

Logic for Computer Science

<http://www.cse.iitd.ac.in/~sak/courses/ilcs/2018-19/2018-19.index.html>

S. Arun-Kumar

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

November 4, 2018

© S. Arun-Kumar

“Where shall I begin, please your Majesty?” he asked.

“Begin at the beginning,” the King said, gravely, “and go on till you come to the end: then stop.”

Lewis Carroll, Alice's Adventures in Wonderland

Contents

0 Background	26
0.1 Mathematical Preliminaries	27
0.1.1 Motivation for the Study of Logic	27
0.1.2 Sets	31
0.1.3 Relations and Functions	36
0.1.4 Operations on Binary Relations	46
0.1.5 Ordering Relations	50
0.1.6 Partial Orders	52
0.1.7 Well-founded Sets and Well-ordered Sets	54
0.1.8 Infinite Sets: Countability and Uncountability	57
0.2 Induction Principles	79
0.2.1 Mathematical Induction	79
0.2.2 Complete Induction	86
0.2.3 Structural Induction	96
0.2.4 Simultaneous Induction	113
0.2.5 Well-ordered Induction	117

0.3	Rooted Trees	124
0.4	Universal Algebra	131
0.4.1	Introduction	132
0.4.2	Data types in functional and declarative languages	132
0.4.3	Terms	140
0.5	Semantics and Meaning	155
0.5.1	Semantics of ground terms	157
0.5.2	Semantics of Open terms	158
0.6	Substitutions	160
0.6.1	Substitutions and Instantiations	160
0.7	The Composition of Substitutions	166
0.7.1	Pure-variable Substitutions	169
0.7.2	Syntactic Unification	176
1	Introduction	189
2	Propositional Logic Syntax	206
2.1	Propositions in Natural Languages	214

2.2	Translation of Natural Language Statements	221
2.3	Associativity and Precedence of Operators	230
3	Semantics of Propositional Logic	243
4	Logical and Algebraic Concepts	257
4.1	Some Meta-Logical Concepts	259
5	Identities and Normal Forms	270
6	Tautology Checking	289
7	Binary Decision Diagrams	308
8	Propositional Unsatisfiability & Resolution	349
8.1	Space Complexity of Propositional Resolution.	360
8.2	Time Complexity of Propositional Resolution	361
9	Analytic Tableaux	370
10	Consistency & Completeness	382

11	The Compactness Theorem	398
12	Maximally Consistent Sets	408
13	Formal Theories	431
14	Proof Theory: Hilbert-style	447
15	Derived Rules	468
16	The Hilbert System: Soundness	491
17	The Hilbert System: Completeness	502
18	Introduction to Predicate Logic	514
18.1	Predicates in Natural Languages	529
19	The Semantics of Predicate Logic	546
20	Substitutions	561
21	Models, Satisfiability and Validity	578

21.1 An excursion into Ordinals	584
21.2 Some more Model Theory	607
22 Structures and Substructures	610
23 Predicate Logic: Proof Theory	627
24 Predicate Logic: Proof Theory (Contd.)	650
25 Existential Quantification	665
26 Normal Forms	682
27 Skolemization	703
28 Substitutions and Instantiations	724
29 Unification	736
29.1 Syntactic Unification as equation solving	738
30 Resolution in FOL	769

31 More on Resolution in FOL	783
32 Resolution: Soundness and Completeness	797
33 Resolution and Tableaux	816
34 Completeness of Tableaux Method	827
35 Completeness of the Hilbert System	836
35.1 Model-theoretic and Proof-theoretic Consistency	839
36 First-Order Theories	850
37 Towards Logic Programming	876
38 Verification of Imperative Programs	896
39 Verification of WHILE Programs	915
40 Many-sorted FOL	945
41 Data Structures	956

41.1	Introduction: Compound Data Structures	959
41.2	Arrays as functions	965
41.3	Inductively Defined Data-types	969
41.4	Lists: The First-order Theory	970
41.5	Stacks: The First-order Theory	985
41.6	Queues: The First Order Theory	992
41.7	Binary Trees: The First-order Theory	999
42	Miscellaneous Examples	1010

List of Slides

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

- 10. Syntactic Identity
- 11. Abstract Syntax Trees
- 12. Subformulae
- 13. Atoms in a Formula
- 14. Degree of a Formula
- 15. Size of a Formula
- 16. Height of a Formula

✓ 3: Semantics of Propositional Logic

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

• 4: Logical and Algebraic Concepts

- 1. Logical Consequence: 1
- 2. Logical Consequence: 2
- 3. Other Theorems
- 4. Logical Implication
- 5. Implication & Equivalence
- 6. Propositional Logic is An Algebra
- 7. Logical Equivalence as a Congruence

• 5: Identities and Normal Forms

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

• 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. Rewriting Conditionals, Biconditionals
 - 12. Convert to NNF
 - 13. Convert to CNF
 - 14. Falsifying CNF
- 7: Binary Decision Diagrams
 - 1. Binary Decision Diagrams
 - 2. The if-then-else Operator 1
 - 3. The if-then-else Operator 2
 - 4. Boolean Expressions: Variables
 - 5. Boolean Expressions: terms
 - 6. if-then-else: test, high and low
 - 7. Examples: Boolean expressions
 - 8. More on if-then-else
 - 9. Functional Completeness with the new operator
 - 10. Another Normal Form: INF
 - 11. The Shannon Expansion
 - 12. Example: Truth table to Decision tree
 - 13. Example:1
 - 14. Example:2
 - 15. Example:3
 - 16. Example:4,5

- 17. BDDs
 - 18. OBDDs
 - 19. ROBDDs
 - 20. Order of Variables
 - 21. ROBDDs: Representing Boolean functions
 - 22. The Canonicity Lemma
 - 23. Consequences of Canonicity
- **8: 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

• **9: 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
- 10: 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. Soundness of the Tableau Method
 - 10. Completeness of the Tableau Method
- 11: The Compactness Theorem
 - 1. Satisfiability of Infinite Sets
 - 2. The Compactness Theorem
 - 3. Inconsistency
 - 4. Consequences of Compactness



• 12: Maximally Consistent Sets

1. Consistent Sets
2. Maximally Consistent Sets
3. Properties of Finite Character: 1
4. Properties of Finite Character: Examples:1
5. Properties of Finite Character: Examples:2
6. Properties of Finite Character: Compactness
7. Properties of Finite Character: Tukey's Lemma
8. Lindenbaum's Theorem

• 13: 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



• 14: Proof Theory: Hilbert-style

- 1. More About Formal Theories
- 2. The Law of The Excluded Middle
- 3. An Example Proof:1
- 4. An Example Proof:2
- 5. An Example Proof:3
- 6. An Example Proof:4
- 7. An Example Proof:5
- 8. Formal Proofs: 1
- 9. Formal Proofs: 2
- 10. Provability and Formal Proofs
- 11. The Deduction Theorem
- 12. About Formal Proofs
- 15: 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
- 16: 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
- 17: The Hilbert System: Completeness
 - 1. Towards Completeness
 - 2. Towards Truth-tables
 - 3. The Truth-table Lemma
 - 4. The Completeness Theorem
- 18: Introduction to Predicate Logic
 - 1. Predicate Logic: Introduction-1
 - 2. Predicate Logic: Introduction-2
 - 3. Internal Structure of Sentences
 - 4. Internal Structure of Sentences
 - 5. Parameterisation-1
 - 6. Parameterisation-2
 - 7. Predicate Logic: Introduction-3
 - 8. Predicate Logic: Symbols
 - 9. Predicate Logic: Signatures
 - 10. Predicate Logic: Syntax of Terms
 - 11. Predicate Logic: Syntax of Formulae

- 12. Precedence Conventions
 - 13. Predicates: Abstract Syntax Trees
 - 14. Subterms
 - 15. Variables in a Term
 - 16. Bound Variables And Scope
 - 17. Bound Variables And Scope: Example
 - 18. Scope Trees
 - 19. Free Variables
 - 20. Bound Variables
 - 21. Closure
- 19: 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

- 20: 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
- 21: Models, Satisfiability and Validity
 - 1. Satisfiability
 - 2. Models and Consistency
 - 3. Examples of Models:1
 - 4. Other Models
 - 5. Examples of Models:2
 - 6. Examples of Models:3
 - 7. Logical Consequence
 - 8. Validity
 - 9. Validity of Sets of Formulae
 - 10. Negations of Semantical Concepts
 - 11. Independence
 - 12. Example of Independence
- 22: 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
- 23: 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. Theories of Predicate Calculus

- 11. Axioms for Equality
- 12. Symmetry and Transitivity
- 13. Symmetry of Equality
- 14. Transitivity of Equality
- 24: 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
- 25: Existential Quantification
 - 1. Existential Quantification
 - 2. Existential Elimination
 - 3. Remarks on Existential Elimination
 - 4. Restrictions on Existential Elimination
 - 5. Equivalence of Proofs
- 26: Normal Forms
 - 1. Natural Deduction: 6
 - 2. Moving Quantifiers
 - 3. Quantifier Movement
 - 4. More on Quantifier Movement

- 5. Quantifier-Free Formulae
- 6. Prenex Normal Forms
- 7. The Prenex Normal Form Theorem
- 8. Prenex Conjunctive Normal Form
- 9. The Herbrand Algebra
- 10. Terms in a Herbrand Algebra
- 11. Herbrand Interpretations
- 12. Herbrand Models
- 13. Ground Quantifier-free Formulae
- 27: Skolemization
 - 1. Skolemization
 - 2. Skolem Normal Forms
 - 3. SCNF
 - 4. Ground Instances
 - 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
- 28: Substitutions and Instantiations
 - 1. Substitutions Revisited
 - 2. Some Simple Facts

- 3. Ground Substitutions
- 4. Composition of Substitutions
- 5. Substitutions: A Monoid
- 29: 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
- 30: 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 and Herbrand's Theorem
- 10. Resolution in FOL
- 11. The Resolution Rule for FOL
- 31: More on Resolution in FOL
 - 1. Standardizing Variables Apart
 - 2. Factoring
 - 3. Example: 1
 - 4. Example: 2
 - 5. Refutation
 - 6. Refutations: Using the Herbrand Universe
- 32: 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
- 33: Resolution and Tableaux
 - 1. FOL: Tableaux
 - 2. FOL: Tableaux Rules
 - 3. FOL Tableaux: Example 1
 - 4. FOL Tableaux: Example 1 Contd
 - 5. First-Order Tableaux
 - 6. FOL Tableaux: Example 2
 - 7. FOL Tableaux: Example 2 Contd
- 34: 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
- 35: Completeness of the Hilbert System
 - 1. Deductive Consistency
 - 2. Models of Deductively Consistent Sets
 - 3. Deductive Completeness
 - 4. The Completeness Theorem
- 36: First-Order Theories

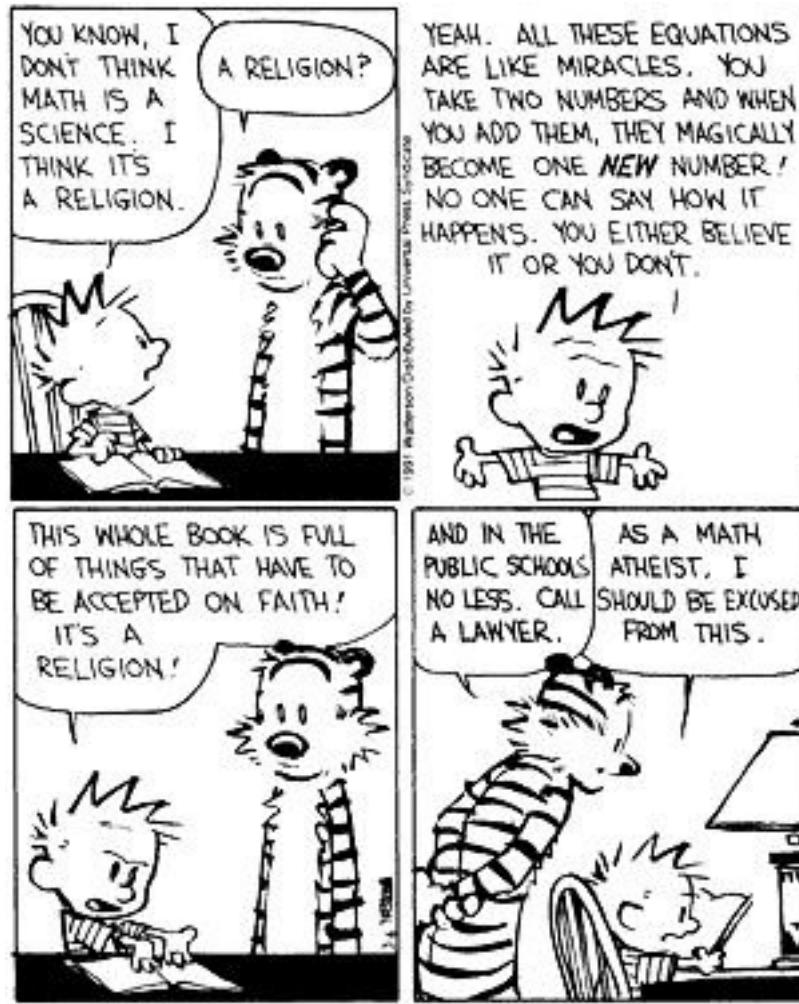
- 1. (Simple) Directed Graphs
- 2. (Simple) Undirected Graphs
- 3. Irreflexive Partial Orderings
- 4. Irreflexive Linear Orderings
- 5. (Reflexive) Preorders
- 6. (Reflexive) Partial Orderings
- 7. (Reflexive) Linear Orderings
- 8. Equivalence Relations
- 9. Peano's Postulates
- 10. The Theory of The Naturals
- 11. Notes and Explanations
- 12. Finite Models of Arithmetic
- 13. A Non-standard Model of Arithmetic
- 14. Z-Chains
- 15. The Principle of Mathematical Induction
- 16. First-order Arithmetic
- 17. Extending First-order Arithmetic
- 37: Towards Logic Programming
 - 1. Reversing the Arrow
 - 2. Arrow Reversal
 - 3. Horn Clauses
 - 4. Program or Rule Clause

- 5. Goal clauses
 - 6. Logic Programs
 - 7. Sorting in Logic
 - 8. Prolog: Sort
 - 9. Prolog: Merge Sort
 - 10. Prolog: Quick Sort
 - 11. Prolog: SEND+MORE=MONEY
 - 12. Prolog: Naturals
- 38: 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
 - 39: 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
- 40: Many-sorted FOL
 - 1. Sortedness
 - 2. Many-Sorted Logic: Symbols
 - 3. Many-Sorted Signatures
 - 4. Many-Sorted Signature: Terms
 - 5. Many-Sorted Predicate Logic
 - 6. Reductions: Guardedness
 - 7. Reductions: Bounded Quantification
- 41: Abstract Data Types
 - 1. Pairs
 - 2. The Array Structure: Signature
 - 3. The Array Structure: Axioms

- 4. The List Structure: Signature
 - 5. The List Structure Axioms: 1
 - 6. The List Structure Axioms: 2
 - 7. The List Structure: Induction
 - 8. The List Structure: Acyclicity 1
 - 9. The List Structure: Acyclicity 2
 - 10. The Stack Structure: Signature
 - 11. The Stack Structure Axioms: 1
 - 12. The Stack Structure Axioms: 2
 - 13. The Stack Structure: Induction
 - 14. The Queue Structure: Signature
 - 15. The Queue Structure Axioms: 1
 - 16. The Queue Structure Axioms: 2
 - 17. The Queue Structure: Induction
 - 18. Binary Trees: Signature
 - 19. The Binary Tree Axioms: 1
 - 20. The Binary Tree Axioms: 2
 - 21. The Binary Tree Axioms: 3
 - 22. The Binary Tree Induction Rule
- 42: Miscellaneous Examples
 - 1. Random Examples

0. Background



©Bill Watterson

0.1. Mathematical 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.

0.1.1. 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.

0.1.2. 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

$$4. A \cap B = B \cap A$$

Commutativity

$$5. A \cap A = A$$

Idempotence

Other properties

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

Distributivity of \cap over \cup

$$2. A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Distributivity of \cup over \cap

$$3. \sim (A \cup B) = \sim A \cap \sim B$$

De Morgan's law $\sim \cup$

$$4. \sim (A \cap B) = \sim A \cup \sim B$$

De Morgan's law $\sim \cap$

$$5. A \cap (\sim A \cup B) = A \cap B$$

Absorption \cup

$$6. A \cup (\sim A \cap B) = A \cup B$$

Absorption \cap

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: \cup

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 .

0.1.3. 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

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

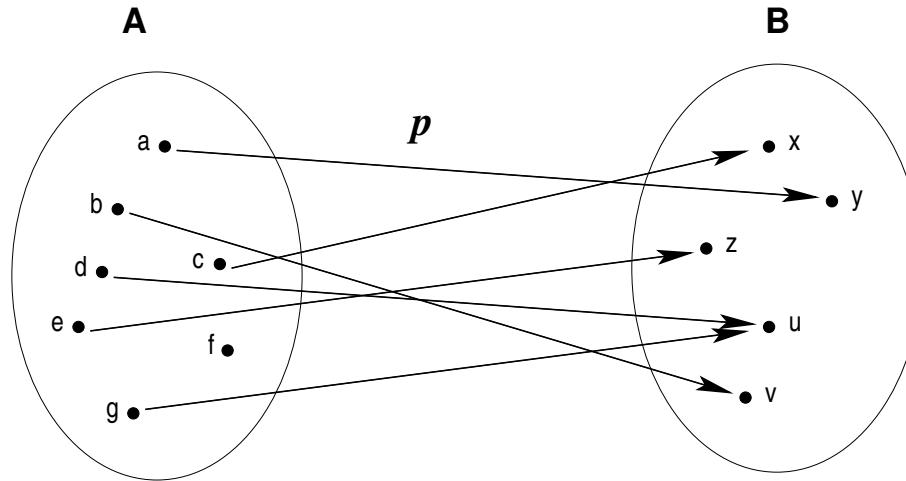


Figure 1: A partial function (*Why is it partial?*)

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 $0 \leq r < b$

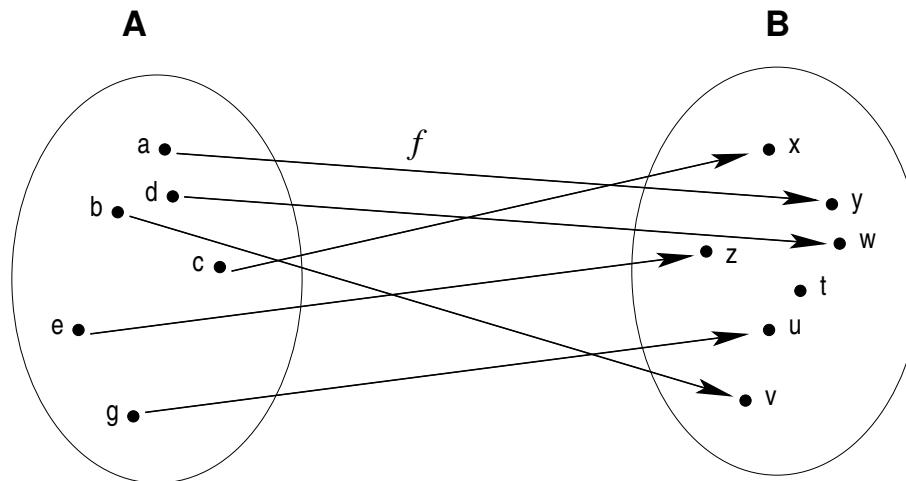


Figure 2: An injective function (*Why is it injective?*)

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 $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.

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 a (total) function of the type $\vec{a} : \{1, 2, \dots, n\} \rightarrow A$. We normally denote such a sequence of length

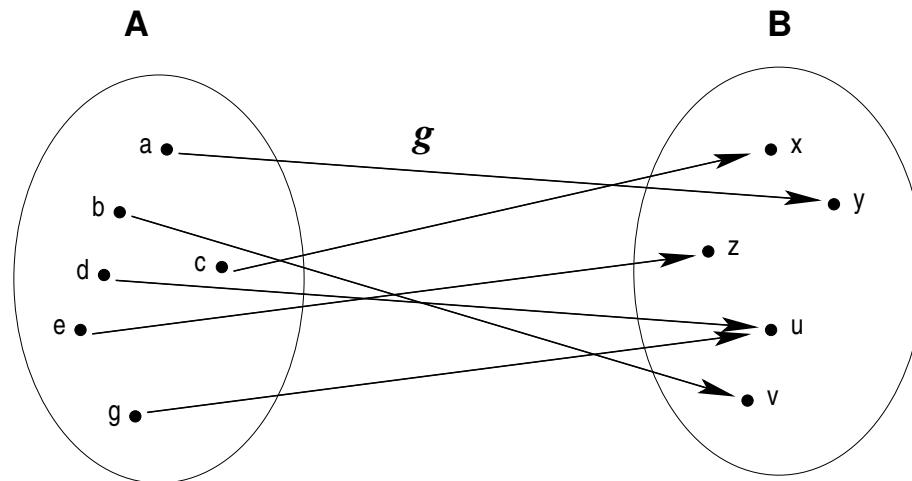


Figure 3: A surjective function (*Why is it surjective?*)

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 $[a_0, a_2, \dots, a_{n-1}]$. 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.

Definition 0.5 An **alphabet** is a finite set of symbols also called **letters**. Any finite sequence of

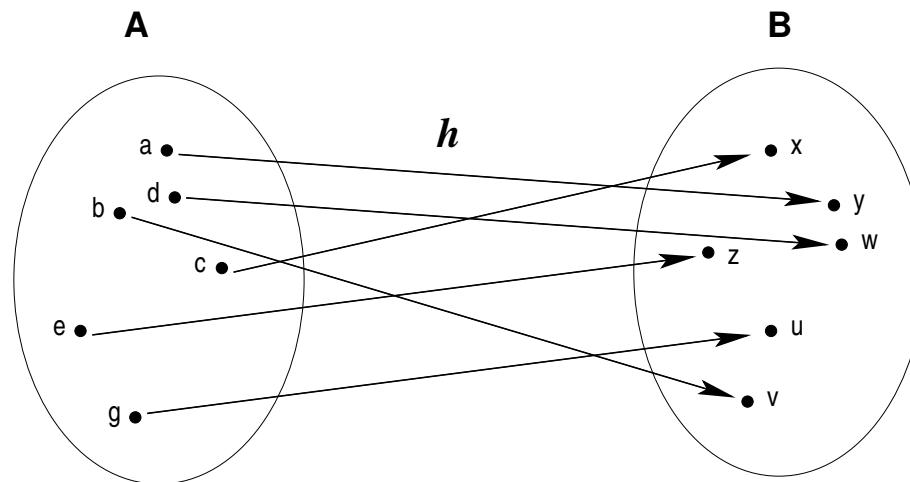


Figure 4: An bijective function (*Why is it bijective?*)

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

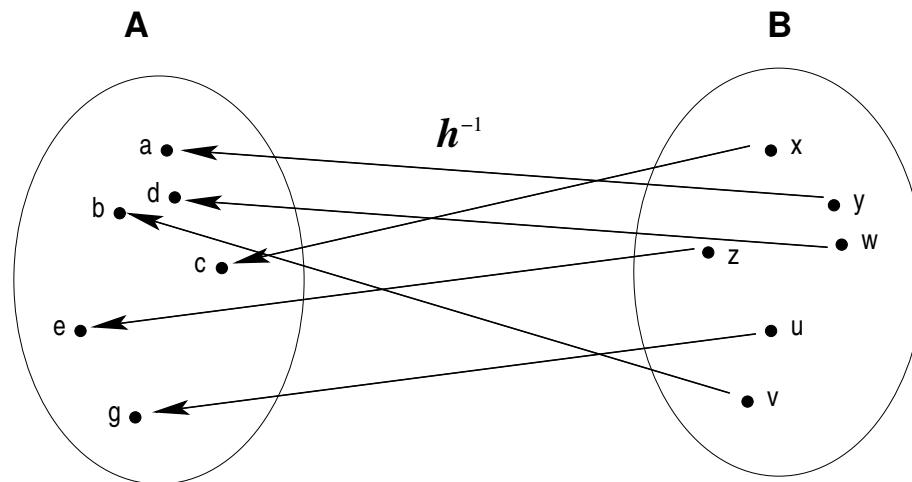


Figure 5: The inverse of the bijective function in Fig 4 (Is it bijective?)

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 .

0.1.4. 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 0.1.8.

Theorem 0.7 (Schroeder-Bernstein Theorem) *Let A and B be sets and let $f : A \xrightarrow{1-1} B$ and $g : B \xrightarrow{1-1} A$ 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 $b_1 \in B$. To define the origin of such an element a_0 we 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. $h : A \xrightarrow[\text{onto}]{I\text{-}I} B$ 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

0.1.5. 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 for all a and b , $a \neq b$, $(a, b) \in \mathcal{R}$ implies $(b, a) \notin \mathcal{R}$.³
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.

³An equivalent definition used in most books is: \mathcal{R} is **antisymmetric** if and only if $(a, b), (b, a) \in \mathcal{R}$ implies $a = b$.

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.

0.1.6. 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**⁴ 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.

⁴also called **total order**

Definition 0.11 A partially ordered set, or poset $\langle A, \leq \rangle$ consists of a set A together with a partial order relation \leq on A .

Fact 0.12 If $\langle A, \leq \rangle$ is a poset, then so is $\langle A, \geq \rangle$ where $\geq = \leq^{-1}$.

Notation. Given a poset $\langle A, \leq \rangle$ and $a, b \in A$, we sometimes write

- $b \geq a$ to mean $a \leq b$,
- $a < b$ to mean $a \leq b$ and $a \neq b$ and
- $b > a$ to mean $a < b$.

Fact 0.13 If $\langle A, \leq \rangle$ is a poset, then $<$ and $>$ are strict partial orders.

For any set A (empty or non-empty) we have that $\langle 2^A, \subseteq \rangle$ is also a poset and in fact, the partial ordering relation \leq on A can be characterised (upto isomorphism) by the subset relation.

Definition 0.14 Given two posets $\langle A, \leq_A \rangle$ and $\langle B, \leq_B \rangle$, a function $f : A \longrightarrow B$ is said to be **order-preserving** if and only if for all $a, a' \in A$, $a \leq_A a'$ implies $f(a) \leq_B f(a')$. The two posets are said

to be (order-) isomorphic if there exists an order-preserving bijection between them. We denote this fact by $\langle A, \leq_A \rangle \cong \langle B, \leq_B \rangle$.

Notice that if $f : A \rightarrow B$ is a bijection then so is $f^{-1} : B \rightarrow A$.

Lemma 0.15 *For each poset $\langle A, \leq \rangle$ there exists a set $\mathcal{A} \subseteq 2^A$ such that $\langle A, \leq \rangle \cong \langle \mathcal{A}, \subseteq \rangle$.*

Proof: For each $x \in A$ let $A_x = \{a \in A \mid a \leq x\}$. Define the set $\mathcal{A} = \{A_x \mid x \in A\} \subseteq 2^A$ and the function $f : A \rightarrow \mathcal{A}$ such that $f(x) = A_x$ for each $x \in A$. It is easy to see that f is bijective and order-preserving i.e. for all $x, y \in A$, $x \leq y$ if and only if $A_x \subseteq A_y$. QED

0.1.7. Well-founded Sets and Well-ordered Sets

We discuss well-orders since an important induction principle (theorem 0.61) depends upon the notion of a well-ordering and generalises the principle of mathematical induction.

Definition 0.16 *Let $\langle A, \prec \rangle$ be a set where $\prec \subseteq A \times A$ is an irreflexive, asymmetric binary relation on A . Let $B \neq \emptyset, B \subseteq A$. An element $b \in B$ is said to be **minimal** if there exists no $a \in B$ such that $a \prec b$. $\langle A, \prec \rangle$ is called **well-founded** if every nonempty subset of A has a minimal element. Equivalently we say that \prec on A is a **well-founded** relation.*

Note that in general, \prec need not be transitive. For example the “immediate-predecessor-of” relation on the naturals given by $\prec_{pred} = \{(n, n + 1) \mid n \in \mathbb{N}\}$ is an example of an irreflexive, asymmetric and intransitive relation which is well-founded.

Definition 0.17 Given a set $\langle A, \prec \rangle$ as in definition 0.16, an **infinite descending chain** is a nonempty subset of elements $\{a_i \in A \mid i \geq 0, a_{i+1} \prec a_i\}$. We normally write this as

$$a_0 \succ a_1 \succ a_2 \succ \dots$$

where $\succ = \prec^{-1}$.

Lemma 0.18 A set $\langle A, \prec \rangle$ is well-founded if and only if it has no infinite descending chain.

Proof:

(\Rightarrow). Assume $\langle A, \prec \rangle$ is well-founded and there is a an infinite descending chain $A' = \{a_i \in A \mid i \geq 0, a_i \succ a_{i+1}\} \subseteq A$. Clearly A' contains no minimal element, which is a contradiction.

(\Leftarrow). Assume there is no subset $A' = \{a_i \in A \mid i \geq 0, a_i \succ a_{i+1}\} \subseteq A$. If $\langle A, \prec \rangle$ is not well-founded, there exists a nonempty subset $B \subseteq A$ which has no minimal element. Consider any $b_0 \in B$. Since b_0 is not minimal there exists $b_1 \in B$ such that $b_0 \succ b_1$. Again b_1 is not

minimal, so there must be a $b_2 \in B$ with $b_1 \succ b_2$. Proceeding in this fashion we find that for each $b_i \prec \dots \prec b_1 < b_0$, there exists a $b_{i+1} \in B$ such that $b_{i+1} \prec b_i < \dots < b_1 < b_0$. We may thus construct a set $B' = \{b_i \in B \mid i \geq 0, b_i > b_{i+1}\} \subseteq B \subseteq A$ which contradicts the assumption that there is no such subset.

QED

The set $B' = \{b_i \in B \mid i \geq 0, b_i > b_{i+1}\}$ is an example of an “*infinite descending chain*”

$$\dots \prec b_{i+1} \prec b_i \prec \dots \prec b_1 \prec b_0$$

We usually say that a well-founded set has no “*infinite descending chain*”.

We have already seen in definition 0.10 that a linear or total order is a connected partial order. We now use this and definition 0.16 in defining a well-ordered set. In general given a partial ordering \leq on a set with a minimal elements we could use the irreflexive, asymmetric subset $<$ of the partial order defined by $x < y$ iff $x \leq y$ and $y \not\leq x$ to define a well-founded set wherever possible. For example, $<$ on the Naturals is such a well-order. However, $<$ on the integers is not a well-order because there are subsets of the integers which have no minimal element. Notice that the real interval $[0, 1]$ even though totally ordered and having a minimal element, is not well-founded because there are subsets which have infinite descending chains such as the open interval $(0, 1)$. Similarly for any

two rational numbers $a < b$, the closed interval of rationals $[a, b]$ is also not well-founded.

Definition 0.19 A **well-ordering** of a set is a well-founded total ordering of the set. A well-ordered set is sometimes called a **woset**.

Fact 0.20 Let $\langle A, \prec \rangle$ be a (nonempty) well-ordered set.

1. Every nonempty subset of A has a unique least element.
2. A has a unique least element.

0.1.8. Infinite Sets: Countability and Uncountability

Definition 0.21 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.

Fact 0.22 A finite union of finite sets is finite. That is, for any $n \in \mathbb{N}$, if $\{A_i \mid 0 \leq i < n, A_i \text{ is a finite set}\}$ is a finite collection of finite sets, then $A = \bigcup_{i=0}^{n-1} A_i$ is a finite set.

Definition 0.23 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 0.24 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 $p :$
$$\mathbb{N} \xrightarrow[\text{onto}]{1-1} \mathbb{P}, \text{with } p(m) \stackrel{\text{df}}{=} m + 1.$$
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.

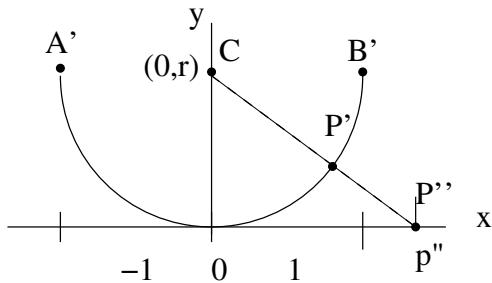


Figure 6: Bijection between the arc $A'B'$ and the real line

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 $z : \mathbb{Z} \xrightarrow[\text{onto}]{I-I} \mathbb{N}$ 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 0.25 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”⁵. This semicircle has a radius $r = \frac{b-a}{\pi}$. 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.

⁵Many of the logical problems with proofs in mathematics also occur because of the use of loose and improperly defined terms such as “bend” and “obvious” even though their meaning might appear to be intuitively clear. A more precise statement might be that for each line segment \overline{AB} of non-zero length, there exists a semicircle $\widehat{A'B'}$ (with radius $\frac{|AB|}{\pi}$) of the same length with the “obvious” 1-1 correspondence between them which maps any point P on \overline{AB} with $0 < AP < b - a$ to a point P' on $\widehat{A'B'}$ such that the length of the arc $\widehat{A'P'}$ is AP . Words like “obvious” and “trivial” are also standard terms used by students in examinations when they know fully well that there are gaps in their proof which they are unable to fill satisfactorily.

Definition 0.26 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 0.27 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.

Definition 0.28 A set is **at most countable** if it is finite or countable.

The following is an interesting characterisation of at most countable sets.

Theorem 0.29 (Equivalents of at most countability). *The following statements are equivalent for any nonempty set A .*

1. *A is at most countable.*
2. *There is a surjective map $f : \mathbb{N} \xrightarrow{\text{onto}} A$.*
3. *There is an injective map $g : A \xrightarrow{1-1} \mathbb{N}$.*

Proof: We prove it by showing that

1. statement 1 implies statement 2
2. statement 2 implies statement 3
3. statement 3 implies statement 1

- *statement 1 implies statement 2.* If A is countable it is clear that there exists a bijection $f : \mathbb{N} \xrightarrow{\text{onto}} A$ which is surjective. If $A = \{a_0, \dots, a_{n-1}\}$ is finite and has $n > 0$ elements then the function $f : \mathbb{N} \longrightarrow A$ such that $f(m) = \begin{cases} a_m & \text{if } 0 \leq m < n \\ a_0 & \text{if } m \geq n \end{cases}$ is clearly surjective.

- *statement 2 implies statement 3.* Let $f : \mathbb{N} \xrightarrow{\text{onto}} A$ be surjective. We then define the following injective function $g : A \xrightarrow{1-1} \mathbb{N}$ such that $g(a) =$ the least m such that $f(m) = a$.
- *statement 3 implies statement 1.* Let $g : A \xrightarrow{1-1} \mathbb{N}$ be an injective map. We may use this injective map to well-order the elements of A , so that $a \sqsubset b$ if and only if $g(a) < g(b)$. If A is finite there is nothing to prove. If it is not finite then let $A_0 = A - \emptyset = A$ has a (unique) least element a_0 under the ordering \sqsubseteq . For each positive integer $j > 0$, let $A_j = A - \{a_0, \dots, a_{j-1}\} \subseteq A$, let a_j be the (unique) least element of A_j . The function $h : a_j \mapsto j$ is a bijection between A and \mathbb{N} , showing that A is countable.

QED

Theorem 0.30 \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

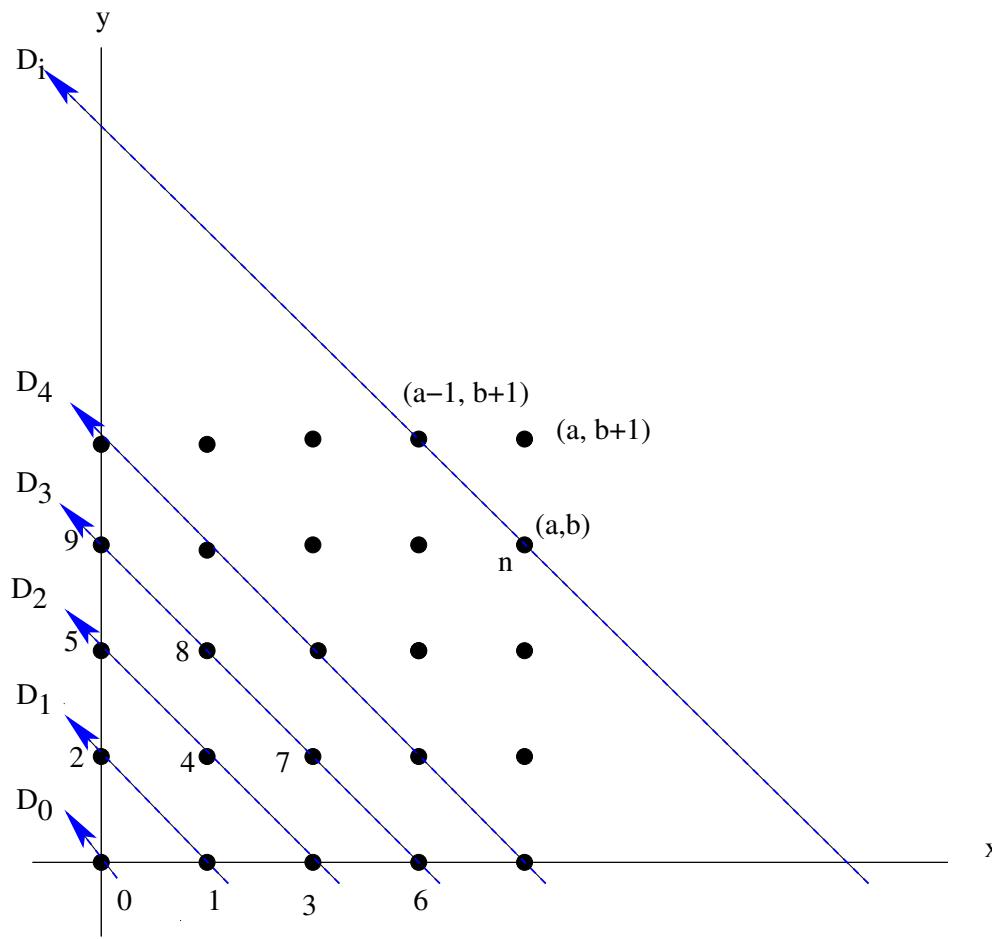


Figure 7: Counting “lattice-points” on the “diagonals”

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

↪ 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 $\overrightarrow{D_0}, \overrightarrow{D_1}, \dots, \overrightarrow{D_{i-1}}$ ⁶

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 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*

⁶Under 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.

numbers"!)

