A$. For the induction step assume for some $k \geq 0$, $A_k \subseteq A$. We have $A_{k+1} = A_k \cup \{f(a_1, \dots, a_{\alpha(f)}) \mid f \in K, a_1, \dots, a_{\alpha(f)} \in A_k\}$. For any $a \in A_k$ we already know by the induction hypothesis that $a \in A$. Any $a \in A_{k+1} - A_k$ is then of

the form $a = f(a_1, \dots, a_{\alpha(f)})$ where $a_1, \dots, a_{\alpha(f)} \in A_k$. Again by the induction hypothesis we have $a_1, \dots, a_{\alpha(f)} \in A$ and since $A \in \mathcal{X}$, $a = f(a_1, \dots, a_{\alpha(f)}) \in A$. Hence $A_{k+1} \subseteq A$. \dashv QED ■

Lemma 0.37 and its proof, in addition to showing us how to obtain least solutions to equations of the form (2) also show the relationship to the principle of mathematical induction. Further the lemma gives us a way to quantify dependence of elements in a construction sequence by assigning each element an order number.

Definition 0.38 *The height of any element a (denoted $\triangle(a)$) in an inductively generated set A is the least index i in the monotonic sequence (3) such that $a \in A_i$.*

In any construction sequence $[a_1, \dots, a_m]$, $a_i \prec a_j$ only if the height of a_i is less than the height of a_j .

Example 0.39 *Let $S \subseteq \mathbb{N}$ be the smallest set of numbers defined by*

$$n ::= 0 \mid n + 1$$

Clearly this defines the smallest set containing 0 and closed under the successor operation on the naturals. It follows that $S = \mathbb{N}$. Notice that the above BNF is merely a rewording of the principle of mathematical induction version 0.

Example 0.39 shows that mathematical induction is merely a particular case of structural induction.

Example 0.40 *The language of minimal logic was defined in example 0.19. We may redefine the language by the following BNF*

$$\mu, \nu ::= a \in \mathbb{A} \mid (\neg\mu) \mid (\mu \rightarrow \nu)$$

We prove that the language is countable.

Proof: We first begin by classifying the formulas of the language according to their depth. Let M_k be the set of formulas of the language such that each formula has a depth at most k for $k \geq 0$. We assume that $M_0 = \mathbb{A}$ and $M_{k+1} = M_k \cup \{(\neg\mu_k), (\mu_k \rightarrow \nu_k) \mid \mu_k, \nu_k \in M_k\}$. Let \overline{M}_k be the set of all formulas of depth $k + 1$ of the form “ $(\neg\mu_k)$ ” and let \vec{M}_k be the set of all formulas of the form “ $(\mu_k \rightarrow \nu_k)$ ”, where $\mu_k, \nu_k \in M_k$. We then have

$$\begin{aligned} M_{k+1} &= M_k \cup \overline{M}_k \cup \vec{M}_k \\ &= M_k \cup (\overline{M}_k - M_k) \cup (\vec{M}_k - M_k) \\ &= M_k \cup N_{k+1} \cup N_{k+1} \end{aligned}$$

Here $N_{k+1} = \overline{N}_{k+1} \cup \overrightarrow{N}_{k+1}$ represents the set of all formulas of depth exactly $k + 1$. \overline{N}_{k+1} consists of exactly those formulas of depth $k + 1$ whose root operator is \neg . Similarly \overrightarrow{N}_{k+1} represents the set of all formulas of depth exactly $k + 1$, whose root operator is \rightarrow . Hence the three sets are mutually disjoint.

$$M_k \cap \overline{N}_{k+1} = \emptyset, \quad M_k \cap \overrightarrow{N}_{k+1} = \emptyset, \quad \overline{N}_{k+1} \cap \overrightarrow{N}_{k+1} = \emptyset$$

The entire language may then be defined as the set $\mathcal{M}_0 = \bigcup_{k \geq 0} M_k = \mathbb{A} \cup \bigcup_{k > 0} \overline{N}_k \cup \bigcup_{k > 0} \overrightarrow{N}_k$

Claim. Each of the sets M_k , \overline{N}_{k+1} and \overrightarrow{N}_{k+1} is countably infinite for all $k \geq 0$.

Proof of claim. We prove this claim by induction on k . The basis is $M_0 = \mathbb{A}$ and it is given that it is countably infinite. The induction step proceeds as follows. We have by the induction hypothesis that M_k is countably infinite. Hence there is a bijection $\text{num}_k : M_k \xrightarrow[\text{onto}]{} \mathbb{N}$. We use num_k to construct the 1 – 1 correspondence num_{k+1} as follows: We may use num_k to define a bijection between \overline{N}_k and \mathbb{N} . Similarly there exists a 1-1 correspondence between \overrightarrow{N}_{k+1} and $\mathbb{N} \times \mathbb{N}$ given by the ordered pair of numbers $(\text{num}_k(\mu_k), \text{num}_k(\nu_k))$ for each $(\mu_k \rightarrow \nu_k) \in \overrightarrow{N}_{k+1}$. But we know that there is a 1-1 correspondence $\text{diag} : \mathbb{N} \times \mathbb{N} \xrightarrow[\text{onto}]{} \mathbb{N}$.

Hence each of the 3 sets M_k , N_{k+1}^\neg and N_{k+1}^\rightarrow is countably infinite. Their union is clearly countably infinite by the following 1-1 correspondence.

$$num_{k+1}(\mu_{k+1}) = \begin{cases} 3 \times num_k(\mu_{k+1}) & \text{if } \mu_{k+1} \in M_k \\ 3 \times num_k(\mu_k) + 1 & \text{if } \mu_{k+1} \equiv \neg \mu_k \in N_{k+1}^\neg \\ 3 \times diag(num_k(\mu_k), num_k(\nu_k)) + 2 & \text{if } \mu_{k+1} \equiv \mu_k \rightarrow \nu_k \in N_{k+1}^\rightarrow \end{cases}$$

Hence M_{k+1} is countably infinite.

Having proved the claim it follows (from the fact that a countable union of countably infinite sets yields a countably infinite set) that M the language of minimal logic is a countably infinite set. QED

We present below a generalization of the principle of mathematical induction to arbitrary inductively defined sets. It provides us a way of reasoning about the properties of structures that are inductively defined.

Theorem. The Principle of Structural Induction (PSI). Let $A \subseteq \mathbb{U}$ be inductively defined by the basis B and the constructor set K .

Principle of Structural Induction (PSI)

A property \mathbf{P} holds for all elements of A provided

Basis. \mathbf{P} is true for all basis elements.

Induction step. For each $f \in K$, \mathbf{P} holds for elements $a_1, \dots, a_{\alpha(f)} \in A$ implies \mathbf{P} holds for $f(a_1, \dots, a_{\alpha(f)})$.

Proof: Let \mathbf{P} be a property of the elements of \mathbb{U} that satisfies the conditions above. Let C be the set of all elements of A that satisfy the property \mathbf{P} , i.e. $C = \{a \in A \mid \mathbf{P} \text{ holds for } a\}$. It is clear that $B \subseteq C \subseteq A$. To show that $C = A$ it suffices to prove that $A - C = \emptyset$. Consider the sequence of sets defined in (3) and the set $A = \bigcup_{i \geq 0} A_i$ as given in equation (4). We prove that for all $i \geq 0$,

$A_i \subseteq C$ by assuming that there is a smallest $i \geq 0$ such that $A_i \not\subseteq C$. Since $A_0 = B \subseteq C$, $A_i \not\subseteq C$ implies $i > 0$. Consider the smallest $i > 0$ such that $A_i \not\subseteq C$. There exists $a \in A_i - A_{i-1}$ which does not satisfy the property \mathbf{P} . Since $a \notin A_{i-1}$, we have $a = f(a_1, \dots, a_{\alpha(f)})$ for some $f \in K$ such that $a_1, \dots, a_{\alpha(f)} \in A_{i-1}$. Further by assumption $A_{i-1} \subseteq C$ and hence property \mathbf{P} holds for each of $a_1, \dots, a_{\alpha(f)}$. This implies by the induction step that \mathbf{P} holds for a , contradicting the assumption

that a does not satisfy \mathbf{P} . Hence there is no smallest i such that $A_i \not\subseteq C$. Therefore for all $i \geq 0$, $A_i \subseteq C$ and hence $A \subseteq C$ from which it follows that \mathbf{P} holds for every element of A . QED ■

Example 0.41 Consider the following BNF of arithmetic expressions

$$e, e' ::= n \in \mathbb{N} \mid (e + e) \mid (e \times e') \mid (e - e')$$

Given a string w of symbols, a string u is called a prefix of w if there is a string v such that $w = u.v$, where $.$ denotes the (con)catenation operation on strings. Clearly the empty string ϵ is a prefix of every string and every string is a prefix of itself. u is called a proper prefix of w if v is a nonempty string.

Let e be any expression generated by the above BNF and let e' be a prefix⁹ of e . Further, let $L(e')$ and $R(e')$ denote respectively the numbers of left and right parentheses in e' . Let \mathbf{P} be the property of strings e in the language of arithmetic expressions given by

For every prefix e' of e , $L(e') \geq R(e')$.

We use the principle of structural induction (theorem 0.11) to prove that property \mathbf{P} holds for all

⁹ e' may or may not be an expression of the language.

expressions in the language.

Basis. It holds for all $n \in N$ because no natural number has any parentheses.

Induction Hypothesis (IH).

For any e of the form “ $(f \odot g)$ ”, where $\odot \in \{+, x, -\}$ and f, g are themselves expressions in the language

1. *For every prefix f' of f , $L(f') \geq R(f')$ and*
2. *For every prefix g' of g , $L(g') \geq R(g')$*

Induction Step. We do a case analysis of all the possible prefixes of e .

- Case $e' = \varepsilon$. $L(e') = R(e')$.
- Case $e' = “.“$. $L(e') = 1 > 0 = R(e')$.
- Case $e' = “(f’“.$ By the induction hypothesis we have $L(f') \geq R(f')$ and hence $L(e') = 1 + L(f') > R(f') = R(e')$.

- Case $e' = "(f' \odot "$. By the induction hypothesis we have $L(f') \geq R(f')$ and hence $L(e') = \frac{1 + L(f')}{1 + L(f')} > R(f') = R(e')$.
- Case $e' = "(f \odot g' "$. By the induction hypothesis we have $L(f) \geq R(f)$ and $L(g') \geq R(g')$. Hence $L(e') = 1 + L(f) + L(g') > R(f) + R(g') = R(e')$.
- Case $e' = "(f \odot g "$. By the induction hypothesis we have $L(f) \geq R(f)$ and $L(g) \geq R(g)$. Hence $L(e') = 1 + L(f) + L(g) > R(f) + R(g) = R(e')$.
- Case $e' = e = "(f \odot g)"$. By the induction hypothesis we have $L(f) \geq R(f)$ and $L(g) \geq R(g)$. Hence $L(e') = L(e) = 1 + L(f) + L(g) \geq R(f) + R(g) + 1 = R(e') = R(e)$.

We leave it as an exercise for the reader to prove that every proof by the principle of mathematical induction may also be translated into a proof by the principle of structural induction (see example 0.39 and the equivalences between the various versions of the principle of mathematical induction and complete induction). We are now ready to show that even though structural induction seems to be more general than mathematical induction they are in fact equivalent in power. In other words, every proof by the principle of structural induction may also be rewritten as a proof by (some version) of the principle of mathematical induction or complete induction. To do this we need the height (see definition 0.38) of each element in an inductively defined set.

Theorem 0.42 Every proof using the principle of structural induction (PSI) may be replaced by a proof using the principle of complete induction (PCI).

Proof: Let A be inductively defined by B , K , \mathbb{U} and let \mathbf{P} be a property of elements of A that has been proved by the principle of structural induction. Let the property $\mathbf{Q}(n)$ for each natural number n be defined as

The property \mathbf{P} holds for all elements of height n

Basis. The basis step $n = 0$ of the proof of property \mathbf{Q} proceeds exactly as the basis step of the proof by PSI with as many cases are required in the proof of by PSI.

The induction hypothesis. The induction hypothesis is the assumption that $\mathbf{Q}(m)$ holds for all $0 \leq m < n$.

The induction step The induction step is simply that if the induction hypothesis holds for all $m < n$ then \mathbf{Q} holds for n . The proof by PSI for each constructor in the induction step is a case in the induction step of the proof $\mathbf{Q}(n)$.

QED



Definition 0.43 Let A be inductively defined by B , K , and \mathbb{U} and let V be any arbitrary set. Then $h : A \rightarrow V$ is said to be an **inductively defined function** if $h(b) = h_0(b)$ and $h(f(a_1, \dots, a_{\alpha(f)})) = h_f(h(a_1), \dots, h(a_{\alpha(f)}))$ where $h_0 : B \rightarrow V$ is a function and for every n -ary constructor $f \in K$, there is an n -ary function $h_f : V^n \rightarrow V$. A relation $\mathcal{R} \subseteq A \times V$ is said to be **inductively defined** if

$$\mathcal{R} = \{(b, h_0(b)) \mid b \in B\} \cup \{(f(a_1, \dots, a_{\alpha(f)}), h_f(v_1, \dots, v_{\alpha(f)})) \mid a_1 \mathcal{R} v_1, \dots, a_{\alpha(f)} \mathcal{R} v_{\alpha(f)}\}$$

Example 0.44 Consider the language \mathcal{P}_0 of propositional logic, where the basis is a countable set \mathbb{A} of atoms, and the language is defined by the BNF

$$\phi, \psi ::= p \in \mathbb{A} \mid (\neg\phi) \mid (\phi \vee \psi) \mid (\phi \wedge \psi)$$

Further let $\mathbf{B} = \langle \{0, 1\}, \bar{}, +, \cdot \rangle$ be the algebraic system whose operations are defined as follows.

$$\begin{aligned} \bar{0} &= 1 && \text{and } \bar{1} = 0 \\ 1 + 1 &= 0 + 1 = 1 + 0 = 1 && \text{and } 0 + 0 = 0 \\ 0 \times 0 &= 0 \times 1 = 1 \times 0 = 0 && \text{and } 1 \times 1 = 1 \end{aligned}$$

Let $\tau_0 : \mathbb{A} \rightarrow \{0, 1\}$ be a truth value assignment for the propositional atoms. Then the truth values of propositional formulas in \mathcal{P}_0 under τ , is the inductively defined function $\mathcal{T} : \mathcal{P}_0 \rightarrow \{0, 1\}$

where $\tau_- = \neg$, $\tau_\vee = +$ and $\tau_\wedge = \times$. Therefore \mathcal{T} extends τ_0 to \mathcal{P}_0 as follows.

$$\begin{aligned}\mathcal{T}[p] &= \tau_0(p) & p \in \mathbb{A} \\ \mathcal{T}[(\neg\phi)] &= \mathcal{T}[\phi] \\ \mathcal{T}[(\phi \vee \psi)] &= \mathcal{T}[\phi] + \mathcal{T}[\psi] \\ \mathcal{T}[(\phi \wedge \psi)] &= \mathcal{T}[\phi] \times \mathcal{T}[\psi]\end{aligned}$$

0.12. Simultaneous Induction

One question that naturally arises is that if induction is merely equation solving, then are there problems or properties which require us to solve a system of simultaneous equations? We answer this question with the following example.

Example 0.45 Consider the following BNF which consists of the generation of three different sets of bit strings (i.e. sequences of 0s and 1s) which are all mutually dependent.

$$\begin{aligned}z &::= 0 \mid z0 \mid i1 \\ i &::= 1 \mid t0 \mid z1 \\ t &::= i0 \mid t1\end{aligned}$$

In effect this BNF defines a system of three (simultaneous) equations.

$$\begin{aligned}Z &= \{0\} \cup \{z0 \in 2^+ \mid z \in Z\} \cup \{i1 \in 2^+ \mid i \in I\} \\I &= \{1\} \cup \{t0 \in 2^+ \mid t \in T\} \cup \{z1 \in 2^+ \mid z \in Z\} \\T &= \{i0 \in 2^+ \mid i \in I\} \cup \{t1 \in 2^+ \mid t \in T\}\end{aligned}$$

For any bit string s let $[s]$ denote the non-negative integer j such that s is a binary representation of j (there could be leading zeroes). Suppose we need to prove the following property P_Z of the set Z .

$$P_Z : Z = \{z \in 2^+ \mid [z] \text{ is a non-negative multiple of } 3\}$$

The intuition which guides the BNF above (or equivalently the above system of equations) is the following.

- Each $z \in Z$ is such that $[z] = 3l$ for some $l \in \mathbb{N}$ i.e. $[z]$ is a multiple of 3. In particular, the bit-string $0 \in Z$. The other sets in the definition of Z are constructed so that if $[z] = 3l$, for some $l \in \mathbb{N}$, then $[z0] = 6l$ is also a multiple of 3 and for each $i \in I$, if $[i] = 3m + 1$ for some $m \in \mathbb{N}$ then $i1 \in Z$ since $[i1] = 2(3m + 1) + 1 = 3(2m + 1)$ is a multiple of 3.
- Likewise the definitions of I and T are such that for each $i \in I$, $[i] = 3m + 1$ for some natural m and for each $t \in T$, $[t] = 3n + 2$ for some natural n .

The same intuition also guides the proof of property \mathbf{P}_Z . However, the proof is dependent upon properties \mathbf{P}_I for the set I and \mathbf{P}_T for T where

$$\mathbf{P}_I : I = \{i \in \mathcal{2}^+ \mid [i] = 3m + 1, m \in \mathbb{N}\}$$

and

$$\mathbf{P}_T : T = \{t \in \mathcal{2}^+ \mid [t] = 3n + 2, n \in \mathbb{N}\}$$

Hence an inductive proof of property \mathbf{P}_Z requires a simultaneous proof of properties \mathbf{P}_I and \mathbf{P}_T .

Lemma 0.46 *The elements of the sets Z , I and T satisfy properties \mathbf{P}_Z , \mathbf{P}_I and \mathbf{P}_T respectively.*

Proof: We proceed by simultaneous induction on the lengths of strings of $\mathcal{2}^+$.

Basis. There are exactly two bit strings of length 1 viz, “0” and “1” and it follows easily that $0 \in Z$ and $1 \in I$.

Induction Hypothesis (IH).

For some $k > 0$,

0. each $z' \in Z$ such that $|z'| = k$ satisfies property \mathbf{P}_Z ,
1. each $i' \in I$ such that $|i'| = k$ satisfies property \mathbf{P}_I , and
2. each $t' \in T$ such that $|t'| = k$ satisfies property \mathbf{P}_T ,

Induction Step. Consider strings of length $k + 1$. We need to show that

0. Property \mathbf{P}_Z holds for $z = z'0$ and $z = i'1$, where $|z'| = |i'| = k$,
1. Property \mathbf{P}_I holds for $i = t'0$ and $i = z'1$, where $|t'| = |z'| = k$,
2. Property \mathbf{P}_T holds for $t = i'0$ and $t = t'1$, where $|i'| = |t'| = k$,

We leave the rest of the proof to be completed by the interested reader.

QED

Exercise 0.2

1. Prove that version 1, 2 and 3 of PMI are mutually equivalent.

2. Prove that $\mathbb{Z}^+ = Z \cup I \cup T$ in example 0.45.
3. Find the fallacy in the following proof by PMI. Rectify it and again prove using PMI.

Theorem:

$$\frac{1}{1*2} + \frac{1}{2*3} + \dots + \frac{1}{(n-1)n} = \frac{3}{2} * \frac{1}{n}$$

Proof: For $n=1$ the LHS is $1/2$ and so is RHS. assume the theorem is true for some $n > 1$. We then prove the induction step.

$$\begin{aligned} LHS &= \frac{1}{1*2} + \frac{1}{2*3} + \frac{1}{(n-1)n} + \dots + \frac{1}{n(n+1)} \\ &= \frac{3}{2} - \frac{1}{n} + \frac{1}{n(n+1)} \\ &= \frac{3}{2} - \frac{1}{n} + \frac{1}{n} - \frac{1}{(n+1)} \\ &= \frac{3}{2} - \frac{1}{n(n+1)} \end{aligned}$$

which is required to be proved.

4. Let A be any set with a reflexive and transitive binary relation \preceq defined on it. That is to say, $\preceq \subseteq A \times A$ satisfies the following conditions.

(a) For every $a \in A$, $a \preceq a$.

(b) For all a, b, c in A , $a \preceq b$ and $b \preceq c$ implies $a \preceq c$.

Then show by induction that $\preceq; R = \preceq^n; R$ for all $n \geq 1$.

5. Show using PSI that \mathcal{T} is well defined.

6. The language of numerals in the arabic notation may be defined by the following simple grammar

$$d ::= 0|1|2|3|4|5|6|7|8|9$$
$$n ::= d|nd$$

Define the value of a string in this language, so that it conforms to the normally accepted meaning of a numeral

7. For the language of propositions we may also call the function τ a state, and look upon the function \mathcal{T} as defining the truth value of a proposition in a given state τ . Now redefine the meaning of a proposition as the set of states in which the proposition has a truth value of 1. if Σ denotes the set of all possible states i.e. $\Sigma = \{\tau | \tau : B \rightarrow \{0, 1\}\}$ then

(a) What is the domain and the range of the function φ ?

(b) Define the function φ by structural induction.

(c) Prove using the principle of structural induction that for any proposition p , and a state τ , $T(p) = 1$ if and only if τ belongs to the φ -meaning of p .

8. Let A be a language ,inductively defined by B , K , U . Define the set of syntax tree, T_A of the elements of A as follows:

(a) For each $b \in B$, there is a single node labelled b ,

(b) For each n -ary operator \odot and $a_1, \dots, a_n, a \in A$, if $\odot(a_1, \dots, a_n) = a$, the syntax tree t of a is a tree with root labelled by \odot and t_1, \dots, t_n as the immediate subtree of t , where t_1, \dots, t_n are the syntax trees corresponding to a_1, \dots, a_n respectively.

(a) Prove that every element of A has a unique syntax tree if A is free.

(b) Give an example to show that every syntax tree need not define a unique element of A .

9. Let L_0 be the language of propositional logic as defined in the last example. Then intuitively speaking, a propositional formula p is a **subformula** of a propositional formula q if the syntax tree of p is a subtree of the syntax tree of q .

(a) Define the notion of subformula inductively.

(b) Let of every formula q , $SF(q)$ denote the set of all subformulas of q . Define $SF(q)$ inductively.

(c) Let $p \preceq q$ if and only if p is a subformula of q . Prove that \preceq is a partial order on L_0 .

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

110 OF 783

QUIT

1. Lecture 1: Introduction

Lecture 1: Introduction

Tuesday 26 July 2011

1. What is Logic?
2. Reasoning, Truth and Validity
3. Examples
4. Objectivity in Logic
5. Formal Logic
6. Formal Logic: Applications
7. Form and Content
8. Facets of Mathematical Logic
9. Logic and Computer Science

What is Logic?

- Logic is about *reasoning*:
 - *validity* of arguments
 - *consistency* among sets of statements
 - matters of *truth* and *falsehood*
- Logic is concerned only about the *form* of reasoning and not about the *content*

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.

Examples

Example 1.1

All humans live forever.

Socrates is human.

Hence Socrates lives forever.

Example 1.2

All humans are born with opposable toes

By adulthood opposable toes become non-opposable

John is an adult human

Hence John has no opposable toes.

Objectivity in Logic

- The traditional logic of Aristotle and Leibniz are essentially philosophical in nature with its main purpose being to investigate the *objective* laws of thought.
- *Objectivity* implies essentially that arguments must be communicable to and verifiable by other people.
- *Objectivity* has always implied *formalizability*.

The colours and shades of Logic.

The name *mathematical logic* may be interpreted in two ways. We may interpret *logic* as a subject treated using mathematical methods. We may just as well, think of it as a subject which formalises the methods of reasoning used in mathematics. This leads us apparently to a paradox. How can *logic* be treated mathematically without using *logic* itself?

We resolve this paradox as follows. We simply separate out the *logic* that we are studying by expressing it formally through a language – the *object language* or the *target language* – from the logic that is used in reasoning about it. The latter logic that is used for reasoning is called the *meta-language*.

We will define the *object language* formally via a grammar (and for good measure we also colour-code the sentences of the object language in **green**). The meta-language however, is the usual “language of mathematics” – a mixture of natural language with mathematical symbols that is normally used in mathematical texts. The words of these sentences will usually appear in black and sometimes in other colours such as **blue** (e.g for **emphasis**) or **red** (e.g. when we want some concept or idea to **stand out**). However since the objects of our study are **green**, the objects when appearing in these sentences will still be **green**.

When treating any object language formally, it also becomes necessary to specify the **meanings**

of phrases, expressions and sentences in the formal language. Since we will be expressing these meanings through objects in mathematical structures, these objects will be coloured **brown**.

The study of logic promises to be pretty colourful, what?

Formal Logic

- Logic as a **formal language** with a **syntax** to which a **semantics** had to be attached.
- In turn using mathematical methods within logic, led to its formalization as **mathematical logic**
- The strict separation of **syntax** from **semantics** made logic a clean and elegant mathematical discipline which could be then applied to the foundational questions of mathematics.

Formal Logic: Applications

- Of great use in analysing questions concerning the foundations of mathematics
- In clarifying reasoning mechanisms within mathematics.
- Identifying various occurrences of *circular* reasoning which were previously very hard to identify.

Form and Content

- The separation of syntax and semantics also led to the separation of form and content and
- allowed the formalization of correct methods of reasoning purely in terms of the syntax.
- allowed the possibility of plugging in a semantics as demanded by an application whenever it was necessary
- allowed the possibility of *mechanizing* reasoning completely.

Facets of Mathematical Logic

1. A *formalization* language for mathematics,
2. A *calculus* clarifying the notion of a mathematical proof
3. *mechanization* of proof
4. A *sub-discipline* of mathematics itself
5. A *mathematical tool* applicable to various branches of mathematics.

Logic and Computer Science

1. *Mechanization* of reasoning within a formalized syntactic setup.
2. Applications to program and system specification and verification.
3. As a declarative programming language
4. Proving theorems within areas of mathematics where the number of cases is too high or the details of proof are too long and tedious.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

124 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

125 OF 783

QUIT

[2. Lecture 2: Propositional Logic Syntax](#)

Lecture 2: Propositional Logic Syntax

Wednesday 27 July 2011

1. Truth and Falsehood: 1
2. Truth and Falsehood: 2
3. Extending the Boolean Algebra
4. Table of Truth & Falsehood
5. Sums & Products
6. Propositional Logic: Syntax
7. Propositional Logic: Syntax - 2
8. Natural Language equivalents
9. Some Remarks
10. Associativity and Precedence
11. Syntactic Identity
12. Abstract Syntax Trees
13. Subformulae
14. Atoms in a Formula
15. Degree of a Formula
16. Size of a Formula
17. Height of a Formula

Truth and Falsehood: 1

Our notion of meaning is restricted to absolute “truth” and absolute “falsehood” (with nothing else in between) of statements.

Definition 2.1 Let $\mathcal{B} = \langle \mathbf{2}, \{\neg, ., +\}, \{\leq, =\} \rangle$ be the 2-element boolean algebra where $\mathbf{2} = \{0, 1\}$.

- We use 1 to denote absolute “truth” and 0 to denote absolute “falsehood” and
- the boolean operations have their **usual meaning** in the booleans.

Truth and Falsehood: 2

In other words, for any $a, b \in \mathbf{2}$,

- $\bar{a} = 1 - a$ is the unary boolean inverse operation,
- $a + b = \max(a, b)$ is the binary summation operation which yields the maximum of two boolean values regarded as natural numbers,
- $a.b = \min(a, b)$ is the binary product operation which yields the minimum of two boolean values regarded as natural numbers,
- $a \leq b$, and $a = b$ denote the usual binary relations “less-than-or-equal-to” and “equals” on natural numbers restricted to the set $\{0, 1\}$.

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

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

Table of Truth & Falsehood

a	\bar{a}
0	1
1	0

a	b	$a.b$	$a + b$	$a \leq b$	$a \doteq b$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Boolean identities

Negation $\bar{\bar{a}} = a$

Comparison $a \leq b = \bar{a} + b$

Equality $a = b = (a \leq b).(b \leq a)$

$$a + 0 = a$$

$$a + 1 = 1$$

$$a + a = a$$

$$a + b = b + a$$

$$(a + b) + c = a + (b + c)$$

$$a + (b.c) = (a + b).(a + c)$$

$$\overline{a + b} = \bar{a}.\bar{b}$$

$$a + \bar{a} = 1$$

$$\bar{0} = 1$$

$$a + (a.b) = a$$

Identity $a.1 = a$

Zero $a.0 = 0$

Idempotence $a.a = a$

Commutativity $a.b = b.a$

Associativity $(a.b).c = a.(b.c)$

Distributivity $a.(b + c) = (a.b) + (a.c)$

De Morgan $\overline{a.b} = \bar{a} + \bar{b}$

Simplification $a.\bar{a} = 0$

Inversion $\bar{1} = 0$

Absorption $a.(a + b) = a$

Sums & Products

The Associativity and Commutativity identities allow us to define summations and products over sets of boolean values.

$$\sum_{0 < i \leq n} a_i \stackrel{df}{=} a_1 + \cdots + a_n$$

$$\prod_{0 < i \leq n} a_i \stackrel{df}{=} a_1 \cdot \cdots \cdot a_n$$

2.1. Propositions in Natural Languages

We now begin our study of formal logic by studying statements in natural language. We will frequently appeal to reasoning methods employed in sentences (actually statements) in natural languages. We initially restrict our study to statements expressed in natural languages. This logic is often called *propositional* or *sentential* logic.

In general sentences in any natural language may be classified into the following forms (usually indicated by the *mood* of the sentence. A standard text book on English grammar defines a *mood* as *the mode or manner in which the action denoted by the verb is represented*. It further illustrates three moods in the English language. [English Grammar 101](#) classifies the verbs in the English language into four moods (including the *infinitive* as a mood. But to get really confounded and confused, the reader needs to refer to [Wikipedia](#) where several moods are defined (e.g. *optative*, *jussive*, *potential*, *inferential*).

The following example sentences are taken from the references above.

Indicative Mood: expresses an assertion, denial, or question.

Little Rock is the capital of Arakansas.

Ostriches cannot fly.

Have you finished your homework?

Imperative Mood: expresses command, prohibition, entreaty, or advice.

Dont smoke in this building.

Be careful!

Have mercy upon us.

Give us this day our daily bread.

Subjunctive Mood: expresses a doubt, a wish or an improbability

God bless you!

I wish I knew his name.

I would rather be paid by cheque.

He walks as though he were drunk.

Loosely speaking natural language sentences which are assertions or denials in the indicative mood may be considered propositions. Questions however, cannot be considered propositions. More accurately only those sentences to which one might (at least theoretically) ascribe a truth value

(such as assertions and denials) may be considered to be propositions in our setting. Hence of the examples given above the only ones which are of interest to us would be

Little Rock is the capital of Arkansas.

(which is an assertion) and

Ostriches cannot fly.

which is a denial (it denies the statement *Ostriches can fly*). The last indicative statement

Have you finished your homework?

is a question for which no truth value can be assigned. It is therefore not a proposition in the sense that we understand propositions. Notice that a denial is an assertion too – it is simply the negation of an assertion and hence is a statement to which a truth value may be assigned. We will treat denials also as assertions and declare that assertions in natural language are all propositions in our sense of the term.

The examples that we have considered above are rather simple. We could consider more examples of assertions, denials and complex assertions which are made up of assertions and denials. Here are a few.

- *God is in his Heaven and all is right with the world.*
- *Time and tide wait for no man.*
- *You can fool some people some of the time, some people all of the time and all the people some of the time, but you cannot fool all the people all the time.*
- *Anybody who becomes the Prime Minister has a clear national and international agenda.*

Propositional Logic: Syntax

Definition 2.3

- A : a countably infinite collection of propositional atoms.
- $\Omega_0 = \{\perp, \top, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$: the set of operators (also called connectives) disjoint from A
- Each operator has a fixed arity defined by α with
 - $\alpha(\perp) = \alpha(\top) = 0$
 - $\alpha(\neg) = 1$ and
 - $\alpha(\odot) = 2$, for $\odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- A and Ω are disjoint and parentheses (and) of various kinds are used for grouping.
- \mathcal{P}_0 is the smallest set generated from A and Ω_0 .

Propositional Logic: Syntax - 2

Alternatively, we may define the language by the following grammar.

Definition 2.4

$$\phi, \psi ::= \perp \mid \top \mid p \in A \mid (\neg\phi) \mid (\phi \wedge \psi) \mid (\phi \vee \psi) \mid (\phi \rightarrow \psi) \mid (\phi \leftrightarrow \psi)$$

Each expression of this language is also called a well-formed formula (wff) (or sentence) of \mathcal{P}_0 . Each atom is a simple formula or sentence. Each sentence having one or more occurrences of unary or binary operators is called a compound formula. \mathcal{P}_0 is the set of all sentences.

Natural Language equivalents

The (unary and binary) operators of the language \mathcal{P}_0 are chosen to denote constructs that allow the construction of compound statements from simple ones. The following table shows the intended meaning of each of the operators in the English language.

Operator	Name	English rendering
\perp	<i>bottom</i>	false
\top	<i>top</i>	true
\neg	<i>negation</i>	not
\wedge	<i>conjunction</i>	and
\vee	<i>disjunction</i>	or
\rightarrow	<i>conditional</i>	if...then
\leftrightarrow	<i>biconditional</i>	if and only if

2.2. Translation of Natural Language Statements

We may translate natural language sentences into propositions by identifying the simplest sentences as atoms and connecting up the simple sentences using the connectives at our disposal. Naturally, many of the subtleties of the use of the connectives in natural languages would be lost in translation. For the purposes of logical reasoning, the table given in [Natural Language equivalents](#) is sufficient for propositional reasoning, if we ignore the subtleties in natural language such as tonal variations, implied tense and judgmental implications that often come loaded with the sentences. We will have more to say on this with respect to particular connectives in the following descriptions.

Negation (\neg). This connective is used to mean [not](#), [it is not the case that](#) and abbreviated negation prefixes such as [non-](#) and [un-](#). If the atom A denotes the simple statement [I am at work](#), then $(\neg A)$ denotes [I am not at work](#). In certain cases opposites could be translated using negation. For example, if the sentence

[This dish is good.](#)

is denoted by the atom B , the sentence

[This dish is bad.](#)

and the statement

This dish is *not good*.

are both translated as $(\neg B)$. Going further, the sentence

This dish is *not bad*.

would be translated as $(\neg\neg B)$. As we shall see both the statements B and $(\neg\neg B)$ would be considered *logically* (though not *syntactically*) *equivalent* and the subtle difference between the expressions *good* and *not bad* would be lost in translation.

Conjunction (\wedge). The usual meaning of conjunction as denoting *and* holds. In addition, the words *but*, *moreover*, *furthermore* and phrases such as *in addition* would all be considered synonymous with *and*. Notice that the sentence

Rama *and* Seeta went to the forest

would be considered synonymous with the sentence

Rama went to the forest *and* Seeta went to the forest

even though the first sentence has an implied meaning of “togetherness” or “simultaneity”.

The operator \wedge is commutative as we shall see later. Therefore for any formulae ϕ and ψ , $\phi \wedge \psi$ would be logically equivalent to $\psi \wedge \phi$. Hence the implied difference between the two sentences (from [7])

Jane got married *and* had a baby.

and

Jane had a baby *and* got married.

is usually lost in translation.

Disjunction (\vee). The usual meaning is an *or* in the *inclusive* sense although *or* is often used in English in the *exclusive* sense. Hence $\phi \vee \psi$ would have to be translated to mean *either ϕ or ψ or both ϕ and ψ* .

In our natural language arguments we will use *or* in the inclusive sense. *either ϕ or ψ* would refer to an *or* that is used in the *exclusive* sense. That is *either ϕ or ψ* would mean that exactly one of the two propositions holds and both cannot hold at the same time. Therefore the sentence

Ram *or* Shyam topped the class.

which could be equivalently rendered in English as a compound sentence

Ram topped the class *or* Shyam topped the class.

Neither of the above sentences rules out the possibility that both Ram and Shyam topped the class, whereas

Either Ram *or* Shyam topped the class.

would mean that exactly one of them could have topped the class.

Analogously one might ask what is the correct rendering of the sentence

Neither Ram *nor* Shyam topped the class.

If r denotes the atomic statement Ram topped the class and s denotes the atomic statement Shyam topped the class, then $(\neg r) \wedge (\neg s)$ would be an accurate propositional rendering of the sentence.

Conditional (\rightarrow). Also called the *material conditional*, it comes pretty close to the English conditional statement of the form “If ϕ then ψ ”. Alternative translations are “ ψ only if ϕ ”, “ ψ provided ϕ ” and “ ψ whenever ϕ ”.

There are other conditionals that we frequently use in English such as “ ψ unless ϕ ” which may be rendered as “ ψ if not ϕ ” and would be represented by the formula $((\neg\phi) \rightarrow \psi)$.

Biconditional (\leftrightarrow) The translation “If ϕ then ψ , not otherwise” is reserved for the biconditional. Alternative translations for $\phi \leftrightarrow \psi$ are “ ϕ if and only if ψ ”, “ ϕ iff ψ ”, “ ϕ exactly when ψ ” and “ ϕ just in case ψ ”.

The following tables adapted from [7] summarise the various English renderings of each operator given arbitrary operands ϕ and ψ .

Operation	English rendering
$\neg\phi$	not ϕ ϕ does not hold It is not the case that ϕ holds

Operation	English rendering
$\phi \wedge \psi$	ϕ and ψ Both ϕ and ψ ϕ but ψ Not only ϕ but ψ ϕ although ψ ϕ despite ψ ϕ yet ψ ϕ while ψ

Operation	English rendering
$\phi \vee \psi$	ϕ or ψ ϕ or ψ or both ϕ and/or ψ ϕ unless ψ ϕ except when ψ

Operation	English rendering
$\phi \rightarrow \psi$	<p>If ϕ then ψ</p> <p>ψ if ϕ</p> <p>ϕ only if ψ</p> <p>When ϕ then ψ</p> <p>ψ when ϕ</p> <p>ϕ only when ψ</p> <p>In case ϕ then ψ</p> <p>ψ in case ϕ</p> <p>ψ provided ϕ</p> <p>ϕ is a sufficient condition for ψ</p> <p>ψ is a necessary condition for ϕ</p>

Operation	English rendering
$\phi \leftrightarrow \psi$	ϕ if and only if ψ ϕ iff ψ ϕ if ψ and ψ if ϕ If ϕ then ψ , and conversely ϕ exactly if ψ ϕ exactly when ψ ϕ just in case ψ ϕ is a necessary and sufficient condition for ψ

Some Remarks

- It is convenient to have a syntactic symbol for “absolute truth” and “absolute falsehood” though it is strictly not necessary.
- \perp and \top are **constants** and hence have no **precedence** associated with them.
- In a formula of the form $\phi \rightarrow \psi$, ϕ is called the *antecedent* and ψ the *consequent*.

2.3. Associativity and Precedence of Operators

Notice that we have defined the language to be fully parenthesized i.e. every *compound* formula is enclosed in a pair of parentheses $((\dots))$. However it may actually be very distracting and confusing to read formulae with too many parentheses. We will define **precedence and associativity conventions** to reduce the number of parentheses while reading and writing formulae.

Associativity conventions. This convention refers to the consecutive occurrences of the same binary operator. Simple examples from school arithmetic are used to illustrate the conventions used in disambiguating expressions which are not fully parenthesized.

Left Associativity is the convention that an expression on numbers like $6 - 3 - 2$ should be read as $((6 - 3) - 2)$ (which would yield the value 1). It should not be read as $(6 - (3 - 2))$ (which would yield the value 5). Here we say that *the subtraction operation associates to the left* or that *subtraction is a left associative operation*. Other binary operations such as addition, multiplication and division on numbers are also left associative.

Right Associativity is the convention used to group consecutive occurrences of powers of numbers. For example 4^{3^2} is to be read as $(4^{(3^2)})$ which would yield the result $4^9 = 262144$. It should not be read as $((4^3)^2)$ which would yield the result $64^2 = 4096$. We say that *exponentiation associates to the right*.

tiation is right associative.

Precedence of operators If two different binary operators occur consecutively in an expression, we need some way to associate and group them unambiguously. This is usually specified for binary operators by a precedence relation. In school arithmetic this usually takes the form of the “bodmas” rule specifying that brackets have the highest precedence followed by division and multiplication which in turn are followed by addition and subtraction. Hence for example an expression such as $3 \times 4 + 5$ is to be read as $((3 \times 4) + 5)$ representing the value 17. It would be wrong to read $3 \times 4 + 5$ as $(3 \times (4 + 5))$ (representing the value $3 \times 9 = 27$) since *multiplication has a higher precedence than addition or multiplication precedes addition*. This is usually denoted $\times \prec +$.

Similarly the expression $3 + 4 \times 5$ is to be read as $(3 + (4 \times 5))$ yielding the value $3 + 20 = 23$ and it would be wrong to read it as $((3 + 4) \times 5)$ (which represents the value $7 \times 5 = 35$).

It is the usual convention in mathematical texts that unless otherwise specified, unary operators have a higher precedence than binary operators.

Precedence of operators with equal precedence When two distinct binary operators have equal precedence (e.g. addition and subtraction on numbers) then our convention dictates that in the absence of parentheses, the left operator has a higher precedence in the expression than the one

on the right i.e. they should be grouped from left to right. Thus $5 + 4 - 3$ is to be read as $((5 + 4) - 3)$ yielding 6 (even though reading it as $(5 + (4 - 3))$ yields the same result). Similarly $5 - 4 + 3$ should be read as $((5 - 4) + 3)$ (yielding the result 4) rather than as $(5 - (4 + 3))$ (which yields -2). For example $24 / 4 \times 3$ would yield 18 by our convention. While this convention is well-established for left associative operators, it is not clear what the convention is when there are consecutive occurrences of distinct right associative operators having equal precedence. However in case of any confusion we may always put in enough parentheses to disambiguate the expression.

Our interest in precedence, however is restricted to being able to translate an expression written in linear form unambiguously into its **abstract syntax tree**. The use of parentheses aids in unambiguously defining an unique abstract syntax tree. Even though we write formulae in linear form we will always implicitly assume that they represent the corresponding abstract syntax tree.

Associativity and Precedence

Our language has been defined to be **fully parenthesized**.

- The binary operators \wedge and \vee are **left associative** i.e. a formula written as $\phi \wedge \psi \wedge \chi$ should be read as $((\phi \wedge \psi) \wedge \chi)$.
- The binary operators \rightarrow and \leftrightarrow , on the other hand are **right associative** i.e. a formula written as $\phi \rightarrow \psi \rightarrow \chi$ should be read as $(\phi \rightarrow (\psi \rightarrow \chi))$.
- The operator precedence convention we follow is:

$\leftrightarrow \prec \rightarrow \prec \vee \prec \wedge \prec \neg$

i.e. \neg has the highest precedence and \leftrightarrow has the lowest.

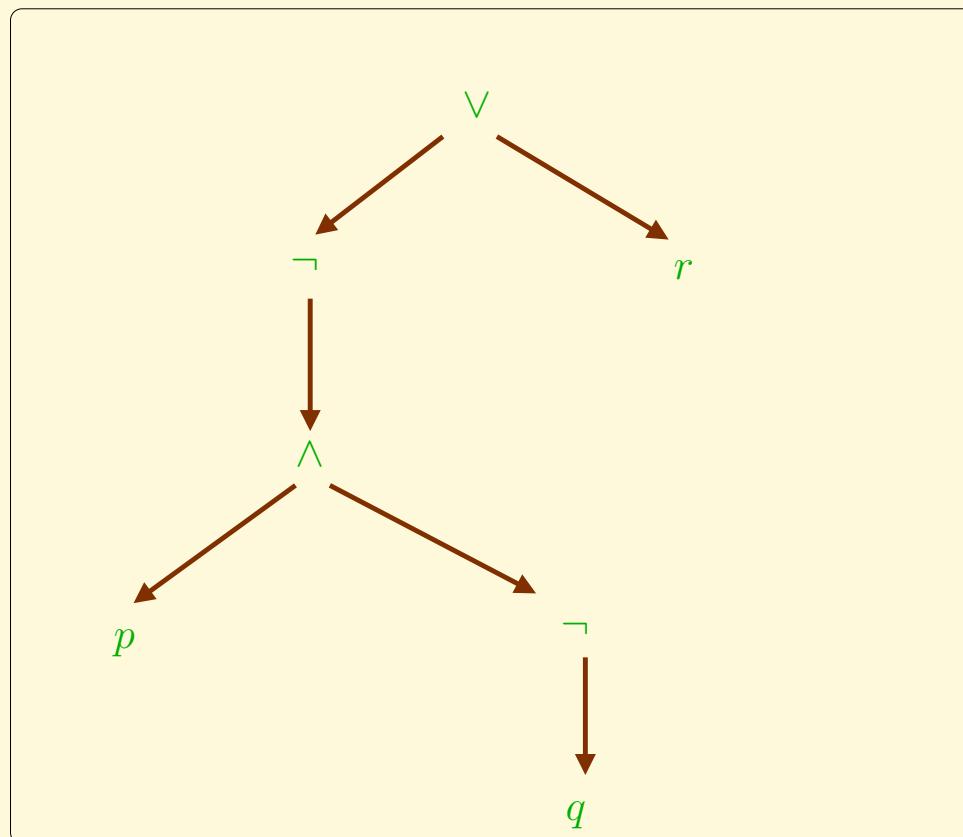
Syntactic Identity

- The precedence rules make a lot of parentheses redundant.
- Instead of propositions written linearly we will rather think of propositions as **abstract syntax trees (AST)**.
- Any two propositions ϕ and ψ which have the same AST will be considered *syntactically identical* and denoted $\phi \equiv \psi$.
- If $\phi \equiv \psi$ then ϕ and ψ differ only in the presence of redundant parentheses.

Abstract Syntax Trees

Abstract syntax trees are rooted directed trees (see definition 2.11)

Example 2.5 *The AST of the formula $(\neg(p \wedge \neg q) \vee r)$.*



Subformulae

Definition 2.6 The set of subformulae of any formula ϕ is the set $SF(\phi)$ defined by induction on the structure of formulae as follows:

$$SF(\perp) = \{\perp\}$$

$$SF(\top) = \{\top\}$$

$$SF(p) = \{p\},$$

for each atom p

$$SF(\neg\psi) = SF(\psi) \cup \{\neg\psi\}$$

$$SF(\psi \odot \chi) = SF(\psi) \cup SF(\chi) \cup \{\psi \odot \chi\}, \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

Atoms in a Formula

Definition 2.7 *The atoms of a formula is defined by induction on the structure of formulae.*

$$\text{atoms}(\perp) = \{\perp\}$$

$$\text{atoms}(\top) = \{\top\}$$

$$\text{atoms}(p) = \{p\}, \quad \text{for each atom } p$$

$$\text{atoms}(\neg\phi) = \text{atoms}(\phi)$$

$$\text{atoms}(\psi \odot \chi) = \text{atoms}(\psi) \cup \text{atoms}(\chi), \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

Degree of a Formula

Definition 2.8 *The degree of a formula is defined by induction on the structure of formulae.*

$$\text{degree}(\perp) = 0$$

$$\text{degree}(\top) = 0$$

$$\text{degree}(p) = 0, \quad \text{for each atom } p$$

$$\text{degree}(\neg\phi) = \text{degree}(\phi)$$

$$\text{degree}(\psi \odot \chi) = 1 + \text{degree}(\psi) + \text{degree}(\chi), \quad \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

Size of a Formula

Definition 2.9 *The size of a formula is defined by induction on the structure of formulae.*

$$\text{size}(\perp) = 1$$

$$\text{size}(\top) = 1$$

$$\text{size}(p) = 1, \quad \text{for each atom } p$$

$$\text{size}(\neg\phi) = 1 + \text{size}(\phi)$$

$$\text{size}(\psi \odot \chi) = 1 + \text{size}(\psi) + \text{size}(\chi), \quad \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

Height of a Formula

Definition 2.10 *The height of a formula is defined by induction on the structure of formulae.*

$$\text{height}(\perp) = 0$$

$$\text{height}(\top) = 0$$

$$\text{height}(p) = 0, \quad \text{for each atom } p$$

$$\text{height}(\neg\phi) = 1 + \text{height}(\phi)$$

$$\text{height}(\psi \odot \chi) = 1 + \max(\text{height}(\psi), \text{height}(\chi)), \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

where max is the maximum of two numbers.

Exercise 2.1

1. Prove that the set \mathcal{P}_0 is countably infinite. Hint: Use the solution given in example 0.40.
2. Let $\mathcal{T}(\mathcal{P}_0)$ be the set of all abstract syntax trees of the language \mathcal{P}_0 . Define a function $AST : \mathcal{P}_0 \longrightarrow \mathcal{T}(\mathcal{P}_0)$ which for any well-formed formula yields the corresponding unique abstract syntax tree of the formula.

2.4. Rooted Trees

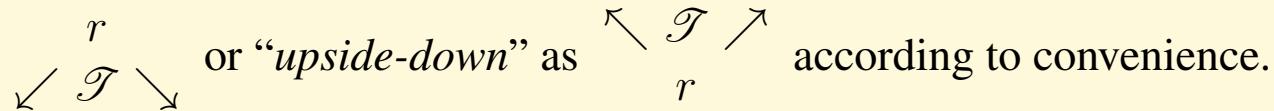
Definition 2.11

- A **directed graph** is a pair $\langle N, \rightarrow \rangle$ consisting of a finite or a countably infinite set N of **nodes** and a set $\rightarrow \subseteq N \times N$ of (**directed**) **edges** such that for any edge $(s, t) \in \rightarrow$ (usually denoted $s \rightarrow t$ in infix notation), s is called the **source** and t the **target** of the edge. s is called a **predecessor** of t and t is called a **successor** of s .
- A (**directed**) **path** of length $k \geq 1$ in a directed graph is a finite sequence of nodes $n_0, n_1, \dots, n_{k-1}, n_k$ such that $n_i \rightarrow n_{i+1}$ for each $i \in \{0, \dots, k-1\}$. In addition if $n_k = n_0$ the path is called a **cycle** of length k , A cycle of length 1 is called a **self-loop**
- A directed graph is called a **directed acyclic graph (DAG)** if it has no cycles.
- An **unordered (directed) tree** is a directed acyclic graph such that there is at most one path between any pair of nodes.
- A **(rooted directed) tree** is a triple $\langle N, \rightarrow, r \rangle$ such that $\langle N, \rightarrow \rangle$ is an unordered (directed) tree and $r \in N$ is a distinguished node called the **root** of the tree and satisfying the following the properties.

- There exists a function $\ell : N \rightarrow \mathbb{N}$ called the **level** such that $\ell(r) = 0$ and for any $s \rightarrow t$, $\ell(t) = \ell(s) + 1$.
- Every node in $N - \{r\}$ has a unique predecessor.

We will be primarily interested in rooted trees and we will simply refer to them as trees.

Notation. We will often represent a tree \mathcal{T} specified only by a name and a root node r either as



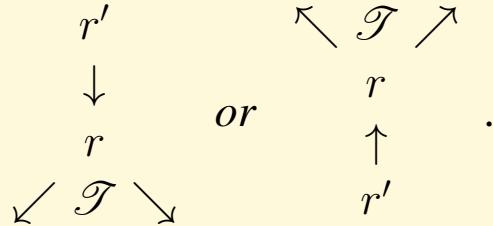
Facts 2.12

- Every node in N is source or a target of one or more edges,
- The \rightarrow relation is irreflexive (i.e. there is no edge whose source and target are the same node).
- The root node has no predecessor.
- A **leaf (node)** is a node with no successor.

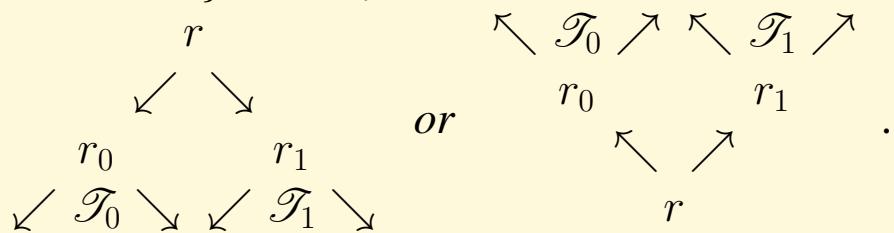
Definition 2.13 Let $\mathcal{T} = \langle N, \rightarrow, r \rangle$ be a rooted tree.

- \mathcal{T} is also called a **tree rooted at r** .
- \mathcal{T} is infinite if N is an infinite set.
- A **path** in \mathcal{T} is a finite or (countably) infinite sequence $r = n_0 \rightarrow n_1 \rightarrow n_2 \rightarrow \dots$ such that $(n_i, n_{i+1}) \in \rightarrow$ for each $i \geq 0$.
- A finite path $r = n_0 \rightarrow n_1 \rightarrow n_2 \rightarrow \dots n_m$ is said to have a length $m \geq 0$. A path which is not finite is said to be **infinite**.
- A **branch** is a maximal path – it is either infinite or is finite $r = n_0 \rightarrow n_1 \rightarrow n_2 \rightarrow \dots n_m$ such that n_m is a leaf.
- The **successors** of a node $s \in N$ is the set $\text{Succ}(s) = \{t \in N \mid s \rightarrow t\}$ and **predecessors** of a node $t \in N$ is the set $\text{Pred}(t) = \{s \in N \mid s \rightarrow t\}$.
- \rightarrow^+ is the transitive closure of the \rightarrow relation and \rightarrow^* is the reflexive-transitive closure of \rightarrow .
- The **descendants** of a node $n \in N$ is the set $\text{Desc}(n) = \{n\} \cup \bigcup_{s \in \text{Succ}(n)} \text{Desc}(s)$ and $\text{Desc}(n) - \{n\}$ is the set of **proper descendants** of n .
- The **ancestors** of a node n is the set $\text{Ancest}(n) = \{n\} \cup \bigcup_{s \in \text{Pred}(n)} \text{Ancest}(s)$ and the **proper ancestors** of n is the set $\text{Ancest}(n) - \{n\}$.

- A **subtree** of a rooted tree $\mathcal{T} = \langle N, \rightarrow, r \rangle$ is a tree $\mathcal{T}' = \langle \text{Desc}(n), \rightarrow', n \rangle$ rooted at a node $n \in \text{Desc}(r)$ such that $s \rightarrow' t$ for $s, t \in \text{Desc}(n)$ if and only if $s \rightarrow t$.
- A rooted tree $\mathcal{T} = \langle N, \rightarrow, r \rangle$ may be **extended at a leaf** n to another tree $\mathcal{T}' = \langle N', \rightarrow', r \rangle$ by an edge $n \rightarrow' n'$ provided $n' \notin N$, $N' = N \cup \{n'\}$ and n is a leaf node of \mathcal{T} .
- A rooted tree $\mathcal{T} = \langle N, \rightarrow, r \rangle$ may be **extended at the root** r to another tree $\mathcal{T}' = \langle N', \rightarrow', r' \rangle$ by an edge $r \rightarrow' r'$ provided $r' \notin N$, $N' = N \cup \{r'\}$. \mathcal{T}' may be denoted



- Two trees $\mathcal{T}_0 = \langle N_0, \rightarrow_0, r_0 \rangle$ and $\mathcal{T}_1 = \langle N_1, \rightarrow_1, r_1 \rangle$ may be **joined** together at a new node r to form a new tree $\mathcal{T} = \langle N, \rightarrow\rightarrow, r \rangle$ where $N = N_0 \cup N_1 \cup \{r\}$, $\rightarrow = \rightarrow_0 \cup \rightarrow_1 \cup \{r \rightarrow r_0, r \rightarrow r_1\}$. \mathcal{T} may be denoted



- $\mathcal{T} = \langle N, \rightarrow, r \rangle$ is said to be **finitely branching** if for each $n \in N$, $Succ(n)$ is a finite set.

Facts 2.14 In any rooted tree $\mathcal{T} = \langle N, \rightarrow, r \rangle$,

1. $N = Desc(r)$
2. $Ances(n) = \{s \in N \mid s \rightarrow^* n\}$ for any $n \in N$.
3. $Desc(n) = \{t \in N \mid n \rightarrow^* t\}$ for any $n \in N$.
4. A rooted tree is acyclic i.e. for all nodes n , $n \not\rightarrow^+ n$.