QED

Theorem 0.31 *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 0.30 it follows that A_∞ is countable. QED

Example 0.32 *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 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 $\text{ord}'(\text{“}\neg\text{”}) = 1$, $\text{ord}'(\text{“}(“}) = 2$, $\text{ord}'(\text{“})”} = 3$, $\text{ord}'(\text{“}\rightarrow\text{”}) = 4$, and $\text{ord}'(\text{“}A\text{”}) = \text{ord}(\text{“}A\text{”}) + 4$, for every $A \in \mathbb{A}$. Let $\text{Syms} = \mathbb{A} \cup \{\text{“}\neg\text{”}, \text{“}(“}, \text{“})”}, \text{“}\rightarrow\text{”}\}$. 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 string⁷ belonging to Syms^* . It is possible to assign a unique positive integer to

⁷This includes even arbitrary strings which are not part of the language. For example, you may have strings such as “ $)\neg($ ”.

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 “ \emptyset ” denotes the empty string.

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

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.

⊤ 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. \dashv

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 0.33 (Cantor’s diagonalization). *The powerset of \mathbb{N} (i.e. $\mathcal{P}(\mathbb{N})$, 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 $2^{\mathbb{N}}$. A simple bijection extends the mapping $i \mapsto i + 1$ to sets in $2^{\mathbb{N}}$ i.e. define $f : 2^{\mathbb{N}} \xrightarrow[\text{onto}]{1-1} 2^{\mathbb{P}}$ such that for each $A \in 2^{\mathbb{N}}$, $f(A) = \{a + 1 \mid a \in A\}$.

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.

⊤ 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. \dashv

QED

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

Theorem 0.34 (The Powerset theorem). *There is no 1-1 correspondence between a set and its powerset.*

Proof: Let A be any set and let 2^A be its powerset. Assume that $g : A \xrightarrow[\text{onto}]{} 2^A$ is a 1-1 correspondence between A and 2^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 0.35 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 $f : \mathcal{P}^{\mathbb{P}} \xrightarrow{1-1} [0, 1)$ and $g : [0, 1) \xrightarrow{1-1} \mathcal{P}^{\mathbb{P}}$ 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 number of the form $0.b_1b_2\dots b_{i-1}0\bar{1}$ equals the real number $0.b_1b_2\dots b_{i-1}\bar{1}0$ which is normal. Every real number in $[1, 0)$ has a unique normal representation. Now consider the function defined by $g(0.b_1b_2b_3\dots) = \{i \in \mathbb{P} \mid b_i = 1\}$. 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>	<i>QED</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>	

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 $f : A \xrightarrow[\text{onto}]^{1\text{-}1} B$ and $g : B \xrightarrow[\text{onto}]^{1\text{-}1} C$, their composition is the function $g \circ f : A \xrightarrow[\text{onto}]^{1\text{-}1} C$.
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 disprove⁸ 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) $\mathcal{R}; (\mathcal{S} \cap \mathcal{T}) = (\mathcal{R}; \mathcal{S}) \cap (\mathcal{R}; \mathcal{T})$

⁸that is, find an example of appropriate relations which actually violate the equality

(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,
- (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 or disprove that for any nonempty relation \mathcal{R} on a nonempty set A ,
- $\mathcal{S} = \mathcal{R}^* \cup (\mathcal{R}^*)^{-1}$ is an equivalence relation,
 - $\mathcal{T} = (\mathcal{R} \cup \mathcal{R}^{-1})^*$ is an equivalence relations.
 - 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.

0.2. 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

0.2.1. 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 0.36 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 (PMI1)

A property \mathfrak{p} holds for all natural numbers provided

Basis. \mathfrak{p} holds for 0, and

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

\mathfrak{p} holds for $n - 1$ implies \mathfrak{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. Alternatively the principle could be expressed as follows, by substituting “ $n \geq 0$ ” for “ $n > 0$ ”, “P holds for n” for “P holds for $n - 1$ ” and “P holds for $n + 1$ ” for “P holds for n” in the induction step.

Principle of Mathematical Induction – Version 1' (PMI1')

A property p holds for all natural numbers provided

Basis. p holds for 0, and

Induction step. For arbitrarily chosen $n \geq 0$,

p holds for n implies p holds for $n + 1$.

Example 0.37 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 \mathfrak{p} holds for all natural numbers $n \geq k$ for some natural number k , provided

Basis. \mathfrak{p} holds for k , and

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

\mathfrak{p} holds for $n - 1$ implies \mathfrak{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 0.38 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 0.39 *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 \mathfrak{p} that holds for all naturals greater than or equal to some given k may be transformed equivalently into a property \mathfrak{q} , which reads exactly like \mathfrak{p} except that all occurrences of “ n ” in \mathfrak{p} are systematically replaced by “ $n + k$ ”. We may then prove the property \mathfrak{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 0.40 *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 versions (PMI1 or PMI1').

0.2.2. 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 0.41 *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 Fibonacci⁹ sequence.

One of the properties of the Fibonacci sequence is that the sequence converges to the “golden ratio”¹⁰. 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

⁹named after Leonardo of Fibonacci.

¹⁰one of the solutions of the equation $x^2 = x + 1$. It was considered an aesthetically pleasing aspect ratio for buildings in ancient Greek architecture.

the principle itself seems “weaker” than the previous versions.

Principle of Complete Induction (PCI)

A property \mathfrak{p} holds for all natural numbers provided

Basis. \mathfrak{p} holds for 0.

Induction step. For an arbitrary $n > 0$

\mathfrak{p} holds for every m , $0 \leq m < n$ implies \mathfrak{p} holds for n .

Example 0.42 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 0.42 but this time we use the principle of mathematical induction instead.

Example 0.43 Let $\mathfrak{p}(n)$ denote the property

$$\text{"}F(n + 1) \leq \phi^n\text{"}$$

Rather than prove the original statement “For all n , $\mathfrak{p}(n)$ ” we instead consider the property $\mathfrak{q}(n)$ which we define as

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

and prove the statement “For all n , $\mathfrak{q}(n)$ ”. 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 $\mathfrak{q}(n - 1)$, holds for an arbitrarily chosen $n > 0$, we need to prove the property \mathfrak{q} for n . But for this it suffices to prove the property \mathfrak{p} for n , since $\mathfrak{q}(n)$ is equivalent to the conjunction of $\mathfrak{q}(n - 1)$ and $\mathfrak{p}(n)$. Hence we prove the property $\mathfrak{p}(n)$.

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

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 0.44 *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 0.42. We merely replace each occurrence of the original property in the form $\mathfrak{p}(n)$ by $\mathfrak{q}(n)$, where the property \mathfrak{q} is defined as

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

Since $\mathfrak{q}(0)$ is the same as $\mathfrak{p}(0)$ 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 , $0 \leq m \leq n - 1$, $\mathfrak{p}(m)$

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

For arbitrarily chosen $n > 0$, $\mathfrak{q}(n - 1)$

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 \mathfrak{q} for all naturals n . Since $\mathfrak{q}(n)$ logically implies $\mathfrak{p}(n)$ it follows that the proof of property \mathfrak{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 \mathfrak{p} may also be considered as defining a set $S = \{x \mid x \text{ satisfies property } \mathfrak{p}\}$. Therefore if a property

φ 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 0.45 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.
2. 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 N^n are all countable.

0.2.3. 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 0.46 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 , \cup 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 0.46.

Lemma 0.47 $A \in \mathcal{X}$ where A and \mathcal{X} are as in definition 0.46.

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 0.50 that such equations may be solved for their unique smallest solution.

Definition 0.48 Let \mathbb{U}, B, K be as in definition 0.46. 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 0.46.

Example 0.49 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”, “ $\text{add}(0, 1)$ ”, “ $\text{mult}(\text{add}(1, 0), \text{mult}(1, 1))$ ” are all am-expressions. On the other hand,

1. “ $\text{add}(0)$ ” 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 \cup be as in definition 0.46. 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 0.50 *Let B , K and \cup be as in definition 0.46 and A_∞ be as in equation (4). Then $A = A_\infty$.*

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

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

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 0.46. 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 \blacksquare

Lemma 0.50 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 0.51 *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 0.52 *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 0.52 shows that mathematical induction is merely a particular case of structural induction.

Example 0.53 *The language of minimal logic was defined in example 0.32. 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 \overrightarrow{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

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

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.

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

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

Claim. Each of the sets M_k , \overleftarrow{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 \overleftarrow{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 , \overleftarrow{N}_{k+1} and \overrightarrow{N}_{k+1} is countably infinite. Their union is clearly countably infinite by the following 1-1 correspondence.

$$\text{num}_{k+1}(\mu_{k+1}) = \begin{cases} 3 \times \text{num}_k(\mu_{k+1}) & \text{if } \mu_{k+1} \in M_k \\ 3 \times \text{num}_k(\mu_k) + 1 & \text{if } \mu_{k+1} \equiv \neg \mu_k \in \overleftarrow{N}_{k+1} \\ 3 \times \text{diag}(\text{num}_k(\mu_k), \text{num}_k(\nu_k)) + 2 & \text{if } \mu_{k+1} \equiv \mu_k \rightarrow \nu_k \in \overrightarrow{N}_k \end{cases}$$

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

Principle of Structural Induction (PSI)

A property \mathfrak{p} holds for all elements of A provided

Basis. \mathfrak{p} is true for all basis elements.

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

Proof: Let \mathfrak{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 \mathfrak{p} , i.e. $C = \{a \in A \mid \mathfrak{p} \text{ holds for } a\}$. 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 \mathfrak{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 \mathfrak{p} holds for each of $a_1, \dots, a_{\alpha(f)}$. This implies by the induction step that \mathfrak{p} holds for a , contradicting the assumption that

a does not satisfy \mathfrak{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 \mathfrak{p} holds for every element of A . QED

Example 0.55 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 prefix¹¹ 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 0.54) to prove that property \mathbf{P} holds for all

¹¹ 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 0.52 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 0.51) of each element in an inductively defined set.

Theorem 0.56 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 , \cup and let \mathfrak{p} be a property of elements of A that has been proved by the principle of structural induction. Let the property $\mathfrak{q}(n)$ for each natural number n be defined as

The property \mathfrak{p} holds for all elements of height n

Basis. The basis step $n = 0$ of the proof of property \mathfrak{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 $\mathfrak{q}(m)$ 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 \mathfrak{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 $\mathfrak{q}(n)$.

QED

Definition 0.57 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 0.58 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.

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

0.2.4. 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 0.59 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 $(s)_2$ 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 \mathfrak{p}_Z of the set Z .

$$\boxed{\mathfrak{p}_Z : Z = \{z \in 2^+ \mid (z)_2 \text{ is a non-negative multiple of } 3\}}$$

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

- Each $z \in Z$ is such that $(z)_2 = 3l$ for some $l \in \mathbb{N}$ i.e. $(z)_2$ 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 $(z)_2 = 3l$, for some $l \in \mathbb{N}$, then $(z0)_2 = 6l$ is also a multiple of 3 and for each $i \in I$, if $(i)_2 = 3m + 1$ for some $m \in \mathbb{N}$ then $i1 \in Z$ since $(i1)_2 = 2(3m + 1) + 1 = 3(2m + 1)$ is a multiple of 3.
- Likewise the definitions of I and T are such that for each $i \in I$, $(i)_2 = 3m + 1$ for some natural m and for each $t \in T$, $(t)_2 = 3n + 2$ for some natural n .

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

$$\mathfrak{p}_I : I = \{i \in 2^+ \mid (i)_2 = 3m + 1, m \in \mathbb{N}\}$$

and

$$\mathfrak{p}_T : T = \{t \in 2^+ \mid (t)_2 = 3n + 2, n \in \mathbb{N}\}$$

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

Lemma 0.60 *The elements of the sets Z , I and T satisfy properties \mathfrak{p}_Z , \mathfrak{p}_I and \mathfrak{p}_T respectively.*

Proof: We proceed by simultaneous induction on the lengths of strings of 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 \mathfrak{p}_Z ,
1. each $i' \in I$ such that $|i'| = k$ satisfies property \mathfrak{p}_I , and
2. each $t' \in T$ such that $|t'| = k$ satisfies property \mathfrak{p}_T ,

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

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

We may use the induction hypothesis then to prove the induction cases. We leave the rest of the proof to be completed by the interested reader.

QED

0.2.5. Well-ordered Induction

We have already defined the notions of well-founded (definition 0.16) and well-ordered (definition 0.19) sets. Well-ordered induction generalises the principle of mathematical induction to any set A which may be enumerated as a sequence $[a_i \mid i \in \mathbb{N}]$ indexed by the non-negative integers. There is an implicit total ordering in the elements such that $a_i < a_j$ if and only if $i < j$. Hence any property $\mathfrak{p}(a_i)$ of elements of the set may be regarded as a property $\mathfrak{q}(i)$ of the index of the element in the set. What makes induction applicable here is the fact that the set \mathbb{N} is “well-ordered” (see definition 0.19) by the relation $<$. Hence were a property \mathfrak{p} to fail for some element a_j in the set A , there would be a *least element* $a_i \leq a_j$ for which it would fail and which in turn translates to the property $\mathfrak{q}(j)$ and $\mathfrak{q}(i)$ failing in such a fashion that for all (if any) $i' < i$, $\mathfrak{q}(i')$ would hold.

Theorem 0.61 (The Principle of Well-ordered Induction). *Let $\langle A, \prec \rangle$ be a well-ordered set.*

Principle of Well-ordered Induction (PWI)

For any $X \subseteq A$, $X = A$ provided

Basis. The (unique) least element of A is in X and

Induction step. For each $a \in A$,

if (for every $a' \in A$, $[a' \prec a \text{ implies } a' \in X]$) then $a \in X^{12}$.

Proof: Suppose $X \neq A$. Then since $X \subseteq A$, we have $B = A - X \neq \emptyset$. Since $B \subseteq A$, and A is well-ordered, B has a unique smallest element $b \in B$. But $b \notin X$, even though for every element $a \prec b$, $a \in X$. But by the induction step $b \in X$, which is a contradiction. QED

Note that in the statement of the above theorem, the basis is included (vacuously) in the induction step. Hence the basis statement in this principle is actually superfluous.

Example 0.62 Consider the following function $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ (equivalently it is a functional

¹²The brackets and parentheses have been put in to avoid ambiguity in the statement.

program) such that

$$g(a, b) = \begin{cases} b & \text{if } a = 0, b > 0 \\ g(b \% a, a) & \text{if } a \neq 0 \end{cases}$$

where $b \% a$ denotes the remainder obtained on dividing b by a . We prove that $g(a, b)$ computes the greatest common divisor (\gcd) of a and b whenever at least one of a, b is non-zero¹³. As a functional program we need to show that it terminates i.e. the function is well-defined for all $(a, b) \neq (0, 0)$.

Proof: Whenever $0 \leq b < a$ we have that $g(a, b) = g(b, a)$ since $b \% a = b$ for $b < a$. Hence it suffices to prove the following claim.

Claim. For all $0 \leq a < b$, $g(a, b) = \gcd(a, b)$

\vdash For $0 = a < b$ it is clear that $g(a, b) = g(0, b) = b = \gcd(0, b)$. For $0 < a_0 < b_0$, we know that the set of arguments of the function g is given by

$$S_{(a_0, b_0)} = \{(a_i, b_i) \mid i \geq 0, a_{i+1} = b_i \% a_i, b_{i+1} = a_i\} \quad (5)$$

This set $S_{(a_0, b_0)}$ is well-ordered by the $<_{lex}$ relation on ordered pairs of numbers which is defined as follows.

$$(a, b) <_{lex} (a', b') \text{ iff } a < a' \text{ or } (a = a' \text{ and } b < b')$$

¹³ $\gcd(0, 0)$ is undefined

Hence if $>_{lex}$ denotes the converse of $<_{lex}$ we have that the sequence

$$(a_0, b_0) >_{lex} (a_1, b_1) >_{lex} \dots$$

is a finite descending sequence bounded (from below) by $(0, b_n)$ for some $b_n, n \geq 0$. By the properties of gcd (from any standard book on Elementary Number theory) we know that $\gcd(a_i, b_i) = \gcd(a_{i+1}, b_{i+1})$, for each $i \geq 0$. Hence $g(a_0, b_0) = \gcd(a_0, b_0)$ for all $0 < a_0 < b_0$. \dashv

QED

Exercise 0.2

1. Prove that version 1, 2 and 3 of PMI are mutually equivalent.
2. Prove that $\mathcal{D}^+ = Z \cup I \cup T$ in example 0.59.
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 , $\mathcal{T}(p) = 1$ if and only if tau 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 .

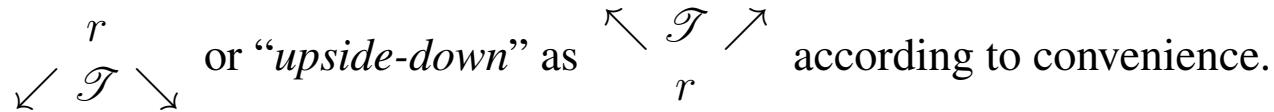
Definition 0.63

- 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



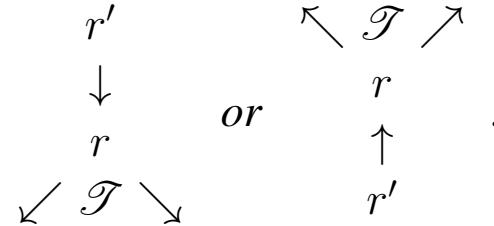
Facts 0.64

- 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.

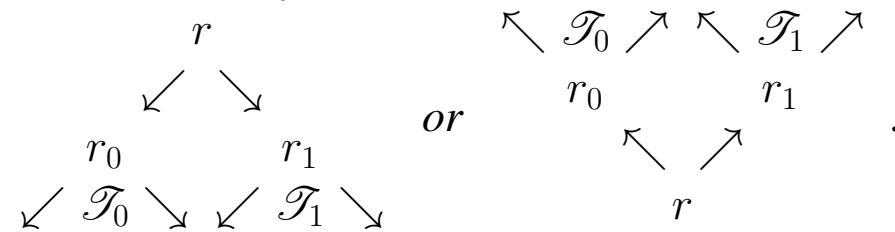
Definition 0.65 Let $\mathcal{T} = \langle N, \rightarrow, r \rangle$ be a rooted tree.

- \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 $\mathcal{T}_1 = \langle N_1, \rightarrow_1, r_1 \rangle$ with $N_0 \cap N_1 = \emptyset$, may be **joined** together at a new node r to form a new tree $\mathcal{T} = \langle N, \rightarrow, r \rangle$ 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 0.66 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 0.67 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 0.68 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 by fact 0.22 is a finite union of finite sets. Then $Desc(n)$ would be finite leading to a contradiction. QED

Lemma 0.69 (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 0.68. 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 0.70 *Every infinite tree is infinitely branching or finitely branching with at least one infinite path.*



Corollary 0.71 (Contrapositive of König's Lemma) *A finitely branching tree is finite if and only if it has no infinite path.*

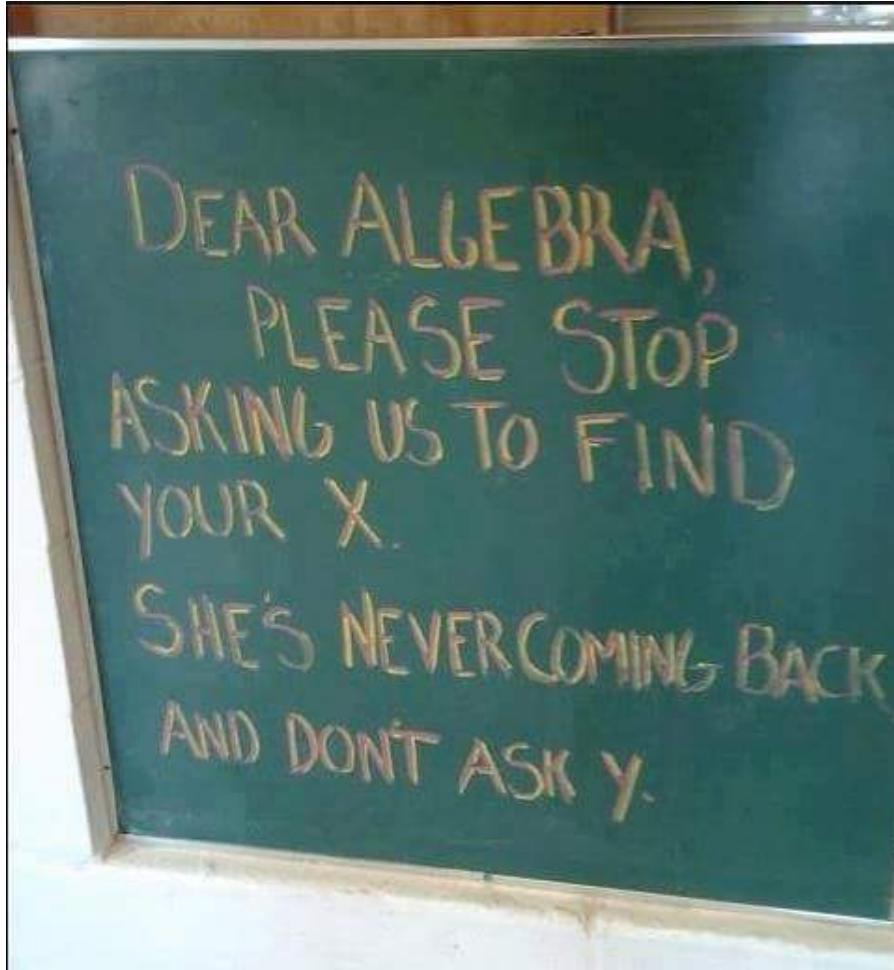


Corollary 0.72 *The root node of any finitely branching infinite tree is infinitary.*



0.4. Universal Algebra

From the internet



0.4.1. Introduction

The basic notions of universal algebra are useful in understanding the essence of data types in object-oriented languages and modern functional languages like [ML](#) and logic programming languages like [Prolog](#). This is especially so in the computations of values and functions in user-defined data-types, pattern-matching etc. Many of our examples in this section will use the data type facility in ML. We use the `typewriter` script for code written in a programming language and programming languages are colour-coded.

But before proceeding with the notions of a data-type we briefly review the notions of trees that will prove to be useful in our exposition.

0.4.2. Data types in functional and declarative languages

We begin with a few examples of simple data types in [ML](#) and [Prolog](#).

Example 0.73 Consider the following ML data type which simulates the natural numbers¹⁴ through a data type definition.

```
datatype nat = z | s of nat | a of nat * nat
```

The data type `nat` is defined recursively to generate all patterns of terms which may be identified with the natural numbers. There may be more than one pattern to represent any given natural number. In this example we have a signature Ω consisting of a 0-ary function `z` (which is meant to represent the number 0), a 1-ary constructor `s` (meant to represent the successor function on the naturals) and a 2-ary `a` (meant to represent addition on natural numbers). This data type does not actually generate the naturals. It can be used to define patterns which satisfy some of the properties of naturals. Examples of patterns which belong to the data type `nat` are the following `z`, `s(z)`, `s(a(s(z), s(s(z))))` which may be thought of as representing the numbers 0, 1 and $4 = 1 + (1 + 2)$ respectively.

The same definition in Prolog would look as follows:

```
nat(z).  
nat(s(X)) :- nat(X).  
nat(a(X, Y)) :- nat(X), nat(Y).
```

¹⁴We always include the number zero in the set of naturals

where `nat` is a property such that

- `nat` is true of `z`,
- `nat` is true of `s (X)` if `nat` is true of `X` for any `X` and
- `nat` is true of `a (X, Y)` if `nat` is true of `X` and `nat` is true of `Y`, for any `X` and `Y`.

Further in both languages it is understood that any object or entity which does not satisfy the formation rules in the data type definition does not belong to the datatype.

Example 0.74 The following data type (to be used quite independently of the data type `nat` defined in example 0.73, otherwise there will be name conflicts if they are both used in the same program) tries to simulate the notion of an integer (with 0, successor and predecessor functions) by means of patterns derived from the data type.

```
datatype integer = z | s of integer | p of integer
```

A pattern such as `p(s(s(p(s(z))))))` is meant to represent the integer value of the expression $0 + 1 - 1 + 1 + 1 - 1$ viz. the value 1. Notice that when we say “meant” to represent we are only

expressing our intent about the use of the constructors. At the level of patterns the given pattern is completely different from other patterns such as $s(z)$, $p(s(s(z)))$, $s(p(s(z)))$ etc. even though our intention may be to consider all these patterns as denoting the same number.

Exercise 0.3 Define the Prolog version of the ML data type integer.

Definition 0.75 A homogeneous or 1-sorted algebra A is an ordered pair (A, \mathcal{F}) where A is called the carrier set and \mathcal{F} is the set of operators/functions/constructors (with each $f \in \mathcal{F}$ having a predefined arity) such that A is closed under the operators of \mathcal{F} . We will often refer to A as the algebra when \mathcal{F} is understood.

The term “1-sorted” or “homogeneous” refers to the fact that there is a single carrier set A and A is closed under all the operations of \mathcal{F} i.e. there is only one kind of element viz. elements of A and each operation $f \in \mathcal{F}$ of arity $n \geq 0$ is of the kind $f : A^n \longrightarrow A$.

Example 0.76 The data type defined in example 0.73 gives us a 1-sorted algebra (A, \mathcal{F}) where the carrier set is the set of all patterns involving z , s and a which respect the arities of the constructors. $\mathcal{F} = \{z, s, a\}$. Here the notion of a sort refers to the carrier set A .

The set of patterns generated by a recursively defined data type could be infinite. Even if it is

finite but large it is usually useful to introduce the concept of a *variable* as a means of naming either arbitrary patterns of certain kinds or as devices for representing and reasoning about unknown patterns. Since we do not want to be limited by the number of names we may require, we assume the existence of an infinite set of names all distinct from any of the symbols of the data type.

Example 0.77 We might be interested in general patterns of the form $p(s(\dots))$ from the data type `integer` defined in example 0.74. We may represent them generally as terms of the form $p(s(x))$ where $x \in V$. Similarly we may want to specify a general computation rule which specifies that any pattern of the form $a(s(\dots), \dots)$ should be replaced by $s(a(\dots, \dots))$ where the \dots and the \dots themselves represent arbitrary patterns from the data type `nat`. It is then more convenient to use variables from V to represent them. We could then specify that terms of the form $a(s(x), y)$ should be replaced by terms of the form $s(a(x, y))$.

We may generalise these notions to a many-sorted algebra where there are a finite number of sorts and a collection of operations where each operation has a predefined type specifying its domain and co-domain.

Definition 0.78 A signature Ω is a pair $\langle S, \mathcal{F} \rangle$ where

- S is a finite nonempty set of **sorts** representing the various carrier sets and

- \mathcal{F} the (possibly empty) set of **function symbols** such that
- \mathcal{F} is equipped with a mapping type : $\mathcal{F} \longrightarrow (S^* \longrightarrow S)$, where each element of S^* is of the form $s_1 \times s_2 \times \cdots \times s_n$ for each $n \geq 0$.
- For each $f \in \mathcal{F}$, $\text{type}(f)$ is called the **type** of f ,
- $\text{sorts}(\Omega) = S$ and $\text{opns}(\Omega) = \mathcal{F}$.

Note that a constant $c \in s$ where $s \in S$ is a function symbol of type $\epsilon \longrightarrow s$ where $\epsilon \in S^*$ denotes the empty cartesian product of S .

Example 0.79

1. $\Omega_{\textcolor{red}{B}_0} = \langle \{\{\textcolor{red}{T}, \perp\}\}, \emptyset \rangle$ is the signature of boolean values. It is a 1-sorted algebra with no operations at all.
2. $\Omega_{\textcolor{red}{B}_1} = \langle \{\textcolor{red}{T}, \perp\}, \{\neg, ., +\} \rangle$ is the signature of a boolean algebra with the usual operations.
3. $\Omega_{\textcolor{blue}{Z}_0} = \langle \{\textcolor{blue}{Z}\}, \emptyset \rangle$ is the signature of integer values. it is a 1-sorted algebra with no operations.
4. $\Omega_{\textcolor{blue}{Z}_1} = \langle \{\textcolor{blue}{Z}\}, \{p, s\} \rangle$ is the 1-sorted algebra of integers where p and s stand for the unary predecessor and successor functions respectively. The two functions satisfy the following identi-

ties for all integers x .

$$p(s(x)) = x = s(p(x)) \quad (6)$$

5. All the above signatures are 1-sorted and are rather trivial and uninteresting. However consider $\Omega_{\mathbb{B}\mathbb{Z}_1} = \langle \{\mathbb{B}, \mathbb{Z}\}, \{p, s, \neg, ., +, <\} \rangle$ which is a 2-sorted algebra containing both booleans and integers and including all the operations as defined in $\Omega_{\mathbb{B}_1}$ and $\Omega_{\mathbb{Z}_1}$. In addition, there is another operation $< : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{B}$ which stands for the usual “less-than” relation on the integers. For example, we then have the following identities for all integers x .

$$p(x) < x = 1 \quad (7)$$

$$x < s(x) = 1 \quad (8)$$

Definition 0.80 Let Ω be a signature and let V be an infinite set of variables such that $\Omega \cap V = \emptyset$. The set $\mathcal{T}_\Omega(V)$ of all Ω -terms over V is defined inductively as the smallest set of terms such that

Basis $V \subseteq \mathcal{T}_\Omega(V)$

Induction Step If $f \in \Omega_n$ for $n \geq 0$ and $t_1, \dots, t_n \in \mathcal{T}_\Omega(V)$, then $f(t_1, \dots, t_n) \in \mathcal{T}_\Omega(V)$

The set of **ground terms** denoted \mathcal{T}_Ω is the set of all terms constructed which do not involve any variables.

The above definition is usually more concisely written as

$$s, t \in \mathcal{T}_\Omega(V) ::= x \in V | f(t_1, \dots, t_n)$$

Example 0.81 Given a set of variables V distinct from all elements in A and \mathcal{F} , $\mathcal{T}_{\mathcal{F}}(V)$ is the set of all terms which may be formed using the variables as well. Clearly $A \subset \mathcal{T}_{\mathcal{F}}(V)$. $A = \mathcal{T}_{\mathcal{F}}$ is the set of ground terms.

Exercise 0.4

1. Specify the ground terms of the datatype of example 0.74.
2. If \mathcal{F} does not contain any 0-ary operators, what is $\mathcal{T}_{\mathcal{F}}$?

0.4.3. Terms

We assume a countably infinite set of *variables/names* V (ranged over by x, y, z possibly decorated) and a set Ω called the *signature* or the set of *operators* (ranged over by $o \in \Omega$) such that $V \cap \Omega = \emptyset$. Each *operator* $o \in \Omega$ has a fixed arity $ar(o) \in \mathbb{N}$. The operators with arity 0, if any, are the *constants*. $\Sigma = V \cup \Omega$ is the set of *symbols*¹⁵.

Definition 0.82 *The set of terms is the set denoted $\mathbb{T}_\Omega(V)$ and defined inductively by the BNF*

$$s, t, u \in \mathbb{T}_\Omega(V) ::= x \in V \mid o(t_1, \dots, t_n) \quad (9)$$

where $ar(o) = n \geq 0$ and $t_1, \dots, t_n \in \mathbb{T}_\Omega(V)$. The set of **ground terms** is the set $\mathbb{T}_\Omega = \mathbb{T}_\Omega(\emptyset) \subseteq \mathbb{T}_\Omega(V)$. Equivalently, the set of ground terms is precisely the variable-free subset of $\mathcal{T}_\Omega(V)$.

\mathbb{T}_Ω is nonempty precisely when there is at least one constant in Ω . Often a constant term $c() \in \mathbb{T}_\Omega(V)$ will be simply written as c . Syntactic identity of terms will be denoted \equiv . In particular if references x and y refer to the same variable then $x \equiv y$ and if they refer to different variables then $x \not\equiv y$. Similar remarks apply for references to terms as well.

Notice that the set of terms $\mathbb{T}_\Omega(V)$ generated by the BNF (9) is closed under the operators of Ω , by

¹⁵Besides these symbols it is usual to have grouping, punctuation, association and scoping symbols which facilitate a linear reading of the term. However since every term defines a unique abstract syntax tree which render these other symbols redundant we regard them purely as *lexical decorations* and not as symbols.

definition (0.82). This leads us to the notion of a term algebra

Definition 0.83 $\langle \mathbb{T}_\Omega(V), \Omega, \{\equiv\} \rangle$ is the **term algebra** of the language defined by the BNF (9).

Example 0.84 Consider the term algebra generated by the set of integers (where each integer is a constant and the operators are the binary operators of addition, multiplication and subtraction). $(a + b)/(c * (d - 5))$ is an example of a term generated by the BNF 9 where a, b, c, d are variables. In this example $ar(+)$ = $ar(*)$ = $ar(-)$ = 2. Figure 8 depicts the abstract syntax tree of this expression.

Notice that the function *depth* defines the depth of the abstract syntax tree of a term and *size* defines the number of symbols from Σ in the term.

In many term algebras such as the λ -calculus, first-order logic, the signature could be infinite because it may contain *operator-schemas* i.e. operators that are parameterized over variables (e.g. $\lambda x.$ and $\forall x$ etc.). The BNF in such cases would be given by

$$s, t, u \in \mathbb{T}_\Omega(V) ::= x \in V \mid o(t_1, \dots, t_n) \mid Ox[t] \quad (10)$$

where O is an operator schema parameterised by the variable $x \in V$. Ox is called a *binding* of the variable x ; the brackets “[” and “]” are used to delimit the *scope* of the bound variable x . In such

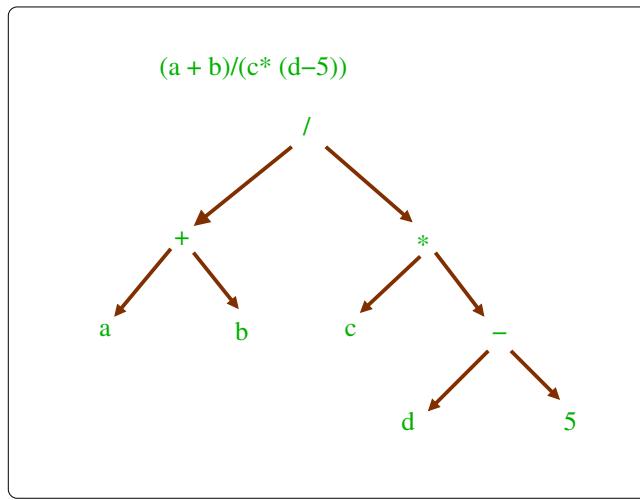


Figure 8: Abstract syntax tree of an integer arithmetic expression

cases we assume the usual notions of *free* and *bound* variables and also the usual rules of *lexical scoping* of bound variables and *binding* occurrences of variables. Each such operator schema along with its parameter (drawn from V) is regarded as a unary operator (operator schemas with multiple parameters can usually be redefined using one variable parameter at a time). Examples of such operator schemas in school mathematics include arbitrary summations and products which usually use an indexing variable.

Example 0.85 Consider the term $\sum_i a_i * b_i$ where the indexing variable i is a parameter of summa-

tion and is hence bound to it, whereas a_i and b_i are free variables. The same term would be rendered linearly in the notation of BNF (10) as $\sum i[a_i * b_i]$ where the scope of the binding of i is delimited by the brackets.

For any term t , $FV(t)$ will be used to denote the *free* variables of t and $Var(t)$ denotes the set of all variables (free and bound) that occur in t . These functions may be defined by induction on the structure of terms as follows.

Definition 0.86

s	$FV(s)$	$Var(s)$
$c()$	\emptyset	\emptyset
x	$\{x\}$	$\{x\}$
$o(t_1, \dots, t_n)$	$\bigcup_{1 \leq i \leq n} FV(t_i)$	$\bigcup_{1 \leq i \leq n} Var(t_i)$
$Ox[t]$	$FV(t) - \{x\}$	$Var(t) \cup \{x\}$

In such term algebras the usual notions of α -conversion and α -equivalence (denoted \equiv_α) may be used to provide an unambiguous but equivalent rendering of the term so that free and bound variables are kept disjoint. Going back to the example of summation we have $\sum_i [a_i * b_i] \equiv_\alpha \sum_j [a_j * b_j]$. The

presence of variable bindings also some times requires useful functions such as “nesting” depth which we shall not consider in this write-up.

What we are considering here and in the sequel are *homogeneous* term algebras, wherein it is assumed that under any interpretation of the terms of the algebra there exists a single set of elements.

A more refined form of a term algebra may be one where the bound variables may be *constrained* in some way, either through a language of constraints such as types (e.g. the ML programming language) or some predicate that defines the constraints on bound variables and operators (such as bounded quantification logic). A BNF for bindings with constraints may be defined as follows:

$$s, t, u \in \mathbb{T}_\Omega(V) ::= x \in V \mid o(t_1, \dots, t_n) \mid Ox : C[t] \quad (11)$$

where C belongs to some language of constraints. One example of such constrained terms is the term $\sum_{i=1}^n a_i * b_i$ where the bound variable i is constrained to vary between the limits 1 and n , where n is a new free variable. This term would be rendered in the BNF (11) as $\sum i : 1 \leq i \leq n [a_i * b_i]$ where $1 \leq i \leq n$ is itself a term in some suitably defined language of constraints.

Notes.

- $\equiv \subseteq \equiv_\alpha$ in algebras with operator schemas, whereas in term algebras that have no operator schemas the two notions are identical (see examples 9 and 10 below).

- We use $FV(t)$ to denote the set of free variables in term t .
- We may use *name-free* representations as done in the lambda calculus. These representations have the following advantages:
 - they separate free variables from bound variables in a term and keep the two sets disjoint,
 - they separate bound variables from other bound variables which happen to have the same name,
 - they make the use of α -conversion redundant since any two terms are α -equivalent iff they have syntactically identical name-free representations. See figure 11 for a name-free representation of the lambda term in figure 10.
- In term algebras with operator schemas, the binding mechanism Ox is to be regarded as a single position in the term with a “single symbol” Ox .

Example 0.87 (See figure 9). Let p , q and r be unary predicates. The first-order logic formula

$$\forall x[p(x) \wedge \exists x[q(x)]] \vee r(x)$$

has two distinct bound variables called x . In addition there is also a free variable x . All three of them are distinct. While the free variable is simply referred to as x the two bound variables may

be referred to as $(x, 0)$ and $(x, 1)$ respectively where the second component in each reference refers to the depth of nesting of the bindings of the two variables in the term. Alternatively, the second component may refer to the unique positions at which they are declared in the formula (see example 0.92 below).

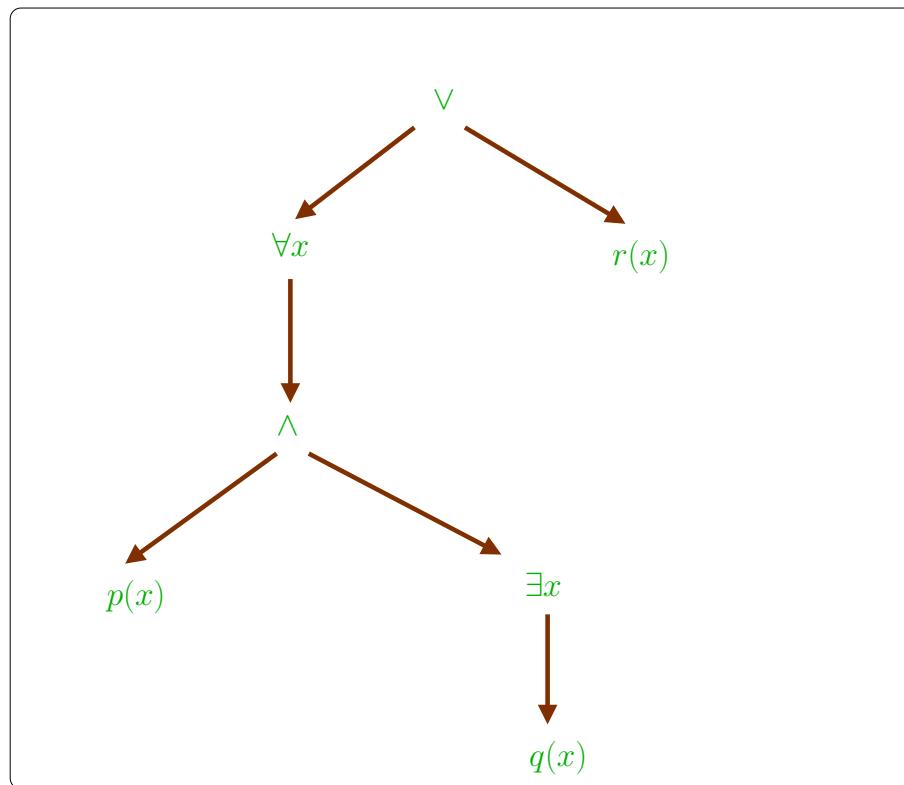


Figure 9: Abstract syntax tree of a FOL formula

Example 0.88 Figure 10 shows the abstract syntax tree of a term in the λ -calculus, where $()$ is used as the symbol for application.

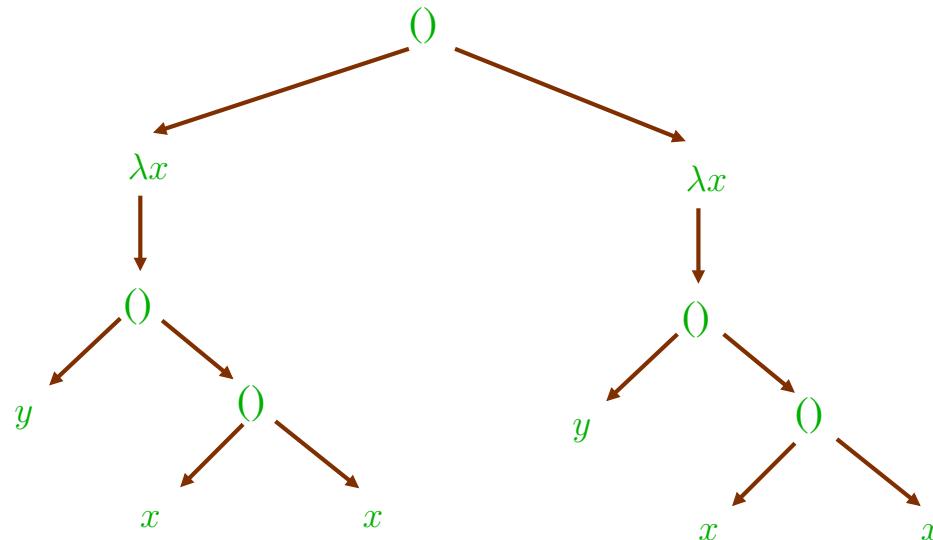


Figure 10: Abstract syntax tree of the λ -term $(\lambda x[(y (x x))] \lambda x[(y (x x))])$

Example 0.89 The name-free representation of the λ -term in example 0.88 is shown in figure 11. The red arrows indicate the variable to be filled in by referring to the position at which the variable is bound. In linear text the name-free representation would be something like $(\lambda[(y (0 0))] \lambda[(y (0 0))])$

where the integer 0 indicates that the bound variable is the one declared at the beginning of the current scope. A term such as $\lambda x[\lambda y[(y\ x)]]$ would have the representation $\lambda[\lambda[(0\ 1)]]$.

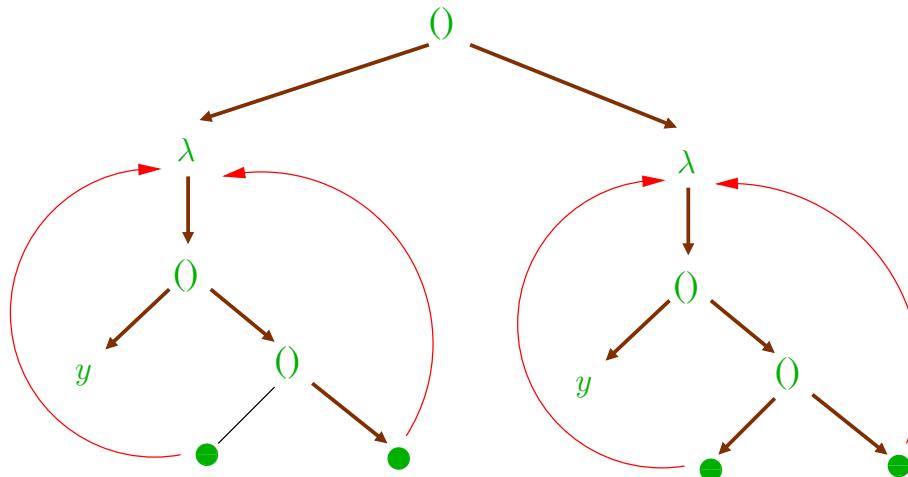


Figure 11: Abstract syntax tree of the name-free λ -term $(\lambda[(y\ (0\ 0))] \lambda[(y\ (0\ 0))])$

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. $(\lambda x[(x\ x)] \lambda x[(x\ x)])$) 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 $pos(p, t)$.

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 $ST(t) - \{t\}$.

Definition 0.90

t	$depth$	$size$	ST	Pos
$c()$	1	1	$\{t\}$	$\{\epsilon\}$
x	1	1	$\{t\}$	$\{\epsilon\}$
$o(t_1, \dots, t_n)$	$1 + Max_{i=1}^n depth(t_i)$	$1 + \sum_{i=1}^n size(t_i)$	$\{t\} \cup \bigcup_{i=1}^n ST(t_i)$	$\{\epsilon\} \cup \bigcup_{i=1}^n i.Pos(t_i)$

- The functions given in the table above are defined by induction on the structure of term t . ϵ is the empty word on strings, $.$ is the catenation operator on strings and $i.Pos(t_i) = \{i.p \mid p \in Pos(t_i)\}$.
- $s \sqsubseteq t$ iff $s \in ST(t)$ 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 $p \in Pos(t)$ 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 $p = i.p' \in Pos(t)$

- For any term t and any position $p \in Pos(t)$, $sym(p, t)$ 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 $p \in Pos(t)$, $sym(p, t) = rootsym(t|_p)$.
- The set of occurrences of a symbol $\sigma \in \Sigma \cup V$ in a term is defined as the set $Occ(\sigma, t)$ of positions in which that symbol occurs i.e. $Occ(\sigma, t) = \{p \in Pos(t) \mid sym(p, t) = \sigma\}$.

The *prefix* ordering on \mathbb{N}^* (defined as $p \preceq q$ iff there exists a string $r \in \mathbb{N}^*$ such that $p.r = q$) is a partial order on strings. The *proper prefix* ordering ($p \prec q$ iff $p \preceq q$ and $p \neq q$) is an irreflexive and transitive ordering.

Fact 0.91

1. For any term t , $size(t) = |Pos(t)|$.
2. \sqsubseteq is a partial order on $\mathcal{T}_\Omega(V)$ i.e. it is reflexive, transitive and anti-symmetric.
3. For any $p, q \in Pos(t)$,

(a) $p \preceq q$ iff $t|_q \sqsubseteq t|_p$ and

(b) $p \prec q$ iff $t|_q \sqsubset t|_p$.

Example 0.92 Let $t \equiv \lambda x[\lambda y[(x\ y)]]$ be a λ -term. Then

$$\begin{aligned} Pos(t) &= \{\epsilon, 1, 1.1, 1.1.1, 1.1.2\} \\ sym(\epsilon, t) &= \lambda x \\ sym(1, t) &= \lambda y \\ sym(1.1, t) &= () \\ sym(1.1.1, t) &= x@\epsilon \\ sym(1.1.2, t) &= y@1 \end{aligned}$$

We have used the string “ $y@1$ ” to represent the fact that the symbol at the given position is a variable named y bound at the position 1. Similarly “ $x@\epsilon$ ” denotes the variable x bound at the position ϵ .

Even though our examples will have operators of single digit arities, theoretically it is possible to have operators of arities of more than one digit. So we write “1.1”, instead of “11” using the catenation operator “.” as a separator between different elements of \mathbb{N} in strings of \mathbb{N}^* .

Example 0.93 Consider the lambda term $u \equiv (\lambda x[(y(xx))]\lambda x[(y(xx))])$. We then have

$$Pos(u) = \{\epsilon, 1, 2, 1.1, 2.1, 1.1.1, 1.1.2, 2.1.1, 2.1.2, 1.1.2.1, 1.1.2.2, 2.1.2.1, 2.1.2.2\}$$

Example 0.94 In the previous example we have

$$Occ(\textcolor{brown}{y}, u) = \{\epsilon, 1.1, 2.1, 1.1.2, 2.1.2\}$$

and

$$Occ(\textcolor{brown}{x}, u) = \{1.1.1, 2.1.1\}$$

However there are two different bound variables which happen to have the same name. Each of them is called " x " and they have binding occurrences at positions 1 and 2 respectively. We refer to them respectively as $x@1$ and $x@2$. Each of these variables has the occurrences given by

$$Occ(\textcolor{brown}{x}@1, u) = \{1.1.2.1, 1.1.2.2\}$$

and

$$Occ(\textcolor{brown}{x}@2, u) = \{2.1.2.1, 2.1.2.2\}$$

respectively.

Note. For any bound variable $x@p$ in a term t ,

1. $p \notin Occ(x @ p, t)$ and
2. for any $q \in Occ(x @ p, t)$, $p \prec q$, by the conventions governing lexical scoping

Exercise 0.5 Let $p, q, r \in \mathbb{N}^*$. Let $p(i)$ denote the i th symbol in the string p for $1 \leq i \leq |p|$ where $|p|$ denotes the length of p . i.e. if $p = k_1.k_2. \dots .k_n \in \mathbb{N}^*$, then $p(i) = k_i$ for all i , $1 \leq i \leq n$, $n \geq 0$. Now prove the following.

1. If $p \preceq q$ there exists a unique suffix $s \in \mathbb{N}^*$ such that $q = p.s$. We write $s = q \setminus p$ and we have the identity $q = p.(q \setminus p)$ for all $p \preceq q$.
2. Let $r = lcp(p, q)$ denote the longest common prefix of p and q . Then there exist unique suffixes $p' = p \setminus r$ and $q' = q \setminus r$ such that $p = r.p'$ and $q = r.q'$. If $p = \epsilon$ or $q = \epsilon$ or $p(1) \neq q(1)$ then $r = \epsilon$.
3. If $p = q$ then $p = q = r = lcp(p, q)$ and $p' = q' = \epsilon$. If $p \neq q$ then $p' \neq q'$.
4. If $p \preceq q$ then $p' = p \setminus lcp(p, q) = \epsilon$.
5. If $p' \neq \epsilon \neq q'$ then $p'(1) \neq q'(1)$ i.e. the leftmost numbers in p' and q' are different and so there exist $i, j \in \mathbb{N}$ such that $p' = i.p''$, $q' = j.q''$ and $i \neq j$.

6. We may define

$$p < q \Leftrightarrow (p \prec q) \vee [p \neq \epsilon \neq q \wedge p'(1) < q'(1)] \quad (12)$$

where $p' = p \setminus lcp(p, q)$ and $q' = q \setminus lcp(p, q)$. Then $<$ is a strict total ordering on $Pos(t)$ for any term t^{16} , i.e. for any any positions $p, q \in Pos(t)$ exactly one of the following is possible

$$p = q \quad \text{or} \quad p < q \quad \text{or} \quad q < p$$

7. Define $p \leq q$ iff $p < q$ or $p = q$. Then $p \preceq q$ implies $p \leq q$.

8. Prove that lcp is a commutative and associative operation.

9. Analogously, we may also define $lcs(p, q)$ the longest common suffix of two strings p and q . Then $lcs(p, q) = (lcp(p^R, q^R))^R$ where p^R is the string obtained by reversing the order of the letters in p .

¹⁶Our definition is consistent with the usual way in which we compare positive integers written without leading zeroes in base 10. In particular, if every element in the strings p and q is less than some integer k , we may think of p and q as $|p|$ -digit and $|q|$ -digit numbers in base k respectively.

0.5. Semantics and Meaning

We normally communicate by using a language. In the case of mathematics, we often need a (formal) language to express various arguments about a mathematical structure. For simplicity we assume that our mathematical models are algebraic systems and there is a need to express the properties of such a system using a formal language. Clearly therefore every distinguished object in the model needs to have some symbol associated with it and every distinguished operation or relation needs to have a symbol of the same arity or be associated with it. Conversely, to every symbol in a formal language we associate a meaning in the mathematical domain in which we wish to express and argue about its properties.

We know that every formal language automatically defines a term algebra. However the terms of such an algebra need to be given precise meanings. In general the term algebra would be some countably infinite set and to specify the meaning we would need to specify it inductively to be able to include all the terms. Further the meaning should be specified using a model which in turn is an algebra (not necessarily a term algebra) with a “similar” structure.

Definition 0.95 Two signatures Ω_1 and Ω_2 are said to be **similar** if there is an arity-preserving bijection between them. That is, there exists a bijection $\mu : \Omega_1 \xrightarrow[\text{onto}]{\text{I-I}} \Omega_2$ such that for every $o_1 \in \Omega_1$

with $\text{ar}(o_1) = n \geq 0$ there exists a unique $o_2 \in \Omega_2$ with $\text{ar}(o_2) = n \geq 0$ and vice-versa. We write $\mu(o_1) = o_2$ and $\mu^{-1}(o_2) = o_1$.

Definition 0.96 Algebras $A_1 = \langle \mathbb{A}_1, \Omega_1, \{=_1\} \rangle$ and $A_2 = \langle \mathbb{A}_2, \Omega_2, \{=_2\} \rangle$ are said to be **similar algebras** if their signatures are similar.

In general, the bijection μ is understood or obvious since usually the operator symbols are usually “overloaded” to save on explicitly defining the correspondence.

Note that the carrier sets \mathbb{A}_1 and \mathbb{A}_2 may not be isomorphic even if the algebras are similar. We use this fact to define the notion of a semantics or meaning of the sentences of a formal language.

A formal language is usually defined to linguistically represent or express objects of some intended model, along with the operations in the model. This needs to be done unambiguously and completely, so that

1. a sentence in the language represents or expresses a unique object in the model (unambiguously) in the context in which it is used, and
2. every sentence in the language does indeed represent an object in the model.

0.5.1. Semantics of ground terms

A formal language defines a term algebra as we have seen earlier. Let Ω be a signature with at least one constant symbol. Then the set of ground (or variable-free) terms, \mathcal{T}_Ω is non-empty. The meaning of a ground term is simply the value that the term is intended to represent. Hence we may define the meaning of a ground term by induction on the structure of the term.

Definition 0.97 *Given an algebra of terms with at least one constant symbol $\mathbf{T}_\Omega(V) = \langle \mathcal{T}_\Omega(V), \Omega, \{\equiv\} \rangle$ and another similar algebra $\mathbf{M} = \langle \mathbb{M}, \Omega', \{=\} \rangle$ with the bijection μ as in definition (0.95) between their signatures. Then a function*

$$\mathcal{V} : \mathcal{T}_\Omega \longrightarrow \mathbb{M}$$

*is called a **meaning** function if the following conditions are satisfied.*

1. *For each constant $c \in \Omega$, $\mathcal{V}[c] = \mu[c]$ and*
2. *For each operator $o \in \Omega$ with $ar(o) = n > 0$ and ground terms g_1, \dots, g_n ,*

$$\mathcal{V}[o(g_1, \dots, g_n)] = \mu(o)(\mathcal{V}[g_1], \dots, \mathcal{V}[g_n])$$

0.5.2. Semantics of Open terms

However while considering the meaning of terms which have variables, the value of the term would depend on the “context” in which the term occurs. This “context” refers to the values of the variables used in the term. This requires us to first assign values to variables from a model which the language putatively represents.

Definition 0.98 *Let V be a set of variables and let \mathbb{M} be any set. A **valuation** $v : V \rightarrow \mathbb{M}$ is a (total) function associating with each variable $x \in V$ a unique value $m \in \mathbb{M}$.*

Further such a language should be “compositional” in the sense that every operation in the term algebra needs to represent an appropriate operation on the object model. To ensure that every term in the language be assigned a suitable value in the model, it is convenient to ensure that the model is itself a similar algebra where each operation in the language represents an operation (or function of the same arity) in the model. In particular, this means that

1. every constant in the term algebra also represents an object in the carrier set of the model and
2. every operation in the model is associated with a unique operation (of exactly the same arity) in the term algebra.

It is usual in any discussion on semantics, to enclose purely syntactical elements within $\llbracket \rrbracket$ to distinguish them from the semantic elements. The reader may safely read it as simply pair of decorated brackets

Definition 0.99 Given a term algebra $T_\Omega(V) = \langle T_\Omega(V), \Omega, \{\equiv\} \rangle$ and another similar algebra $M = \langle M, \Omega', \{=\} \rangle$ with the bijection μ as in definition (0.95) between their signatures. Then a function

$$\mathcal{M} : (V \longrightarrow M) \longrightarrow T_\Omega(V) \longrightarrow M$$

is called a **meaning** or **semantic** function if for each valuation $v : V \longrightarrow M$, the following conditions are satisfied.

1. For each variable $x \in V$, $\mathcal{M}\llbracket x \rrbracket(v) = v\llbracket x \rrbracket$ and
2. For each operator $o \in \Omega$ with $ar(o) = n \geq 0$ and terms t_1, \dots, t_n ,

$$\mathcal{M}\llbracket o(t_1, \dots, t_n) \rrbracket(v) = \mu(o)(\mathcal{M}\llbracket t_1 \rrbracket(v), \dots, \mathcal{M}\llbracket t_n \rrbracket(v))$$

Given μ and v the function \mathcal{M} associates with each term t a unique value, called its **meaning**.

0.6. Substitutions

0.6.1. Substitutions and Instantiations

We formally start by defining the notion of a substitution as a function.

Definition 0.100 A **substitution** θ is a (total) function $\theta : V \rightarrow \mathcal{T}_\Omega(V)$ which is almost everywhere identity. $S_\Omega(V)$ is the set of all substitutions.

Since a substitution is almost everywhere the identity we have that except for a finite set $X \subseteq V$, for all $y \in V - X$, $\theta(y) = y$ and if $X = \{x_1, \dots, x_n\}$ for some $n \geq 0$, then for each i , $1 \leq i \leq n$, $\theta(x_i) = t_i$ for some $t_i \in \mathcal{T}_\Omega(V)$. We usually write θ as a finite set of the form

$$\theta = \{t/x \mid \theta(x) = t, t \neq x\}$$

where only the non-identical elements of the substitution are specified as pairs t/x and each such pair is read as “ t replaces x ” or more simply as “ t for x ”¹⁷. Since θ is actually a function it follows that for any i, j , $1 \leq i, j \leq n$, $i \neq j$ implies $x_i \neq x_j$. $\text{dom}(\theta) = \{x_i \mid 1 \leq i \leq n\}$ refers to the set of variables whose image under θ is not the identity¹⁸ and $\text{ran}(\theta) = \{t_i \mid 1 \leq i \leq n\}$. 1 is

¹⁷Sounds like a campaign slogan e.g. “Obama for President”. But some authors prefer the notation x/t to be read as “ x is replaced by t ”. Usually there would be no confusion except when t is itself a variable, as happens in section 0.7.1 on Pure-variable Substitutions.

¹⁸We need to clearly distinguish between the domain of a (partial) function $f : A \rightharpoonup B$ given by $\text{Dom}(f) = \{a \in A \mid f(a) \in B\} \subseteq A$ and $\text{dom}(\theta)$. Since by definition 0.104, θ is a total function we have $\text{Dom}(\theta) = V$ an infinite set whereas $\text{dom}(\theta) \subset_f \text{Dom}(\theta)$ is a finite subset of V . Similarly the range of a function given by $\text{Ran}(f) = \{b \in B \mid \exists a \in A : b = f(a)\}$ in the case of θ is

the **identity** substitution and $\text{dom}(\mathbf{1}) = \emptyset = \text{ran}(\mathbf{1})$. But θ is also a finite set, so we may use set theoretic operations on substitutions. However one needs to be careful while doing this, since the class $S_\Omega(V)$ of all substitutions over a set of variables is not closed under set theoretic operations such as union.

α conversion and equivalence. The definition of α equivalence below firstly shows that we may use substitutions to rename bound variables in a term without changing the meaning.

Definition 0.101 *For any term t ,*

$$Ox[t] \equiv_\alpha Oy[\{y/x\}t]$$

provided

1. $y \notin FV(t)$ and
2. For every subterm $u \in ST(t)$ in which y occurs bound, $x \notin FV(u)$.

Notice that y could occur bound in some sub-term of t and yet the two bound variables would not clash with each other because of the conversion.

$Ran(\theta) = \{t \in T_\Omega(V) \mid \exists x \in V : t = \theta(x)\}$ is different from $\text{ran}(\theta)$ which is a finite set.

Free variable capture. But care is required to actually prevent the so-called “capture” of free variables, because it can alter the meaning of expressions to some wholly unintended one¹⁹. The following example illustrates the meaning of this term

Example 0.102 Consider the expression $\sum_{z=1}^n y$ which may have been obtained as a result of some derivation or some calculation. Assuming all the symbols here have their usual meanings in the integers, we may safely assume that for any value of the free variables n and y , the value of this expression should be ny (i.e the product of n and y). So if we were to substitute any integer-valued function such as $y = f(x) = x^2$, we expect the result to be obtained by substituting $f(x)$ for y yielding $n.f(x) = nx^2$. However, if it so happened that f was defined by using the variable z instead of x as in $y = f(z) = z^2$, the corresponding result obtained is unfortunately not $n.f(z)$. Instead it becomes the sum of the first n squares of positive integers which is the unintended result of the substitution. This has happened because the variable z which is free in the expression $f(z)$ has got “captured” by the binding of z in $\sum_{z=1}^n y$. So whereas x^2/y is a permitted substitution, z^2/y should not be permitted because the free variable z may be captured by the binding $\sum_{z=1}^n$ in the host term.

Definition 0.103 An element t/x in a substitution is **admissible** for a host term s if no free occur-

¹⁹Loosely put, you might get wrong or wholly unexpected and unintended answers.

rence of x in s occurs within the scope of a binding of any free variable in t . A substitution θ is **admissible** for an expression s if every element of θ is admissible for s .

So definition 0.104 carefully identifies substitution with only the syntactic replacement of *free variables* in a host term with their images under the substitution such that the *free variables in the image* do not get bound by binding occurrences in an enclosing scope of the host term.

Definition 0.104 Given a term s and a substitution $\theta = \{t_i/x_i \mid t_i \not\equiv x_i, 1 \leq i \leq n\}$ admissible for s , θs denotes the new term u obtained by **instantiation** of s by θ (also called **application** of θ to s). θs is defined by induction on the structure of s as follows²⁰.

Basis

Case $s \equiv c()$. $\theta c() = c()$

Case $s \equiv x \notin \text{dom}(\theta)$. $\theta x = x$

Case $s \equiv x_i \in \text{dom}(\theta)$. $\theta x_i = t_i$

Induction

Case $s \equiv o(s_1, \dots, s_m)$. $\theta o(s_1, \dots, s_m) = o(\theta s_1, \dots, \theta s_m)$.

²⁰ We are defining this as a prefix operation, though many authors prefer to use the postfix notation “ $s\theta$ ”. There should be no confusion except when dealing with compositions of substitutions (see subsection 0.7).

Case $s \equiv Ox[s']$, $x \notin \text{dom}(\theta)$. $\theta Ox[s'] = Ox[\theta s']$

Case $s \equiv Ox[s']$, $t/x \in \theta$. $\theta Ox[s'] = Ox[(\theta - \{t/x\})s']$

Intuitively θs is a new term u obtained by replacing each *free* occurrence in s of each variable $x_i \in \text{dom}(\theta) \cap FV(t)$ by t_i . There may be several occurrences of x_i in t . Our definition of instantiation is actually *total* instantiation of a substitution. But in certain cases²¹ one could also consider *partial* applications of substitutions wherein only certain free occurrences of the variable x_i defined by its position in t may be replaced.

For any $t/x \in \theta$ Since in any substitution $\theta = \{t_i/x_i \mid t_i \not\equiv x_i, 1 \leq i \leq n\}$, $\text{depth}(t_i) \geq \text{depth}(x_i) = 1$ and $\text{size}(t_i) \geq \text{size}(x_i) = 1$ for all $1 \leq i \leq n$, the effect of applying a substitution is always depth and size non-decreasing.

Fact 0.105 *Let θ be a substitution and t a term. Then*

1. $\text{depth}(\theta t) \geq \text{depth}(t)$
2. $\text{size}(\theta t) \geq \text{size}(t)$.

²¹e.g. instantiation rules in first-order logic

Exercise 0.6 Assume t/x is not admissible in some term s , yet it is necessary to perform the substitution. How would you perform it?

Definition 0.106

- $\theta = \{t_i/x_i \mid t_i \not\equiv x_i, 1 \leq i \leq n\}$ 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 $\tau u = t$.

Exercise 0.7 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.

0.7. The Composition of 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 χ such that for every term t ,

$$\tau(\theta u) \equiv \chi u \quad (13)$$

Then χ 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 (51) 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.

- 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_i .
- 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 .
- 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 .

Case 1 requires τ to be applied separately. 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. Further all elements such that $\tau s_i \equiv x_i$ are removed from χ , 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.

Definition 0.107 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 χ such that

$$\chi = \{\tau s_i/x_i \mid 1 \leq i \leq k, \tau s_i \not\equiv x_i\} \cup \{t_j/y_j \mid 1 \leq j \leq m, y_j \notin \text{dom}(\theta)\}$$

Exercise 0.8 Prove that for any substitutions θ and τ , $\tau \circ \theta = \tau \cup \theta$ iff $\text{dom}(\theta) \cap \text{dom}(\tau) = \emptyset$ and

$$\text{dom}(\tau) \cap \bigcup_{t \in \text{ran}(\theta)} FV(t) = \emptyset$$

Lemma 0.108 *Given substitutions θ , τ , χ and a term t , we have*

1. $\theta \circ \mathbf{I} = \mathbf{I} \circ \theta = \theta$
2. $(\tau \circ \theta)t \equiv \tau(\theta t)$
3. $\chi \circ (\tau \circ \theta) = (\chi \circ \tau) \circ \theta$

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 28.4. 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

$$(\chi \circ (\tau \circ \theta))t \equiv \chi((\tau \circ \theta)t) \equiv \chi(\tau(\theta t)) \equiv (\chi \circ \tau)(\theta t) \equiv ((\chi \circ \tau) \circ \theta)t$$

QED

Exercise 0.9 A substitution θ is called **idempotent** if $\theta \circ \theta = \theta$. Now complete the statement of the following lemma and prove it.

Lemma 0.109 A substitution $\theta = \{t_i/x_i \mid 1 \leq i \leq m\}$ for some $m \geq 0$ is idempotent iff $\text{dom}(\theta) \dots$

0.7.1. Pure-variable Substitutions

Definition 0.110

- θ is a **pure-variable** substitution if $\text{ran}(\theta) \subseteq V$.

- A pure-variable substitution $\theta = \{y_1/x_1, \dots, y_n/x_n\}$ is a **renaming substitution** on a term t if
 - all the y_i are distinct and
 - $(FV(t) - \text{dom}(\theta)) \cap \text{ran}(\theta) = \emptyset$

Example 0.111 A pure-variable substitution may replace more than one distinct name by a single name e.g. $\theta = \{z/x, z/y\}$. A pure-variable substitution may also identify several names with a single name e.g. if $f(x, y)$ is a term and $\tau = \{x/y\}$ then $\{x/y\}f(x, y) \equiv f(x, x)$. However neither of these is a renaming substitution for $f(x, y)$ because in the case of θ the replacements are not all distinct and in the case of τ , the range of τ is not disjoint from $(FV(f(x, y)) - \text{dom}(\tau))$. If $FV(f(x, y)) = \{x, y\}$, then $\chi = \{u/x, v/y\}$ is a renaming substitution for $f(x, y)$. Now it is however clear that we may define an inverse $\chi^{-1} = \{x/u, y/v\}$ which is a renaming substitution for the term $\chi f(x, y)$ and in fact

Note that a renaming substitution always requires to be judged along with a term. A renaming substitution for one term may not be a renaming substitution for all terms. On the other hand, the notion of a pure-variable substitution is purely syntactic and independent of the terms to which it is applied.

Fact 0.112

1. θ is a renaming substitution for all terms.

2. The following statements are equivalent

- (a) θ is a pure-variable substitution.
- (b) $\text{size}(\theta t) = \text{size}(t)$ for all terms t .

Lemma 0.113 If t and u are variants, then there exist renaming substitutions θ and τ such that $\theta t \equiv u$ and $t \equiv \tau u$.

Proof: Since t and u are variants of each other there exist substitutions θ_1 and τ_1 such that $\theta_1 t \equiv u$ and $\tau_1 u \equiv t$. Define $\theta = \{s/x \in \theta_1 \mid x \in \text{Var}(t)\}$ and $\tau = \{t/y \in \tau_1 \mid y \in \text{Var}(u)\}$ Clearly then

$$\theta t \equiv u \text{ and } \tau u \equiv t$$

Hence $\tau(\theta t) \equiv t$. Since substitutions are always depth and size non-decreasing we have

$$\text{depth}(t) = \text{depth}(\tau(\theta t)) \geq \text{depth}(\theta t) \geq \text{depth}(t)$$

and

$$\text{size}(t) = \text{size}(\tau(\theta t)) \geq \text{size}(\theta t) \geq \text{size}(t)$$

Hence both θ and τ are pure-variable substitutions and since $\theta t \equiv u$ and $\tau u \equiv t$, both of them are renaming substitutions. QED

Definition 0.114 Let θ be any pure-variable substitution containing a maximal subset τ of the form $\tau = \{v_1/v_0, v_2/v_1, \dots, v_k/v_{k-1}\}$ for $k > 1$. Then τ is called a **substitution chain** of length k . In addition, if $v_k \equiv v_0$ then τ is a **substitution cycle** of length k . By convention, any singleton $\{x\}$ for $x \notin \text{dom}(\theta)$ is a substitution cycle of length 1.

Proposition 0.115 Let θ be any pure-variable injective substitution.

1. \mathbf{I} is an injective substitution.
2. $\text{dom}(\theta) = \text{ran}(\theta)$.
3. θ induces a partitioning of the set V of variables into substitution cycles.
4. An injective pure-variable substitution on V is a bijection on V .

Proof: Let θ be an injective substitution.

1. Trivial.
2. Follows from the following claims.

Claim. $\text{ran}(\theta) \subseteq \text{dom}(\theta)$.

\vdash We have $\theta(v) = v$ for every $v \in V - \text{dom}(\theta)$. For any $z \in \text{ran}(\theta)$ there exists $x \in$

$dom(\theta)$ such that $\theta x = z \not\equiv x$. If $z \notin dom(\theta)$ then $\theta z = z$ and $\theta x = z$ which contradicts the assumption that θ is injective. Hence $z \in dom(\theta)$ which implies $ran(\theta) \subseteq dom(\theta)$. \dashv

Claim. $dom(\theta) \subseteq ran(\theta)$.

\vdash For any $v_0 \in dom(\theta)$, $\theta v_0 = v_1 \not\equiv v_0$ for some variable $v_1 \in ran(\theta)$. Since $ran(\theta) \subseteq dom(\theta)$, $v_1 \in dom(\theta)$ and so $\theta v_1 = v_2 \not\equiv v_1$. If $v_2 \equiv v_0$ then $v_0 \in ran(\theta)$ and the proof is complete. Otherwise if $v_2 \not\equiv v_0$ then there must be some $v_3 \not\equiv v_2$ such that $\theta v_2 \equiv v_3$. Again if $v_3 \equiv v_0$ then there is nothing left to prove. Otherwise by the injectivity of θ , v_3 must be different from v_0, v_1 and v_2 and $v_3 \in dom(\theta)$. We then consider θv_3 . This process cannot go on indefinitely since both $dom(\theta)$ and $ran(\theta)$ are finite. Hence eventually, there would be some $v_k \equiv v_0$ such that $\{v_0, v_1, \dots, v_{k-1}\} \subseteq ran(\theta)$. Hence $dom(\theta) \subseteq ran(\theta)$.

\dashv

3. For every variable $x \in V - dom(\theta)$, $\{x\}$ is a substitution cycle of length 1. If $\theta \neq 1$ then for any $v_0 \in dom(\theta)$, by the proof of 2. above there exists a substitution cycle

$$\tau_1 = \{v_1/v_0, v_2/v_1, \dots, v_0/v_{k-1}\}$$

of length k for some $k > 1$. Let $\theta_1 = \theta - \tau_1$ and $V_1 = V - \{v_i \mid 0 \leq i < k\}$.

Claim. θ_1 is an injective substitution over V_1

\vdash Follows from the fact that $dom(\theta_1) = dom(\theta) - dom(\tau_1)$ \dashv

Since θ is finite we have $|\theta| > |\theta_1|$. Proceeding as above we would find a finite sequence $\tau_1, \tau_2, \dots, \tau_m$ and a decreasing sequence of injective substitutions

$$\theta = \theta_0 \supset \theta_1 \supset \theta_2 \supset \cdots \supset \theta_m$$

such that $\theta_{i+1} = \theta_i - \tau_{i+1}$ for $0 \leq i < m$, and eventually $\theta_m = \mathbf{1}$.

The required partition of V induced by θ is the set $\Pi_\theta(V)$ defined as follows

$$\Pi_\theta(V) = \{dom(\tau_i) \mid 1 \leq i \leq m\} \cup \{\{x\} \mid x \in V - dom(\theta)\}$$

QED

Theorem 0.116 A pure-variable substitution θ is injective iff $dom(\theta) = ran(\theta)$.

Proof: (\Rightarrow) has been proven in proposition 0.7.1.

(\Leftarrow) Let $\theta = \{\theta(x_i)/x_i \mid 1 \leq i \leq n\}$. Since θ is a finite set of size $n \geq 0$ with $dom(\theta) = \{x_i \mid 1 \leq i \leq n\} = \{\theta(x_i) \mid 1 \leq i \leq n\} = ran(\theta)$, it follows that the sequence $\langle \theta(x_i) \mid 1 \leq i \leq n \rangle$ must be a permutation of the sequence $\langle x_i \mid 1 \leq i \leq n \rangle$ with all the elements distinct. Hence θ is injective.

QED

Definition 0.117 An injective pure-variable substitution is called a **pure-renaming substitution**.

The subtle difference between a renaming substitution for a term and a pure-renaming substitution is captured by the following corollary.

Corollary 0.118

1. A pure-variable substitution ρ on V is a pure-renaming substitution iff $(V - \text{dom}(\rho)) \cap \text{ran}(\rho) = \emptyset$.
2. A pure-renaming substitution on V is a renaming substitution for all terms in $\mathcal{T}_\Omega(V)$.
3. If $\rho = \{y_i/x_i \mid 1 \leq i \leq n\}$ is a pure-renaming substitution then so is $\rho^{-1} = \{x_i/y_i \mid 1 \leq i \leq n\}$.

Lemma 0.119 A pure-variable substitution θ is idempotent (i.e. $\theta \circ \theta = \theta$) iff $\text{dom}(\theta) \cap \text{ran}(\theta) = \emptyset$.

Proposition 0.120 Every pure-variable substitution may be expressed as the composition of an injective substitution and an idempotent one.