Definition 2.15 Let $\mathcal{T} = \langle N, \rightarrow, r \rangle$ be a tree rooted at node $r \in N$. A node $n \in N$ is called **infinitary** if $Desc(n)$ is an infinite set otherwise it is called **finitary**.

Lemma 2.16 In a finitely branching tree, every infinitary node has an infinitary successor.

Proof: If not, then for some infinitary node n , $Succ(n)$ is finite and for each $s \in Succ(n)$, $Desc(s)$ is finite which implies $Desc(n) = \{n\} \cup \bigcup_{s \in Succ(n)} Desc(s)$ which is a finite union of finite sets. Then $Desc(n)$ would be finite. QED ■

Lemma 2.17 (König's Lemma) Every finitely branching infinite tree has an infinite path.

Proof: Assume $\mathcal{T} = \langle N, \rightarrow, r \rangle$ is a finitely branching infinite rooted tree which has no infinite path. Clearly since $N = Desc(r)$ is infinite, r is infinitary. r has an infinitary successor by lemma 2.16. Hence there exists a maximal path in \mathcal{T} all of whose nodes are infinitary. This path has to be infinite, otherwise there would be a last node in the path which is infinitary but has no successors, which is impossible. QED ■

Corollary 2.18 *Every infinite tree is infinitely branching or finitely branching with at least one infinite path.*



Corollary 2.19 (Contrapositive of König's Lemma) *A finitely branching tree is finite if and only if it has no infinite path.*



HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

168 OF 783

QUIT

Lecture 3: Semantics of Propositional Logic

Friday 29 July 2011

1. Semantics of Propositional Logic: 1
2. Semantics of Propositional Logic: 2
3. Models and Satisfiability
4. Example: Abstract Syntax trees
5. Tautology, Contradiction, Contingent

Semantics of Propositional Logic: 1

Definition 3.1 A truth assignment is a function τ which assigns to each atom a truth value.

$$\tau : A \rightarrow \{0, 1\}$$

The **truth value** of a proposition ϕ is defined by induction on the structure of propositions as a function

$$\mathcal{T}[\![\cdot]\!]_\tau : \mathcal{P}_0 \rightarrow \{0, 1\}$$

We use $\stackrel{df}{=}$ to denote equality by definition.

Semantics of Propositional Logic: 2

$$\mathcal{T}[\![\perp]\!]_{\tau} \stackrel{df}{=} 0$$

$$\mathcal{T}[\![\top]\!]_{\tau} \stackrel{df}{=} 1$$

$$\mathcal{T}[\![p]\!]_{\tau} \stackrel{df}{=} \tau(p) \text{ for each atom } p$$

$$\mathcal{T}[\![\neg\phi]\!]_{\tau} \stackrel{df}{=} \overline{\mathcal{T}[\![\phi]\!]_{\tau}}$$

$$\mathcal{T}[\![\phi \wedge \psi]\!]_{\tau} \stackrel{df}{=} \mathcal{T}[\![\phi]\!]_{\tau} \cdot \mathcal{T}[\![\psi]\!]_{\tau}$$

$$\mathcal{T}[\![\phi \vee \psi]\!]_{\tau} \stackrel{df}{=} \mathcal{T}[\![\phi]\!]_{\tau} + \mathcal{T}[\![\psi]\!]_{\tau}$$

$$\mathcal{T}[\![\phi \rightarrow \psi]\!]_{\tau} \stackrel{df}{=} \mathcal{T}[\![\phi]\!]_{\tau} \leq \cdot \mathcal{T}[\![\psi]\!]_{\tau}$$

$$\mathcal{T}[\![\phi \leftrightarrow \psi]\!]_{\tau} \stackrel{df}{=} \mathcal{T}[\![\phi]\!]_{\tau} \doteq \mathcal{T}[\![\psi]\!]_{\tau})$$

Models and Satisfiability

Definition 3.2 A truth assignment τ is called a **model** of a formula ϕ (denoted $\tau \Vdash \phi$), if and only if $\mathcal{T}[\![\phi]\!]_{\tau} = 1$. τ is said to satisfy the formula ϕ .

Definition 3.3 A formula is **satisfiable** if it has a model. Otherwise it is said to be **unsatisfiable**. A set S of formulae is **satisfiable** if there is a model τ which satisfies every formula in S (denoted $\tau \Vdash S$).

Fact 3.4 For any finite set $S = \{\phi_i \mid 1 \leq i \leq n\}$ of formulae, $\tau \Vdash S$ if and only if $\tau \Vdash \phi_1 \wedge \dots \wedge \phi_n$.

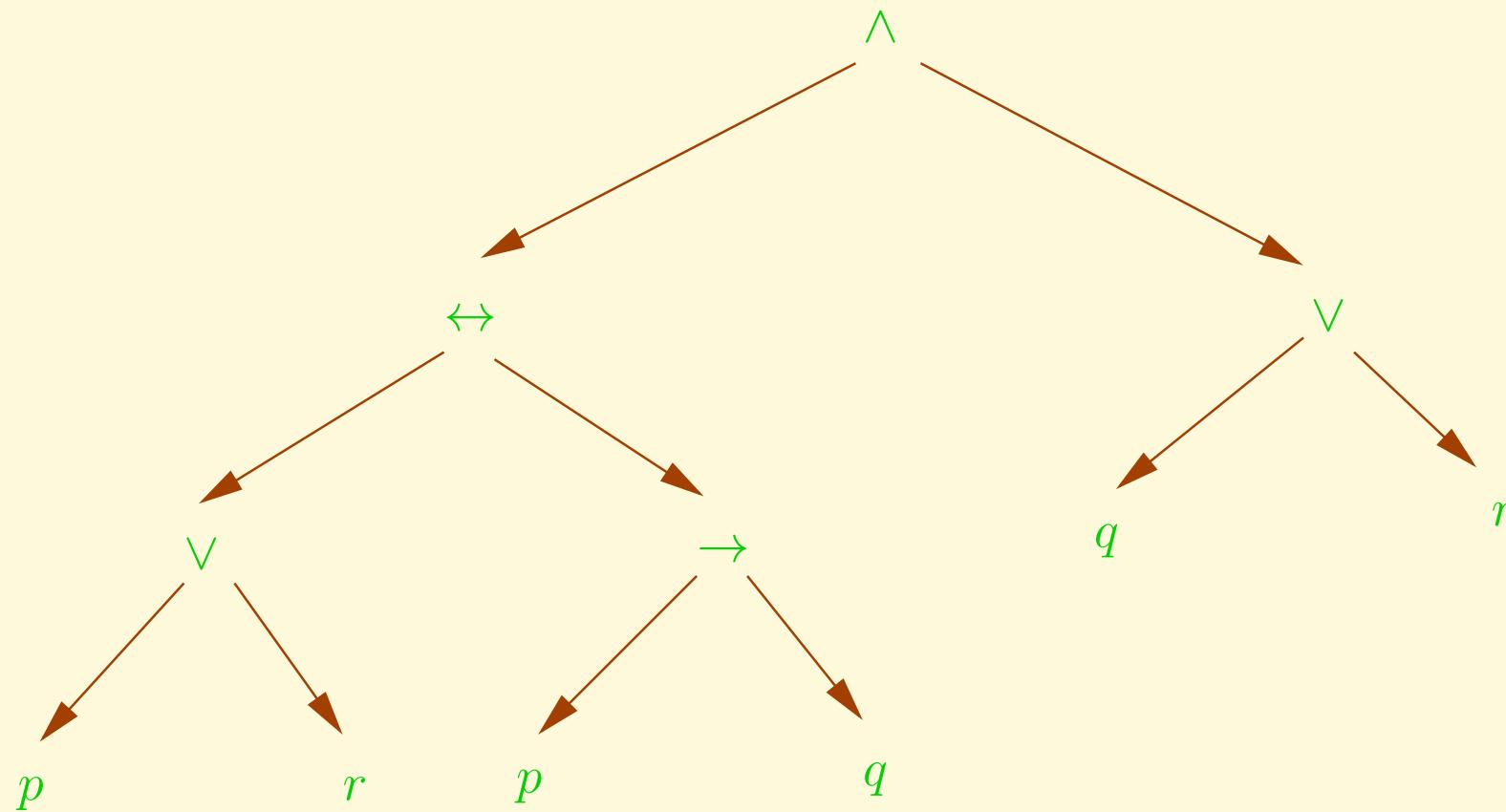
Example: Abstract Syntax trees

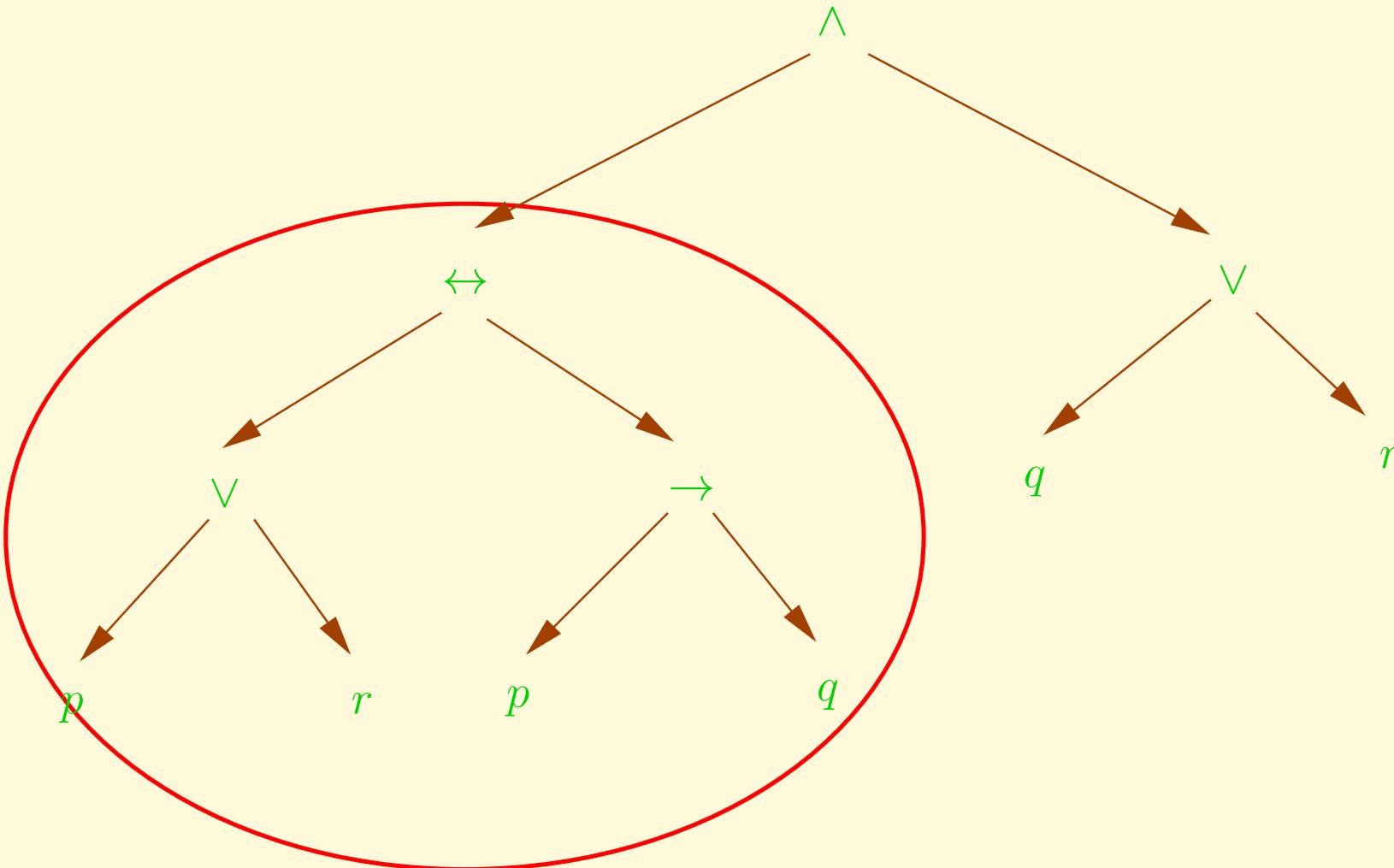
In the following example we illustrate the evaluation of the truth value of the proposition

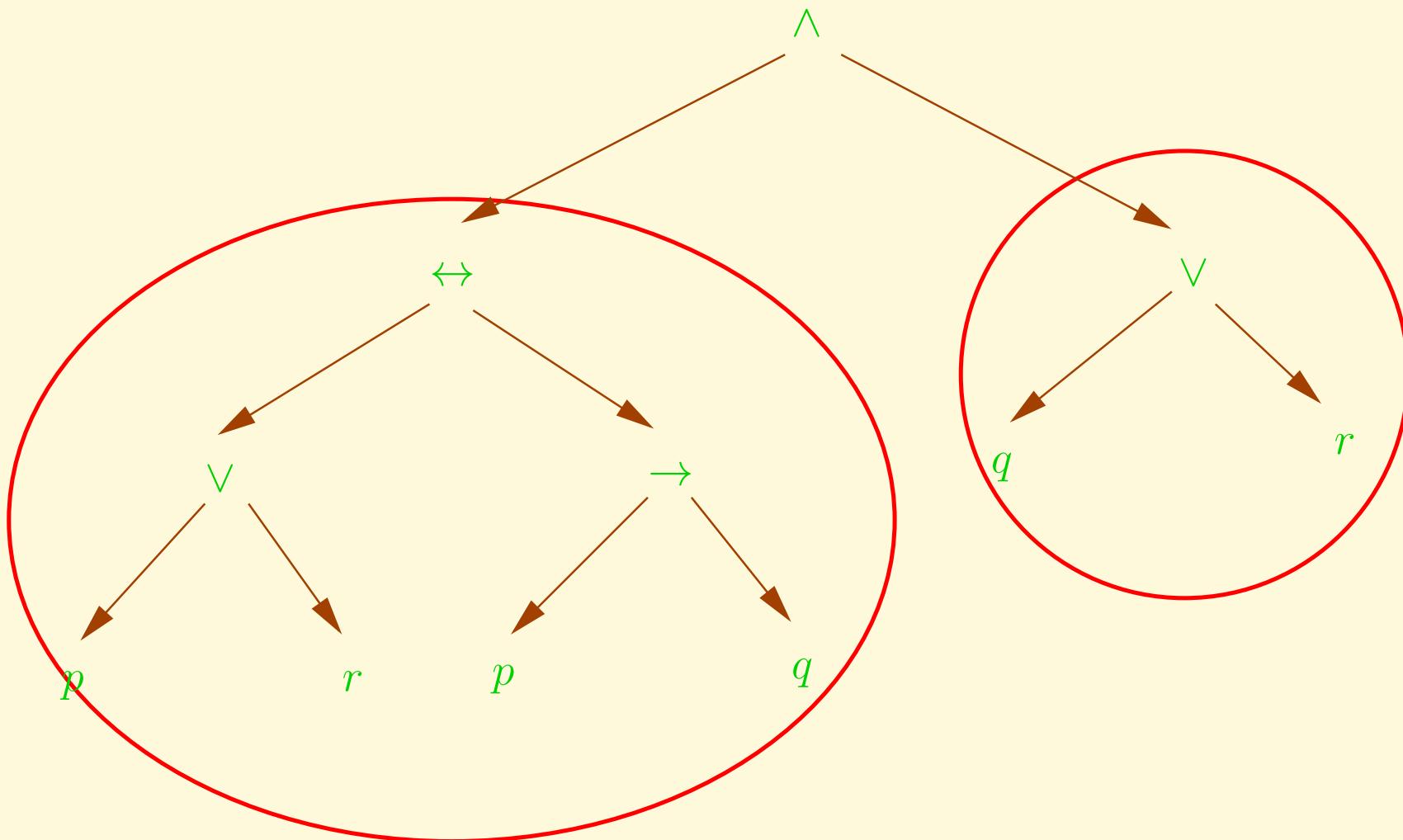
$$(((p \vee r) \leftrightarrow (p \rightarrow q)) \wedge (q \vee r))$$

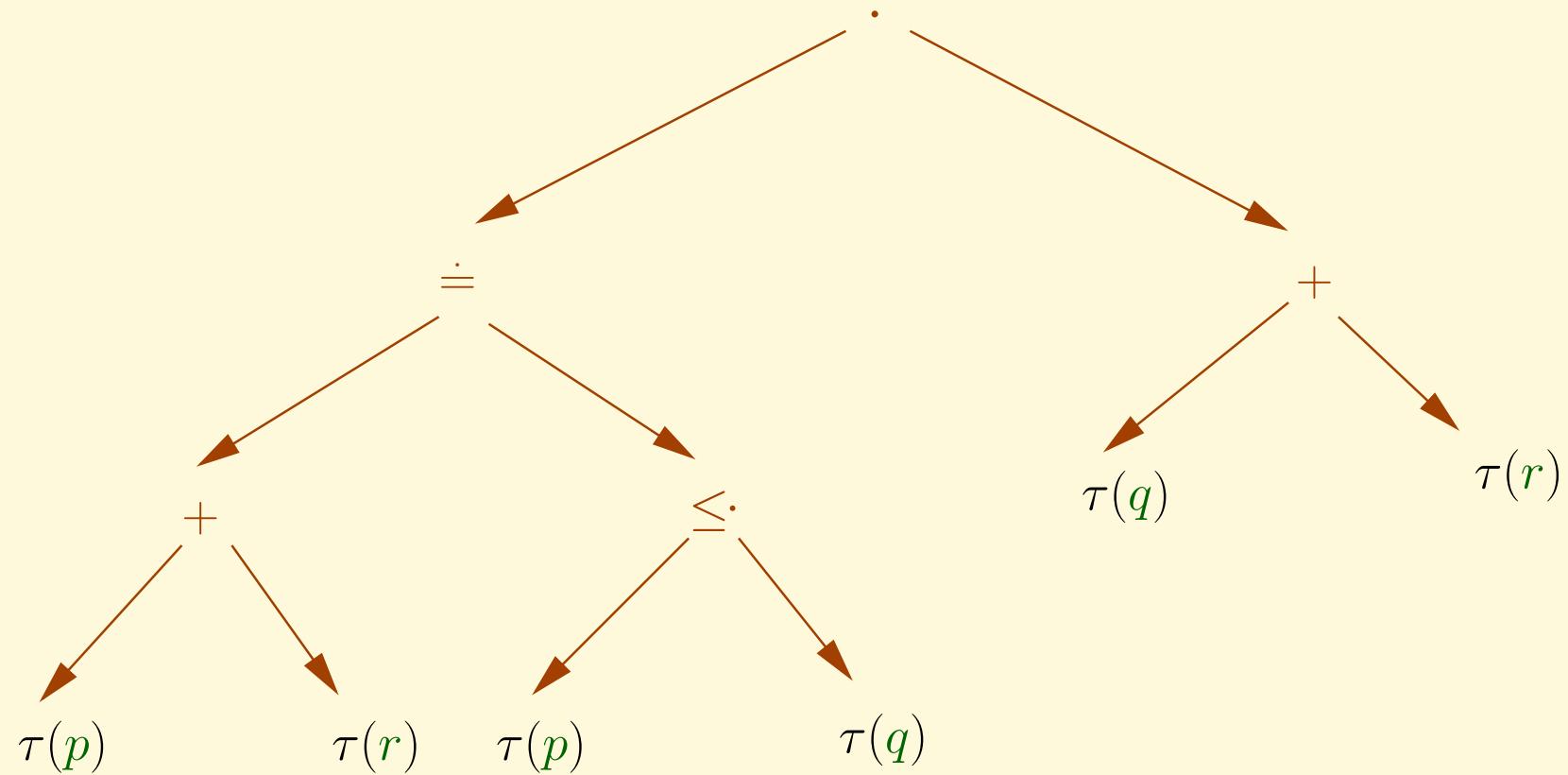
using the **semantics** to

$$(((\tau(p) + \tau(r)) \doteq (\tau(p) \leq \tau(q))).(\tau(q) + \tau(r)))$$









Tautology, Contradiction, Contingent

Definition 3.5 A proposition is said to be a **tautology or logically valid** if it is true under all truth assignments.

contradiction or unsatisfiable if it is false under all truth assignments.

contingent if it is neither a tautology nor a contradiction

- A formula is **satisfiable** if it is not a contradiction.
- A formula is **falsifiable** if it is not a tautology.

Exercise 3.1

1. Prove that for each truth assignment τ , the function $\mathcal{T}[\cdot]_\tau$ is a homomorphism between the algebras $\mathcal{P}_0 = \langle \mathcal{P}_0, \Omega \rangle$ and $\mathcal{B} = \langle \{\textcolor{brown}{0}, \textcolor{brown}{1}\}, \{\textcolor{brown}{\neg}, \textcolor{brown}{.}, \textcolor{brown}{+}, \leq, \dot{=}\} \rangle$.
2. Prove that if two truth assignments τ and τ' are exactly the same for elements in $\text{atoms}(\phi)$ for any formula ϕ , then $\tau \Vdash \phi$ if and only if $\tau' \Vdash \phi$.
3. Any homomorphism $h : \mathcal{A} \rightarrow \mathcal{B}$ from an algebra \mathcal{A} to another algebra \mathcal{B} (thus establishing a 1-1 correspondence between the signatures of the two algebras) induces an equivalence relation on \mathcal{A} . What is the nature of the equivalence relation $=_\tau$ induced by $\mathcal{T}[\cdot]_\tau$ for a truth assignment τ ?

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

181 OF 783

QUIT

4. Lecture 4: Logical and Algebraic Concepts

Lecture 4: Logical and Algebraic Concepts

Tuesday 02 August 2011

1. Logical Consequence: 1
2. Logical Consequence: 2
3. Other Theorems
4. Logical Implication
5. Implication & Equivalence
6. Logical Equivalence as a Congruence

Logical Consequence: 1

Definition 4.1 A proposition $\phi \in \mathcal{P}_0$ is called a **logical consequence** of a set $\Gamma \subseteq \mathcal{P}_0$ of formulas (denoted $\Gamma \models \phi$) if any truth assignment that satisfies all formulas of Γ also satisfies ϕ .

- When $\Gamma = \emptyset$ then logical consequence reduces to **logical validity**.
- $\models \phi$ denotes that ϕ is logically valid.
- $\Gamma \not\models \phi$ denotes that ϕ is not a logical consequence of Γ .
- $\nvDash \phi$ denotes that ϕ is logically invalid.

Logical Consequence: 2

Theorem 4.2 Let $\Gamma = \{\phi_i \mid 1 \leq i \leq n\}$ be a finite set of propositions, and let ψ be any proposition. Then $\Gamma \models \psi$ if and only if $((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi)$ is a tautology.

Proof: (\Rightarrow) Assume $((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi$ is not a tautology. Then there exists a truth assignment τ such that

$$\begin{aligned} \mathcal{T}\llbracket((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi)\rrbracket_{\tau} &= 0 \\ \text{iff } (\mathcal{T}\llbracket(\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n)\rrbracket_{\tau} \leq \mathcal{T}\llbracket\psi\rrbracket_{\tau}) &= 0 \\ \text{iff } (\mathcal{T}\llbracket(\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n)\rrbracket_{\tau}) &= 1 \text{ and } \mathcal{T}\llbracket\psi\rrbracket_{\tau} = 0 \\ \text{iff } \mathcal{T}\llbracket\phi_1\rrbracket_{\tau} = \dots = \mathcal{T}\llbracket\phi_n\rrbracket_{\tau} &= 1 \text{ and } \mathcal{T}\llbracket\psi\rrbracket_{\tau} = 0 \end{aligned}$$

which contradicts the notion of logical consequence.

(\Leftarrow) Assume $((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi$ is a tautology, and suppose $\Gamma \not\models \psi$. Then there exists a truth assignment τ such that

$$\mathcal{T}\llbracket\phi_1\rrbracket_{\tau} = \dots = \mathcal{T}\llbracket\phi_n\rrbracket_{\tau} = 1 \text{ and } \mathcal{T}\llbracket\psi\rrbracket_{\tau} = 0$$

From the previous proof, we obtain

$$\mathcal{T}\llbracket((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi)\rrbracket_{\tau} = 0$$

from which it follows that $((\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \rightarrow \psi)$ is not a tautology, contradicting our assumption. QED ■

The following results may also be proved using the semantics of propositional logic.

Other Theorems

Theorem 4.3 Let $\Gamma = \{\phi_i \mid 1 \leq i \leq n\}$ be a finite set of propositions, and let ψ be any proposition. Then

1. $\Gamma \models \psi$ if and only if $\models \phi_1 \rightarrow (\phi_2 \rightarrow \dots (\phi_n \rightarrow \psi) \dots)$
2. $\Gamma \models \psi$ if and only if $((\dots ((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \wedge \neg\psi)$ is a contradiction.

Corollary 4.4 A formula ϕ is a tautology iff $\neg\phi$ is a contradiction (unsatisfiable).



Logical Implication

Definition 4.5 A formula ϕ logically implies another formula ψ (denoted $\phi \Rightarrow \psi$) iff $\models \phi \rightarrow \psi$. \Rightarrow is called (logical) implication.

Definition 4.6 A formula ϕ is logically equivalent to another formula ψ (denoted $\phi \Leftrightarrow \psi$) iff $\models \phi \leftrightarrow \psi$. \Leftrightarrow is called (logical) equivalence.

Implication & Equivalence

Fact 4.7

1. $\phi \Rightarrow \psi$ iff $\{\phi\} \models \psi$.
2. $\phi \Leftrightarrow \psi$ iff $\phi \Rightarrow \psi$ and $\psi \Rightarrow \phi$.
3. \Rightarrow is a preordering (reflexive and transitive) relation on \mathcal{P}_0 .
4. \Leftrightarrow is the kernel of \Rightarrow i.e. $\Leftrightarrow = \Rightarrow \cap \Rightarrow^{-1}$ and is hence indeed an equivalence relation on \mathcal{P}_0 .



Logical Equivalence as a Congruence

Theorem 4.8 *Logical equivalence is a congruence relation on \mathcal{P}_0 i.e. if $\phi \Leftrightarrow \psi$ then*

- $\neg\phi \Leftrightarrow \neg\psi$ and
- for each $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ and every formula χ we have

$$\begin{aligned}\phi * \chi &\Leftrightarrow \psi * \chi \\ \chi * \phi &\Leftrightarrow \chi * \psi\end{aligned}$$



Exercise 4.1

1. Use the *semantics of propositional logic* to prove theorem 4.3 and corollary 4.4.
2. Prove the facts 4.7.
3. Prove that logical equivalence is indeed a congruence relation on \mathcal{P}_0 .
4. For each truth assignment τ let $=_\tau$ denote the equivalence defined in exercise 3.1. What is the relationship between \Leftrightarrow and $=_\tau$?
5. We may define the notion of a **precongruence** for preorders analogously to the notion of congruence for equivalences, i.e. a preorder is a precongruence if it is preserved under each operator. Is \Rightarrow a precongruence on \mathcal{P}_0 ? If so prove it, otherwise identify the operators which preserve the relation \Rightarrow and for operators which do not preserve the relation \Rightarrow give examples to show that they are not preserved.
6. Prove that $\phi_1, \dots, \phi_n \models \psi$ if and only if for each i , $1 \leq i \leq n$, $\phi_1, \dots, \phi_{i-1}, \phi_{i+1}, \dots, \phi_n, \neg\psi \models \neg\phi_i$.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

192 OF 783

QUIT

Lecture 5: Identities and Normal Forms

Friday 05 August 2011

1. Adequacy
2. Adequacy: Examples
3. Functional Completeness
4. Duality
5. Principle of Duality
6. Negation Normal Forms: 1
7. Negation Normal Forms: 2
8. Conjunctive Normal Forms
9. CNF

Some identities

Negation $\neg\neg\phi \Leftrightarrow \phi$

$\phi \vee \perp$	\Leftrightarrow	ϕ	Identity	$\phi \wedge \top$	\Leftrightarrow	ϕ
$\phi \vee \top$	\Leftrightarrow	\top	Zero	$\phi \wedge \perp$	\Leftrightarrow	\perp
$\phi \vee \phi$	\Leftrightarrow	ϕ	Idempotence	$\phi \wedge \phi$	\Leftrightarrow	ϕ
$\phi \vee \psi$	\Leftrightarrow	$\psi \vee \phi$	Commutativity	$\phi \wedge \psi$	\Leftrightarrow	$\psi \wedge \phi$
$(\phi \vee \psi) \vee \chi$	\Leftrightarrow	$\phi \vee (\psi \vee \chi)$	Associativity	$(\phi \wedge \psi) \wedge \chi$	\Leftrightarrow	$\phi \wedge (\psi \wedge \chi)$
$\phi \vee (\psi \wedge \chi)$	\Leftrightarrow	$(\phi \vee \psi) \wedge (\phi \vee \chi)$	Distributivity	$\phi \wedge (\psi \vee \chi)$	\Leftrightarrow	$(\phi \wedge \psi) \vee (\phi \wedge \chi)$
$\neg(\phi \vee \psi)$	\Leftrightarrow	$\neg\phi \wedge \neg\psi$	De Morgan	$\neg(\phi \wedge \psi)$	\Leftrightarrow	$\neg\phi \vee \neg\psi$
$\phi \vee \neg\phi$	\Leftrightarrow	\top	Simplification	$\phi \wedge \neg\phi$	\Leftrightarrow	\perp
$\neg\perp$	\Leftrightarrow	\top	Inversion	$\neg\top$	\Leftrightarrow	\perp
$\phi \vee (\phi \wedge \psi)$	\Leftrightarrow	ϕ	Absorption	$\phi \wedge (\phi \vee \psi)$	\Leftrightarrow	ϕ

Adequacy

Some Other Important identities are:

$$\phi \leftrightarrow \psi \Leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \quad (5)$$

$$\phi \rightarrow \psi \Leftrightarrow \neg\phi \vee \psi \quad (6)$$

Definition 5.1 A set of operators $O \subseteq \Omega$ is said to be **adequate** for propositional logic, if for every formula in \mathcal{P}_0 there is a logically equivalent formula using only the operators in O .

Adequacy: Examples

Example 5.2

1. From the identities (5) and (6) and the two *Simplification* identities it is clear that $O = \{\neg, \wedge, \vee\}$ is an adequate set of operators for \mathcal{P}_0 .
2. Further given that $O = \{\neg, \wedge, \vee\}$ is adequate and using the *De Morgan* identity and *Negation*, we have that

$$\phi \wedge \psi \Leftrightarrow \neg\neg(\phi \wedge \psi) \Leftrightarrow \neg(\neg\phi \vee \neg\psi)$$

and hence $\{\neg, \vee\}$ is an adequate set.

3. We may use the other *De Morgan* identity

$$\phi \vee \psi \Leftrightarrow \neg\neg(\phi \vee \psi) \Leftrightarrow \neg(\neg\phi \wedge \neg\psi)$$

to conclude that $\{\neg, \wedge\}$ is adequate.

Functional Completeness

It is quite possible that one could extend \mathcal{P}_0 with new operators (perhaps 3-ary or 4-ary or indeed of any arity) and thus make the language more *expressive*.

Definition 5.3 A set O of operators for propositional logic is **functionally complete** (also called **expressively adequate**) if any formula built up using the operators of O is logically equivalent to a formula using operators only from Ω .

Lemma 5.4 $\{\neg, \wedge, \vee\}$ is a functionally complete set.

Proof

Proof of lemma 5.4.

Proof: By the semantics of propositional logic, every operator of propositional logic corresponds to an operator of the same arity on the boolean set $\{0, 1\}$. The proof follows from the construction of truth tables in the boolean algebra \mathcal{B} , since every truth table may be expressed using the boolean operators $\{\neg, ., +\}$.

	a_1	\cdots	a_n	b
0	0	\cdots	0	b_0
\vdots	\vdots	\vdots	\vdots	\vdots
\vdots	\vdots	\vdots	\vdots	\vdots
\vdots	\vdots	\vdots	\vdots	\vdots
$2^n - 1$	1	\cdots	1	$b_{2^n - 1}$

Let $o_n : \mathcal{D}^n \longrightarrow \mathcal{D}$ be any n -ary operator on boolean values. Typically there exists a truth table for the function $o_n(a_1, \dots, a_n)$, which defines for each possible set of boolean values of the arguments the boolean value of $o_n(a_1, \dots, a_n) = b$. This truth table consists of 2^n rows and $n + 1$ columns as shown below where $b_0, \dots, b_{2^n - 1} \in \mathcal{D}$ (we have numbered the rows by the decimal equivalent of the binary number that the bit-vector (a_1, \dots, a_n) denotes in each case). For $0 \leq i \leq 2^n$, $1 \leq j \leq n$, let $a_{i_j} \in \mathcal{D}$ be the value assigned to variable a_j such that $b_i = o_n(a_{i_1}, \dots, a_{i_n})$. We may express the function as

$$o_n(a_1, \dots, a_n) = \sum_{0 \leq r \leq 2^n - 1} \left(\prod_{1 \leq j \leq n} a_{r_j}^* \right)^*$$

where for each row r , and each j , $1 \leq j \leq n$, $a_{r_j}^* = 1$ if $a_{r_j} = 1$ and $a_{r_j}^* = 0$ otherwise. Similarly

the product $(\prod_{1 \leq j \leq n} a_j^*)$ is inverted if $b_r = 0$ otherwise $(\prod_{1 \leq j \leq n} a_j^*)^* = 1$. Alternatively we have

$$o_n(a_1, \dots, a_n) = \sum_{b_r=1, 0 \leq r < 2^n} \left(\prod_{a_{r_j}=1, 1 \leq j \leq n} a_{r_j} \right)$$

QED



Duality

Definition 5.5

- Two formulas ϕ and ψ are called **duals** of each other if each can be obtained from the other by simultaneously replacing all occurrences of
 - \wedge by \vee ,
 - \vee by \wedge ,
 - \perp by \top and
 - \top by \perp
- \wedge and \vee are duals of each other and
- \top and \perp are duals of each other

Principle of Duality

Theorem 5.6 If $\text{atoms}(\phi) = \{p_1, \dots, p_n\}$ and $\phi \equiv o(p_1, \dots, p_n)$ then

$$\neg\phi \Leftrightarrow o^*(\neg p_1, \dots, \neg p_n)$$

where o^* is the dual of o .

Proof: By structural induction and the use of the De Morgan and Simplification laws. QED

Negation Normal Forms: 1

The adequacy and functional completeness of the set $\{\neg, \wedge, \vee\}$ may be used to build normal forms (or standard forms) by using the logical equivalences as rewrite rules.

Definition 5.7

- A **literal** is an atom ($p \in A$) or its negation ($\neg p$ for $p \in A$). Atoms are called **positive literals** and their negations are called **negative literals**.
- A formula is in **negation normal form** if it is built up from literals using only the operators \vee and \wedge .

Negation Normal Forms: 2

Lemma 5.8 *Every formula in \mathcal{P}_0 is logically equivalent to one in negation normal form.*

Proof: It suffices to consider only formulas containing the operators \neg , \wedge and \vee , and for every occurrence of negation to push it inward using the **De Morgan** identities. QED ■

Conjunctive Normal Forms

Definition 5.9

- A **disjunction of literals** is a formula δ of the form

$$\delta \equiv \lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_m \equiv \bigvee_{1 \leq i \leq m} \lambda_i$$

where $m \geq 0$.

- A **conjunctive normal form** is a formula γ of the form $\delta_1 \wedge \delta_2 \wedge \cdots \wedge \delta_n$ where δ_j for each $1 \leq j \leq n$ is a disjunction of literals.

We may analogously define a *conjunction of literals* and a *disjunctive normal form (DNF)*.

CNF

Theorem 5.10 *Every formula in \mathcal{P}_0 is logically equivalent to a conjunctive normal form.*

Proof: It suffices to consider only negation normal forms. In each case use the **distributive laws** to distribute \vee over \wedge and use the **negation law** to remove multiple contiguous occurrences of negations. QED ■

Exercise 5.1

1. Prove that the following sets are adequate for \mathcal{P}_0 .

- (a) $\{\rightarrow, \neg\}$
- (b) $\{\rightarrow, \perp\}$

2. Prove that $\{\wedge, \vee\}$ is not an adequate set.

3. Prove that if $O \subseteq \Omega$ then $\neg \in O$ or $\perp \in O$.

4. Define BNFs to generate exactly

- (a) the set \mathcal{N}_0 of negation normal forms
- (b) the set \mathcal{C}_0 of conjunctive normal forms
- (c) the set \mathcal{D}_0 of disjunctive normal forms.

5. Using principles of induction prove that

- (a) $\mathcal{C}_0 \subset \mathcal{N}_0 \subset \mathcal{P}_0$,
- (b) $\mathcal{D}_0 \subset \mathcal{N}_0 \subset \mathcal{P}_0$,

6. Let $\phi \equiv o_1(p_1, \dots, p_n)$ and $\psi \equiv o_2(p_1, \dots, p_n)$ be formulas such that $\phi \Leftrightarrow \psi$. Then prove that

$$\begin{aligned}o_1(\neg p_1, \dots, \neg p_n) &\Leftrightarrow o_2(\neg p_1, \dots, \neg p_n) \\ \neg o_1^*(p_1, \dots, p_n) &\Leftrightarrow \neg o_2^*(p_1, \dots, p_n)\end{aligned}$$

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

209 OF 783

[QUIT](#)

[6. Lecture 6: Tautology Checking](#)

Lecture 6: Tautology Checking

Tuesday 09 August 2011

1. Arguments
2. Arguments: 2
3. Validity & Falsification
4. Translation into propositional Logic
5. Atoms in Argument
6. The Representation
7. Propositional Rendering
8. The Strategy
9. Checking Tautology
10. Computing the CNF
11. Falsifying CNF

Arguments

A typical informally stated argument might go as follows:

If prices rise, then the poor and the salaried class will be unhappy.

If taxes are increased then the businessmen will be unhappy.

If the poor and the salaried class or the businessmen are unhappy, the Government will not be re-elected.

Inflation will rise if Government expenditure exceeds its revenue.

Government expenditure will exceed its revenue unless taxes are increased or the Government resorts to deficit financing or takes a loan from the IMF to cover the deficit.

If the Government resorts to deficit financing then inflation will rise.

If inflation rises, the prices will also rise.

The Government will get reelected.

Therefore the Government will take a loan from the IMF.

Arguments: 2

A typical informally stated argument might go as follows:

If prices rise, then the poor and the salaried class will be unhappy.

If taxes are increased then the businessmen will be unhappy.

If the poor and the salaried class are unhappy or the businessmen are unhappy,
the Government will not be re-elected.

Inflation will rise if Government expenditure exceeds its revenue.

Government expenditure will exceed its revenue unless taxes are increased or
the Government resorts to deficit financing or takes a loan from the IMF to cover the deficit.

If the Government resorts to deficit financing then inflation will rise.

If inflation rises, the prices will also rise.

The Government will get reelected.

Therefore the Government will take a loan from the IMF.

Arguments in natural language

It turns out that the correctness or otherwise of most arguments depends entirely on the “shapes” of the formulae concerned rather than their intrinsic meaning. Take the previous argument. Suppose we uniformly replace the various atoms by say sentences from nursery rhymes as the following table shows:

Prices rise

The poor and . . .

Taxes are increased

The businessmen will be unhappy

The Government will get re-elected

Inflation will rise

Government expenditure . . . revenue

The Government resorts . . .

The Government takes a loan . . . deficit

Mary has a little lamb

Little Bo-Peep loses her sheep

Jack and Jill go up the hill

Humpty-Dumpty sits on the wall

I am little teapot

Little Jack Horner sits in a corner

The boy stands on the burning deck

Wee Willie Winkie runs through the town

Eensy Weensy spider climbs up the water spout

Then we get the following ridiculous sounding argument

If Mary has a little lamb, then Little Bo-Peep loses her sheep.

If Jack and Jill go up the hill, then Humpty-Dumpty sits on a wall.

If Little Bo-Peep loses her sheep or Humpty-Dumpty sits on a wall, then Little Miss Muffet sits on a tuffet.

Little Jack Horner sits in a corner if the boy stands on the burning deck.

The boy stands on the burning deck unless Jack and Jill go up the hill or Wee Willie Winkie runs through the town or Eensy Weensy spider climbs up the water spout

If Wee Willie Winkie runs through the town then Little Jack Horner sits in a corner

If Little Jack Horner sits in a corner then Mary has a little lamb.

Little Miss Muffet sits on a tuffet.

Therefore Eensy Weensy spider climbs up the water spout.

But if the original argument is logically valid then so is the new one, since logical validity depends entirely on the so called connectives that make up the propositions and their effect on truth values.

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.

Translation into propositional Logic

But even after translating the argument into a suitable propositional logic form, it would be quite impossible to verify the validity of the argument by truth table, since the number of “atomic” propositions could be very large.

Atoms in Argument

The Argument

```
val rise = ATOM "Prices rise";  
val pandsun = ATOM "The poor and ... will be unhappy";  
val taxes = ATOM "Taxes are increased";  
val busun = ATOM "The businessmen will be unhappy";  
val reelect = ATOM "The Government will be re-elected";  
val inflation = ATOM "Inflation will rise";  
val exceeds = ATOM "Government expenditure ... revenue";  
val deffin = ATOM "The Government resorts ...";  
val imf = ATOM "The Govt. takes a loan ... deficit";
```

In this case the the truth table would have $2^9 = 512$ rows. Further, the truth table will use *all* the atoms, even the irrelevant ones.

The Representation

```
datatype Prop = ATOM of string
              | NOT of Prop
              | AND of Prop * Prop
              | OR of Prop * Prop
              | IMP of Prop * Prop
              | EQL of Prop * Prop
```

Propositional Rendering

The Argument

```
val hyp1 = IMP (rise , pandsun);  
val hyp2 = IMP (taxes , busun);  
val hyp3 = IMP (OR (pandsun , busun) , NOT(reelect));  
val hyp4 = IMP (exceeds , inflation);  
val hyp5 = IMP (exceeds , NOT(OR(taxes , OR(deffin , imf))));  
val hyp6 = IMP (deffin , inflation);  
val hyp7 = IMP (inflation , rise);  
val hyp8 = reelect;  
val conc1 = imf;  
val H = [hyp1 , hyp2 , ... , hyp8];  
val Arg1 = (H, conc1);
```

Atoms in Argument

The Strategy

We need to either show that

$$Arg1 = IMP (\text{bigAND } H, \text{conc1})$$

is a tautology or can be falsified. Using theorem 4.2 for validity

```
val bigAND = leftReduce (AND);
fun Valid ((H, P):Argument) =
  if null (H) then tautology (P)
  else tautology (IMP (bigAND (H), P))
```

and for falsification we use

```
fun falsifyArg ((H, P): Argument) =
  if null (H) then falsify (cnf(P))
  else falsify (cnf (IMP (bigAND (H), P)))
```

Checking Tautology

Checking for tautology crucially involves finding falsifying truth assignments for at last one of the conjuncts in the CNF of the argument.

```
fun tautology2 (P) =
  let val Q = cnf (P);
      val LL = falsify (Q)
  in   if null (LL) then (true , [])
        else (false , LL)
  end
```

Computing the CNF

1. Rewrite implications and equivalences
2. Convert to NNF by pushing negation inward
3. Distribute *OR* over *AND*

cnflistlist.sml

```
(*===== THE SIGNATURE PropLogic =====
signature PropLogic =
sig
  exception Atom_exception
  datatype Prop =
    ATOM of string |
    NOT of Prop |
    AND of Prop * Prop |
    OR of Prop * Prop |
    IMP of Prop * Prop |
    EQL of Prop * Prop
  type Argument = Prop list * Prop
  val show : Prop -> unit
  val showArg : Argument -> unit
  val falsifyArg : Argument -> Prop list list
  val Valid : Argument -> bool * Prop list list
end;

(* Propositional formulas *)

===== THE STRUCTURE PL ===== *)
(* structure PL:PropLogic = *)
structure PL = (* This is for debugging purposes only *)
struct

  datatype Prop =
```

```

ATOM of string
NOT of Prop
AND of Prop * Prop
OR of Prop * Prop
IMP of Prop * Prop
EQL of Prop * Prop
;

(* ----- Propositions to CNFs ----- *)

```

```

exception Atom_exception;
fun newatom (s) = if s = "" then raise Atom_exception
                  else (ATOM s);
fun drawChar (c, n) =
    if n>0 then (print(str(c)); drawChar(c, (n-1)))
    else ();
fun show (P) =
    let fun drawTabs (n) = drawChar (#"\t", n);
          fun showTreeTabs (ATOM a, n) = (drawTabs (n);
                                              print (a);
                                              print ("\n")
                                         )
         | showTreeTabs (NOT (P), n) = (drawTabs(n); print ("NOT");
                                         showTreeTabs (P, n+1)
                                         )
         | showTreeTabs (AND (P, Q), n) =
             (showTreeTabs (P, n+1);
              drawTabs (n); print("AND\n"));

```

```

        showTreeTabs (Q, n+1)
    )
| showTreeTabs (OR (P, Q), n) =
( showTreeTabs (P, n+1);
drawTabs (n); print("OR\n");
showTreeTabs (Q, n+1)
)

| showTreeTabs (IMP (P, Q), n) =
( showTreeTabs (P, n+1);
drawTabs (n); print("IMPLIES\n");
showTreeTabs (Q, n+1)
)
| showTreeTabs (EQL (P, Q), n) =
( showTreeTabs (P, n+1);
drawTabs (n); print("IFF\n");
showTreeTabs (Q, n+1)
)
;

in ( print ("\n"); showTreeTabs(P, 0); print ("\n"))

end
;

```

(* The function below evaluates a formula given a truth assignment.
The truth assignment is given as a list of atoms that are true
(all other atoms are false).

*)

```
fun lookup (x:Prop, []) = false
| lookup (x, h::L)    = (x = h) orelse lookup (x, L)
;

fun eval (ATOM a, L)      = lookup (ATOM a, L)
| eval (NOT (P), L)     = if eval (P, L) then false else true
| eval (AND (P, Q), L)  = eval (P, L) andalso eval (Q, L)
| eval (OR (P, Q), L)   = eval (P, L) orelse eval (Q, L)
| eval (IMP (P, Q), L)  = eval (OR (NOT (P), Q), L)
| eval (EQL (P, Q), L)  = (eval (P, L) = eval (Q, L))
;
```

(* We could also write a tautology checker with out using truth assignments by first converting everything into a normal form.
*)

(* First rewrite implications and equivalences *)

```
fun rewrite (ATOM a)      = ATOM a
| rewrite (NOT (P))     = NOT (rewrite (P))
| rewrite (AND (P, Q))  = AND (rewrite (P), rewrite (Q))
| rewrite (OR (P, Q))   = OR (rewrite (P), rewrite (Q))
| rewrite (IMP (P, Q))  = OR (NOT (rewrite (P)), rewrite (Q))
| rewrite (EQL (P, Q))  = rewrite (AND (IMP (P, Q), IMP (Q, P)))
;
```

(* Convert all formulas not containing IMP or EQL into Negation Normal Form .
*)

```
fun nnf (ATOM a)          = ATOM a
| nnf (NOT (ATOM a))     = NOT (ATOM a)
| nnf (NOT (NOT (P)))   = nnf (P)
| nnf (AND (P, Q))       = AND (nnf(P), nnf(Q))
| nnf (NOT (AND (P, Q))) = nnf (OR (NOT (P), NOT (Q)))
| nnf (OR (P, Q))        = OR (nnf(P), nnf(Q))
| nnf (NOT (OR (P, Q)))  = nnf (AND (NOT (P), NOT (Q)))
;
;
```

(* Distribute OR over AND to get a NNF into CNF *)

```
fun distOR (P, AND (Q, R)) = AND (distOR (P, Q), distOR (P, R))
| distOR (AND (Q, R), P)  = AND (distOR (Q, P), distOR (R, P))
| distOR (P, Q)           = OR (P, Q)
```

(* Now the CNF can be easily computed *)

```
fun conj_of_disj (AND (P, Q)) = AND (conj_of_disj (P), conj_of_disj (Q))
| conj_of_disj (OR (P, Q))   = distOR (conj_of_disj (P), conj_of_disj (Q))
| conj_of_disj (P)          = P
;
;
```

```
fun cnf (P) = conj_of_disj (nnf (rewrite (P)));
```

```

(* ----- Propositions to CNFs ends ----- *)

(* ----- CNFs to lists of lists of literals ----- *)

(* Convert a clause into a list of literals *)
fun flattenOR (OR (A, B)) = (flattenOR A) @ (flattenOR B)
| flattenOR C             = [C] (* assuming C is a literal *)

(* Convert a CNF into a list of lists of clauses *)
fun flattenAND (AND (Q, R)) = (flattenAND Q) @ (flattenAND R)
| flattenAND (P)           = [flattenOR P] (* assuming P is a clause *)

(* Sort the litListList using some ordering and remove duplicates while sorting *)

(* Define an ordering litLess on literals: *)

fun litLess (ATOM (a), ATOM (b))      = a < b (* lexicographic *)
| litLess (NOT(ATOM a), NOT(ATOM b)) = a < b
  (* every negative literal is smaller than every positive literal *)
| litLess (NOT(ATOM a), ATOM (b))    = true
| litLess (ATOM (a), NOT(ATOM b))    = false

(* Extend the ordering to lists of literals *)

fun clauseLess ([], [])            = false
| clauseLess ([], _)              = true
| clauseLess (_ , [])            = false

```

```

| clauseLess (h1::T1, h2::T2) =
  (litLess (h1, h2)) orelse
  ((h1=h2) andalso clauseLess (T1, T2))

(* Define mergeSortRD to remove duplicates as sorting proceeds *)

fun mergeSortRD R [] = []
| mergeSortRD R [h] = [h]
| mergeSortRD R L = (* can't split a list unless it has > 1 element *)
let fun split [] = ([], [])
| split [h] = ([h], [])
| split (h1::h2:::t) =
    let val (left, right) = split t;
        in (h1::left, h2::right)
    end;
val (left, right) = split L;
fun mergeRD (R, [], []) = []
| mergeRD (R, [], L2) = L2
| mergeRD (R, L1, []) = L1
| mergeRD (R, (L1 as h1::t1), (L2 as h2::t2)) =
    if h1=h2 then mergeRD (R, t1, L2) (* remove a copy *)
    else if R(h1, h2) then h1::(mergeRD (R, t1, L2))
    else h2::(mergeRD (R, L1, t2));
val sortedLeft = mergeSortRD R left;
val sortedRight = mergeSortRD R right;
in mergeRD (R, sortedLeft, sortedRight)
end;

```

```

(* Now sort the list of lists of literals removing duplicates *)

fun sortRD LL = (* First sort each clause and then the list of clauses *)
  let val sortedClauses = map (mergeSortRD litLess) LL
  in mergeSortRD clauseLess sortedClauses
  end;

(* Putting everything together *)

fun prop2listlist P = sortRD (flattenAND (cnf P))
end (* struct *);

open PL;

(* Testing prop2listlist =====
- val god = ATOM "There is a God";
val god = ATOM "There is a God" : Prop
- val oscient = ATOM "God is omniscient";
val opotent = ATOM "God is omnipotent";
val evil = ATOM "There is Evil";
val know = ATOM "God knows there is Evil";
val prevent = ATOM "God prevents Evil";

val hy1 = IMP (god, AND (oscient, opotent));
val hy2 = IMP (oscient, know);
val hy3 = IMP (opotent, prevent);
val hy4 = evil;
val conc = NOT (god);

```

```

val oscient = ATOM "God is omniscient" : Prop
- GC #0.0.0.1.7.187: (1 ms)
val opotent = ATOM "God is omnipotent" : Prop
- val evil = ATOM "There is Evil" : Prop
- val know = ATOM "God knows there is Evil" : Prop
- val prevent = ATOM "God prevents Evil" : Prop
-- val hy1 = IMP (ATOM "There is a God",AND (ATOM #,ATOM #)) : Prop
- val hy2 = IMP (ATOM "God is omniscient",ATOM "God knows there is Evil") : Prop
- val hy3 = IMP (ATOM "God is omnipotent",ATOM "God prevents Evil") : Prop
- val hy4 = ATOM "There is Evil" : Prop
- val conc = NOT (ATOM "There is a God") : Prop
-- prop2listlist hy1;
val it =
  [[NOT (ATOM "There is a God"),ATOM "God is omnipotent"],
   [NOT (ATOM "There is a God"),ATOM "God is omniscient"]]: Prop list list
- prop2listlist hy2;
val it = [[NOT (ATOM "God is omniscient"),ATOM "God knows there is Evil"]]
: Prop list list
- val andhyp = AND (hy1, AND (hy2, AND(hy3, hy4)));
val andhyp = AND (IMP (ATOM #,AND #),AND (IMP #,AND #)) : Prop
- prop2listlist andhyp;
val it =
  [[NOT (ATOM "God is omnipotent"),ATOM "God prevents Evil"],
   [NOT (ATOM "God is omniscient"),ATOM "God knows there is Evil"],
   [NOT (ATOM "There is a God"),ATOM "God is omnipotent"],
   [NOT (ATOM "There is a God"),ATOM "God is omniscient"],
   [ATOM "There is Evil"]]: Prop list list

```

```

- val a = ATOM "a";
val a = ATOM "a" : Prop
- val b = ATOM "b";
val b = ATOM "b" : Prop
- val c = ATOM "c";
val c = ATOM "c" : Prop
- val one = IMP (a, OR (a, a));
val one = IMP (ATOM "a",OR (ATOM #,ATOM #)) : Prop
- val two = EQL(b, OR(b, NOT(b)));
val two = EQL (ATOM "b",OR (ATOM #,NOT #)) : Prop
- val three = EQL(AND(a, a), OR(NOT(a), OR(a, NOT(a))));
GC #0.0.0.1.8.221: (1 ms)
val three = EQL (AND (ATOM #,ATOM #),OR (NOT #,OR #)) : Prop
- val p211 = prop2listlist;
val p211 = fn : Prop -> Prop list list
- p211 one;
val it = [[NOT (ATOM "a"),ATOM "a"]]: Prop list list
- p211 two;
val it = [[NOT (ATOM "b"),ATOM "b"],[ATOM "b"]]: Prop list list
- p211 three;
val it = [[NOT (ATOM "a"),ATOM "a"],[ATOM "a"]]: Prop list list
- p211 (OR(one, OR(two, three)));
val it =
  [[NOT (ATOM "a"),NOT (ATOM "b"),ATOM "a",ATOM "b"],
   [NOT (ATOM "a"),ATOM "a",ATOM "b"]]: Prop list list
-
===== *)
```

Falsifying CNF

1. Suffices to find a falsification of at least one conjunct
2. A conjunct in the CNF can be false iff all the disjuncts in it are false.
3. A disjunct is false iff it does not contain a “complementary pair”.

Assume the CNF is $Q \equiv \bigwedge_{i=1}^m D_i$ where each $D_i \equiv \bigvee_{j=1}^{n_i} L_{ij}$ where the literals of $D_i = P_i \cup N_i$ where P_i is the set of positive literals (atoms) and N_i consists of the atoms appearing as negative literals.