Exercise 0.10 Prove proposition 0.120.

0.7.2. Syntactic Unification

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.

Example 0.121 Let f and g be distinct binary operators.

1. The terms $f(x, y)$ and $f(v, w)$ may be unified by the substitution $\theta = \{x/v, y/w\}$ since $\theta f(u, v) \equiv f(x, y) \equiv \theta f(x, y)$. They may also be unified by θ^{-1} .
2. Let r, s, t be any three terms. Then $f(x, y)$ and $f(v, w)$ (where x, y, v, w are variables) 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 $f(y, x)$ 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 $\rho f(x, y) \not\equiv \rho f(y, x)$.

4. The terms $f(x, y)$ and $f(y, x)$ can be unified by $\chi = \{x/y\}$ since $\chi f(x, y) \equiv f(x, x) \equiv \chi f(y, x)$.
5. The terms $f(x, y)$ and $g(x, y)$ cannot be unified by any substitution.

The following facts are easy to prove and may be used without any mention of them in the sequel.

Fact 0.122 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 $q \in Pos$, $p \preceq q$ implies $s|_q \equiv t|_q$.
2. If $rootsym(s|_p) = o_1 \not\equiv o_2 = rootsym(t|_p)$ then s and t are not unifiable under any substitution.
3. s and t are unifiable iff for every position $p \in Pos$, $rootsym(s|_p) \not\equiv rootsym(t|_p)$ implies at least one of the symbols is a variable i.e. $\{rootsym(s|_p), rootsym(t|_p)\} \cap V \neq \emptyset$

Exercise 0.11 Generalize the facts 29.6 to a set S of terms where $|S| \geq 2$.

There is a certain sense in which θ may be regarded as being more general than τ in example 0.121.

Definition 0.123

- 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 $\theta \succsim \tau$) if $\theta \gtrsim \tau$ and $\tau \not\gtrsim \theta$.

Fact 0.124

1. \gtrsim is a preordering relation on $S_\Omega(V)$ i.e. it is a reflexive and transitive relation.
2. \succsim is an irreflexive and transitive relation on $S_\Omega(V)$.

Definition 0.125

- Let $T = \{t_i \mid 1 \leq i \leq n\}$ be a set of terms. A substitution θ is called a **unifier** of T if $\theta t_1 \equiv \theta t_2 \equiv \dots \equiv \theta t_n$. T is **unifiable** if it has a unifier.
- θ is called a **most general unifier (mgu)** of T if for every unifier τ of T , $\theta \gtrsim \tau$.
- θ is **as general as** τ iff $\theta \sim \tau$.

Fact 0.126

1. If ρ is a pure-renaming substitution such that $\theta = \rho \circ \tau$, then $\tau = \rho^{-1} \circ \theta$.
2. \sim is an equivalence relation on $S_\Omega(V)$.
3. If a set of terms S is unifiable then it has a mgu.
4. If θ and τ are both mgu's of a set S then $\theta \sim \tau$.

Exercise 0.12 In example 0.121 identify the relationships among the different substitutions θ , θ^{-1} , τ , χ , ρ and ρ^{-1}

We now proceed to develop the algorithm for finding the most general unifier of a set S of terms. In the sequel we lift some of the standard functions on terms to sets of terms. Hence 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 $rootsym(T|_p) = \{rootsym(t|_p) \mid t \in T\}$ etc.

Definition 0.127 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 $p \prec q$, $|rootsym(T|_p)| = 1$.

We have seen that $Pos(t)$ for any term t is partially ordered by the relation \prec which is faithful to the proper sub-term ordering on $ST(t)$. 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 $u = o(t_1, t_2 \dots, t_n)$ we would like to specify *recursively* that

- the root position 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) and
- 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).

See exercise 0.5 for this total ordering

If all operators in Ω are always used in prefix form (as given in the BNF ??) then each term may also be regarded as a string in $(\Sigma \cup \{(\,),\})^*$. The ordering $<$ on $Pos(t)$ simply becomes the prefix ordering on the well-formed terms of $\mathcal{T}_\Omega(V)$.

The function Min used in algorithm 1 is the minimum position with respect to the total ordering $<$

Algorithm 1 Disagreement

Require: $|T| > 1$ {There is a position at which at least two terms have different root symbols}

```
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( $T, p, P$ )  $\stackrel{df}{=}$ 
   {If the root symbols are all the same then there must be a deeper position at which they disagree. So clearly  $P \neq \emptyset$ }
4: if  $|rootsym(T|_p)| = 1$  then
5:   let  $p' = Min(P); P' = P - \{p'\}$  in
6:   DISAGREE( $T, p', P'$ )
7:   end let
8: else {A disagreement has been found at position  $p$ }
9:   return  $T|_p$ 
10: end if
```

on positions in a term.

Example 0.128 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 1 however will compute the leftmost disagreement D_1 always.

Example 0.129 Consider the set $S_2 = \{f(g(z), x, h(g(z))), f(z, h(y), h(y)\}$. The disagreement set

$\{g(z), x\}$ is such that S_2 is not unifiable, because for any substitution θ , 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 0.130 Consider the set $S_3 = \{a, x, h(g(z)), f(b, h(y), h(y)\}$, 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 0.131 Consider the set $S_4 = \{h(z), x, h(g(z)), f(g(x), h(y), h(y)\}$. Here we have a disagreement set $D_4 = \{h(z), g(x)\}$. Since $h(z)$ cannot be unified with $g(x)$ (by merely substituting free variables by terms), S_4 is not unifiable.

Exercise 0.13 Let S' be the disagreement set of S .

1. Can $|S'|$ be different from $|S|$? Justify your answer.
2. If $S = S'$ then under what conditions is S unifiable?
3. If $S \neq S'$ then what can you say about the depths of terms in S' as compared to the depths of terms in S ?

Fact 0.132 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 the original set of terms at a particular position such that at least two distinct 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. Our algorithm 2 constructs a sequence of singleton substitutions whose composition yields a unifier if it exists.

Exercise 0.14 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.

Example 0.133 Consider the set $S = S_1$ in example 29.13. Starting with $\theta_0 = \mathbf{I}$ we go through the following steps to obtain a unifier of S .

Algorithm 2 Unification

Require: $S \subseteq_f T_\Omega(V)$ and $|S| > 1$

Ensure: If S is not unifiable then **fail** else $\exists \theta \in S_\Omega(V) : |\theta S| = 1$ and θ is a mgu of S

```
1: UNIFY( $S$ )  $\stackrel{\text{df}}{=}$  PARTIALUNIFY( $\mathbf{1}, S$ ) where
2: PARTIALUNIFY( $\theta, S$ )  $\stackrel{\text{df}}{=}$ 
3: let  $T = \theta S$  in
4:   if  $|T| = 1$  then
5:     return  $\theta$  { $\theta$  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:         PARTIALUNIFY( $\{t/x\} \circ \theta, S$ )
10:        { $T = \theta S \wedge |\{t/x\}T| < |T|$ }
11:        { $|\{t/x\} \circ \theta| < |\theta S| \leq |S|$ }
12:       else {Occurs check fails so  $S$  is not unifiable}
13:         return fail
14:       end if
15:     end let
16:   end if
17: end let
```

i	θ_i	$\theta_i S$	D_i
0	$\theta_0 = \mathbf{1}$	$\theta_0 S = \{f(a, x, h(g(z))), f(z, h(y), h(y)\}$	$D_0 = \{a, z\}$
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 0.134 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 algorithm 2 yields the following sequence of substitutions: $\theta_1 = \{g(x, x)/y\}$ from which we get $\theta_1 S = \{f(g(x, x), z, w), f(g(x, x), g(g(x, x), g(x, x)), g(z, z))\}$ and then $\theta_2 = \{g(g(x, x), g(x, x))/z\}$ which yields

$$\theta_2 S = \{f(g(x, x), g(g(x, x), g(x, x)), w),$$

$f(g(x, x), g(g(x, x), g(x, x)), g(g(x, x), g(x, x))), g(g(x, x), g(x, x)))\}$ 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 2 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 .

Theorem 0.135 (The Unification Theorem) Algorithm 2 terminates satisfying its postcondition (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) for any set of terms

satisfying its *preconditions* ($S \subseteq_f \mathcal{T}_\Omega(V)$ and $|S| > 1$).

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 7 in algorithm 2 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 $\mathbf{1} = \theta_0, \theta_1, \dots, \theta_k$ 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 $\rho = \tau_i \circ \theta_i$. For $i = 0$ clearly $\tau_0 = \rho$. Assume for some j , $0 \leq j < k$, there exists τ_j such that $\rho = \tau_j \circ \theta_j$. Clearly since θ_jS is not a singleton, a disagreement set D_j will be found for θ_jS . 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 $\tau_jx = \tau_jt$. 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 $\tau_{j+1} = \tau_j - \{\tau_jx/x\}$.

Case $x \in \text{dom}(\tau_j)$. Then $\tau_j = \{\tau_jx/x\} \cup \tau_{j+1} = \{\tau_jt/x\} \cup \tau_{j+1}$. And 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 0.15 Prove that if S is unifiable then the mgu computed by algorithm 2 is idempotent.

1. Introduction

1: Introduction

CALVIN & HOBBES by Bill Watterson



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.

I'D SAY I'VE
HAD A PRETTY
GOOD LIFE
SO FAR.



©Bill Watterson

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.

NO RETRACTABLE CLAWS,
NO OPPOSABLE TOES,
NO PREHENSILE TAIL,
NO COMPOUND EYES,
NO FANGS, NO WINGS..

..SIGHHH...



©Bill Watterson

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 November 4, 2018

Go BACK

FULL SCREEN

CLOSE

211 OF 1040

QUIT

2. Propositional Logic Syntax

2: Propositional Logic Syntax

“I suppose when you tell the tale, you deviate from the truth a lot?”

“Quite a good deal. I have always found the truth an excellent thing to deviate from.”

P. G. Wodehouse, *Sunset at Blandings*

1. Truth and Falsehood: 1
2. Truth and Falsehood: 2
3. Extending the Boolean Algebra
4. Sums & Products
5. Propositional Logic: Syntax
6. Propositional Logic: Syntax - 2
7. Natural Language equivalents
8. Some Remarks
9. Associativity and Precedence
10. Syntactic Identity
11. Abstract Syntax Trees
12. Subformulae
13. Atoms in a Formula
14. Degree of a Formula
15. Size of a Formula
16. 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 $\mathbf{2} = \langle \mathbf{2}, \{\neg, ., +\}, \{\leq, =\} \rangle$ 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.

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

Truth and Falsehood: 2

Alternatively if 0 and 1 are regarded as naturals, for any $a, b \in 2$,

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

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

Let $2' = \langle \mathbb{2}, \{\bar{,}., +, \leq^\cdot, \doteq^\cdot\}, \{\leq, =\} \rangle$ denote the extended algebra.

Boolean identities

Inverse $\bar{\bar{a}} = a$

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.

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

[Skip to Propositional Logic: Syntax](#)

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 Arkansas.

Ostriches cannot fly.

Have you finished your homework?

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

Don't 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

$$\phi, \psi ::= \perp \mid \top \mid p \in A \mid (\neg\phi) \mid (\phi \wedge \psi) \mid (\phi \vee \psi) \mid (\phi \rightarrow \psi) \mid (\phi \leftrightarrow \psi) \quad (14)$$

*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

Skip to [Some Remarks](#)

2.2. Translation of Natural Language Statements

We may translate natural language sentences (assertions and denials) 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 lanaguages 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 [8])

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.

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 [8] 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
$\phi \leftrightarrow \psi$	<p>ϕ if and only if ψ</p> <p>ϕ iff ψ</p> <p>ϕ if ψ and ψ if ϕ</p> <p>If ϕ then ψ, and conversely</p> <p>ϕ exactly if ψ</p> <p>ϕ exactly when ψ</p> <p>ϕ just in case ψ</p> <p>ϕ is a necessary and sufficient condition for ψ</p>

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 (ref. section 2.3) associated with them.
- In a formula of the form $\phi \rightarrow \psi$, ϕ is called the *antecedent* and ψ the *consequent*.
- To maintain a pleasing symmetry one could have also defined an operator \leftarrow , where $\phi \leftarrow \psi$ is intended to be read as “ ϕ if ψ ”, but that would only reverse the arrow \rightarrow and not provide any additional power of expression.

Skip to [Associativity and Precedence](#)

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.

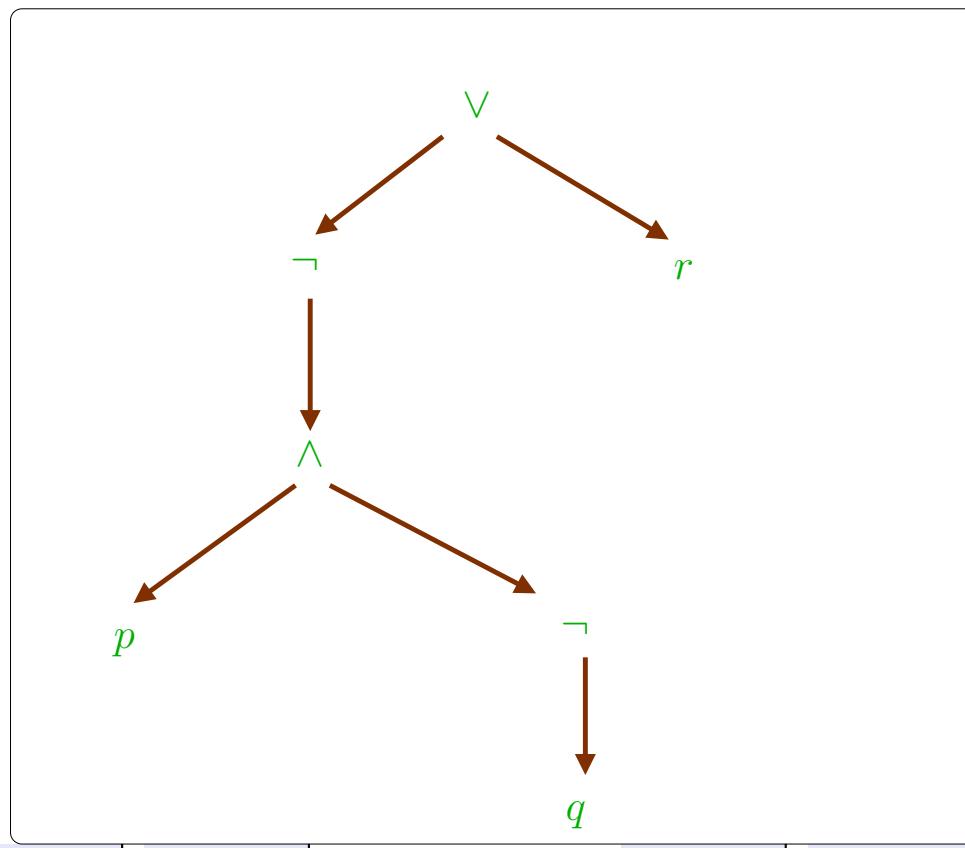
Syntactic Identity

- 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 0.63)

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:

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

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

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

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

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

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 0.53.
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.

3: Semantics of Propositional Logic

“If there's no meaning in it,” said the King, “that saves a world of trouble, you know, as we needn't try to find any.

Lewis Carroll, *Alice's Adventures in Wonderland*

1. Semantics of Propositional Logic: 1
2. Semantics of Propositional Logic: 2
3. A 1-1 Correspondence
4. Models and Satisfiability
5. Example: Abstract Syntax trees
6. 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

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

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

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

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

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

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

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

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

A 1-1 Correspondence

Notice the 1-1 correspondence between the set of operators of the language of propositional logic and those of the algebra $\mathbf{2}'$.

$$\begin{array}{rcl} \perp & \longleftrightarrow & 0 \\ \top & \longleftrightarrow & 1 \\ \neg & \longleftrightarrow & - \\ \wedge & \longleftrightarrow & . \\ \vee & \longleftrightarrow & + \\ \rightarrow & \longleftrightarrow & \leq \\ \leftrightarrow & \longleftrightarrow & \equiv \end{array}$$

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 $\tau \Vdash S$). Otherwise S is **unsatisfiable**.

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 $\tau \Vdash \phi_1 \wedge \dots \wedge \phi_n$.

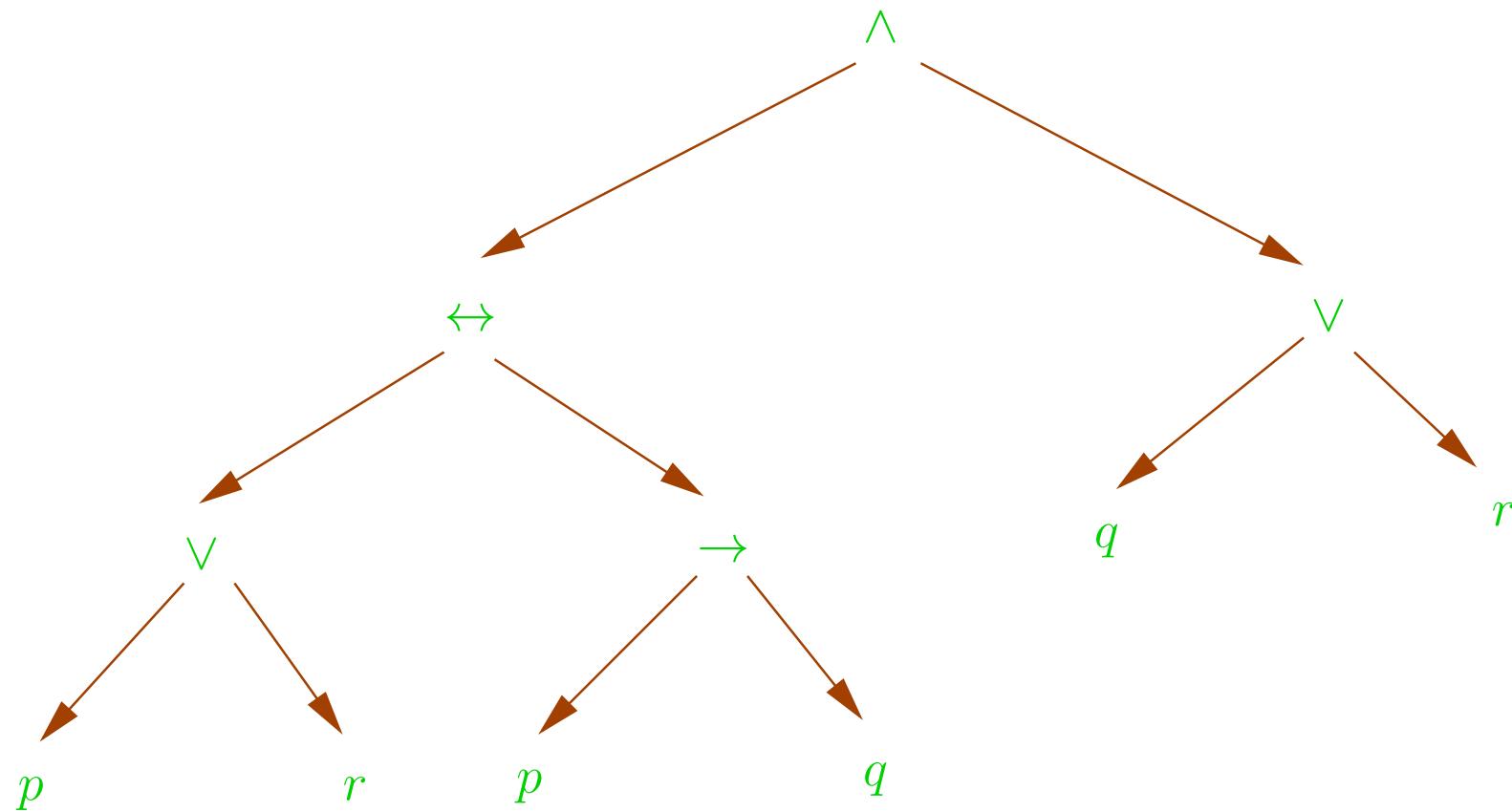
Example: Abstract Syntax trees

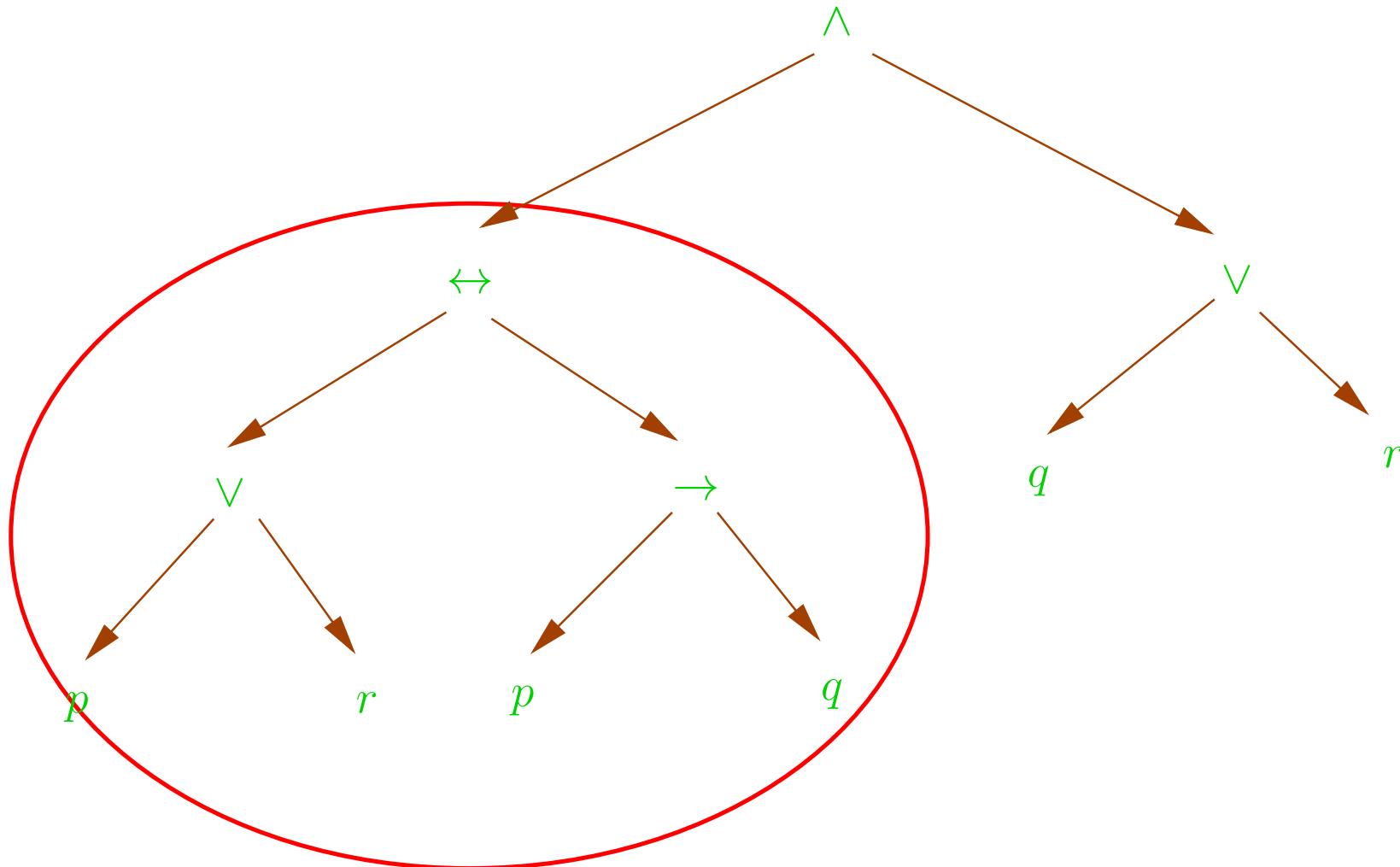
In the following example we illustrate the evaluation of the truth value of the proposition

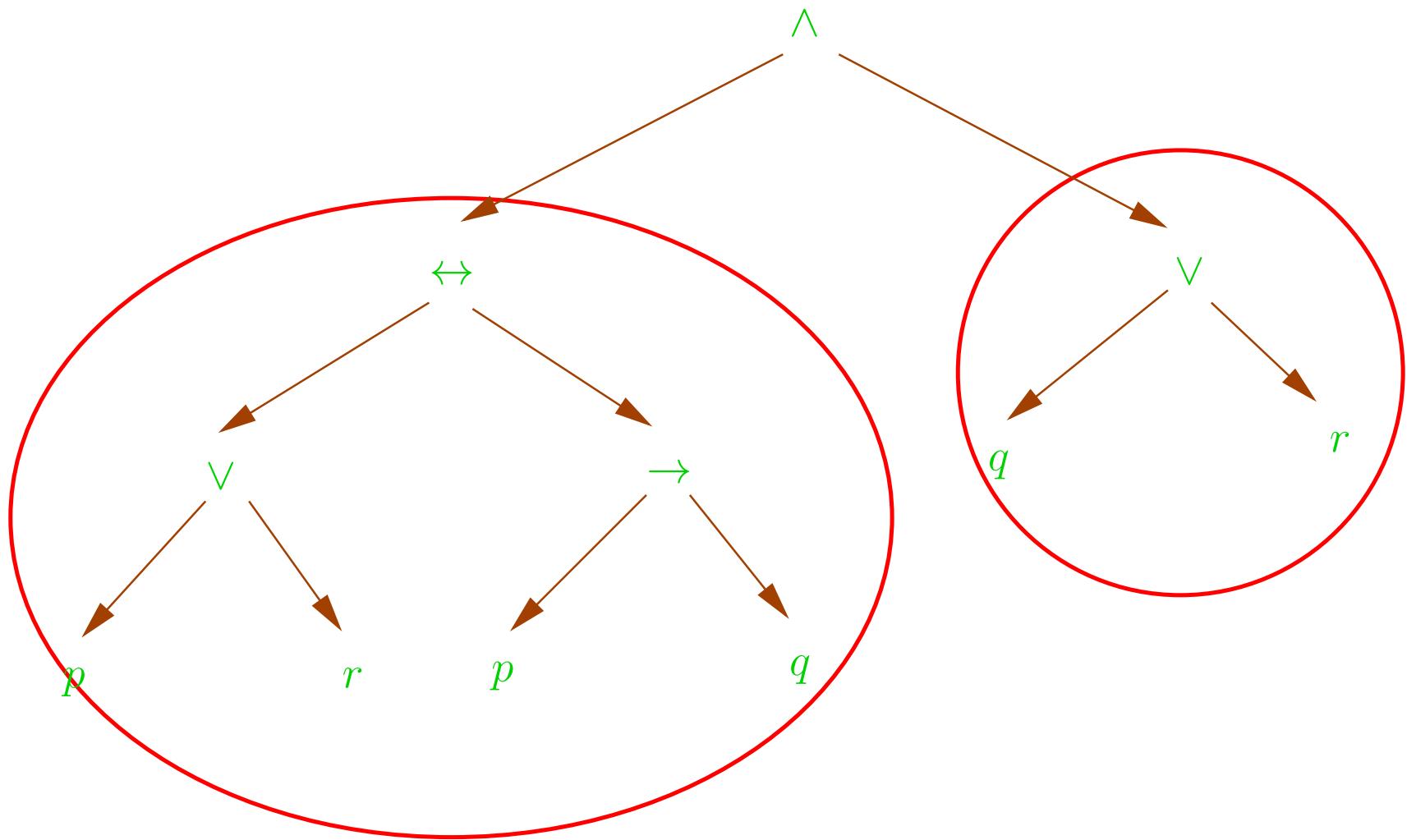
$$(((p \vee r) \leftrightarrow (p \rightarrow q)) \wedge (q \vee r))$$

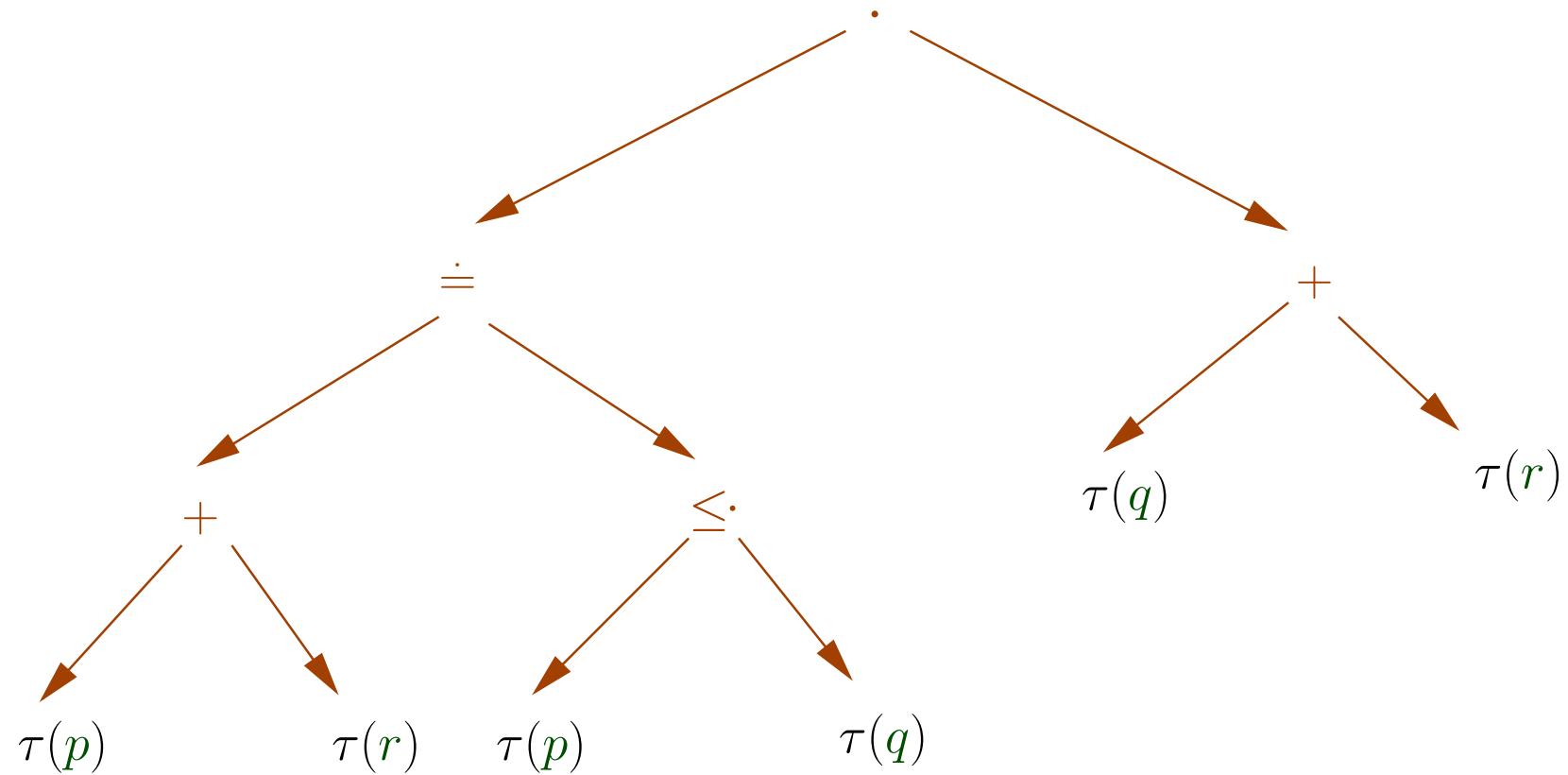
using the **semantics** to

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









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 ?

4. Logical and Algebraic Concepts

4: Logical and Algebraic Concepts

1. Logical Consequence: 1
2. Logical Consequence: 2
3. Other Theorems
4. Logical Implication
5. Implication & Equivalence
6. Propositional Logic is An Algebra
7. Logical Equivalence as a Congruence

4.1. Some Meta-Logical Concepts

We now introduce some semantical concepts that will be used in the sequel to deal with various aspects of logic as a discipline for reasoning, as a theory of reasoning and as a mathematical discipline in its own right.

The main concepts that we need are the following:

- Logical consequence
- Validity and invalidity
- Logical implication and
- Logical equivalence

Later on we will find that we also require the following meta-logical concepts.

- Satisfiability (of a set of formulae)
- Consistency (of a set of formulae)

Many of these notions can be quite confusing esp. because they have very similar or analogous concepts already present in the logical language. Further most of the literature on logic tends to identify many of them even though they are conceptually. It is therefore important that the reader carefully follow the colour coding that distinguishes the object language from the language of reasoning about the objects (viz. the formulae) in the object language.

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 (definition 4.1).

(\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. We may also say ψ is logically implied by ϕ and denote it by $\psi \Leftarrow \phi$.

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

Propositional Logic is An Algebra

Propositional Logic taken together with the concepts of logical implication and logical equivalence then becomes an algebraic structure whose ground terms are isomorphic to the boolean expressions of the **boolean algebra** $2'$.

$$P_0 = \langle \mathcal{P}_0; \{\perp, \top, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}; \{\Rightarrow, \Leftrightarrow\} \rangle$$

Fact 4.8

1. $\mathcal{P}_0 = \mathcal{T}_{\Omega_0}(A)$ i.e. is the set of terms (section 0.4.3) *inductively generated* from the signature Ω_0 .
2. The set \mathcal{T}_{Ω_0} of ground terms is the set of propositions which contain no atoms.

Logical Equivalence as a Congruence

Theorem 4.9 *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 , $1 \leq i \leq n$, $\phi_1, \dots, \phi_{i-1}, \phi_{i+1}, \dots, \phi_n, \neg\psi \models \neg\phi_i$.

5. Identities and Normal Forms

5: Identities and Normal Forms

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

Some identities (Compare with the Boolean identities)

Negation

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

Conditional

$$\phi \rightarrow \psi \Leftrightarrow \neg\phi \vee \psi$$

Biconditional

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

$$\phi \vee \perp \Leftrightarrow \phi$$

Identity

$$\phi \wedge \top \Leftrightarrow \phi$$

$$\phi \vee \top \Leftrightarrow \top$$

Zero

$$\phi \wedge \perp \Leftrightarrow \perp$$

$$\phi \vee \phi \Leftrightarrow \phi$$

Idempotence

$$\phi \wedge \phi \Leftrightarrow \phi$$

$$\phi \vee \psi \Leftrightarrow \psi \vee \phi$$

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

$$\neg\top \Leftrightarrow \perp$$

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

Absorption

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

Adequacy

Consider the two identities:

$$\phi \leftrightarrow \psi \Leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \quad (15)$$

$$\phi \rightarrow \psi \Leftrightarrow \neg\phi \vee \psi \quad (16)$$

Definition 5.1 A set of operators $O \subseteq \Omega$ is said to be **adequate** for \mathcal{P}_0 , 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 (15) and (16) 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

Corollary 5.5 Any adequate set of operators for Ω_0 is also functionally complete for \mathcal{P}_0 .

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 $\mathbf{2}$. We show in the sequel that 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 in the table 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 $0 \leq i \leq 2^n - 1$, $1 \leq j \leq n$, 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

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

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

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. It is possible to take some of the meta-logical concepts defined earlier (4.1) and convert them into operators of propositional logic. As an example let us take the concept of *logical validity* and define a new unary operator \models with the following meaning

$$\mathcal{T}[\models \phi]_\tau \stackrel{\text{df}}{=} \begin{cases} 1 & \text{if for every truth assignment } \tau', \mathcal{T}[\phi]_{\tau'} = 1 \\ 0 & \text{otherwise} \end{cases}$$

(a) By *functional completeness* this operator should be expressible (upto logical equivalence) using one of the adequate sets of operators. Express $\models \phi$ using only the operators in $\{\rightarrow, \neg\}$.

- (b) Define another unary operator \Vdash which exactly represents the concept of satisfiability of a formula i.e. $\Vdash \phi$ is true exactly when there is exists a truth assignment which makes ϕ true and false otherwise.
- (c) Express $\Vdash \phi$ using only the operators $\{\neg, \vee\}$.

Duality

Definition 5.6 *The dual of a formula ϕ , denoted ϕ^∂ , is defined by induction on the structure of formulae as follows*

$$\perp^\partial \stackrel{df}{=} \top , \quad \top^\partial \stackrel{df}{=} \perp$$

$$(\phi \wedge \psi)^\partial \stackrel{df}{=} \phi^\partial \vee \psi^\partial , \quad (\phi \vee \psi)^\partial \stackrel{df}{=} \phi^\partial \wedge \psi^\partial$$

$$(\phi \rightarrow \psi)^\partial \stackrel{df}{=} \phi^\partial \leftarrow \psi^\partial , \quad (\phi \leftarrow \psi)^\partial \stackrel{df}{=} \phi^\partial \rightarrow \psi^\partial$$

$$(\phi \leftrightarrow \psi)^\partial \stackrel{df}{=} \phi^\partial \leftrightarrow \psi^\partial$$

Dual Formulae

More informally, formulae ϕ and ψ are called **duals** of each other if each can be obtained from the other by *simultaneously* replacing all occurrences of

- \perp by \top and \top by \perp
- \wedge by \vee and \vee by \wedge ,
- \rightarrow by \leftarrow and \leftarrow by \rightarrow ,
- \leftrightarrow by itself (self-dual).

Dual Operators

We may then directly extend this notion to the individual operators themselves. That is,

- \top and \perp are **duals** of each other
- \wedge and \vee are **duals** of each other
- \rightarrow and \leftarrow are duals of each other and
- \leftrightarrow is its own dual (*self-dual*)

Definition 5.7

$$\begin{aligned}\perp^\partial &= \top, \quad \top^\partial = \perp \\ \wedge^\partial &= \vee, \quad \vee^\partial = \wedge \\ \rightarrow^\partial &= \leftarrow, \quad \leftarrow^\partial = \rightarrow \\ \leftrightarrow^\partial &= \leftrightarrow\end{aligned}$$

By extension for any compound operator o , o^∂ denotes its dual.

Principle of Duality

Theorem 5.8

1. If $\text{atoms}(\phi) = \{p_1, \dots, p_n\}$, then if $\phi \equiv o(p_1, \dots, p_n)$ then

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

2. if $\phi \Rightarrow \psi$ then $\psi^\partial \Rightarrow \phi^\partial$ and

3. if $\phi \Leftrightarrow \psi$ then $\psi^\partial \Leftrightarrow \phi^\partial$ and

Proof: By structural induction and the use of the De Morgan
and other 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 5.9

- 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 5.10 *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 5.11

- 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 5.12 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 5.2

1. 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.

2. Using principles of induction prove that

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

3. 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

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

6. Tautology Checking

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. Rewriting Conditionals, Biconditionals
12. Convert to NNF
13. Convert to CNF
14. 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.

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 theorem 4.2
2. Using one of the parts of theorem 4.3.
3. Using a truth table after converting the argument into a single propositional logic formula.

If the argument is not valid, then a **falsifying** truth 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 tedious 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
              | COND of Prop * Prop
              | BIC of Prop * Prop
```

Propositional Rendering

The Argument

```
val hyp1 = COND (rise , pandsun);  
val hyp2 = COND (taxes , busun);  
val hyp3 = COND (OR (pandsun , busun) , NOT(reelect));  
val hyp4 = COND (exceeds , inflation);  
val hyp5 = COND (exceeds , NOT(OR(taxes , OR(deffin , imf))));  
val hyp6 = COND (deffin , inflation);  
val hyp7 = COND (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

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

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 (COND (bigAND (H), P))
```

and for falsification we use

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

Next Computing CNF

Falsifying CNF

Computing the CNF

1. Rewrite conditionals and biconditionals
2. Convert to NNF by pushing negation inward
3. Convert to CNF by distributing *OR* over *AND*

Rewriting Conditionals, Biconditionals

```
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 (COND (P, Q)) = OR (NOT (rewrite (P)), rewrite (Q))
|   rewrite (BIC (P, Q)) = rewrite (AND (COND (P, Q), COND (Q, P)))
```

Convert to NNF

(* Assume all conditionals and biconditionals have been rewritten *)

```
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)))
```

Convert to CNF

(* Assume formula is in NNF *)

(* 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)
```

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

```
fun cnf (P) = C_of_D (nnf (rewrite (P)))
```

[Back to the strategy](#)

Falsifying CNF

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 can be falsified iff $P_i \cap N_i = \emptyset$.

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	Mary has a little lamb
The poor and ...	Little Bo-Peep loses her sheep
Taxes are increased	Jack and Jill go up the hill
The businessmen will be unhappy	Humpty-Dumpty sits on the wall
The Government will get re-elected	Little Miss Muffet sits on a tuffet
Inflation will rise	Little Jack Horner sits in a corner
Government expenditure ... revenue	The boy stands on the burning deck
The Government resorts ...	Wee Willie Winkie runs through the town
The Government takes a loan ... deficit	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 will *not* sit 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.

7: Binary Decision Diagrams

1. Binary Decision Diagrams
2. The if-then-else Operator 1
3. The if-then-else Operator 2
4. Boolean Expressions: Variables
5. Boolean Expressions: terms
6. if-then-else: test, high and low
7. Examples: Boolean expressions
8. More on if-then-else
9. Functional Completeness with the new operator
10. Another Normal Form: INF
11. The Shannon Expansion
12. Example: Truth table to Decision tree
13. Example:1
14. Example:2
15. Example:3
16. Example:4,5
17. BDDs
18. OBDDs
19. ROBDDs

- 20. Order of Variables
- 21. ROBDDs: Representing Boolean functions
- 22. The Canonicity Lemma
- 23. Consequences of Canonicity

Binary Decision Diagrams

- The **1-1 correspondence** between the operators of the model of truth viz. $\mathbf{2'}$ and that of Ω_0 implies that any expression in $\mathbf{2'}$ may be directly translated into a propositional formula in which the variables of the expression attain the status of propositional atoms.
- It also means that propositional reasoning could be effectively handled in the model itself provided there are efficient ways of representing the models of propositions.
- We already know that any **adequate** set of operators for Ω_0 is also **functionally complete** for \mathcal{P}_0 (corollary 5.5).
- We define a new ternary operator inspired by the if-then-else that most programming languages possess to get a more efficient data structure for propositional verification.

The if-then-else Operator 1

We begin by extending the algebra $2'$ to $2'' = \langle 2, \{\bar{,}, ., +, \leq\cdot, \doteq, ?: \}, \{\leq, =\} \rangle$ by adding a new ternary (3-ary) operator called the *if-then-else* denoted $a?b : c$ and defined by the following identity

$$a?b : c = a.b + \bar{a}.c \quad (17)$$

from which the following identities follow easily

$$1?b : c = b$$

$$0?b : c = c$$

The if-then-else Operator 2

Correspondingly we may define in propositional logic a new ternary operator ? : whose meaning is defined for each truth assignment τ as follows:

$$\mathcal{T}[\![\phi ? \psi : \chi]\!]_{\tau} \stackrel{df}{=} \mathcal{T}[\![\phi]\!]_{\tau} \textcolor{red}{?} \mathcal{T}[\![\psi]\!]_{\tau} \textcolor{brown}{:} \mathcal{T}[\![\chi]\!]_{\tau}$$

Boolean Expressions: Variables

Definition 7.1 Let Var be an infinite collection of variable symbols called boolean variables.

We may build up boolean expressions using the variables of Var (typically lower case symbols such as $a, b, c \dots$ suitably decorated by sub-scripts or super-scripts).

Boolean Expressions: terms

The set of boolean expressions may be generated from the following grammar:

Definition 7.2

$$s, t, u ::= 0 \mid 1 \mid a \in Var \mid (\bar{t}) \mid (s.t) \mid (s + t) \mid (s \leq t) \mid (s \doteq t) \mid s?t : u \quad (18)$$

Any term generated by the above grammar is called a boolean expression.

if-then-else: test, high and low

Definition 7.3 In any boolean (sub-)expression of the form $(s?t : u)$,

- s is called the **test**,
- t is the **high arm** and
- u is the **low arm**

Examples: Boolean expressions

Example 7.4 We could have various examples of boolean expressions (involving the if-then-else operator)

- $(a \doteq b)?(c?(a + d) : (b.c)) : a.d$
- $(a.b)?(c + d) : ((c.b) \leq d)$
- $(a?b : c)?(a.c) : (b?c : (c + a))$

More on if-then-else

Lemma 7.5 *The set of operators $\{0, 1, ?: :\}$ is sufficient to express (upto equality) all the expressions of $2'$.*

Proof: It is easy to prove the following identities from which the claim of the lemma follows:

$$\bar{a} = a?0 : 1$$

$$a.b = a?b : 0 = b?a : 0$$

$$a + b = a?1 : b = b?1 : a$$

$$a \leq b = a?b : 1$$

$$a \doteq b = a?b : \bar{b} = a?b : (b?0 : 1)$$

QED

Distributive properties of if-then-else

It is interesting that the if-then-else operator satisfies various distributive laws which may be used in optimizing compilers to reduce the number of evaluations of conditionals in programs.

Lemma 7.6 Let $s = a?s^1 : s^0$ and $t = a?t^1 : t^0$ where s^0 , s^1 , t^0 and t^1 are boolean expressions not involving the boolean variable a . Then

$$\bar{s} = a?\overline{s^1} : \overline{s^0} \quad (19)$$

and for any binary operator $\odot \in \{., +, \leq\cdot, \div\}$,

$$s \odot t = a?(s^1 \odot t^1) : (s^0 \odot t^0) \quad (20)$$

Proof

[skip to Functional Completeness with if-then-else](#)

Proof of Lemma 7.6

Proof: We use the definition (17) and the boolean identities to obtain the required results. The reader is encouraged to provide the justification for each step and fill in the gaps where multiple steps have been skipped. In particular, we have proved these equations in such an order that some of the steps from the later proofs use the previously proved identitites.

Equation (19). We have

$$\begin{aligned} & \overline{s} \\ &= \overline{a.s^1 + \bar{a}.s^0} \\ &= \overline{(a.s^1).(\bar{a}.s^0)} \\ &= (\bar{a} + \overline{s^1}).(\bar{\bar{a}} + \overline{s^0}) \\ &= \bar{a}.\overline{s^0} + a.\overline{s^1} + \overline{s^1}.\overline{s^0} \\ &= \bar{a}.\overline{s^0} + \bar{a}.\overline{s^0}.\overline{s^1} + a.\overline{s^1} + a.\overline{s^1}.\overline{s^0} \\ &= \bar{a}.\overline{s^0}.(1 + \overline{s^1}) + a.\overline{s^1}.(1 + \overline{s^0}) \\ &= \bar{a}.\overline{s^0} + a.\overline{s^1} \\ &= a?\overline{s^1} : \overline{s^0} \end{aligned}$$

Equation (20-+).

$$\begin{aligned} & s + t \\ &= (a.s^1 + \bar{a}.s^0) + (a.t^1 + \bar{a}.t^0) \\ &= a.(s^1 + t^1) + \bar{a}.(s^0 + t^0) \\ &= a?(s^1 + t^1) : (s^0 + t^0) \end{aligned}$$

Equation (20- \cdot).

$$\begin{aligned} & s.t \\ &= (a.s^1 + \bar{a}.s^0).(a.t^1 + \bar{a}.t^0) \\ &= a.s^1.t^1 + \bar{a}.s^0.t^0 \\ &= a?(s^1.t^1) : (s^0.t^0) \end{aligned}$$

Equation (20- $\leq\cdot$).

$$\begin{aligned} & s \leq\cdot t \\ &= \bar{s} + t \\ &= (a?\bar{s}^1 : \bar{s}^0) + (a?t^1 : t^0) \\ &= a?(s^1 \leq\cdot t^1) : (s^0 \leq\cdot t^0) \end{aligned}$$

Equation (20- \doteq).

$$\begin{aligned} & s \doteq t \\ &= (s \leq\cdot t).(t \leq\cdot s) \\ &= (a?(s^1 \leq\cdot t^1) : (s^0 \leq\cdot t^0)).(a?(t^1 \leq\cdot s^1) : (t^0 \leq\cdot s^0)) \\ &= a?((s^1 \leq\cdot t^1).(t^1 \leq\cdot s^1)) : ((s^0 \leq\cdot t^0).(t^0 \leq\cdot s^0)) \\ &= a?(s^1 \doteq t^1) : (s^0 \doteq t^0) \end{aligned}$$

QED

[skip to Functional Completeness with if-then-else](#)

Exercise 7.1 Lemma 7.6 may be generalised in the following ways. Prove each of them.

1. For boolean expressions r, s, t, u, v and any (infix) binary operator \odot ,

$$(r?s : t) \odot (r?u : v) = r?(s \odot u) : (t \odot v) \quad (21)$$

2. Let $o_n : \mathcal{D}^n \rightarrow \mathcal{D}$ be any n -ary boolean operator for any $n > 0$. If $t_i = a?t_i^1 : t_i^0$ for all i , $1 \leq i \leq n$ and boolean variable a , then

$$o_n(t_1, \dots, t_n) = a?o_n(t_1^1, \dots, t_n^1) : o_n(t_1^0, \dots, t_n^0) \quad (22)$$

3. Let $o_n : \mathcal{D}^n \rightarrow \mathcal{D}$ be any n -ary boolean operator for any $n > 0$. If $t_i = s?t_i^1 : t_i^0$ for all i , $1 \leq i \leq n$ and any boolean expression s , then

$$o_n(t_1, \dots, t_n) = s?o_n(t_1^1, \dots, t_n^1) : o_n(t_1^0, \dots, t_n^0) \quad (23)$$

4. **Composition.** Let f and g be boolean expressions and let a be any boolean variable (which may or may not occur in f or g). Prove that

$$\{g/a\}f = g?\{1/a\}f : \{0/a\}f \quad (24)$$

5. **Generalised Composition.** Let $f(a_{m-1}, \dots, a_0)$ be a boolean function of m variables and let g_{m-1}, \dots, g_0 be boolean expressions. Generalise equation 24 to complete the following equation.

$$\{g_{m-1}/a_{m-1}, \dots, g_0/a_0\}f = \dots \dots \dots \quad (25)$$

Functional Completeness with the new operator

Theorem 7.7 *The set $\{\perp, \top, ?: \}$ is adequate for Ω_0 . Hence it is also functionally complete for \mathcal{P}_0 .*

Proof: Follows from lemma 7.5 and the 1-1 correspondence between the operators of \mathcal{Z}' and Ω_0 , which may be further extended with the new if-then-else operator. QED

Another Normal Form: INF

Definition 7.8 An expression in 2^n is said to be in **if-then-else normal form (INF)** if it is built up from variables using only the operators in $\{0, 1, ?: \}$ such that every test is performed only on a boolean variable.

Notice that none of the boolean expressions of example 7.4 is in INF.

The Shannon Expansion

For convenience we use the notation $s(a_{n-1}, \dots, a_0)$ to denote that s is a boolean expression built up from n variables a_{n-1}, \dots, a_0 .

Theorem 7.9 (The Shannon Expansion) *Every boolean expression $s(a_{n-1}, \dots, a_0)$ may be expressed in INF.*

QED

Proof of Theorem 7.9

Proof:

By induction on n the number of boolean variables in the expression.

Basis. For $n = 1$, $s \equiv s(a_0)$. Regardless of how the expression has been formed s has only the following possibilities — $a_0?0 : 0$, $a_0?1 : 0$, $a_0?0 : 1$ and $a_0?1 : 1$ — all of which are in INF.

Induction Hypothesis (IH).

For each k , $1 \leq k < n$, $n > 1$, every boolean expression $s(a_{k-1}, \dots, a_0)$ may be expressed in INF $t(a_{k-1}, \dots, a_0)$ such that $s = t$.

Induction Step. Let $s \equiv s(a_{n-1}, \dots, a_0)$ be a boolean expression. Now consider the expressions $s_i \equiv \{i/a_{n-1}\}s$, $i \in \{0, 1\}$ obtained by replacing all occurrences of a_{n-1} in s by i . Clearly both $s_0 \equiv s_0(a_{n-2}, \dots, a_0)$ and $s_1 \equiv s_1(a_{n-2}, \dots, a_0)$ are boolean expressions in $n - 1$ variables. By the induction hypothesis $s_0 = t_0$ and $s_1 = t_1$ where both t_0 and t_1 are in INF. It then follows that $s = a_{n-1}?t_1 : t_0$, which is also in INF.

QED

Example: Truth table to Decision tree

Consider the following boolean expression

$$s \equiv (a \doteq b)?(c?(a + d) : (b.c)) : a.d$$

whose truth table may be constructed as shown below

a	b	c	d	s
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

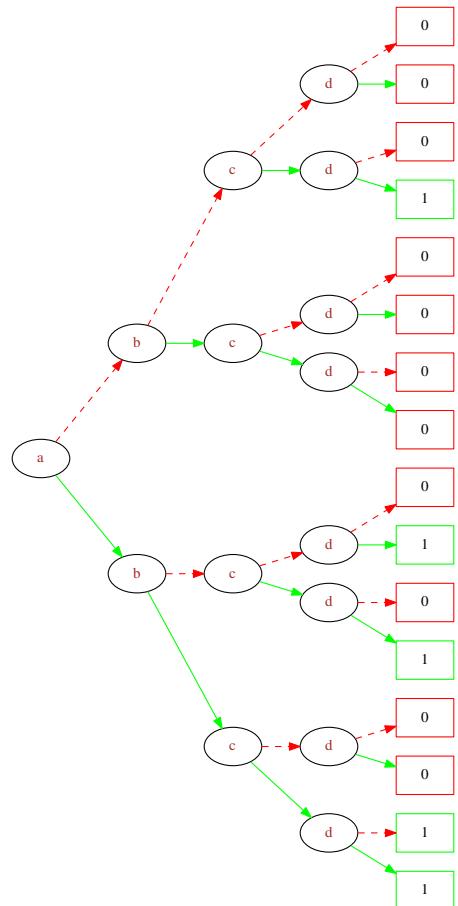
a	b	c	d	s
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

a	b	c	d	s
0	0	0	0	0
	0	0	1	0
	0	1	0	0
	0	1	1	1
	1	0	0	0
	1	0	1	0
	1	1	0	0
	1	1	1	0
1	0	0	0	0
	0	0	1	1
	0	1	0	0
	0	1	1	1
	1	0	0	0
	1	0	1	0
	1	1	0	1
	1	1	1	1

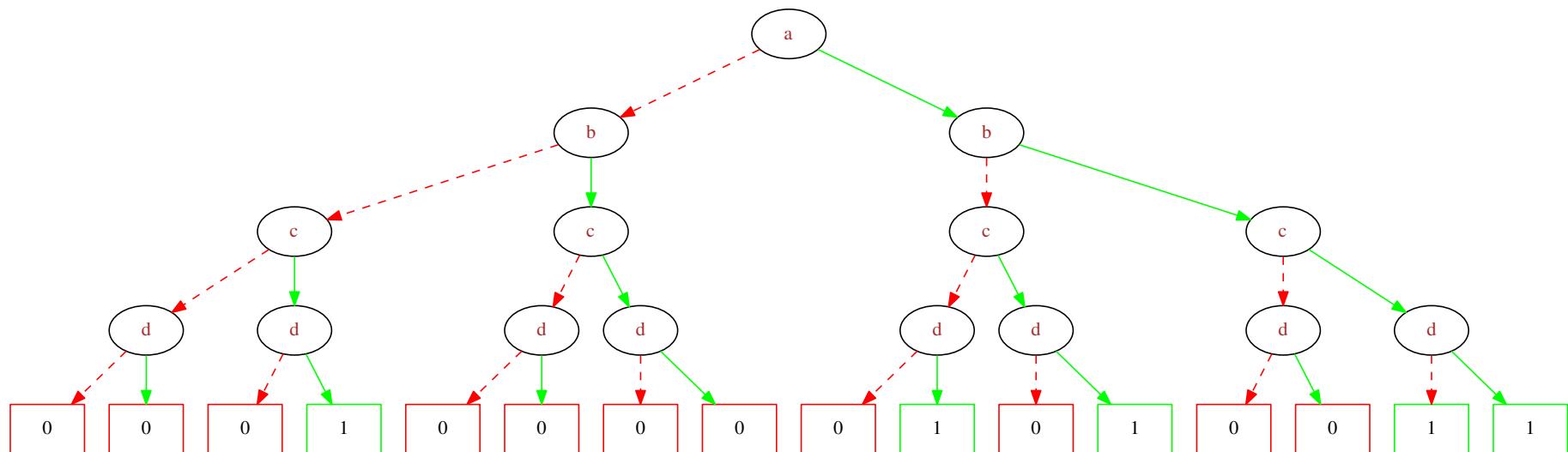
a	b	c	d	s
0	0	0	0	0
		0	1	0
		1	0	0
		1	1	1
	1	0	0	0
		0	1	0
		1	1	0
		1	1	0
1	0	0	0	0
		0	1	1
		1	0	0
		1	1	1
	1	0	0	0
		0	1	0
		1	1	1
		1	1	1

a	b	c	d	s
0	0	0	0	0
		0	1	0
		1	1	1
	1	0	0	0
		0	1	0
		1	0	0
1	0	0	0	0
		0	1	1
		0	0	0
	1	1	1	1
		0	0	0
		0	1	0
	1	0	0	1
		1	1	1

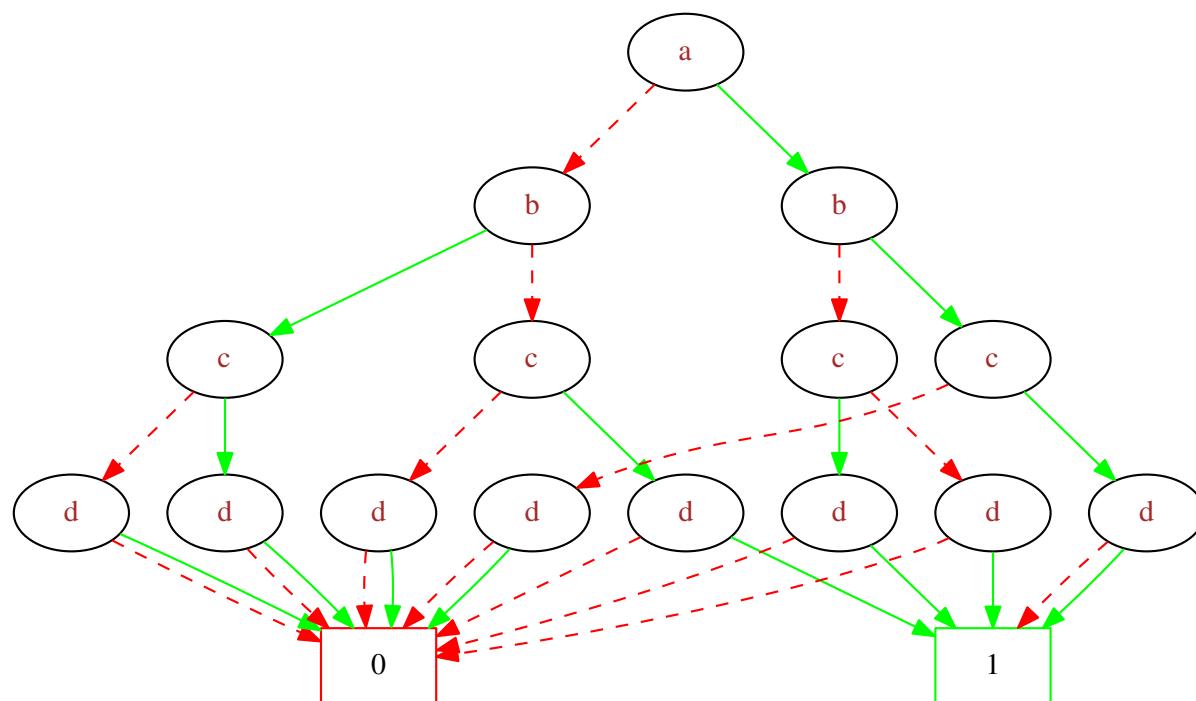
a	b	c	d	s
0	0	0	0	0
		0	1	0
		0	0	0
		1	1	1
	1	0	0	0
		0	1	0
		0	0	0
		1	1	0
1	0	0	0	0
		0	1	1
		0	0	0
		1	1	1
	1	0	0	0
		0	1	0
		0	0	1
		1	1	1



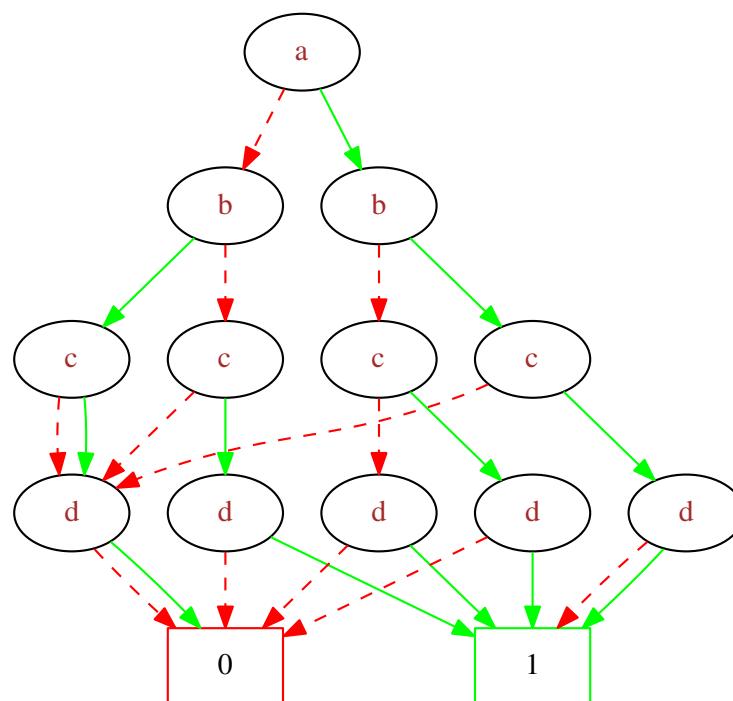
Example:1

$$(a \doteq b)?(c?(a + d) : (b.c)) : a.d$$


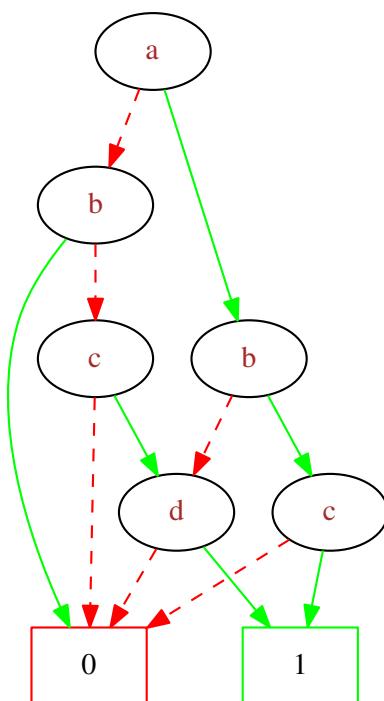
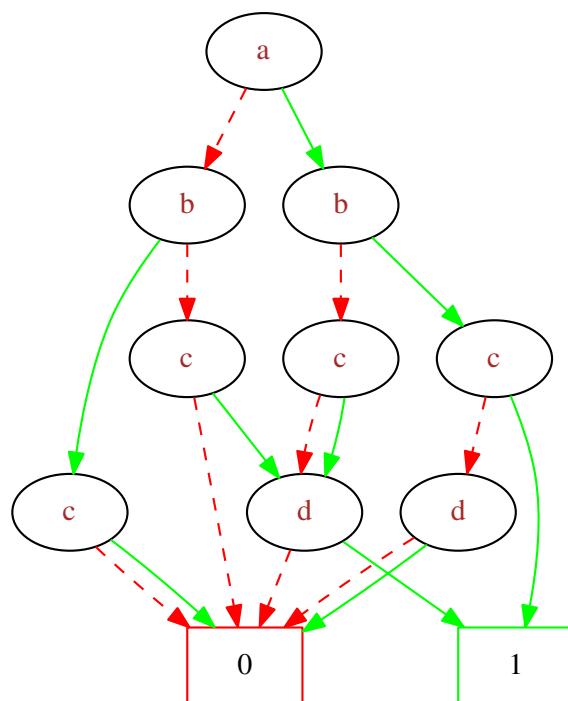
Example:2

$$(a \doteq b)?(c?(a + d) : (b.c)) : a.d$$


Example:3

$$(a \doteq b)?(c?(a + d) : (b.c)) : a.d$$


Example:4,5

$$(a \doteq b)?(c?(a + d) : (b.c)) : a.d$$


BDDs

Definition 7.10 A binary decision diagram (BDD) is a rooted directed acyclic graph (rooted DAG) with terminal nodes there is at most one terminal (out-degree zero) node labelled 0 and at most one terminal node labelled 1, non-terminal nodes a set of variable nodes labelled with a boolean variable name; each variable node has out-degree two called the low-edge and the high-edge respectively. With each non-terminal node n labelled by a variable a with low-edge going to node n_0 and high-edge to node n_1 respectively we associate the following functions:

$$var(n) = a, \ lo(n) = n_0, \ hi(n) = n_1$$

OBDDs

Definition 7.11 A *BDD* is an **ordered BDD (OBDD)** if on all paths, the variables respect a given linear order.

For example we will write our boolean expressions s as $s(a, b, c, d)$ to denote that s is built up (at most) from the variables a , b , c and d and the variables follow the linear order $a < b < c < d$.

ROBDDs

Definition 7.12 An OBDD is called a **reduced OBDD (ROBDD)** if it satisfies the following conditions for all nonterminal nodes m, n ,

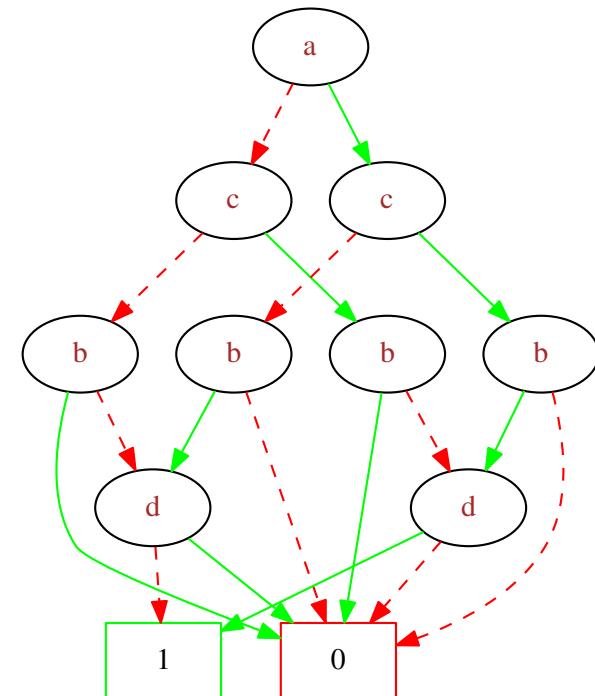
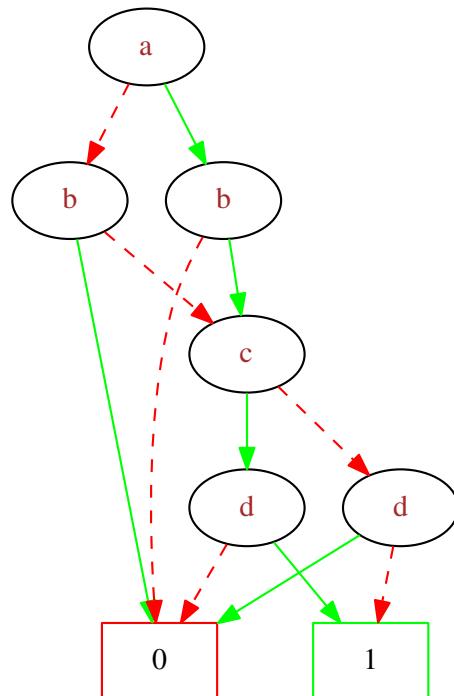
Test Irredundancy: $lo(n) \neq hi(n)$, i.e. both out-edges from a variable node cannot have the same target node.

Node non-isomorphism: $var(m) = var(n)$, $lo(m) = lo(n)$ and $hi(m) = hi(n)$ implies $m = n$, i.e. no two distinct nodes can have isomorphic sub-DAGs rooted at them.

Order of Variables

Consider the same expression with different variable orderings i.e.

$$s(a, b, c, d) \equiv (a \doteq b).(c \doteq d) \equiv t(a, c, b, d)$$



ROBDDs: Representing Boolean functions

Lemma 7.13 *For a given ordering of variables, every ROBDD represents a unique boolean function.*

Proof: We prove it by induction on the structure of nodes in a ROBDD. We may think of each node n as representing a boolean function in INF as follows expressed as a term $\textcolor{red}{t}^n$.

- $\textcolor{red}{t}^0 = 0$
- $\textcolor{red}{t}^1 = 1$
- $\textcolor{red}{t}^n = \text{var}(n) ? \textcolor{red}{t}^{\text{hi}(n)} : \textcolor{red}{t}^{\text{lo}(n)}$

QED

The Canonicity Lemma

We have seen right from the [The Shannon Expansion theorem](#) and its [proof](#) that the structure of the BDD does not depend upon the expression, but only on the [boolean function](#) that the expression denotes. Given a boolean function $f : \mathbb{2}^n \rightarrow \mathbb{2}$ of n variables we show that for a given total ordering of the variables, there is a unique ROBDD which represents that function.

Lemma 7.14 *Given any m -ary ($m \geq 0$) boolean function $f : \mathbb{2}^m \rightarrow \mathbb{2}$ and a linear ordering $a_{m-1} < \dots < a_0$ of its variables, there is a unique ROBDD representing f .*

Proof of Canonicity lemma 7.14.

Proof: By induction on the number of variables.

Basis. For $m = 0$, there exist exactly the two constants 0 with ROBDD 0 and 1 with ROBDD 1 .

Induction Hypothesis (IH).

For each k , $0 \leq k < m$, every k -ary boolean function with variable ordering $a_{k-1} < \dots < a_0$ has unique ROBDD.

Induction Step. Let $f : \mathbb{2}^m \longrightarrow \mathbb{2}$ with $m > 0$ and ordering $a_{m-1} < \dots < a_0$. The two functions $f_0(a_{m-2}, \dots, a_0) \stackrel{df}{=} f(0, a_{m-2}, \dots, a_0)$ and $f_1(a_{m-2}, \dots, a_0) \stackrel{df}{=} f(1, a_{m-2}, \dots, a_0)$ satisfy the equation

$$f(a_{m-1}, \dots, a_0) = a_{m-1}? f_1(a_{m-2}, \dots, a_0) : f_0(a_{m-2}, \dots, a_0) \quad (26)$$

By the induction hypothesis there exist *unique ROBDDs* representing f_1 and f_0 each rooted at nodes n_1 and n_0 respectively such that $t^{n_0} = f_0$ and $t^{n_1} = f_1$.

Case $n_0 = n_1$. Then $f_0 = t^{n_0} = t^{n_1} = f_1$ and hence $t = t^{n_0} = t^{n_1}$. Notice that in this case there is no occurrence of any node n such that $\text{var}(n) = a_{m-1}$. If such a node were introduced, the resulting BDD would not be reduced.

Case $n_0 \neq n_1$. Again by the induction hypothesis, $t^{n_0} = f_0 \neq f_1 = t^{n_1}$. We define a new node n with $\text{var}(n) = a_{m-1}$, $\text{lo}(n) = n_0$ and $\text{hi}(n) = n_1$ which is a reduced OBDD representing equation (26). The uniqueness of the resulting ROBDD again follows from the construction and the induction hypothesis.

QED

Consequences of Canonicity

- Checking equality of two boolean expressions with a given variable ordering is a constant time operation: because if ROBDDs are represented using a global hash table, it is just a matter of checking whether the root nodes of the two ROBDDs are the same node.
- For any propositional formula, checking whether it is a **tautology** amounts to checking whether it is identical to **1** and **contradiction** amounts to checking whether it is identical to **0**. **contingent** amounts to checking whether it has both nodes **1** and **0**.

8: 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

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 8.1 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

$$\mathcal{T}[D]_\tau = (\sum_{1 \leq i \leq m} \mathcal{T}[\lambda_{1,i}]_\tau) + (\sum_{1 \leq j \leq n, j \neq j_0} \mathcal{T}[\lambda_{2,j}]_\tau) = 1 \text{ if } p \not\equiv \lambda_{1,i_0} \text{ and}$$

$$\mathcal{T}[D]_\tau = (\sum_{1 \leq i \leq m, i \neq i_0} \mathcal{T}[\lambda_{1,i}]_\tau) + (\sum_{1 \leq j \leq n} \mathcal{T}[\lambda_{2,j}]_\tau) = 1 \text{ if } \neg p \not\equiv \lambda_{2,j_0} \text{ and hence } \tau \Vdash D. \quad \text{QED} \quad \blacksquare$$

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 8.2 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 8.1 and from problem 5 of exercise 4.1, where it should have been shown that both \wedge and \vee preserve logical implication. QED ■

Corollary 8.3 If Δ_1 and Δ_0 are as in lemma 8.2 then if Δ_1 is unsatisfiable then so is Δ_0 .

□

Theorem 8.4 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 8.2 to the result of each iteration we get that satisfiability and logical implication are preserved. QED ■

8.1. 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 $2^{2(n-1)} = O(2^{2n})$.

8.2. 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 8.5 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 8.6 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 8.7 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 8.8 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 8.9 We use *resolution* to prove

$$(p \vee q) \wedge (p \vee r) \wedge \neg p \models q \wedge r$$

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 8.10 In example 8.9 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.

9: 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

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 9.1 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

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

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

An Example Tableau Proof

Example 9.2 We provide a tableau proof of the formula

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

The black arrows denote the paths in the tableau. The blue dashed arrows provide the justification for each step of the tableau unless it is derived from its parent in the path. The last node in a closed path is marked by a \times . The undirected lines join the complementary pairs occurring in the path and provide the justification for closing the path.

$$\underline{0. \quad \neg((p \wedge q) \vee (\neg p \wedge r)) \rightarrow ((\neg p \vee q) \wedge (p \vee r))}$$

$$\underline{1. ((p \wedge q) \vee (\neg p \wedge r)) \wedge (\neg((\neg p \vee q) \wedge (p \vee r)))}$$

$$2. \underline{(p \wedge q) \vee (\neg p \wedge r))}$$

$$3. \quad \neg((\neg p \vee q) \wedge (p \vee \neg r))$$

$$\underline{4. \neg(\neg p \vee q)}$$

7. $\neg q$

1

1

V

$$\underline{p \vee q} \quad | \quad | \quad \underline{5. \neg(p \vee r)}$$

$\neg p$

p

10. $\neg p$

10

12. $\frac{q}{\checkmark}$

→ ↘

I C S November 4, 2016

Go BACK

FULL SCREEN

CLOSE

386 OF 1040

QUIT

10. Consistency & Completeness

10: 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. Soundness of the Tableau Method
10. Completeness of the Tableau Method

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 10.1 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 10.2 Every non-empty subset of a consistent set is also consistent

Lemma 10.3 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 10.3

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 10.4 A tableau is completed if no leaf in any path may be extended.

Corollary 10.5 If Γ is satisfiable then there exists a completed tableau rooted at Γ which has a satisfiable leaf.

Corollary 10.6 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 10.7 A finite or infinite set Γ is a Hintikka set if

0. $\perp \notin \Gamma$ and for any $p \in A$, $\{p, \neg p\} \not\subseteq \Gamma$,

1. If $\neg\neg\phi \in \Gamma$ then $\phi \in \Gamma$,

2. If $\psi \odot \chi \in \Gamma$ for $\odot \in \{\wedge, \neg\vee, \neg\rightarrow\}$ then $\{\psi', \chi'\} \subseteq \Gamma$,

3. If $\psi \oplus \chi \in \Gamma$ 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

$\psi \odot \chi$	ψ'	χ'	$\psi \oplus \chi$	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 10.8 *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

Exercise 10.1

1. Which of the following are Hintikka sets?
 - (a) The empty set of formulae.
 - (b) The set \mathcal{P}_0 of all propositional formulae.
 - (c) The set A of propositional atoms.
2. Prove by structural induction that for any proposition ϕ and a Hintikka set Γ , $\{\phi, \neg\phi\} \notin \Gamma$.