Then D_i is false iff $P_i \cap N_i = \emptyset$.

tautology1.sml

```
signature PropLogic =
sig
  exception Atom_exception
  datatype Prop =
    ATOM of string
    NOT of Prop
    AND of Prop * Prop
    OR of Prop * Prop
    IMP of Prop * Prop
    EQL of Prop * Prop
  type Argument = Prop list * Prop
  val show : Prop -> unit
  val showArg : Argument -> unit
  val falsifyArg : Argument -> Prop list list
  val Valid : Argument -> bool * Prop list list
end;

(* Propositional formulas *)

structure PL:PropLogic =
(* structure PL = *) (* This is for debugging purposes only *)
struct

  datatype Prop =
    ATOM of string
    |
```

```

NOT of Prop
AND of Prop * Prop
OR of Prop * Prop
IMP of Prop * Prop
EQL of Prop * Prop
;

exception Atom_exception;
fun newatom (s) = if s = "" then raise Atom_exception
                  else (ATOM s);
fun drawChar (c, n) =
    if n>0 then (print(str(c)); drawChar(c, (n-1)))
    else ();
fun show (P) =
    let fun drawTabs (n) = drawChar (#"\t", n);
          fun showTreeTabs (ATOM a, n) = (drawTabs (n);
                                              print (a);
                                              print ("\n"))
          | showTreeTabs (NOT (P), n) = (drawTabs(n); print ("NOT");
                                           showTreeTabs (P, n+1)
                                         )
          | showTreeTabs (AND (P, Q), n) =
            (showTreeTabs (P, n+1);
             drawTabs (n); print("AND\n");
             showTreeTabs (Q, n+1)
           )
          | showTreeTabs (OR (P, Q), n) =

```

```

        ( showTreeTabs (P, n+1);
        drawTabs (n); print("OR\n");
        showTreeTabs (Q, n+1)
      )

|   showTreeTabs (IMP (P, Q), n) =
  ( showTreeTabs (P, n+1);
    drawTabs (n); print("IMPLIES\n");
    showTreeTabs (Q, n+1)
  )
|   showTreeTabs (EQL (P, Q), n) =
  ( showTreeTabs (P, n+1);
    drawTabs (n); print("IFF\n");
    showTreeTabs (Q, n+1)
  )
;
in  ( print ("\n"); showTreeTabs(P, 0); print ("\n"))
end
;

(* The function below evaluates a formula given a truth assignment.
The truth assignment is given as a list of atoms that are assigned
"true" (implicitly all other atoms are assumed to have been
assigned "false").
*)

```

```

fun lookup (x:Prop, []) = false
| lookup (x, h::L) =
  if (x = h) then true
  else lookup (x, L)
;

fun eval (ATOM a, L) = lookup (ATOM a, L)
| eval (NOT (P), L) = if eval (P, L) then false else true
| eval (AND (P, Q), L) = eval (P, L) andalso eval (Q, L)
| eval (OR (P, Q), L) = eval (P, L) orelse eval (Q, L)
| eval (IMP (P, Q), L) = eval (OR (NOT (P), Q), L)
| eval (EQL (P, Q), L) = (eval (P, L) = eval (Q, L))
;

(* We first convert every proposition into a normal form.
*)

(* First rewrite implications and equivalences *)

fun rewrite (ATOM a)          = ATOM a
| rewrite (NOT (P))         = NOT (rewrite (P))
| rewrite (AND (P, Q))      = AND (rewrite(P), rewrite(Q))
| rewrite (OR (P, Q))       = OR (rewrite(P), rewrite(Q))
| rewrite (IMP (P, Q))      = OR (NOT (rewrite(P)), rewrite(Q))
| rewrite (EQL (P, Q))      = rewrite (AND (IMP(P, Q), IMP (Q, P)))
;

```

(* Convert all formulas not containing IMP or EQL into Negation Normal Form.

*)

```
fun nnf (ATOM a)      = ATOM a
| nnf (NOT (ATOM a)) = NOT (ATOM a)
| nnf (NOT (NOT (P))) = nnf (P)
| nnf (AND (P, Q))    = AND (nnf(P), nnf(Q))
| nnf (NOT (AND (P, Q))) = nnf (OR (NOT (P), NOT (Q)))
| nnf (OR (P, Q))     = OR (nnf(P), nnf(Q))
| nnf (NOT (OR (P, Q))) = nnf (AND (NOT (P), NOT (Q)))
;
;
```

(* Distribute OR over AND to get a NNF into CNF *)

```
fun distOR (P, AND (Q, R)) = AND (distOR (P, Q), distOR (P, R))
| distOR (AND (Q, R), P) = AND (distOR (Q, P), distOR (R, P))
| distOR (P, Q)          = OR (P, Q)
```

(* Now the CNF can be easily computed *)

```
fun conj_of_disj (AND (P, Q)) = AND (conj_of_disj (P), conj_of_disj (Q))
| conj_of_disj (OR (P, Q))  = distOR (conj_of_disj (P), conj_of_disj (Q))
| conj_of_disj (P)          = P
;
```

```
fun cnf (P) = conj_of_disj (nnf (rewrite (P)));
```

(* A proposition in CNF is a tautology
iff
Every conjunct is a tautology
iff
Every disjunct in every conjunct contains both positive and negative
literals of at least one atom

So we construct the list of all the positive and negative atoms in every
disjunct to check whether the lists are all equal. We need a binary
function on lists to determine whether two lists are disjoint

*)

```
fun isPresent (a, []) = false
| isPresent (a, b::L) = (a = b) orelse isPresent (a, L)
; 
```

```
fun disjoint ([] , M) = true
| disjoint (L, []) = true
| disjoint (L as a::LL, M as b::MM)=
    not(isPresent (a, M)) andalso
    not(isPresent(b, L)) andalso
    disjoint (LL, MM)
; 
```

(* ABHISHEK : Defining a total ordering on atoms (lexicographic
ordering on underlying strings), and extending it to a list of atoms.
*)

```

exception notAtom;

fun atomLess (a, b) = case (a, b) of
    (ATOM(x), ATOM(y)) => x < y
  | (_, _)                => raise notAtom;

fun listLess (a, b) = case (a, b) of
    (_, [])                 => false
  | ([], _)                => true
  | (x :: lx, y :: ly)     => if atomLess(x, y) then true
                                else if atomLess(y, x) then false
                                else listLess(lx, ly);

```

(* ABHISHEK : Once we have a list of falsifiers , we would want to remove any duplication , firstly of atoms within a falsifier , and secondly of falsifiers themselves .

In order to do this , we maintain all lists in some sorted order .
 Instead of sorting a list with a possibly large number of duplicates , we check for duplicates while inserting , and omit insertion if a previous instance is detected .

*)

```

fun merge less ([] , l2) = l2
| merge less (l1 , []) = l1
| merge less (x :: l1 , y :: l2) =
  if less(x, y) then x :: merge less (l1 , y :: l2)

```

```

else if less(y,x) then y::merge less (x::l1,l2)
else merge less (x::l1,l2);

(* ABHISHEK : Claim is that if all lists are built through the above
function , then there is no need to sort or remove duplicates.

Hence all '@' operations have been replaced by merge.

*)

exception not_CNF;

fun positives (ATOM a)      = [ATOM a]
| positives (NOT (ATOM _))= []
| positives (OR (P, Q))   = merge atomLess (positives (P), positives (Q))
| positives (P)           = raise not_CNF
;

fun negatives (ATOM _)     = []
| negatives (NOT (ATOM a))= [ATOM a]
| negatives (OR (P, Q))   = merge atomLess (negatives (P), negatives (Q))
| negatives (P)           = raise not_CNF
;

(* Check whether a formula in CNF is a tautology *)

fun taut (AND (P, Q)) = taut (P) andalso taut (Q)
| taut (P) = (* if it is not a conjunction then it must be a disjunct *)
            not (disjoint (positives (P), negatives (P)))

```

```
;  
  
fun tautology1 (P) =  
  let val Q = cnf (P)  
  in  taut (Q)  
  end  
;  
  
(* The main problem with the above is that it checks whether a given  
   proposition is a tautology , but whenever it is not, it does not yield  
   a falsifying truth assignment. We rectify this problem below.  
*)
```

(*

Firstly , as in the case of the function lookup , we will assume a truth assignment is a list of atoms which are assigned the truth value "true" and that any atom that is not present in the list has been assigned "false".

Assume Q is a proposition in CNF. Then it is only necessary to list out all the lists of truth assignments that can falsify Q.

Suppose Q is in CNF, but not necessarily a tautology . Further let

$$Q = \text{AND} (D_1, \dots, D_n)$$

where each D_i is a disjunction of literals . Each $D_i = P_i + N_i$ where

Pi and Ni are the lists of atoms denoting the positive and negative literals respectively.

Q would be "falsified" if at least one of the Di can be made false. Di can be made false only if it does not contain a "complementary pair", i.e. there exists no atom a such that both a and \sim a occur in Di. Hence for Di to be falsified it is necessary that the lists Pi and Ni are disjoint (if there is no atom common to Pi and Ni, there is no "complementary pair" in Di).

Since Di is a disjunction of literals, it can be falsified only by assigning every literal in Di the value "false". This can be done only by assigning all the atoms in Pi the value "false" and all the atoms in Ni the value "true".

In other words, if Pi and Ni are disjoint, then Ni is a truth assignment which falsifies the proposition Q. We refer to Ni as a FALSIFIER of Q.

Therefore the FALSIFIERS of Q are exactly the list of negative atoms of each disjunct which does not contain a complementary pair. By checking each disjunct in Q we may list out ALL the possible FALSIFIERS of Q.

If Q has no FALSIFIER then no disjunct Di can be made false i.e. every disjunct does indeed have a complementary pair. We may then conclude that Q is a tautology.

*)

```

(* The following function assumes Q is in CNF and outputs a list of list
of atoms that can falsify Q. If this list of list of atoms is empty then
clearly Q is a tautology.
*)

fun falsify (Q) =
  let fun list_Falsifiers (AND (A, B)) =
          merge listLess (list_Falsifiers (A), list_Falsifiers (B))
    | list_Falsifiers (A) = (* Assume A is a disjunct of literals *)
        let val PLA = positives (A) (* no uniq required *)
            val NLA = negatives (A)
        in if disjoint (PLA, NLA) then [NLA]
           else []
        end
    in list_Falsifiers (Q)
  end
;

fun tautology2 (P) =
  let val Q = cnf (P);
      val LL = falsify (Q)
  in if null (LL) then (true, [])
     else (false, LL)
  end
;

val tautology = tautology2;

```

(*

We may use the tautology checker to prove various arguments logically valid or logically invalid. An argument consists of a set of propositions called the "hypotheses" and a (single) proposition called the "conclusion". Loosely speaking , an argument is similar to a theorem of mathematics. The argument is logically valid if the conclusion is a logical consequence of the hypotheses. More accurately , if in every truth assignment which makes all the hypotheses true , the conclusion is also invariably true then the argument is logically valid.

Symbolically if H_1, \dots, H_m are propositions and C is another proposition then the argument $(\{H_1, \dots, H_m\}, C)$ is logically valid (equivalently , C is a logical consequence of $\{H_1, \dots, H_m\}$) if and only if the (compound) proposition

$$(H_1 \wedge \dots \wedge H_m) \Rightarrow C$$

is a tautology .

An argument which is not logically valid is logically invalid. In particular if there exists a truth assignment under which all the hypotheses are true but the conclusion is false , then the argument is invalid .

Any argument is trivially logically valid if there is no truth assignment under which every hypothesis is true. In other words ,

if the set of hypotheses is an inconsistent set then regardless of what the conclusion is , the argument is always logically valid.
The set of hypotheses $\{H_1, \dots, H_m\}$ is "inconsistent" if and only if $(H_1 \wedge \dots \wedge H_m)$ is a "contradiction" (it is false for every truth assignment).

*)

```
type Argument = Prop list * Prop;

fun showArg (A: Argument) =
  let fun printArg (A:Argument as ([] , c)) =
    (drawChar (#"-", 80); print("\n");
     show (c); print ("\n\n"))
  )
  | printArg (A:Argument as (p::plist , c)) =
    (show (p); print ("\n");
     printArg (plist , c)
    )
  in (print ("\n\n"); printArg (A))
  end
;

fun leftReduce (F) =
  let exception emptylist;
      fun lr ([] ) = raise emptylist
      | lr ([a]) = a
      | lr (a::L) = F (a, lr (L))
```

```

in    lr
end
;

val bigAND = leftReduce (AND);

fun Valid ((L, P):Argument) =
  if null (L) then tautology (P)
  else tautology (IMP (bigAND (L), P))
;

fun falsifyArg ((L, P): Argument) =
  if null (L) then falsify (cnf(P))
  else falsify (cnf (IMP (bigAND (L), P)))
;

end (* struct *);

(* open PL; *)

```

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

249 OF 783

QUIT

Lecture 7: Propositional Unsatisfiability

Wednesday 10 August 2011

1. Tautology Checking
2. CNFs: Set of Sets of Literals
3. Propositional Resolution
4. Clean-up
5. The Resolution Method
6. The Algorithm
7. Resolution Examples: Biconditional
8. Resolution Examples: Exclusive-Or
9. Resolution Refutation: 1
10. Resolution Refutation: 2
11. Resolvent as Logical Consequence
12. Logical Consequence by Refutation

Tautology Checking

1. Involves conversion of IMP (bigAND H, conc1) into CNF which increases the size of the formula.
2. Involves **checking falsifiability** of the argument.
3. CNF can be obtained more easily for the formula $(\dots((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \wedge \neg\psi$
 - Convert each individual ϕ_i and $\neg\psi$ to CNF
 - and then append all the lists to obtain the required list.
4. More efficient to use theorem **4.3.2** if the technique involves CNF.

CNFs: Set of Sets of Literals

Given a formula ϕ whose CNF is

$$\gamma \equiv \bigwedge_{1 \leq i \leq m} \delta_i$$

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

$$\delta_i \equiv \bigvee_{1 \leq j \leq n_i} \lambda_{i,j}$$

is a disjunction of literals, we will often write γ as a set of sets of literals as follows:

$$\gamma = \{\{\lambda_{1,1}, \dots, \lambda_{1,n_1}\}, \dots, \{\lambda_{m,1}, \dots, \lambda_{m,n_m}\}\}$$

Note: For the sake of brevity we will abuse notation by identifying a clause (set of literals) with the formula denoting their disjunction and a set of clauses with the formula denoting their conjunction.

Propositional Resolution

To show $\Gamma \models \psi$ we show that $\bigwedge \Gamma \wedge \neg\psi$ is false by first transforming $\bigwedge \Gamma \wedge \neg\psi$ to a formula in CNF.

This CNF is represented as a *set of sets of literals*. Let Δ be the set of sets of literals.

1. Each set $C \in \Delta$ is called a **clause**.
2. Each clause in Δ represents a disjunction of literals.
3. The empty clause $\{\}$ represents a contradiction.
4. The unsatisfiability of the set Δ is shown by deriving the empty clause.

Clean-up

Let Δ be a finite set of clauses.

1. For all clauses C, C' , if $C \subseteq C'$, then C' may be *deleted* from Δ without affecting logical equivalence.
2. Any clause containing *complementary pairs* of literals, may be deleted from Δ without affecting logical equivalence.
3. From any clause, *duplicate* occurrences of a literal may be *deleted* without affecting logical equivalence.

The resulting clause set Δ' is said to be **clean**.

$$\bigwedge_{C \in \Delta} \bigvee_{L \in C} L \Leftrightarrow \bigwedge_{C' \in \Delta'} \bigvee_{L' \in C'} L'$$

The Resolution Method

For any clean set Δ and an atom p let $\Lambda = \{C \in \Delta \mid p \in C\}$ and $\bar{\Lambda} = \{\bar{C} \in \Delta \mid \neg p \in \bar{C}\}$

Since Δ is a clean set

1. $\Lambda \cap \bar{\Lambda} = \emptyset$.
2. However C and \bar{C} may not be disjoint.
3. For each $C \in \Lambda$ and $\bar{C} \in \bar{\Lambda}$, $p \notin \bar{C}$ and $\neg p \notin C$.

The new set of clauses obtained after resolution

$$\begin{aligned} \text{resolve}(\Delta, p) &\stackrel{df}{=} (\Delta - (\Lambda \cup \bar{\Lambda})) \cup \\ &\{D \mid D = (C - \{p\}) \cup (\bar{C} - \{\neg p\}), C \in \Lambda, \bar{C} \in \bar{\Lambda}\} \end{aligned}$$

is called the **resolvent**.

Lemma 7.1 If $C_1 = \{\lambda_{1,i} \mid 1 \leq i \leq m\}$ and $C_2 = \{\lambda_{2,j} \mid 1 \leq j \leq n\}$ are clauses such that $p \in C_1 - C_2$ and $\neg p \in C_2 - C_1$ and $D = (C_1 - \{p\}) \cup (C_2 - \{\neg p\})$. Then

1. $C_1 \wedge C_2 \Rightarrow D$ and
2. resolve preserves satisfiability, i.e. every truth assignment that satisfies both C_1 and C_2 also satisfies D .

Proof: From the semantics of propositional logic it follows that for any truth assignment τ , $\tau \Vdash C_1 \wedge C_2$ if and only if $\tau \Vdash C_1$ and $\tau \Vdash C_2$. Hence we may prove both parts of the lemma by considering an arbitrary truth assignment τ such that $\tau \Vdash C_1 \wedge C_2$. It suffices to show (for both parts) that $\tau \Vdash D$. $\tau \Vdash C_1$ implies that for some i_0 , $1 \leq i_0 \leq m$, $\mathcal{T}[\lambda_{1,i_0}]_\tau = 1$ and for some j_0 , $1 \leq j_0 \leq n$, $\mathcal{T}[\lambda_{2,j_0}]_\tau = 1$. Further since p and $\neg p$ are complementary, it is impossible that both $p \equiv \lambda_{1,i_0}$ and $\neg p \equiv \lambda_{2,j_0}$ hold simultaneously. Hence at least one of the two literals $\lambda_{1,i_0}, \lambda_{2,j_0}$ is in D . It follows therefore that

$$\mathcal{T}[D]_\tau = (\sum_{1 \leq i \leq m} \mathcal{T}[\lambda_{1,i}]_\tau) + (\sum_{1 \leq j \leq n, j \neq j_0} \mathcal{T}[\lambda_{2,j}]_\tau) = 1 \text{ if } p \not\equiv \lambda_{1,i_0} \text{ and}$$

$$\mathcal{T}[D]_\tau = (\sum_{1 \leq i \leq m, i \neq i_0} \mathcal{T}[\lambda_{1,i}]_\tau) + (\sum_{1 \leq j \leq n} \mathcal{T}[\lambda_{2,j}]_\tau) = 1 \text{ if } \neg p \not\equiv \lambda_{2,j_0} \text{ and hence } \tau \Vdash D. \text{ QED} \quad \blacksquare$$

The Algorithm

Require: Δ a clean set of clauses

```
1: while ( $\{\}$   $\notin \Delta$ )  $\wedge \exists(p, \neg p) \in \Delta$  do
2:    $\Delta' := resolve(\Delta, p)$ 
3:    $\Delta := Clean-up(\Delta')$ 
4: end while
```

Note:

1. $\{\}$ is the empty clause which represents the proposition \perp (it represents the disjunction of an empty set of literals).
2. The presence of the empty clause in a set of clauses also indicates the unsatisfiability of the set of clauses.

Lemma 7.2 Let $\Delta_1 = \text{Clean-up}(\text{resolve}(\Delta_0))$. Then

1. $\Delta_0 \Rightarrow \Delta_1$ and
2. if Δ_0 is satisfiable then Δ_1 is satisfiable.

Proof: The proof follows directly from the proof of lemma 7.1 and from problem 5 of exercise 4.1, where it should have been shown that both \wedge and \vee preserve logical implication. QED ■

Corollary 7.3 If Δ_1 and Δ_0 are as in lemma 7.2 then if Δ_1 is unsatisfiable then so is Δ_0 .

□

Theorem 7.4 Given a clean non-empty set Δ_0 of non-empty clauses, the propositional resolution algorithm terminates in at most $|\text{atoms}(\Delta_0)|$ iterations, deriving either an empty clause or a set of non-empty clauses which are satisfied by every model of the original set Δ_0 .

Proof: Since in each iteration one atom and its negation are completely eliminated, $|\text{atoms}(\Delta_0)|$ is the number of iterations possible. Further by applying lemma 7.2 to the result of each iteration we get that satisfiability and logical implication are preserved. QED ■

7.1. Space Complexity of Propositional Resolution.

Assume n is the number of atoms of which Δ is made up. After some iterations of **resolution** and **cleanup**, assume there are k distinct atoms in the set of clauses on which **resolution** is applied. After performing the **cleanup** procedure there could be *at most* 2^k clauses with each clause containing at most k literals. Assuming each literal occupies a unit of memory, the space requirement is given by $\text{size}(\Delta) = 2^k k$ literals.

For any complementary pair $(p, \neg p)$, it is possible that at most half of the 2^k (i.e. 2^{k-1}) clauses contain p and the other half contain $\neg p$. This would be the worst-case scenario, as it yields the maximum number of new clauses. Therefore in performing a single step of resolution over all possible pairs of clauses to yield a new set Δ' of clauses, a maximum of $2^{k-1} \times 2^{k-1}$ unions of distinct pairs of clauses needs to be performed. Before applying the **clean-up** procedure the space required could be as high as $2^{k-1} \times 2^{k-1} = 2^{2(k-1)} > 2^k k$ for $k > 4$. But since Δ' is made up of at most $(k-1)$ atoms, $\text{size}(\Delta') \leq 2^{k-1}(k-1)$, after **clean-up** the space requirement reduces to $2^{k-1}(k-1)$. Since $k \leq n$ the maximum space required after the first application of resolution and before cleaning up exceeds the space required for all other iterations and is bounded by $2^{2(n-1)} = O(2^{2n})$.

7.2. Time Complexity of Propositional Resolution

Given a space of $2^k k$ to represent the clauses containing at most k atoms, we require a time proportional to this amount of space in order to identify which clauses have to be resolved against a particular complementary pair. After **resolution** we create a space of $2^{2(k-1)}$ which has to be scanned for the **cleanup** operations. Hence the amount of time required to perform a step of resolution and the amount of time required to perform the cleanup are both proportional to $2^{2(k-1)}$. Hence the total time required for performing **resolution** followed by **cleanup** in n iterations (which is the maximum possible) is given by

$$T(n) \geq \sum_{k=1}^n 2^{2k-2}$$

which is clearly exponential.

Hence both the *worst case* time and space complexities are exponential in the number of atoms.

Resolution Examples: Biconditional

Example 7.5 Sometimes there may be more than one complementary pair of literals in the same pair of clauses. Consider the biconditional operator (\leftrightarrow) on atomic propositions p and q . We have

$$p \leftrightarrow q \Leftrightarrow (p \vee \neg q) \wedge (\neg p \vee q) \equiv \{\{p, \neg q\}, \{\neg p, q\}\}$$

Applying resolution on the pair of literals p and $\neg p$ we obtain the clause set which after clean-up yields the empty set of clauses.

$$\{\{q, \neg q\}\} \equiv \{\} \equiv \top$$

There is really nothing more to this resolvent than that \top is a logical consequence of any proposition (including \perp).

Resolution Examples: Exclusive-Or

Example 7.6 The exclusive-or operation \oplus is simply the negation of the biconditional operator \leftrightarrow . Hence we have

$$p \oplus q \Leftrightarrow (p \vee q) \wedge (\neg p \vee \neg q) \equiv \{\{p, q\}, \{\neg p, \neg q\}\}$$

which on resolution on the pair $(p, \neg p)$ and subsequent clean-up again yields the empty set of clauses.

$$\{\{q, \neg q\}\} \equiv \{\} \equiv \top$$

Resolution Refutation: 1

Example 7.7 Consider the simple logical consequence

$$p \wedge q \models p$$

which we prove by resolution refutation. The set of clauses representing the hypothesis and the negation of the conclusion is $\{\{p\}, \{q\}, \{\neg p\}\}$. Resolving on the pair $(p, \neg p)$ yields

$$\{\{\}, \{q\}\}$$

Notice that $\{\} \equiv \perp \equiv \{\{\}, \{q\}\}$.

Since for any clause $C \neq \emptyset$, $C \supseteq \{\}$, the clean-up always reduces every set of clauses Δ to $\{\{\}\}$ whenever $\{\} \in \Delta$.

Resolution Refutation: 2

Example 7.8 Consider the simple logical consequence

$$p \wedge q \models p \leftrightarrow q$$

which we prove by resolution refutation. Negating the conclusion yields $p \oplus q \equiv \{\{p, q\}, \{\neg p, \neg q\}\}$. The set of clauses representing the hypothesis and the negation of the conclusion is $\{\{p\}, \{q\}, \{p, q\}, \{\neg p, \neg q\}\}$ which after clean-up yields

$$\{\{p\}, \{q\}, \{\neg p, \neg q\}\}$$

Resolving on the pair $(p, \neg p)$ produces

$$\{\{q\}, \{\neg q\}\}$$

and then on the pair $(q, \neg q)$ produces the empty clause $\{\{\}\}$.

Resolvent as Logical Consequence

The set of clauses resulting from any application of **resolution** is a set of clauses that represents a logical consequence of the original set of clauses

Example 7.9 *We use resolution to prove*

$$(p \vee q) \wedge (p \vee r) \wedge \neg p \Leftrightarrow q \wedge r$$

The set of clauses $\{\{p, q\}, \{p, r\}, \{\neg p\}\}$ may be resolved on the pair $(p, \neg p)$ to yield the set

$$\{\{q\}, \{r\}\} \equiv q \wedge r$$

Note that we have resolved all occurrences of the complementary pair $(p, \neg p)$.

Logical Consequence by Refutation

Example 7.10 In example 7.9 since we resolve all occurrences of the complementary pair $(p, \neg p)$, we could not have proved that $\neg p \wedge q \wedge r$ is also a logical consequence of $(p \vee q) \wedge (p \vee r) \wedge \neg p$ which it indeed is. We may use refutation for this purpose by noting that $\neg(\neg p \wedge q \wedge r) \equiv \{p, \neg q, \neg r\}$. We then resolve the set $\{\{p, q\}, \{p, r\}, \{\neg p\}, \{p, \neg q, \neg r\}\}$ on the pair $(p, \neg p)$ to obtain the set $\{\{q\}, \{r\}, \{\neg q, \neg r\}\}$ which may be resolved on the pairs $(q, \neg q)$ and $(r, \neg r)$ subsequently to yield the empty clause.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

269 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

270 OF 783

QUIT

Lecture 8: Analytic Tableaux

Friday 12 August 2011

1. Against Resolution
2. The Analytic Tableau Method
3. Basic Tableaux Facts
4. Tableaux Rules
5. Structure of the Rules
6. Tableaux
7. Slim Tableaux

Against Resolution

1. For any argument, it was still necessary to transform the argument into a mammoth formula in CNF in order to be able to perform *resolution*.
2. The numbers of clauses and sizes of clauses could temporarily increase as a result of resolution.
3. Termination relied on the reduction of the number of atoms at each step of resolution.
4. Resolution also requires a clean-up of the initial set of clauses to work correctly.

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.

Basic Tableaux Facts

Theorem 8.1 For any truth assignment τ , and formulae ϕ, ψ ,

$$\mathcal{T}[\neg\phi]_\tau = 1 \Rightarrow \mathcal{T}[\phi]_\tau = 0$$

$$\mathcal{T}[\neg\phi]_\tau = 0 \Rightarrow \mathcal{T}[\phi]_\tau = 1$$

$$\mathcal{T}[\phi \wedge \psi]_\tau = 1 \Rightarrow \mathcal{T}[\phi]_\tau = 1 \text{ } \& \text{ } \mathcal{T}[\psi]_\tau = 1$$

$$\mathcal{T}[\phi \wedge \psi]_\tau = 0 \Rightarrow \mathcal{T}[\phi]_\tau = 0 \quad | \quad \mathcal{T}[\psi]_\tau = 0$$

$$\mathcal{T}[\phi \vee \psi]_\tau = 1 \Rightarrow \mathcal{T}[\phi]_\tau = 1 \quad | \quad \mathcal{T}[\psi]_\tau = 1$$

$$\mathcal{T}[\phi \vee \psi]_\tau = 0 \Rightarrow \mathcal{T}[\phi]_\tau = 0 \text{ } \& \text{ } \mathcal{T}[\psi]_\tau = 0$$

$$\mathcal{T}[\phi \rightarrow \psi]_\tau = 1 \Rightarrow \mathcal{T}[\phi]_\tau = 0 \quad | \quad \mathcal{T}[\psi]_\tau = 1$$

$$\mathcal{T}[\phi \rightarrow \psi]_\tau = 0 \Rightarrow \mathcal{T}[\phi]_\tau = 1 \text{ } \& \text{ } \mathcal{T}[\psi]_\tau = 0$$

$$\mathcal{T}[\phi \leftrightarrow \psi]_\tau = 1 \Rightarrow \mathcal{T}[\phi]_\tau = 1 = \mathcal{T}[\psi]_\tau \quad | \quad \mathcal{T}[\phi]_\tau = 0 = \mathcal{T}[\psi]_\tau$$

$$\mathcal{T}[\phi \leftrightarrow \psi]_\tau = 0 \Rightarrow \mathcal{T}[\phi]_\tau = 0 = \overline{\mathcal{T}[\psi]_\tau} \quad | \quad \mathcal{T}[\phi]_\tau = 1 = \overline{\mathcal{T}[\psi]_\tau}$$



Tableaux Rules

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

Structure of the Rules

The **tableaux rules** are of two kinds:

Elongation rules Each of the rules $\neg\neg$, \wedge , $\neg\vee$ and $\neg \rightarrow$ elongates the path without increasing the size of the tableau (since the original formula to which a rule was applied may be discarded)

Branching rules These are the rules $\neg\wedge$, \vee , $\neg \rightarrow$, \leftrightarrow and $\neg \leftrightarrow$ which lead to branching of the tableau.

Tableaux

1. A *tableau* is a tree where each path of the tableau represents a conjunction of “unbroken” formulae.
2. Each application of the **tableaux rules** preserves satisfiability of the conjunction of “unbroken” formulae in each path.
3. A path of the tableau is **closed** if it contains a complementary pair (the conjunction of the formulae in the path is clearly **unsatisfiable**).
4. The result of applying a tableau rule to an ancestor node has to be distributed in all branches of its descendants.
5. A tableau is **closed** if every path in the tableau is closed signifying that the original set of formulae is unsatisfiable.

Slim Tableaux

1. Any formula which has been broken up by a tableau rule may be discarded.
2. Any branch which has been closed may be discarded.
3. Any formula which dominates several branches of the tableau creates multiple copies (one in each branch of its descendants) when it is broken up.

By applying the elongation rules first the number of branches over which elongation rules have to be replicated can be reduced

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

280 OF 783

QUIT

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

281 OF 783

[QUIT](#)

Lecture 9: Consistency & Completeness

Wednesday 17 August 2011

1. Tableaux Rules: Restructuring
2. Tableaux Rules: 2
3. Tableau Proofs
4. Consistency
5. Unsatisfiability
6. Hintikka Sets
7. Hintikka's Lemma
8. Tableaux and Hintikka sets
9. Completeness

Tableaux Rules: Restructuring

In general the **elongation and branching rules** of the tableau look like this

$$\begin{array}{c|c} \text{Elongation. } \frac{\phi}{\psi} & \text{Branching. } \frac{\phi}{\psi \mid \chi} \\ \hline & \chi \end{array}$$

where ψ and χ are **subformulae** of ϕ .

Let $\Gamma = \Delta \cup \{\phi\}$ where $\phi \notin \Delta$ be a set of formulae. It will be convenient to use sets of formulae in the tableau rules. The elongation and branching rules are rendered as follows respectively

$$\begin{array}{c|c} \text{Elongation. } \frac{\Delta \cup \{\phi\}}{\Delta \cup \{\psi, \chi\}} & \text{Branching. } \frac{\Delta \cup \{\phi\}}{\Delta \cup \{\psi\} \mid \Delta \cup \{\chi\}} \end{array}$$

Tableaux Rules: 2

$\perp.$	$\frac{\Delta \cup \{\phi, \neg\phi\}}{\{\perp\}}$	$\neg\neg.$	$\frac{\Delta \cup \{\neg\neg\phi\}}{\Delta \cup \{\phi\}}$
$\wedge.$	$\frac{\Delta \cup \{\phi \wedge \psi\}}{\Delta \cup \{\phi, \psi\}}$	$\neg\wedge.$	$\frac{\Delta \cup \{\neg(\phi \wedge \psi)\}}{\Delta \cup \{\neg\phi\} \mid \Delta \cup \{\neg\psi\}}$
$\vee.$	$\frac{\Delta \cup \{\phi \vee \psi\}}{\Delta \cup \{\phi\} \mid \Delta \cup \{\psi\}}$	$\neg\vee.$	$\frac{\Delta \cup \{\neg(\phi \vee \psi)\}}{\Delta \cup \{\neg\phi, \neg\psi\}}$
$\rightarrow.$	$\frac{\Delta \cup \{\phi \rightarrow \psi\}}{\Delta \cup \{\neg\phi\} \mid \Delta \cup \{\psi\}}$	$\neg\rightarrow.$	$\frac{\Delta \cup \{\neg(\phi \rightarrow \psi)\}}{\Delta \cup \{\phi, \neg\psi\}}$

$\leftrightarrow.$	$\frac{\Delta \cup \{\phi \leftrightarrow \psi\}}{\Delta \cup \{\phi \wedge \psi\} \mid \Delta \cup \{\neg\phi \wedge \neg\psi\}}$
$\neg\leftrightarrow.$	$\frac{\Delta \cup \{\neg(\phi \leftrightarrow \psi)\}}{\Delta \cup \{\phi \wedge \neg\psi\} \mid \Delta \cup \{\neg\phi \wedge \psi\}}$

Tableau Proofs

1. A tableau is a tree rooted at a node containing a set Γ of formulas
2. Each application of
 - a **elongation rule** $\frac{\Gamma}{\Gamma'}$ to a leaf Γ of the tableau extends the path to Γ' ,
 - a **branching rule** $\frac{\Gamma}{\Gamma' \mid \Gamma''}$ to a leaf Γ of the tableau extends the tableau to two leaves Γ' and Γ'' .
3. A path of the tableau is **closed** if its leaf is $\{\perp\}$.
4. The tableau is **closed** if every path is closed, otherwise the tableau is **open**.

Consistency

Definition 9.1 A set Γ of formulas is **consistent** if it is satisfiable i.e. there is a truth assignment under which every formula of Γ is true. Otherwise, it is **inconsistent**.

Fact 9.2 Every non-empty subset of a consistent set is also consistent

Lemma 9.3 Each tableau rule preserves satisfiability in the following sense.

Elongation Rules $\frac{\Gamma}{\Gamma'}$ If the numerator Γ is satisfiable then so is the denominator Γ' .

Branching Rules $(\frac{\Gamma}{\Gamma' \mid \Gamma''})$ If the numerator Γ is satisfiable then at least one of the denominators Γ' or Γ'' is satisfiable.

Proof outline of lemma 9.3

Proof: It may be shown that for any truth assignment τ ,

Elongation Rules $\frac{\Gamma}{\Gamma'}$. if every formula in Γ is true then every formula in Γ' is also true under τ .

Branching Rules $\frac{\Gamma}{\Gamma'|\Gamma''}$. if every formula in Γ is true under τ then every formula in Γ' or every formula in Γ'' is true under τ .

QED ■

Unsatisfiability

Definition 9.4 A tableau is completed if no leaf in any path may be extended.

Corollary 9.5 If Γ is satisfiable then there exists a completed tableau rooted at Γ which has a satisfiable leaf.

Corollary 9.6 A set Γ is unsatisfiable if there exists a closed tableau rooted at Γ .

Question. If a completed tableau rooted at Γ is closed could there be other completed tableaux rooted at Γ which might be open?

Hintikka Sets

Definition 9.7 A finite or infinite set Γ is a Hintikka set if

1. $\perp \notin \Gamma$ and for any $p \in A$, $\{p, \neg p\} \not\subseteq \Gamma$,
2. If $\phi \equiv \psi \odot \chi \in \Gamma$ for $\odot \in \{\wedge, \neg\vee, \neg\rightarrow\}$ then $\{\psi', \chi'\} \subseteq \Gamma$,
3. If $\phi \equiv \psi \oplus \chi \in \Gamma$ for $\oplus \in \{\vee, \neg\wedge, \rightarrow, \leftrightarrow, \neg\leftrightarrow\}$ then $\{\psi', \chi'\} \cap \Gamma \neq \emptyset$

where ψ' and χ' are defined by the following table

$\phi \equiv \psi \odot \chi$	ψ'	χ'	$\phi \equiv \psi \oplus \chi$	ψ'	χ'
$\psi \wedge \chi$	ψ	χ	$\neg(\psi \wedge \chi)$	$\neg\psi$	$\neg\chi$
$\neg(\psi \vee \chi)$	$\neg\psi$	$\neg\chi$	$\psi \vee \chi$	ψ	χ
$\neg(\psi \rightarrow \chi)$	ψ	$\neg\chi$	$\psi \rightarrow \chi$	$\neg\psi$	χ
			$\psi \leftrightarrow \chi$	$\psi \wedge \chi$	$\neg\psi \wedge \neg\chi$
			$\neg(\psi \leftrightarrow \chi)$	$\neg\psi \wedge \chi$	$\psi \wedge \neg\chi$

Hintikka's Lemma

Lemma 9.8 *Every Hintikka set is satisfiable.*

Proof: Let Γ be a Hintikka set. For any atom p , since $\{p, \neg p\} \not\subseteq \Gamma$, consider the following truth assignment τ .

1. $\tau(p) = 1$ if $p \in \Gamma$,
2. $\tau(p) = 0$ if $\neg p \in \Gamma$ and
3. if $\{p, \neg p\} \cap \Gamma = \emptyset$ then choose any value (say 1 for definiteness).

We may then show by induction on the **degree** of formulae in Γ that each formula in Γ is satisfiable. QED

Tableaux and Hintikka sets

Theorem 9.9

Let $\Gamma_0, \Gamma_1 \dots, \Gamma_n$ be an open path of a completed tableau. Then $\bigcup_{0 \leq m \leq n} \Gamma_m$ is a Hintikka set.

Proof: We may prove that each rule in Tableaux Rules: 2 creates a path for the construction of Hintikka sets. QED

Completeness

Theorem 9.10 Completeness of the Tableau Method

1. If ϕ is a tautology then every completed tableau rooted at $\{\neg\phi\}$ is closed.
2. Every tautology is provable by the tableau method.

Proof:

1. Suppose \mathcal{T} is a completed tableau rooted at $\{\neg\phi\}$ which is open. Then by corollary 9.5 $\neg\phi$ must be satisfiable and hence ϕ cannot be a tautology. Hence \mathcal{T} must be closed.
2. If ϕ is a tautology that cannot be proved by the tableau method, there must exist a completed tableau rooted at $\{\neg\phi\}$ which has an open path. But that implies $\{\neg\phi\}$ is satisfiable which implies that ϕ is not a tautology.

QED

HOME PAGE



LCS

GO BACK

FULL SCREEN

CLOSE

293 OF 783

QUIT

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

294 OF 783

[QUIT](#)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

295 OF 783

QUIT

Lecture 10: The Compactness Theorem

Friday 19 August 2011

1. Satisfiability of Infinite Sets
2. The Compactness Theorem
3. Inconsistency
4. Consequences of Compactness

Satisfiability of Infinite Sets

From corollaries 9.5 and 9.6 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?

The Compactness Theorem

Theorem 10.3 (The Compactness Theorem) *A (countably) infinite set is satisfiable if all its nonempty finite subsets are satisfiable.*



Corollary 10.4 *Any (finite or infinite) set of formulae is satisfiable iff all its non-empty finite subsets are satisfiable.*

Note:

- If Γ is a countably infinite set then it can be placed in 1-1 correspondence with the set \mathbb{N} of naturals and hence there is some enumeration of its formulae and each formula carries an unique index from \mathbb{N} .

Proof of the Compactness Theorem

Proof: Let Γ be a countably infinite set of propositions. Then clearly Γ may be enumerated in some order, say

$$\{\phi_0, \phi_1, \phi_2, \dots\} \quad (7)$$

where each ϕ_j has the unique index $j \geq 0$. For each $m \geq 0$, let $\Gamma_m = \{\phi_0, \phi_1, \phi_2, \dots, \phi_m\}$.

Claim. Every nonempty finite subset of Γ is satisfiable iff for each $m \geq 0$, Γ_m is satisfiable.

$\vdash (\Rightarrow)$ clearly holds since each Γ_m is a finite subset.

(\Leftarrow) Let $\emptyset \neq \Delta \subseteq_f \Gamma$. Let $k \geq 0$ be the index of the formula with the highest index in Δ . Clearly $\Delta \subseteq_f \Gamma_k$. Since the set Γ_k is satisfiable, by corollary 10.2, Δ is also satisfiable. \dashv

Hence it suffices to prove that if each of the Γ_i , $i \geq 0$, is satisfiable then Γ is satisfiable.

Consider a tableau \mathcal{T}_0 rooted at Γ_0 constructed using the **tableau rules**. Since Γ_0 is satisfiable, \mathcal{T}_0 has one or more open paths. Extend each of the open paths with the formula ϕ_1 and continue the tableau. The resulting tableau \mathcal{T}_1 is for the set Γ_1 and it does not close either. Hence tableaux \mathcal{T}_k for each Γ_k may be extended to yield open tableaux \mathcal{T}_{k+1} for Γ_{k+1} .

Consider the final tableau \mathcal{T} obtained by this process of extension. \mathcal{T} is a *finitely branching infinite tree* with at least one path that does not close. By König's Lemma 2.17 there is an infinite path. Let

Φ be the set of all formulae in this path. Since this path contains each of the formulae $\phi_i \in \Gamma$, we have $\Gamma \subseteq \Phi$ and further Φ is a Hintikka set. By Hintikka's lemma 9.8 this set must be satisfiable. QED ■

Inconsistency

Corollary 10.5 A set Γ is *inconsistent* if some nonempty finite subset of Γ is unsatisfiable.

Proof: Follows from the compactness theorem 10.3 and its corollary 10.4. QED ■

Facts 10.6

1. Any superset of an inconsistent set is also inconsistent.
2. Any set containing a complementary pair is inconsistent.
3. (see *table*) If $\Delta \cup \{\psi', \chi'\}$ is inconsistent then so is $\Delta \cup \{\phi\}$ where $\phi \equiv \psi \odot \chi$
4. (see *table*) If both $\Delta \cup \{\psi'\}$ and $\Delta \cup \{\chi'\}$ are inconsistent then so is $\Delta \cup \{\phi\}$ where $\phi \equiv \psi \oplus \chi$.

Consequences of Compactness

Corollary 10.7 *Given a finite or infinite set Γ , and a formula ψ*

1. $\Gamma \cup \{\neg\psi\}$ is inconsistent iff there exists $\Delta = \{\phi_i \mid 1 \leq i \leq n\} \subseteq_f \Gamma$, $n \geq 0$, such that $\Delta \cup \{\neg\psi\}$ is inconsistent.
2. $\Gamma \models \psi$ iff $\Delta \models \psi$ iff $(\bigwedge \Delta) \rightarrow \psi$ is a tautology, for some $\Delta = \{\phi_i \mid 1 \leq i \leq n\} \subseteq_f \Gamma$.

Hence

1. to show that an argument is valid it suffices to prove that the conclusion follows from a finite subset of the hypotheses.
2. to show invalidity of an argument it suffices to find a finite subset of the hypotheses which are inconsistent with the conclusion.

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

304 OF 783

[QUIT](#)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

305 OF 783

QUIT

Lecture 11: Maximally Consistent Sets

Tuesday 23 August 2011

1. Consistent Sets
2. Properties of Finite Character: 1
3. Properties of Finite Character: 2
4. Properties of Finite Character: 3
5. Maximally Consistent Sets
6. Lindenbaum's Theorem

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.*



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. QED ■

Properties of Finite Character: 1

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

Notation: $S \Vdash \mathfrak{p}$ denotes the statement “ S has property \mathfrak{p} ”.

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.*

Properties of Finite Character: 3

Theorem 11.4 (Tukey's Lemma) *For any denumerable universe U and any property \mathfrak{p} of finite character of subsets of U , any set $S \subseteq U$ such that $S \Vdash \mathfrak{p}$ can be extended to a maximal set S_∞ such that $S \subseteq S_\infty \subseteq U$ with $S_\infty \Vdash \mathfrak{p}$.*

Proof of Tukey's Lemma

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

$$a_1, a_2, a_3, \dots \quad (8)$$

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

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

Clearly we have the infinite chain

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

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

Claim. $S_\infty \Vdash \mathfrak{p}$.

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

Claim. S_∞ is maximal.

Suppose there exists an element $a \in U$ such that $S_\infty \cup \{a\} \Vdash \mathfrak{p}$. We know from the

claim above that $S_\infty \Vdash \mathfrak{p}$ and from the construction of each S_i , that $S_i \Vdash \mathfrak{p}$ for each $i \geq 0$. Clearly $a = a_{j+1}$ for some $j \geq 0$ in the enumeration (8). Hence $S_j \cup \{a_{j+1}\} \Vdash \mathfrak{p}$. But $S_j \cup \{a_{j+1}\} = S_{j+1} \subseteq S_\infty$. Hence $S_\infty = S_\infty \cup \{a\}$ and S_∞ is maximal. \dashv

QED



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.

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_∞ . QED ■

Alternative Proof of Lindenbaum's Theorem *ab initio*

Proof: Let Γ be a consistent set. Since \mathcal{P}_0 is generated from a countably infinite set of atoms and a finite set of operators, \mathcal{P}_0 is a countably infinite set (see problem 1 of exercise 2.1). Hence the formulae of \mathcal{P}_0 can be enumerated in some order

$$\phi_1, \phi_2, \phi_3, \dots \quad (9)$$

Starting with $\Gamma = \Gamma_0$ consider the sets

$$\Gamma_{i+1} = \begin{cases} \Gamma_i \cup \{\phi_{i+1}\} & \text{if } \Gamma_i \cup \{\phi_{i+1}\} \text{ is consistent} \\ \Gamma_i & \text{otherwise} \end{cases}$$

Clearly we have the infinite chain

$$\Gamma = \Gamma_0 \subseteq \Gamma_1 \subseteq \Gamma_2 \subseteq \dots$$

such that each Γ_i is consistent. Let $\Gamma_\infty = \bigcup_{i \geq 0} \Gamma_i$.

Claim. Γ_∞ is consistent.

Let $\Delta \subseteq_f \Gamma_\infty$, then since Δ is finite, it must be the subset of some Γ_i . Since Γ_i is consistent, so is Δ . Hence every finite subset of Γ_∞ is consistent. Therefore by the compactness theorem Γ_∞ is consistent. \dashv

Claim. Γ_∞ is maximal.

⊤ Suppose there exists a formula ϕ such that $\Gamma_\infty \cup \{\phi\}$ is consistent. Clearly $\phi \equiv \phi_{i+1}$ for some $i \geq 0$ in the enumeration (9). Since $\Gamma_\infty \cup \{\phi_{i+1}\}$ is consistent, by fact 9.2 $\Gamma_i \cup \{\phi_{i+1}\} \subseteq \Gamma_\infty \cup \{\phi_{i+1}\}$ is also consistent. But then $\Gamma_{i+1} = \Gamma_i \cup \{\phi_{i+1}\} \subseteq \Gamma_\infty$ and hence $\Gamma_\infty = \Gamma_\infty \cup \{\phi\}$. Hence Γ_∞ is maximal. \dashv

QED

Exercise 11.1

1. Let $\langle P, \leq \rangle$ be a finite partial order. Prove using König's lemma that every element of P lies between a maximal element and a minimal element i.e. for each $a \in P$ there exist a minimal element $l \in P$ and a maximal element $u \in P$ such that $l \leq a \leq u$.
2. Prove that every maximally consistent set is a Hintikka set.
3. For any given consistent set Γ of formulae, there may exist more than one maximally consistent extension. Give examples of Γ and ψ such that there are two maximally consistent extensions, Γ_∞ and Γ'_∞ with $\psi \in \Gamma_\infty$ and $\neg\psi \in \Gamma'_\infty$.
4. (Tarski's theorem) For any set Γ , of formulae, the set Γ^{\models} called the **closure under logical consequence** is defined as

$$\Gamma^{\models} = \{ \psi \in \mathcal{P}_0 \mid \Delta \models \psi, \text{ for some } \Delta \subseteq_f \Gamma \}$$

Let $\mathcal{MC}(\Gamma) = \{ \Gamma_\infty \mid \Gamma_\infty \text{ is a maximally consistent extension of } \Gamma \}$ be the set of all maximally consistent extensions of Γ . Prove that

$$\Gamma^{\models} = \bigcap_{\Gamma_\infty \in \mathcal{MC}(\Gamma)} \Gamma_\infty$$

for every consistent set Γ .

5. **(Interpolation)** For any finite set $V \subseteq_f A$, define $T_V = \{\tau_V \mid \tau_V : V \rightarrow \{\perp, \top\}\}$ and for any formula χ such that $V \subseteq \text{atoms}(\chi)$ and any $\tau_V \in T_V$, let $\tau_V(\chi)$ denote the formula obtained from χ by replacing all occurrences of each atom $p \in V$ by the atom $\tau_V(p)$. Further let $T_V(\chi) = \{\tau_V(\chi) \mid \tau_V \in T_V\}$.

Let $X, Y, Z \subseteq_f A$ be pairwise disjoint (finite) sets of atoms and let ϕ and ψ be formulae such that

- $\text{atoms}(\phi) \subseteq X \cup Y$,
- $\text{atoms}(\psi) \subseteq Z \cup Y$ and
- $\models \phi \rightarrow \psi$

(a) Let $\lambda \stackrel{df}{=} \bigvee T_X(\phi)$ and $\rho \stackrel{df}{=} \bigwedge T_Z(\psi)$. Then prove that

- i. $\models \phi \rightarrow \lambda$
- ii. $\models \lambda \rightarrow \rho$
- iii. $\models \rho \rightarrow \psi$

(b) Prove that for any formula θ with $\text{atoms}(\theta) \subseteq Y$, if $\models \phi \rightarrow \theta$ and $\models \theta \rightarrow \psi$ then $\models \lambda \rightarrow \theta$ and $\models \theta \rightarrow \rho$. θ is called an **interpolant** of ϕ and ψ .

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

321 OF 783

[QUIT](#)

Lecture 12: Formal Theories

Wednesday 24 August 2011

1. Introduction to Reasoning
2. Proof Systems: 1
3. Requirements of Proof Systems
4. Proof Systems: Desiderata
5. Formal Theories
6. Formal Language
7. Axioms and Inference Rules
8. Axiomatic Theories
9. Syntax and Decidability
10. A Hilbert-style Proof System
11. Rule Patterns

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 invalidity.

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.

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.

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?*

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.

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

Formal Theories

Axioms and Inference Rules

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 $R. \frac{X_1 \dots X_m}{Y}$ 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**

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.*

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*).

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

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

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

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

- A single rule of inference *modus ponens*

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

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.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

335 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

336 OF 783

QUIT

Lecture 13: Proof Theory: Hilbert-style

Wednesday 26 August 2011

1. More About Formal Theories
2. An Example Proof
3. Formal Proofs
4. Provability and Formal Proofs
5. The Deduction Theorem
6. About Formal Proofs

More About Formal Theories

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

Theorem 13.1 *Let Γ and Δ be finite 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 $\Gamma \vdash \psi$.*



An Example Proof

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . As is normal practice in mathematics the proof is presented as a sequence of steps.

1. $\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)$ $\{\phi/X, \phi \rightarrow \phi/Y\}K$

An Example Proof

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . As is normal practice in mathematics the proof is presented as a sequence of steps.

1. $\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)$ $\{\phi/X, \phi \rightarrow \phi/Y\}K$
2. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
 $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$

An Example Proof

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . As is normal practice in mathematics the proof is presented as a sequence of steps.

1. $\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)$ $\{\phi/X, \phi \rightarrow \phi/Y\}K$
2. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
 $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$
3. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$ $\{2, 1\}MP$

An Example Proof

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . As is normal practice in mathematics the proof is presented as a sequence of steps.

1. $\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)$ $\{\phi/X, \phi \rightarrow \phi/Y\}K$
2. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
 $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$
3. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$ $\{2, 1\}MP$
4. $(\phi \rightarrow (\phi \rightarrow \phi))$ $\{\phi/X, \phi/Y\}K$

An Example Proof

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . As is normal practice in mathematics the proof is presented as a sequence of steps.

1. $\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)$ $\{\phi/X, \phi \rightarrow \phi/Y\}K$
2. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
 $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$
3. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$ $\{2, 1\}MP$
4. $(\phi \rightarrow (\phi \rightarrow \phi))$ $\{\phi/X, \phi/Y\}K$
5. $\phi \rightarrow \phi$ $\{3, 4\}MP$

where each step is justified as an instance of an axiom schema or a rule.

However the proof is better written out as an “upside-down” *proof tree* where

1. each node is a formula. The leaves are axioms (or empty in the case of axiom schemas).
2. each internal node is an application of a rule of appropriate arity applied to the appropriate target (definition 2.11) nodes in the tree.
3. The line-segments between the various levels on the tree show how a node depends on the nodes in the immediately “succeeding” level.
4. the root node is the final formula to be proven.
5. The labels on each line-segment separating a direct consequence from its premise(s) also provide the justification.

The **same proof** may then be rendered as follows:

$$\frac{1 \frac{\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)}{3 \frac{2 \frac{(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))}{5 \frac{((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))}{\phi \rightarrow \phi}}}{4 \frac{(\phi \rightarrow (\phi \rightarrow \phi))}{\phi \rightarrow \phi}}}$$

where the justifications of each step are

1. $\{\phi/X, \phi \rightarrow \phi/Y\}K$
2. $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$
3. $\{2, 1\}MP$
4. $\{\phi/X, \phi/Y\}K$
5. $\{3, 4\}MP$

Each node in this proof tree is said to be an **instance** of a rule or an axiom-schema.

Formal Proofs

Definition 13.2

- A formal proof of a formula ϕ from a finite set Γ of formulae is a finite tree of formulae
 - rooted at the formula ϕ ,
 - the leaves are axioms or instances of axiom schemas or members from Γ .
 - each non-leaf node is a direct consequence of one or more nodes at the “succeeding” level by virtue of application of a rule of inference of the appropriate arity.
- ϕ is said to be (formally) provable from Γ in the proof system \mathcal{H}_0 and denoted $\Gamma \vdash_{\mathcal{H}_0} \phi$ if there exists a formal proof of ϕ in the system \mathcal{H}_0 .
- ϕ is a (formal) theorem if $\Gamma = \emptyset$ and is denoted $\vdash_{\mathcal{H}_0} \phi$

Provability and Formal Proofs

Facts 13.3 Given any theory $\mathbb{T} = \langle \mathcal{L}, \mathcal{A}, \mathcal{R} \rangle$ and a wff $\psi \in \mathcal{L}$,

1. If ψ is an axiom or instance of an axiom-schema then $\vdash \psi$ and hence ψ is a formal theorem.
2. If $\vdash \psi$ then all the leaf nodes in any proof tree of ψ are either axioms or instances of axiom-schemas.
3. If ψ is an axiom or an instance of an axiom-schema then $\Gamma \vdash \psi$ for any $\Gamma \subseteq \mathcal{L}$.
4. For any $\phi \in \Gamma$, $\Gamma \vdash \phi$.

The Deduction Theorem

Notation Given a set Γ and a formula ϕ , “ $\Gamma, \phi \vdash \psi$ ” denotes “ $\Gamma \cup \{\phi\} \vdash \psi$ ”

Theorem 13.4 (The Deduction Theorem) *For all $\Gamma \subseteq_f \mathcal{L}_0$ and formulae ϕ and ψ , $\Gamma, \phi \vdash \psi$ if and only if $\Gamma \vdash \phi \rightarrow \psi$.*



The Deduction theorem justifies our usual notion of a direct proof from the hypotheses of a conditional conclusion – the *antecedent* of the conditional is added to the assumptions and the *consequent* is proven.

Proof of The Deduction Theorem (theorem 13.4)

Proof: (\Rightarrow). Assume $\Gamma, \phi \vdash \psi$. Then there exists a proof tree \mathcal{T} rooted at ψ with nodes $\psi_1, \dots, \psi_m \equiv \psi$. Then the following stronger claim proves the required result.

Claim. $\Gamma \vdash \phi \rightarrow \psi_i \text{ for all } i, 1 \leq i \leq m$.

\vdash By induction on $k = \ell(\psi_1) - \ell(\psi_i)$ in \mathcal{T} (see definition 2.11 for ℓ).

Basis. $k = \ell(\psi_1) - \ell(\psi_i) = 0$. Then ψ_i is either a premise or an axiom. We have the following cases to consider.

Case $\psi_i \equiv \phi$. Then the claim follows from **reflexivity** and monotonicity (theorem 13.1).

Case $\psi_i \in \Gamma \text{ or } \psi_i \text{ is an axiom}$. In either case there exists a subtree \mathcal{T}_i (of \mathcal{T}) rooted at ψ_i which may be used to construct the tree \mathcal{T}' as follows. Assume there are i' steps in the proof of ψ_i .

$$\frac{i' \frac{\nwarrow \mathcal{T}_i \nearrow}{\psi_i} \quad i' + 1 \frac{\psi_i \rightarrow (\phi \rightarrow \psi_i)}{}}{i' + 2 \frac{}{\phi \rightarrow \psi_i}}$$

which proves the claim.