Tableaux and Hintikka sets

Theorem 10.9 Let $\Gamma_0, \Gamma_1 \dots, \Gamma_n$ be an open path of a completed tableau. Then $\Gamma = \bigcup_{0 \leq m \leq n} \Gamma_m$ is a Hintikka set.

Proof: We may prove that each rule in **Tableaux Rules: 2** creates a path for the construction of Hintikka sets. For any open path of the completed tableau, choose a truth assignment τ such that for each atom p , if $p \in \Gamma$ then $\tau(p) = 1$ QED

Soundness of the Tableau Method

Theorem 10.10 (Soundness of the Tableau Method). *If ϕ is a tautology then every completed tableau rooted at $\{\neg\phi\}$ is closed.*

Proof: Suppose \mathcal{T} is a completed tableau rooted at $\{\neg\phi\}$ which is open. Then by corollary 10.5 $\neg\phi$ must be satisfiable and therefore ϕ cannot be a tautology. Hence \mathcal{T} must be closed. QED

Completeness of the Tableau Method

Theorem 10.11 (Completeness of the Tableau Method). *Every tautology is provable by the tableau method.*

Proof: 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

11: The Compactness Theorem

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

Satisfiability of Infinite Sets

From corollaries 10.5 and 10.6 we have

Corollary 11.1 *A finite set Γ is unsatisfiable iff there is a closed tableau rooted at Γ .*



Corollary 11.2 *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 11.3 (The Compactness Theorem) *A (countably) infinite set is satisfiable if all its nonempty finite subsets are satisfiable.*

Proof

Corollary 11.4 *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 such that every finite subset of Γ is consistent. We need to prove that Γ is consistent.

Since Γ is countable, the formulae in Γ may be enumerated in some order, say

$$\{\phi_0, \phi_1, \phi_2, \dots\} \tag{27}$$

where each ϕ_j has the unique index $j \geq 0$. For each $m \geq 0$, let $\Gamma_m = \{\phi_0, \phi_1, \phi_2, \dots, \phi_m\}$. Further, by the hypothesis, each Γ_m is satisfiable. In particular, we may assume that for each $m \geq 0$, there exists a truth assignment τ_m such that every formula in Γ_m under τ_m is true (however the τ_m may all be different).

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 (under some truth assignment τ_k , by corollary 11.2), Δ is also satisfiable under τ_k . \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 0.69 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 theorem 10.9. By Hintikka's lemma 10.8 this set must be satisfiable. In fact the truth assignment that satisfies Γ is any truth assignment τ which ensures that $\tau(p) = 1$ if $p \in \Phi$ and $\tau(p) = 0$ if $\neg p \in \Phi$. For any atom q such that $\{q, \neg q\} \cap \Phi = \emptyset$, $\tau(q)$ could be any truth value. QED

Inconsistency

Corollary 11.5 A set Γ is *inconsistent* if some nonempty finite subset of Γ is unsatisfiable.

Proof: Follows from the compactness theorem 11.3 and its corollary 11.4. QED

Facts 11.6

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 11.7 *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.

12: Maximally Consistent Sets

1. Consistent Sets
2. Maximally Consistent Sets
3. Properties of Finite Character: 1
4. Properties of Finite Character: Examples:1
5. Properties of Finite Character: Examples:2
6. Properties of Finite Character: Compactness
7. Properties of Finite Character: Tukey's Lemma
8. Lindenbaum's Theorem

Consistent Sets

Lemma 12.1 *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 12.1

Proof: Suppose Γ is consistent but both Γ_0 and Γ_1 are inconsistent. Then by compactness and by definition 11.5 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 11.6.1 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

Maximally Consistent Sets

Definition 12.2 A set Δ is a **maximally consistent set** if it is satisfiable and no proper superset of Δ is consistent.

Corollary 12.3 For any maximally consistent set Δ , and any formula ϕ , either $\phi \in \Delta$ or $\neg\phi \in \Delta$ but not both.

Our proof of the compactness theorem (theorem 11.3) relied on the use of the **Tableau Method**. However since the notions of satisfiability, consistency and maximal consistency are semantical notions, it is intuitively clear that the property of compactness should not depend on any particular proof system. Hence it must be possible to prove the compactness theorem without invoking any proof method. So we shall try to prove the compactness theorem for Propositional Logic only from semantical notions. Of course if we have not yet proved the compactness theorem, we cannot also assume that consistency/satisfiability is a property of finite character. However we may assume lemma 12.1 which refers to how consistent sets may be extended, definition 12.2 which defines maximally consistent sets and corollary 12.3 which denies the existence of both a formula and its negation in a maximally consistent set.

Finite Satisfiability

Definition 12.4 A set Γ is called **finitely satisfiable** if every non-empty finite subset of Γ is satisfiable.

If Γ is satisfiable, then clearly it is also finitely satisfiable. So we concentrate on only one part of the compactness theorem.

Maximally Consistent Sets and Compactness

Theorem 12.5 (The Compactness Theorem (\Leftarrow)). Γ is satisfiable if it is finitely satisfiable.

Proof: (\Leftarrow) Assume Γ is finitely satisfiable. The theorem follows from the following two claims.

Claim 1. There exists a maximal set $\Delta \supseteq \Gamma$ such that Δ is finitely satisfiable.

Claim 2. There exists a single truth assignment τ such that every formula in Δ is true under τ .

QED

Claim. There exists a maximal set $\Delta \supseteq \Gamma$ such that Δ is finitely satisfiable.

⊤ Since \mathcal{P}_0 is countably infinite there exists an enumeration

$$\phi_1, \phi_2, \dots$$

Starting from $\Delta_0 = \Gamma$ we now construct the sets Δ_{n+1} by induction as follows:

$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{\phi_{n+1}\} & \text{if } \Delta_n \cup \{\phi_{n+1}\} \text{ is finitely satisfiable} \\ \Delta_n \cup \{\neg\phi_{n+1}\} & \text{otherwise} \end{cases}$$

Let $\Delta = \bigcup_{n \geq 0} \Delta_n$. It is clear that $\Gamma \subseteq \Delta$ and for every ϕ either ϕ or $\neg\phi$ is in Δ . Hence Δ is maximal. Further, by lemma 12.1 each Δ_n is finitely satisfiable. Therefore Δ is finitely satisfiable because every finite subset of Δ is contained in some Δ_n for some $n \geq 0$ and each Δ_n is finitely satisfiable. Hence Δ is a maximal finitely satisfiable superset of Γ . \dashv

Claim. There exists a single truth assignment τ such that every formula in Δ is true under τ .

Since Δ is maximal, for every atom p , either p or $\neg p$ belongs to Δ . Define the truth assignment τ such that $\tau(p) = 1$ if $p \in \Delta$ and $\tau(p) = 0$ otherwise. We may now prove by induction on the structure of ϕ that

$$\mathcal{T}[\![\phi]\!]_{\tau} = 1 \text{ iff } \phi \in \Delta$$

This structural induction proof is quite easy and left to the reader.

⊣

Properties of Finite Character: 1

Definition 12.6 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: Examples:1

Example 12.7 *The property of a partially ordered set being totally ordered (definition 0.10) 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.

Properties of Finite Character: Examples:2

Example 12.8 However the property of a totally ordered set $\langle T, \leq \rangle$ being well-ordered (definition 0.19) 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).

Properties of Finite Character: Compactness

By the corollary 11.4 to the compactness theorem, consistency/satisfiability is a property of finite character.

Properties of Finite Character: Tukey's Lemma

Theorem 12.9 (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 (theorem 12.9)

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

$$a_1, a_2, a_3, \dots \quad (28)$$

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 (28). 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

Lindenbaum's Theorem

Theorem 12.10 (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 12.6 and corollary 11.4 consistency of sets of formulae is a property of finite character in the universe \mathcal{P}_0 . From theorem 12.9 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

$$\phi_1, \phi_2, \phi_3, \dots \quad (29)$$

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 (29). Since $\Gamma_\infty \cup \{\phi_{i+1}\}$ is consistent, by fact 10.2 $\Gamma_i \cup \{\phi_{i+1}\} \subseteq \Gamma_\infty \cup \{\phi_{i+1}\}$ 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 12.1

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 .

13: 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

Pure mathematics is the subject in which we do not know what we are talking about and whether what we are saying is true.

Bertrand Russell

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 13.1 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.

The Axioms and Inference rules together constitute a proof system for the theory.

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 13.2

- *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 syntactic substitution (see Section 0.6).
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 13.3 \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.

14: Proof Theory: Hilbert-style

1. More About Formal Theories
2. The Law of The Excluded Middle
3. An Example Proof:1
4. An Example Proof:2
5. An Example Proof:3
6. An Example Proof:4
7. An Example Proof:5
8. Formal Proofs: 1
9. Formal Proofs: 2
10. Provability and Formal Proofs
11. The Deduction Theorem
12. About Formal Proofs

More About Formal Theories

We use the symbol “ $\vdash_{\mathcal{H}_0}$ ” to denote provability using the proof system \mathcal{H}_0 . $\Gamma \vdash_{\mathcal{H}_0} \psi$ denotes that ψ is formally provable from the set Γ in the proof system \mathcal{H}_0 . The subscript is omitted whenever the proof system is clear from the context.

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

Theorem 14.1 *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 $\Gamma \vdash \psi$.*



The Law of The Excluded Middle

We formally deduce $\phi \rightarrow \phi$ for any formula ϕ using the proof system \mathcal{H}_0 . This formula is also called “the law of the excluded middle” and is an important one which distinguishes “classical logic” (which is what this course is all about) and what are known as “intuitionistic logics”. The formula is **logically equivalent** to the formula $\neg\phi \vee \phi$. It is also a reflexivity property of the conditional.

An Example Proof:1

As is normal practice in mathematics the proof is presented as a sequence of steps.

$$1. \phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi) \quad \{\phi/X, \phi \rightarrow \phi/Y\}K$$

where each step is justified as an instance of an axiom schema or a rule.

An Example Proof:2

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. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
 $\{\phi/X, \phi \rightarrow \phi/Y, \phi/Z\}S$

where each step is justified as an instance of an axiom schema or a rule.

An Example Proof:3

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. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$ $\{2, 1\}MP$

where each step is justified as an instance of an axiom schema or a rule.

An Example Proof:4

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. $(\phi \rightarrow (\phi \rightarrow \phi))$ $\{\phi/X, \phi/Y\}K$

where each step is justified as an instance of an axiom schema or a rule.

An Example Proof:5

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 0.63) 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:

$$\frac{1 \frac{\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)}{3 \frac{2 \frac{(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))}{5 \frac{((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))}{\phi \rightarrow \phi}}}{4 \frac{(\phi \rightarrow (\phi \rightarrow \phi))}{(\phi \rightarrow (\phi \rightarrow \phi))}}$$

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: 1

Definition 14.2 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.

Formal Proofs: 2

Definition 14.3

- 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 14.4 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 14.5 (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 14.5)

Proof: (\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 14.1) $\Gamma, \phi \vdash \phi \rightarrow \psi$ 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 .

$$\frac{m+2 \frac{m}{\phi \rightarrow \psi} \mathcal{T}}{\psi} \quad \frac{m+1 \overline{\phi}}{\psi}$$

\mathcal{T}' is a proof of $\Gamma, \phi \vdash \psi$.

(\Rightarrow). Assume $\Gamma, \phi \vdash \psi$. Then there exists a proof tree \mathcal{T} rooted at ψ with nodes $\psi_1, \dots, \psi_m \equiv \psi$ such that every leaf node has a smaller index than a non-leaf node and every non-leaf node has a larger index than any of its children. It is clear that ψ_1 will be a leaf node and ψ_m (having the highest index) is the root of the proof tree.

Then the following stronger claim proves the required result.

Claim. $\Gamma \vdash \phi \rightarrow \psi_i$ for all i , $1 \leq i \leq m$.

⊤ By induction on the structure of the proof tree \mathcal{T}

Basis. ψ_i is a leaf node. 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 **the law of the Excluded Middle** and monotonicity (theorem 14.1).

Case $\psi_i \in \Gamma$ or ψ_i 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}'_i 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 $1 \leq i < l$ for some index $l \leq m$.

Induction Step. If ψ_l is a leaf node then the proof is just as in the basis. Hence assume it is not a leaf node. 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 children say ψ_i and ψ_j with $i \neq j$ such that $i, j < l$. 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}'_l rooted at $\phi \rightarrow \psi_l$ from \mathcal{T}'_i and \mathcal{T}'_j as follows.

$$\frac{j' + 2 \frac{j' \frac{\nwarrow \mathcal{T}'_j \nearrow}{\phi \rightarrow (\psi_i \rightarrow \psi_l)} \quad j' + 1 \frac{(\phi \rightarrow (\psi_i \rightarrow \psi_l)) \rightarrow ((\phi \rightarrow \psi_i) \rightarrow (\phi \rightarrow \psi_l))}{(\phi \rightarrow \psi_i) \rightarrow (\phi \rightarrow \psi_l)}}{j' + i' + 3} \quad j' + i' + 2 \frac{\nwarrow \mathcal{T}'_i \nearrow}{\phi \rightarrow \psi_i}}{\phi \rightarrow \psi_l}$$

where $j' + 1$ is an instance of **S**, and $j' + 2$ and $j' + i' + 3$ are both applications of **MP** to their respective children in the tree.

⊣

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?

15: 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

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 15.1 A sequent is a meta-formula of the form $\Gamma \vdash \phi$.

Derived Rules

- By **substitutivity** (theorem 14.1) 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 ”.
- **Reflexivity** may be expressed in sequent form as a derived rule.
- The Deduction Theorem and its converse may be rendered in sequent form as derived rules.
- 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 15.2 For all ϕ , ψ and χ ,

$$\vdash (\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi))$$

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 15.2 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 15.1

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 15.2. Assume Γ_1 , Γ_2 and Γ_3 are as in the proof of theorem 15.2.

$$\frac{\text{DT} \Rightarrow \frac{\Gamma_3 \vdash \psi \rightarrow \chi}{\Gamma_2 \vdash \phi \rightarrow (\psi \rightarrow \chi)}}{\text{MP} \frac{\text{S}}{\frac{\Gamma_2 \vdash (\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \chi))}{\Gamma_2 \vdash (\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \chi)}} \quad \Gamma_2 \vdash \phi \rightarrow \psi}$$
$$\frac{\text{DT} \Rightarrow \frac{\Gamma_2 \vdash \phi \rightarrow \chi}{\Gamma_1 \vdash (\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi)}}{\text{DT} \Rightarrow \frac{}{\vdash (\phi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \rightarrow \chi))}}$$

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

$$\text{DNE} \rightarrow . \frac{}{\Gamma \vdash \neg\neg X \rightarrow X}$$

$$\text{DNI} \rightarrow . \frac{}{\Gamma \vdash X \rightarrow \neg\neg X}$$

Proof of derived rule DNE and axiom schema DNE \rightarrow

Proof:

$$\begin{array}{c} \text{K } \frac{}{\neg\neg\phi \rightarrow (\neg\phi \rightarrow \neg\neg\phi)} \\ \text{T} \Rightarrow \frac{\text{R2} \Rightarrow \frac{\text{N } \frac{(\neg\phi \rightarrow \neg\neg\phi) \rightarrow ((\neg\phi \rightarrow \neg\phi) \rightarrow \phi)}{(\neg\phi \rightarrow \neg\neg\phi) \rightarrow \phi}}{\text{R} \Rightarrow \frac{}{\neg\phi \rightarrow \neg\phi}}}{\text{DT} \Leftarrow \frac{\neg\neg\phi \rightarrow \phi}{\neg\neg\phi \vdash \phi}} \end{array}$$

QED

Proof of derived rule DNI and axiom schema DNI \rightarrow

Proof:

$$\begin{array}{c} \text{K } \frac{}{\phi \rightarrow (\neg\neg\neg\phi \rightarrow \phi)} \\ \text{T} \Rightarrow \frac{\text{MP } \frac{\text{N } \frac{(\neg\neg\neg\phi \rightarrow \neg\phi) \rightarrow ((\neg\neg\neg\phi \rightarrow \phi) \rightarrow \neg\neg\phi)}{(\neg\neg\neg\phi \rightarrow \phi) \rightarrow \neg\neg\phi}}{\text{DNE } \frac{}{\neg\neg\neg\phi \rightarrow \neg\phi}}}{\text{DT} \Leftarrow \frac{\phi \rightarrow \neg\neg\phi}{\phi \vdash \neg\neg\phi}} \end{array}$$

QED

Exercise 15.2

1. Give a formal proof of Peirce's law $((\phi \rightarrow \psi) \rightarrow \phi) \rightarrow \phi$.

2. 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.

3. 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 .

4. Prove the following axiom schemas in \mathcal{H}_0 . In each case you are allowed to use any version of the theorems previously proven.

$$\boxed{(a) \perp. \quad \frac{}{\neg X \rightarrow (X \rightarrow Y)}}$$

What can you conclude about the system \mathcal{H}_0 from your proof?

(b)

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

(c)

$$\text{N2. } \frac{}{X \rightarrow (\neg Y \rightarrow \neg(X \rightarrow Y))}$$

(d)

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

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 .

$$\text{I. } \frac{\begin{array}{c} \Gamma, X \vdash \neg Y \\ \Gamma, X \vdash Y \end{array}}{\Gamma \vdash \neg X}$$

5. Prove the derived axiom

$$\text{C2. } \frac{}{\Gamma \vdash (\neg X \rightarrow X) \rightarrow X}$$

Derived Operators

$$\top \stackrel{df}{=} X \rightarrow X$$

$$\perp \stackrel{df}{=} \neg(X \rightarrow X)$$

$$X \vee Y \stackrel{df}{=} \neg X \rightarrow Y$$

$$X \wedge Y \stackrel{df}{=} \neg(X \rightarrow \neg Y)$$

$$X \leftrightarrow Y \stackrel{df}{=} \neg((X \rightarrow Y) \rightarrow \neg(Y \rightarrow X))$$

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 15.3

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 .

16: 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

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 16.1 A *formal theory is unsound if every wff is a theorem. Otherwise it is said to be sound.*

Points to ponder:

- If Γ is an infinite set then any proof of $\Gamma \vdash \psi$ would be a **finite tree** requiring only a finite subset $\Gamma_f \subseteq_f \Gamma$ of assumptions ($\Gamma_f \vdash_{\mathcal{H}_0} \psi$)
- By **corollary 11.7** every logical consequence is deducible from a finite subset of assumptions.
- Our proof system \mathcal{H}_0 (or even \mathcal{G}_0) is sound if it allows us to derive only conclusions which *preserve truth* under all truth assignments i.e. $\Gamma \vdash_{\mathcal{H}_0} \psi$ must imply that ψ is a **logical consequence** of the assumptions Γ .
- By theorem **4.2** every valid argument in propositional logic can be represented by a tautology.

Soundness of the Hilbert System

Lemma 16.2

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

QED

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 16.2

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 16.3 *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 16.4 *The system \mathcal{H}_0 is sound.*

17: The Hilbert System: Completeness

1. Towards Completeness
2. Towards Truth-tables
3. The Truth-table Lemma
4. The Completeness Theorem

Towards Completeness

1. By theorem 16.3 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 15.3.4, \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 17.1 (The Truth-table Lemma) 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 17.1

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 $n \geq 0$ occurrences 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.

$$\frac{\text{DNI} \rightarrow \frac{\Gamma \vdash \psi \rightarrow \neg\neg\psi}{\Gamma \vdash \psi}}{\text{MP} \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \neg\neg\psi \equiv \phi^*}}$$

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 $\frac{\Gamma \vdash \psi^*}{\nwarrow \mathcal{T}_1 \nearrow}$ and $\frac{\Gamma \vdash \chi^*}{\nwarrow \mathcal{T}_2 \nearrow}$. Here again we have three subscases.

Subcase $\mathcal{T}[\psi]_\tau = 0$. We have $\psi^* \equiv \neg\psi$ so tree \mathcal{T}_1 is $\frac{\Gamma \vdash \neg\psi}{\nwarrow \mathcal{T}_1 \nearrow}$, $\mathcal{T}[\phi]_\tau = 1$ and $\phi^* \equiv \phi \equiv \psi \rightarrow \chi$. We then have the proof tree

$$\text{MP} \frac{\frac{\perp \quad \frac{\Gamma \vdash \neg\psi \rightarrow (\psi \rightarrow \chi)}{\nwarrow \mathcal{T}_1 \nearrow}}{\Gamma \vdash \psi \rightarrow \chi \equiv \phi^*}}{\Gamma \vdash \neg\psi}$$

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 $\frac{\Gamma \vdash \chi}{\nwarrow \mathcal{T}_2 \nearrow}$. We may then construct the following proof tree to prove the claim.

$$\text{MP} \frac{\mathsf{K} \frac{}{\Gamma \vdash \chi \rightarrow (\psi \rightarrow \chi)} \quad \frac{\nearrow \mathcal{T}_2 \searrow}{\Gamma \vdash \chi}}{\Gamma \vdash \psi \rightarrow \chi \equiv \phi^*}$$

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 $\frac{\nearrow \mathcal{T}_1 \searrow}{\Gamma \vdash \psi}$ and $\frac{\nearrow \mathcal{T}_2 \searrow}{\Gamma \vdash \neg\chi}$ using which we construct the following proof tree to prove our claim.

$$\text{N2} \frac{}{\Gamma \vdash \psi \rightarrow (\neg\chi \rightarrow \neg(\psi \rightarrow \chi))} \quad \text{MP} \frac{\frac{\nearrow \mathcal{T}_1 \searrow}{\Gamma \vdash \psi} \quad \frac{\nearrow \mathcal{T}_2 \searrow}{\Gamma \vdash \neg\chi}}{\Gamma \vdash \neg\chi \rightarrow \neg(\psi \rightarrow \chi)} \quad \text{MP} \frac{}{\Gamma \vdash \neg(\psi \rightarrow \chi) \equiv \phi^*}$$

QED

The Completeness Theorem

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

Theorem 17.2 (The Completeness Theorem). *Every tautology is a formal theorem of \mathcal{H}_0 .*



Proof of the Hilbert Completeness Theorem 17.2

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 17.1), there exists a distinct proof tree, $\frac{\Gamma_{s_k} \vdash \phi}{\mathcal{T}_{s_k}}$ for each such s_k . For each bit-string

$s_j \in \{0, 1\}^j$, $0 < j \leq k$, we construct the proof tree $\frac{\Gamma_{s_{j-1}} \vdash \phi}{\mathcal{T}_{s_{j-1}}}$ from the proof trees $\frac{\Gamma_{s_{j-1}0} \vdash \phi}{\mathcal{T}_{s_{j-1}0}}$

and $\frac{\Gamma_{s_{j-1}1} \vdash \phi}{\mathcal{T}_{s_{j-1}1}}$, 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 $\frac{\Gamma_{s_0} \vdash \phi}{\mathcal{T}_{s_0}}$ from

the 2^k distinct proof trees $\frac{\Gamma_{s_k} \vdash \phi}{\mathcal{T}_{s_k}}$.

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.

$$\frac{\text{DT} \Rightarrow \frac{\text{MP} \frac{\Gamma_{s_{j-1}0} \vdash \phi}{\Gamma_{s_{j-1}} \vdash \neg p_j \rightarrow \phi}}{\text{DT} \Rightarrow \frac{\text{MP} \frac{\Gamma_{s_{j-1}1} \vdash \phi}{\Gamma_{s_{j-1}} \vdash p_j \rightarrow \phi}}{\text{C} \frac{\Gamma_{s_{j-1}} \vdash (p_j \rightarrow \phi) \rightarrow ((\neg p_j \rightarrow \phi) \rightarrow \phi)}{\Gamma_{s_{j-1}} \vdash (\neg p_j \rightarrow \phi) \rightarrow \phi}}}}{\Gamma_{s_{j-1}} \vdash \phi}$$

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

18: Introduction to Predicate Logic

1. Predicate Logic: Introduction-1
2. Predicate Logic: Introduction-2
3. Internal Structure of Sentences
4. Internal Structure of Sentences
5. Parameterisation-1
6. Parameterisation-2
7. Predicate Logic: Introduction-3
8. Predicate Logic: Symbols
9. Predicate Logic: Signatures
10. Predicate Logic: Syntax of Terms
11. Predicate Logic: Syntax of Formulae
12. Precedence Conventions
13. Predicates: Abstract Syntax Trees
14. Subterms
15. Variables in a Term
16. Bound Variables And Scope
17. Bound Variables And Scope: Example
18. Scope Trees
19. Free Variables

20. Bound Variables

21. Closure

Predicate Logic: Introduction-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

Example 18.1

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”.

Internal Structure of Sentences

There are a large number of propositions which could be parameterized.

Aristotle is a human

Example 18.2

All humans are mortal.

Aristotle is a human.

Therefore Aristotle is mortal.

Internal Structure of Sentences

There are a large number of propositions which could be parameterized.

All humans are mammals

Example 18.3

All humans are mammalian.

Aristotle is a human.

Therefore Aristotle is mammalian.

Parameterisation-1

x is a human

All humans are *y*

Example 18.4

All humans are y.

x is a human.

Therefore x is y.

Parameterisation-2

x is a *z*

All *z* are *y*

Example 18.5

All *z* are *y*.

x is a *z*.

Therefore *x* is *y*.

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 18.6 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 18.7 Given a signature Σ , the set $\mathcal{T}_\Sigma(V)$ 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 $t_1, \dots, t_m \in \mathcal{T}_\Sigma(V)$. 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 18.8 Given a signature Σ and the set $\mathcal{T}_\Sigma(V)$ 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	$ $	\top
	$ \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.

18.1. Predicates in Natural Languages

The translation of propositional sentences expressed in natural language into propositions (as defined in section 2.1) was actually pretty straight-forward (with a few exceptions). In general, one could easily identify the propositional connectives between the component sentences by a more or less direct transliteration. The translation was more or less “syntax-directed” and the structure of compound sentences was reflected in the structure of the translated propositions. Even then we had to take several exceptions into account in the use of conjunctions such as *and* and *or*.

The issue of translating natural language sentences with quantifiers is somewhat more complicated. One major complication is the fact that properties of objects, qualifications and subclasses of objects sharing common properties all have the same structure in natural language. The so-called *is-a* relationship is overloaded in a natural language. Another complication relates to the fact that predicate logic is about relationships and a property of an object is merely taken as a unary relation.

We begin first with identifying quantifiers and then go deeper into the issue of translation. Necessarily, most of what we state here with respect to translation is not to be taken as “gospel truth” but merely as a guideline and even those may be applicable only to the English language.

Operation	English rendering
$\forall x[\phi]$	for all x, ϕ for each x, ϕ for every x, ϕ for any x, ϕ

Operation	English rendering
$\exists x[\phi]$	for some x, ϕ there exists x , such that ϕ for at least one x, ϕ

Operation	English rendering
$\neg\forall x[\phi]$	not all x, ϕ not the case that for each x, ϕ not for every x, ϕ

Operation	English rendering
$\neg\exists x[\phi]$	for no x, ϕ there does not exist x , such that ϕ there is no x , such that ϕ

is-a statements

Consider the following statements which are universal statements with some implicit containment or implicational relationship between classes of objects.

1. *All humans are mortal.*
2. *All Greeks are human.*

3. All non-Greeks are also human.

Here the class of *humans* is a sub-class of the class of *mortals*, the class of *Greeks* is a sub-class of the class of *humans*, and the class of *non-Greeks* is a sub-class of the class of *humans*. We may think of unary predicates $m(x)$, $h(x)$, $g(x)$ as representing the characteristic property of the three classes respectively. In particular, they represent the properties of “mortality”, “human-ness” and “greek-hood”. The translation of the above sentences into predicate logic is then in terms of the properties of the sub-classes.

1. $\forall x[h(x) \rightarrow m(x)]$
2. $\forall x[g(x) \rightarrow h(x)]$
3. $\forall x[\neg g(x) \rightarrow h(x)]$

Once a sub-class is characterised by a property, other properties could also be used as qualifiers in conjunction with it. For example,

1. All rich humans are mortal.
2. All rich humans are obese mortals.

may be translated as follows (letting *r* stand for the quality of being *rich* and *o* for *obesity*).

1. $\forall x[(h(x) \wedge r(x)) \rightarrow m(x)]$

$$2. \forall x[(h(x) \wedge r(x)) \rightarrow (m(x) \wedge o(x))]$$

Action predicates

Besides the **is-a** relationships, predicates may also be used to represent actions (which are actually verbs and not adjectives or qualifiers). Consider the following statements.

1. *Ram wedded Sita.*
2. *The Pandavas wedded Draupadi.*
3. *Elizabeth Taylor was married.*

“Reasonable” translations which capture the meanings of these statements in terms of the binary action predicate *w* which we use to denote both *wedded* and *married* yield the following (with *r* for *Ram*, *s* for *Sita*, *d* for *Draupadi*, *p* for the class *Pandavas*, and *e* for *Elizabeth Taylor*).

1. $w(r, s)$
2. $\forall x[p(x) \rightarrow w(x, d)]$
3. $\exists x[w(x, e)]$

Then there are statements like

1. Nobody loves Charlie Brown.

2. Everybody loves Calvin.
3. There is nobody who does not love Calvin.

whose “reasonable” translations using l as a binary predicate symbol for *loves* and constant symbols b for *Charlie Brown* and c for *Calvin* could go something like

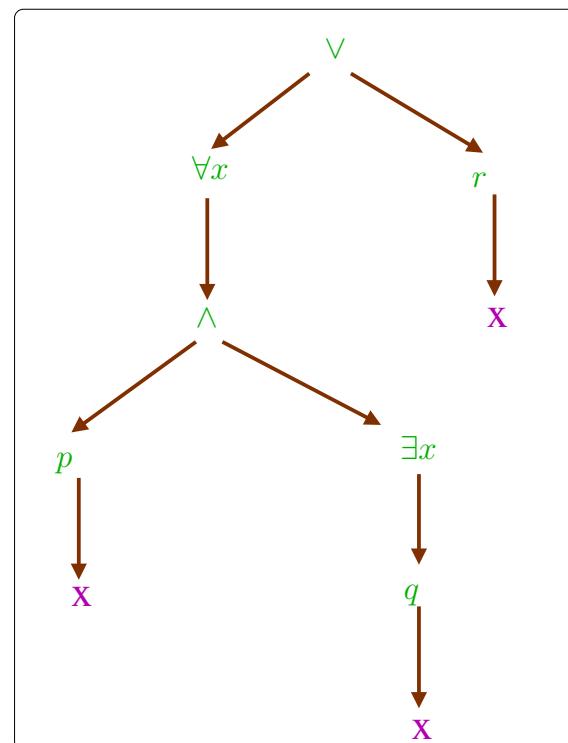
1. $\neg\exists x[l(x, b)]$
2. $\forall x[l(x, c)]$
3. $\neg\exists x[\neg l(x, c)]$

Notice that the last two statements mean the same thing, however they have syntactically different renderings. We will also see that they are actually logically equivalent statements.

One could go on and one with several examples. But since this is not meant to be exhaustive (and could never be) we will not take it any further, but instead rely on the “common-sense” to capture the intended meaning of a predicate or a sentence with quantifiers.

Predicates: Abstract Syntax Trees

Example 18.9 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 18.10 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 18.11

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

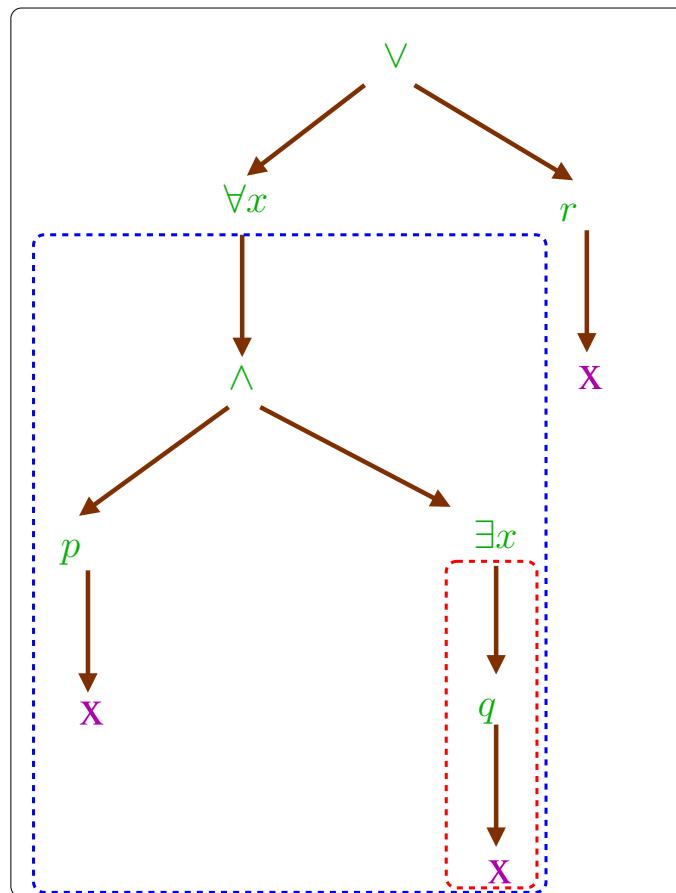
Definition 18.12 *In a formula of the form $\mathcal{O}x[\psi]$ the variable x is said to be **bound** by the quantifier \mathcal{O} . The brackets $[\dots]$ delimit the **scope** of the binding.*

Example 18.13 *In the **abstract syntax tree** of the predicate*

$$\forall x[p(x) \wedge \exists x[q(x)]] \vee r(x)$$

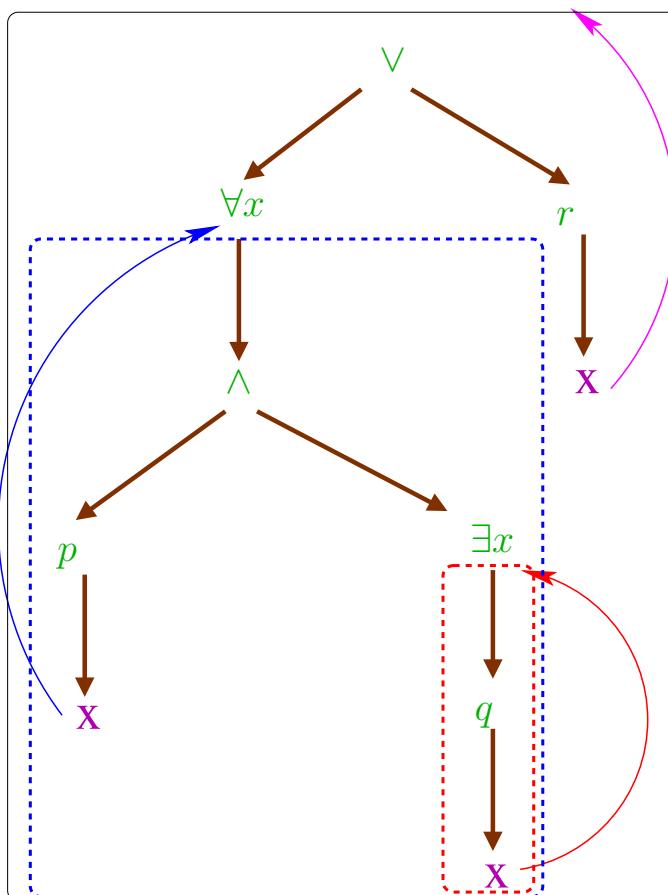
*the scopes of the various bindings are indicated by dashed boxes. Notice that the **red** scope is a “hole” in the **blue** scope.*

Bound Variables And Scope: Example



Scope Trees

The abstract syntax tree also determines the scope of the individual bound variables.



Free Variables

Definition 18.14 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 18.15 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 18.15 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 18.16

1. A formula ϕ is called **closed** if $FV(\phi) = \emptyset$. A closed formula is also called a **sentence**. $\mathcal{P}_1^\circ(\Sigma) \subseteq \mathcal{P}_1(\Sigma)$ is the set of first-order sentences over Σ .
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 18.1

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.

19: The Semantics of Predicate Logic

“There's glory for you!”

“I don't know what you mean by ‘glory’,” Alice said. Humpty Dumpty smiled contemptuously. “Of course you don't – till I tell you. I meant ‘there's a nice knock-down argument for you!’”

“But ‘glory’ doesn't mean ‘a nice knock-down argument’,” Alice objected.

“When I use a word,” Humpty Dumpty said in rather a scornful tone, “it means just what I choose it to mean – neither more nor less.”

“The question is,” said Alice, “whether you can make words mean so many different things.”

“The question is,” said Humpty Dumpty, “which is to be master – that's all.”

Lewis Carroll, *Through the Looking-Glass*

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 $p_{\mathbf{A}} \in \mathcal{D} = \{0, 1\}$ 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.
4. Another way to deal with *partial functions* is to “**guard**” the predicates in which they occur in such a way as to bound the domain of allowable values only to those for which the function is defined. The guard is itself another predicate. This is a technique that is useful where certain functions or predicates are defined only for some subset of the allowable values. Also see equation 65 and the predicate ϕ_{decomp} in section 41.4.

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 19.1 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 19.2 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 19.3 *Given a Σ -structure \mathbf{A} , a valuation (also called state) is a function $v_{\mathbf{A}} : V \rightarrow |\mathbf{A}|$ which assigns to each variable a unique value in $|\mathbf{A}|$.*

Definition 19.4 *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 19.5 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 19.6 *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 19.7 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 19.8

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 $v_X = v[\textcolor{violet}{x}_1 := \textcolor{red}{a}_1, \dots, \textcolor{violet}{x}_n := \textcolor{red}{a}_n]$ or $v_X = v[\textcolor{violet}{x}_1, \dots, \textcolor{violet}{x}_n := \textcolor{red}{a}_1, \dots, \textcolor{red}{a}_n]$.