Induction Hypothesis (IH).

$\Gamma \vdash \phi \rightarrow \psi_i$ for all i such that $\ell(\psi_i) > l$ for some $l \geq \ell(\psi_m)$.

Induction Step. Since ψ_l is a non-leaf node it is neither an axiom nor a premise and must have been obtained by virtue of the rule **MP** applied to its immediate successors say ψ_i and ψ_j with $i \neq j$ such that $\ell(\psi_i), \ell(\psi_j) > l$. Without loss of generality we may assume $\psi_j \equiv \psi_i \rightarrow \psi_l$.

By the induction hypothesis, we know $\Gamma \vdash \phi \rightarrow \psi_i$ and $\Gamma \vdash \phi \rightarrow \psi_j$. Hence there exist proof trees \mathcal{T}'_i of i' nodes rooted at $\phi \rightarrow \psi_i$ and \mathcal{T}'_j of j' nodes rooted at $\phi \rightarrow \psi_j \equiv \phi \rightarrow (\psi_i \rightarrow \psi_l)$ respectively. We construct the tree \mathcal{T}' rooted at $\phi \rightarrow \psi_l$ from \mathcal{T}'_i and \mathcal{T}'_j as follows.

$$j' + 2 \frac{j' \frac{\nearrow \mathcal{T}'_j \searrow}{\phi \rightarrow (\psi_i \rightarrow \psi_l)} \quad j' + 1 \frac{(\phi \rightarrow (\psi_i \rightarrow \psi_l)) \rightarrow ((\phi \rightarrow \psi_i) \rightarrow (\phi \rightarrow \psi_l))}{(\phi \rightarrow \psi_i) \rightarrow (\phi \rightarrow \psi_l)}}{j' + i' + 3} \frac{\nearrow \mathcal{T}'_i \searrow}{\phi \rightarrow \psi_i}$$

where $j' + 1$ is an instance of **S**, and $j' + 2$ and $j' + i' + 3$ are both applications of **MP** to their respective immediate successors in the tree.

\dashv

(\Leftarrow). Assume $\Gamma \vdash \phi \rightarrow \psi$. Let \mathcal{T} be a formal proof tree rooted at $\phi \rightarrow \psi$ with m nodes for some $m > 0$. By monotonicity (theorem 13.1) $\Gamma, \phi \vdash \phi \rightarrow \psi$ is proven by the same tree. We may extend \mathcal{T} to the tree \mathcal{T}' by adding a new $(m + 1)$ -st leaf node ϕ and creating the $(m + 2)$ -nd root node ψ .

$$\frac{m+2}{\begin{array}{c} m \frac{\nearrow \mathcal{T} \nearrow}{\phi \rightarrow \psi} \\ \psi \end{array}} \quad \frac{m+1}{\phi}$$

\mathcal{T}' is a proof of $\Gamma, \phi \vdash \psi$. QED

■

About Formal Proofs

- Since a formal proof is a tree, it is acyclic (i.e. there is no circularity in the proof).
- Every formal proof is a *finite* tree i.e. a proof is a finitary object.

Questions.

1. What if the set of assumptions is infinite?
2. Are there statements which have only infinite proofs and no finite proof?

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

350 OF 783

[QUIT](#)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

351 OF 783

QUIT

Lecture 14: Derived Rules

Tuesday 30 Aug 2011

1. Simplifying Proofs
2. Derived Rules
3. The Sequent Form
4. Proof trees in sequent form
5. Transitivity of Conditional
6. Derived Double Negation Rules
7. Derived Operators
8. Rules for Derived Operators

Simplifying Proofs

- The deduction theorem allows “*movement*” of sub-formulae between the set (sequence) of assumptions and the formula to be proven.
- Hence the set (sequence) of formulae which form the assumptions is an important part of the proof.
- We use the notion of a *sequent* to formalize this *movement* which may take place at any stage.

Definition 14.1 A sequent is a meta-formula of the form $\Gamma \vdash \phi$.

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 “ \vdash ”.
- 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.

The Sequent Form

Let Γ be a sequence of formulae.

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

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

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

$$MP. \frac{\begin{array}{c} \Gamma \vdash X \rightarrow Y \\ \Gamma \vdash X \end{array}}{\Gamma \vdash Y}$$

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

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

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

Proof trees in sequent form

Theorem 14.2 For all ϕ , ψ and χ ,

$$(\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi))$$

Proof: Let $\Gamma_1 = \phi \rightarrow \psi$, $\Gamma_2 = \Gamma_1, \psi \rightarrow \chi$ and $\Gamma_3 = \Gamma_2, \phi$

$$\begin{array}{c} \text{MP} \frac{\Gamma_3 \vdash \phi \rightarrow \psi \quad \Gamma_3 \vdash \phi}{\Gamma_3 \vdash \psi} \quad \Gamma_3 \vdash \psi \rightarrow \chi \\ \text{MP} \frac{}{\Gamma_3 \vdash \psi} \quad \text{DT} \Rightarrow \frac{\Gamma_2, \phi \vdash \chi}{\Gamma_2 \vdash \phi \rightarrow \chi} \\ \text{DT} \Rightarrow \frac{\Gamma_2 \vdash \phi \rightarrow \chi}{\Gamma_1 \vdash (\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi)} \\ \text{DT} \Rightarrow \frac{\Gamma_1 \vdash (\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi)}{\vdash (\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi))} \end{array}$$

QED



Transitivity of Conditional

From theorem 14.2 we get a derived axiom schema

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

But equivalently by applying the derived rule $DT \Leftarrow$ to $T \rightarrow$ above we also get a derived rule of inference which is often more convenient to use.

$$T \Rightarrow . \frac{\Gamma \vdash X \rightarrow Y \quad \Gamma \vdash Y \rightarrow Z}{\Gamma \vdash X \rightarrow Z}$$

Exercise 14.1

1. Prove that each of the axiom schemas in \mathcal{H}_0 represents a collection of tautologies.
2. Prove that *Modus Ponens in sequent form* preserves logical consequence i.e. if $\Gamma \models \phi \rightarrow \psi$ and $\Gamma \models \phi$ then $\Gamma \models \psi$.
3. Using the above prove that the proof system \mathcal{H}_0 is sound i.e. If $\Gamma \vdash_{\mathcal{H}_0} \psi$ then $\Gamma \models \psi$.
4. Find the fallacy in the following proof of theorem 14.2. Assume Γ_1 , Γ_2 and Γ_3 are as in the proof of theorem 14.2.

$$\frac{\text{DT} \Rightarrow \frac{\Gamma_3 \vdash \psi \rightarrow \chi}{\Gamma_2 \vdash \phi \rightarrow (\psi \rightarrow \chi)}}{\text{MP} \frac{\text{S}}{\frac{\Gamma_2 \vdash (\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \chi))}{\Gamma_2 \vdash (\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \chi)}} \quad \Gamma_2 \vdash \phi \rightarrow \psi}$$
$$\frac{\text{DT} \Rightarrow \frac{\Gamma_2 \vdash \phi \rightarrow \chi}{\Gamma_1 \vdash (\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi)}}{\text{DT} \Rightarrow \frac{}{\vdash (\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi))}}$$

5. Prove the following derived rule of inference (You may use any of the derived rules of inference in addition to the usual proof rules).

$$\boxed{\text{R2} \Rightarrow . \frac{\Gamma \vdash X \rightarrow (Y \rightarrow Z) \quad \Gamma \vdash Y}{\Gamma \vdash X \rightarrow Z}}$$

6. Could we have consequently reordered our theorems by first proving $\mathbf{R2} \rightarrow$ and then proving $\mathbf{T} \rightarrow$? Discuss whether there is anything fallacious in this approach.

Derived Double Negation Rules

$$\text{DNE. } \frac{\Gamma \vdash \neg\neg X}{\Gamma \vdash X}$$

$$\text{DNI. } \frac{\Gamma \vdash X}{\Gamma \vdash \neg\neg X}$$

The following proof trees yield proofs of the derived double negation **elimination** and **introduction** rules respectively.
Alternatively we may regard them as derived axiom schemas

$$\text{DNE} \rightarrow . \quad \frac{}{\Gamma \vdash \neg\neg X \rightarrow X}$$

$$\text{DNI} \rightarrow . \quad \frac{}{\Gamma \vdash X \rightarrow \neg\neg X}$$

Proof of derived rule DNE and axiom schema DNE \rightarrow

Proof:

$$\begin{array}{c} \mathsf{K} \frac{}{\neg\neg\phi \rightarrow (\neg\phi \rightarrow \neg\neg\phi)} \\ \mathsf{T} \Rightarrow \frac{\mathsf{R2} \Rightarrow \frac{\mathsf{N} \frac{}{(\neg\phi \rightarrow \neg\neg\phi) \rightarrow ((\neg\phi \rightarrow \neg\phi) \rightarrow \phi)}}{(\neg\phi \rightarrow \neg\neg\phi) \rightarrow \phi}}{\mathsf{R} \Rightarrow \frac{}{\neg\phi \rightarrow \neg\phi}} \\ \mathsf{DT} \Leftarrow \frac{\neg\neg\phi \rightarrow \phi}{\neg\neg\phi \vdash \phi} \end{array}$$

QED ■

Proof of derived rule DNI and axiom schema DNI \rightarrow

Proof:

$$\begin{array}{c} \mathsf{K} \frac{}{\phi \rightarrow (\neg\neg\neg\phi \rightarrow \phi)} \\ \mathsf{T} \Rightarrow \frac{\mathsf{MP} \frac{\mathsf{N} \frac{}{(\neg\neg\neg\phi \rightarrow \neg\phi) \rightarrow ((\neg\neg\neg\phi \rightarrow \phi) \rightarrow \neg\neg\phi)}}{(\neg\neg\neg\phi \rightarrow \phi) \rightarrow \neg\neg\phi}}{\mathsf{DNE} \frac{}{\neg\neg\neg\phi \rightarrow \neg\phi}} \\ \mathsf{DT} \Leftarrow \frac{\phi \rightarrow \neg\neg\phi}{\phi \vdash \neg\neg\phi} \end{array}$$

QED ■

Exercise 14.2

1. Prove the axiom schema

$$\boxed{\mathsf{N'}. \quad \frac{}{(\neg Y \rightarrow \neg X) \rightarrow (X \rightarrow Y)}}$$

A deduction theorem variant of this schema is also called the modus tollens rule or the contrapositive rule.

2. A variant of the system \mathcal{H}_0 is the system \mathcal{H}'_0 obtained by replacing the schema N by N' .

(a) Prove the axiom schema N in the system \mathcal{H}'_0 .

(b) Prove the double negation rules DNE and DNI in \mathcal{H}'_0 .

3. Prove the following axiom schemas in \mathcal{H}_0 . In each case you are allowed to use any version of the theorems previously proven.

(a) $\boxed{\perp. \quad \frac{}{\neg X \rightarrow (X \rightarrow Y)}} \quad$ What can you conclude about the system \mathcal{H}_0 from your proof?

(b) $\boxed{\mathsf{N}''. \quad \frac{}{(X \rightarrow Y) \rightarrow (\neg Y \rightarrow \neg X)}}$

(c) N2.

$$X \rightarrow (\neg Y \rightarrow \neg(X \rightarrow Y))$$

(d) C.

$$(X \rightarrow Y) \rightarrow ((\neg X \rightarrow Y) \rightarrow Y)$$

Derive the proof by cases rule

Cases.

$$\frac{\Gamma, X \vdash Y \quad \Gamma, \neg X \vdash Y}{\Gamma \vdash Y}$$

(e) Derive the proof by contradiction also called the indirect proof method rule I in the system \mathcal{H}_0 .

$$\text{I. } \frac{\Gamma, X \vdash \neg Y \quad \Gamma, X \vdash Y}{\Gamma \vdash \neg X}$$

4. Prove the derived axiom

C2. $\frac{}{\Gamma \vdash (\neg X \rightarrow X) \rightarrow X}$

Derived Operators

\top	$\stackrel{df}{=}$	$\phi \rightarrow \phi$	for all ϕ
\perp	$\stackrel{df}{=}$	$\neg(\phi \rightarrow \phi)$	for all ϕ
$\phi \vee \psi$	$\stackrel{df}{=}$	$\neg\phi \rightarrow \psi$	
$\phi \wedge \psi$	$\stackrel{df}{=}$	$\neg(\phi \rightarrow \neg\psi)$	
$\phi \leftrightarrow \psi$	$\stackrel{df}{=}$	$\neg((\phi \rightarrow \psi) \rightarrow \neg(\psi \rightarrow \phi))$	

Several other binary and other operators of varying arities may be defined.

Rules for Derived Operators

Corresponding to each derived operator defined as $O(X_1, \dots, X_n) \stackrel{df}{=} \omega(X_1, \dots, X_n)$ where ω is constructed only from the set $\{\neg, \rightarrow\}$ we have the introduction and elimination rules.

$$\text{OE. } \frac{\Gamma \vdash O(X_1, \dots, X_n)}{\Gamma \vdash \omega(X_1, \dots, X_n)}$$

$$\text{OI. } \frac{\Gamma \vdash \omega(X_1, \dots, X_n)}{\Gamma \vdash O(X_1, \dots, X_n)}$$

Gentzen's System

Natural Deduction

- Gentzen's Natural Deduction system is not a minimal system, instead it is somewhat more natural in the sense that it has explicit *introduction* and *elimination* rules for each operator.
- We present sequent version of the system in the following.
- Further there is some redundancy in the rules. Not all the rules may actually be useful, but they possess a pleasing symmetry.
- However, it is necessary to prove both the soundness and the completeness of the system.
- We refer to the system as \mathcal{G}_0 .

Natural Deduction: 1

	Introduction	Elimination
\perp	$\perp I. \frac{\Gamma \vdash X \wedge \neg X}{\Gamma \vdash \perp}$	$\perp E. \frac{\Gamma \vdash \perp}{\Gamma \vdash X}$
\top	$\top I. \frac{}{\Gamma \vdash \top}$	$\top E. \frac{\Gamma \vdash \top}{\Gamma \vdash X \vee \neg X}$

Natural Deduction: 2

	Introduction	Elimination
\neg	$\neg I.$ $\frac{\Gamma, X \vdash \perp}{\Gamma \vdash \neg X}$	$\neg E.$ $\frac{\Gamma, \neg X \vdash \perp}{\Gamma \vdash X}$
$\neg\neg$	$\neg\neg I.$ $\frac{\Gamma \vdash X}{\Gamma \vdash \neg\neg X}$	$\neg\neg E.$ $\frac{\Gamma \vdash \neg\neg X}{\Gamma \vdash X}$

Natural Deduction: 3

	Introduction	Elimination
\vee	$\vee I 1. \frac{\Gamma \vdash X}{\Gamma \vdash X \vee Y}$ $\vee I 2. \frac{\Gamma \vdash Y}{\Gamma \vdash X \vee Y}$	$\vee E. \frac{\Gamma \vdash X \vee Y}{\begin{array}{l} \Gamma \vdash X \vee Y \\ \Gamma \vdash X \rightarrow Z \\ \Gamma \vdash Y \rightarrow Z \end{array}}$

Natural Deduction: 4

	Introduction	Elimination	
\wedge	$\wedge I. \frac{\Gamma \vdash X}{\Gamma \vdash X \wedge Y}$	$\wedge E1. \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash X}$	$\wedge E2. \frac{\Gamma \vdash X \wedge Y}{\Gamma \vdash Y}$

Natural Deduction: 5

	Introduction	Elimination	
→	→ I. $\frac{\Gamma, X \vdash Y}{\Gamma \vdash X \rightarrow Y}$	→ E. $\frac{\Gamma \vdash X}{\Gamma \vdash Y}$	
↔	↔ I. $\frac{\Gamma \vdash X \rightarrow Y \quad \Gamma \vdash Y \rightarrow X}{\Gamma \vdash X \leftrightarrow Y}$	↔ E1. $\frac{\Gamma \vdash X \leftrightarrow Y}{\Gamma \vdash X \rightarrow Y}$	↔ E2. $\frac{\Gamma \vdash X \leftrightarrow Y}{\Gamma \vdash Y \rightarrow X}$

Exercise 14.3

1. Prove the *logical equivalences* of \mathcal{P}_0 using the system \mathcal{H}_0 .
2. Prove the non-obvious *logical equivalences* of \mathcal{P}_0 in the system \mathcal{G}_0 .
3. Derive each of the rules of \mathcal{G}_0 from the system \mathcal{H}_0 . You may use the rules OE and OI as and when needed for each operator.
4. Derive the axiom schemas K, S and N in \mathcal{G}_0 .

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

374 OF 783

QUIT

Lecture 15: The Hilbert System: Soundness

Tuesday 06 September 2011

1. Formal Theory: Issues
2. Formal Theory: Incompleteness
3. Soundness of Formal Theories
4. Soundness of the Hilbert System
5. Soundness of the Hilbert System

Formal Theory: Issues

The major questions concerning any formal theory are two-fold:

Soundness. Is the theory sound? Especially considering that we may not have well-defined models in which to test the truth or falsehood of statements.

Completeness. Is the theory complete? That is, is every fact provable from the axioms and inference rules of the theory?

Formal Theory: Incompleteness

Suppose a given formal theory is incomplete. There are several possibilities.

Question 1. Can the theory be made complete by adding or changing some axioms and inference rules (without making the theory inconsistent)?

Question 2. Is the theory *inherently incomplete*? That is, is there no way of achieving completeness by adding only a finite number of new axioms and inference rules?

Soundness of Formal Theories

Given that the proof theory may be the only finitary tool available to us in reasoning about some domain we need to define the notion of consistency of the theory in terms of the proof-theoretic notions.

Definition 15.1 A *formal theory is unsound if every wff is a theorem. Otherwise it is said to be sound.*

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.



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 .

Proof of lemma 15.2

Proof:

1. We prove the case of any instance of the axiom schema K. We need to show that for all ϕ and ψ , $\phi \rightarrow (\psi \rightarrow \phi)$ is a tautology. Suppose it is not a tautology. Then there exists a truth assignment τ such that $\mathcal{T}[\![\phi \rightarrow (\psi \rightarrow \phi)]\!]_{\tau} = 0$ which is possible only if $\mathcal{T}[\![\phi]\!]_{\tau} = 1$ and $\mathcal{T}[\![\psi \rightarrow \phi]\!]_{\tau} = 0$ which in turn is possible only if $\mathcal{T}[\![\psi]\!]_{\tau} = 1$ and $\mathcal{T}[\![\phi]\!]_{\tau} = 0$ which is impossible. Hence there is no such truth assignment. So $\phi \rightarrow (\psi \rightarrow \phi)$ must be a tautology.

A similar reasoning may be applied to the axiom schemas S and N.

2. Assume for some ϕ and ψ that ϕ and $\phi \rightarrow \psi$ are tautologies but ψ is not. It is easy to see that for any truth assignment τ , $\mathcal{T}[\![\psi]\!]_{\tau} = 0$

implies $\mathcal{T}[\![\phi]\!]_{\tau} = 0$ contradicting the assumption that ϕ is a tautology.

QED



Soundness of the Hilbert System

Theorem 15.3 *Every formal theorem of \mathcal{H}_0 is a tautology.*



The theorem follows by induction on the heights of proof trees, since every leaf would be a tautology and every internal node preserves tautologousness.

Corollary 15.4 *The system \mathcal{H}_0 is sound.*

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

384 OF 783

[QUIT](#)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

385 OF 783

QUIT

Lecture 16: The Hilbert System: Completeness

Wednesday 07 Sep 2011

1. Towards Completeness
2. Towards Truth-tables
3. The Truth-table Lemma
4. The Completeness Theorem

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.

Towards Truth-tables

1. Restricting ourselves to showing that every tautology is provable in \mathcal{H}_0 is sufficient.
2. But we proceed to show that every truth table can be *simulated* as a proof in \mathcal{H}_0 , thereby capturing all of the semantic features of the language \mathcal{P}_0 in its proof theory.

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

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



Proof of lemma 16.1

Proof: By induction on the number n of operators in ϕ . Let $\Gamma = \{p_1^*, \dots, p_k^*\}$.

Basis. $n = 0$. Then ϕ is an atom say, $\phi \equiv p_1$. The claim then trivially follows since $p_1^* \equiv \phi^*$.

Induction Hypothesis (IH).

The claim holds for all wffs with less than $n \geq 0$ occurrence of the operators.

Induction Step. Suppose ϕ is a wff with n operators. Then there are two cases to consider.

Case $\phi \equiv \neg\psi$, where ψ has less than n operators. Then by the induction hypothesis we have

have a proof tree $\frac{\Gamma \vdash \psi^*}{\Gamma \vdash \psi}$.

Subcase $\mathcal{T}[\psi]_\tau = 1$. Then $\mathcal{T}[\phi]_\tau = 0$ and $\psi^* \equiv \psi$ and $\phi^* \equiv \neg\phi \equiv \neg\neg\psi$. Then we have the following deduction.

$$\frac{\text{DNI} \rightarrow \frac{\Gamma \vdash \psi \rightarrow \neg\neg\psi \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \psi}}{\Gamma \vdash \neg\neg\psi \equiv \phi^*}}{\text{MP}}$$

Subcase $\mathcal{T}[\psi]_\tau = 0$. Then $\mathcal{T}[\phi]_\tau = 1$ and $\psi^* \equiv \neg\psi$ and $\phi^* \equiv \phi \equiv \neg\psi$. By the induction hypothesis we have $\Gamma \vdash \neg\psi \equiv \phi^*$.

Case $\phi \equiv \psi \rightarrow \chi$. where each of ψ and χ has less than n operators. By the induction hypothesis there exist proof trees $\frac{\Gamma \vdash \psi^*}{\mathcal{T}_1}$ and $\frac{\Gamma \vdash \chi^*}{\mathcal{T}_2}$. Here again we have three subscases.

Subcase $\mathcal{T}[\psi]_\tau = 0$. We have $\psi^* \equiv \neg\psi$ so tree \mathcal{T}_1 is $\frac{\Gamma \vdash \neg\psi}{\mathcal{T}_1}$, $\mathcal{T}[\phi]_\tau = 1$ and $\phi^* \equiv \phi \equiv \psi \rightarrow \chi$. We then have the proof tree

$$\text{MP} \frac{\frac{\perp \quad \frac{\Gamma \vdash \neg\psi \rightarrow (\psi \rightarrow \chi)}{\Gamma \vdash \psi \rightarrow \chi \equiv \phi^*}}{\mathcal{T}_1}}{\Gamma \vdash \neg\psi}$$

which proves the claim.

Subcase $\mathcal{T}[\chi]_\tau = 1$. Then $\chi^* \equiv \chi$ and $\mathcal{T}[\phi]_\tau = 1$ and $\phi^* \equiv \phi \equiv \psi \rightarrow \chi$. Hence tree \mathcal{T}_2 is $\frac{\Gamma \vdash \chi}{\mathcal{T}_2}$. We may then construct the following proof tree to prove the claim.

$$\text{MP} \frac{\text{K} \frac{}{\Gamma \vdash \chi \rightarrow (\psi \rightarrow \chi)} \quad \frac{\nwarrow \mathcal{T}_2 \nearrow}{\Gamma \vdash \chi}}{\Gamma \vdash \psi \rightarrow \chi \equiv \phi^*}$$

Subcase $\mathcal{T}[\psi]_\tau = 1$ and $\mathcal{T}[\chi]_\tau = 0$. Then $\psi^* \equiv \psi$ and $\chi^* \equiv \neg\chi$ and $\mathcal{T}[\phi]_\tau = 0$ from which we get $\phi^* \equiv \neg(\psi \rightarrow \chi)$. By induction hypotheses we therefore have the trees $\frac{\nwarrow \mathcal{T}_1 \nearrow}{\Gamma \vdash \psi}$ and $\frac{\nwarrow \mathcal{T}_2 \nearrow}{\Gamma \vdash \neg\chi}$ using which we construct the following proof tree to prove our claim.

$$\text{N2} \frac{}{\Gamma \vdash \psi \rightarrow (\neg\chi \rightarrow \neg(\psi \rightarrow \chi))} \quad \text{MP} \frac{\text{MP} \frac{\frac{\nwarrow \mathcal{T}_1 \nearrow}{\Gamma \vdash \psi} \quad \frac{\nwarrow \mathcal{T}_2 \nearrow}{\Gamma \vdash \neg\chi}}{\Gamma \vdash \neg\chi \rightarrow \neg(\psi \rightarrow \chi)}}{\Gamma \vdash \neg(\psi \rightarrow \chi) \equiv \phi^*}$$

QED ■

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 .*



Proof of the Hilbert Completeness Theorem 16.2

Proof: Let ϕ be a tautology expressed in the language \mathcal{L}_0 and let $atoms(\phi) = \{p_1, \dots, p_k\}$. Since every row of the truth-table for ϕ assigns it a truth value 1 we have $\phi^* \equiv \phi$. Each bit-string $s_k \in \{0, 1\}^k$ indexes a row of the truth table containing 2^k distinct rows. Let $\Gamma_{s_k} = \{p_i^* \mid 1 \leq i \leq k\}$ denote the set of assumptions for the s_k -th row of the truth table. By the truth table lemma

(lemma 16.1), there exists a distinct proof tree, $\Gamma_{s_k} \vdash \phi$ for each such s_k . For each bit-string

$s_j \in \{0, 1\}^j$, $0 < j \leq k$, we construct the proof tree from the proof trees $\Gamma_{s_{j-1}0} \vdash \phi$

and $\Gamma_{s_{j-1}1} \vdash \phi$, where $s_{j-1}0$ and $s_{j-1}1$ are the two bit-strings of length j , which differ only in the right-most bit.

Note that $s_0 = \epsilon$ is the empty string, $\Gamma_\epsilon = \emptyset$ and we need to construct the proof tree from

the 2^k distinct proof trees $\Gamma_{s_k} \vdash \phi$.

Now consider any two proof trees whose indexes differ only in the rightmost bit. That is, for any

$s_{j-1} \in \{0, 1\}^{j-1}$, we have the proof trees $\Gamma_{s_{j-1}1} \vdash \phi$ and $\Gamma_{s_{j-1}0} \vdash \phi$. We construct the proof tree $\Gamma_{s_{j-1}} \vdash \phi$ as follows.

$$\frac{\text{DT} \Rightarrow \frac{\text{DT} \Rightarrow \frac{\Gamma_{s_{j-1}0} \vdash \phi}{\Gamma_{s_{j-1}} \vdash \neg p_j \rightarrow \phi}}{\text{MP} \frac{\text{DT} \Rightarrow \frac{\Gamma_{s_{j-1}1} \vdash \phi}{\Gamma_{s_{j-1}} \vdash p_j \rightarrow \phi}}{\text{MP} \frac{\text{C} \frac{\Gamma_{s_{j-1}} \vdash (p_j \rightarrow \phi) \rightarrow ((\neg p_j \rightarrow \phi) \rightarrow \phi)}{\Gamma_{s_{j-1}} \vdash (\neg p_j \rightarrow \phi) \rightarrow \phi}}{\Gamma_{s_{j-1}} \vdash \phi}}$$

We can thus eliminate the atom p_j from the assumptions by applying the above proof procedure to all pairs of proof trees whose assumptions differ only in the value of p_j^* .

Thus the 2^k proof trees are combined pairwise to produce 2^{k-1} proof trees that are independent of the atom p_k . Proceeding in a like manner we may eliminate all the atoms one by one by using similar proof constructions so that finally we obtain a single monolithic proof tree $\Gamma_\epsilon \vdash \phi$ where $\Gamma_\epsilon = \emptyset$, thus concluding the proof that the tautology ϕ is a formal theorem of \mathcal{H}_0 . QED ■

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

397 OF 783

[QUIT](#)

Lecture 17: Introduction to Predicate Logic

Friday 09 Sep 2011

1. Predicate Logic: Introduction-1
2. Predicate Logic: Introduction-2
3. Predicate Logic: Introduction-3
4. Predicate Logic: Symbols
5. Predicate Logic: Signatures
6. Predicate Logic: Syntax of Terms
7. Predicate Logic: Syntax of Formulae
8. Precedence Conventions
9. Predicates: Abstract Syntax Trees
10. Subterms
11. Variables in a Term
12. Bound Variables And Scope
13. Bound Variables And Scope: Example
14. Scope Trees
15. Free Variables
16. Bound Variables
17. Closure

Predicate Logic: Introduction-1

Example 17.1 *There are many kinds of arguments which cannot be proven in propositional logic and which require the notion of quantifiers. The most famous one perhaps containing only very basic and simple declarative statements is*

All humans are mortal.

Socrates is a human.

Therefore Socrates is mortal.

Predicate Logic: Introduction-2

The validity of this argument does not depend on any *propositional connectives* since there are none. Hence it is not provable in propositional logic.

However it is valid and its validity depends upon

- the *internal structure* of the sentences,
- the meaning of certain operative words and phrases such as “All”
- certain properties of objects e.g. “mortal”
- the description of certain classes by their properties and membership or other relations on these classes e.g. “is a”.

Predicate Logic: Introduction-3

1. The deeper relationships that exist between otherwise simple propositions require parametrisation of propositions so that the inter-relationships become clear.
2. Parametrised propositions are called *predicates*.
3. Each predicate specifies either a property (1-ary predicates) or a relationship between objects (n -ary predicates).
4. The propositions of propositional logic are 0-ary predicates.
5. Mathematical theories are often about collections of infinite objects whose relationships and inter-relationships are finite expressions involving *functions*, *relations* and *expressions* involving them.

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.

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.

Predicate Logic: Syntax of Terms

Definition 17.3 *Given a signature Σ , the set $T(\Sigma)$ of Σ -terms is defined inductively by the following grammar*

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

where $f : s^m \rightarrow s \in \Sigma$ and $t_1, \dots, t_m \in T(\Sigma)$. If $m = 0$ then $f()$ is called a **constant** and simply written f . We usually use the symbols a, b, c, \dots to denote constants.

Predicate Logic: Syntax of Formulae

Definition 17.4 Given a signature Σ and the set $T(\Sigma)$ of Σ -terms.

- A **Σ -atomic formula** or **Σ -atom** is a string of the form $p(t_1, \dots, t_n)$ where $p : s^n \in \Sigma$. $A(\Sigma)$ is the set of Σ -atoms.
- $\Omega_1 = \Omega_0 \cup \{\forall x, \exists x \mid x \in V\}$
- The set of $\mathcal{P}_1(\Sigma)$ of Σ -formulas is defined inductively by the following grammar.

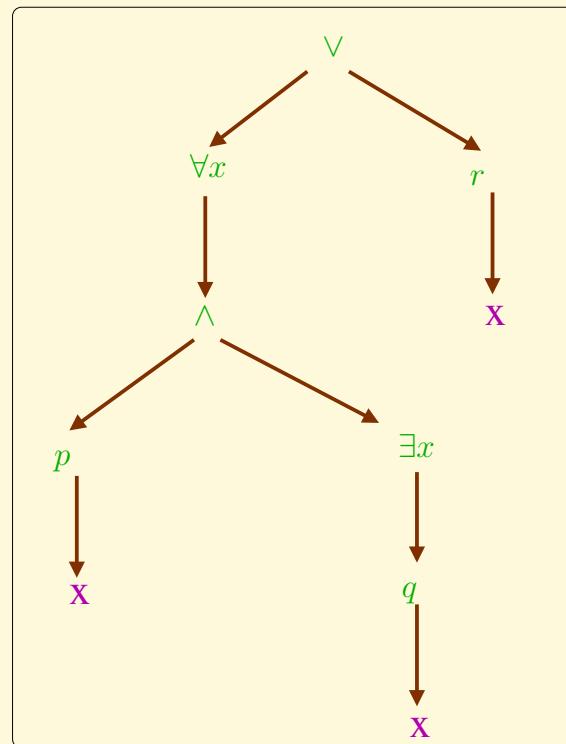
$\phi, \psi ::=$	\perp	$ $	T
	$ \quad p(t_1, \dots, t_n) \in A(\Sigma)$	$ $	$(\neg \phi)$
	$ \quad (\phi \wedge \psi)$	$ $	$(\phi \vee \psi)$
	$ \quad (\phi \rightarrow \psi)$	$ $	$(\phi \leftrightarrow \psi)$
	$ \quad \forall x[\phi]$	$ $	$\exists x[\phi]$

Precedence Conventions

- The operator precedence conventions are as before.
- The two new operators are called the universal quantifier (\forall) and existential quantifier (\exists) respectively and are parameterised by variables.
- The scope of the (variable in a) quantified formula is delimited by the matching pair of brackets ([and]).
- If a formula ϕ is preceded by several quantifiers (e.g. $\forall x[\exists y[\forall z[\phi]]]$) we collapse the scoping brackets where there is no ambiguity (e.g $\forall x\exists y\forall z[\phi]$).
- We will think of both Σ -terms and Σ -formulae as abstract syntax trees. The brackets delimiting the scope of a quantified variable then become redundant.

Predicates: Abstract Syntax Trees

Example 17.5 Let p , q and r be unary predicates. The first-order logic formula $\forall x[p(x) \wedge \exists x[q(x)]] \vee r(x)$



Subterms

Definition 17.6 For each term t , $ST(t)$ denotes the set of subterms of t (including t itself). The set of proper subterms of t is the set $ST(t) - \{t\}$.

t	$depth$	$size$	ST
$c()$	1	1	$\{t\}$
x	1	1	$\{t\}$
$f(t_1, \dots, t_n)$	$1 + Max_{i=1}^n depth(t_i)$	$1 + \sum_{i=1}^n size(t_i)$	$\{t\} \cup \bigcup_{i=1}^n ST(t_i)$

Variables in a Term

For any term t , $Var(t)$ denotes the set of all variables that occur in t . These functions may be defined by induction on the structure of terms as follows.

Definition 17.7

t	$Var(t)$
$c()$	\emptyset
x	$\{x\}$
$f(t_1, \dots, t_n)$	$\bigcup_{1 \leq i \leq n} Var(t_i)$

Bound Variables And Scope

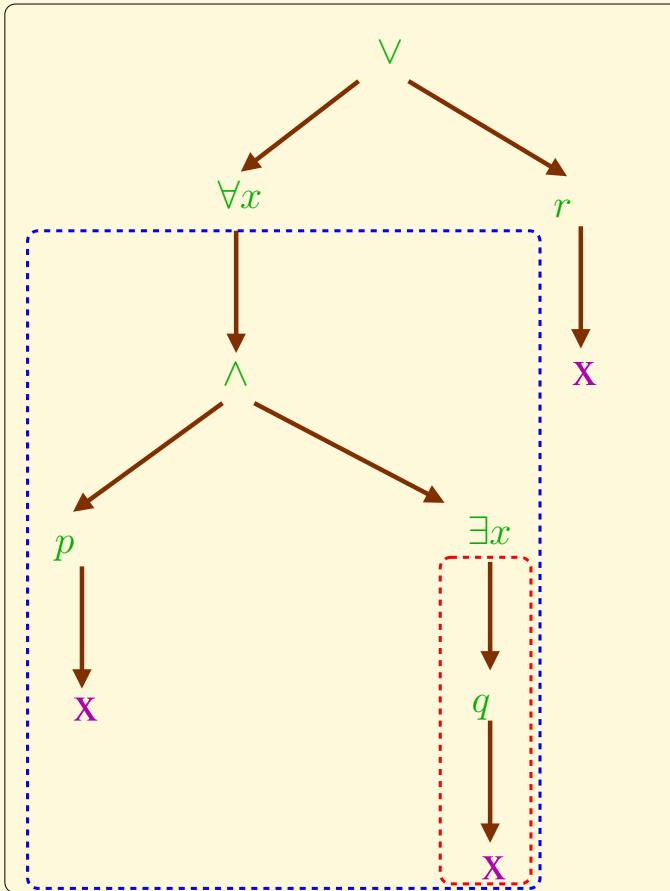
Definition 17.8 *In a formula of the form $\mathcal{O}x[\psi]$ the variable x is said to be **bound** by the quantifier \mathcal{O} . The brackets $[\dots]$ delimit the **scope** of the binding.*

Example 17.9 *In the **abstract syntax tree** of the predicate*

$$\forall x[p(x) \wedge \exists x[q(x)]] \vee r(x)$$

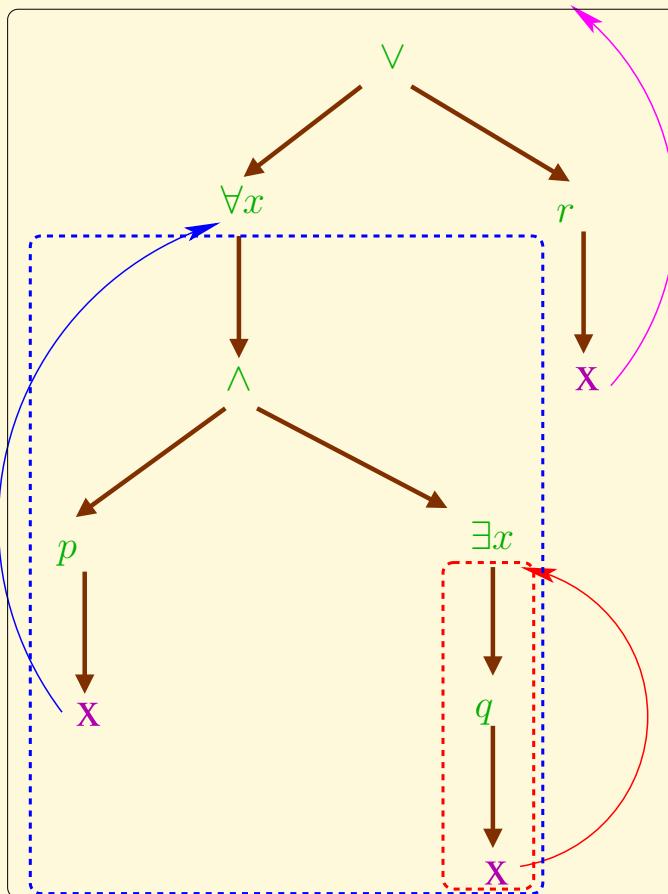
*the scopes of the various bindings are indicated by dashed boxes. Notice that the **red** scope is a “hole” in the **blue** scope.*

Bound Variables And Scope: Example



Scope Trees

The abstract syntax tree also determines the scope of the individual bound variables.



Free Variables

Definition 17.10 For any predicate ϕ the set of free variables occurring in it (denoted $FV(\phi)$) and the set of sub-formulae (denoted $SF(\phi)$) are defined by induction on the structure of predicates.

ϕ	$FV(\phi)$	$SF(\phi)$	Condition
$p(t_1, \dots, t_n)$	$\bigcup_{1 \leq i \leq n} Var(t_i)$	$\{p(t_1, \dots, t_n)\}$	
$\neg\psi$	$FV(\psi)$	$\{\neg\psi\} \cup SF(\psi)$	
$o(\psi, \chi)$	$FV(\psi) \cup FV(\chi)$	$\{o(\psi, \chi)\} \cup SF(\psi) \cup SF(\chi)$	$o \in \Omega_0 - \{\neg\}$
$\mathcal{O}x[\psi]$	$FV(\psi) - \{x\}$	$\{\mathcal{O}x[\psi]\} \cup SF(\psi)$	$\mathcal{O} \in \{\forall, \exists\}$

Bound Variables

Definition 17.11 If $\mathcal{O}x[\psi]$ is a sub-formula of some formula ϕ then ψ is said to be the scope of the quantifier $\mathcal{O}x$ and every free occurrence of the variable x in the formula ψ is said to be bound in the scope of the quantifier $\mathcal{O}x$ in which it occurs.

Notice that if $\mathcal{O}x[\chi]$ is a sub-formula of ψ in the definition 17.11 then any $x \in FV(\chi)$ is not a free variable of ψ .

We may write $\phi(x_1, \dots, x_m)$ to indicate that $FV(\phi) \subseteq \{x_1, \dots, x_m\}$.

Closure

Definition 17.12

1. A formula ϕ is called **closed** if $FV(\phi) = \emptyset$.
2. The **universal closure** of $\phi(x_1, \dots, x_m)$ denoted $\vec{\forall}[\phi]$ is defined as the formula $\forall x_1, \dots, x_m[\phi]$.
3. The **existential closure** of $\phi(x_1, \dots, x_m)$ denoted $\vec{\exists}[\phi]$ is defined as the formula $\exists x_1, \dots, x_m[\phi]$.
4. A **literal** is an atomic formula or its negation. For any literal λ , $\bar{\lambda}$ will denote its negation.

Exercise 17.1

1. Translate the following statements into first-order logic statements. (You may use the function symbols that are normally used in mathematics, e.g. “0” for zero, “=” for equality, “+” for addition etc.). The names x , y etc. stand for variables.

- Every number has a unique successor.
- Not every number has a predecessor.
- The sum of any two odd numbers is even.
- x is a prime.
- There is no largest prime.
- x is a divisor of y .
- x and y are relatively prime.
- Define the notion of greatest common divisor of two numbers as a ternary predicate $\text{gcd}(x, y, z)$ in terms of the previous parts. In other words, $\text{gcd}(x, y, z)$ stands for the statement

z is the greatest common divisor of x and y

2. Symbolize the following arguments in first order logic You may assume in each case that the universe of discourse contains the various relations mentioned as predicates (and nothing more!).

- *There is a man whom all men despise. Therefore at least one man despises himself.*
- *All hotels are expensive and depressing. Some hotels are shabby. Therefore some expensive hotels are shabby.*
- *Anyone who is loved loves everyone. No one loves Charlie Brown. Therefore no one loves anyone.*
- *Whoever visited the building was observed. Anyone who had observed Ajay, would have remembered him. Nobody remembered Ajay. Therefore Ajay did not visit the building.*
- *If all drugs are contaminated then all negligent technicians are scoundrels. If there are any drugs that are contaminated then all drugs are contaminated and unsafe. All pesticides are drugs. Only the negligent are absent-minded. Therefore if any technician is absent-minded, then if some pesticides are contaminated, then he is a scoundrel.*
- *Some criminal robbed the mansion. Whoever robbed the mansion broke in or had an accomplice among the servants. To break in one would have to smash the door or pick the lock. Only an expert locksmith could have picked the lock. Had anyone smashed the door, he would*

have been heard. Nobody was heard. If the criminal who robbed the mansion managed to fool the guard, he must have been a convincing actor. Nobody could rob the mansion unless he fooled the guard. No criminal could be both an expert locksmith and a convincing actor. Therefore some criminal had an accomplice among the servants.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

420 OF 783

QUIT

Lecture 18: The Semantics of Predicate Logic

Wednesday 14 Sep 2011

1. Structures
2. Notes on Structures
3. Infix Convention
4. Expansions and Reducts
5. Valuations and Interpretations
6. Evaluating Terms
7. Coincidence Lemma for Terms
8. Variants
9. Variant Notation
10. Semantics of Formulae
11. Notes on the Semantics

Structures

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

- a non-empty set $A = |\mathbf{A}|$ 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 $p_{\mathbf{A}} \in \mathbf{2} = \{0, 1\}$ 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.

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.

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

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

Expansions and Reducts

Definition 18.2 Let $\Sigma \subseteq \Omega$ be signatures. A Σ -structure \mathbf{A} is called a **reduct** to an Ω -structure \mathbf{B} if for all $f, p \in \Sigma$ iff

- $|\mathbf{A}| = |\mathbf{B}|$,
- $f_{\mathbf{A}} = f_{\mathbf{B}}$ for each $f \in \Sigma$ and
- $p_{\mathbf{A}} = p_{\mathbf{B}}$ for each $p \in \Sigma$

\mathbf{B} is also called an **expansion** of \mathbf{A} and is denoted $\mathbf{A} \triangleleft \mathbf{B}$

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.

Evaluating Terms

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

$$\begin{aligned}\mathcal{V}_{\mathbf{A}}[\![x]\!]_v &\stackrel{df}{=} v(x), & x \in V \\ \mathcal{V}_{\mathbf{A}}[\![f(t_1, \dots, t_m)]!]_v &\stackrel{df}{=} f_{\mathbf{A}}(\mathcal{V}[\![t_1]\!]_v, \dots, \mathcal{V}[\![t_m]\!]_v), & f : s^m \rightarrow s \in \Sigma\end{aligned}$$

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 “ \mathbf{A} ” is often omitted when the context makes clear the structure that is being referred to.

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 (\mathbf{A}, v) and (\mathbf{A}, v') , and a term t , if $v(x) = v'(x)$ for each $x \in \text{Var}(t)$, 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 $a_1, \dots, a_m \in |\mathbf{A}|$ in some Σ -interpretation (\mathbf{A}, v) , then we write $t_{\mathbf{A}}(a_1, \dots, a_m)$ 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.

Variants

Definition 18.7 Two valuations $v, v' : V \rightarrow |A|$ are said to be X -variants of each other (denoted $v =_X v'$ for any $X \subseteq V$ if for all $y \in V - X$, $v(y) = v'(y)$, i.e. they differ from each other at most in the values for variables from X .

Fact 18.8

1. For any $X \subseteq V$ and valuation v , v is an X -variant of itself i.e.

$$v =_X v.$$

2. $=_X$ is an equivalence relation on valuations.

Variant Notation

Notation.

1. When $X = \{\textcolor{violet}{x}\}$ is a singleton we refer to v and v' as $\textcolor{violet}{x}$ -variants and denote it by $v =_{\backslash \textcolor{violet}{x}} v'$.
2. If $v_{\textcolor{violet}{x}} =_{\backslash \textcolor{violet}{x}} v$ and $v_{\textcolor{violet}{x}}(\textcolor{violet}{x}) = \textcolor{red}{a} \in |\mathbf{A}|$ we write $v_{\textcolor{violet}{x}} = v[\textcolor{violet}{x} := \textcolor{red}{a}]$.
3. If $X = \{\textcolor{violet}{x}_1, \dots, \textcolor{violet}{x}_n\}$, $v_X =_{\backslash X} v$ and $v_X(\textcolor{violet}{x}_i) = \textcolor{red}{a}_i \in |\mathbf{A}|$, for each i , $1 \leq i \leq n$, then we write $v_X = v[\textcolor{violet}{x}_1 := \textcolor{red}{a}_1, \dots, \textcolor{violet}{x}_n := \textcolor{red}{a}_n]$.

Semantics of Formulae

Let $(\mathbf{A}, v_{\mathbf{A}})$ be a Σ -interpretation. Then $\mathcal{T}[\![\phi]\!]_v$ is defined by induction on the structure of ϕ . We omit the **propositional connectives** as being obvious and concentrate only on the other constructs.

$$\begin{aligned}\mathcal{T}_{\mathbf{A}}[\![p(t_1, \dots, t_n)]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} 1 & \text{if } (\mathcal{V}_{\mathbf{A}}[\![t_1]\!]_{v_{\mathbf{A}}, \dots, \mathcal{V}_{\mathbf{A}}[\![t_n]\!]_{v_{\mathbf{A}}}) \in p_{\mathbf{A}} \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{T}_{\mathbf{A}}[\![\forall x[\phi]]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \prod \{\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v'_{\mathbf{A}}} \mid v'_{\mathbf{A}} =_{\backslash x} v_{\mathbf{A}}\} \\ \mathcal{T}_{\mathbf{A}}[\![\exists x[\phi]]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \sum \{\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v'_{\mathbf{A}}} \mid v'_{\mathbf{A}} =_{\backslash x} v_{\mathbf{A}}\}\end{aligned}$$

Definitions of Σ, Π

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

Notes on the Semantics

1. Truth value is now evaluated under a *valuation* instead of a *truth assignment*.
2. A *valuation* defines a *truth assignment* to *parameterised propositions* depending upon the values of the parameters.
3. The set $\{\mathcal{T}_A[\phi]_{v'_A} \mid v'_A \text{ is an } x\text{-variant of } v_A\}$ is nonempty since v_A is also an x -variant of itself.
4. The number of x -variants of v equals the cardinality of $|A|$.
5. Even though there may be infinitely many x -variants of a valuation v_A , the set $\{\mathcal{T}_A[\phi]_{v'_A} \mid v'_A \text{ is an } x\text{-variant of } v_A\}$ cannot have more than *two* elements. Hence the sum and product are both *finitary*.

Exercise 18.1

1. Define interpretations on universes of discourse containing at least 3 elements and give a valuation in which the following hold. Assume p and q are atomic predicate symbols of any positive arity of your choice.
 - (a) $\neg(p \wedge \exists x[q] \rightarrow \exists x[p \wedge q])$
 - (b) $\neg(\exists x[p \rightarrow q] \rightarrow (\forall x[p] \rightarrow q))$
2. Prove that the following predicates have no models at all where p and q are atomic predicates (of any positive arity).
 - (a) $\neg(p \rightarrow \forall x[q] \rightarrow \forall x[p \rightarrow q])$
 - (b) $\neg((\forall x[p] \rightarrow q) \rightarrow \exists x[p \rightarrow q])$
3. Consider the universe of discourse to be the set of all nodes of graphs and let the atomic binary predicate symbol e stand for the edge relation on nodes, i.e. $e(x, y)$ stands for *there is an edge from x to y* . Further let “=” stand for the usual identity relation on nodes. What properties on graphs do the following first-order predicates define?

- (a) $\forall x[\exists y[\neg(x = y) \wedge e(x, y)]]$
- (b) $\forall x[\forall y[e(x, y) \rightarrow \neg(x = y)]]$
- (c) $\forall x[\forall y[\neg(x = y) \rightarrow (e(x, y) \rightarrow e(y, x))]]$
- (d) $\forall x[\forall y[\forall z[e(x, y) \wedge e(y, z) \rightarrow e(x, z)]]]$

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

436 OF 783

QUIT

Lecture 19: Substitutions

Friday 16 September 2011

1. Coincidence Lemma for Formulae
2. Substitutions
3. Instantiation of Terms
4. The Substitution Lemma for Terms
5. Admissibility
6. Instantiations of Formulae
7. The Substitution Lemma for Formulae

Coincidence Lemma for Formulae

The following lemma analogous to lemma 18.6 may be proved by induction on the structure of formulae.

Lemma 19.1 *Given two Σ -interpretations (\mathbf{A}, v) and (\mathbf{A}, v') , and a formula ϕ , if $v(x) = v'(x)$ for each $x \in FV(\phi)$, then $\mathcal{T}[\![\phi]\!]_v = \mathcal{T}[\![\phi]\!]_{v'}.$*



Proof of The Coincidence Lemma for formulae (lemma 19.1)

Variant Notation

Semantics of Formulae

Proof: By induction on the structure of ϕ . However the interesting cases are those of atomic predicates and quantified formulae.

Case $\phi \equiv p(t_1, \dots, t_n)$ where p is an n -ary predicate symbol. For each $x \in FV(p(t_1, \dots, t_n)) = \bigcup_{1 \leq i \leq n} Var(t_i)$ we have $v(x) = v'(x)$. Hence for each t_i we have $\mathcal{V}[t_i]_v = \mathcal{V}[t_i]_{v'}$ from which we get $\mathcal{T}[p(t_1, \dots, t_n)]_v = \mathcal{T}[p(t_1, \dots, t_n)]_{v'}$.

Case $\phi \equiv \mathcal{O}x[\psi]$, $\mathcal{O} \in \{\forall, \exists\}$. Assume $\psi \equiv \psi(x_1, \dots, x_n)$. We consider two cases.

Sub-case $x \notin FV(\psi)$. Then $x \notin \{x_1, \dots, x_n\}$ and hence for every v_x and v'_x which are x -variants of v and v' respectively we have $\mathcal{T}[\psi(x_1, \dots, x_n)]_{v_x} = \mathcal{T}[\psi(x_1, \dots, x_n)]_{v'_x}$ from which we obtain

$$\prod \{\mathcal{T}[\psi]_{v_x} \mid v_x =_{\setminus x} v\} = \prod \{\mathcal{T}[\psi]_{v'_x} \mid v'_x =_{\setminus x} v'\}$$

and

$$\sum \{\mathcal{T}[\psi]_{v_x} \mid v_x =_{\setminus x} v\} = \sum \{\mathcal{T}[\psi]_{v'_x} \mid v'_x =_{\setminus x} v'\}$$

which implies $\mathcal{T}[\phi]_{v_x} = \mathcal{T}[\phi]_{v'_x}$.

Sub-case $x \in FV(\psi)$. Then $FV(\psi) = \{x, x_1, \dots, x_n\}$. Let $T(x) = \{\mathcal{T}[\![\psi]\!]_{v_x} \mid v_x =_{\setminus x} v\}$ and $T'(x) = \{\mathcal{T}[\![\psi]\!]_{v'_x} \mid v'_x =_{\setminus x} v'\}$. Note that $T(x), T'(x) \in \{\{0\}, \{1\}, \{0, 1\}\}$. Assume for some $\odot \in \{\prod, \sum\}$, $\mathcal{T}[\![\phi]\!]_{v_x} \neq \mathcal{T}[\![\phi]\!]_{v'_x}$. Then $T(x) \neq T'(x)$ which implies there exists $a \in A = |\mathbf{A}|$ such that for $v_x = v[x := a]$ and $v'_x = v'[x := a]$, $\mathcal{T}[\![\psi]\!]_{v_x} \neq \mathcal{T}[\![\psi]\!]_{v'_x}$. But this is impossible since $FV(\psi) = \{x, x_1, \dots, x_n\}$, $v_x(x) = a = v'_x(x)$ and for each $x_i \in \{x_1, \dots, x_n\}$, we have $v_x(x_i) = v'_x(x_i)$. Hence $\mathcal{T}[\![\phi]\!]_{v_x} = \mathcal{T}[\![\phi]\!]_{v'_x}$. QED ■

Substitutions

Definition 19.2

- A **substitution** θ is a (total) function $\theta : V \rightarrow T(\Sigma)$ which is almost everywhere the identity.
- $S_{\Omega_1}(V)$ is the set of all substitutions.
- θ is called a **ground substitution** if $FV(\theta(x)) = \emptyset$.
- The **domain** of a substitution θ is the finite set $dom(\theta) = \{x \mid x \not\equiv \theta(x)\}$. θ acts on the variables in $dom(\theta)$.

Notes and notation.

- Equivalently a substitution θ may be represented as a *finite* (possibly empty) set $\theta = \{s/x \mid \theta(x) = s \not\equiv x\}$ containing only the non-identical elements and their images under θ .

Instantiation of Terms

Definition 19.3 Let θ be a substitution.

- The application of θ to a term $t \in T(\Sigma)$ is denoted θt and defined inductively as follows.

$$\theta y = y, \quad y \notin \text{dom}(\theta)$$

$$\theta x = \theta(x), \quad x \in \text{dom}(\theta)$$

$$\theta f(t_1, \dots, t_m) = f(\theta t_1, \dots, \theta t_m), \quad f : s^m \rightarrow s \in \Sigma$$

- θt is called a (substitution) instance of t .

The Substitution Lemma for Terms

Lemma 19.4 Given a Σ -interpretation (\mathbf{A}, v) , a term t and a substitution $\{s/x\}$, let $\mathcal{V}[\![s]\!]_v = a \in |\mathbf{A}|$. Then

$$\mathcal{V}[\!\{s/x\}t]\!]_v = \mathcal{V}[\!t]\!]_{v[x:=a]}$$



Proof of The Substitution Lemma for Terms 19.4

Proof: By induction on the structure of t .

Case $t \equiv x$. Then $\{s/x\}x \equiv s$ and we have $\mathcal{V}[\{s/x\}t]_v = \mathcal{V}[s]_v = a = \mathcal{V}[t]_{v[x:=a]}$.

Case $t \equiv y \not\equiv x$. Then $\{s/x\}y \equiv y$ and $\mathcal{V}[\{s/x\}t]_v = \mathcal{V}[y]_v = \mathcal{V}[t]_{v[x:=a]}$ since $v(y) = v[x := a](y)$.

Case $t \equiv f(t_1, \dots, t_m)$. Then

$$\begin{aligned} & \mathcal{V}[\{s/x\}t]_v \\ &= \mathcal{V}[\{s/x\}f(t_1, \dots, t_m)]_v \\ &= \mathcal{V}[f(\{s/x\}t_1, \dots, \{s/x\}t_m)]_v \\ &= f_A(\mathcal{V}[\{s/x\}t_1]_v, \dots, \mathcal{V}[\{s/x\}t_m]_v) \\ &= f_A(\mathcal{V}[t_1]_{v[x:=a]}, \dots, \mathcal{V}[t_m]_{v[x:=a]}) \quad \text{By the induction hypothesis} \\ &= \mathcal{V}[f(t_1, \dots, t_m)]_{v[x:=a]} \end{aligned}$$

QED



Admissibility

The occurrence of bound variables in formulae requires careful handling when substitutions are applied to formulae. Intuitively, an element $s/x \in \theta$ is **admissible** in a formula ϕ if the variables of s remain free after instantiating the formula.

Definition 19.5 Let θ be a substitution

- An element $s/x \in \theta$ is **admissible** in
 - $p(t_1, \dots, t_n)$,
 - $\neg\phi$ if it is admissible in ϕ ,
 - $(\phi \odot \psi)$ if it is admissible in both ϕ and ψ
 - $\partial x[\phi]$,
 - $\partial y[\phi]$ if $x \not\equiv y$, $y \notin FV(s)$ and s/x is admissible in ϕ .
- θ is **admissible** in ϕ if every element of θ is admissible in ϕ .

Instantiations of Formulae

Definition 19.6 Let θ be a substitution.

- The application of θ to a formula $\phi \in \mathcal{P}_1(\Sigma)$ is denoted $\theta\phi$ and defined inductively as follows.

$$\theta p(t_1, \dots, t_n) = p(\theta t_1, \dots, \theta t_n), \quad p : s^n \in \Sigma$$

$$\theta \neg \psi = \neg(\theta \psi),$$

$$\theta(\psi \odot \chi) = (\theta \psi \odot \theta \chi), \quad \odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

$$\theta \mathcal{O}x[\psi] = \mathcal{O}x[\theta' \psi], \quad \theta' = \theta - \{\theta(x)/x\}, \quad \mathcal{O} \in \{\forall, \exists\}$$

The Substitution Lemma for Formulae

Lemma 19.7 *Given a Σ -interpretation (\mathbf{A}, v) , a formula ϕ and an admissible substitution $\{s/x\}$, let $\mathcal{V}[s]_v = a \in |\mathbf{A}|$. Then $\mathcal{T}[\{s/x\}\phi]_v = \mathcal{T}[\phi]_{v[x:=a]}$.*



Proof of The Substitution Lemma for Formulae 19.7

Proof: Case $x \notin FV(\phi)$. Then it trivially follows that $\{s/x\}\phi \equiv \phi$ and $\mathcal{T}[\{\mathbf{s}/x\}\phi]_v = \mathcal{T}[\phi]_{v_x}$ for all x -variants of v .

Case $x \in FV(\phi)$. We proceed by induction on the structure of ϕ .

Sub-case $\phi \equiv p(t_1, \dots, t_n)$. We have

$$\begin{aligned} & \mathcal{T}[\{\mathbf{s}/x\}\phi]_v \\ = & \mathcal{T}[\{\mathbf{s}/x\}p(t_1, \dots, t_n)]_v \\ = & \mathcal{T}[p(\{s/x\}t_1, \dots, \{s/x\}t_n)]_v \\ = & \mathcal{T}[p(t_1, \dots, t_n)]_{v[x:=a]} \quad \text{By lemma 19.4} \end{aligned}$$

The sub-cases involving the propositional connectives are trivial and the only interesting cases left are those of quantified formulae.

Sub-case $\phi \equiv \exists y[\psi]$. Since $x \in FV(\phi)$ clearly $x \neq y$ and hence $x \in FV(\psi)$. Further since $\{s/x\}$ is admissible in ϕ , $y \notin Var(s)$. This implies that

1. $\{s/x\}$ is admissible in ψ ,
2. $\{s/x\}\phi \equiv \exists y[\{s/x\}\psi]$ and

3. $a = \mathcal{V}[\![s]\!]_v = \mathcal{V}[\![s]\!]_{v[y:=b]}$ for arbitrary $b \in |\mathbf{A}|$ since $y \notin Var(s)$.

By the induction hypothesis for any $b \in |\mathbf{A}|$ we have

$$\mathcal{T}[\![\{s/x\}\psi]\!]_{v[y:=b]} = \mathcal{T}[\![\psi]\!]_{v[y:=b][x:=a]} = \mathcal{T}[\![\psi]\!]_{v[x:=a][y:=b]}$$

from which we get

$$\{\mathcal{T}[\![\{s/x\}\psi]\!]_{v_y} \mid v_y =_{\backslash y} v\} = \{\mathcal{T}[\![\psi]\!]_{v[x:=a]_y} \mid v[x := a]_y =_{\backslash y} v[x := a]\}$$

which implies for any quantifier $\mathcal{T}[\![\{s/x\}\phi]\!]_v = \mathcal{T}[\![\phi]\!]_{v[x:=a]}$ regardless of the quantifier involved.

QED



Exercise 19.1

1. Let (\mathbf{A}_1, v_1) be a Σ_1 -interpretation and (\mathbf{A}_2, v_2) a Σ_2 -interpretation such that $|\mathbf{A}_1| = |\mathbf{A}_2|$. Let $\Sigma = \Sigma_1 \cap \Sigma_2$.

(a) Let t be a Σ -term. Prove that if (\mathbf{A}_1, v_1) and (\mathbf{A}_2, v_2) agree on the symbols occurring in t , then $\mathcal{V}_{\mathbf{A}_1}[\![t]\!]_{v_1} = \mathcal{V}_{\mathbf{A}_2}[\![t]\!]_{v_2}$.

(b) Let ϕ be a Σ -formula. Prove that (\mathbf{A}_1, v_1) and (\mathbf{A}_2, v_2) agree on the symbols occurring in ϕ , then $\mathcal{T}_{\mathbf{A}_1}[\![\phi]\!]_{v_1} = \mathcal{T}_{\mathbf{A}_2}[\![\phi]\!]_{v_2}$.

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

451 OF 783

[QUIT](#)

Lecture 20: Models, Satisfiability and Validity

Tuesday 20 September 2011

1. Satisfiability
2. Models and Consistency
3. Examples of Models:1
4. Examples of Models:2
5. Examples of Models:3
6. Logical Consequence
7. Validity
8. Validity of Sets of Formulae
9. Negations of Semantical Concepts

Satisfiability

Definition 20.1 (Satisfaction).

- A Σ -interpretation (\mathbf{A}, v) satisfies a Σ -formula ϕ denoted $(\mathbf{A}, v) \Vdash \phi$ if and only if $\mathcal{T}[\![\phi]\!]_v = 1$.
- ϕ is said to be satisfiable in \mathbf{A} if there is a valuation v such that $(\mathbf{A}, v) \Vdash \phi$.
- A Σ -formula ϕ is satisfiable if there exists a Σ -interpretation that satisfies it.

Models and Consistency

Definition 20.2 (Models).

- A Σ -structure \mathbf{A} is a **model** (or more accurately a Σ -**model**) of a Σ -formula $\phi \in \mathcal{P}_1(\Sigma)$ (denoted $\mathbf{A} \Vdash \phi$) if and only if for all valuations v , $(\mathbf{A}, v) \Vdash \phi$.
- For a nonempty set Φ of Σ -formulas,
 - a Σ -structure \mathbf{A} is a Σ -**model of Φ** , (denoted $\mathbf{A} \Vdash \Phi$) if and only if it is a model of every formula in Φ .
 - Φ is said to be an **axiom system** for all models of Φ .
 - Φ is said to be **consistent** iff it has a Σ -**model**.

Examples of Models:1

Example 20.3 Let $\Sigma = \{0 : \rightarrow s, +1 : s \rightarrow s, = : s^2, < : s^2\}$, let

$$\mathbf{N} = \langle \mathbb{N}; 0, +1; =, < \rangle$$

be the Σ -structure where \mathbb{N} is the set of naturals, $+1$ is the unary successor function, $=$ is the atomic binary equality predicate and $<$ is the binary “less-than” predicate. Let

$$\phi_1 \stackrel{df}{=} \neg(+1(x) = 0)$$

$$\phi_2 \stackrel{df}{=} (+1(x) = +1(y)) \rightarrow (x = y)$$

$$\phi_3 \stackrel{df}{=} \forall x \exists y [y = +1(x)]$$

$$\phi_4 \stackrel{df}{=} \forall x [\neg(x = 0) \rightarrow \exists y [x = +1(y)]]$$

We have $\mathbf{N} \Vdash \Phi$ where $\Phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$

Examples of Models:2

Example 20.4 Let $\Sigma = \{0 : \rightarrow s, + : s^2 \rightarrow s, = : s^2\}$ and let \mathbf{Z} be the Σ -structure

$$\mathbf{Z} = \langle \mathbb{Z}; 0, +; = \rangle$$

where \mathbb{Z} is the set of integers, 0 and $+$ represent the integer zero and the binary addition operation respectively, and $=$ is the atomic binary equality predicate. \mathbf{Z} is a model of the following set Φ of formulae

$$\phi_{associative} \stackrel{df}{=} \forall x, y, z [(x + y) + z = x + (y + z)] \quad (10)$$

$$\phi_{identity} \stackrel{df}{=} \forall x [x + 0 = x] \quad (11)$$

$$\phi_{right-inverse} \stackrel{df}{=} \forall x \exists y [x + y = 0] \quad (12)$$

Examples of Models:3

Example 20.5 The set Φ defined in the previous example is the set of axioms which defines the notion of a group in algebra. The addition of an extra axiom

$$\phi_{\text{commutative}} \stackrel{df}{=} \forall x, y [x + y = y + x] \quad (13)$$

i.e. $\Phi' = \Phi \cup \{\phi_{\text{commutative}}\}$ excludes all non-commutative groups from the models of the set Φ' .

Logical Consequence

Definition 20.6 A Σ -formula ψ is a logical consequence of a set Φ of Σ -formulae, denoted $\Phi \models \psi$, if and only if every model \mathbf{A} of Φ is also a model of ψ . If Φ is empty then ψ is said to be logically valid (denoted $\models \psi$).

Example 20.7 Let Σ be the signature in example 20.4 and let the formula $\phi_{\text{left-inverse}} \stackrel{\text{df}}{=} \forall x \exists y [y + x = 0]$. A typical proof in a mathematics text that $\phi_{\text{left-inverse}}$ is a logical consequence of the axioms of group theory (example 20.4) might go as follows.

Proof: Let x be any element. Then by axiom $\phi_{right-inverse}$ there exists y such that

$$x + y = 0 \quad (14)$$

Again by $\phi_{right-inverse}$ for some z we get

$$y + z = 0 \quad (15)$$

We then have

$$\begin{aligned} y + x &= (y + x) + 0 && (\phi_{identity}) \\ &= (y + x) + (y + z) && (15) \\ &= y + (x + (y + z)) && (\phi_{associative}) \\ &= y + ((x + y) + z) && (\phi_{associative}) \\ &= y + (0 + z) && (14) \\ &= (y + 0) + z && (\phi_{associative}) \\ &= y + z && (\phi_{identity}) \\ &= 0 && (15) \end{aligned}$$

QED



Effectively from the group axioms we have extracted fresh knowledge about groups in general namely that, the existence of a right inverse for each element of the group implies the existence of a left-inverse. In fact, by replacing the opening line of the proof by

Let x and y be elements such that

we could claim that the following formula

$$\phi_{\text{left-right-inverse}} \stackrel{df}{=} \forall x, y [(x + y = 0) \rightarrow (y + x = 0)]$$

is also a logical consequence of the group axioms.

Validity

Definition 20.8 (Validity). Let \mathbf{A} be a Σ -structure, ϕ and ψ be Σ -formulae.

- $(\mathbf{A} \Vdash \phi)$. ϕ is valid in \mathbf{A} if and only if \mathbf{A} is a model of ϕ .
- $(\mathfrak{A} \Vdash \phi)$. ϕ is valid in a class \mathfrak{A} of Σ -structures if it is valid in each structure $\mathbf{A} \in \mathfrak{A}$.
- $(\Vdash \phi)$. ϕ is (logically) valid if and only if every Σ -structure is a model of ϕ .
- $(\phi \Leftrightarrow \psi)$. ϕ is (logically) equivalent to ψ if $\Vdash \phi \leftrightarrow \psi$.

Validity of Sets of Formulae

Definition 20.9 (Validity of a set of formulae). *Let Φ be a set of Σ -formulae.*

- ($\mathbf{A} \Vdash \Phi$). Φ is valid in \mathbf{A} if and only if \mathbf{A} is a model of each $\phi \in \Phi$.
- ($\mathfrak{A} \Vdash \Phi$). Φ is valid in a class \mathfrak{A} of Σ -structures if $\mathbf{A} \Vdash \Phi$ for each $\mathbf{A} \in \mathfrak{A}$.
- ($\Vdash \Phi$). Φ is valid if and only if every Σ -structure is a model of each $\phi \in \Phi$.

We have deviated in our meta-logical notation from standard textbooks. Notice from the definitions of **logical consequence**, **logical validity** and **validity of sets of formulae** that $\Phi \models \psi$ if and only if for every structure Σ -structure \mathbf{A} , $\mathbf{A} \Vdash \Phi$ implies $\mathbf{A} \Vdash \psi$. Most textbooks on first-order logic actually use only one overloaded symbol \models for both concepts. However we have decided to keep them separate since the concept of logical consequence involves sets of **formulae** of a formal language whereas the other refers specifically to **models**.

Notice also that for any formula ϕ , both $\Vdash \phi$ and $\models \phi$ denote that ϕ is **(logically) valid**. By extension, therefore logical equivalence defined as $\Vdash \phi \leftrightarrow \psi$ may equally well be defined as $\models \phi \leftrightarrow \psi$.

Negations of Semantical Concepts

- We use the symbols \nVdash , $\not\models$, $\not\Leftarrow$ to denote the negations of the corresponding relations.

Exercise 20.1

1. Prove that \mathbf{A} is a model of ϕ iff \mathbf{A} is a model of $\vec{\forall}[\phi]$.
2. Prove that $\Phi \models \psi$ iff $\vec{\forall}[\Phi] \models \vec{\forall}[\phi]$, where $\vec{\forall}[\Phi]$ denotes the universal closure of each formula in Φ .
3. Prove that $\Vdash \phi$ if and only if $\models \phi$ (i.e. $\emptyset \models \phi$). Hence in most books on logic, the same symbol \models is used for both logical consequence and for validity in models).
4. Prove that $\Vdash \phi$ if and only if $\Vdash \vec{\forall}[\phi]$.
5. Prove that ϕ is satisfiable in \mathbf{A} if and only if $\vec{\exists}[\phi]$ is satisfiable in \mathbf{A} .
6. Show that $\phi \vee \neg\phi$ is valid for any formula ϕ .
7. In general every first-order logic formula which has a tautological “shape” in propositional logic is a valid formula. Formalize this notion and prove it.
8. Prove that \forall and \exists are “duals” i.e. show that
 - (a) $\Vdash \forall x[\phi] \leftrightarrow \neg\exists x[\neg\phi]$

(b) $\Vdash \exists x[\phi] \leftrightarrow \neg \forall x[\neg \phi]$

9. Let $\mathcal{O}x[\psi] \in SF(\phi)$ and $y \notin FV(\psi)$. Let ϕ' be obtained from ϕ by replacing $\mathcal{O}x[\psi]$ by $\mathcal{O}y[\{y/x\}\psi]$. Then ϕ and ϕ' are said to be α -equivalent and denoted by $\phi \equiv_\alpha \phi'$. Prove that two α -equivalent formulae are equivalent.

10. If $x \notin FV(\psi)$ then $\mathcal{O}x[\psi]$ is equivalent to ψ .

11. Give examples of interpretations (A, v) to show that the following formulae are not valid.

(a) $\exists x[\psi] \rightarrow \forall x[\psi]$

(b) $\forall x \exists y[\psi(x, y)] \rightarrow \exists y \forall x[\psi(x, y)]$

12. Prove that for any binary predicate ψ ,

$$\forall x, y[\psi(x, y) \rightarrow \psi(y, x)] \Leftrightarrow \forall x, y[\psi(x, y) \leftrightarrow \psi(y, x)]$$

13. Prove that the following formulae are valid.

(a) $\exists x[\psi(x) \rightarrow \forall x[\psi(x)]]$.

(b) $\exists x \forall y[\psi(x, y)] \rightarrow \forall y \exists x[\psi(x, y)]$

14. Prove that $\phi \Leftrightarrow \psi$ iff $\phi \models \psi$ and $\psi \models \phi$.

15. A student claimed that the following definition is a stronger definition of logical consequence than definition 20.6. Justify or refute his claim.

Definition 20.10 A Σ -formula ψ is a **logical consequence'** of a set Φ of Σ -formulae, denoted $\Phi \models' \psi$, if and only if for every interpretation (\mathbf{A}, v) , $(\mathbf{A}, v) \Vdash \phi$ for each $\phi \in \Phi$ implies $(\mathbf{A}, v) \Vdash \psi$.

20.1. Some Model Theory

When we consider the notion of a model, we come across various notions which express properties of the models. For example, if $<$ denotes an irreflexive ordering relation, then a sentence such as $\forall x \exists y [x < y]$ specifies that there is no greatest element in the ordering. Such a sentence therefore has no finite models. Its negation however has only finite models. Another sentence such as $\forall x \forall y [x = y]$ in the language of first-order logic with equality has only singleton models.

If all models of a set Φ of sentences are finite then there must be a *finite bound on the size of the models* (i.e. the carrier set of the structure must be of a fixed finite cardinality). Otherwise, as the following theorem shows there would also be infinite models. For instance as Enderton [3] states:

It is a priori conceivable that there might be some very subtle equation of group theory that was true in every finite group but false in every infinite group.

But theorem 20.11 assures us that such a possibility does not exist.

Theorem 20.11 *If a set Φ of Σ -formulae has arbitrarily large finite models, then it has an infinite model.*

Proof: Assume Φ has arbitrarily large finite models. Now consider the following formulae ψ_k for each $k \geq 2$,

$$\psi_2 \stackrel{df}{=} \exists x_1, x_2 [\neg(x_1 = x_2)]$$

$$\psi_3 \stackrel{df}{=} \exists x_1, x_2, x_3 [\neg(x_1 = x_2) \wedge \neg(x_1 = x_3) \wedge \neg(x_2 = x_3)]$$

⋮ ⋮ ⋮

where each ψ_k states that there are at least k distinct elements in the model. Now consider the set $\Psi = \Phi \cup \{\psi_k \mid k \geq 2\}$. By hypothesis, every finite subset of Ψ has a model. By the compactness theorem therefore Ψ also has a model. However clearly no model of Ψ can be finite. Hence Ψ must have an infinite model. This infinite model is also a model of Φ since $\Phi \subset \Psi$. QED ■

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

471 OF 783

QUIT

Lecture 21: Structures and Substructures

Wednesday 21 September 2011

1. Satisfiability and Expansions
2. Distinguishability
3. Evaluations under Different Structures
4. Isomorphic Structures
5. The Isomorphism Lemma
6. Substructures
7. Substructure Examples
8. Quantifier-free Formulae
9. Lemma on Quantifier-free Formulae
10. Universal and Existential Formulae
11. The Substructure Lemma

Satisfiability and Expansions

Our notions of satisfiability, consequence and validity are all with respect to a specific signature.

Lemma 21.1 *Let $\Sigma_1 \subseteq \Sigma_2$. For any set Φ of Σ_1 -formulae, Φ is satisfiable with respect to Σ_1 iff Φ is satisfiable with respect to Σ_2 .*

Proof: Essentially the interpretations of the common symbols should coincide, whereas the interpretations of the symbols in $\Sigma_2 - \Sigma_1$ may be arbitrary. It then follows from problem 1 of exercise 19.1. QED ■

Distinguishability

Example 21.2 For $\Sigma = \{= : s^2, < : s^2\}$ consider the Σ -structures $\mathbf{Z} = \langle \mathbb{Z}; ; =, < \rangle$ and $\mathbf{Q} = \langle \mathbb{Q}; ; =, < \rangle$ where \mathbb{Z} is the set of integers, \mathbb{Q} is the set of rational numbers, $<$ are respectively the “less-than” relations on the two sets respectively. Now consider the formula

$$\phi_{density} \stackrel{df}{=} \forall x, z [x < z \rightarrow \exists y [(x < y) \wedge (y < z)]]$$

Clearly $\mathbf{Q} \Vdash \phi_{density}$ whereas $\mathbf{Z} \not\Vdash \phi_{density}$.

$\phi_{density}$ distinguishes the two structures.

Evaluations under Different Structures

When evaluating terms and formulae under different structures say \mathbf{A} and \mathbf{B} using valuations $v_{\mathbf{A}} : V \rightarrow A$ and $v_{\mathbf{B}} : V \rightarrow B$ where $A = |\mathbf{A}|$ and $B = |\mathbf{B}|$ we use $\mathcal{V}_{\mathbf{A}}$, $\mathcal{T}_{\mathbf{A}}$ and $\mathcal{V}_{\mathbf{B}}$, $\mathcal{T}_{\mathbf{B}}$ respectively to distinguish the possibly different values and truth values.

Isomorphic Structures

Example 21.3 Let $\Sigma = \{0 : \rightarrow s, +1 : s \rightarrow s, = : s^2, < : s^2\}$.

Now consider the structures

$$\mathbb{N} = \langle \mathbb{N}; 0, +1; =, < \rangle$$

$$2\mathbb{N} = \langle 2\mathbb{N}; 0, +2; =, < \rangle$$

where $2\mathbb{N}$ is the set of even natural numbers, $+1, +2$ denote the respective “successor” functions, $<$ is the usual “less-than” relation on both structures. The two structures are clearly isomorphic and there is an isomorphism which $\pi : \mathbb{N} \rightarrow 2\mathbb{N}$ such that $\pi(n) = 2n$ which along with the inverse map $\pi^{-1}(2n) = n$ maps one structure exactly onto the other.

Isomorphic Σ -structures cannot be distinguished by $\mathcal{P}_1(\Sigma)$.

The Isomorphism Lemma

Lemma 21.4 (The Isomorphism Lemma). *If \mathbf{A} and \mathbf{B} are isomorphic Σ -structures then for all formulae ϕ , $\mathbf{A} \Vdash \phi$ if and only if $\mathbf{B} \Vdash \phi$.*



Corollary 21.5 *If $\pi : \mathbf{A} \cong \mathbf{B}$ then for any formula $\phi(x_1, \dots, x_n)$ and any valuation $v_{\mathbf{A}}, v_{\mathbf{B}}$ and values $a_1, \dots, a_n \in |\mathbf{A}|$,*

$$(\mathbf{A}, v_{\mathbf{A}}[x_1 := a_1, \dots, x_n := a_n]) \Vdash \phi \\ \text{iff}$$

$$(\mathbf{B}, v_{\mathbf{B}}[x_1 := \pi(a_1), \dots, x_n := \pi(a_n)]) \Vdash \phi$$



Proof of the Isomorphism Lemma 21.4

Proof: Assume there is an isomorphism $\pi : \mathbf{A} \cong \mathbf{B}$. Since π is an isomorphism, $\pi : A \xrightarrow[\text{onto}]^{1-1} B$ is a bijection that preserves structure, where $A = |\mathbf{A}|$ and $B = |\mathbf{B}|$. Hence $\pi^{-1} : B \xrightarrow[\text{onto}]^{1-1} A$ also exists and preserves structure. Further we have

1. $\pi(f_{\mathbf{A}}(a_1, \dots, a_m)) = f_{\mathbf{B}}(\pi(a_1), \dots, \pi(a_m))$ for every m -ary function $f_{\mathbf{A}}$,
2. $\pi^{-1}(f_{\mathbf{B}}(b_1, \dots, b_m)) = f_{\mathbf{A}}(\pi^{-1}(b_1), \dots, \pi^{-1}(b_m))$ for every m -ary function $f_{\mathbf{B}}$,
3. $(a_1, \dots, a_n) \in p_{\mathbf{A}}$ iff $(\pi(a_1), \dots, \pi(a_n)) \in p_{\mathbf{B}}$ for each n -ary relation $p_{\mathbf{A}}$.
4. $(b_1, \dots, b_n) \in p_{\mathbf{B}}$ iff $(\pi^{-1}(b_1), \dots, \pi^{-1}(b_n)) \in p_{\mathbf{A}}$ for each n -ary relation $p_{\mathbf{B}}$.

Further for each $v_{\mathbf{A}} : V \longrightarrow A$, $\pi \circ v_{\mathbf{A}} : V \longrightarrow B$ and for each $v_{\mathbf{B}} : V \longrightarrow B$, $\pi^{-1} \circ v_{\mathbf{B}} : V \longrightarrow A$.

For every formula ϕ , we may prove the stronger claims,

$$\begin{aligned}\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v_{\mathbf{A}}} &= \mathcal{T}_{\mathbf{B}}[\![\phi]\!]_{\pi \circ v_{\mathbf{A}}} \\ \mathcal{T}_{\mathbf{B}}[\![\phi]\!]_{v_{\mathbf{B}}} &= \mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{\pi^{-1} \circ v_{\mathbf{B}}}\end{aligned}$$

The proof is by induction on the structure of ϕ and is left as an exercise to the interested reader.
QED

The Isomorphism Lemma then raises the following interesting question which we will answer later.

Question. Are Σ -structures which satisfy the same Σ -formulae isomorphic?

But regardless of the answer to the above question the isomorphism lemma also brings the realization that for any given set of Σ -formulae there could be more than one model, in fact a class of models, for a given nonempty set Φ of Σ -formulae. Our notions of validity may therefore include the following besides those given in definition 20.8.

Substructures

Definition 21.6 A Σ -structure \mathbf{A} is a **substructure** of another Σ -structure \mathbf{B} (denoted $\mathbf{A} \subseteq \mathbf{B}$) if

- $\emptyset \neq |\mathbf{A}| \subseteq |\mathbf{B}|$,
- for each $f : s^m \rightarrow s \in \Sigma$, $f_{\mathbf{A}} = f_{\mathbf{B}} \upharpoonright |\mathbf{A}|^m$,
- for each $p : s^n \in \Sigma$, $p_{\mathbf{A}} = p_{\mathbf{B}} \cap |\mathbf{A}|^n$.

where $f_{\mathbf{B}} \upharpoonright |\mathbf{A}|^m$ denotes the restriction of $f_{\mathbf{B}}$ to elements of $|\mathbf{A}|^m$.

Facts 21.7 If $\mathbf{A} \subseteq \mathbf{B}$, then

1. $|\mathbf{A}|$ is Σ -closed, i.e. for each $f : s^m \rightarrow s \in \Sigma$ and each $(a_1, \dots, a_m) \in |\mathbf{A}|^m$, $f_{\mathbf{B}}(a_1, \dots, a_m) \in |\mathbf{A}|$.
2. Conversely for each $X \subseteq |\mathbf{B}|$, such that X is Σ -closed there exists a unique Σ -substructure $\mathbf{X} \subseteq \mathbf{B}$.

Substructure Examples

Example 21.8

1. $\mathbf{N} = \langle \mathbb{N}; 0, +; = \rangle$ is a substructure of $\mathbf{Z} = \langle \mathbb{Z}; 0, +; = \rangle$ which in turn is a substructure of $\mathbf{Q} = \langle \mathbb{Q}; 0, +; = \rangle$ which in turn is a substructure of $\mathbf{R} = \langle \mathbb{R}; 0, +; = \rangle$.
2. $2\mathbf{N} = \langle 2\mathbb{N}; 0, +; = \rangle$ is a substructure of $\mathbf{N} = \langle \mathbb{N}; 0, +; = \rangle$.
3. However the odd numbers are not closed under addition and hence they do not form a substructure of \mathbf{N} under addition.

Quantifier-free Formulae

Definition 21.9 For any signature Σ , the set $\mathcal{QF}_1(\Sigma)$ of quantifier-free formulae is given by the following grammar

$$\begin{array}{ccl} \chi, \xi ::= & \perp & \mid \top \\ & \mid p(t_1, \dots, t_n) \in A(\Sigma) & \mid (\neg \chi) \\ & \mid (\chi \wedge \xi) & \mid (\chi \vee \xi) \\ & \mid (\chi \rightarrow \xi) & \mid (\chi \leftrightarrow \xi) \end{array}$$

Lemma on Quantifier-free Formulae

Lemma 21.10 Let $\mathbf{A} \subseteq \mathbf{B}$ be Σ -structures and let $v_{\mathbf{A}}$ be any valuation in \mathbf{A} . Then for every Σ -term t and every quantifier-free formula χ

$$\mathcal{V}_{\mathbf{A}}[t]_{v_{\mathbf{A}}} = \mathcal{V}_{\mathbf{B}}[t]_{v_{\mathbf{A}}}$$

$$\mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = \mathcal{T}_{\mathbf{B}}[\chi]_{v_{\mathbf{A}}}$$

Proof: Use the Isomorphism lemma 21.4 on the common subset of the two carrier sets using the identity isomorphism.
QED



Universal and Existential Formulae

Definition 21.11 For any signature Σ , the set $\mathcal{O}_1(\Sigma)$ of \mathcal{O} -formulae for each $\mathcal{O} \in \{\forall, \exists\}$ is defined inductively by the following grammar

$$\begin{aligned}\phi, \psi ::= & \chi \in \mathcal{QF}_1(\Sigma) \\ & | \quad (\phi \wedge \psi) \quad | \quad (\phi \vee \psi) \\ & | \quad \mathcal{O}x[\phi]\end{aligned}$$

$\forall_1(\Sigma)$ is the set of universal Σ -formulae and $\exists_1(\Sigma)$ is the set of existential Σ -formulae.

The Substructure Lemma

Lemma 21.12 (The Substructure Lemma). *Let $\mathbf{A} \subseteq \mathbf{B}$ be Σ -structures and let $\phi(x_1, \dots, x_n) \in \forall_1(\Sigma)$. Then for any valuations $v_{\mathbf{A}}, v_{\mathbf{B}}$ and $a_1, \dots, a_n \in A$,*

if $(\mathbf{B}, v_{\mathbf{B}}[x_1 := a_1, \dots, x_n := a_n]) \Vdash \phi$
then $(\mathbf{A}, v_{\mathbf{A}}[x_1 := a_1, \dots, x_n := a_n]) \Vdash \phi$

Proof: By induction on the structure of universal formulae.
QED ■

Exercise 21.1

1. **Theorem 21.13 (The Homomorphism Theorem).** Let $\varpi : A \longrightarrow B$ be a homomorphism from a structure \mathbf{A} into a structure \mathbf{B} and let $v_{\mathbf{A}}$ be a valuation.
 - (a) For any term t , $\mathcal{V}_{\mathbf{B}}[\![t]\!]_{\varpi \circ v_{\mathbf{A}}} = \varpi(\mathcal{V}_{\mathbf{A}}[\![t]\!]_{v_{\mathbf{A}}})$
 - (b) If ϖ is injective then for any quantifier-free formula χ , $\mathcal{T}_{\mathbf{B}}[\![\chi]\!]_{\varpi \circ v_{\mathbf{A}}} = \varpi(\mathcal{T}_{\mathbf{A}}[\![\chi]\!]_{v_{\mathbf{A}}})$

Prove theorem 21.13.
2. Prove that if ϖ is not surjective, the two structures may be distinguished by a formula.
3. In part 1b of theorem 21.13 why is the condition of injectivity necessary? (Hint. Let $=$ be the atomic binary equality predicate whose semantics is defined in an obvious fashion. Now show that if ϖ is not injective then the two structures can be distinguished using equality.)
4. Prove that if ϖ is surjective but not necessarily injective, then in the absence of any predicate from which equality or inequality of terms may be expressed or derived, part 1b of theorem 21.13 may be extended to quantified formulae.

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

488 OF 783

[QUIT](#)

Lecture 22: Predicate Logic: Proof Theory

Friday 23 September 2011

1. Proof Theory: First-Order Logic
2. Proof Rules: Hilbert-Style
3. The Mortality of Socrates
4. The Mortality of the Greeks
5. Faulty Proof:2
6. A Correct Proof
7. The Sequent Forms
8. The Case of Equality
9. Semantics of Equality
10. Axioms for Equality
11. Symmetry and Transitivity
12. Symmetry of Equality
13. Transitivity of Equality

Proof Theory: First-Order Logic

1. A first-order proof system for a mathematical theory with signature Σ consists of two parts

Logical Axioms and Rules These are extensions of existing propositional proof systems to deal with quantification. However, they are still parameterised on Σ since the use of terms and substitution is involved in them.

Non-logical axioms These are the axioms which define the models to which they are applicable.

2. **(First-order) Predicate Calculus:** The first-order theory with no non-logical axioms.

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

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

Admissibility of Substitutions

Some explanations are perhaps in order regarding the “side conditions” attached to the axioms above.

$\forall E$. The side condition “ $t \equiv x$ ” allows the substituted term to be the variable x , which is not surprising. But this is explicitly mentioned only because the side condition “ $\{t/x\}$ admissible in X ” does not include the possibility of replacing x by itself.

But more important is the side condition “ $\{t/x\}$ admissible in X ”. If this condition were not present then we could have the following situation which would be clearly unsound.

Let $p(x, y)$ be a binary predicate and let $\phi(x) \stackrel{df}{=} \neg \forall y[p(x, y)]$.

Now clearly $\{y/x\}$ is not admissible in $\phi(x)$. If the condition of admissibility were absent then we could have replaced the free variable x with y yielding the following instance of axiom schema $\forall E$

$$\forall x[\neg \forall y[p(x, y)]] \rightarrow \neg \forall y[p(y, y)] \quad (16)$$

Now consider any model consisting of at least two distinct elements and let $p(x, y)$ denote the identity relation on the carrier set. Clearly in such a model while the antecedent $\forall x[\neg \forall y[p(x, y)]]$ is clearly valid (true for each valuation), the consequent $\neg \forall y[p(y, y)]$ would be invalid (false for

certain valuations).

- forall D. In this axiom schema if we were to relax the side-condition and allow the quantifier to be moved in even if $x \in FV(X)$ it would result in the following situation when $X = \phi(x) = Y$.

$$\forall x[\phi(x) \rightarrow \phi(x)] \rightarrow (\phi(x) \rightarrow \forall x[\phi(x)]) \quad (17)$$

Whereas the antecedent of the formula in (17) is logically valid, the truth of the consequent is not guaranteed in any interpretation in which $\phi(x)$ is true only for some elements of the domain and false for others.

- forall I. The hypothesis of the rule asserts that a formula $\phi(\dots, x, \dots)$ holds when the free variable x is replaced uniformly throughout the formula by any variable that does not occur free in ϕ . Then clearly the formula holds for any “arbitrary” value that x may take, and hence the variable x may be universally quantified.

The Mortality of Socrates

The Mortality of Socrates

A translation into predicate logic involves

- including two unary atomic predicates h and m denoting properties of objects (*human* and *mortal* respectively),
- including a constant s (for *Socrates*).

$$\forall x[h(x) \rightarrow m(x)], h(s) \vdash m(s)$$

$$\frac{\forall E \frac{\forall x[h(x) \rightarrow m(x)]}{h(s) \rightarrow m(s)}}{MP \frac{h(s) \rightarrow m(s)}{m(s)}}$$

The Mortality of the Greeks

All humans are mortal.

All Greeks are human.

Therefore all Greeks are mortal.

$$\forall x[h(x) \rightarrow m(x)], \forall x[g(x) \rightarrow h(x)] \vdash \forall x[g(x) \rightarrow m(x)]$$

A faulty proof:

$$T \Rightarrow \frac{\forall x[g(x) \rightarrow h(x)] \quad \forall x[h(x) \rightarrow m(x)]}{\forall x[g(x) \rightarrow m(x)]}$$

Faulty Proof:2

$$\frac{\frac{\forall E \frac{\forall x[g(x) \rightarrow h(x)]}{g(y) \rightarrow h(y)} \quad \forall E \frac{\forall x[h(x) \rightarrow m(x)]}{h(z) \rightarrow m(z)}}{?? \frac{g(y) \rightarrow m(y)}{\forall I \frac{g(y) \rightarrow m(y)}{\forall x[g(x) \rightarrow m(x)]}}}$$

A Correct Proof

$$\begin{array}{c} \forall E \frac{\forall x[g(x) \rightarrow h(x)]}{g(y) \rightarrow h(y)} \quad \forall E \frac{\forall x[h(x) \rightarrow m(x)]}{h(y) \rightarrow m(y)} \\ T \Rightarrow \frac{}{\forall I \frac{g(y) \rightarrow m(y)}{\forall x[g(x) \rightarrow m(x)]}} \end{array}$$

The Sequent Forms

The sequent forms of K, S, N, MP are as before. The sequent forms of the quantification rules are as follows:

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

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

$$\forall I. \frac{\Gamma \vdash \{y/x\}X}{\Gamma \vdash \forall x[X]}, y \notin FV(X) \cup FV(\Gamma)$$

Note that the variable y being quantified should not occur free in any of the assumptions Γ .

The Case of Equality

1. In most algebraic formulations equality is a necessary binary predicate of the signature.
2. In other cases where equality may not otherwise play a prominent role, it becomes necessary because of one or more of the following reasons.
 - (a) *Syntactically distinct* terms may represent the same value in a structure either because of some axioms or because of some identifications made in a valuation i.e. it is possible that even though $s \not\equiv t$, $\mathcal{V}_A[s]_{v_A} = \mathcal{V}_A[t]_{v_A}$. (see also exercise 21.1).
 - (b) Two *differently named* entities are proven to be the same entity (generally in proofs of uniqueness or in proofs by contradiction).

Generally in mathematics, if there are two named entities x and y and nothing is stated about the relationship between them, we can neither assume they stand for distinct entities nor can we exclude the possibility that they both stand for the same entity.

For instance, it becomes essential when speaking of models which contain at least two elements, to make a first-order statement like $\exists x \exists y [\neg(x = y)]$ which states that there exist at least two distinct objects.

Example 22.2 Consider any first-order theory of boolean algebra. If we allow for the possibility that $1 = 0$ every equation of boolean algebra would still be satisfied and the boolean identities would still continue to hold. But in addition we would also have $0 \leq 1 \leq 0$. If we were to adopt this as a model for truth, then every formula would be implied by every other formula and the whole edifice of mathematical logic as we know it would collapse to a triviality without the assumption that the value 1 is distinct from the value 0.

On the other hand, it may so happen that one may define properties of objects and prove a theorem stating in effect that if the two objects x and y both satisfy a certain property ϕ , then they are both the same object. This is usually expressed as $(\phi(x) \wedge \phi(y)) \rightarrow (x = y)$.

Example 22.3 The definition of injective functions in any standard mathematics text uses equality

or inequality in its definition. A function $f : A \rightarrow B$ is injective if for any two distinct elements a, a' such that $a \neq a'$, $f(a) \neq f(a')$. Alternatively, f is injective if $f(a) = f(a')$ implies $a = a'$.

The use of equality or inequality in all such situations is inescapable. Hence equality has a special place in mathematics and logic and is usually required as a basic relation between named entities, even when it is not primarily a relation of interest in the models that are being studied.

Semantics of Equality

The semantics of the binary infix atomic predicate $=$ is defined as follows:

$$\mathcal{T}_A[s = t]_{v_A} \stackrel{df}{=} \begin{cases} 1 & \text{if } \mathcal{V}_A[s]_{v_A} = \mathcal{V}_A[t]_{v_A} \\ 0 & \text{otherwise} \end{cases}$$

In the sequel we do not explicitly include equality in the signature, but assume that it is present as part of the language of **First-order Predicate Logic with Equality**.

(First-order Predicate Calculus with Equality) The first-order theory with no non-logical axioms except the axioms for equality.

Axioms for Equality

Equality usually is a reflexive, symmetric, transitive and substitutive relation on structures. However the following axioms are sufficient.

$$= R. \frac{}{t = t}$$

$$= C. \frac{}{(s = t) \rightarrow (\{s/x\}u = \{t/x\}u)}$$

$$= S. \frac{}{(s = t) \rightarrow (\{s/x\}X \rightarrow \{t/x\}X)}, \{s/x\}, \{t/x\} \text{ admissible in } X$$

Explanations.

=R This is the **reflexivity** axiom schema, which asserts that any term equals itself.

=C This is the **congruence** axiom schema and it asserts that equals may be substituted for equals in any term context.

=S This is the **substitutivity** axiom schema which again asserts that the replacement of equals by equals does not alter the truth value of formulae. However due to the presence of quantifiers and bound variables one must ensure that the substitution of equals by equals does not result in the capture of free variables of either *s* or *t*.

Symmetry and Transitivity

The rule of substitutivity ($=S$) is sufficiently powerful to force the properties of symmetry and transitivity with the help of reflexivity and modus ponens.

$$= \text{Sym. } \frac{\Gamma \vdash s=t}{\Gamma \vdash t=s}$$

$$= \text{T. } \frac{\Gamma \vdash s=t \quad \Gamma \vdash t=u}{\Gamma \vdash s=u}$$

Symmetry of Equality

Proof of derived rule $=\text{Sym}$

Let $\Gamma = \{s = t\}$ and Let $\phi \stackrel{df}{=} x = s$. Then we have

$$\begin{aligned}\{s/x\}\phi &\equiv s = s \\ \{t/x\}\phi &\equiv t = s\end{aligned}$$

$$\text{MP} \frac{\Gamma \vdash s = t \quad =S \frac{}{\Gamma \vdash s = t \rightarrow (s = s \rightarrow t = s)}}{\text{MP} \frac{\Gamma \vdash s = s \rightarrow t = s}{\Gamma \vdash t = s}} =R \frac{}{\Gamma \vdash s = s}$$

Transitivity of Equality

Proof of derived rule =T

Let $\Delta = \{s = t, t = u\}$ and $\psi \equiv s = x$. Then

$$\frac{\text{MP} \frac{\text{=S } \overline{\Delta \vdash t = u \rightarrow (s = t \rightarrow s = u)} \quad \Delta \vdash t = u}{\Delta \vdash s = t \rightarrow s = u}}{\Delta \vdash s = u}$$

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

509 OF 783

[QUIT](#)

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

510 OF 783

[QUIT](#)

Lecture 23: Predicate Logic: Proof Theory (Contd.)

Tuesday 27 September 2011

1. Alpha Conversion
2. The Deduction Theorem for Predicate Calculus
3. Useful Corollaries
4. Soundness of Predicate Calculus
5. Soundness of The Hilbert System

Alpha Conversion

Notation We use “ $\phi \dashv\vdash \psi$ ” as an abbreviation for the two statements “ $\phi \vdash \psi$ ” and “ $\psi \vdash \phi$ ”.

Lemma 23.1 *For every formula ϕ for which $\{y/x\}$ is admissible*

$$\forall x[\phi] \dashv\vdash \forall y[\{y/x\}\phi]$$

Proof: If $\{y/x\}$ is admissible in ϕ then we may readily see that $\{x/y\}$ is also admissible in $\{y/x\}\phi$ since $x \notin FV(\{y/x\}\phi)$ and

$$\{x/y\}\{y/x\}\phi \equiv \phi$$

Further since $x \notin FV(\forall x[\phi])$ we have by axiom schema $\forall E$ $\forall x[\phi] \rightarrow \phi$ i.e.

$$\forall x[\phi] \rightarrow \{x/y\}\{y/x\}\phi$$

We then have the following **proofs**. QED



Proof trees of $\forall x[\phi] \dashv\vdash \forall y[\{y/x\}\phi]$

$$\text{MP} \frac{\overline{\forall x[\phi] \vdash \forall x[\phi]} \quad \forall E \frac{}{\forall x[\phi] \vdash \forall x[\phi] \rightarrow \{x/y\}\{y/x\}\phi}}{\forall I \frac{\forall x[\phi] \vdash \{x/y\}\{y/x\}\phi}{\forall x[\phi] \vdash \forall y[\{y/x\}\phi]}}$$

$$\text{MP} \frac{\overline{\forall y[\{y/x\}\phi] \vdash \forall y[\{y/x\}\phi]} \quad \forall E \frac{}{\forall y[\{y/x\}\phi] \vdash \forall y[\{y/x\}\phi] \rightarrow \{y/x\}\phi}}{\forall I \frac{\forall y[\{y/x\}\phi] \vdash \{y/x\}\phi}{\forall y[\{y/x\}\phi] \vdash \forall x[\phi]}}$$

The Deduction Theorem for Predicate Calculus

Theorem 23.2 (The Deduction Theorem for Predicate Calculus).

Let $\Gamma \subseteq_f \mathcal{L}_1$ and $\phi, \psi \in \mathcal{L}_1$.

1. (DT \Leftarrow). If $\Gamma \vdash \phi \rightarrow \psi$ then $\Gamma, \phi \vdash \psi$.
2. (DT \Rightarrow). Let $\Gamma, \phi \vdash \psi$. Then $\Gamma \vdash \phi \rightarrow \psi$ if no variable in $FV(\phi)$ is generalized (by an application of $\forall I$) in the proof.



Proof of the Deduction Theorem for Predicate Calculus (theorem 23.2)

Proof:

1. ($\text{DT} \Leftarrow$). The proof is identical to that of the corresponding **propositional case**.
2. ($\text{DT} \Rightarrow$). Assume $\Gamma, \phi \vdash \psi$. Then there exists a proof tree \mathcal{T} rooted at ψ with nodes $\psi_1, \dots, \psi_m \equiv \psi$. Then the stronger claim that each step of the proof of ψ_i can be matched by a proof of $\phi \rightarrow \psi_i$ is again proven here by induction on $k = \ell(\psi_1) - \ell(\psi_i)$. The proof proceeds in a manner similar to the corresponding **propositional case** for all applications of the propositional axioms and the inference rule (MP). We consider only the cases of quantification.

If ψ_j for $0 < j \leq m$ is an axiom (including an instance of $\forall E$ or $\forall D$) or $\psi_j \in \Gamma$, then the proof tree \mathcal{T}'_j rooted at $\phi \rightarrow \psi_j$ is constructed from the proof tree \mathcal{T}_j rooted at ψ_j as follows.

$$\frac{j' \frac{j' \frac{\wedge \mathcal{T}_j \wedge}{\psi_j}}{j' + 1 \frac{\psi_j \rightarrow (\phi \rightarrow \psi_j)}{\phi \rightarrow \psi_j}}}{j' + 2}$$

Suppose ψ_j was obtained by the application of the axiom schema $\forall I$ on some ψ_i such that $\ell(\psi_i) > \ell(\psi_j)$. Then $\psi_j \equiv \forall x[\psi_i]$. By the induction hypothesis there exists a proof tree \mathcal{T}'_i rooted at

$\phi \rightarrow \psi_i$ and such that no free variable of ϕ has been generalized in the application. Further $x \notin FV(\phi)$. We may now extend \mathcal{T}'_i to a proof tree \mathcal{T}'_j as follows.

$$i' + 3 \frac{i' \frac{i' \frac{\phi \rightarrow \psi_i}{\forall x[\phi \rightarrow \psi_i]} \nearrow \mathcal{T}'_i \nearrow}{\forall x[\phi \rightarrow \psi_i] \rightarrow (\phi \rightarrow \forall x[\psi_i])}}{\phi \rightarrow \forall x[\psi_i] \equiv \phi \rightarrow \psi_j}$$

QED



Useful Corollaries

Corollary 23.3 *If the proof of $\Gamma, \phi \vdash \psi$ involves no generalization of any free variable of ϕ then $\Gamma \vdash \phi \rightarrow \psi$.*

Corollary 23.4 *If ϕ is a closed formula and $\Gamma, \phi \vdash \psi$ then $\Gamma \vdash \phi \rightarrow \psi$.*

Corollary 23.5 *If no free variable of $\Gamma = \{\phi_1, \dots, \phi_m\}$ is generalized in a proof of $\Gamma \vdash \psi$, then $\vdash \phi_1 \rightarrow \dots \rightarrow \phi_m \rightarrow \psi$.*

Soundness of Predicate Calculus

Proposition 23.6 *Every wff of \mathcal{L}_1 which is an instance of a tautology of Propositional logic is a theorem of PC and may be obtained using only the axiom schemas K, S, N and the rule MP.*



Theorem 23.7 (Consistency of $\text{Th}(\text{PC})$). *The theory PC is consistent i.e. the set $\text{Th}(\text{PC})$ of theorems of PC form a consistent set.*



Proof of theorem 23.7

Proof: Define the *erasure* of a formula $e(\phi)$ as the propositional formula obtained by deleting all terms and all quantifiers and retaining only the atomic predicate symbols and propositional connectives. The function e may be defined by induction on the structure of the formula ϕ .

- *Claim 0.* For each instance of the axioms of \mathcal{H}_1 , the erasure of the formulae yields tautologies (of propositional logic)
- *Claim 1.* The erasure of each application of the rules of \mathcal{H}_1 , preserves tautologous-ness, i.e. if the erasure of the premises of a rule are all propositional tautologies then so is the erasure of the conclusion.
- *Claim 2.* The erasure of every formula in $\text{Th}(\mathbb{PC})$ is a tautology.

If $\text{Th}(\mathbb{PC})$ were inconsistent then $\text{Th}(\mathbb{PC}) = \mathcal{L}_1$. In particular, there exist formulae $\phi, \neg\phi \in \text{Th}(\mathbb{PC})$. But then by the definition of erasure we have $e(\neg\phi) \equiv \neg e(\phi)$ which contradicts *Claim 2*. QED ■

Soundness of The Hilbert System

Theorem 23.8 (Soundness of \mathcal{H}_1). *Every theorem of \mathcal{H}_1 is logically valid.*



Proof of theorem 23.8

Proof: Here are some easily proven claims which hold for any interpretation of Σ -formulae and hence also hold for the predicate calculus.

- *Claim 1* Every instance of a (propositional) tautology is true for every interpretation.
- *Claim 2* All instances of the axioms $\forall E$ and $\forall D$ are logically valid.
- *Claim 3* The rules MP and $\forall I$ preserve logical validity i.e. if the premises of the rules are logically valid formulae under every Σ -interpretation then so are the conclusions.

It then follows by induction on the structure of the proof tree that every theorem is logically valid.
QED

Exercise 23.1

1. Prove the arguments in Problem 2 of exercise 17.1 using the system \mathcal{H}_1 .
2. Prove the conclusions of Problem 2 in exercise 25.1 using only the proof rules of \mathcal{H}_1 .
3. Let $\psi \in SF(\phi)$. If ϕ' is obtained from ϕ by replacing zero or more occurrences of ψ by ψ' . Then prove the following.

(a) $\vec{\forall}[\psi \leftrightarrow \psi'] \vdash \phi \leftrightarrow \phi'$

(b) $\psi \leftrightarrow \psi' \vdash \phi \leftrightarrow \phi'$

(c) We also have the derived rule of inference $\Leftrightarrow . \quad \frac{\Gamma \vdash \psi \leftrightarrow \psi' \quad \Gamma \vdash \phi}{\Gamma \vdash \phi'}$ which is the only rule that allows the use of a rule that applies to proper sub-formulae of a formula.

4. If ϕ and ψ are formulae such that $x \notin FV(\phi)$, then the following are theorems of \mathcal{H}_1 ($\exists x[\phi]$ is an abbreviation of $\neg \forall x[\neg \phi]$).

(a) $\vdash \phi \rightarrow \forall x[\phi]$ and hence $\vdash \phi \leftrightarrow \forall x[\phi]$

(b) $\vdash \exists x[\phi] \rightarrow \phi$ and hence by rule $\exists I \vdash \exists x[\phi] \leftrightarrow \phi$

(c) $\vdash \forall x[\phi \rightarrow \psi] \leftrightarrow (\phi \rightarrow \forall x[\psi])$

(d) $\vdash \forall x[\psi \rightarrow \phi] \leftrightarrow (\exists x[\psi] \rightarrow \phi)$

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

524 OF 783

[QUIT](#)

Lecture 24: Existential Quantification

Wednesday 28 September 2011

1. Existential Generalisation
2. Existential Elimination
3. Remarks on Existential Elimination
4. Restrictions on Existential Elimination
5. Equivalence of Proofs

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

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

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

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

Proof of Derived rule $\exists I$

Assuming $\{t/x\}$ is admissible in ϕ we have

$$\frac{\text{DNI} \quad \frac{\forall E \frac{\Gamma \vdash \forall x[\neg\phi] \rightarrow \neg\{t/x\}\phi}{MP \frac{\Gamma \vdash (\forall x[\neg\phi] \rightarrow \neg\{t/x\}\phi) \rightarrow (\neg\neg\{t/x\}\phi \rightarrow \neg\forall x[\neg\phi])}{\Gamma \vdash \neg\neg\{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]}}}{T \rightarrow \frac{}{\Gamma \vdash \{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]}}$$

Assuming $\Gamma \vdash \{t/x\}\phi$, the last step gives

$$MP \frac{\overline{\Gamma \vdash \{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]} \quad \overline{\Gamma \vdash \{t/x\}\phi}}{\Gamma \vdash \neg\forall x[\neg\phi] \equiv \exists x[\phi]}$$

Existential Elimination

Existential generalisation from a constant introduced somehow into the proof is very common in mathematics.

Example 24.1 Consider a purported proof of

$$\exists x[\phi \rightarrow \psi], \forall x[\phi] \vdash \exists x[\psi]$$

where x may be a free variable of both ϕ and ψ . The standard practice is to assume the existence of some “constant symbol” a and proceed with it to eventually generalize.

Let $\Gamma = \{\exists x[\phi \rightarrow \psi], \forall x[\phi]\}$.

A $\Gamma \vdash \exists x[\psi]$ proof is as follows.

A proof using $\exists E$

$$\text{MP} \frac{\exists E \frac{\Gamma \vdash \exists x[\phi \rightarrow \psi]}{\Gamma \vdash \{a/x\}(\phi \rightarrow \psi) \equiv \{a/x\}\phi \rightarrow \{a/x\}\psi} \quad \forall E \frac{\Gamma \vdash \forall x[\phi]}{\Gamma \vdash \{a/x\}\phi}}{\exists I \frac{\Gamma \vdash \{a/x\}\psi}{\Gamma \vdash \exists x[\psi]}}$$

Unless the same “constant symbol” is used both in the application of $\exists E$ and $\forall E$ the proof will not go through. Here is a proof not involving the use of the constant (which is reminiscent of a proof by contradiction) which assumes there is no value for x which will make ψ true. Let $\Delta = \{\forall x[\phi], \forall x[\neg\psi]\}$

$$\begin{array}{c}
 \forall E \frac{\Delta \vdash \forall x[\phi]}{\Delta \vdash \phi} \quad \forall E \frac{\Delta \vdash \forall x[\neg\psi]}{\Delta \vdash \neg\psi} (\neg(\phi \rightarrow \psi) \equiv \phi \wedge \neg\psi) \\
 \wedge I \frac{}{\forall I \frac{\Delta \vdash \neg(\phi \rightarrow \psi)}{\Delta \vdash \forall x[\neg(\phi \rightarrow \psi)]}} \\
 \text{DT} \Rightarrow \frac{}{\forall x[\phi] \vdash \forall x[\neg\psi] \rightarrow \forall x[\neg(\phi \rightarrow \psi)]}
 \end{array}$$

Note that the application of $\text{DT} \Rightarrow$ is correct since no free variable of Δ has been generalised in the proof so far. We may now proceed as follows. First from rule N' we get

$$N' \frac{}{\forall x[\phi] \vdash (\forall x[\neg\psi] \rightarrow \forall x[\neg(\phi \rightarrow \psi)]) \rightarrow (\neg \forall x[\neg(\phi \rightarrow \psi)] \rightarrow \neg \forall x[\neg\psi])}$$

An application of MP then yields

$$\text{DT} \Leftarrow \frac{\forall x[\phi] \vdash \neg \forall x[\neg(\phi \rightarrow \psi)] \rightarrow \neg \forall x[\neg\psi]}{\forall x[\phi], \neg \forall x[\neg(\phi \rightarrow \psi)] \equiv \exists x[\phi \rightarrow \psi] \vdash \neg \forall x[\neg\psi] \equiv \exists x[\psi]}$$

Remarks on Existential Elimination

1. $\exists E$ is not a derived rule.
2. Proofs which utilize $\exists E$ are denoted $\vdash_{\exists E}$.
3. There are some restrictions on the use of rules in $\vdash_{\exists E}$ proofs.
4. Proofs involving existential formulae which utilize $\exists E$ are likely to be more direct and intuitively simpler than proofs which avoid all uses of $\exists E$ even for existential formulae.