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.

$$\begin{aligned}\mathcal{T}_{\mathbf{A}}[\![p(t_1, \dots, t_n)]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} 1 & \text{if } (\mathcal{V}_{\mathbf{A}}[\![t_1]\!]_{v_{\mathbf{A}}, \dots, \mathcal{V}_{\mathbf{A}}[\![t_n]\!]_{v_{\mathbf{A}}}) \in p_{\mathbf{A}} \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{T}_{\mathbf{A}}[\![\forall x[\phi]]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \prod \{\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v'_{\mathbf{A}}} \mid v'_{\mathbf{A}} =_{\setminus x} v_{\mathbf{A}}\} \\ \mathcal{T}_{\mathbf{A}}[\![\exists x[\phi]]\!]_{v_{\mathbf{A}}} &\stackrel{df}{=} \sum \{\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v'_{\mathbf{A}}} \mid v'_{\mathbf{A}} =_{\setminus x} v_{\mathbf{A}}\}\end{aligned}$$

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 $\{\mathcal{T}_A[\phi]_{v'_A} \mid v'_A =_{\setminus x} v_A\}$ 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. $1 \leq |\{\mathcal{T}_A[\phi]_{v'_A} \mid v'_A =_{\setminus x} v_A\}| \leq 2$ since there are only two possible truth values (even though there may be infinitely many x -variants of a valuation v_A). Hence the **sum and product** in the semantics of quantifiers are both *finitary*.

Exercise 19.1

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) $\forall x[\forall y[\forall z[e(x, y) \wedge e(y, z) \rightarrow e(x, z)]]]$

20. Substitutions

20: 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

Coincidence Lemma for Formulae

The following lemma analogous to lemma 19.6 may be proved by induction on the structure of formulae.

Lemma 20.1 *Given two Σ -interpretations (\mathbf{A}, v) and (\mathbf{A}, v') , and a formula ϕ , if $v(\textcolor{violet}{x}) = v'(\textcolor{violet}{x})$ for each $\textcolor{violet}{x} \in FV(\phi)$, then $\mathcal{T}[\![\phi]\!]_v = \mathcal{T}[\![\phi]\!]_{v'}.$*



Proof of The Coincidence Lemma for formulae (lemma 20.1)

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 $\phi \equiv \mathcal{O}(x_1, \dots, x_n)$. 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 $\mathcal{T}[\phi]_v = \mathcal{T}[\phi]_{v'_x}$.

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 20.2

- A **substitution** θ is a (total) function $\theta : V \rightarrow T(\Sigma)$ which is almost everywhere the identity.
- $S_\Sigma(V)$ 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 20.3 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 20.4 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 20.4

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

Free variable capture. Much care is required to actually prevent the so-called “capture” of free variables during the application of a substitution, because “free variable capture” by a binding of the same variable name (within the scope of the binding) can alter the meaning of expressions to some wholly unintended one²². The following example illustrates the meaning of this term

Example 20.5 Consider the expression $\sum_{z=1}^n y$ which may have been obtained as a result of some derivation or some calculation. Assuming all the symbols here have their usual meanings in the integers, we may safely assume that for any value of the free variables n and y , the value of this expression should be ny (i.e the product of n and y). So if we were to substitute any integer-valued function such as $y = x^2$, we expect that on substituting x^2 for y would yield the result nx^2 . However, if it so happened that y was defined to be $y = z^2$ instead, then the corresponding result obtained is unfortunately not nz^2 . Instead it becomes the sum of the first n squares of positive integers which is an unintended result of the substitution because of the confusion between two different occurrences of the variable z . This happens because the variable z which is free in the definition $y = z^2$ has got “captured” by the binding of z in $\sum_{z=1}^n y$ (see example 0.85?).

So whereas $\{x^2/y\}$ is a permitted substitution, the substitution $\{z^2/y\}$ cannot be permitted because the free variable z would be “captured” by the binding $\sum_{z=1}^n$ in the host term. This is expressed by saying that the substitution $\{z^2/y\}$ is not admissible in $\sum_{z=1}^n y$. The free variable z (unlike the bound variable z in the expression $\sum_{z=1}^n y$) has a meaning and significance outside the scope of the expression $\sum_{z=1}^n y$. The bound variable z , on the other hand, has its meaning and significance limited to the expression $\sum_{z=1}^n y$ and has no other significance outside it.

The only way to perform the substitution $\{z^2/y\}$ is to first ensure that all bound occurrences of z in the expression $\sum_{z=1}^n y$

²²Loosely put, you might get wrong or wholly unexpected and unintended answers.

are renamed to some variable which preferably does not clash with any other variable in that context. Since $\sum_{z=1}^n y$ may equally well be written $\sum_{u=1}^n y$ without changing any meanings it would then be safe to apply the substitution $\{z^2/y\}$ to $\sum_{u=1}^n y$ to yield the expected result.

We therefore require to define when a substitution of a term for a variable is “admissible”.

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** for a formula ϕ if the (free) variables of s remain free after instantiating the formula.

Definition 20.6 Let θ be a substitution

- An element $s/x \in \theta$ is **admissible** for
 - $p(t_1, \dots, t_n)$,
 - $\neg\phi$ if it is admissible for ϕ ,
 - $(\phi \odot \psi)$ if it is admissible for both ϕ and ψ , $\odot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
 - $\partial x[\phi]$,
 - $\partial y[\phi]$ if $x \not\equiv y$, $y \notin FV(s)$ and s/x is admissible for ϕ .
- θ is **admissible** for ϕ if every element of θ is admissible for ϕ .

Instantiations of Formulae

Definition 20.7 Let θ be a substitution which is admissible for ϕ .

- The application of θ to a formula $\phi \in \mathcal{P}_1(\Sigma)$ is denoted $\theta\phi$ and defined inductively as follows.

$$\theta p(t_1, \dots, t_n) = p(\theta t_1, \dots, \theta t_n), \quad p : s^n \in \Sigma$$

$$\theta \neg \psi = \neg(\theta \psi),$$

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

$$\theta \O \mathbf{x}[\psi] = \O \mathbf{x}[\theta' \psi], \quad \theta' = \theta - \{\theta(\mathbf{x})/\mathbf{x}\}, \quad \O \in \{\forall, \exists\}$$

Fact 20.8 If θ is a substitution which is admissible for ϕ then every $\theta' \subseteq \theta$ is also admissible for ϕ .

The Substitution Lemma for Formulae

Lemma 20.9 *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 20.9

Proof: Case $x \notin FV(\phi)$. Then it trivially follows that $\{s/x\}\phi \equiv \phi$ and $\mathcal{T}[\{s/x\}\phi]_v = \mathcal{T}[\phi]_{v_x}$ 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

$$\begin{aligned} & \mathcal{T}[\{s/x\}\phi]_v \\ = & \mathcal{T}[\{s/x\}p(t_1, \dots, t_n)]_v \\ = & \mathcal{T}[p(\{s/x\}t_1, \dots, \{s/x\}t_n)]_v \\ = & \mathcal{T}[p(t_1, \dots, t_n)]_{v[x:=a]} \quad \text{By lemma 20.4} \end{aligned}$$

The sub-cases involving the propositional connectives are trivial and the only interesting cases left are those of quantified formulae.

Sub-case $\phi \equiv \mathcal{O}y[\psi]$. Since $x \in FV(\phi)$ clearly $x \neq y$ and hence $x \in FV(\psi)$. Further since $\{s/x\}$ is admissible for ϕ , $y \notin Var(s)$. This implies that

1. $\{s/x\}$ is admissible for ψ ,
2. $\{s/x\}\phi \equiv \mathcal{O}y[\{s/x\}\psi]$ 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 20.1

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 $\mathcal{T}_{\mathbf{A}_1}[\!\phi]\!]_{v_1} = \mathcal{T}_{\mathbf{A}_2}[\!\phi]\!]_{v_2}$.

21: Models, Satisfiability and Validity

Everything is vague to a degree you do not realise till you have tried to make it precise.

Bertrand Russell, *Philosophy of Logical Atomism*, 1918

1. Satisfiability
2. Models and Consistency
3. Examples of Models:1
4. Other Models
5. Examples of Models:2
6. Examples of Models:3
7. Logical Consequence
8. Validity
9. Validity of Sets of Formulae
10. Negations of Semantical Concepts
11. Independence
12. Example of Independence

Satisfiability

Definition 21.1 (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 21.2 (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 21.3 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

$$\begin{array}{ll} \phi_1 \stackrel{df}{=} \neg(+1(x) = 0) & \phi_2 \stackrel{df}{=} (+1(x) = +1(y)) \rightarrow (x = y) \\ \phi_3 \stackrel{df}{=} \forall x \exists y [y = +1(x)] & \phi_4 \stackrel{df}{=} \forall x [\neg(x = 0) \rightarrow \exists y [x = +1(y)]] \end{array}$$

We have $\mathbf{N} \Vdash \Phi$ where $\Phi = \{\phi_1, \phi_2, \phi_3, \phi_4\}$

Other Models

A consistent set of axioms may describe not just one model but a class of models (some of them wholly unintended by the propounder of the axioms!).

Example 21.4 Consider an alphabet consisting of a single symbol $|$. Consider the carrier set $|^*$ consisting of finite sequences of $|$ (including the empty sequence denoted by ϵ)

$$\text{Bars} = \langle |^*; \epsilon, |.; =, < \rangle$$

where for any $x \in |^*$, $|.x = |x|$, $=$ is the usual equality relation on sequences of symbols and $<$ is the “proper prefix” ordering on sequences.

It is easy to verify that $\text{Bars} \Vdash \Phi$ and is hence a model of the axioms of example 21.3.

21.1. An excursion into Ordinals

For a given set of non-logical axioms it may be possible to construct several interesting models. For one we show that an alternative model satisfying the axioms that we have given for the naturals (example 21.3) may be easily constructed inductively using only sets starting from the empty set. Just to make clear the similarities and the differences of our construction with what we usually associate as the naturals we use underscored versions of symbols for the naturals. Let

$$\underline{0} \stackrel{df}{=} \emptyset$$

This is called the “zeroth limit ordinal”. It is also the “smallest finite ordinal”. This is the basis of our inductive construction. The inductive step for the construction of the successor ordinal $\underline{n+1}$ from \underline{n} for each natural number $n > 0$ is defined as

$$\underline{n+1} \stackrel{df}{=} \underline{n} \cup \{\underline{n}\}$$

That is, each successor ordinal (i.e. the image of the function $+1$ applied to each ordinal) is obtained from the previous ordinal by taking its union with the singleton set containing the previous ordinal.

Notice that this inductive construction yields

$$\begin{aligned}\underline{0} &= \emptyset \\ \underline{1} &= \{\underline{0}\} \\ \underline{2} &= \{\underline{0}, \underline{1}\} \\ \underline{3} &= \{\underline{0}, \underline{1}, \underline{2}\} \\ \vdots &\quad \vdots \quad \vdots \\ \underline{n+1} &= \{\underline{0}, \underline{1}, \underline{2}, \dots, \underline{n}\} \\ \vdots &\quad \vdots \quad \vdots\end{aligned}$$

and so on *ad infinitum*.

The ordinals satisfy the following properties for each natural m, n , which leads us to the definition of $<$ on the ordinals. To put it succinctly we have

$$\underline{m} < \underline{n} \stackrel{df}{=} \underline{m} \in \underline{n} \text{ iff } \underline{m} \subset \underline{n} \text{ iff } m < n \tag{30}$$

and by obvious extensions to the definitions of \leq , $>$ and \geq respectively.

Further notice that for each natural n we have

$$\underline{n} = \{\underline{m} \mid \underline{m} < \underline{n}\} = \bigcup_{\underline{m} < \underline{n}} \underline{m} = \{\underline{m} \mid m < n\} = \bigcup_{m < n} \underline{m} \quad (31)$$

Each ordinal is a set consisting of all the ordinals that are less than it. By extension there is absolutely nothing that prevents us from considering the set consisting of all the finite ordinals,

$$\underline{\omega} \stackrel{df}{=} \bigcup_{m \in \mathbb{N}} \underline{m} \quad (32)$$

and claiming it to be an ordinal too. $\underline{\omega}$ consists of all the finite ordinals and is greater than any of them. We see that for all $n \in \mathbb{N}$, by (30) we have that $\underline{n} < \underline{\omega}$ since $\underline{n} \in \underline{\omega}$ (equivalently $\underline{n} \subset \underline{\omega}$). Hence $\underline{\omega}$ is also an ordinal and consists of all the ordinals that are less than itself. $\underline{\omega}$ is the “first limit” ordinal that is “transfinite”. But our ordinal construction (unlike in the case \mathbb{N}) need not stop here. We could apply the same definition and come up with successor transfinite ordinals too.

$$\begin{aligned}
 \underline{\omega + 1} &\stackrel{df}{=} \underline{\omega} \cup \{\underline{\omega}\} \\
 &= \{\underline{0}, \underline{1}, \dots, \underline{\omega}\} \\
 \underline{\omega + 2} &\stackrel{df}{=} \underline{\omega + 1} \cup \{\underline{\omega + 1}\} \\
 &= \{\underline{0}, \underline{1}, \dots, \underline{\omega}, \underline{\omega + 1}\} \\
 \underline{\omega + 3} &\stackrel{df}{=} \underline{\omega + 2} \cup \{\underline{\omega + 2}\} \\
 &= \{\underline{0}, \underline{1}, \dots, \underline{\omega}, \underline{\omega + 1}, \underline{\omega + 2}\} \\
 &\vdots &&\vdots \\
 \underline{\omega + n + 1} &\stackrel{df}{=} \underline{\omega + n} \cup \{\underline{\omega + n}\} \\
 &= \{\underline{0}, \underline{1}, \dots, \underline{\omega}, \underline{\omega + 1}, \underline{\omega + 2}, \dots, \underline{\omega + n}\} \\
 &\vdots &&\vdots
 \end{aligned}$$

which brings us to the next limit ordinal viz.

$$\underline{\omega + \omega} \stackrel{df}{=} \{\underline{0}, \underline{1}, \dots, \underline{\omega}, \underline{\omega + 1}, \underline{\omega + 2}, \dots, \underline{\omega + n}, \dots\} \quad (33)$$

which we refer to as $\underline{\omega.2}$ (and not $\underline{2\omega}$) (we could also identify $\underline{\omega}$ with $\underline{\omega.1}$ and $\underline{0}$ with $\underline{\omega.0}$). Proceeding

further in a similar manner we get the sequence of ordinals

$$\underline{0}, \dots, \underline{\omega} = \underline{\omega.1}, \dots, \underline{\omega.2}, \dots, \underline{\omega.3}, \dots, \underline{\omega.n}, \dots, \underline{\omega.\omega} \quad (34)$$

We refer to $\underline{\omega.\omega}$ as $\underline{\omega^2}$. We could continue this even further obtaining ordinals,

$$\underline{0}, \dots, \underline{\omega} = \underline{\omega^1}, \dots, \underline{\omega^2}, \dots, \underline{\omega^3}, \dots, \underline{\omega^n}, \dots, \underline{\omega^\omega} \quad (35)$$

There really is no reason to stop here. Proceeding further from $\underline{\omega^\omega}$ we obtain

$$\underline{0}, \dots, \underline{\omega^\omega} = \underline{\omega^{(\omega^1)}}, \dots, \underline{\omega^{(\omega^2)}}, \dots, \underline{\omega^{(\omega^3)}}, \dots, \underline{\omega^{(\omega^n)}}, \dots, \underline{\omega^{(\omega^\omega)}} \quad (36)$$

Following the normal precedence convention for exponentials and assuming that exponentiation is right associative (see subsection 2.3) we may write the above sequence without the messy parentheses as follows.

$$\underline{0}, \dots, \underline{\omega^\omega} = \underline{\omega^{\omega^1}}, \dots, \underline{\omega^{\omega^2}}, \dots, \underline{\omega^{\omega^3}}, \dots, \underline{\omega^{\omega^n}}, \dots, \underline{\omega^{\omega^\omega}} \quad (37)$$

We really don't need to stop here and we could actually take it further and create a “staircase of ω s with an ω number of steps” as shown below

$$\underline{\omega}, \dots, \underline{\omega^\omega}, \dots, \underline{\omega^{\omega^\omega}}, \dots, \underline{\omega^{\omega^{\omega^{\omega^\dots}}}}, \dots, \underline{\omega^{\omega^{\omega^{\omega^{\dots}}}}}, \quad (38)$$

Let's stop here. Even though this construction is quite mind-boggling, it turns out that all these ordinals are only countable – i.e. we have not even scratched the surface of uncountability!

Exercise 21.1

1. Prove that $\omega + 1$ is a countable set.
2. Prove that ω^2 is a countable set.
3. Prove that each of the transfinite ordinals described in (34), (35), (36), (37) and (38) is countable.
4. Assuming that an ordinal α is countable, prove that its successor $\alpha + 1$ is also countable.
5. Assuming that $\beta \stackrel{df}{=} \bigcup_{\alpha < \beta} \alpha$ and each α is a countable ordinal, prove that β is also countable.
6. Assuming that the class of ordinals can be further extended “ad infinitum”, which of the axioms in example 21.3 does this class fail to satisfy? Justify your answer with an example from the transfinite ordinals.
7. If we limit ourselves to the class of ordinals ending with the last ordinal in (38) which other axioms of example 21.3 does it fail to satisfy?

Examples of Models:2

Example 21.5 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

$$\phi_{associative} \stackrel{df}{=} \forall x, y, z [(x + y) + z = x + (y + z)] \quad (39)$$

$$\phi_{identity} \stackrel{df}{=} \forall x [x + 0 = x] \quad (40)$$

$$\phi_{right-inverse} \stackrel{df}{=} \forall x \exists y [x + y = 0] \quad (41)$$

Examples of Models:3

Example 21.6 *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*

$$\phi_{\text{commutative}} \stackrel{df}{=} \forall x, y [x + y = y + x] \quad (42)$$

i.e. $\Phi' = \Phi \cup \{\phi_{\text{commutative}}\}$ excludes all non-commutative groups from the models of the set Φ' .

Logical Consequence

Definition 21.7 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 21.8 Let Σ be the signature in example 21.5 and let the formula

$$\phi_{\text{left-inverse}} \stackrel{\text{df}}{=} \forall x \exists y [y + x = 0] \quad (43)$$

A typical proof in a mathematics text that $\phi_{\text{left-inverse}}$ is a logical consequence of the axioms of group theory (example 21.5) might go as follows.

Proof: Let x be any element. Then by axiom $\phi_{right-inverse}$ there exists y such that

$$x + y = 0 \quad (44)$$

Again by $\phi_{right-inverse}$ for some z we get

$$y + z = 0 \quad (45)$$

We then have

$$\begin{aligned} y + x &= (y + x) + 0 && (\phi_{identity}) \\ &= (y + x) + (y + z) && (45) \\ &= y + (x + (y + z)) && (\phi_{associative}) \\ &= y + ((x + y) + z) && (\phi_{associative}) \\ &= y + (0 + z) && (44) \\ &= (y + 0) + z && (\phi_{associative}) \\ &= y + z && (\phi_{identity}) \\ &= 0 && (45) \end{aligned}$$

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

$$\phi_{\text{left-right-inverse}} \stackrel{df}{=} \forall x, y [(x + y = 0) \rightarrow (y + x = 0)] \quad (46)$$

is also a logical consequence of the group axioms. The formula (46) makes a “stronger” statement about groups in general viz. that **the right inverse of an element not only exists, but is its left inverse as well.**

Validity

Definition 21.9 (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 21.10 (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 $\phi \in \Phi$.
- ($\mathbf{\mathcal{A}} \Vdash \Phi$). Φ is valid in a class $\mathbf{\mathcal{A}}$ of Σ -structures if $\mathbf{A} \Vdash \Phi$ for each $\mathbf{A} \in \mathbf{\mathcal{A}}$.
- ($\Vdash \Phi$). Φ is valid if and only if every Σ -structure is a model of each $\phi \in \Phi$.

Meta-operators and overloading

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 $\models \phi \leftrightarrow \psi$ for all sentences (i.e. closed formulae) of first-order logic. However $\Vdash \phi \leftrightarrow \psi$ and $\models \phi \leftrightarrow \psi$ are not equivalent for formulae with free variables (see problem 15 in exercises 21.3)

Exercise 21.2

1. Consider the axioms ϕ_1 and ϕ_2 of example 21.3 replaced by

$$\phi'_1 \stackrel{df}{=} \forall x[\neg(+1(x) = 0)] \quad \phi'_2 \stackrel{df}{=} \forall x[(+1(x) = +1(y)) \rightarrow (x = y)]$$

- (a) Does **N** $\Vdash \Phi' = \{\phi'_1, \phi'_2, \phi_3, \phi_4\}$ hold? Why or why not?
 - (b) Does **Bars** $\Vdash \Phi' = \{\phi'_1, \phi'_2, \phi_3, \phi_4\}$ hold? Why or why not?
 - (c) Let $\phi''_2 \stackrel{df}{=} \forall x, y[(+1(x) = +1(y)) \rightarrow (x = y)]$ and let $\Phi'' = \{\phi'_1, \phi''_2, \phi_3, \phi_4\}$. Then do the following hold? Why or why not?
 - i. **Bars** $\Vdash \Phi''$
 - ii. **N** $\Vdash \Phi''$
2. Based on your answers to the above question what can you say is the relation between free variables in the axioms of a model, validity and logical consequence?
3. If instead of universally quantifying the statements, we had existentially quantified them, would your answers to the questions in the problems change? Justify.

Negations of Semantical Concepts

- We use the symbols \nVdash , $\not\models$, $\not\Leftarrow$ to denote the negations of the corresponding relations.

Independence

Given a nonempty set Φ of axioms that define a structure or a family of structures, it is interesting to know whether some of the axioms could be derived from the others in the set. If such is indeed the case then those which could be derived, may be dropped from the set Φ , since the set of logical consequences would be no different.

Definition 21.11 (Independence) A formula ψ is independent of a non-empty set Φ of formulae if $\Phi \not\models \psi$. Φ is an independent set (of formulae) if for every formula $\phi \in \Phi$, ϕ is independent of $\Phi - \{\phi\}$.

Example of Independence

Example 21.12 Let Σ be signature given in example 21.5. Consider the set of Σ -formulae

$$\Phi = \{\phi_{\text{associative}}, \phi_{\text{identity}}, \phi_{\text{right-inverse}}\}$$

It turns out that they form an independent set. In particular, we prove that each of the formulae in Φ is independent of the other two, by considering Σ -models which satisfy two of the formulae but not the third.

1. $\{\phi_{\text{associative}}, \phi_{\text{identity}}\} \not\models \phi_{\text{right-inverse}}$
2. $\{\phi_{\text{identity}}, \phi_{\text{right-inverse}}\} \not\models \phi_{\text{associative}}$
3. $\{\phi_{\text{associative}}, \phi_{\text{right-inverse}}\} \not\models \phi_{\text{identity}}$

1. **Independence of $\phi_{right-inverse}$.** Consider the structure $\mathbf{N} = \langle \mathbb{N}; 0, +; = \rangle$ under the usual interpretation. Since no positive number has an additive inverse it follows

$$\begin{aligned}\mathbf{N} &\Vdash \phi_{associative} \\ \mathbf{N} &\Vdash \phi_{identity} \\ \text{but } \mathbf{N} &\nvDash \phi_{right-inverse}\end{aligned}$$

Hence $\{\phi_{associative}, \phi_{identity}\} \not\models \phi_{right-inverse}$.

2. **Independence of $\phi_{associative}$.** Consider the structure $\mathbf{Z}_- = \langle \mathbb{Z}; 0, -; = \rangle$ where $+$ is interpreted as the integer subtraction operation $-$. Since \mathbb{Z} is closed under subtraction but is not associative and 0 is an identity for subtraction and there exist right inverses for each integer (0 is its own right inverse), it follows that

$$\begin{aligned}\mathbf{Z}_- &\Vdash \phi_{identity} \\ \mathbf{Z}_- &\Vdash \phi_{right-inverse} \\ \text{but } \mathbf{Z}_- &\nvDash \phi_{associative}\end{aligned}$$

Hence $\{\phi_{identity}, \phi_{right-inverse}\} \not\models \phi_{associative}$

3. **Independence of $\phi_{identity}$.** Let $\mathcal{B} = \{0, 1\}$ and let \mathcal{B}° be the set of all nonempty sequences of 0 s and 1 s separated by \circ which have no consecutive occurrences of either 0 or 1 . In particular, let \circ

be defined as the associative operation on such sequences such that

$$\begin{aligned}1 \circ 1 &= 0 = 0 \circ 0 \\s \circ 1 \circ 1 \circ t &= s \circ t = s \circ 0 \circ 0 \circ t\end{aligned}$$

$\mathbf{2}^\circ = \langle \mathbb{Z}^\circ; 0, \circ; = \rangle$. Further for every sequence, s , there exists a unique sequence s^R called the “reverse of s ” such that $s \circ s^R = 0$. It then follows that

$$\begin{aligned}\mathbf{2}^\circ &\Vdash \phi_{associative} \\ \mathbf{2}^\circ &\Vdash \phi_{right-inverse} \\ \text{but } \mathbf{2}^\circ &\not\Vdash \phi_{identity}\end{aligned}$$

Hence $\{\phi_{associative}, \phi_{right-inverse}\} \not\models \phi_{identity}$

Exercise 21.3

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 21.7. Justify or refute his claim.

Definition 21.13 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 $(\mathbf{A}, v) \Vdash \psi$.

16. Prove that $\phi_{commutative}$ is independent of the axioms Φ .

21.2. Some more Model Theory

When we consider the notion of a model, we come across various notions which express properties of the models. For example, if \langle 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 [4] 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 21.14 assures us that such a possibility does not exist.

Theorem 21.14 *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 $k \geq 2$,

$$\begin{aligned}\psi_2 &\stackrel{df}{=} \exists x_1, x_2 [\neg(x_1 = x_2)] \\ \psi_3 &\stackrel{df}{=} \exists x_1, x_2, x_3 [\neg(x_1 = x_2) \wedge \neg(x_1 = x_3) \wedge \neg(x_2 = x_3)] \\ &\vdots \quad \vdots \quad \vdots\end{aligned}$$

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 November 4, 2018

Go BACK

FULL SCREEN

CLOSE

615 OF 1040

QUIT

22: 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

Satisfiability and Expansions

Our notions of satisfiability, consequence and validity are all with respect to a specific signature.

Lemma 22.1 *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 20.1. QED

Distinguishability

Example 22.2 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 $\mathbf{Z} \nvDash \phi_{density}$.

$\phi_{density}$ distinguishes the two structures.

Evaluations under Different Structures

When evaluating terms and formulae under different structures say A and B using valuations $v_A : V \rightarrow A$ and $v_B : V \rightarrow B$ where $A = |A|$ and $B = |B|$ we use $\mathcal{V}_A, \mathcal{T}_A$ and $\mathcal{V}_B, \mathcal{T}_B$ respectively to distinguish the possibly different values and truth values.

Isomorphic Structures

Example 22.3 Let $\Sigma = \{0 : \rightarrow s, +1 : s \rightarrow s, = : s^2, < : s^2\}$.

Now consider the structures

$$\begin{aligned}\mathbf{N} &= \langle \mathbb{N}; 0, +1; =, < \rangle \\ 2\mathbf{N} &= \langle 2\mathbb{N}; 0, +2; =, < \rangle\end{aligned}$$

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 22.4 (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 22.5 *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 22.4

Proof: Assume there is an isomorphism $\pi : \mathbf{A} \cong \mathbf{B}$. Since π is an isomorphism, $\pi : A \xrightarrow[\text{onto}]{} B$ is a bijection that preserves structure, where $A = |\mathbf{A}|$ and $B = |\mathbf{B}|$. Hence $\pi^{-1} : B \xrightarrow[\text{onto}]{} A$ 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 $v_{\mathbf{A}} : V \rightarrow A$, $\pi \circ v_{\mathbf{A}} : V \rightarrow B$ and for each $v_{\mathbf{B}} : V \rightarrow B$, $\pi^{-1} \circ v_{\mathbf{B}} : V \rightarrow A$.

For every formula ϕ , we may prove the following stronger claims by induction on the structure of ϕ .

$$\begin{aligned}\mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{v_{\mathbf{A}}} &= \mathcal{T}_{\mathbf{B}}[\![\phi]\!]_{\pi \circ v_{\mathbf{A}}} \\ \mathcal{T}_{\mathbf{B}}[\![\phi]\!]_{v_{\mathbf{B}}} &= \mathcal{T}_{\mathbf{A}}[\![\phi]\!]_{\pi^{-1} \circ v_{\mathbf{B}}}\end{aligned}$$

The proof is 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 21.9.

Substructures

Definition 22.6 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 22.7 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 22.8

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 22.9 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 22.10 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 22.4 on the common subset of the two carrier sets using the identity isomorphism.
QED

Universal and Existential Formulae

Definition 22.11 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 22.12 (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 22.1

1. **Theorem 22.13 (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 22.13.

2. Prove that if ϖ is not surjective, the two structures may be distinguished by a formula.

3. In part 1b of theorem 22.13 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 22.13 may be extended to quantified formulae.

23: 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. Theories of Predicate Calculus
11. Axioms for Equality
12. Symmetry and Transitivity
13. Symmetry of Equality
14. 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 23.1 $\mathcal{H}_1(\Sigma)$, 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 $\forall E$

$$\forall x[\neg \forall y[p(x, y)]] \rightarrow \neg \forall y[p(y, y)] \quad (47)$$

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 $X = \phi(x) = Y$.

$$\forall x[\phi(x) \rightarrow \phi(x)] \rightarrow (\phi(x) \rightarrow \forall x[\phi(x)]) \quad (48)$$

Whereas the antecedent of the formula in (48) 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

$$\begin{array}{c} \forall E \frac{\forall x[g(x) \rightarrow h(x)]}{g(y) \rightarrow h(y)} \quad \forall E \frac{\forall x[h(x) \rightarrow m(x)]}{h(z) \rightarrow m(z)} \\ \hline ?? \frac{g(y) \rightarrow h(y)}{\forall I \frac{g(y) \rightarrow m(y)}{\forall x[g(x) \rightarrow m(x)]}} \end{array}$$

A Correct Proof

$$\frac{\frac{\forall E \frac{\forall x[g(x) \rightarrow h(x)]}{g(y) \rightarrow h(y)}}{\frac{\forall E \frac{\forall x[h(x) \rightarrow m(x)]}{h(y) \rightarrow m(y)}}{\frac{\forall I \frac{g(y) \rightarrow m(y)}{\forall x[g(x) \rightarrow m(x)]}}{}}}}{}}$$

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 22.1).
 - (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 23.2 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 23.3 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**.

Theories of Predicate Calculus

The First-order theory of the Predicate Calculus will be denoted by $\text{Th}(\mathbb{P}\mathbb{C})$.

The First-order theory of Predicate Calculus with Equality with no non-logical axioms except the axioms for equality will be denoted by $\text{Th}(\mathbb{P}\mathbb{C} =)$.

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 =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

$$\frac{\text{MP} \frac{\text{=S } \Delta \vdash t = u \rightarrow (s = t \rightarrow s = u)}{\Delta \vdash t = u}}{\Delta \vdash s = t \rightarrow s = u} \Delta \vdash s = u$$

24: 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

Alpha Conversion

Notation We use “ $\phi \dashv\vdash \psi$ ” as an abbreviation for the two statements “ $\phi \vdash \psi$ ” and “ $\psi \vdash \phi$ ”.

Lemma 24.1 *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 $\forall x[\phi] \dashv\vdash \forall y[\{y/x\}\phi]$

Proof tree of $\forall x[\phi] \vdash \forall y[\{y/x\}\phi]$

$$\text{MP} \frac{\overline{\forall x[\phi] \vdash \forall x[\phi]} \quad \forall E \frac{\overline{\forall x[\phi] \vdash \forall x[\phi] \rightarrow \{x/y\}\{y/x\}\phi}}{\forall x[\phi] \vdash \{x/y\}\{y/x\}\phi}}{\forall x[\phi] \vdash \forall y[\{y/x\}\phi]}$$

Proof tree of $\forall y[\{y/x\}\phi] \vdash \forall x[\phi]$

$$\text{MP} \frac{\overline{\forall y[\{y/x\}\phi] \vdash \forall y[\{y/x\}\phi]} \quad \forall E \frac{\overline{\forall y[\{y/x\}\phi] \vdash \forall y[\{y/x\}\phi] \rightarrow \{y/x\}\phi}}{\forall y[\{y/x\}\phi] \vdash \{y/x\}\phi}}{\forall y[\{y/x\}\phi] \vdash \forall x[\phi]}$$

The Deduction Theorem for Predicate Calculus

Theorem 24.2 (The Deduction Theorem for Predicate Calculus).

Let $\Gamma \subseteq_f \mathcal{L}_1(\Sigma)$ and $\phi, \psi \in \mathcal{L}_1(\Sigma)$.

1. (DT \Leftarrow). If $\Gamma \vdash \phi \rightarrow \psi$ then $\Gamma, \phi \vdash \psi$.
2. (DT \Rightarrow). Let $\Gamma, \phi \vdash \psi$ such that no variable in $FV(\phi)$ is generalized (by an application of $\forall I$) in the proof. Then $\Gamma \vdash \phi \rightarrow \psi$.



Proof of the Deduction Theorem for Predicate Calculus (theorem 24.2)

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. 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.

$$\frac{j' \frac{\bigwedge \mathcal{T}_j \bigwedge}{\psi_j} \quad j' + 1 \frac{\psi_j \rightarrow (\phi \rightarrow \psi_j)}{\phi \rightarrow \psi_j}}{j' + 2 \frac{}{}}$$

Suppose ψ_j was obtained by the application of the axiom schema $\forall I$ on some ψ_i such that $i < j$. 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.

$$\frac{i' + 3}{\phi \rightarrow \forall x[\psi_i] \equiv \phi \rightarrow \psi_j} \frac{i' + 1 \frac{i'}{\phi \rightarrow \psi_i} \quad i' + 2 \frac{\forall x[\phi \rightarrow \psi_i] \rightarrow (\phi \rightarrow \forall x[\psi_i])}{\forall x[\psi_i] \rightarrow (\phi \rightarrow \forall x[\psi_i])}}{\forall x[\phi \rightarrow \psi_i] \rightarrow (\phi \rightarrow \forall x[\psi_i])}$$

QED

Useful Corollaries

Corollary 24.3 *If the proof of $\Gamma, \phi \vdash \psi$ involves no generalization of any free variable of ϕ then $\Gamma \vdash \phi \rightarrow \psi$.*

Corollary 24.4 *If ϕ is a closed formula and $\Gamma, \phi \vdash \psi$ then $\Gamma \vdash \phi \rightarrow \psi$.*

Corollary 24.5 *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 24.6 *Every wff of $\mathcal{L}_1(\Sigma)$ 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 24.7 (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 24.7

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 $\text{Th}(\text{PC})$ is a tautology.

If $\text{Th}(\text{PC})$ were inconsistent then $\text{Th}(\text{PC}) = \mathcal{L}_1$. In particular, there exist formulae $\phi, \neg\phi \in \text{Th}(\text{PC})$. 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 24.8 (Soundness of \mathcal{H}_1). *Every theorem of \mathcal{H}_1 is logically valid.*



Proof of theorem 24.8

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 24.1

1. Prove the arguments in Problem 2 of exercise 18.1 using the system \mathcal{H}_1 .
 2. Prove the conclusions of Problem 2 in exercise 26.1 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) $\vdash \forall x[\psi \rightarrow \phi] \leftrightarrow (\exists x[\psi] \rightarrow \phi)$

25: Existential Quantification

1. Existential Quantification
2. Existential Elimination
3. Remarks on Existential Elimination
4. Restrictions on Existential Elimination
5. Equivalence of Proofs

Existential Quantification

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 $\exists I$

Proof: Assuming $\{t/x\}$ is admissible in ϕ suppose $\Gamma \vdash \{t/x\}\phi$.

$$\frac{\text{MP} \quad \frac{\forall E \quad \overline{\Gamma \vdash \forall x[\neg\phi] \rightarrow \neg\{t/x\}\phi} \quad N'' \quad \overline{\Gamma \vdash (\forall x[\neg\phi] \rightarrow \neg\{t/x\}\phi) \rightarrow (\neg\neg\{t/x\}\phi \rightarrow \neg\forall x[\neg\phi])}}{\Gamma \vdash \neg\neg\{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]}$$

From the root of the tree we have

$$T \rightarrow \frac{}{\Gamma \vdash \neg\neg\{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]} \quad \text{DNI} \quad \frac{\Gamma \vdash \{t/x\}\phi \rightarrow \neg\neg\{t/x\}\phi}{\Gamma \vdash \{t/x\}\phi \rightarrow \neg\forall x[\neg\phi]}$$

Assuming $\Gamma \vdash \{t/x\}\phi$, the last step gives

$$\text{MP} \quad \frac{\Gamma \vdash \{t/x\}\phi \rightarrow \neg\forall x[\neg\phi] \quad \Gamma \vdash \{t/x\}\phi}{\Gamma \vdash \neg\forall x[\neg\phi] \equiv \exists x[\phi]}$$

Note. In the last step “ $\equiv \exists x[\phi]$ ” is not really part of the proof tree. It is just an abuse of (meta-)notation to indicate that $\neg\forall x[\neg\phi]$ is really its **syntactic equivalent** $\exists x[\phi]$. QED

Existential Elimination

Existential generalisation from a constant introduced somehow into the proof is very common in mathematics.

Example 25.1 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 proof using $\exists E$ ($\Gamma \vdash_{\exists E} \exists x[\psi]$)

$$\frac{\begin{array}{c} \exists E \frac{\Gamma \vdash \exists x[\phi \rightarrow \psi]}{\Gamma \vdash \{a/x\}(\phi \rightarrow \psi) \equiv \{a/x\}\phi \rightarrow \{a/x\}\psi} \\ MP \end{array}}{\exists I \frac{\Gamma \vdash \{a/x\}\psi}{\Gamma \vdash \exists x[\psi]}} \quad \forall E \frac{\Gamma \vdash \forall x[\phi]}{\Gamma \vdash \{a/x\}\phi}$$

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.