Restrictions on Existential Elimination

Definition 24.2 A proof $\vdash_{\exists E}$ is correct provided

1. Each application of $\exists E$ should use a “fresh constant” symbol not used previously in the proof.
2. If constant symbol a (earlier introduced by an application of $\exists E$ to a formula $\exists y[\psi]$) appears in a formula $\{a/x\}\phi$ in a proof, then $\forall z[\{a/x\}\phi]$ cannot be deduced for any variable $z \in FV(\exists y[\psi]) \cap FV(\{a/x\}\phi)$ (by applying $\forall I$ to $\{a/x\}\phi$).
3. A formula $\{a/x\}\phi$, where a is a constant symbol and $x \in FV(\phi)$ can only be generalised to $\exists x[\phi]$.

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 $\Gamma \vdash_{\exists E} \phi$.

Proof of theorem 24.3

Proof: Let \mathcal{T} rooted at $\Gamma \vdash_{\exists E} \phi$ be a correct proof which involves one or more applications of rule $\exists E$. Assume there are k applications of rule $\exists E$ in $\Gamma \vdash_{\exists E} \phi$.

Claim 1. For each i , $1 \leq i \leq k$ there exist proof trees

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{i-1}/y_{i-1}\}\psi_{i-1} \vdash \exists y_i[\psi_i]$$

which are completely free from any application of rule $\exists E$ and

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_i/y_i\}\psi_i \vdash_{\exists E} \phi$$

which have $(k - i)$ applications of rule $\exists E$.

Proof of claim 1. Starting from the highest level of \mathcal{T} there exists a subtree

$$\exists E \frac{\Gamma \vdash \exists y_1[\psi_1]}{\Gamma \vdash_{\exists E} \{a_1/y_1\}\psi_1}$$

such that the proof $\Gamma \vdash \exists y_1[\psi_1]$ does not involve any application of rule $\exists E$. For $i = 1$, this is also the required tree $\Gamma \vdash \exists y_1[\psi_1]$.

By adding the formula $\{a_1/y_1\}\psi_1$ to the set of assumptions Γ and removing the subtree $\Gamma \vdash \exists y_1[\psi_1]$ from \mathcal{T} we get a proof tree $\Gamma, \{a_1/y_1\}\psi_1 \vdash_{\exists E} \phi$ in which there are only $k - 1$ applications of rule $\exists E$ and $\Gamma, \{a_1/y_1\}\psi_1 \vdash_{\exists E} \{a_1/y_1\}\psi_1$ is a leaf node of \mathcal{T}^1 .

Starting with \mathcal{T}^1 we again remove the next application of rule $\exists E$ viz.

$$\exists E \frac{\Gamma, \{a_1/y_1\}\psi_1 \vdash \exists y_2[\psi_2]}{\Gamma, \{a_1/y_1\}\psi_1 \vdash_{\exists E} \{a_2/y_2\}\psi_2}$$

to obtain

$$\Gamma, \{a_1/y_1\}\psi_1, \{a_2/y_2\}\psi_2 \vdash_{\exists E} \phi$$

It is again clear that

$$\Gamma, \{a_1/y_1\}\psi_1 \vdash \exists y_2[\psi_2]$$

does not involve any application of rule $\exists E$ and is the required tree

$$\Gamma, \{a_1/y_1\}\psi_1 \vdash \exists y_2[\psi_2]$$

Proceeding in this fashion we get proof trees

$$\Gamma, \{a_1/y_1\}\psi_1, \dots \{a_i/y_i\}\psi_i \vdash_{\exists E} \phi$$

in which there are exactly $k - i$ applications of rule $\exists E$ and for $i = k$ we get

$$\Gamma, \{a_1/y_1\}\psi_1, \dots \{a_k/y_k\}\psi_k \vdash \phi$$

which is completely free from all applications of rule $\exists E$.

Further it is also clear that for each i , $1 \leq i \leq k$ there exist proof trees

$$\Gamma, \{a_1/y_1\}\psi_1, \dots \{a_{i-1}/y_{i-1}\}\psi_{i-1} \vdash \exists y_i[\psi_i]$$

which are also completely free from any application of rule $\exists E$.

End of proof of claim 1

By part 2 of definition 24.2 no variable $z \in \bigcup_{1 \leq i \leq k} FV(\exists y_i[\psi_i])$ is generalized anywhere in the proof of $\Gamma \vdash_{\exists E} \phi$. Hence the conditions for the application of DT \Rightarrow hold.

By DT \Rightarrow we get

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \{a_k/y_k\}\psi_k \rightarrow \phi \quad (18)$$

Since the original proof is correct by definition 24.2, there is no occurrence of any of the constants a_i , $1 \leq i \leq k$, in ϕ . Further the proof (18) does not utilise the constant a_k as anything more than a symbol. It also does not generalise on a_k anywhere within the proof. Clearly then replacing this constant a_k by a “fresh” variable symbol z_k (which occurs nowhere in any of the proofs including proof (18)) does not affect the correctness of the proof.

Take a fresh variable z_k which does not occur anywhere in the proof (18) and replace all occurrences of a_k by z_k to obtain proof (19).

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \{z_k/y_k\}\psi_k \rightarrow \phi \quad (19)$$

Proof (19) may now be extended as follows.

$$\forall I \frac{\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \{z_k/y_k\}\psi_k \rightarrow \phi}{\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \forall z_k[\{z_k/y_k\}\psi_k \rightarrow \phi]}$$

which is again a correct proof.

Applying exercise 23.1.4d to the last step gives us

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \exists y_k[\psi_k] \rightarrow \phi$$

By claim 1 we know there exists a proof of

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \exists y_k[\psi_k]$$

which is completely free of any application of rule $\exists E$. By applying rule MP we therefore get

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \phi$$

By a similar process we may eliminate each of the constants a_{k-1} down to a_1 to eventually obtain a proof $\Gamma \vdash \phi$.

QED



Exercise 24.1

1. Use the method outlined in the proof of theorem 24.3 to transform the proof $\Gamma \vdash_{\exists E} \exists x[\psi]$ of example 24.1 to one without the use of rule $\exists E$.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

541 OF 783

QUIT

Lecture 25: Normal Forms

Friday 30 September 2011

1. Natural Deduction: 6
2. Moving Quantifiers
3. Quantifier Movement
4. More on Quantifier Movement
5. Prenex Normal Forms
6. The Prenex Normal Form Theorem
7. Prenex Conjunctive Normal Form
8. The Herbrand Algebra
9. Terms in a Herbrand Algebra
10. Herbrand Interpretations
11. Herbrand Models
12. Ground Quantifier-free Formulae

Natural Deduction: 6

The introduction and elimination rules for the propositional operators along with the rules $\forall I$, $\forall E$, $\exists I$ and $\exists E$ comprise the system \mathcal{G}_1 .

	Introduction	Elimination
\forall	$\forall I.$ $\frac{\Gamma \vdash \{t/x\}X}{\Gamma \vdash \forall x[X]}$ $\{t/x\}$ admissible in X	$\forall E.$ $\frac{\Gamma \vdash \forall x[X]}{\Gamma \vdash \{t/x\}X}$
\exists	$\exists I.$ $\frac{\Gamma \vdash \{t/x\}X}{\Gamma \vdash \exists x[X]}$	$\exists E.$ $\frac{\Gamma \vdash \exists x[\phi]}{\Gamma \vdash \{a/x\}\phi}$ $a \notin FV(\Gamma) \cup FV(\exists x[\phi])$ is fresh

Exercise 25.1

1. Prove the arguments in Problem 2 of exercise 17.1 using Natural Deduction.
2. There have been frequent complaints that Logic (of any order) is cold-blooded of the first order. Let's dispel this notion. Consider the following premises.

All the world loves a lover. Romeo loves Juliet.

Now prove the following conclusions using Natural Deduction.

- (a) Therefore I love you.
- (b) Therefore Love loves Love.¹⁰
- (c) Therefore if I love you, then you love me.
- (d) Therefore you love yourself.
- (e) Therefore everyone loves everyone.

¹⁰This is of course a dirty trick!

3. Refer to the premises in Problem 2 above. Which of the conclusions becomes invalid if the premise Romeo loves Juliet is removed? Further, does it follow that love is an equivalence relation?

Moving Quantifiers

Notation.

1. $\overrightarrow{\mathcal{O}x}$ denotes a sequence of quantifiers

$$\mathcal{O}_1 x_1 \mathcal{O}_2 x_2 \dots \mathcal{O}_m x_m$$

where $m \geq 0$ and each $\mathcal{O}_i \in \{\forall, \exists\}$, for $1 \leq i \leq m$.

2. For any quantifier $\mathcal{O} \in \{\forall, \exists\}$, $\bar{\mathcal{O}}$ denotes its dual. That is, if $\mathcal{O} = \forall$, then $\bar{\mathcal{O}} = \exists$ and if $\mathcal{O} = \exists$ then $\bar{\mathcal{O}} = \forall$.

Quantifier Movement

Lemma 25.1 Let $z \notin FV(\phi) \cup FV(\psi) \cup \{x_1, \dots, x_n\}$. Then the following logical equivalences hold for $\mathcal{O}' \in \{\forall, \exists\}$.

1. $\overrightarrow{\mathcal{O}x}\neg\mathcal{O}'y[\phi] \Leftrightarrow \overrightarrow{\mathcal{O}x}\overline{\mathcal{O}'y}[\neg\phi]$
2. $\overrightarrow{\mathcal{O}x}[\mathcal{O}'y[\phi] \vee \psi] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\{z/y\}\phi \vee \psi]$
3. $\overrightarrow{\mathcal{O}x}[\phi \vee \mathcal{O}'y[\psi]] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\phi \vee \{z/y\}\psi]$



More on Quantifier Movement

We may use the above lemma to obtain *prenexing* rules for the propositional connectives \wedge and \rightarrow as well, as shown in the following corollary. However, note the change of quantifier that marks the transformation of \rightarrow in the last equivalence.

Corollary 25.2

4. $\overrightarrow{\mathcal{O}x}[\mathcal{O}'y[\phi] \wedge \psi] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\{z/y\}\phi \wedge \psi]$
5. $\overrightarrow{\mathcal{O}x}[\phi \wedge \mathcal{O}'y[\psi]] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\phi \wedge \{z/y\}\psi]$
6. $\overrightarrow{\mathcal{O}x}[\phi \rightarrow \mathcal{O}'y[\psi]] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\phi \rightarrow \{z/y\}\psi]$
7. $\overrightarrow{\mathcal{O}x}[\mathcal{O}'y[\phi] \rightarrow \psi] \Leftrightarrow \overrightarrow{\mathcal{O}x}\mathcal{O}'z[\{z/y\}\phi \rightarrow \psi]$



Exercise 25.2

1. Prove lemma 25.1.
2. Prove corollary 25.2.

Having obtained the above results we are now ready to prove the Prenex normal form theorem. But first let's define the form precisely.

Prenex Normal Forms

Definition 25.3 *The set $\mathcal{PNF}_1(\Sigma)$ of prenex normal forms is generated by the grammar*

$$\phi, \psi ::= \chi \in \mathcal{QF}_1(\Sigma) \mid \forall \textcolor{violet}{x}[\psi] \mid \exists \textcolor{violet}{x}[\psi]$$

*A formula is in prenex normal form if all the quantifiers occurring in the formula appear as a prefix $\overrightarrow{\partial \mathbf{x}}$ (called the **prenex**) of a body that is “quantifier-free” and consists only of atomic predicates with propositional connectives.*

The Prenex Normal Form Theorem

Theorem 25.4 (Prenex Normal Forms). *For any formula ϕ there exists a logically equivalent formula ψ in prenex normal form (PNF).*



Proof of theorem 25.4

Proof: Given a formula ϕ we go through the following steps.

1. Replace all subformulas of the form $\theta \leftrightarrow \chi$ by subformulas

$$(\theta \rightarrow \chi) \wedge (\chi \rightarrow \theta)$$

respectively to yield a new formula ϕ' which is free of all occurrences of the connective \leftrightarrow .

2. Use α -conversion to obtain unique names for all bound and free variables¹¹.
3. Now proceed by induction on the structure of ϕ' by systematically applying the results obtained from lemma 25.1 and corollary 25.2. This would yield a formula ψ in prenex normal form.

QED



¹¹That is, ensure that no two quantifiers use the same bound variable and no variable occurs both free and bound in the formula.

Prenex Conjunctive Normal Form

Given a formula in prenex normal form, its body consists entirely of propositional connectives atomic predicates. By theorem 5.10 every propositional form may be converted into CNF. We may apply the same method to the body of a formula in PNF to obtain a **Prenex Conjunctive Normal Form (PCNF)**. So we have

Corollary 25.5 (PCNF). *For any formula ϕ there exists a logically equivalent formula ψ in prenex conjunctive normal form (PCNF).*



The Herbrand Algebra

Definition 25.6 Let Σ be a signature containing at least one constant symbol a . A term $t \in T(\Sigma)$ is said to be **ground** if $Var(t) = \emptyset$. $T_0(\Sigma) \subseteq T(\Sigma)$ is the set of ground terms (also called the **Herbrand Universe**). A literal $p(t_1, \dots, t_n)$ or $\neg p(t_1, \dots, t_n)$ containing no variables is called a **ground literal**.

Definition 25.7 A Σ -algebra $H(\Sigma)$ where Σ has at least one constant symbol, is called a **Herbrand algebra** iff $|H(\Sigma)| = T_0(\Sigma)$.

Terms in a Herbrand Algebra

In a Herbrand algebra $\mathbf{H}(\Sigma)$

- every function symbol represents itself. That is for each f :
 $s^m \rightarrow s \in \Sigma$, $f_{\mathbf{H}(\Sigma)} = f$
- a valuation is simply a function $v_{\mathbf{H}(\Sigma)} : V \rightarrow T_0(\Sigma)$
- However the predicate symbols require to be associated with relations on ground terms in some way.

Herbrand Interpretations

Lemma 25.8 Given a Herbrand interpretation $(\mathbf{H}, v_{\mathbf{H}})$ where for each variable x , $v(x) = s_x \in T_0(\Sigma)$. For any term t with $\text{Var}(t) = \{x_1, \dots, x_k\}$

$$\mathcal{V}_{\mathbf{H}}[\![t]\!]_{v_{\mathbf{H}}} = \{s_{x_1}/x_1, \dots, s_{x_k}/x_k\}^t$$

That is every valuation defines a substitution of variables by ground terms.

■ Here valuations represent merely substitutions on terms.

Herbrand Models

Definition 25.9 A Herbrand model of a set Φ of Σ -formulae is merely a valuation $v_{\mathbf{H}}$ such that every formula in Φ is true under the substitution defined by $v_{\mathbf{H}} \upharpoonright FV(\Phi)$.

Ground Quantifier-free Formulae

Theorem 25.10 Let Σ be a signature containing at least one constant and let $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ be a nonempty set of ground literals. Then

1. $\bigwedge_{1 \leq i \leq k} \lambda_i$ has a model iff Λ does not contain a complementary pair.
2. $\bigwedge_{1 \leq i \leq k} \lambda_i$ is never logically valid
3. $\bigvee_{1 \leq i \leq k} \lambda_i$ always has a model
4. $\bigvee_{1 \leq i \leq k} \lambda_i$ is logically valid iff it has a complementary pair.

Proof of theorem 25.10

Proof:

1. Clearly if Λ contains a complementary pair it does not have a model. Conversely assume it does not contain a complementary pair. We may define a Herbrand algebra \mathbf{H}_Λ as follows: For each atomic predicate symbol $p : s^n$ define $p_{\mathbf{H}_\Lambda} = \{(t_1, \dots, t_n) \in T_0(\Sigma) \mid p(t_1, \dots, t_n) \in \Lambda\}$

Clearly $\mathbf{H}_\Lambda \Vdash \lambda_i$ for each $\lambda_i \in \Lambda$ since if $\lambda_i \equiv p(t_1, \dots, t_n)$ and then $p(t_1, \dots, t_n) \in \Lambda$ and $(t_1, \dots, t_n) \in p_{\mathbf{H}_\Lambda}$. On the other hand if $\lambda_i \equiv \neg p(t_1, \dots, t_n)$ then $p(t_1, \dots, t_n) \notin \Lambda$ and hence $(t_1, \dots, t_n) \notin p_{\mathbf{H}_\Lambda}$ otherwise it would contradict the assumption that Λ contains no complementary pair. Hence $\mathbf{H}_\Lambda \Vdash \lambda_i$ for each λ_i . Hence $\bigwedge_{1 \leq i \leq k} \lambda_i$ has a model.

2. $\bigwedge_{1 \leq i \leq k} \lambda_i$ cannot be valid since from the previous part we know that $\overline{\lambda_i}$ has a model, where $\overline{\lambda_i}$ is the complement of λ_i .
3. $\bigvee_{1 \leq i \leq k} \lambda_i$ has a model because λ_i has a model.

4. $\bigvee_{1 \leq i \leq k} \lambda_i$ is valid iff $\bigwedge_{1 \leq i \leq k} \bar{\lambda}_i$ has no model iff $\bar{\Lambda} = \{\bar{\lambda}_i \mid 1 \leq i \leq k\}$ contains a complementary pair iff Λ contains a complementary pair.

QED

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

562 OF 783

QUIT

Lecture 26: Skolemization

Tuesday 04 Oct 2011

1. Skolemization
2. Skolem Normal Forms
3. SCNF
4. Ground Instance
5. Herbrand's Theorem
6. The Herbrand Tree of Interpretations
7. Compactness of Sets of Ground Formulae
8. Compactness of Closed Formulae
9. The Löwenheim-Skolem Theorem

Skolemization

Theorem 26.1 (Skolem Normal Form Theorem.) Let $\phi \equiv \vec{\forall} \mathbf{x} \exists \mathbf{y} [\psi]$ where $\mathbf{x} = x_1, \dots, x_n$ and \mathbf{y} are all distinct variables and ψ does not contain any occurrence of any of the quantifiers $\exists x_i$. Let $\Sigma_g = \Sigma \cup \{g : s^n \rightarrow s\}$ be an *expansion* of the signature Σ . Then

1. every model of $\phi' \equiv \vec{\forall} \mathbf{x} [\{g(x_1, \dots, x_n)/y\} \psi]$ is a model of ϕ .
2. every model of ϕ can be expanded to a model of ϕ' .



Corollary 26.2 Let ϕ and ϕ' be as in theorem 26.1. Then

1. there exists a model of ϕ iff there exists a model of ϕ' .
2. ϕ is unsatisfiable iff ϕ' is unsatisfiable.

Proof of theorem 26.1

Proof: We first of all note that $\{g(x_1, \dots, x_n)/y\}$ is admissible in ψ .

1. We have

$$\models \phi' \rightarrow \phi$$

and hence for every Σ_g -model $\mathbf{B} \Vdash \phi'$, $\mathbf{B} \Vdash \phi$ holds too.

2. Conversely for every Σ -structure \mathbf{A} , such that $\mathbf{A} \Vdash \phi$, for each $(a_1, \dots, a_n) \in |\mathbf{A}|^n$ there exists *at least one* element $a \in |\mathbf{A}|$ such that

$$\mathcal{T}[\![\psi]\!]_{v[x_1:=a_1, \dots, x_n:=a_n][y:=a]} = 1$$

Define a function g which for each n -tuple $(a_1, \dots, a_n) \in |\mathbf{A}|^n$ provides a single element $a \in |\mathbf{A}|$ such that

$$\mathcal{T}[\![\psi]\!]_{v[x_1:=a_1, \dots, x_n:=a_n][y:=a]} = 1$$

Then clearly for every $(a_1, \dots, a_n) \in |\mathbf{A}|^n$ we have

$$\mathcal{T}[\![\psi]\!]_{v[x_1:=a_1, \dots, x_n:=a_n][y:=g_{\mathbf{A}}(a_1, \dots, a_n)]} = 1$$

Now let $\mathbf{A} \lhd \mathbf{B}$ where \mathbf{B} is a Σ_g -algebra with $g_{\mathbf{B}} = g$. It is then clear that for every valuation $v_{\mathbf{B}}$,

$$\mathcal{T}[\![\{g(x_1, \dots, x_n)/y\}\psi]\!]_{v_{\mathbf{B}}} = 1$$

and hence $\mathbf{B} \Vdash \phi'$.

QED



Skolem Normal Forms

Definition 26.3

1. The set $\mathcal{SNF}_1(\Sigma)$ of **Skolem normal forms** is the set of *universal closures* of quantifier-free formulae. The quantifier-free formula is called the **body** of the SNF.
2. A formula is in **Skolem conjunctive normal form (SCNF)** if it is a SNF whose body is in CNF. $\mathcal{SCNF}_1(\Sigma)$ is the set of Σ -formulae in SCNF.

SCNF

Theorem 26.4 For every sentence (closed formula) $\phi \in \mathcal{P}_1(\Sigma)$ there is an algorithm sko to construct a closed universal formula $\psi \in \mathcal{SCNF}_1(\Sigma)$ such that ϕ has a model iff ψ has a model.



Definition 26.5

1. The function g in theorem 26.1 is called a **Skolem function**
2. The process of constructing the function g in theorem 26.1 is called **Skolemization**.
3. ϕ and $sko(\phi) \equiv \psi$ are said to be **equi-satisfiable**.

Proof of theorem 26.4

Proof: The procedure may be briefly outlined as follows.

1. Clearly if ϕ is a closed formula, by theorem 25.5 we can construct a logically equivalent formula $\phi_0 \in \mathcal{PCNF}_1(\Sigma)$.
2. If there is no existential quantifier ϕ_0 then the formula ϕ_0 is in SCNF. Otherwise Skolemize the left-most occurrence of an existential quantifier to obtain a formula ϕ_1 .
3. ϕ_1 has one existential quantifier less than ϕ_0 . Perform step 2 on ϕ_1 .

Each execution of step 2 of the above procedure results in a formula which satisfies the conclusions of theorem 26.1. ■QED

Exercise 26.1

1. Prove that $\models \phi' \rightarrow \phi$ in theorem 26.1.
2. Prove that $\vdash \phi' \rightarrow \phi$ in theorem 26.1.
3. Skolemization does not produce a SNF that is unique upto logical equivalence. Construct an

example of a formula ϕ which has two (or even more) different SNFS, ϕ' , ϕ'' such that

$$\phi \not\leq \phi' \not\leq \phi'' \not\leq \phi$$

Ground Instance

Definition 26.6 Let Σ be a signature containing at least one constant and let Φ be a nonempty set of closed universal Σ -formulae. For any $\phi \equiv \vec{\forall}[\chi]$ where $\chi \in Q\mathcal{F}_1(\Sigma)$, the **ground-instances** of ϕ denoted $\mathfrak{g}(\phi)$ is the set $\{\{t_1/x_1, \dots, t_n/x_n\}\chi \mid FV(\chi) = \{x_1, \dots, x_n\}, t_1, \dots, t_n \in T_0(\Sigma)\}$ and $\mathfrak{g}(\Phi) = \bigcup_{\phi \in \Phi} \mathfrak{g}(\phi)$.

Herbrand's Theorem

Theorem 26.7 (Herbrand's Theorem). *Let Σ and Φ be as in definition 26.6. Then the following statements are equivalent.*

1. Φ has a model.
2. Φ has a Herbrand model.
3. $\mathfrak{H}(\Phi)$ has a model.
4. $\mathfrak{H}(\Phi)$ has a Herbrand model.



Proof of theorem 26.7

Proof: Clearly the following implications are trivial.

Statement 2: " Φ has a Herbrand model" \Rightarrow Statement 1: " Φ has a model"



Statement 4: " $\mathbf{g}(\Phi)$ has a Herbrand model" \Rightarrow Statement 3: " $\mathbf{g}(\Phi)$ has a model"

It suffices therefore to prove only the following claim.

Claim. Statement 3 \Rightarrow Statement 2.

Proof of claim. Let $\mathbf{A} \Vdash \mathbf{g}(\Phi)$. We define a Herbrand interpretation \mathbf{H} as follows. For each $p : s^n \in \Sigma$ let

$$p_{\mathbf{H}} = \{(t_1, \dots, t_n) \in T_0(\Sigma) \mid \mathbf{A} \Vdash p(t_1, \dots, t_n)\}$$

In particular if p is an atomic proposition then $p_{\mathbf{H}} = p_{\mathbf{A}}$. With this construction exactly the same atomic formulae are valid in both \mathbf{A} and \mathbf{H} . This result may be extended by structural induction to arbitrary universally closed quantifier-free formulae. It is then easy to see that if $\mathbf{A} \Vdash \mathbf{g}(\Phi)$ then $\mathbf{H} \Vdash \Phi$. QED ■

The Herbrand Tree of Interpretations

Let Σ be a signature containing at least one constant symbol. Let P_0, P_1, P_2, \dots be an enumeration of all the *ground atomic formulae* of $\mathcal{P}_1(\Sigma)$. The **Herbrand Tree of interpretations** is the infinite tree shown schematically below. Each infinite path (called a **Herbrand Base**) of the tree represents a Herbrand interpretation.

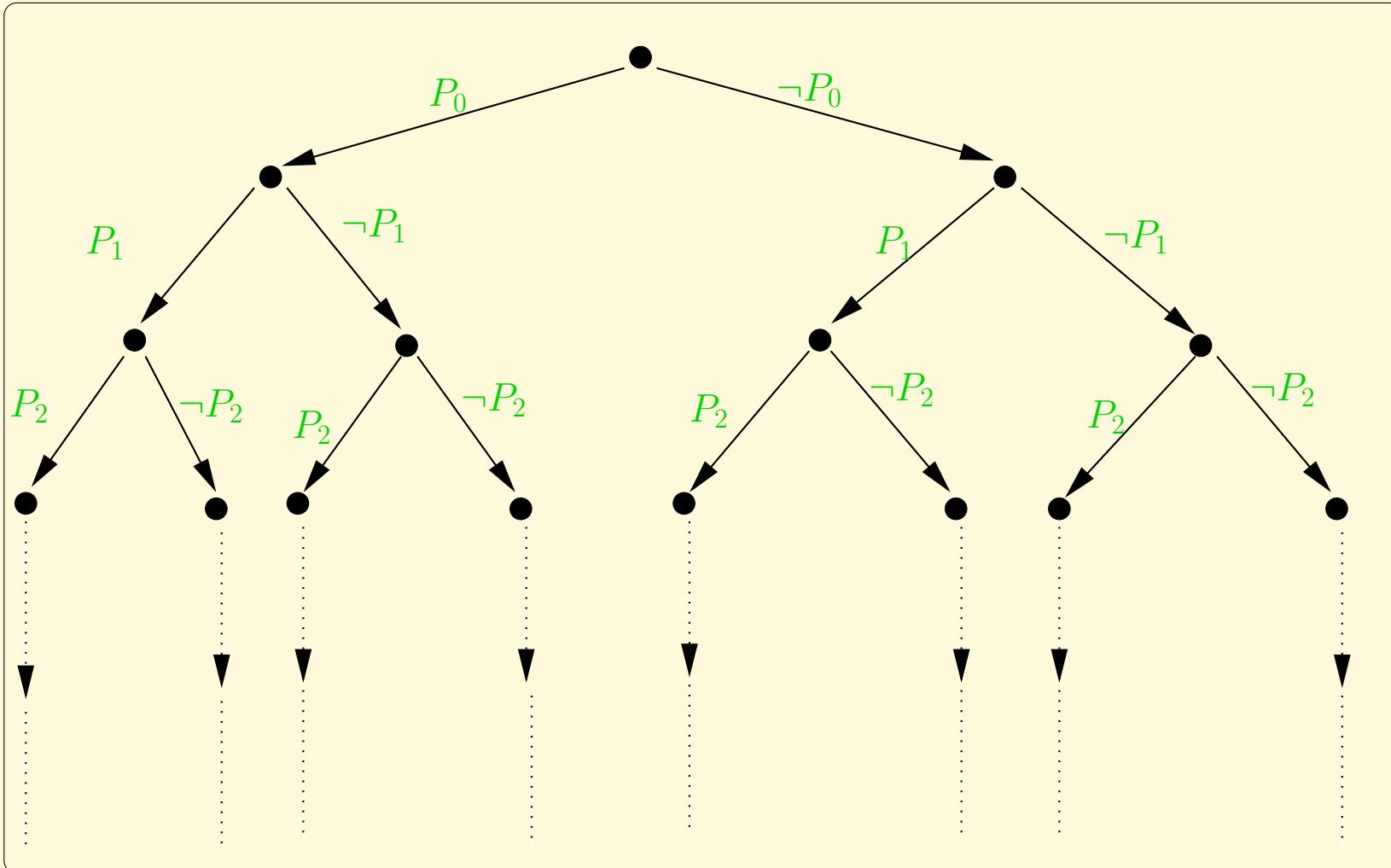


Figure 8: The Herbrand tree of interpretations

Compactness of Sets of Ground Formulae

Lemma 26.8 (Compactness of a set of closed quantifier-free formulae). *Let Θ be a (finite or infinite) set of ground quantifier-free formulae. Then Θ has a model iff every finite subset of Θ has a model.*



Proof of lemma 26.8

Proof: (\Rightarrow) Clearly if Θ has a model then every finite subset of Θ also has a model.

(\Leftarrow) Assume every finite subset of Θ has a model but Θ itself does not have a model. By theorem 26.7 each finite subset of Θ has a Herbrand model. We identify each path π in the Herbrand tree with a valuation v_π . Then since Θ does not have a model, it does not have a Herbrand model. Hence for every path π there exists a formula $\chi_\pi \in \Theta$ such that $(\mathbf{H}, v_\pi) \not\models \chi_\pi$. In fact there exists a finite point ℓ_{χ_π} in each path π at which $(\mathbf{H}, v_\pi) \not\models \chi_\pi$ since χ_π is made up of only a finite number of ground atoms.

Claim. $\{\chi_\pi \mid (\mathbf{H}, v_\pi) \not\models \chi_\pi\}$ is a finite set.

Proof of claim.

Consider the tree \mathcal{T}_H obtained from the Herbrand tree such that from each path π the subtree rooted at $\ell_{\chi_\pi} + 1$ has been removed. Hence \mathcal{T}_H is a finitely branching tree with only finite-length paths. Hence by (the contra-positive of) König's lemma (lemma 2.17: any finitely-branching tree with only finite-length paths must be finite) \mathcal{T}_H must be a finite tree where each path π' is an initial segment of an infinite path π from the Herbrand tree of interpretations. The leaf-nodes of each of these paths π' determines a formula χ_π that is not satisfied. Clearly then the set consisting of these formulae viz.

$\{\chi_\pi \mid (\mathbf{H}, v_\pi) \not\models \chi_\pi\}$ is then a finite set.

End of proof of claim.

It is clear that the finite set $\{\chi_\pi \mid (\mathbf{H}, v_\pi) \not\models \chi_\pi\}$ is a finite subset of Θ that does not possess a Herbrand model, contradicting the assumption that all finite subsets of Θ possess a Herbrand model.
QED



Compactness of Closed Formulae

Theorem 26.9 (Compactness of closed formulae) *A set Φ of closed Σ -formulae has a model iff every finite subset of Φ has a model.*



Corollary 26.10 (Finite Unsatisfiability). *A set Φ of closed Σ -formulae is unsatisfiable iff there is a non-empty finite unsatisfiable subset of Φ .*



Proof of theorem 26.9

Proof: (\Rightarrow) is trivial.

(\Leftarrow) Assume Φ does not possess a model but every finite subset of Φ has a model. Transform each formula into SNF. Since Φ has no model $sko(\Phi) = \{sko(\phi) \mid \phi \in \Phi\}$ has no model either (by theorem 26.4). By Herbrand's theorem (theorem 26.7) the set $g(sko(\Phi))$ also does not possess a model. By lemma 26.8 we can find a finite subset of $g(sko(\Phi))$ which does not have a Herbrand model. This finite set is a subset of a finite subset of Φ that does not possess a model. Hence there is a finite subset of Φ which does not have a model, contradicting the assumption that every finite subset of Φ has a model. QED ■

The Löwenheim-Skolem Theorem

Theorem 26.11 (The Löwenheim-Skolem Theorem). *If a set Φ of closed formulae has a model, then it has a model with a domain which is at most countable.*

Proof: Assume Φ has a model. Then $sko(\Phi)$ has a model too. By theorem 26.7 $sko(\Phi)$ has a Herbrand model. Since a Herbrand model has a domain which is at most countable and since every model of $sko(\Phi)$ is also a model of Φ , it follows that Φ has a model with at most a countable domain. QED ■

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

583 OF 783

QUIT

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

584 OF 783

[QUIT](#)

Lecture 27: Substitutions and Instantiations

Tuesday 11 October 2011

1. Substitutions Revisited
2. Some Simple Facts
3. Ground Substitutions
4. Composition of Substitutions
5. Substitutions: A Monoid

Substitutions Revisited

We have defined **substitutions** and **instantiations** earlier. In light of the Löwenheim-Skolem theorem [26.11](#), we

1. require more powerful operations on syntactic substitutions to exploit the construction of Herbrand models,
2. need to extend the theory of substitutions to include a *composition* operator for substitutions.
3. need to give a programming interpretation to First-order logic.

Some Simple Facts

Definition 27.1

1. $\mathbf{1}$ is the empty or identity substitution which replaces no variable when applied to any term and $\text{dom}(\mathbf{1}) = \emptyset = \text{ran}(\mathbf{1})$.
2. $S_\Omega(V)$ is the class of all substitutions over a set of variables V for a given signature Ω .

Fact 27.2 Let θ be a substitution and t a term. Then

1. $\text{depth}(\theta t) \geq \text{depth}(t)$
2. $\text{size}(\theta t) \geq \text{size}(t)$.

Ground Substitutions

Definition 27.3

- $\theta = \{t_i/x_i \mid t_i \not\equiv x_i, 1 \leq i \leq n\}$ is a **ground substitution** if each $t_i, 1 \leq i \leq n$, is a **ground term**.
- A term u is called an **instance** of a term t if there exists a substitution θ such that $u \equiv \theta t$.
- u is a **ground instance** of t if u is an instance of t and is ground.
- u is a **common instance** of two or more terms t_1, \dots, t_n if there exist substitutions $\theta_1, \dots, \theta_n$ such that

$$u \equiv \theta_1 t_1 \equiv \dots \equiv \theta_n t_n$$

- Terms t and u are called **variants** of each other if there exist substitutions θ and τ such that $\theta t \equiv u$ and $\tau u = t$.

Composing Substitutions

We will often require to perform substitutions in sequence i.e. it may be necessary to first apply a substitution θ on a term t yielding a term θt to which another substitution τ may be applied to yield a term $\tau(\theta t)$. We would like to answer the question of how to define a single substitution ρ such that for every term u ,

$$\tau(\theta u) \equiv \rho u \quad (20)$$

Then ρ is the *composition* of τ with θ . Before presenting the formal definition of composition we try to understand how such a composition must be defined to ensure that equation (20) holds. Let $\theta = \{s_1/x_1, \dots, s_k/x_k\}$ and $\tau = \{t_1/y_1, \dots, t_m/y_m\}$. We have $dom(\theta) = X = \{x_1, \dots, x_k\}$ and $dom(\tau) = Y = \{y_1, \dots, y_m\}$. The effect of θ on any term u is to replace each free occurrence of each variable x_i by the term s_i simultaneously for $1 \leq i \leq k$. The terms s_i could contain (free) variables drawn from X and Y . It could also happen that some of the terms s_i may simply be variables themselves. Consider a single variable x_i . If $s_i \equiv z$ for some variable z , then θu would simply have z occurring free in all those positions of u where x_i occurs free. Of course, free occurrences of x_i could be present in θu because of some other variable substitution (say $s_{i'}/x_{i'}$ for some $i' \neq i$). Hence it is clear that all free occurrences of any $x \in X$ in θt are due to the application of the substitution θ . Further, for any $y_j \in Y$, we have the following possibilities.

1. *Case $y_j \in Y - X$ and $y_j \in FV(u)$.* All such free occurrences of y_j in u will be present in the same positions in θu as well. The effect of τ would be to replace them all with t_j .
2. *Case $y_j \in Y - X$ and $y_j \notin FV(u)$.* New free occurrences may arise due to the substitution θ . The effect of the application of τ will replace all of them by t_j .

3. Case $y_j \equiv x_i$ for some $x_i \in X$. In this case the only free occurrences of y_j possible are those which occur after applying θ .

To summarise

1. Case 1 requires τ to be applied separately.
2. The effect of τ on cases 2 and 3 may be captured by applying τ to the range of θ . Once that is done one may even remove the element t_j/y_j from the substitution, since it would have no effect.
3. Further all elements such that $\tau s_i \equiv x_i$ are removed from ρ , since we are interested in specifying the substitution as a finite set of non-identical replacements.

With this understanding we are ready to tackle our definition of composition.

Composition of Substitutions

Definition 27.4 Given substitutions $\theta = \{s_1/x_1, \dots, s_k/x_k\}$ and $\tau = \{t_1/y_1, \dots, t_m/y_m\}$, their composition $\tau \circ \theta$ is a new substitution ρ such that

$$\rho = \{\tau s_i/x_i \mid 1 \leq i \leq k, \tau s_i \not\equiv x_i\} \cup \{t_j/y_j \mid 1 \leq j \leq m, y_j \notin \text{dom}(\theta)\}$$

Substitutions: A Monoid

Lemma 27.5 *Given substitutions θ, τ, ρ and a term t , we have*

1. $\theta \circ \mathbf{1} = \mathbf{1} \circ \theta = \theta$
2. $(\tau \circ \theta)t \equiv \tau(\theta t)$
3. $\rho \circ (\tau \circ \theta) = (\rho \circ \tau) \circ \theta$



Proof of lemma 27.5

Proof: We assume $\theta = \{s_1/x_1, \dots, s_k/x_k\}$ and $\tau = \{t_1/y_1, \dots, t_m/y_m\}$ and $\rho = \tau \circ \theta$ as in definition 27.4. Then

1. Trivial.
2. We prove this by induction on the structure of terms. The case of constants is trivial. The induction case will also follow once the cases of simple variables has been proven. So we simply prove this case for simple variables. For any variable x we have the following cases.

Case $x \notin \text{dom}(\theta)$. Then clearly $(\theta x) \equiv x$ and $\tau(\theta x) \equiv \tau x$. Since the first component of the union in the definition of ρ does not apply, we have $\rho x \equiv \tau x$.

Case $x \equiv x_i \in \text{dom}(\theta)$ for some i , $1 \leq i \leq m$. In this case $\rho x_i \equiv \tau s_i$ and since $\theta x_i \equiv s_i$ we have $\tau(\theta x_i) \equiv \tau s_i$.

3. For any term t we have from the previous proof

$$(\rho \circ (\tau \circ \theta))t \equiv \rho((\tau \circ \theta)t) \equiv \rho(\tau(\theta t)) \equiv (\rho \circ \tau)(\theta t) \equiv ((\rho \circ \tau) \circ \theta)t$$

QED



Exercise 27.1

1. Let $u \equiv \theta t$. Give examples of t , u and θ such that $FV(t) \neq \emptyset$, u is ground but θ is not a ground substitution.
2. Prove that for any substitutions θ and τ , $\tau \circ \theta = \tau \cup \theta$ iff $dom(\theta) \cap dom(\tau) = \emptyset$ and $dom(\tau) \cap \bigcup_{t \in ran(\theta)} FV(t) = \emptyset$
3. A substitution θ is called **idempotent** if $\theta \circ \theta = \theta$. Now complete the statement of the following lemma and prove it.

Lemma 27.6 A substitution $\theta = \{t_i/x_i \mid 1 \leq i \leq m\}$ for some $m \geq 0$ is idempotent iff $dom(\theta) \dots$.

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

596 OF 783

[QUIT](#)

Lecture 28: Unification

Wednesday 12 October 2011

1. Unifiability
2. Unification Examples:1
3. Unification Examples:2
4. Generality of Unifiers
5. Generality: Facts
6. Most General Unifiers
7. More on Positions
8. Disagreement Set
9. Example: Disagreement 1
10. Example: Occurs Check
11. Example: Disagreement 3
12. Example: Disagreement 4
13. Disagreement and Unifiability
14. The Unification Theorem

Syntactic unification is the problem of finding substitutions θ so as to make two or more terms syntactically identical. It may be thought of as a special form of equation solving where one attempts to find solutions to the problem $s \equiv t$ by finding suitable instances of the variables in the two terms in order to make the two terms look identical. The solution of such an equation on essentially uninterpreted terms is a substitution and the process of finding this solution is called *unification*. As in normal equation solving the substitution is to be applied to all the terms that have to be unified. Moreover as in equation solving, it is possible that no solution exists. A set consisting of two or more terms is said to be *unifiable* if such a substitution exists.

We will use words like “occurrence”, “sub-term”, “depth”, “size” quite liberally. In the light of the presence of several occurrences of operators, free variables and bound variables (including different bound variable occurrences signifying different variables but possessing the same name e.g. $(\lambda x[(xx)] \lambda x[(xx)])$) in a term, it is useful to define a unique *position* for each symbol in a term t . For any term t we have a set of strings $Pos(t) \subseteq \mathbb{N}^*$ which is the set of positions occurring in t . Further for each $p \in Pos(t)$, there is a unique symbol occurring at that position denoted by $pos(p, t)$.

Definition 28.1

t	$depth$	$size$	ST	pos
c	1	1	$\{\textcolor{violet}{t}\}$	$\{\epsilon\}$
x	1	1	$\{\textcolor{violet}{t}\}$	$\{\epsilon\}$
$o(t_1, \dots, t_n)$	$1 + \max_{i=1}^n depth(\textcolor{violet}{t}_i)$	$1 + \sum_{i=1}^n size(\textcolor{violet}{t}_i)$	$\{\textcolor{violet}{t}\} \cup \bigcup_{i=1}^n ST(\textcolor{violet}{t}_i)$	$\{\epsilon\} \cup \bigcup_{i=1}^n i.pos(\textcolor{violet}{t}_i)$

- The functions given in the table above are defined by induction on the structure of term $\textcolor{violet}{t}$. ϵ is the empty word on strings, $.$ is the catenation operator on strings and $i.pos(\textcolor{violet}{t}_i) = \{i.p \mid p \in pos(\textcolor{violet}{t}_i)\}$.
- $s \sqsubseteq t$ iff $s \in ST(\textcolor{violet}{t})$ is the **subterm** relation on terms. s is a **proper subterm** of t (denoted $s \sqsubset t$) iff $s \sqsubseteq t$ and $s \not\equiv t$.
- For any t , the subterm at position $p \in pos(\textcolor{violet}{t})$ is denoted $t|_p$ and defined by induction on p as follows: $t|_\epsilon \equiv t$, and for $t \equiv o(t_1, \dots, t_n)$, $t|_{i.p'} \equiv t_i|_p$ if $p = i.p' \in pos(\textcolor{violet}{t})$
- For any term t and any position $p \in pos(\textcolor{violet}{t})$, $sym(p, t)$ yields the symbol at position p in the term t .
- The position ϵ is called the **root position** and the symbol at the root position is called the **root symbol**. Hence $rootsym(t) = sym(\epsilon, t)$ and for any position $p \in pos(\textcolor{violet}{t})$, $sym(p, t) = rootsym(t|_p)$.

- The set of occurrences of a symbol $\sigma \in \Sigma \cup V$ in a term is defined as the set $\text{Occ}(\sigma, t)$ of positions in which that symbol occurs i.e. $\text{Occ}(\sigma, t) = \{p \in \text{pos}(t) \mid \text{sym}(p, t) = \sigma\}$.

Facts 28.2 For any term t and positions $p, q \in \text{pos}(t)$, $t|_q \sqsubset t|_p$ iff $p \prec q$.

Unifiability

Definition 28.3 A nonempty finite set of terms $\{t_i \mid 1 \leq i \leq n\}$, $n > 1$ is said to be **unifiable** if there exists a substitution θ such that

$$\theta t_1 \equiv \theta t_2 \equiv \cdots \equiv \theta t_n$$

θ is called a **unifier** of $\{t_i \mid 1 \leq i \leq n\}$

Unification Examples:1

Example 28.4 Let f and g be distinct binary operators and $x, y, v, w \in V$.

1. The terms $f(x, y)$ and $f(v, w)$ may be unified by the substitution $\theta = \{x/v, y/w\}$ since $\theta f(v, w) \equiv f(x, y) \equiv \theta f(x, y)$. They may also be unified by $\theta^{-1} = \{v/x, w/y\}$.
2. Let r, s, t be any three terms. Then $f(x, y)$ and $f(v, w)$ may be unified by the substitution $\tau = \{g(s, t)/v, f(r, r)/w, g(s, t)/x, f(r, r)/y\}$.
3. The terms $f(x, y)$ and $f(y, x)$ may be unified by $\chi = \{x/y\}$ since $\chi f(x, y) \equiv f(x, x) \equiv \chi f(y, x)$.

Unification Examples:2

Example 28.5 Let f and g be distinct binary operators.

1. The terms $f(x, y)$ and $g(x, y)$ cannot be unified by any substitution.
2. The terms $f(x, y)$ and $f(y, x)$ cannot be unified by $\rho = \{x/y, y/x\}$ since $\rho f(x, y) \equiv f(y, x)$ and $\rho f(y, x) \equiv f(x, y)$. Hence $\rho f(x, y) \not\equiv \rho f(y, x)$.

The following facts are easy to prove and may be used without any mention of them in the sequel.

Fact 28.6 Let s and t be any two terms and let $p \in pos = pos(s) \cap pos(t)$. Then

1. If $s|_p \equiv t|_p$ for any position $p \in pos$ then for all positions $p, q \in pos$, $p \preceq q$ implies $s|_q \equiv t|_q$.
2. If $\text{rootsym}(s|_p) = o_1 \not\equiv o_2 = \text{rootsym}(t|_p)$ then s and t are not unifiable under any substitution.
3. If s and t are unifiable then for every position $p \in pos$, $\text{rootsym}(s|_p) \not\equiv \text{rootsym}(t|_p)$ implies at least one of the symbols is a variable i.e. $\{\text{rootsym}(s|_p), \text{rootsym}(t|_p)\} \cap V \neq \emptyset$

Exercise 28.1

1. Generalize the fact 28.6 to nonempty finite sets of terms.
2. Construct an example to show that the converse of fact 28.6.3 does not hold.

Generality of Unifiers

There is a certain sense in which θ may be regarded as being more general than τ in example 28.4.

Definition 28.7

- A substitution θ is at least as general as another substitution τ (denoted $\theta \gtrsim \tau$) if there exists a substitution χ such that $\tau = \chi \circ \theta$.
- $\theta \sim \tau$ if $\theta \gtrsim \tau \gtrsim \theta$.
- θ is strictly more general than τ (denoted $\theta \gtrless \tau$) if $\theta \gtrsim \tau$ and $\tau \not\gtrsim \theta$.

Generality: Facts

Fact 28.8

1. \gtrsim is a preordering relation on $\mathbf{S}_\Omega(V)$ i.e. it is a reflexive and transitive relation.
2. \gtrneq is an irreflexive and transitive relation on $\mathbf{S}_\Omega(V)$.
3. \sim is an equivalence relation on $\mathbf{S}_\Omega(V)$.

Most General Unifiers

Definition 28.9 Let $T = \{t_i \mid 1 \leq i \leq n\}$ be a unifiable set of terms. A substitution θ is called a **most general unifier (mgu)** of T if for each unifier τ of T , there exists a substitution ρ such that $\tau = \rho \circ \theta$.

Fact 28.10

1. If a set of terms T is unifiable then it has a mgu.
2. If θ and τ are both mgu's of a set T then $\theta \sim \tau$.
3. If θ and τ are both mgu's of a set T then there exist (pure-variable) substitutions $\rho, \rho^{-1} : V \rightarrow V$ such that $\rho \circ \theta = \tau$ and $\theta = \rho^{-1} \circ \tau$

More on Positions

For any nonempty set of terms T , we have

$$Pos(T) = \bigcap_{t \in T} \{pos(t)\} \neq \emptyset$$

and for any position $p \in Pos(T)$,

$$T|_p = \{t|_p \mid t \in T\}$$

and

$$rootsym(T|_p) = \{rootsym(t|_p) \mid t \in T\}$$

Disagreement Set

Definition 28.11 Given a set T ($|T| > 1$) of terms (also viewed as a set of abstract syntax trees), the **disagreement set** of T is defined as the set $T|_q$ of subterms rooted at some position q such that

1. not all the terms in $T|_q$ have the same root symbol and
2. for every $p \prec q$, $|\text{rootsym}(T|_p)| = 1$, where \prec is the proper-prefix relation on strings.

We have seen that $\text{pos}(\textcolor{violet}{t})$ for any term $\textcolor{violet}{t}$ is partially ordered by the relation \prec which inverts the proper sub-term ordering on $\text{ST}(\textcolor{violet}{t})$ (Fact 28.2). For the purpose of specifying the unification algorithm, it is useful to define a *total order* $<$ on the positions of terms which is consistent with \prec . Intuitively if $\textcolor{violet}{u} = o(t_1, t_2 \dots, t_n)$ we would like to specify *recursively* that

- the root position $\textcolor{violet}{u}$ precedes the root positions of all the subterms t_1, \dots, t_n . (which is taken care of by the prefix ordering \prec on positions) i.e. $\epsilon < i$ for all $1 \leq i \leq n$
- for each i, j such that $1 \leq i < j \leq n$, the position of the root of t_i precedes that of t_j in the total ordering.
- If $i < j$, then the position of the root of any proper subterm of t_i precedes the position of any subterm of t_j (including the root).

Definition 28.12 For any positions $p, q \in \text{pos}(\textcolor{violet}{t})$, $p < q$ iff one of the following conditions holds.

- $p = \epsilon \neq q$ or
- $(p = i.p', q = i.q' \text{ and } p' < q')$ or
- $(p = i.p', q = j.q' \text{ and } i < j)$.

If all operators in Ω are always used in prefix form then each term may also be regarded as a string in $(\Sigma \cup \{(,)\})^*$. The ordering $<$ on $pos(\textcolor{violet}{t})$ simply becomes the left-to-right ordering of symbols in the well-formed terms of $\mathcal{T}_\Omega(V)$ represented as strings.

Algorithm: Computing a Disagreement Set

Require: $|T| > 1$ {At least two different terms}

```
1: DISAGREEMENT( $T$ )  $\stackrel{df}{=}$  DISAGREE( $T, \epsilon, Pos(T) - \{\epsilon\}$ ) where
2:  $Pos(T) = \bigcap_{t \in T} Pos(t)$  {At least  $\epsilon \in Pos(T)$ } and
3: DISAGREE( $T, p, P$ )  $\stackrel{df}{=}$ 
4: if  $|rootsym(T|_p)| = 1$  then
5:   if  $P \neq \emptyset$  then
6:     let  $p' = Min(P); P' = P - \{p'\}$  in
7:     DISAGREE( $T, p', P'$ )
8:   end let
9: else {There is no disagreement}
10:  return fail
```

```
11: end if  
12: else {A disagreement has been found at position  $p$ }  
13:   return  $T|_p$   
14: end if
```

The function Min used in the algorithm above is the minimum position with respect to the total ordering $<$ on positions (definition 28.12) in a term.

Example: Disagreement 1

Example 28.13 Consider the set of terms

$$S_1 = \{f(a, x, h(g(z))), f(z, h(y), h(y))\}$$

where a is a constant, f is a ternary operator and g and h are unary operators. In this case, reading the terms from left to right we get a disagreement set $D_1 = \{a, z\}$. On the other hand, reading from right to left we obtain the disagreement set $D'_1 = \{g(z), y\}$ which requires going down one level deeper.

The algorithm however will compute the leftmost disagreement D_1 always.

Example: Occurs Check

Example 28.14 Consider the set

$$S_2 = \{f(g(z), x, h(g(z))), f(z, h(y), h(y))\}$$

The disagreement set $\{g(z), z\}$ is such that S_2 is not unifiable, because for any substitution θ of θz can never be syntactically identical with $\theta g(z)$. This is an example of the notorious occurs check problem. Hence S_2 is not unifiable.

Example: Disagreement 3

Example 28.15 Consider the set

$$S_3 = \{f(a, x, h(g(z))), f(b, h(y), h(y))\}$$

where a and b are both constant symbols. Here a disagreement set is $D_3 = \{a, b\}$. Again it is clear that S_3 is not unifiable.

Example: Disagreement 4

Example 28.16 Consider the set

$$S_4 = \{f(h(z), x, h(g(z))), f(g(x), h(y), h(y))\}$$

Here we have a disagreement set $D_4 = \{h(z), g(x)\}$. Since $h(z)$ cannot be unified with $g(x)$ under any substitution of free variables, S_4 is not unifiable.

Disagreement and Unifiability

Fact 28.17 If S' is the disagreement set of S then

1. S is unifiable implies S' is unifiable.
2. If S is unifiable and θ' is a mgu of S' then there exists a mgu θ of S such that $\theta' \gtrsim \theta$.

The above facts reduce the problem of finding a unifier if it exists, to that of systematically finding disagreement sets and unifying them.

Finding a unifier for a disagreement set is a pre-requisite for finding a unifier for the original set of terms. A disagreement set consists of subterms of the original set of terms at a particular position such that at least two distinct (sub-)terms exist in the set. Further a disagreement set is unifiable only if there is at most one non-variable term in it. By choosing a substitution $\{t/x\}$ where both t and x are terms in the disagreement set satisfying the condition $x \notin FV(t)$, there is a possibility of unifying the disagreement set. The **algorithm** constructs a sequence of singleton substitutions whose composition yields a most general unifier if it exists.

Algorithm: Unification

Require: $S \subseteq_f \mathcal{T}_\Omega(V)$ and $|S| > 1$

Ensure: If S is not unifiable then `fail` else $\exists \theta \in \mathbf{S}_\Omega(V) : |\theta S| = 1$ and θ is a mgu of S

1: $\text{UNIFY}(S) \stackrel{df}{=} \text{PARTIALUNIFY}(\mathbf{1}, S)$ **where**

2: $\text{PARTIALUNIFY}(\theta, S) \stackrel{df}{=}$

3: **let** $T = \theta S$ **in**

4: **if** $|T| = 1$ **then**

5: **return** $\theta \{ \theta \text{ is a mgu of } S \}$

6: **else** {There is a position at which at least two terms are different}

7: **let** $D = \text{DISAGREEMENT}(T)$ **in**

```

8:   if  $\exists x \in D \cap V : \forall t \in T [x \notin FV(t)]$  then
9:     Choose  $t \in T : x \notin FV(t)$ 
10:    PARTIALUNIFY( $\{t/x\} \circ \theta, S$ )
11:     $\{T = \theta S \wedge |\{t/x\}T| < |T|\}$ 
12:     $\{|(\{t/x\} \circ \theta)S| < |\theta S| \leq |S|\}$ 
13:    else {Occurs check fails so  $S$  is not unifiable}
14:      return fail
15:    end if
16:  end let
17: end if
18: end let

```

Example 28.18 Consider the set $S = S_1$ in example 28.13. Starting with $\theta_0 = \mathbf{I}$ we go through the following steps to obtain a unifier of S .

i	θ_i	$\theta_i S$	D_i
0	$\theta_0 = \mathbf{I}$	$\theta_0 S = \{f(a, x, h(g(z))), f(z, h(y), h(y))\}$	$D_0 = \{a, z\}$
1	$\theta_1 = \{a/z\} \circ \theta_0$	$\theta_1 S = \{f(a, x, h(g(z))), f(\textcolor{red}{a}, h(y), h(y))\}$	$D_1 = \{x, h(y)\}$
2	$\theta_2 = \{h(y)/x\} \circ \theta_1$	$\theta_2 S = \{f(a, \textcolor{red}{h}(y), h(g(z))), f(a, h(y), h(y))\}$	$D_2 = \{g(z), y\}$
3	$\theta_3 = \{g(z)/y\} \circ \theta_2$	$\theta_3 S = \{f(a, h(\textcolor{red}{g}(z)), h(\textcolor{red}{g}(z)))\}$	$D_3 = \emptyset$

Hence the required unifier is $\theta_3 = \{g(z)/y\} \circ \{h(y)/x\} \circ \{a/z\} \circ \mathbf{I} = \{a/z, h(g(z))/x, g(z)/y\}$.

Example 28.19 Let $S = \{f(y, z, w), f(g(x, x), g(y, y), g(z, z))\}$ where f is a ternary operator and g is a binary operator. An attempt to apply the algorithm yields the following sequence of substitutions: $\theta_1 = \{g(x, x)/y\}$ from which we get $\theta_1 S = \{f(\textcolor{red}{g}(x, x), z, w), f(g(x, x), g(\textcolor{red}{g}(x, x), g(x, x)), g(z, z))\}$ and then $\theta_2 = \{g(g(x, x), g(x, x))/z\}$ which yields

$\theta_2 S = \{f(g(x, x), \textcolor{red}{g}(g(x, x), g(x, x)), w), f(g(x, x), g(g(x, x), g(x, x)), g(\textcolor{red}{g}(g(x, x), g(x, x)), g(g(x, x), g(x, x))))\}$ and finally
 $\theta_3 = \{g(g(g(x, x), g(x, x)), g(g(x, x), g(x, x)))/w\} \circ \theta_2$

Hence in general there are pathological cases which make the algorithm very expensive to run, having a complexity that is exponential in the length of the input i.e. to unify the set

$$\{f(x_1, \dots, x_n), f(g(x_0, x_0), \dots, g(x_{n-1}, x_{n-1}))\}$$

would require a substitution that has $2^k - 1$ occurrences of the symbol g in the substitution of the variable x_k .

The Unification Theorem

Theorem 28.20 (The Unification Theorem) *The unification algorithm terminates satisfying its postcondition (If S is not unifiable then fail else $\exists \theta \in S_\Omega(V) : |\theta S| = 1$ and θ is a mgu of S) for any set of terms satisfying its preconditions ($S \subseteq_f T_\Omega(V)$ and $|S| > 1$).*



Proof of theorem 28.20

Proof:

Claim. Termination

Let $V_0 = \bigcup_{s \in S} FV(s)$ be the (finite) set of free variables occurring in S . With each execution of line 8 in the unification algorithm either it terminates because it fails or a substitution $\{t/x\}$ such that $x \notin FV(t)$ is generated with $\theta_{k+1} = \{t/x\} \circ \theta_k$, we have $\theta_{k+1}S$ has one variable less than θ_kS . Since V_0 is finite the algorithm must terminate. \dashv

Claim. If S is not unifiable then it terminates returning fail.

Trivial. \dashv

Claim. If S is unifiable then it terminates returning a most general unifier θ

Let ρ be any unifier of S , and let $\mathbf{1} = \theta_0, \theta_1, \dots, \theta_k = \theta$ be the sequence of substitutions generated by the algorithm. We prove by induction that for every θ_i , there exists a substitution τ_i such that $\rho = \tau_i \circ \theta_i$.

Basis. For $i = 0$ clearly $\tau_0 = \rho$.

Induction Hypothesis (*IH*).

Assume for some j , $0 \leq j < k$, there exists τ_j such that $\rho = \tau_j \circ \theta_j$.

Induction Step. Clearly since $\theta_j S$ is not a singleton, a disagreement set D_j will be found for $\theta_j S$. Since $\rho = \tau_j \circ \theta_j$ is a unifier of S , clearly τ_j must unify D_j , which means there exists a variable x and a term t with $x \notin FV(t)$ such that τ_j unifies D_j which in effect implies $\tau_j x = \tau_j t$. Without loss of generality we may assume $\{t/x\}$ is the chosen substitution so that $\theta_{j+1} = \{t/x\} \circ \theta_j$. Now define $\tau_{j+1} = \tau_j - \{\tau_j x/x\}$.

Case $x \in \text{dom}(\tau_j)$. Then $\tau_j = \{\tau_j x/x\} \cup \tau_{j+1} = \{\tau_j t/x\} \cup \tau_{j+1}$. Since $x \notin FV(t)$, we have $\tau_j = \{\tau_{j+1} t/x\} \cup \tau_{j+1} = \tau_{j+1} \circ \{t/x\}$ by the definition of composition. Finally from $\rho = \tau_j \circ \theta_j$ we get $\rho = \tau_{j+1} \circ \{t/x\} \circ \theta_j = \tau_{j+1} \circ \theta_{j+1}$.

Case $x \notin \text{dom}(\tau_j)$. Then $\tau_j = \tau_{j+1}$ and each element of D_j is a variable and $\tau_j = \tau_{j+1} \circ \{t/x\}$. Thus $\rho = \tau_j \circ \theta_j = \tau_{j+1} \circ \{t/x\} \circ \theta_j = \tau_{j+1} \circ \theta_{j+1}$ as required.

Since for any unifier ρ of S there exists τ_k such that $\rho = \tau_k \circ \theta_k$, θ_k must be a mgu of S . \dashv

QED

Exercise 28.2

1. Generalize the facts 28.6 to a set S of terms where $|S| \geq 2$.
2. Identify the relationships among the different substitutions θ , θ^{-1} , τ , χ , ρ and ρ^{-1} in examples 28.4 and 28.5
3. Let D be the disagreement set of S .
 - (a) Can $|D|$ be different from $|S|$? Justify your answer.
 - (b) If $S = D$ then under what conditions is S unifiable?
 - (c) If $S \neq D$ then what can you say about the depths of terms in D as compared to the depths of terms in S ?
4. Construct an example of a set S of terms with disagreement set D in which there exist a variable x and a term t such that $x \in FV(t)$ and yet the set S is unifiable.
5. Prove that if S is unifiable then the mgu computed by the **unification algorithm** is idempotent.

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

629 OF 783

[QUIT](#)

Lecture 29: Resolution in FOL

Friday 14 October 2011

1. Recapitulation
2. SCNFs and Models
3. SCNFs and Unsatisfiability
4. Representing SCNFs
5. Clauses: Terminology
6. Clauses: Ground Instances
7. Facts about Clauses
8. Clauses: Models
9. Clauses: Herbrand's Theorem
10. Resolution in FOL

Recapitulation

1. For any set $\Phi \cup \{\psi\}$ (where Φ may or may not be empty) of closed Σ -formulae $\Phi \models \psi$ iff $\Phi \cup \{\neg\psi\}$ is unsatisfiable.
2. A non-empty set Φ of closed Σ -formulae is unsatisfiable iff it contains a non-empty finite unsatisfiable subset.
3. A set Φ of closed Σ -formulae has a model iff it has a Herbrand model
4. A non-empty finite set Φ of closed Σ -formulae is unsatisfiable iff the formula $\psi \equiv \bigwedge_{\phi \in \Phi} \phi$ is unsatisfiable.

SCNFs and Models

1. A closed Σ -formula ψ has a model iff the universally closed Σ -formula $sko(\psi)$ has a model.
2. Every closed Σ -formula ψ may be transformed into an “*equisatisfiable*” (universally closed) formula in SCNF.
3. A universally closed Σ -formula ψ in SNF has model iff $\mathfrak{g}(\psi)$ the **ground-instances** (see definition 26.6) of ψ has a model.

SCNFs and Unsatisfiability

1. A closed Σ -formula ψ and the (universally closed) Σ -formula $sko(\psi)$ are “*equi-unsatisfiable*”.
2. A universally closed Σ -formula ψ in SNF is unsatisfiable iff $\mathfrak{s}(\psi)$ is unsatisfiable.
3. Since $\mathfrak{s}(\psi)$ consists of only closed formulae, $\mathfrak{s}(\psi)$ is unsatisfiable iff there is a finite subset of $\mathfrak{s}(\psi)$ which is unsatisfiable.

Representing SCNFs

Definition 29.1 Let the SCNF $\text{sko}(\psi)$ be represented by a set

$$\text{sko}(\psi) = \{C_i \mid 1 \leq i \leq m\}$$

such that

$$\text{sko}(\psi) \equiv \vec{\forall} [\bigwedge_{1 \leq i \leq m} C_i]$$

where each (quantifier-free) conjunct C_i , $1 \leq i \leq m$,

$$C_i \equiv \bigvee_{1 \leq j \leq n_i} \lambda_j$$

is called a clause and is represented by a set

$$C_i = \{\lambda_j \mid 1 \leq j \leq n_i\}$$

of literals.

Clauses: Terminology

Definition 29.2

1. A clause is a finite set of literals.
2. The empty clause is the empty set of literals ($\{\}$).
3. A ground clause is a clause with no occurrences of variables.
4. For any substitution θ , and clause $C = \{\lambda_j \mid 1 \leq j \leq n\}$,
 $\theta C = \{\theta \lambda_j \mid 1 \leq j \leq n\}$.

Compare with clauses in propositional logic

Clauses: Ground Instances

Definition 29.3

1. *The set of ground instances of a clause C is the set*

$$\mathfrak{g}(C) = \{\theta C \mid \theta \text{ is a ground substitution}\}$$

2. *For any set $S = \{C_i \mid 1 \leq i \leq m\}$ of clauses, the set of ground instances of S is the set*

$$\mathfrak{g}(S) = \bigcup_{C \in S} \mathfrak{g}(C)$$

Facts about Clauses

Lemma 29.4 Let $\{C_i \mid 1 \leq i \leq m\}$ be a set of clauses. Then

$$\vec{\forall}[\bigwedge_{1 \leq i \leq m} C_i] \Leftrightarrow \bigwedge_{1 \leq i \leq m} \vec{\forall}[C_i]$$

Proof: Follows from the semantics of \forall and \wedge or alternatively from corollary 25.2. QED ■

Notice that even if there are free variables common between two clauses, this lemma holds, mainly because of the fact that there are no existential quantifiers. For example

$$\forall x[p(x) \wedge q(x)] \Leftrightarrow \forall x[p(x)] \wedge \forall y[q(y)] \Leftrightarrow \forall x, y[p(x) \wedge q(y)]$$

Clauses: Models

Definition 29.5

1. A structure \mathbf{A} is a model of a

- clause $C = \{\lambda_j \mid 1 \leq j \leq n\}$ (denoted $\mathbf{A} \Vdash C$) iff $n > 0$ and $\mathbf{A} \Vdash \vec{\forall}[\bigvee_{1 \leq j \leq n} \lambda_j]$.
- a set S of clauses (denoted $\mathbf{A} \Vdash S$) if it is a model of every clause in S .

2. $S \models C$ iff every model of S is also a model of C .

Note: An empty clause has no models.

Clauses: Herbrand's Theorem

Proposition 29.6

1. A set S of clauses possesses a model iff every finite subset of S possesses a model.
2. A set S of clauses is unsatisfiable iff there is a finite subset $S' \subseteq_f S$ of clauses which is unsatisfiable iff $\mathfrak{g}(S')$ does not possess a model.



Resolution in FOL

Compare with resolution in propositional logic

Let S be a set of clauses, $C_i, C_j \in S$ with $i \neq j$, $FV(C_i) \cap FV(C_j) = \emptyset$ and p an atomic predicate symbol such that

- $C_i = C'_i \cup \{p(\vec{s}_{i'}) \mid 1 \leq i' \leq m_i\}$ and $C_j = C'_j \cup \{\neg p(\vec{t}_{j'}) \mid 1 \leq j' \leq m_j\}$
- $L = \{p(\vec{s}_{i'}) \mid 1 \leq i' \leq m_i\} \cup \{p(\vec{t}_{j'}) \mid 1 \leq j' \leq m_j\}$ is a set of unifiable literals.
- $\mu = \text{UNIFY}(L)$ is an mgu of L
- $C'_{ij} = \mu(C'_i \cup C'_j) = (\mu C'_i) \cup (\mu C'_j)$ is called the *resolvent* of C_i and C_j .

$$\text{Res1 } \frac{S}{(S - \{C_i, C_j\}) \cup \{C'_{ij}\}}$$

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

642 OF 783

QUIT

Lecture 30: More on Resolution in FOL

Tuesday 18 October 2011

1. Standardizing Variables Apart
2. Factoring
3. Example: 1
4. Example: 2
5. Refutation

Standardizing Variables Apart

1. The free variables of distinct clauses *must* be disjoint and variables should be renamed if necessary.

Example 30.1 Let $S = \{C_1, C_2\}$ where

$$C_1 = \{p(x)\}$$

$$C_2 = \{\neg p(f(x))\}$$

S represents the set of closed formulae

$$\{\forall x[p(x)], \forall x[\neg p(f(x))]\}$$

which is clearly unsatisfiable. However the empty clause cannot be derived (because of the *occurs check*) unless x is renamed in one of the clauses.

Factoring

- Unlike in the case of propositional resolution it should be possible to eliminate several literals at once

Example 30.2 Consider the set $S = \{C_1, C_2\}$ where

$$C_1 = \{p(x), p(y)\}$$

$$C_2 = \{\neg p(u), \neg p(v)\}$$

S is clearly unsatisfiable but by removing only one literal at a time with the substitution $\{x/u\}$ yields the new clause

$$C'_{12} = \{p(y), \neg p(v)\}$$

from which the empty clause cannot be derived.

Example: 1

Example 30.3 Consider the set $S = \{C_1, C_2\}$ where

$$\begin{aligned}C_1 &= \{\neg q(x, y), \neg q(y, z), q(x, z)\} \\&\equiv \forall x, y, z [(q(x, y) \wedge q(y, z)) \rightarrow q(x, z)]\end{aligned}$$

which represents transitivity and

$$\begin{aligned}C_2 &= \{\neg q(u, v), q(v, u)\} \\&\equiv \forall u, v [q(u, v) \rightarrow q(v, u)]\end{aligned}$$

which represents symmetry.

A logical consequence of these two properties is the property derived below.

$$\frac{C_1 = \{\neg q(\textcolor{violet}{x}, y), \underline{\neg q(y, z)}, q(x, z)\}, \{\neg q(u, v), \underline{q(v, u)}\} = C_2}{\{\neg q(\textcolor{violet}{x}, y), q(x, z), \neg q(z, y)\} = C'_{12}} \mu = \{z/u, y/v\}$$

$$\begin{aligned} C'_{12} &= \{\neg q(x, y), \neg q(z, y), q(x, z)\} \\ &\equiv \forall x, y, z [(q(x, y) \wedge q(z, y)) \rightarrow q(x, z)] \end{aligned}$$

Example: 2

Example 30.4 Suppose we need to prove that if a binary relation p is reflexive

$$\phi_{p\text{-reflexive}} \stackrel{df}{=} \forall x[p(x, x)]$$

and euclidean

$$\phi_{p\text{-euclidean}} \stackrel{df}{=} \forall x, y, z[(p(x, y) \wedge p(x, z)) \rightarrow p(y, z)]$$

then it is also symmetric

$$\phi_{p\text{-symmetric}} \stackrel{df}{=} \forall x, y[p(x, y) \rightarrow p(y, x)]$$

After renaming bound variables and converting into SCNF to get the clauses

$$C_1 = \{p(\underline{x}, \underline{x})\}$$

$$C_2 = \{\neg p(\underline{u}, v), \neg p(\underline{u}, w), p(v, w)\}$$

The mgu $\mu = \{x/u, x/w\}$ yields the required clause

$$C'_{12} = \{\neg p(x, v), p(v, x)\}$$

Refutation

Refutation in propositional logic

Example 30.5 We could also prove symmetry in example 30.4 by a *refutation* as follows. Taking the negation of the conclusion we get

$$\begin{aligned} & \neg \phi_{p\text{-symmetry}} \\ \Leftrightarrow & \exists u, v [p(u, v) \wedge \neg p(v, u)] \\ (\text{Sko}) \quad & p(a, b) \wedge \neg p(b, a) \\ \equiv & \{\{p(a, b)\}, \{\neg p(b, a)\}\} \\ \stackrel{df}{=} & \{C_3, C_4\} \end{aligned}$$

where a and b are skolem constants.

$$\frac{\{C_1 = \{p(x, x)\}, C_2 = \{\neg p(u, v), \neg p(u, w), p(v, w)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}}{\{C'_{12} = \{\neg p(x, v), p(v, x)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}} \mu = \{x/u, x/w\}$$

$$\frac{\{C'_{12} = \{\neg p(x, v), p(v, x)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}}{\{C'_{124} = \{\neg p(a, b)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}} \mu' = \{a/x, b/v\}$$

$$\frac{\{C'_{124} = \{\neg p(a, b)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}}{\{\{\}\}} \mu'' = \mathbf{1}$$

Alternatively a proof starting with the clauses C_3 and C_4 works too.

$$\frac{\{C_1 = \{p(x, x)\}, C_2 = \{\neg p(u, v), \neg p(u, w), p(v, w)\}, C_3 = \{p(a, b)\}, C_4 = \{\neg p(b, a)\}\}}{\{C_1 = \{p(x, x)\}, C_3 = \{p(a, b)\}, C'_{24} = \{\neg p(u, b), \neg p(u, a)\}\}} \theta_1 = \{b/v, a/w\}$$

$$\frac{\{C_1 = \{p(x, x)\}, C'_{24} = \{\neg p(u, b), \neg p(u, a)\}\}}{\{C_1 = \{p(x, x)\}, C''_{234} = \{\neg p(a, a)\}\}} \theta_2 = \{a/u\}$$

$$\frac{\{C_1 = \{p(x, x)\}, C''_{234} = \{\neg p(a, a)\}\}}{\{\{\}\}} \theta_3 = \{a/x\}$$

Exercise 30.1

1. Prove the conclusion of example 30.3 by a refutation.
2. Are there any other unifiers by which symmetry may be proved in example 30.4?

3. Try a refutation proof using $\nu = \{x/u, x/w, x/v\}$ as the first unifier in example 30.5. Why doesn't it work?
4. Try a refutation proof using $\nu = \{x/u, x/w, y/v\}$ as the first unifier in example 30.5. Why does it work?

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

654 OF 783

[QUIT](#)

Lecture 31: Resolution: Soundness and Completeness

Wednesday 19 October 2011

1. Soundness of FOL Resolution
2. Ground Clauses
3. The Lifting Lemma
4. Lifting Lemma: Figure
5. Completeness of Resolution Refutation: 1
6. Completeness of Resolution Refutation: 2
7. Completeness of Resolution Refutation: 3

Soundness of FOL Resolution

Lemma 31.1 *The resolvent C'_{ij} obtained by resolving the clauses C_i and C_j in the **resolution method** is a logical consequence of the set $\{C_i, C_j\}$.*



The following theorem then follows.

Theorem 31.2 *If S' is the set of clauses obtained by a single application of the resolution rule **Res1**, then $S \models S'$.*



Corollary 31.3 *If the empty clause is derivable from a set S of clauses, then S is unsatisfiable.*



Proof of lemma 31.1

Proof: Assume

- $C_i = C'_i \cup \{p(\vec{s}_{i'}) \mid 1 \leq i' \leq m_i\}$,
- $C_j = C'_j \cup \{\neg p(\vec{t}_{j'}) \mid 1 \leq j' \leq m_j\}$,
- $FV(C_i) \cap FV(C_j) = \emptyset$ and
- $L = \{p(\vec{s}_{i'}) \mid 1 \leq i' \leq m_i\} \cup \{p(\vec{t}_{j'}) \mid 1 \leq j' \leq m_j\}$ is a set of unifiable literals.

Let $\mathbf{A} \Vdash \{C_i, C_j\}$. Therefore

$$\mathbf{A} \Vdash \vec{\forall}[\bigvee_{\lambda_i \in C_i} \lambda_i] \quad (21)$$

$$\mathbf{A} \Vdash \vec{\forall}[\bigvee_{\lambda_i \in C_j} \lambda_j] \quad (22)$$

and for any substitution θ we have

$$\mathbf{A} \Vdash \theta \bigvee C_i \quad (23)$$

$$\mathbf{A} \Vdash \theta \bigvee C_j \quad (24)$$

If θ is a unifier of L and $\theta L = \{\lambda\}$ we get

$$\mathbf{A} \Vdash \bigvee (\{\lambda\} \cup \theta C'_i) \quad (25)$$

$$\mathbf{A} \Vdash \bigvee (\{\bar{\lambda}\} \cup \theta C'_j) \quad (26)$$

Let $\theta C'_i = \{\kappa_{i'} \mid 1 \leq i' \leq k\}$ and $\theta C'_j = \{\lambda_{j'} \mid 1 \leq j' \leq l\}$. Then we have the following table which shows a case analysis for the various values of k and l .

	$\{\lambda\} \cup \theta C'_i \Leftrightarrow$	$\{\bar{\lambda}\} \cup \theta C'_j \Leftrightarrow$	$\theta C'_i \cup \theta C'_j \Leftrightarrow$
$k = 0 = l$	λ	$\bar{\lambda}$	$\{\}$
$k = 0, l > 0$	λ	$\lambda \rightarrow (\lambda_1 \vee \dots \vee \lambda_l)$	$\lambda_1 \vee \dots \vee \lambda_l$
$k > 0, l = 0$	$\bar{\lambda} \rightarrow (\kappa_1 \vee \dots \vee \kappa_k)$	$\bar{\lambda}$	$\kappa_1 \vee \dots \vee \kappa_k$
$k, l > 0$	$\neg(\kappa_1 \vee \dots \vee \kappa_k) \rightarrow \lambda$	$\lambda \rightarrow (\lambda_1 \vee \dots \vee \lambda_l)$	$\neg(\kappa_1 \vee \dots \vee \kappa_k) \rightarrow (\lambda_1 \vee \dots \vee \lambda_l)$

It is easy to see that in each case

$$\{\{\lambda\} \cup \theta C'_i, \{\bar{\lambda}\} \cup \theta C'_j\} \models \theta C'_i \cup \theta C'_j \quad (27)$$

It follows also from (21), (22) and (27) that $\mathbf{A} \Vdash \vec{\forall}[\bigvee C'_{ij}]$ and hence C'_{ij} is a logical consequence of the set $\{C_i, C_j\}$. QED ■

Ground Clauses

Theorem 31.4 (Completeness of Resolution Refutation for ground clauses). *Let G be a set of ground clauses. If G does not possess a model, the empty clause ($\{\}$) may be derived by Res0.*



Here Res0 is the propositional resolution rule given by

$$\text{Res0} \quad \frac{S}{(S - \{C_i, C_j\}) \cup \{C'_{ij}\}}$$

where $C'_{ij} = C'_i \cup C'_j$. Note that there is no substitution involved anywhere since all clauses are ground.

Proof of theorem 31.4.

Proof: Let $G = \{C_i \mid 1 \leq i \leq n, n > 0\}$ be a set of n clauses. If $\{\} \in G$ there is nothing to prove. So assume $\{\} \notin G$. Consider the following measure

$$\#G = (\sum_{1 \leq i \leq n} |C_i|) - n$$

Clearly, $\#G = 0$ iff every clause is made up of a single literal. We proceed to prove the theorem by induction on $\#G$.

Basis. $\#G = 0$. Then each $C_i = \{\lambda_i\}$ and $G \equiv \bigwedge_{1 \leq i \leq k} \lambda_i$ and by theorem 25.10, G is unsatisfiable iff it contains a complementary pair. Clearly by rule Res0 the resolvent of this complementary pair is the empty clause.

Induction Hypothesis (IH).

For all k , $0 \leq \#G = k < m$ for some $m > 0$, if G does not possess a model, then the empty clause is derivable from G .

Induction Step. Assume $\#G = m > 0$. There must be at least one clause C_i which contains more than one literal. So let $C_i = \{\lambda_i\} \cup D_i$ with $\lambda_i \notin D_i \neq \emptyset$. Let $G_{i_1} = (G - \{C_i\}) \cup \{D_i\}$ and $G_{i_2} = (G - \{C_i\}) \cup \{\lambda_i\}$. Clearly $\#G_{i_1} < \#G$ and $\#G_{i_2} < \#G$. Further if G does not have a model then neither G_{i_1} nor G_{i_2} has a model (if either of them had a model then so would G since $C_i \equiv \lambda_i \vee \bigvee D_i$). By the induction hypothesis,

1. there exists a resolution proof \mathcal{R}_1 from G_{i_1} which derives the empty clause and
2. there is another resolution proof \mathcal{R}_2 from G_{i_2} which also derives the empty clause.

Notice that since we are dealing only with ground literals, all resolvents are obtained by applying the rule **Res0**.

Consider the proof \mathcal{R}'_1 obtained from \mathcal{R}_1 by adding the literal λ_i to G_{i_1} and performing exactly the same sequence of resolutions.

- Case 1. If the proof \mathcal{R}_1 did not involve the use of any of the literals from D_i and the empty clause was derived, then clearly the same sequence with λ_i included would also derive the empty clause and that completes the proof.
- Case 2 On the other hand if one or more steps in proof \mathcal{R}_1 involved literals from D_i then the resulting proof \mathcal{R}'_1 may derive the clause $\{\lambda_i\}$ in place of the empty clause. However we do know that the empty clause is derived from G_{i_2} in proof \mathcal{R}_2 . This implies there exist

resolution steps in \mathcal{R}_2 involving the literal λ_i which derive the empty clause. Therefore there exists at least one clause containing the literal $\bar{\lambda}_i$ in the set of final clauses obtained in \mathcal{R}'_1 . By applying the resolution steps of \mathcal{R}_2 which do not appear anywhere in \mathcal{R}'_1 , the empty clause would again be derived.

QED



The Lifting Lemma

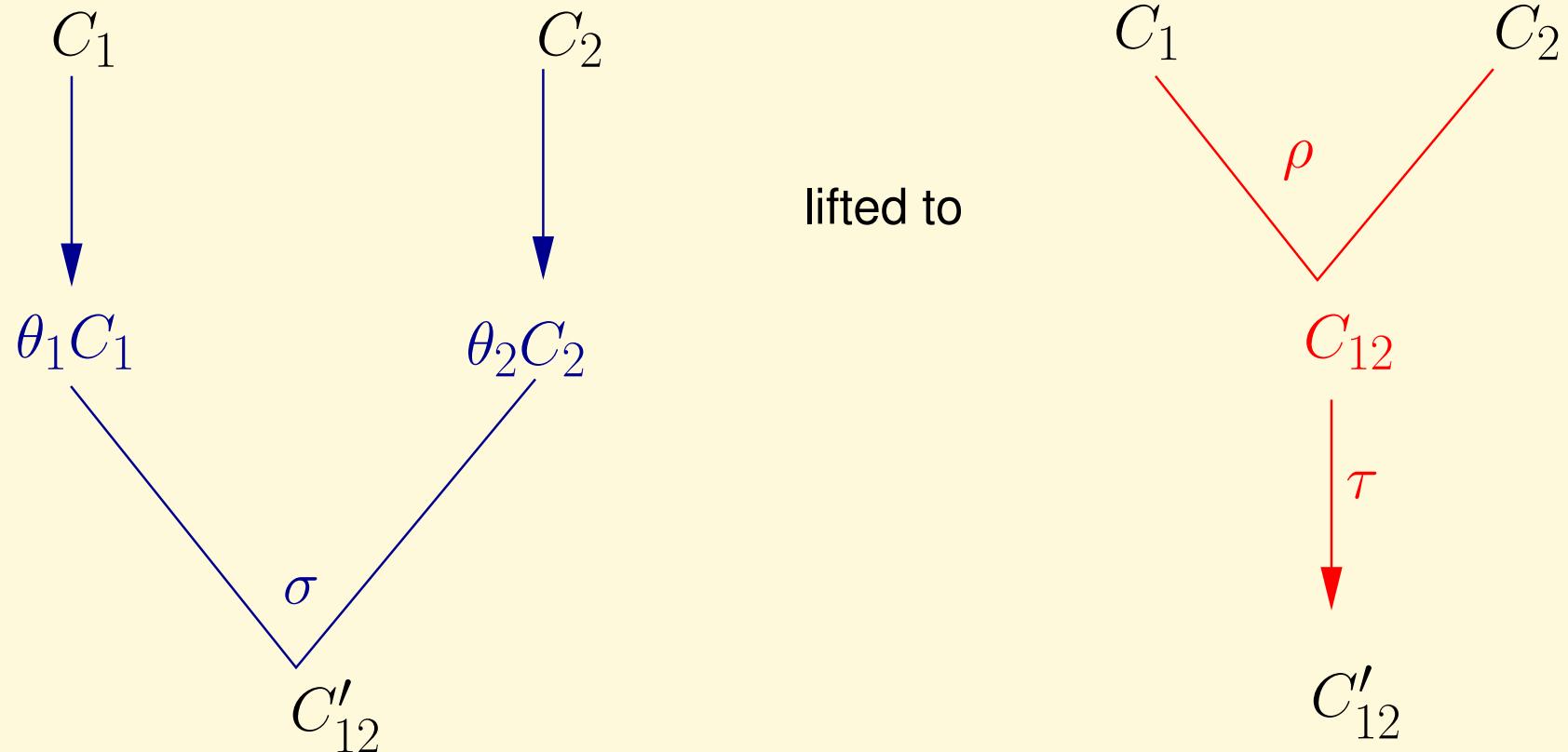
Lemma 31.5 (Lifting Lemma). (see *figure*) Let C_1 and C_2 be clauses and let $\theta_1, \theta_2, \sigma$ be substitutions such that

- $FV(C_1) \cap FV(C_2) = \emptyset$,
- $FV(\theta_1 C_1) \cap FV(\theta_2 C_2) = \emptyset$ and
- C'_{12} is the resolvent of $\theta_1 C_1$ and $\theta_2 C_2$ via a substitution σ , by a single application of resolution.

Then there exists a resolvent C_{12} of C_1 and C_2 by a single application of resolution via a substitution ρ and a substitution τ such that $C'_{12} \equiv \tau C_{12}$.



Lifting Lemma: Figure



Proof of lemma 31.5

Proof: Let

$$C_1 = C'_1 \cup L_1 \text{ where } L_1 = \{\lambda_i \mid 1 \leq i \leq m, m > 0\}$$

$$C_2 = C'_2 \cup L_2 \text{ where } L_2 = \{\overline{\lambda_j} \mid 1 \leq j \leq n, n > 0\}$$

such that σ is a mgu of $(\theta_1 L_1) \cup (\theta_2 \overline{L_2})$ and $C'_{12} = \sigma((\theta_1 C'_1) \cup (\theta_2 C'_2))$.

Since $FV(C_1) \cap FV(C_2) = \emptyset$, $dom(\theta_1) \cap dom(\theta_2) = \emptyset$ and since $FV(\theta_1 C_1) \cap FV(\theta_2 C_2) = \emptyset$ we have $FV(ran(\theta_1)) \cap FV(ran(\theta_2)) = \emptyset$ and hence $\theta_1 C'_1 = (\theta_1 \cup \theta_2) C'_1$ and $\theta_2 C'_2 = (\theta_1 \cup \theta_2) C'_2$.

Since σ is a mgu of $(\theta_1 L_1) \cup (\theta_2 \overline{L_2})$ we have $\sigma \circ (\theta_1 \cup \theta_2)$ is a unifier of $L_1 \cup \overline{L_2}$. If $L_1 \cup \overline{L_2}$ is unifiable, then it has a most general unifier $\rho \gtrsim \sigma \circ (\theta_1 \cup \theta_2)$ such that $C'_{12} = \rho(C'_1 \cup C'_2)$ is the resolvent of C_1 and C_2 .

$\rho \gtrsim \sigma \circ (\theta_1 \cup \theta_2)$ implies there exists a substitution τ such that

$$\tau \circ \rho = \sigma \circ (\theta_1 \cup \theta_2)$$

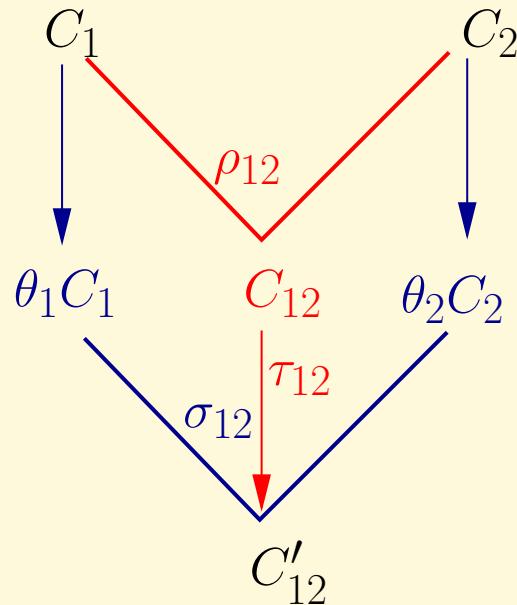
and

$$C'_{12} = \tau(\rho(C'_1 \cup C'_2)) = (\sigma \circ (\theta_1 \cup \theta_2))(C'_1 \cup C'_2)$$

QED



A superposed version of the figure is shown below.



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 (lemma **31.5**) 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**.

Completeness of Resolution Refutation: 2

1. The **lifting lemma 31.5** guarantees that if the substitutions θ_1 and θ_2 are *ground*, then there exists a corresponding *ground substitution* τ which produces the same effect after resolution.
2. By Herbrand's theorem **26.7** a set Φ is unsatisfiable iff a finite subset of ground instances of Φ is unsatisfiable.
3. To prove the completeness of resolution refutation it is sufficient to consider only the *finite set of ground clauses* from which the empty clause $\{\}$ may be derived.

Completeness of Resolution Refutation: 3

Theorem 31.6 (Completeness of Resolution Refutation). *If a set Φ of clauses is unsatisfiable then the empty clause is derivable from Φ .*

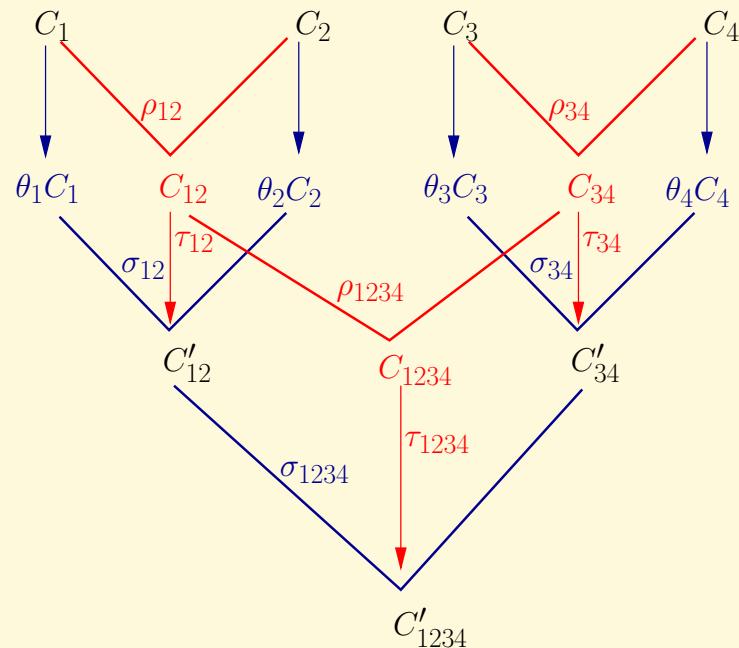


Proof of theorem 31.6

Proof: Without loss of generality we may assume that the variables in every clause are disjoint from the variables occurring in any other clause.

By Herbrand's theorem 26.7 there exists a finite set of ground clauses $G = \{gC_i \mid 1 \leq i \leq m\} \subseteq_f g(\Phi)$ such that G is unsatisfiable. Each $gC_i \in G$ is obtained by a substitution on some clause i.e. $gC_i = \theta_i C_i$ for some substitution θ_i and some clause $C_i \in \Phi$. Further for $i \neq j$, $dom(\theta_i) \cap dom(\theta_j) = \emptyset$ and since all the clauses in G are ground the disjointness conditions of the lifting lemma are trivially satisfied.

Each application of rule Res0 in G may be lifted to finding a mgu of the appropriate clauses. This fact may be proved by induction on the height of the resolution proof tree and we leave it as an exercise to the interested reader. However the following diagram illustrates it for two steps of a resolution proof tree.



QED

■

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

673 OF 783

QUIT

Lecture 32: Resolution and Tableaux

Tuesday 01 November 2011

1. FOL: Tableaux
2. FOL: Tableaux Rules
3. FOL Tableaux: Example 1
4. First-Order Tableaux
5. FOL Tableaux: Example 2

FOL: Tableaux

1. The tableau method in principle is similar to
 - natural deduction in its use of syntactical decomposition,
 - resolution in using unsatisfiability to prove validity.
2. The tableau method for both propositional and predicate logic has some advantages over resolution.
3. FOL resolution requires formulae to be converted into PCNF and then SCNF before resolution may be applied.
4. As in the case of Natural Deduction the tableau method uses the rules $\exists E$ and $\forall E$ to decompose quantified formulae along with the same restrictions.

FOL: Tableaux Rules

Besides the usual **tableau rules** for the propositional connectives we have the following.

$\forall.$	$\frac{\forall x[\phi]}{\{t/x\}\phi}$	$\neg\forall.$	$\frac{\neg\forall x[\phi]}{\neg\{a/x\}\phi}$
$\exists.$	$\frac{\exists x[\phi]}{\{a/x\}\phi}$	$\neg\exists.$	$\frac{\neg\exists x[\phi]}{\neg\{t/x\}\phi}$

1. The restrictions on the use of a constant symbol a in both rules $\neg\forall.$ and $\exists.$ are the same as those for $\exists E$ in both \mathcal{H}_1 and \mathcal{G}_1 .
2. Since $\forall E$ holds for all terms t , the rules $\forall.$ and $\neg\exists.$ may have to be applied several times before unsatisfiability can be proven.

FOL Tableaux: Example 1

Example 32.1 Let c be a constant symbol and f a unary function symbol. Then $\Phi = \{\neg p(c), p(f(f(c))), \forall x[p(x) \vee \neg p(f(x))]\}$ is unsatisfiable.

	$\neg p(c)$
	$p(f(f(c)))$
	$\forall x[p(x) \vee \neg p(f(x))]$
	$p(c) \vee \neg p(f(c))$
$p(c)$	$\neg p(f(c))$
■	$p(f(c)) \vee \neg p(f(f(c)))$
	$p(f(c))$
	$\neg p(f(f(c)))$
	■
	■

Notice the **two applications** of rule $\forall E$. ■ indicates a closed path in the tableau.

First-Order Tableaux

1. Unlike propositional tableaux, any satisfiable set Φ of quantified formulae can potentially yield an *infinite* tableau, since a formula of the form $\forall x[\phi] \in \Phi$ can have an infinite number of instances.
2. For unsatisfiable sets, closed finite tableaux may be constructed by applying the following heuristics
 - Whenever possible apply propositional rules before applying quantifier rules
 - Apply rules $\exists.$ and $\neg\forall.$ before applying $\forall.$ and $\neg\exists.$ in order to direct the proof towards a propositional contradiction.

FOL Tableaux: Example 2

We prove that $\forall x[p(x) \rightarrow q(x)] \models \forall x[p(x)] \rightarrow \forall x[q(x)]$

1.	$\forall x[p(x) \rightarrow q(x)]$	
2.	$\neg(\forall x[p(x)] \rightarrow \forall x[q(x)])$	
3.	$\forall x[p(x)]$	($\neg \rightarrow$ on 2.)
4.	$\neg\forall x[q(x)]$	($\neg \rightarrow$ on 2.)
5.	$\neg q(a)$	($\neg \forall$ on 4.)
6.	$p(a)$	(\forall on 3.)
7.	$p(a) \rightarrow q(a)$	(\forall on 1.)
8.	$\neg p(a)$ ($\neg \rightarrow$ on 7.)	9. $q(a)$ ($\neg \rightarrow$ on 7.)
10.	■ (6., 8.)	11. ■ (5., 9.)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

681 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

682 OF 783

QUIT

Lecture 33: Completeness of Tableaux Method

Wednesday 02 November 2011

1. First-order Hintikka Sets
2. Hintikka's Lemma for FOL
3. First-order tableaux and Hintikka sets
4. Soundness of First-order Tableaux
5. Completeness of First-order Tableaux

First-order Hintikka Sets

Definition 33.1 A finite or infinite set Γ is a **first-order Hintikka set** with respect to $\mathcal{P}_1(\Sigma)$ if

- 1-3. Γ is a (propositional) Hintikka set (definition 9.7) such that all the atomic propositions are ground.
- 4.
 - $\forall x[\phi] \in \Gamma$ implies $\{t/x\}\phi \in \Gamma$,
 - $\neg\exists x[\phi] \in \Gamma$ implies $\neg\{t/x\}\phi \in \Gamma$
for every $t \in T_0(\Sigma)$ (ground term).
- 5.
 - $\exists x[\phi] \in \Gamma$ implies $\{t/x\}\phi \in \Gamma$,
 - $\neg\forall x[\phi] \in \Gamma$ implies $\neg\{t/x\}\phi \in \Gamma$
for at least one $t \in T_0(\Sigma)$ (ground term).

Hintikka's Lemma for FOL

Lemma 33.2 *If Σ contains at least one constant symbol, then every first-order Hintikka set with respect to $\mathcal{P}_1(\Sigma)$ is satisfiable in a Herbrand model.*

Proof: We define a Herbrand interpretation of the formulae as follows. For each n -ary atomic predicate symbol p , $p(t_1, \dots, t_n)$ for ground terms t_1, \dots, t_n is true if and only if $p(t_1, \dots, t_n) \in \Gamma$. By the definition of a Hintikka set we know $\{p(t_1, \dots, t_n), \neg p(t_1, \dots, t_n)\} \not\subseteq \Gamma$. Hence all the atomic sentences in Γ are satisfiable under any valuation $v_{\mathbf{H}}$. We may then proceed to show by structural induction on each $\phi \in \mathcal{P}_1(\Sigma)$ that $\phi \in \Gamma$ implies $\mathbf{H} \Vdash \phi$. QED ■

First-order tableaux and Hintikka sets

Lemma 33.3 *If a tableau rooted at a closed formula ϕ has an open path then the set of formulae on the path form a first-order Hintikka set.*

Proof: We may prove that each rule in **Tableaux Rules** and **FOL: Tableaux Rules** creates a path for the construction of Hintikka sets. QED ■

Soundness of First-order Tableaux

Theorem 33.4 (Soundness of First-order Tableau Rules). *If there is a closed tableau rooted at a closed formula $\neg\phi$ then $\models \phi$.*

Proof: Suppose there is a closed tableau rooted at $\neg\phi$. Then every path in the tableau is closed because of the occurrence of a complementary pair in the path. On the other hand if $\not\models \phi$, i.e. ϕ is not valid, then $\neg\phi$ is satisfiable, which implies that there is an open path in the tableau rooted at $\neg\phi$, clearly a contradiction. QED ■

Completeness of First-order Tableaux

Theorem 33.5 (Completeness of First-order Tableaux). *If a closed formula ϕ is valid, then there exists a closed tableau rooted at $\neg\phi$.*

Proof: If there is no closed tableau rooted at $\neg\phi$ then there exists at least one *open* path in each such tableau. The set of formulae on this path form a Hintikka set and hence they are all simultaneously satisfiable, which implies there is a Herbrand model satisfying $\neg\phi$, in which case $\not\models \phi$. QED

[HOME PAGE](#)



LCS

[Go BACK](#)

[FULL SCREEN](#)

[CLOSE](#)

690 OF 783

[QUIT](#)

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

691 OF 783

QUIT

Lecture 34: Completeness of the Hilbert System

Wednesday 09 November 2011

1. Deductive Consistency
2. Models of Deductively Consistent Sets
3. Deductive Completeness
4. The Completeness Theorem

Deductive Consistency

Definition 34.1 A set $\Phi \subseteq \mathcal{L}_1(\Sigma)$ is deductively consistent iff there does not exist a formula ϕ such that $\Phi \vdash_{\mathcal{H}_1} \phi$ and $\Phi \vdash_{\mathcal{H}_1} \neg\phi$.

This definition is equivalent to other possible definitions such as those given below which may all be derived from rule \perp .

Lemma 34.2 *The following statements are equivalent.*

1. $\Phi \subseteq \mathcal{L}_1(\Sigma)$ is deductively consistent.
2. There does not exist a formula ψ such that $\Phi \vdash_{\mathcal{H}_1} \neg(\psi \rightarrow \psi)$
3. There exists a formula which is not provable.



34.1. Model-theoretic and Proof-theoretic Consistency

We have earlier defined the notion of consistency of a set of formulae in **propositional logic** (see also lemma 11.1) leading to the notions of **maximal consistency** and **Lindenbaum's theorem** which enabled us to extend a consistent set of propositions to a maximally consistent one. Later we have also defined the notion of consistency of sets of predicate logic formulae in definition 20.2. Notice that definition 9.1 though worded differently, also states that a set of propositions is consistent only if it has a model. The notion of a model in sentential logic however, refers to the existence of a truth assignment under which all the sentences are true (simultaneously). Hence both in sentential and predicate logic the notion of consistency refers to the existence of a model. These notions of consistency are **model-theoretic** since they are intimately associated with the existence of a model.

We have reserved the term “**deductive consistency**” (definition 34.1) to a **proof-theoretic** notion obtained from deductions rather than models. *A priori* there is no reason to believe that the two notions are equivalent unless we can prove that our deductive system is sound and complete. While **soundness** has been proven we need to prove completeness before claiming that the model-theoretic notion of consistency and the proof-theoretic one are equivalent.

We need to carry our analogies between model-theory and proof theory a little further to the domain

maximal consistent sets (indeed some of the proof ideas would be analogous too!) in order to be able to prove the completeness of the system \mathcal{H}_1 . We refer to such maximally consistent sets obtained through deductions as being *deductively complete*. The main difference however, is that we restrict ourselves to only closed formulae as will be evident soon.

Models of Deductively Consistent Sets

Lemma 34.3 *If $\Phi \subseteq \mathcal{L}_1(\Sigma)$ has a model then it is deductively consistent.*

Proof: Let $\mathbf{A} \Vdash \psi$ for each $\psi \in \Phi$. If Φ is not deductively consistent, then there exists ϕ such that $\Phi \vdash_{\mathcal{H}_1} \phi$ and $\Phi \vdash_{\mathcal{H}_1} \neg\phi$. However since \mathcal{H}_1 is **sound** it follows that $\mathbf{A} \Vdash \phi$ and $\mathbf{A} \Vdash \neg\phi$ which is a contradiction. QED ■

Deductive Completeness

Lemma 34.4 For any $\Phi \subseteq \mathcal{L}_1(\Sigma)$, $\Phi \vdash_{\mathcal{H}_1} \phi$ iff $\Phi \vdash_{\mathcal{H}_1} \forall[\phi]$ iff $\Phi \cup \{\neg\forall[\phi]\}$ is not deductively consistent.

□

We restrict our attention to only deductively consistent and complete sets.

Definition 34.5 A (deductively consistent) set $\Phi \subseteq \mathcal{L}_1(\Sigma)$ is deductively complete iff for every closed formula ϕ , $\Phi \vdash_{\mathcal{H}_1} \phi$ or $\Phi \vdash_{\mathcal{H}_1} \neg\phi$.

Proof of lemma 34.4

Proof:

- $\Phi \vdash_{\mathcal{H}_1} \phi$ iff $\Phi \vdash_{\mathcal{H}_1} \forall[\phi]$ is obvious from rules $\forall I$ and $\forall E$.
- (\Rightarrow) Suppose $\Phi \vdash_{\mathcal{H}_1} \phi$. Then by monotonicity (theorem 13.1) $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \phi$ and hence $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \forall[\phi]$. Further since $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \neg\forall[\phi]$, it follows that $\Phi \cup \{\neg\forall[\phi]\}$ is not deductively consistent.
(\Leftarrow) Suppose $\Phi \cup \{\neg\forall[\phi]\}$ is not deductively consistent. Then there exists a formula (by lemma 34.2) ψ such that $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \neg(\psi \rightarrow \psi)$. From $\vdash_{\mathcal{H}_1} \psi \rightarrow \psi$ and \perp we obtain $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \forall[\phi]$. By corollary 23.4 (Deduction theorem for closed formulae) we obtain $\Phi \vdash_{\mathcal{H}_1} \neg\forall[\phi] \rightarrow \forall[\phi]$. It follows from the derived axiom C2 that $\Phi \vdash_{\mathcal{H}_1} (\neg\forall[\phi] \rightarrow \forall[\phi]) \rightarrow \forall[\phi]$ from which we obtain $\Phi \vdash_{\mathcal{H}_1} \forall[\phi]$ by a single application of MP.

QED



Notes on proof of lemma 34.4.

1. The universal closure is required in the lemma, because in general the inconsistency of $\Phi \cup \{\neg\phi\}$ does not imply $\Phi \vdash_{\mathcal{H}_1} \phi$.

Example 34.6 Let $\Phi = \{\neg\forall x[\neg p(x)]\}$ and $\phi \equiv p(x)$. Then $\Phi \cup \{\neg p(x)\}$ is inconsistent. However, it is not possible to prove $\Phi \vdash_{\mathcal{H}_1} p(x)$.

2. Hence the maximally consistent sets of propositional logic translate into deductively complete sets in FOL. And this maximal completeness can only be shown for closed formulae and not for arbitrary formulae with free variables.
3. Clearly deductive completeness therefore is restricted to closed formulae.

The following is the proof-theoretic analogue of [Lindenbaums theorem](#). Even the proof of the theorem mirrors the [alternative proof](#) of Lindenbaum's theorem.

Theorem 34.7 (The Extension Theorem) *Every deductively consistent set may be extended to a deductively complete set.*

Proof: Let Φ be a nonempty deductively consistent set of Σ -formulae. For any enumeration of [closed](#) Σ -formulae

$$\psi_1, \psi_2, \psi_3, \dots \quad (28)$$

define the chain of sets $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \dots$ starting with $\Phi_0 = \Phi$ as follows:

$$\Phi_{i+1} = \begin{cases} \Phi_i & \text{if } \Phi_i \vdash_{\mathcal{H}_1} \psi_i \\ \Phi_i \cup \{\neg\psi_i\} & \text{otherwise} \end{cases}$$

Claim. Each Φ_i is deductively consistent.

⊤ By induction on i . For $i = 0$, $\Phi_0 = \Phi$ is given to be deductively consistent. Assuming Φ_i is deductively consistent, we have that if $\Phi_{i+1} = \Phi$ it is obviously deductively consistent. Otherwise $\Phi_{i+1} = \Phi \cup \{\neg\psi_i\}$ and by lemma 34.4 since $\Phi_i \not\vdash_{\mathcal{H}_1} \psi_i$ and ψ is a closed formula, Φ_{i+1} must be deductively consistent. ⊢

Then $\Phi_\infty = \bigcup_{i \geq 0} \Phi_i$ is the desired set.

Claim. Φ_∞ is deductively consistent.

⊤ Suppose not. Then by lemma 34.2, for some formula ϕ , we have $\Phi_\infty \vdash_{\mathcal{H}_1} \neg(\phi \rightarrow \phi)$. However since such a proof is finite there exists a finite subset $\Psi \subseteq_f \Phi$, such that $\Psi \vdash_{\mathcal{H}_1} \neg(\phi \rightarrow \phi)$. Since Ψ is finite there exists a $k \geq 0$ such that $\Psi \subseteq \Phi_k$ which implies $\Phi_k \vdash_{\mathcal{H}_1} \neg(\phi \rightarrow \phi)$ contradicting the previous claim that Φ_k is deductively consistent. ⊢

Claim. Φ_∞ is deductively complete.

⊤ For any arbitrary closed formula ψ , ψ occurs in the enumeration 28 at some position, say $\psi \equiv \psi_m$ for some $m \geq 0$. If $\Phi_m \vdash_{\mathcal{H}_1} \psi_m$ then $\Phi_\infty \vdash_{\mathcal{H}_1} \psi_m$. Otherwise $\Phi_{m+1} = \Phi_m \cup \{\neg\psi_m\}$ and $\Phi_\infty \vdash_{\mathcal{H}_1} \neg\psi_m$. By definition 34.5 Φ_∞ is deductively complete. ⊤

The Completeness Theorem

Theorem 34.8 (Gödel's Completeness Theorem). $\Phi \models \phi$ implies $\Phi \vdash_{\mathcal{H}_1} \phi$. When $\Phi = \emptyset$ we have that all valid formulae of $\mathcal{L}_1(\Sigma)$ are theorems of \mathcal{H}_1 .

QED



Proof: Assume $\Phi \models \phi$ and suppose $\Phi \not\vdash_{\mathcal{H}_1} \phi$. Then by lemma 34.4 $\Phi \cup \{\neg \forall[\phi]\}$ is deductively consistent and hence possesses a model \mathbf{A} . But that implies $\mathbf{A} \Vdash \Phi$ but $\mathbf{A} \not\Vdash \phi$, which by definition means $\Phi \not\models \phi$, a contradiction. QED



HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

704 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

705 OF 783

QUIT

Lecture 35: First-Order Theories

Friday 11 November 2011

1. (Simple) Directed Graphs
2. (Simple) Undirected Graphs
3. Irreflexive Partial Orderings
4. Irreflexive Linear Orderings
5. (Reflexive) Preorders
6. (Reflexive) Partial Orderings
7. (Reflexive) Linear Orderings
8. Equivalence Relations
9. Peano's Postulates
10. The Theory of The Naturals
11. Notes and Explanations
12. Finite Models of Arithmetic
13. A Non-standard Model of Arithmetic
14. Z-Chains

(Simple) Directed Graphs

$$\Sigma = \{; e : s^2\}$$

$$\phi_{e-\text{irreflexivity}} \stackrel{df}{=} \forall x [\neg e(x, x)]$$

$$\Phi_{DG} \stackrel{df}{=} \{\phi_{e-\text{irreflexivity}}\}$$

(Simple) Undirected Graphs

$$\Sigma$$

$$= \{; e : s^2\}$$

$$\phi_{e-\text{irreflexivity}}$$

$$\stackrel{df}{=} \forall x[\neg e(x, x)]$$

$$\phi_{e-\text{symmetry}}$$

$$\stackrel{df}{=} \forall x, y[e(x, y) \rightarrow e(y, x)]$$

$$\Phi_{UG}$$

$$\stackrel{df}{=} \{\phi_{e-\text{irreflexivity}}, \phi_{e-\text{symmetry}}\}$$

Equivalently $\phi'_{e-\text{symmetry}} \stackrel{df}{=} \forall x, y[e(x, y) \leftrightarrow e(y, x)]$ (see exercise 20.1.12) may be used in place of $\phi_{e-\text{symmetry}}$.

Irreflexive Partial Orderings

$$\begin{aligned}\Sigma &= \{ ; \langle \} \\ \phi_{<-irreflexivity} &\stackrel{df}{=} \forall x[\neg(x < x)] \\ \phi_{<-transitivity} &\stackrel{df}{=} \forall x, y, z[((x < y) \wedge (y < z)) \rightarrow (x < z)] \\ \Phi_{IPO} &\stackrel{df}{=} \{\phi_{<-irreflexivity}, \phi_{<-transitivity}\}\end{aligned}$$

Irreflexive Linear Orderings

 Σ $= \{; <\}$ $\phi_{<-irreflexivity}$ $\stackrel{df}{=} \forall x[\neg(x < x)]$ $\phi_{<-transitivity}$ $\stackrel{df}{=} \forall x, y, z[((x < y) \wedge (y < z)) \rightarrow (x < z)]$ $\phi_{<-trichotomy}$ $\stackrel{df}{=} \forall x, y[(x < y) \vee (x = y) \vee (y < x)]$ Φ_{ILO} $\stackrel{df}{=} \{\phi_{<-irreflexivity}, \phi_{<-transitivity}, \phi_{<-trichotomy}\}$

(Reflexive) Preorders

$$\begin{aligned}\Sigma &= \{ ; \leq : s^2 \} \\ \phi_{\leq-\text{reflexivity}} &\stackrel{df}{=} \forall x [x \leq x] \\ \phi_{\leq-\text{transitivity}} &\stackrel{df}{=} \forall x, y, z [(x \leq y) \wedge (y \leq z)) \rightarrow (x \leq z)] \\ \Phi_{Pre} &\stackrel{df}{=} \{\phi_{\leq-\text{reflexivity}}, \phi_{\leq-\text{transitivity}}\}\end{aligned}$$

(Reflexive) Partial Orderings

$$\begin{aligned}\Sigma &= \{\; ; \leq : s^2\} \\ \phi_{\leq-\text{reflexivity}} &\stackrel{df}{=} \forall x[x \leq x] \\ \phi_{\leq-\text{transitivity}} &\stackrel{df}{=} \forall x, y, z[((x \leq y) \wedge (y \leq z)) \rightarrow (x \leq z)] \\ \phi_{\leq-\text{antisymmetry}} &\stackrel{df}{=} \forall x, y[((x \leq y) \wedge (y \leq x)) \rightarrow x = y] \\ \Phi_{PO} &\stackrel{df}{=} \{\phi_{\leq-\text{reflexivity}}, \phi_{\leq-\text{transitivity}}, \\ &\quad \phi_{\leq-\text{antisymmetry}}\}\end{aligned}$$

(Reflexive) Linear Orderings

$$\begin{aligned}\Sigma &= \{ ; \leq : s^2 \} \\ \phi_{\leq-\text{reflexivity}} &\stackrel{df}{=} \forall x [x \leq x] \\ \phi_{\leq-\text{transitivity}} &\stackrel{df}{=} \forall x, y, z [(x \leq y) \wedge (y \leq z)) \rightarrow (x \leq z)] \\ \phi_{\leq-\text{dichotomy}} &\stackrel{df}{=} \forall x, y [(x \leq y) \vee (y \leq x)] \\ \Phi_{LO} &\stackrel{df}{=} \{\phi_{\leq-\text{reflexivity}}, \phi_{\leq-\text{transitivity}}, \phi_{\leq-\text{dichotomy}}\}\end{aligned}$$

Equivalence Relations

$$\begin{aligned}\Sigma &= \{ ; \sim : s^2 \} \\ \phi_{\sim-\text{reflexivity}} &\stackrel{df}{=} \forall x [x \sim x] \\ \phi_{\sim-\text{symmetry}} &\stackrel{df}{=} \forall x, y [(x \sim y) \rightarrow (y \sim x)] \\ \phi_{\sim-\text{transitivity}} &\stackrel{df}{=} \forall x, y, z [((x \sim y) \wedge (y \sim z)) \rightarrow (x \sim z)] \\ \Phi_{Equiv} &\stackrel{df}{=} \{\phi_{\sim-\text{reflexivity}}, \phi_{\sim-\text{symmetry}}, \phi_{\sim-\text{transitivity}}\}\end{aligned}$$

Peano's Postulates

P1. 0 is a natural number.

P2. If x is a natural number then x^{+1} (called the *successor* of x) is a natural number.

P3. $0 \neq x^{+1}$ for any natural number x .

P4. $x^{+1} = y^{+1}$ implies $x = y$

P5. Let P be a property that may or may not hold of every natural number. If

Basis. 0 has the property P and

Induction Step. whenever a natural number x has the property P , x^{+1} also has the property P

then all natural numbers have the property P .

The Theory of The Naturals

$$\begin{aligned}\Sigma_S &= \{0 : \rightarrow s, {}^{+1} : s \rightarrow s; = : s^2\} \\ \phi_{0\text{-notsuccessor}} &\stackrel{df}{=} \forall x[\neg(x^{+1} = 0)] \\ \phi_{+1\text{-injective}} &\stackrel{df}{=} \forall x \forall y[x^{+1} = y^{+1} \rightarrow x = y] \\ \phi_{\neq 0 \rightarrow \text{successor}} &\stackrel{df}{=} \forall y[\neg(y = 0) \rightarrow \exists x[y = x^{+1}]] \\ \phi_{(+1)^n\text{-distinct}} &\stackrel{df}{=} \forall x[\neg(x^{(+1)^n} = x)] \\ \Phi_{(+1)^\infty\text{-distinct}} &\stackrel{df}{=} \{\phi_{(+1)^n\text{-distinct}} \mid n > 0\} \\ \Phi_S &\stackrel{df}{=} \{\phi_{0\text{-notsuccessor}}, \phi_{+1\text{-injective}}, \phi_{\neq 0 \rightarrow \text{successor}}\} \\ &\quad \cup \Phi_{(+1)^\infty\text{-distinct}}\end{aligned}$$

Notes and Explanations

1. $x^{(+1)^n}$ denotes the n -fold application of $+1$ to x .
2. $\phi_{\neq 0 \rightarrow \text{successor}}$ says that every “non-zero” element must have a “predecessor”.
3. $\mathbf{N}_S = \langle \mathbb{N}, \Sigma_S \rangle$ is a model of the axioms Φ_S .
4. The infinite collection of axioms $\Phi_{(+1)^n-\text{distinct}}$ is necessary to obtain models that are countable.
5. The axioms $\Phi_{(+1)^\infty-\text{distinct}}$ ensure that there are no finite models of Φ_S .

Finite Models of Arithmetic

1. If the infinite collection $\Phi_{(+1)^\infty\text{-}distinct}$ is replaced by a finite collection for some $m > 0$ i.e.

$$\Phi_{(+1)^{m>n>0}\text{-}distinct} = \{\phi_{(+1)^n\text{-}distinct} \mid m > n > 0\}$$

then both finite and countable models are possible.

2. If in addition to $\Phi_{(+1)^{m>n>0}\text{-}distinct}$ we also include the axiom

$$\psi_{modulo_m} \stackrel{df}{=} \forall x [x^{(+1)^m} = x]$$

we get models $\mathbf{Z}_{S_m} = \langle \mathbb{Z}_m, \Sigma_S \rangle$ for the integers modulo m and there are no infinite models.

A Non-standard Model of Arithmetic

Consider the model $\mathbf{N}_S = \langle \mathbb{N}, \Sigma_S \rangle$ 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 \mathbb{Z}' .

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

Z-Chains

- \mathbb{Z}' is called a *Z-Chain*.
- $\mathbf{N}'_S = \langle \mathbb{N} \cup \mathbb{Z}', \Sigma_S \rangle$ is also a *countable* model of the axioms Φ_S .
- Further \mathbf{N}_S and \mathbf{N}'_S are **not isomorphic**
- We could add a countable number of distinct *Z-chains*, \mathbb{Z}'' , \mathbb{Z}''' , \mathbb{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*.

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

722 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

723 OF 783

QUIT

Lecture 36: Towards Logic Programming

Tuesday 15 November 2011

1. Reversing the Arrow
2. Horn Clauses
3. Goal clauses
4. Logic Programs
5. Sorting in Logic
6. Prolog: Selection Sort
7. Prolog: Merge Sort
8. Prolog: Quick Sort
9. Prolog: SEND+MORE=MONEY
10. Prolog: Naturals

Reversing the Arrow

Let

$$\phi \leftarrow \psi \stackrel{df}{=} \psi \rightarrow \phi$$

Consider any clause $C = \{\pi_1, \dots, \pi_p\} \cup \{\neg\nu_1, \dots, \neg\nu_n\}$ where π_i , $1 \leq i \leq p$ are *positive literals* and $\neg\nu_j$, $1 \leq j \leq n$ are the *negative literals*. Then we have

$$\begin{aligned} C &\Leftrightarrow \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \vee (\bigvee_{1 \leq j \leq n} \neg\nu_j)] \\ &\Leftrightarrow \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \vee \neg(\bigwedge_{1 \leq j \leq n} \nu_j)] \\ &\Leftrightarrow \vec{\forall}[(\bigwedge_{1 \leq j \leq n} \nu_j) \rightarrow (\bigvee_{1 \leq i \leq p} \pi_i)] \\ &\equiv \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \leftarrow (\bigwedge_{1 \leq j \leq n} \nu_j)] \\ &\stackrel{df}{=} \pi_1, \dots, \pi_p \leftarrow \nu_1, \dots, \nu_n \end{aligned}$$

Horn Clauses

Definition 36.1 *Given a clause*

$$C \stackrel{df}{=} \pi_1, \dots, \pi_p \leftarrow \nu_1, \dots, \nu_n$$

- *Then C is a **Horn clause** if $0 \leq p \leq 1$.*
- C is called a
 - **program clause or rule clause** if $p = 1$,
 - **fact or unit clause** if $p = 1$ and $n = 0$,
 - **goal clause or query** if $p = 0$,
- *Each ν_j is called a **sub-goal** of the goal clause.*

Goal clauses

Given a goal clause

$$\begin{aligned} G &\stackrel{df}{=} \leftarrow \nu_1, \dots, \nu_n \\ &\Leftrightarrow \vec{\forall}[\neg\nu_1 \vee \dots \vee \neg\nu_n] \\ &\Leftrightarrow \neg\vec{\exists}[\nu_1 \wedge \dots \wedge \nu_n] \end{aligned}$$

If $\vec{y} = FV(\nu_1 \wedge \dots \wedge \nu_n)$ then the goal is to prove that there exists an assignment to \vec{y} which makes $\nu_1 \wedge \dots \wedge \nu_n$ true.

Logic Programs

Definition 36.2 A logic program is a finite set of **Horn clauses**, i.e. it is a set of rules $P = \{h^1, \dots, h^k\}$, $k \geq 0$ with $h^l \equiv \pi^l \leftarrow \nu_1^l, \dots, \nu_{n_l}^l$, for $0 \leq l \leq k$. π^l is called the **head** of the rule and $\nu_1^l, \dots, \nu_{n_l}^l$ is the **body** of the rule.

Given a logic program P and a **goal** clause $G = \{\nu_1, \dots, \nu_n\}$ the basic idea is to show that

$$\begin{aligned} P \cup \{G\} \text{ is unsatisfiable} \\ \Leftrightarrow \vec{\forall}[\neg\nu_1 \vee \dots \vee \neg\nu_n] \text{ is a logical consequence of } P \\ \Leftrightarrow \vec{\exists}[\nu_1 \wedge \dots \wedge \nu_n] \text{ is a logical consequence of } P \end{aligned}$$

Sorting in Logic

$sort(x, y)$	$\leftarrow perm(x, y), sorted(y)$
$sorted(nil)$	\leftarrow
$sorted(x.nil)$	\leftarrow
$sorted(x.y.z)$	$\leftarrow lesseq(x, y), sorted(y.z)$
$lesseq(x, x)$	\leftarrow
$lesseq(x, y)$	$\leftarrow x < y$
$perm(nil, nil)$	\leftarrow
$perm(x.y, u.v)$	$\leftarrow delete(u, x.y, z), perm(z, v)$
$delete(x, x.y, y)$	\leftarrow
$delete(x, y.z, y.w)$	$\leftarrow delete(x, z, w)$
	$\leftarrow sort([2, 8, -1, 10, 4, 2], x)$

Prolog: Selection Sort

```
selectSort(X, Y) :- permutation(X, Y),  
                      sorted(Y).
```

```
sorted([]).
```

```
sorted([H|T]).
```

```
sorted([F,S|T]) :- lesseq(F, S),  
                      sorted(S|T).
```

```
lesseq(F, S) :- F=S.
```

```
lesseq(F, S) :- F<S.
```

```
permutation([], []).
```

```
permutation([H|T], [F|R]) :- delete(H, [H|T], Z),  
                                permutation(Z, R).
```

```
delete(H, [H|T], T).
```

```
delete(X, [H|T], [H|U]) :- delete(X, T, U).
```

Prolog: Merge Sort

```
mergeSort([], []).
mergeSort([H|T], [H|T]).
mergeSort([F,S,T], [S1|T1]) :- split([F,S,T], [S1|T1]),
                                mergeSort([S1|T1], [S1|T1]),
                                mergeSort([T1|T], [T1|T]),
                                merge([S1|T1], [T1|T], [S1|T1]).  
split([], [], []).
split([H|T], [H|T], []).
split([F,S,T], [F1,U1,S1,V1], [T1|T2]) :- split([S,T], [S1,V1], [T1|T2]),
                                              split([F,U], [F1,U1], [T2|T2]).  
merge([], L, L).
merge(L, [], L).
merge([F,B], [H,T], [F,U]) :- F < H, merge([B], [H,T], [U]).  
merge([F,B], [H,T], [H,V]) :- H < F, merge([F,B], [T], [V]).
```

Prolog: Quick Sort

```
quicksort([], []).
quicksort([H|T], [H|T]) :- !.
quicksort([H|T], S) :- partition(H, T, L, G),
                     quicksort(L, Ls),
                     quicksort(G, Gs),
                     append(Ls, [H|T], Lsh),
                     append(Lsh, Gs, S).

partition(M, [], [], []).
partition(M, H|T, H.Lesser, Greater) :- H < M,
                                         !, partition(M, T, Lesser, Greater).
partition(M, H|T, Lesser, H.Greater) :- M < H,
                                         !, partition(M, T, Lesser, Greater).

append([], L, L).
append([H|T], L, [H|A]) :- append(T, L, A).
```

Prolog: SEND+MORE=MONEY

```
smm :- L = [S,E,N,D,M,O,R,Y] ,  
        Digits = [0,1,2,3,4,5,6,7,8,9] ,  
        assign_digits(L, Digits) ,  
        M > 0, S > 0 ,  
        1000*S + 100*E + 10*N + D +  
        1000*M + 100*O + 10*R + E =:=  
        10000*M + 1000*O + 100*N + 10*E + Y ,  
        write(' ') , write(S) , write(E) , write(N) , write(D) , nl ,  
        write(' + ') , write(M) , write(O) , write(R) , write(E) , nl ,  
        write(' -----') , nl ,  
        write(' = ') , write(M) , write(O) , write(N) , write(E) , write(Y) , nl  
  
select(Z, [Z|R], R).  
select(Z, [Y|Zs], [Y|Ys]):- select(Z, Zs, Ys).  
  
assign_digits([], _List).  
assign_digits([D|Ds], List):- select(D, List, NewList) ,  
                           assign_digits(Ds, NewList).
```

Prolog: Naturals

```
isnf(z).  
isnf(s(X)) :- isnf(X).  
rewrite(X, X) :- isnf(X).  
rewrite(s(X), s(Y)) :- rewrite(X, Y), isnf(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))) :- rewrite(X, Y), even(Y).  
odd(X) :- not even(X). % negation as failure  
/* rewrite(a(a(s(z), s(s(z))), a(s(z), s(s(z)))), X).  
X = s(s(s(s(s(z)))))) */
```

```

/* Implementing double-ended queues through constructors */
deq(nullq).
deq(fnq(A, D)) :- integer(A), deq(D).
deq(rnq(B, D)) :- integer(B), deq(D).

nonnull(fnq(A, D)) :- integer(A), deq(D).
nonnull(rnq(B, D)) :- integer(B), deq(D).

deq(fdq(D)) :- nonnull(D).
deq(rdq(D)) :- nonnull(D).

nf(nullq).
nf(fnq(A, D)) :- integer(A), nf(D).

rewrite(D, D) :- nf(D).
%induction step for normal forms
rewrite(fnq(A, D), fnq(A, E)) :- integer(A), rewrite(D, E).

% for all forms other than normal forms
rewrite(rnq(B, nullq), fnq(B, nullq)):- integer(B). % basis of induction
rewrite(rdq(fnq(A, nullq)), nullq). % basis of induction

% rewrite(fdq(fnq(A, nullq)), nullq) follows from the more general rewrite
rewrite(fdq(fnq(A, D)), E):- integer(A), rewrite(D, E). % fdq for all nonnull

rewrite(rnq(B, fnq(A, D)), fnq(A, E)) :- % for rnq on normal forms
    integer(A), integer(B),
    rewrite(D, F),

```

```

rewrite(rnq(B, F), E), nf(E).

rewrite(rnq(B, D), E) :- % for rnq on other forms
    integer(B),
    rewrite(D, F),
    rewrite(rnq(B, F), E).

rewrite(rdq(fnq(A, D)), fnq(A, E)) :- % for rdq on normal forms
    integer(A),
    rewrite(D, F), nonnull(F),
    rewrite(rdq(F), E).

rewrite(rdq(D), E) :- % for rdq on other forms
    rewrite(D, F), rewrite(rdq(F), E).

% rewrite(rdq(rnq(B, D)), D) follows by induction from the various rewrites above
fv(fnq(A, D), D):- integer(A), deq(D). % value at the front of the dequeue
rv(fnq(A, nullq), B) :- integer(A), A=B.
rv(fnq(A, D), B) :- integer(A), rewrite(D, E), rv(E, B).
rv(D, B) :- rewrite(D, E), rv(E, B).

/* Testing
% Restoring file /usr/local/lib/Yap/startup
YAP version Yap-5.1.1
?- % reconsulting /home/sak/prolog/deques.P...
% reconsulted /home/sak/prolog/deques.P in module user, 0 msec 4096 bytes
yes
?- rewrite(fnq(2, fnq(1, nullq)), X).

```

```
X = fnq(2,fnq(1,nullq)) ?  
yes  
?- rv(fnq(1, fnq(2, nullq)), B).  
B = 2 ?  
yes  
?- rv(rnq(3, fnq(1, fnq(2, nullq))), B).  
B = 3 ?  
yes  
?- rewrite(rnq(4, fdq(fnq(1, fnq(2, rnq(3, nullq))))), X).  
X = fnq(2,fnq(3,fnq(4,nullq))) ?  
yes  
?- rv(rnq(4, fdq(fnq(1, fnq(2, rnq(3, nullq))))), B).  
B = 4 ?  
yes  
?- rv(rdq(rnq(4, fdq(fnq(1, fnq(2, rnq(3, nullq))))), B).  
B = 3 ?  
yes  
*/
```

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

739 OF 783

QUIT

Lecture 37: Verification of Imperative Programs

Wednesday 16 November 2011

1. The WHILE Programming Language
2. Programs As State Transformers
3. The Semantics of WHILE
4. Programs As Predicate Transformers
5. Correctness Assertions
6. Total Correctness of Programs
7. Examples: Factorial 1
8. Examples: Factorial 2

The WHILE Programming Language

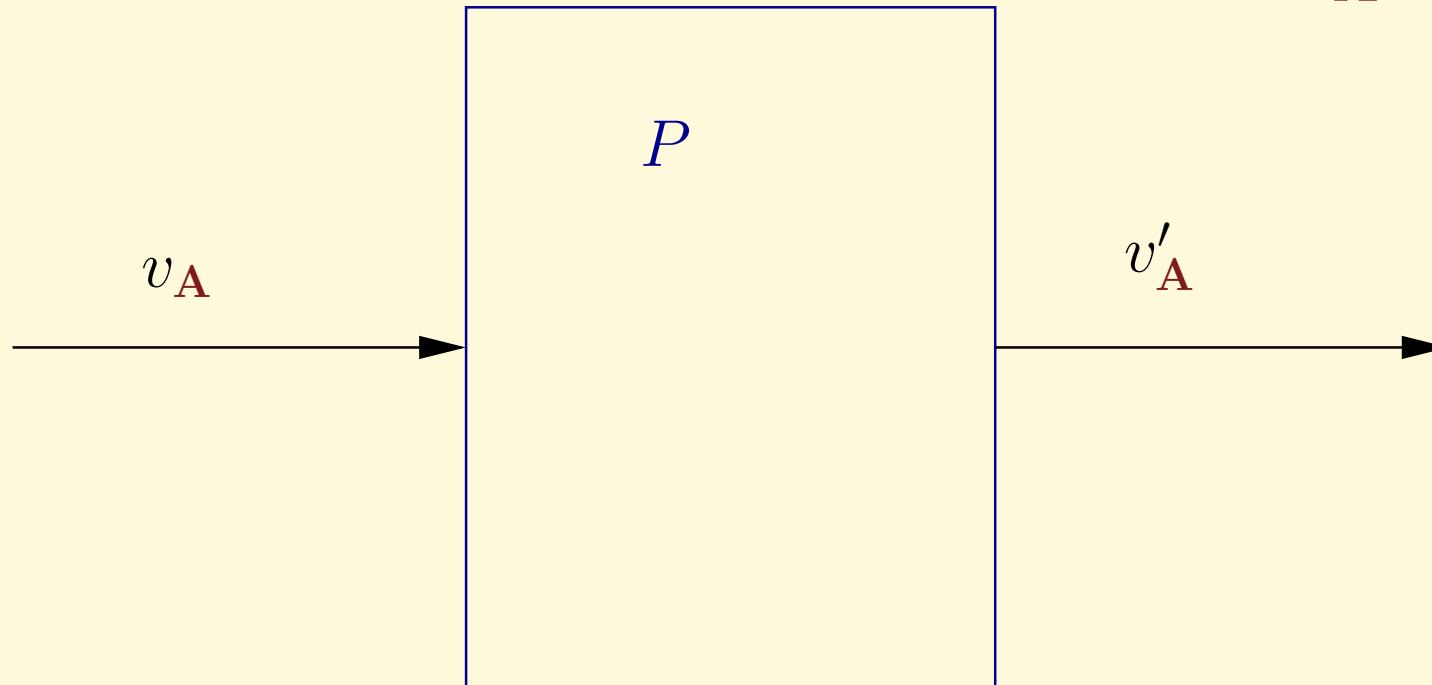
Let Σ be any signature. Then the programming language $\mathcal{WH}(\Sigma)$ is defined by the following BNF.

$$\begin{aligned} P, Q ::= & \epsilon \mid x := t \mid P; Q \mid [P] \mid \\ & \chi?P : Q \mid \\ & \{\chi?P\} \end{aligned}$$

where $\chi \in \mathcal{QF}(\Sigma)$.

Programs As State Transformers

A program is merely a *state transformer* which takes a valuation v_A of variables and yields another valuation v'_A .



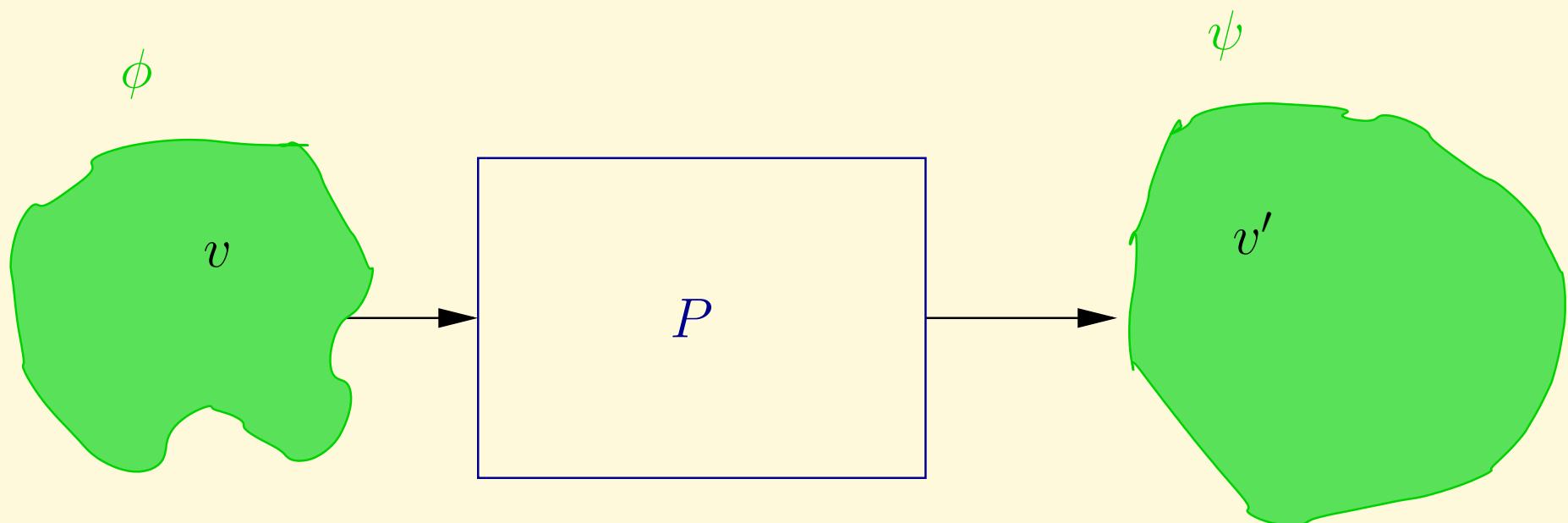
The Semantics of WHILE

Let \mathbf{A} be a Σ -algebra. Let $\mathbf{V}_{\mathbf{A}} = \{v_{\mathbf{A}} \mid v_{\mathbf{A}} : V \rightarrow |\mathbf{A}|\}$ be the set of all valuations (also called *states*). The meaning of a program P is given by $\mathcal{M}_{\mathbf{A}}[P] : \mathbf{V}_{\mathbf{A}} \rightarrow \mathbf{V}_{\mathbf{A}}$

$$\begin{aligned}\mathcal{M}_{\mathbf{A}}[\epsilon]_{v_{\mathbf{A}}} &\stackrel{df}{=} v_{\mathbf{A}} \\ \mathcal{M}_{\mathbf{A}}[x := t]_{v_{\mathbf{A}}} &\stackrel{df}{=} v_{\mathbf{A}}[x := \mathcal{V}_{\mathbf{A}}[t]_{v_{\mathbf{A}}}] \\ \mathcal{M}_{\mathbf{A}}[[P]]_{v_{\mathbf{A}}} &\stackrel{df}{=} \mathcal{M}_{\mathbf{A}}[P]_{v_{\mathbf{A}}} \\ \mathcal{M}_{\mathbf{A}}[P; Q]_{v_{\mathbf{A}}} &\stackrel{df}{=} (\mathcal{M}_{\mathbf{A}}[Q] \circ \mathcal{M}_{\mathbf{A}}[P])_{v_{\mathbf{A}}} \\ \mathcal{M}_{\mathbf{A}}[\chi?P : Q]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} \mathcal{M}_{\mathbf{A}}[P]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 1 \\ \mathcal{M}_{\mathbf{A}}[Q]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 0 \end{cases} \\ \mathcal{M}_{\mathbf{A}}[\{\chi?P\}]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} \mathcal{M}_{\mathbf{A}}[P; \{\chi?P\}]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 1 \\ v_{\mathbf{A}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 0 \end{cases}\end{aligned}$$

Programs As Predicate Transformers

We may also view a program as a transformer of properties. Consider a predicate ϕ to represent a set of possible valuations i.e. $V_A(\phi) = \{v_A \mid (A, v_A) \Vdash \phi\}$. Then a program transforms a state satisfying ϕ into a state satisfying some other predicate ψ .



Correctness Assertions

Definition 37.1 A partial correctness assertion (also called a Hoare-triple) is a triple of the form $\{\phi\} P \{\psi\}$ where ϕ is a formula called the precondition, P is a program and ψ is the postcondition.

Definition 37.2

- $\{\phi\} P \{\psi\}$ holds in a state v_A (denoted $(A, v_A) \Vdash \{\phi\} P \{\psi\}$), if $(A, v_A) \Vdash \phi$ and $M_A[P]_{v_A} = v'_A$ implies $(A, v'_A) \Vdash \psi$
- $\{\phi\} P \{\psi\}$ is valid in A , (denoted $A \models \{\phi\} P \{\psi\}$) if $(A, v_A) \Vdash \{\phi\} P \{\psi\}$ for every state v_A .

Total Correctness of Programs

Definition 37.3 A total correctness assertion is a triple of the form $[\phi] P [\psi]$ where ϕ is a formula called the precondition, P is a program and ψ is the postcondition.

Definition 37.4

- $[\phi] P [\psi]$ holds in a state v_A (denoted $(A, v_A) \Vdash [\phi] P [\psi]$), if $(A, v_A) \Vdash \phi$ implies for some v'_A $M_A[P]_{v_A} = v'_A$ and $(A, v'_A) \Vdash \psi$.
- $[\phi] P [\psi]$ is valid in A , (denoted $A \models \{\phi\} P \{\psi\}$) if $(A, v_A) \Vdash [\phi] P [\psi]$ for every state v_A .

Examples: Factorial 1

Let $\Sigma \supseteq \mathbb{Z} \cup \{! : s, +, -, * : s^2 \rightarrow s; =, > : s^2\}$.

Example 37.5 Let $P_1 \stackrel{df}{=} p := 1; \{\neg(x = 0)?[p := p * x; x := x - 1]\}$.
Let $\mathbf{Z} = \langle \mathbb{Z}, \Sigma \rangle$. Then

1. $\mathbf{Z} \models \{x = x_0\} P_1 \{p = x_0!\}$
2. However $\mathbf{Z} \not\models [x = x_0] P_1 [p = x_0!]$ since P_1 will not terminate for negative values of x .

Examples: Factorial 2

Example 37.6 Let $P_2 \stackrel{df}{=} p := 1; \{x > 0? [p := p * x; x := x - 1]\}$.
Let $\mathbf{Z} = \langle \mathbb{Z}, \Sigma \rangle$. Then

1. $\mathbf{Z} \models \{x = x_0 \geq 0\} P \{p = x_0!\}$
2. $\mathbf{Z} \models [x = x_0 \geq 0] P [p = x_0!]$
3. A more “technically complete” specification is
 $\mathbf{Z} \models [x = x_0] P [(x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)]$

Exercise 37.1

1. For any program P defined on a signature Σ what do the following correctness formulae mean?

- (a) $\{\top\} P \{\top\}$
- (b) $\{\top\} P \{\perp\}$
- (c) $\{\perp\} P \{\top\}$
- (d) $\{\perp\} P \{\perp\}$
- (e) $[\top] P [\top]$
- (f) $[\top] P [\perp]$
- (g) $[\perp] P [\top]$
- (h) $[\perp] P [\perp]$

2. Which of the correctness formulae given in problem 1 are

- (a) always valid?
- (b) always unsatisfiable?

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

751 OF 783

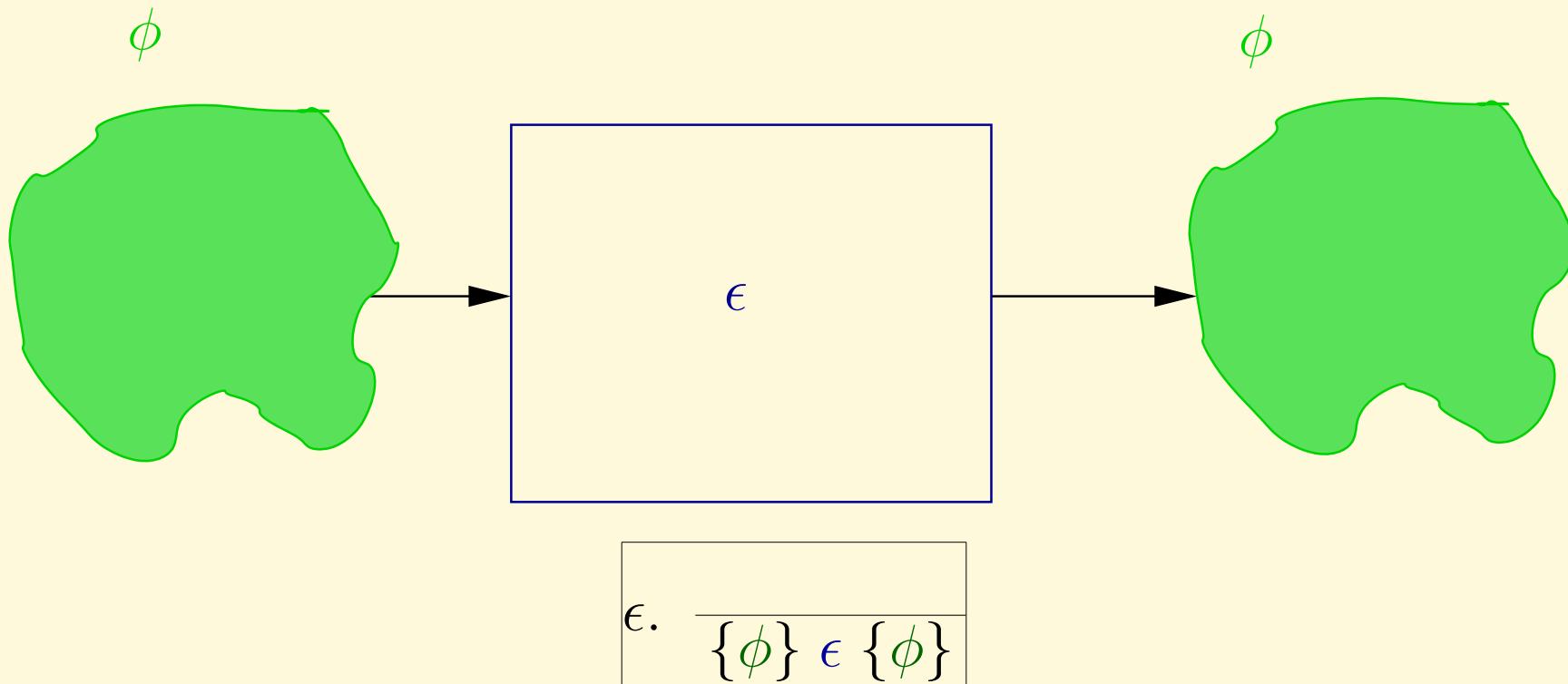
QUIT

Lecture 38: Verification of WHILE Programs

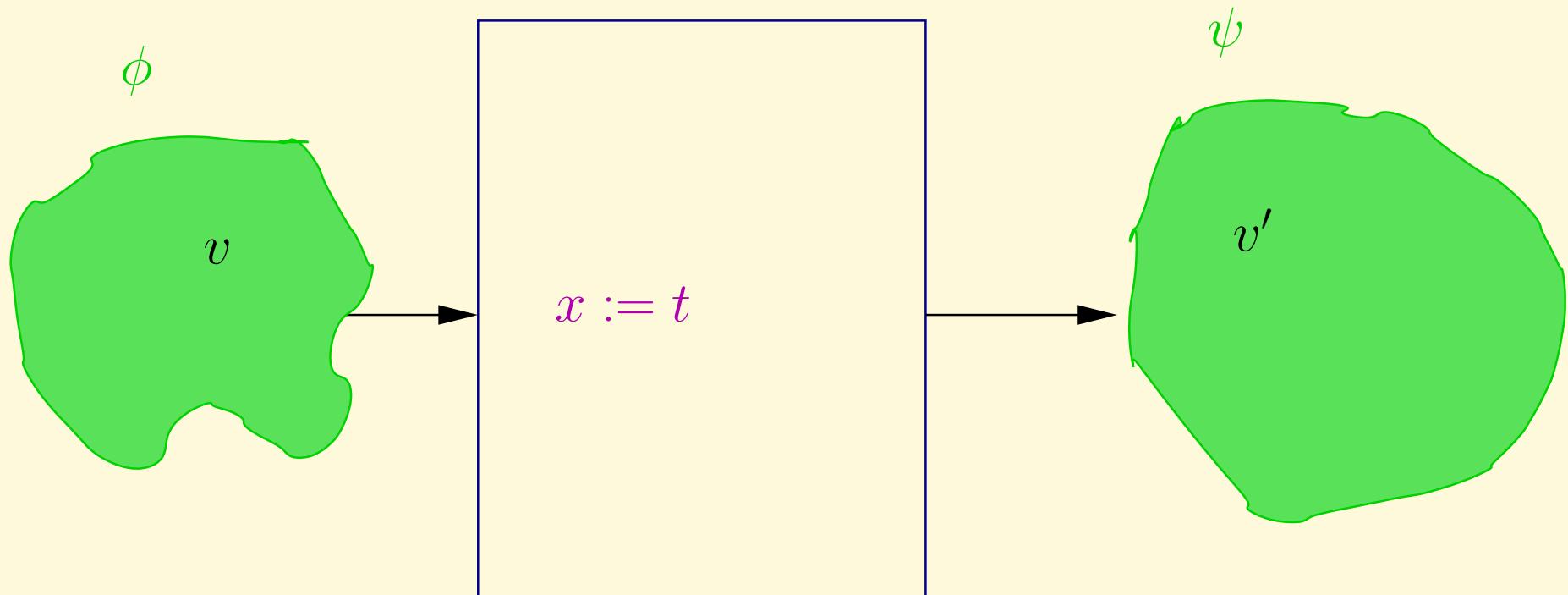
Friday 18 November 2011

1. Proof Rule: Epsilon
2. Proof Rule: Assignment
3. Proof Rule: Composition
4. Proof Rule: The Conditional
5. Proof Rule: The While Loop
6. The Consequence Rule
7. Proof Rules for Partial Correctness
8. Example: Factorial 1
9. Towards Total Correctness
10. Termination and Total Correctness
11. Example: Factorial 2
12. Notes on Example: Factorial
13. Example: Factorial 2 Made Complete
14. An Open Problem: Collatz

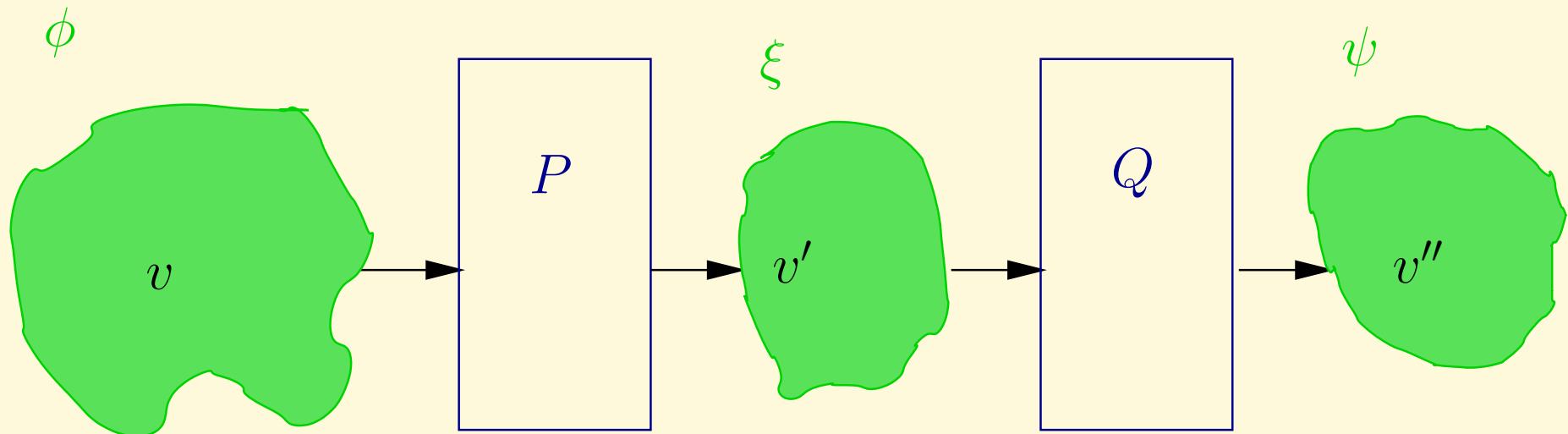
Proof Rule: Epsilon



Proof Rule: Assignment

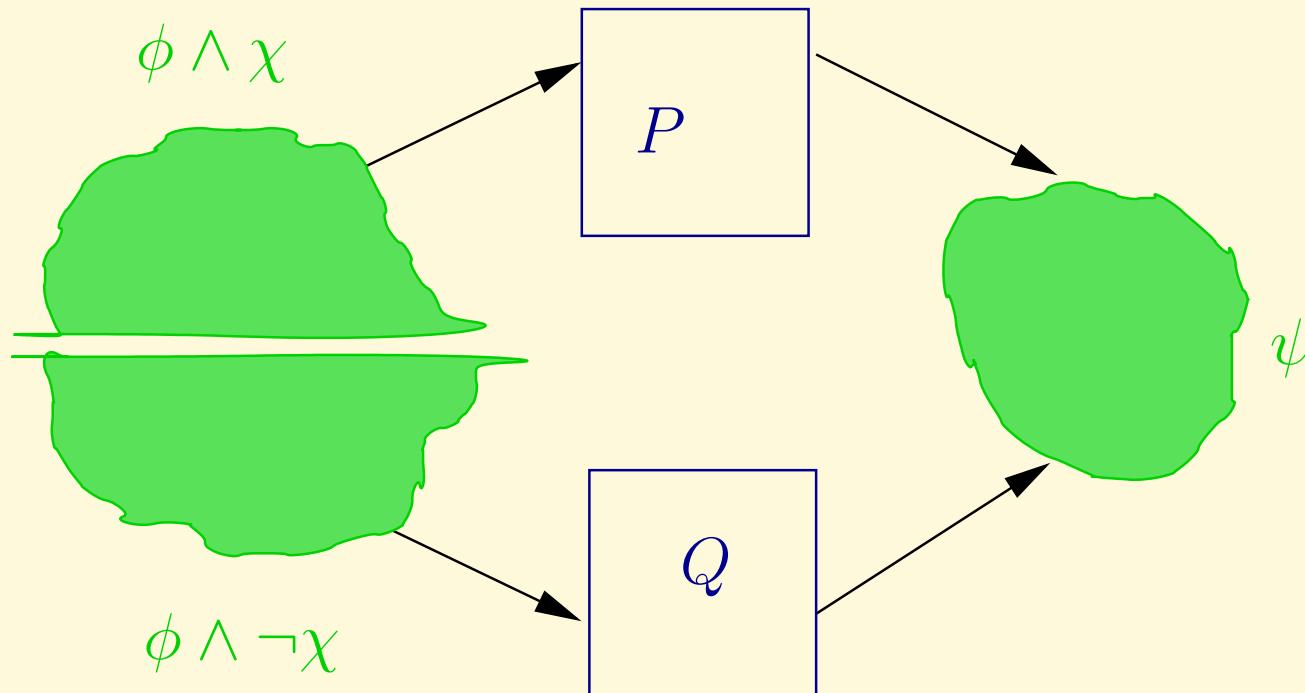

$$:= \cdot \frac{\{\phi\} \ x := t \ \{\psi\}}{} \ (\phi \equiv \{t/x\}\psi)$$

Proof Rule: Composition



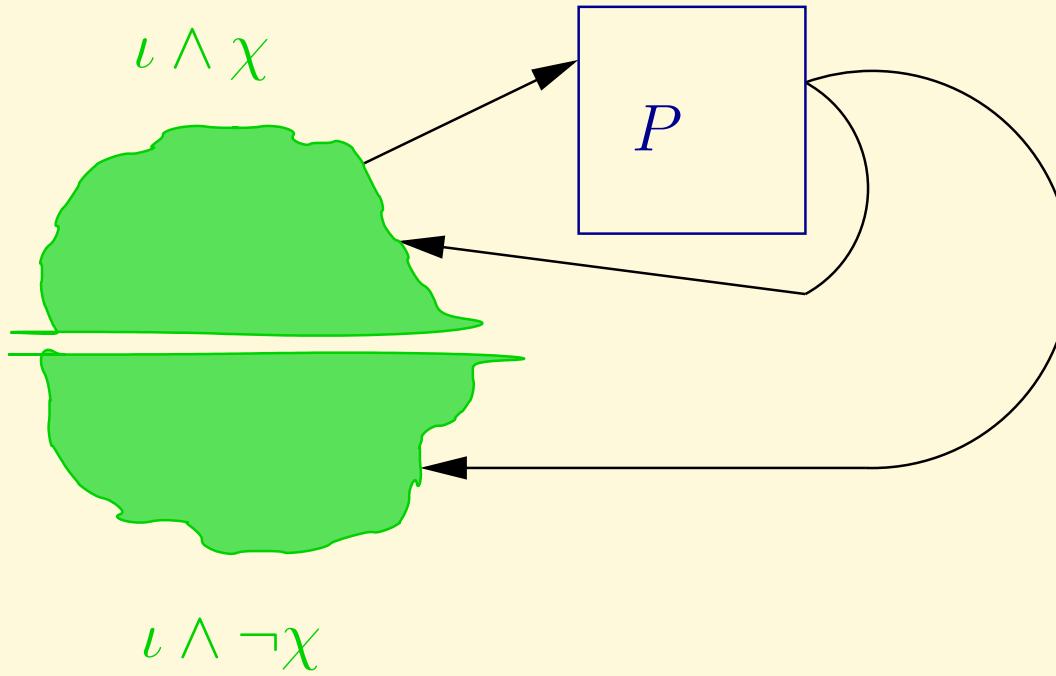
$$\frac{\{\phi\} P \{\xi\} \\ ; . \quad \{\xi\} Q \{\psi\}}{\{\phi\} P; Q \{\psi\}}$$

Proof Rule: The Conditional



$$\boxed{\frac{? : . \quad \begin{array}{c} \{\phi \wedge \chi\} \ P \ \{\psi\} \\ \{\phi \wedge \neg\chi\} \ Q \ \{\psi\} \end{array}}{\{\phi\} \ \chi ? P : Q \ \{\psi\}}}$$

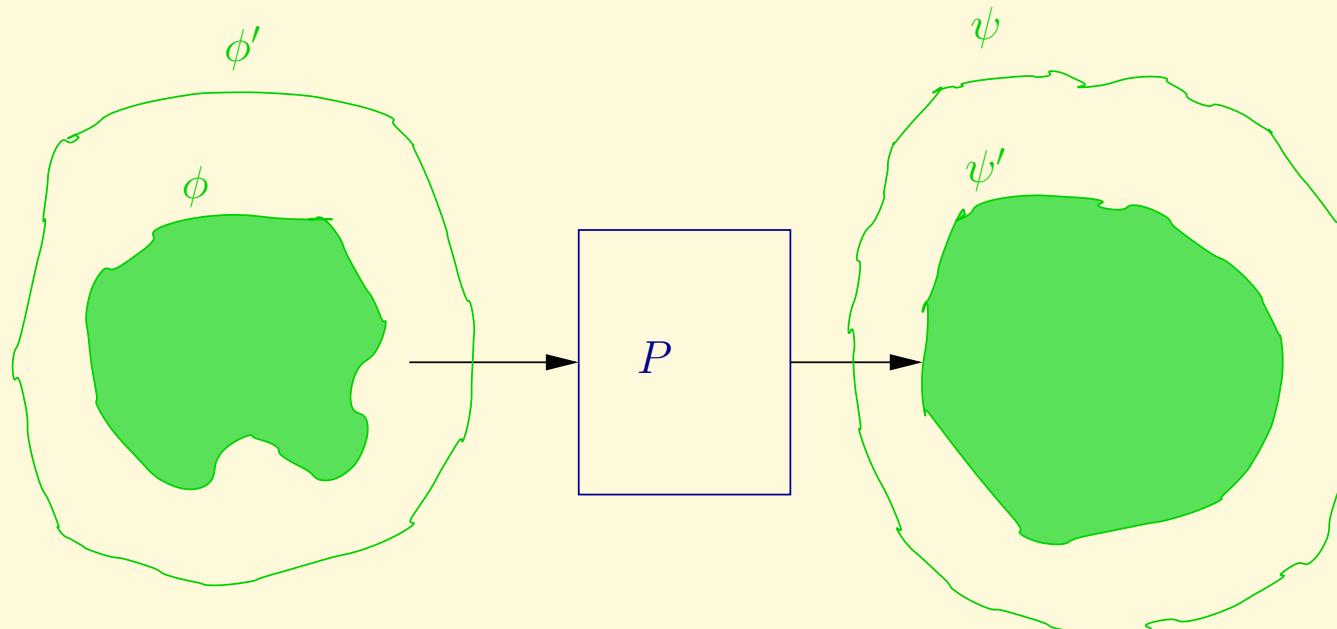
Proof Rule: The While Loop



$$\{\} \cdot \frac{\{\iota \wedge \chi\} \ P \ \{\iota\}}{\{\iota\} \ \{\chi?P\} \ \{\iota \wedge \neg\chi\}}$$

ι in the while rule is called the **loop invariant**.

The Consequence Rule



$$\Rightarrow \frac{\begin{array}{c} \phi \Rightarrow \phi' \\ \{\phi'\} P \{\psi'\} \\ \psi' \Rightarrow \psi \end{array}}{\{\phi\} P \{\psi\}}$$

Proof Rules for Partial Correctness

$$\epsilon. \quad \{\phi\} \in \{\phi\}$$

$$:= . \quad \{\{t/x\}\psi\} \ x := t \ \{\psi\}$$

$$[] . \quad \begin{array}{c} \{\phi\} \ P \ \{\psi\} \\ \{\phi\} [P] \ \{\psi\} \end{array}$$

$$\{\}. \quad \frac{\{\iota \wedge \chi\} \ P \ \{\iota\}}{\{\iota\} \ \{\chi?P\} \ \{\iota \wedge \neg \chi\}}$$

$$\vdots . \quad \frac{\{\phi\} \ P \ \{\xi\} \quad \{\xi\} \ Q \ \{\psi\}}{\{\phi\} \ P; Q \ \{\psi\}}$$

$$? \vdots . \quad \frac{\{\phi \wedge \chi\} \ P \ \{\psi\} \quad \{\phi \wedge \neg \chi\} \ Q \ \{\psi\}}{\{\phi\} \ \chi?P : Q \ \{\psi\}}$$

$$\Rightarrow . \quad \frac{\begin{array}{c} \phi \Rightarrow \phi' \\ \{\phi'\} \ P \ \{\psi'\} \\ \psi' \Rightarrow \psi \end{array}}{\{\phi\} \ P \ \{\psi\}}$$

ι in the while rule is called the **loop invariant**.

Example: Factorial 1

	$\{x = x_0\}$
$p := 1;$	$\{x = x_0 \wedge p = 1\}$
	$\Rightarrow \{(x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!)\}$
	$\Rightarrow \{\phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!)\} \equiv \{\iota\}$
$\{\neg(x = 0)?$	$\{\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \neg(x = 0)\}$
	$\Rightarrow \{\phi_0 \wedge (x > 0) \wedge (p * x! = x_0!)\}$
$[p := p * x;$	$\{\phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \}$
$x := x - 1]$	$\{\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!)\}$
	$\Rightarrow \{\iota\}$
$}$	$\{\iota \wedge (x = 0)\}$
	$\Rightarrow \{\phi_0 \wedge (x = 0) \wedge (p * x! = x_0!)\}$
	$\Rightarrow \{(x_0 \geq 0 \rightarrow p = x_0!)\}$

where $\phi_0 \equiv (x_0 \geq 0)$

Towards Total Correctness

1. The only construct which may not terminate is the while loop.
2. Termination of the while loop: Define a “measure” called the **bound function**, $\beta : \text{Dom}(\beta) \rightarrow W$ where
 - $\langle W, < \rangle$ is a well-ordered set (a set with no infinite descending sequences $w > w' > w'' > \dots$) with a least element 0 such that $w < 0$ does not hold true for any $w \in W$.
 - $\text{Dom}(\beta)$ is the tuple of possible values of the program variables (\vec{v}).
 - Often $\langle W, < \rangle = \langle \mathbb{N}, < \rangle$
 - $\iota \wedge \chi \Rightarrow \beta(\vec{v}) > 0$ and $\beta(\vec{v}) = 0 \Rightarrow \iota \wedge \neg \chi$
 - Each execution of the body of the loop decreases the value of the bound function.

Termination and Total Correctness

$$\epsilon! \quad \frac{}{[\phi] \epsilon [\phi]}$$

$$:=! \quad \frac{}{[\{t/x\}\psi] x := t [\psi]}$$

$$[]! \quad \frac{[\phi] P [\phi]}{[\phi] [P] [\phi]}$$

$$; ! \quad \frac{[\phi] P [\xi] \quad [\xi] Q [\psi]}{[\phi] P; Q [\psi]}$$

$$? : ! \quad \frac{[\phi \wedge \chi] P [\psi] \quad [\phi \wedge \neg \chi] Q [\psi]}{[\phi] \chi?P : Q [\psi]}$$

$$\Rightarrow ! \quad \frac{\begin{array}{c} \phi \Rightarrow \phi' \\ [\phi'] P [\psi'] \\ \psi' \Rightarrow \psi \end{array}}{[\phi] P [\psi]}$$

$$\{\}! \quad \frac{[\iota \wedge \chi \wedge \beta(\vec{v}) = b_0 > 0] P [\iota \wedge \beta(\vec{v}) < b_0]}{[\iota] \{\chi?P\} [\iota \wedge \neg \chi]}$$

where

- β is a **bound function** and
- ι is the **loop invariant**.

c.f. Proof Rules for Partial Correctness

Example: Factorial 2

$$\begin{array}{ll} & [x = x_0] \\ p := 1; & [x = x_0 \wedge p = 1] \\ & \Rightarrow [(x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)] \\ & \Rightarrow [\phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!)] \equiv [\iota] \\ \{x > 0? & [\phi_0 \wedge (x > 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0] \\ [p := p * x; & [\phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \\ & \quad \beta(x, p) = x = \beta_0 > 0] \\ x := x - 1] & [\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x < \beta_0 > 0]] \\ & \Rightarrow [\iota] \\ \} & [\phi_0 \wedge (x = 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = 0] \\ & \Rightarrow [(x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)] \equiv [\iota] \end{array}$$

where $\phi_0 \equiv (x_0 \geq 0) \wedge (x_0 < 0 \rightarrow p = 1)$

c.f. Partial correctness proof

Notes on Example: Factorial

- The loop invariant $\textcolor{teal}{\psi}$ is a conjunction of
 - ϕ_0 whose truth is trivially unaffected by the changes in state produced by the loop body, and
 - the formula $(\textcolor{violet}{x} \geq 0 \rightarrow p * x! = x_0!)$ which
 - * *holds initially* before control enters the loop,
 - * *holds after* the condition has been checked,
 - * *fails to hold* after the first command of the body has been executed,
 - * *is restored* at the end of the loop body, and
 - * *holds after exiting* the loop
- Notice the progress of the **bound function** which is completely internal to the working of the loop and is never part of the specification of the program.

Example: Factorial 2 Made Complete

A more complete proof including the use of the assignment clearly delineated is given below.

$$\begin{aligned} & [x = x_0] \equiv [\phi_0] \\ \Rightarrow & [x = x_0 \wedge 1 = 1] \equiv [\{1/p\}\phi_1] \\ p := 1; & [x = x_0 \wedge p = 1] \equiv [\phi_1] \\ \Rightarrow & [(x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)] \\ \Rightarrow & [\phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!)] \equiv [\iota] \\ \{x > 0? & [\phi_0 \wedge (x > 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0] \equiv [\phi'_1] \\ \Rightarrow & [\{p * x/p\}\psi_1] \\ [p := p * x; & [\phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0] \equiv [\psi_1] \\ \Rightarrow & [\{x - 1/x\}\psi_2] \\ x := x - 1 & [\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x < \beta_0 > 0] \equiv [\psi_2] \\] & [\psi_2] \\ \Rightarrow & [\iota] \\ \} & [\iota \wedge \neg(x > 0) \wedge \beta(x, p) = x = 0] \\ \Rightarrow & [\phi_0 \wedge (x = 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = 0] \\ \Rightarrow & [(x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)] \equiv [\iota] \end{aligned}$$

An Open Problem: Collatz

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

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

where

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

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

768 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

769 OF 783

QUIT

Lecture 39: Concluding Remarks

Wednesday 07 December 2011

1. Summary
2. The Limitations of Predicate Logic
3. Sortedness
4. Many-Sorted Logic: Symbols
5. Many-Sorted Signatures
6. Many-Sorted Signature: Terms
7. Many-Sorted Predicate Logic
8. Reductions

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.

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.

Sortedness

- A treatment that works mainly with a 1-sorted term algebra often does not address the interesting problems of a mathematical theory e.g. the first-order theory of directed or undirected graphs.
- To begin to address even the simplest problems of graph theory requires counting and the power of the first order theory of numbers.
- The problems of second-order logic (quantification over first order properties) may be expressed in many-sorted first order logic by allowing the power set of a set to be included in the universe of discourse of a many-sorted first order logic.

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 $S = \{\textcolor{blue}{s}_i \mid 1 \leq i \leq k\}$, a k -sorted logic consists of*

- *A countable set V_i of variables of sort $\textcolor{blue}{s}_i$ for each $\textcolor{blue}{s}_i \in S$.*
- *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 $\textcolor{blue}{s}_i \in S$.*

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 $s_i \in S$.

Many-Sorted Signature: Terms

Definition 39.3 Given a S -sorted signature Σ , the set $T(\Sigma) = \biguplus_{1 \leq i \leq k} T_i(\Sigma)$ 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 $f(t_1, \dots, t_m) : s_{i_0}$.

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 \mathbb{A}_i for each sort.
- The semantics is then a minor extension of the one for the 1-sorted case

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,

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

780 OF 783

QUIT

HOME PAGE



LCS

Go BACK

FULL SCREEN

CLOSE

781 OF 783

QUIT

References

- [1] I. M. Copi. *Symbolic Logic*. Macmillan, London, UK, 1979.
- [2] H. D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, New York, USA, 1994.
- [3] H. B. Enderton. *A Mathematical Introduction to Logic*. Elsevier India, New Delhi, India, 2001.
- [4] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, USA, 1990.
- [5] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, UK, 2000.
- [6] John Kelly. *The Essence of Logic*. Prentice-Hall India, New Delhi, India, 1997.
- [7] S. C. Kleene. *Mathematical Logic*. Dover Publications Inc., New York, USA, 1967.
- [8] E. Mendelson. *Introduction to Mathematical Logic*. D. Van Nostrand Co. Inc., Princeton, New Jersey, USA, 1963.
- [9] Anil Nerode and R. Shore. *Logic for Applications*. Springer-Verlag, New York, USA, 1993.
- [10] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, Germany, 1968.
- [11] V. Sperschneider and G. Antoniou. *Logic: A Foundation for Computer Science*. Addison-Wesley Publishing Company, Reading, UK, 1991.

Thank You!

Any Questions?