A proof without using $\exists E$ ($\Gamma \vdash \exists x[\psi]$)

Let $\Delta = \{\forall x[\phi], \forall x[\neg\psi]\}$

$$\frac{\begin{array}{c} \forall E \frac{\Delta \vdash \forall x[\phi]}{\Delta \vdash \phi} \quad \forall E \frac{\Delta \vdash \forall x[\neg\psi]}{\Delta \vdash \neg\psi} \\ \wedge I \frac{}{\Delta \vdash \neg(\phi \rightarrow \psi)} (\neg(\phi \rightarrow \psi) \equiv \phi \wedge \neg\psi) \\ \forall I \frac{}{\Delta \vdash \forall x[\neg(\phi \rightarrow \psi)]} \\ DT \Rightarrow \frac{}{\forall x[\phi] \vdash \forall x[\neg\psi] \rightarrow \forall x[\neg(\phi \rightarrow \psi)]} \end{array}}{\forall x[\phi] \vdash \forall x[\neg\psi] \rightarrow \forall x[\neg(\phi \rightarrow \psi)]}$$

Note that the application of **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

$$\text{N"} \frac{}{\forall x[\phi] \vdash (\forall x[\neg\psi] \rightarrow \forall x[\neg(\phi \rightarrow \psi)]) \rightarrow (\neg\forall x[\neg(\phi \rightarrow \psi)] \rightarrow \neg\forall x[\neg\psi])}$$

An application of **MP** then yields

$$\text{DT} \Leftarrow \frac{\forall x[\phi] \vdash \neg\forall x[\neg(\phi \rightarrow \psi)] \rightarrow \neg\forall x[\neg\psi]}{\forall x[\phi], \neg\forall x[\neg(\phi \rightarrow \psi)] \equiv \exists x[\phi \rightarrow \psi] \vdash \neg\forall x[\neg\psi] \equiv \exists x[\psi]}$$

Note. The whole proof could have been written as a single “monolithic” proof tree with full justification. We have however split the tree and written out the various sub-trees in order to explain some of the steps and to be able to fit it into the limited page-size available.

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 25.2 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 25.3 (Existential-Elimination Elimination Theorem).

If $\Gamma \vdash_{\exists E} \phi$ is a correct proof then $\Gamma \vdash \phi$ (i.e. ϕ is provable from Γ without making use of the $\exists E$ rule) provided no constants introduced in the proof $\Gamma \vdash_{\exists E} \phi$ occur in ϕ .



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 25.3

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 $\frac{\nwarrow \mathcal{T} \nearrow}{\Gamma \vdash_{\exists E} \phi}$.

Claim 1. For each i , $1 \leq i \leq k$ there exist proof trees

$$\frac{\nwarrow \mathcal{T}'_i \nearrow}{\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{i-1}/y_{i-1}\}\psi_{i-1} \vdash \exists y_i[\psi_i]}$$

which are completely free from any application of rule $\exists E$ and

$$\frac{\nwarrow \mathcal{T}^i \nearrow}{\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_i/y_i\}\psi_i \vdash_{\exists E} \phi}$$

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

$$\frac{}{\exists E \frac{\nwarrow \mathcal{T}_1 \nearrow}{\Gamma \vdash \exists y_1[\psi_1]} \quad \frac{}{\Gamma \vdash_{\exists E} \{a_1/y_1\}\psi_1}}$$

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 25.2 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

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \{a_k/y_k\}\psi_k \rightarrow \phi \quad (49)$$

Since the original proof is correct by definition 25.2, there is no occurrence of any of the constants a_i , $1 \leq i \leq k$, in ϕ . Further the proof (49) 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 (49)) does not affect the correctness of the proof.

Take a fresh variable z_k which does not occur anywhere in the proof (49) and replace all occurrences of a_k by z_k to obtain proof (50).

$$\Gamma, \{a_1/y_1\}\psi_1, \dots, \{a_{k-1}/y_{k-1}\}\psi_{k-1} \vdash \{z_k/y_k\}\psi_k \rightarrow \phi \quad (50)$$

Proof (50) 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 24.1.4d 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

Step (50) seems like some sleight of hand procedure which could look unsound to some readers, but its soundness cannot be questioned as the following argument shows.

To simplify our argument, let us assume that a proof of $\Gamma \vdash \exists y[\psi]$ exists. We know that the system \mathcal{H}_1 is sound (theorem 24.8). For simplicity assume that y is the only free variable of ψ . Further let $\Gamma \vdash \{a/y\}\psi \rightarrow \phi$ and suppose the proof obtained by replacing all occurrences of a by a fresh variable z were not sound, then there exists an interpretation \mathbf{A} such that $\mathbf{A} \Vdash \Gamma$, and $\mathbf{A} \Vdash \exists y[\psi]$, but that for some valuation v , where $v(y) = a$, $(\mathbf{A}, v) \Vdash \psi$ holds but $\mathbf{A} \Vdash \phi$ does not hold. But then this implies that $(\mathbf{A}, v) \not\Vdash \{a/y\}\psi \rightarrow \phi$ and hence $\mathbf{A} \not\Vdash \{a/y\}\psi \rightarrow \phi$. However $\Gamma \vdash \{a/y\}\psi \rightarrow \phi$ and the soundness of \mathcal{H}_1 imply that $(\mathbf{A}, v) \Vdash \{a/y\}\psi \rightarrow \phi$ which is a contradiction.

Exercise 25.1

1. Use the method outlined in the proof of theorem 25.3 to transform the proof $\Gamma \vdash_{\exists E} \exists x[\psi]$ of example 25.1 to one without the use of rule $\exists E$.

26: Normal Forms

1. Natural Deduction: 6
2. Moving Quantifiers
3. Quantifier Movement
4. More on Quantifier Movement
5. Quantifier-Free Formulae
6. Prenex Normal Forms
7. The Prenex Normal Form Theorem
8. Prenex Conjunctive Normal Form
9. The Herbrand Algebra
10. Terms in a Herbrand Algebra
11. Herbrand Interpretations
12. Herbrand Models
13. 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 26.1

1. Prove the arguments in Problem 2 of exercise 18.1 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.*²³
 - (c) *Therefore if I love you, then you love me.*
 - (d) *Therefore you love yourself.*
 - (e) *Therefore everyone loves everyone.*
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?

²³This is of course a dirty trick!

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 26.1 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 26.2

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 26.2

1. Prove lemma 26.1.
2. Prove corollary 26.2.

Having obtained the above results we are now ready to prove the Prenex normal form theorem. But first let's define the form precisely.

Quantifier-Free Formulae

Definition 26.3 *The set $\mathcal{QF}_1(\Sigma)$ of quantifier-free formulae over a signature Σ is generated by the grammar*

$$\chi, \theta ::= p(t_1, \dots, t_n) \mid \neg \chi \mid \chi \wedge \theta \mid \chi \vee \theta \mid \chi \rightarrow \theta \mid \chi \leftrightarrow \theta$$

Prenex Normal Forms

Definition 26.4 *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 26.5 (Prenex Normal Forms). *For any formula ϕ there exists a logically equivalent formula ψ in prenex normal form (PNF).*



Proof of theorem 26.5

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 variables²⁴.
3. Now proceed by induction on the structure of ϕ' by systematically applying the results obtained from lemma 26.1 and corollary 26.2. This would yield a formula ψ in prenex normal form.

QED



²⁴That 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 5.12 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 26.6 (PCNF). *For any formula ϕ there exists a logically equivalent formula ψ in prenex conjunctive normal form (PCNF).*



The Herbrand Algebra

Definition 26.7 Let Σ be a signature containing at least one constant symbol a . A term $t \in \mathbb{T}_\Sigma(V)$ is said to be **ground** if $Var(t) = \emptyset$ (see definition 0.82). $\mathbb{T}_\Sigma \subseteq \mathbb{T}_\Sigma(V)$ 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**.

Substitution lemma for terms

Substitution lemma for formulae

Definition 26.8 A Σ -algebra $\mathbf{H}(\Sigma)$ where Σ has at least one constant symbol, is called a **Herbrand algebra** iff $|\mathbf{H}(\Sigma)| = \mathbb{T}_\Sigma$.

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 $v_{\mathbf{H}(\Sigma)} : V \rightarrow T_\Sigma$
- However the predicate symbols require to be associated with relations on ground terms in some way.
- When Σ is understood, we will often omit it and simply write \mathbf{H} instead of $\mathbf{H}(\Sigma)$ and $v_{\mathbf{H}}$ instead of $v_{\mathbf{H}(\Sigma)}$.

Herbrand Interpretations

Lemma 26.9 Given a Herbrand interpretation $(\mathbf{H}, v_{\mathbf{H}})$ where for each variable x , $v_{\mathbf{H}}(x) = s_x \in \mathbb{T}_{\Sigma}$. 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 26.10 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 26.11 Let Σ be a signature containing at least one constant and let $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ be a nonempty finite 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 26.11

Proof: Notice that every literal in Λ is ground and hence each of them is an instance of an atomic predicate (or negation of one) and has no free variables. Hence each of them is actually a proposition. Further note that for $i \neq j$, λ_i and λ_j could be instances of the same atomic predicate symbol or instances of negations of the same atomic predicate symbol or one could be a positive instance and the other a negative instance of the same atomic predicate.

1. Clearly if Λ contains a complementary pair it does not have a model since $\bigwedge_{1 \leq i \leq k} \lambda_i \Leftrightarrow \perp$. Conversely assume it does not contain a complementary pair. We may define a Herbrand model (algebra) \mathbf{H}_Λ as follows: For each atomic predicate symbol $p : s^n$ define

$$p_{\mathbf{H}_\Lambda} = \{(t_1, \dots, t_n) \in \mathbb{T}_\Sigma \mid p(t_1, \dots, t_n) \in \Lambda\}$$

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 unless each λ_i is valid. From the previous part we know that both λ_i and $\overline{\lambda_i}$ have models, 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 $\bigwedge_{1 \leq i \leq k} \overline{\lambda_i}$ has no model iff $\overline{\Lambda} = \{\overline{\lambda_i} \mid 1 \leq i \leq k\}$ contains a complementary pair iff Λ contains a complementary pair.

QED

27: Skolemization

1. Skolemization
2. Skolem Normal Forms
3. SCNF
4. Ground Instances
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 27.1 (Skolem Normal Form Theorem.) Let $\phi \equiv \vec{\forall} \vec{x} \exists \vec{y} [\psi]$ where $\vec{x} = x_1, \dots, x_n$ and \vec{y} are all distinct variables and ψ does not contain any occurrence of any of the quantifiers ∂x_i . Let $\Sigma_g = \Sigma \cup \{g : s^n \rightarrow s\}$ be an *expansion* of the signature Σ . Then

1. every model of $\phi' \equiv \vec{\forall} \vec{x} [\{g(x_1, \dots, x_n)/y\} \psi]$ is a model of ϕ .
2. every model of ϕ can be expanded to a model of ϕ' .

□

Corollary 27.2 Let ϕ and ϕ' be as in theorem 27.1. 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 27.1

Proof: 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 $\mathbf{A} \triangleleft \mathbf{B}$ 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 27.3

1. *The set $\mathcal{SNF}_1(\Sigma)$ of Skolem normal forms (SNF) 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 27.4 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 $\Sigma \triangleleft \Sigma'$ and ϕ has a Σ -model iff ψ has a Σ' -model.



Definition 27.5

1. The function g in theorem 27.1 is called a Skolem function
2. The process of constructing the function g in theorem 27.1 is called Skolemization.
3. ϕ and $sko(\phi) \equiv \psi$ are said to be equi-satisfiable.

Proof of theorem 27.4

Proof: The procedure may be briefly outlined as follows.

1. Clearly if ϕ is a closed formula, by theorem 26.6 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 27.1. QED

Exercise 27.1

1. Prove that $\models \phi' \rightarrow \phi$ in theorem 27.1.
2. Prove that $\vdash \phi' \rightarrow \phi$ in theorem 27.1.
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 Instances

Definition 27.6 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 $\{\{t_1/x_1, \dots, t_n/x_n\}\chi \mid FV(\chi) = \{x_1, \dots, x_n\}, t_1, \dots, t_n \in \mathbb{T}_\Sigma\}$ and $\mathfrak{g}(\Phi) = \bigcup_{\phi \in \Phi} \mathfrak{g}(\phi)$.

Herbrand's Theorem

Theorem 27.7 (Herbrand's Theorem). *Let Σ and Φ be as in definition 27.6. 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 27.7

Proof: Clearly the following implications are trivial.

Statement 2: " Φ has a Herbrand model" \Rightarrow Statement 1: " Φ has a model"



Statement 4: " $\mathfrak{g}(\Phi)$ has a Herbrand model" \Rightarrow Statement 3: " $\mathfrak{g}(\Phi)$ has a model"

It suffices therefore to prove only the following claim.

Claim. Statement 3 \Rightarrow Statement 2.

Proof of claim. Let $\mathbf{A} \Vdash \mathfrak{g}(\Phi)$. We define a Herbrand interpretation $(\mathbf{H}, v_{\mathbf{H}})$ as follows. For each $p : s^n \in \Sigma$ let

$$p_{\mathbf{H}} = \{(t_1, \dots, t_n) \in \mathbb{T}_{\Sigma} \mid \mathbf{A} \Vdash p(t_1, \dots, t_n)\}$$

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 $\mathbf{A} \Vdash \mathfrak{g}(\Phi)$ 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.

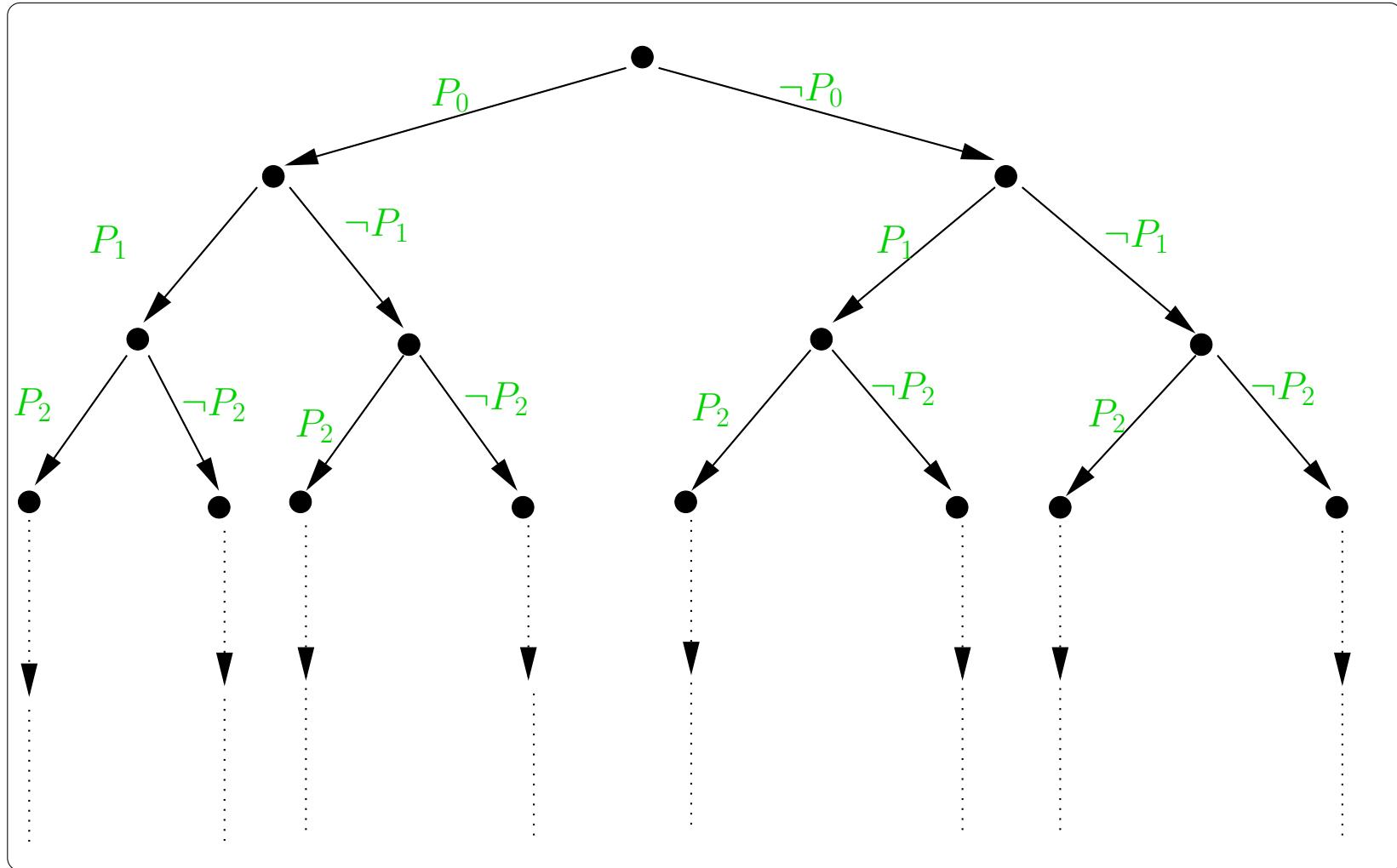


Figure 12: The Herbrand tree of interpretations

Compactness of Sets of Ground Formulae

Lemma 27.8 (Compactness of a set of ground 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 27.8

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 Herbrand's theorem (theorem 27.7) 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 0.69: 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 27.9 (Compactness of closed formulae) *A set Φ of closed Σ -formulae has a model iff every finite subset of Φ has a model.*



Corollary 27.10 (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 27.9

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 27.4). By Herbrand's theorem (theorem 27.7) the set $g(sko(\Phi))$ also does not possess a model. By lemma 27.8 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 model, contradicting the assumption that every finite subset of Φ has a model. QED

The Löwenheim-Skolem Theorem

Theorem 27.11 (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 27.7 $sko(\Phi)$ 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

28: Substitutions and Instantiations

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 27.11, 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 28.1

1. *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 28.2 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 28.3

- $\theta = \{t_i/x_i \mid t_i \not\equiv x_i, 1 \leq i \leq n\}$ 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 ,

$$\tau(\theta u) \equiv \rho u \quad (51)$$

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 (51) 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 28.4 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 28.5 *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 28.5

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 28.4. 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 28.1

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 28.6 A substitution $\theta = \{t_i/x_i \mid 1 \leq i \leq m\}$ for some $m \geq 0$ is idempotent iff $dom(\theta) \dots$.

29: 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

29.1. Syntactic Unification as equation solving

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 syntactically 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.

More generally, given a non-empty finite collection of Σ -terms, $T = \{t_i \in \mathbb{T}_\Sigma(V) \mid 1 \leq i \leq n\}$, the problem of unification is to find a substitution θ , if one exists, such that

$$|\theta T| = |\{t_i \in \mathbb{T}_\Sigma(V) \mid 1 \leq i \leq n\}| = 1$$

and otherwise to be able to declare that T cannot be unified.

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 29.1

t	$depth$	$size$	ST	pos
c	1	1	$\{t\}$	$\{\epsilon\}$
x	1	1	$\{t\}$	$\{\epsilon\}$
$o(t_1, \dots, t_n)$	$1 + Max_{i=1}^n depth(t_i)$	$1 + \sum_{i=1}^n size(t_i)$	$\{t\} \cup \bigcup_{i=1}^n ST(t_i)$	$\{\epsilon\} \cup \bigcup_{i=1}^n i.pos(t_i)$

- The functions given in the table above are defined by induction on the structure of term t . ϵ is the empty word on strings, $.$ is the catenation operator on strings and $i.pos(t_i) = \{i.p \mid p \in pos(t_i)\}$.
- $s \sqsubseteq t$ iff $s \in ST(t)$ 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 $p \in pos(t)$ is denoted $t|_p$ and defined by induction on p as follows: $t|_\epsilon \equiv t$, and for $t \equiv o(t_1, \dots, t_n)$, $t|_p \equiv t_i|_{p'}$ if $p = i.p' \in pos(t)$

- For any term t and any position $p \in pos(t)$, $sym(p, t)$ 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 $p \in pos(t)$, $sym(p, t) = rootsym(t|_p)$.
- The set of occurrences of a symbol $\sigma \in \Sigma \cup V$ in a term is defined as the set $Occ(\sigma, t)$ of positions in which that symbol occurs i.e. $Occ(\sigma, t) = \{p \in pos(t) \mid sym(p, t) = \sigma\}$.

Facts 29.2 For any term t and positions $p, q \in pos(t)$, $t|_q \sqsubset t|_p$ iff $p \prec q$.

Unifiability

Definition 29.3 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 29.4 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 29.5 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 29.6 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 29.1

1. Generalize the fact 29.6 to nonempty finite sets of terms.
2. Construct an example to show that the converse of fact 29.6.3 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 29.4.

Definition 29.7

- 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 29.8

1. \gtrsim is a preordering relation on $\mathbf{S}_\Omega(V)$ i.e. it is a reflexive and transitive relation.
2. \gtrless is an irreflexive and transitive relation on $\mathbf{S}_\Omega(V)$.
3. \sim is an equivalence relation on $\mathbf{S}_\Omega(V)$.
4. If $\theta \sim \tau$ and $\rho \circ \theta = \tau$ then ρ is a pure-variable substitution.

Most General Unifiers

Definition 29.9 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 29.10

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 29.11 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 29.2). 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 29.12 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 29.12) in a term.

Example: Disagreement 1

Example 29.13 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 29.14 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 29.15 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 29.16 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 29.17 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: $S \subseteq_f \mathbb{T}_\Omega(V)$ 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: UNIFY( $S$ )  $\stackrel{df}{=} \text{PARTIALUNIFY}(\mathbf{1}, S)$  where
2: PARTIALUNIFY( $\theta, T$ )  $\stackrel{df}{=}$ 
3: if  $|T| = 1$  then
4:   return  $\theta$  { $\theta$  is a mgu of  $S$ }
5: else {There is a position at which at least two terms are different}
6:   let  $D = \text{DISAGREEMENT}(T)$  in
7:   if  $\exists x \in D \cap V : \forall t \in T[x \notin FV(t)]$  then
8:     Choose  $t \in T : x \notin FV(t)$ 
9:     PARTIALUNIFY( $\{t/x\} \circ \theta, \{t/x\}T$ )
10:    { $T = \theta S \wedge |\{t/x\}T| < |T| \wedge |(\{t/x\} \circ \theta)S| < |\theta S| \leq |S|$ }
11:   else {Occurs check fails so  $S$  is not unifiable}
12:     return fail
13:   end if
14: end let
15: end if
```

Example 29.18 Consider the set $S = S_1$ in example 29.13. 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	$\theta_1 = \{a/z\} \circ \theta_0$	$\theta_1 S = \{f(a, x, h(g(\textcolor{red}{a}))), f(\textcolor{red}{a}, h(y), h(y)\}$	$D_1 = \{x, h(y)\}$
2	$\theta_2 = \{h(y)/x\} \circ \theta_1$	$\theta_2 S = \{f(a, \textcolor{red}{h}(y), h(g(\textcolor{red}{a}))), f(a, h(y), h(y)\}$	$D_2 = \{g(\textcolor{red}{a}), y\}$
3	$\theta_3 = \{g(a)/y\} \circ \theta_2$	$\theta_3 S = \{f(a, h(\textcolor{red}{g}(a)), h(\textcolor{red}{g}(a))\}$	$D_3 = \emptyset$

Hence the required unifier is $\theta_3 = \{g(a)/y\} \circ \{h(y)/x\} \circ \{a/z\} \circ \mathbf{I} = \{a/z, h(g(z))/x, g(z)/y\}$.

Example 29.19 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 29.20 (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 29.20

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 7 in the **unification algorithm** either it terminates because it fails or a substitution $\{t_{j+1}/x_{j+1}\}$ such that $x_{j+1} \notin FV(t_{j+1})$ is generated with $\theta_{j+1} = \{t_{j+1}/x_{j+1}\} \circ \theta_j$, we have $\theta_{j+1}S$ has one variable less than θ_jS . 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

The following are all simple consequences of the algorithm and the proof of its correctness.

Corollary 29.21 Let θ be the mgu of a unifiable set S computed by the *unification algorithm*. Then for some $k > 0$

1. $\theta = \theta_k = \{t_k/x_k\} \circ \cdots \circ \{t_1/x_1\}$.
2. For each j with $0 < j \leq k$, let $\theta_j = \{t_j/x_j\} \circ \cdots \circ \{t_1/x_1\}$. Then
 - (a) $\{x_j, \dots, x_1\} \cap FV(t_j) = \emptyset$
 - (b) $\{x_j, \dots, x_1\} \cap FV(\theta_j S) = \emptyset$
 - (c) If ρ is any unifier of S , then for each j , $0 < j \leq k$,
 - i. $\rho = \tau_j \circ \theta_j$ for some τ_j .
 - ii. for each i , $0 < i \leq j$, $(\theta_i \circ \theta_j)S = \theta_j S$



Exercise 29.2

1. Generalize the facts 29.6 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 29.4 and 29.5
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.

Hint: Use corollary 29.21

30: 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 and Herbrand's Theorem
10. Resolution in FOL
11. The Resolution Rule for 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 27.6) 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 30.1 Let the SCNF $sko(\psi)$ be represented by a set

$$sko(\psi) = \{C_i \mid 1 \leq i \leq m\}$$

such that

$$sko(\psi) \equiv \vec{\forall} [\bigwedge_{1 \leq i \leq m} C_i]$$

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 30.2

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 30.3

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 30.4 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 26.2. 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 30.5

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 and Herbrand's Theorem

From Herbrand's Theorem, compactness and finite unsatisfiability we have

Proposition 30.6

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 $\text{g}(S')$ does not possess a model.



Resolution in FOL

Compare with resolution in propositional logic

Let S be a non-empty finite 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

- $L_i = \{p(\vec{s}_{i'}) \in C_i \mid 1 \leq i' \leq m_i\} \neq \emptyset$
- $L_j = \{\neg p(\vec{t}_{j'}) \in C_j \mid 1 \leq j' \leq m_j\} \neq \emptyset$
- $L = L_i \cup \overline{L_j}$ is a set of unifiable literals.
- $C'_i = C_i - L_i$ and $C'_j = C_j - L_j$
- $\mu = \text{UNIFY}(L)$ 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 .

The Resolution Rule for FOL

Compare with resolution in propositional logic

$$\text{Res1} \quad \frac{S}{(S - \{C_i, C_j\}) \cup \{C'_{ij}\}}$$

Note.

1. L_i and L_j are merely non-empty finite sets of literals such that $L = L_i \cup \overline{L_j}$ is unifiable. They need not exhaust all literals naming p in C_i or C_j .
2. Unlike **propositional resolution**, each application of the resolution rule may not eliminate any atomic predicate symbol.
3. Therefore the resolvent C'_{ij} could contain occurrences of the predicate symbol p (in both positive and negative forms) which may be used for **other** steps in resolution.

31: More on Resolution in FOL

1. Standardizing Variables Apart
2. Factoring
3. Example: 1
4. Example: 2
5. Refutation
6. Refutations: Using the Herbrand Universe

Standardizing Variables Apart

1. The free variables of distinct clauses *must* be disjoint and variables should be renamed if necessary.

Example 31.1 Let $S = \{C_1, C_2\}$ where

$$\begin{aligned}C_1 &= \{p(\textcolor{violet}{x})\} \\C_2 &= \{\neg p(\textcolor{violet}{f}(x))\}\end{aligned}$$

S represents the set of closed formulae

$$\{\forall \textcolor{violet}{x}[p(\textcolor{violet}{x})], \forall \textcolor{violet}{x}[\neg p(\textcolor{violet}{f}(x))]\}$$

which is clearly unsatisfiable. However the empty clause cannot be derived (because of the *occurs check*) unless $\textcolor{violet}{x}$ 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 31.2 Consider the set $S = \{C_1, C_2\}$ where

$$\begin{aligned}C_1 &= \{p(\textcolor{violet}{x}), p(\textcolor{violet}{y})\} \\C_2 &= \{\neg p(\textcolor{violet}{u}), \neg p(\textcolor{violet}{v})\}\end{aligned}$$

S is clearly unsatisfiable but by removing only one literal at a time with the substitution $\{\textcolor{violet}{x}/\textcolor{violet}{u}\}$ yields the new clause

$$C'_{12} = \{p(\textcolor{violet}{y}), \neg p(\textcolor{violet}{v})\}$$

from which the empty clause cannot be derived.

Example: 1

Example 31.3 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.

$$\frac{C_1 = \{\neg q(\textcolor{violet}{x}, y), \underline{\neg q(\textcolor{violet}{y}, z)}, q(\textcolor{violet}{x}, z)\}, \{\neg q(\textcolor{violet}{u}, v), \underline{q(\textcolor{violet}{v}, u)}\} = C_2}{\{\neg q(\textcolor{violet}{x}, y), q(\textcolor{violet}{x}, z), \neg q(\textcolor{violet}{z}, y)\} = C'_{12}} \mu = \{z/u, y/v\}$$

$$\begin{aligned} C'_{12} &= \{\neg q(\textcolor{violet}{x}, y), \neg q(\textcolor{violet}{z}, y), q(\textcolor{violet}{x}, z)\} \\ &\equiv \forall \textcolor{violet}{x}, y, z [(q(\textcolor{violet}{x}, y) \wedge q(\textcolor{violet}{z}, y)) \rightarrow q(\textcolor{violet}{x}, z)] \end{aligned}$$

Example: 2

Example 31.4 Suppose we need to prove that if a binary relation p is reflexive

$$\phi_{p\text{-reflexive}} \stackrel{df}{=} \forall \mathbf{x}[p(\mathbf{x}, \mathbf{x})]$$

and euclidean

$$\phi_{p\text{-euclidean}} \stackrel{df}{=} \forall \mathbf{x}, \mathbf{y}, \mathbf{z}[(p(\mathbf{x}, \mathbf{y}) \wedge p(\mathbf{x}, \mathbf{z})) \rightarrow p(\mathbf{y}, \mathbf{z})]$$

then it is also symmetric

$$\phi_{p\text{-symmetric}} \stackrel{df}{=} \forall \mathbf{x}, \mathbf{y}[p(\mathbf{x}, \mathbf{y}) \rightarrow p(\mathbf{y}, \mathbf{x})]$$

After renaming bound variables and converting into SCNF to get the clauses

$$\begin{aligned}C_1 &= \{\underline{p(x, x)}\} \\C_2 &= \{\neg p(u, v), \neg p(u, w), p(v, w)\}\end{aligned}$$

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 31.5 We could also prove symmetry in example 31.4 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.

$$\frac{\{C_1 = \{p(\textcolor{violet}{x}, \textcolor{violet}{x})\}, C_2 = \{\neg p(\textcolor{violet}{u}, \textcolor{violet}{v}), \neg p(\textcolor{violet}{u}, \textcolor{violet}{w}), p(\textcolor{violet}{v}, \textcolor{violet}{w})\}, C_3 = \{p(\textcolor{red}{a}, \textcolor{red}{b})\}, C_4 = \{\neg p(\textcolor{red}{b}, \textcolor{red}{a})\}\}}{\{C'_{12} = \{\neg p(\textcolor{violet}{x}, \textcolor{violet}{v}), \underline{p(\textcolor{violet}{v}, \textcolor{violet}{x})}\}, C_3 = \{p(\textcolor{red}{a}, \textcolor{red}{b})\}, C_4 = \{\neg p(\textcolor{red}{b}, \textcolor{red}{a})\}\}} \mu = \{x/u, x/w\}$$

$$\frac{\{C'_{124} = \{\neg p(\textcolor{red}{a}, \textcolor{red}{b}), \}, C_3 = \{p(\textcolor{red}{a}, \textcolor{red}{b})\}, \}}{\{\{\}\}} \mu' = \{a/x, b/v\}$$

$$\frac{\{\}}{\{\{\}\}} \mu'' = \mathbf{1}$$

Alternatively a proof starting with the clauses C_3 and C_4 works too.

$$\frac{\{C_1 = \{p(\textcolor{violet}{x}, \textcolor{violet}{x})\}, C_2 = \{\neg p(\textcolor{violet}{u}, \textcolor{violet}{v}), \neg p(\textcolor{violet}{u}, \textcolor{violet}{w}), p(\textcolor{violet}{v}, \textcolor{violet}{w})\}, C_3 = \{p(\textcolor{red}{a}, \textcolor{red}{b})\}, C_4 = \{\neg p(\textcolor{red}{b}, \textcolor{red}{a})\}\}}{\{C_1 = \{p(\textcolor{violet}{x}, \textcolor{violet}{x})\}, C_3 = \{p(\textcolor{red}{a}, \textcolor{red}{b})\}, C'_{24} = \{\neg p(\textcolor{violet}{u}, \textcolor{red}{b}), \neg p(\textcolor{violet}{u}, \textcolor{red}{a})\}\}} \theta_1 = \{b/v, a/w\}$$

$$\frac{\{C_1 = \{p(\textcolor{violet}{x}, \textcolor{violet}{x})\}, C'_{234} = \{\neg p(\textcolor{red}{a}, \textcolor{red}{a})\}\}}{\{\{\}\}} \theta_2 = \{a/u\}$$

$$\frac{\{\}}{\{\{\}\}} \theta_3 = \{a/x\}$$

Refutations: Using the Herbrand Universe

Example 31.6 Let a be a constant symbol and f a unary function symbol. We use first-order resolution to prove that

$$\Phi = \{\neg p(a), p(f(f(a))), \forall x[p(x) \vee \neg p(f(x))]\}$$

is unsatisfiable.

The Herbrand universe consists of at least the following terms

$$\{a, f(a), f(f(a)), \dots, f^n(a), \dots\}$$

where $f^n(a)$ refers to the n -fold application of f to a .

The set of ground clauses of Φ therefore is a some superset of

$$\{\{\neg p(a)\}, \{p(f^2(a))\}\} \cup \{\{p(f^i(a)), \neg p(f^{i+1}(a))\} \mid i \geq 0\}$$

We could take the clauses in the given order as follows:

$$\begin{aligned} & \{\neg p(a)\}, \{p(f^2(a))\}, \{p(a), \neg p(f(a))\}, \{p(f(a)), \neg p(f^2(a))\}, \\ & \{p(f^2(a))\}, \neg p(f^3(a)), \dots \end{aligned}$$

Resolving on $p(a)$ $\{p(f^2(a))\}, \{\neg p(f(a))\}, \{p(f(a)), \neg p(f^2(a))\}, \dots$

Resolving on $p(f(a))$ $\{p(f^2(a))\}, \{\neg p(f^2(a))\}, \{p(f^2(a)), \neg p(f^3(a))\}, \dots$

Resolving on $p(f^2(a))$ $\{\}, \{p(f^2(a)), \neg p(f^3(a))\}, \dots$

by which the empty clause has been derived through propositional resolution. To use first-order resolution one would proceed as follows

$$\{\neg p(a)\}, \{p(f^2(a))\}, \{p(x), \neg p(f(x))\}$$

substituting $\{a/x\}$ $\{\neg p(a)\}, \{p(f^2(a))\}, \{p(a), \neg p(f(a))\}, \{p(x_1), \neg p(f(x_1))\}$

Resolving on $p(a)$ $\{p(f^2(a))\}, \{\neg p(f(a))\}, \{p(x_1), \neg p(f(x_1))\}, \dots$

substituting $\{f(a)/x_1\}$ $\{p(f^2(a))\}, \{\neg p(f(a))\}, \{\neg p(f^2(a))\}, \{p(x_2)\}, \neg p(x_2)\}, \dots$

Resolving on $p(f^2(a))$ $\{\}, \{p(x_2)\}, \neg p(x_2)\}$

which has derived the empty clause.

Exercise 31.1

1. Prove the conclusion of example 31.3 by a refutation.
2. Are there any other unifiers by which symmetry may be proved in example 31.4?
3. Try a refutation proof using $\nu = \{x/u, x/w, x/v\}$ as the first unifier in example 31.5. Why doesn't it work?
4. Try a refutation proof using $\nu = \{x/u, x/w, y/v\}$ as the first unifier in example 31.5. Why does it work?

32: 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

Soundness of FOL Resolution

Lemma 32.1 *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 32.2 *If S' is the set of clauses obtained by a single application of the resolution rule **Res1**, then $S \models S'$.*



Corollary 32.3 *If the empty clause is derivable from a set S of clauses, then S is unsatisfiable.*



Proof of lemma 32.1

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

$$\mathbf{A} \Vdash \vec{\forall}[\bigvee_{\lambda_i \in C_i} \lambda_i] \quad (52)$$

$$\mathbf{A} \Vdash \vec{\forall}[\bigvee_{\lambda_j \in C_j} \lambda_j] \quad (53)$$

and for any substitution θ we have

$$\mathbf{A} \Vdash \theta \bigvee C_i \quad (54)$$

$$\mathbf{A} \Vdash \theta \bigvee C_j \quad (55)$$

If θ is a unifier of L and $\theta L = \{\lambda\}$ we get

$$\mathbf{A} \Vdash \bigvee (\{\lambda\} \cup \theta C'_i) \quad (56)$$

$$\mathbf{A} \Vdash \bigvee (\{\bar{\lambda}\} \cup \theta C'_j) \quad (57)$$

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

$$\{\{\lambda\} \cup \theta C'_i, \{\bar{\lambda}\} \cup \theta C'_j\} \models \theta C'_i \cup \theta C'_j \quad (58)$$

It follows also from (52), (53) and (58) 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 32.4 (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 $C'_{ij} = C'_i \cup C'_j$ and for some literal λ , $C_i = C'_i \cup \{\lambda\}$ and $C_j = C'_j \cup \{\bar{\lambda}\}$. Note that there is no substitution involved anywhere since all clauses are ground.

Proof of theorem 32.4.

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 26.11, 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 some $m > 0$, for all k , $0 \leq \#G = k < m$, 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 D_i resulting in the set G and performing exactly the same sequence of resolutions as in \mathcal{R}_1 .

- 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

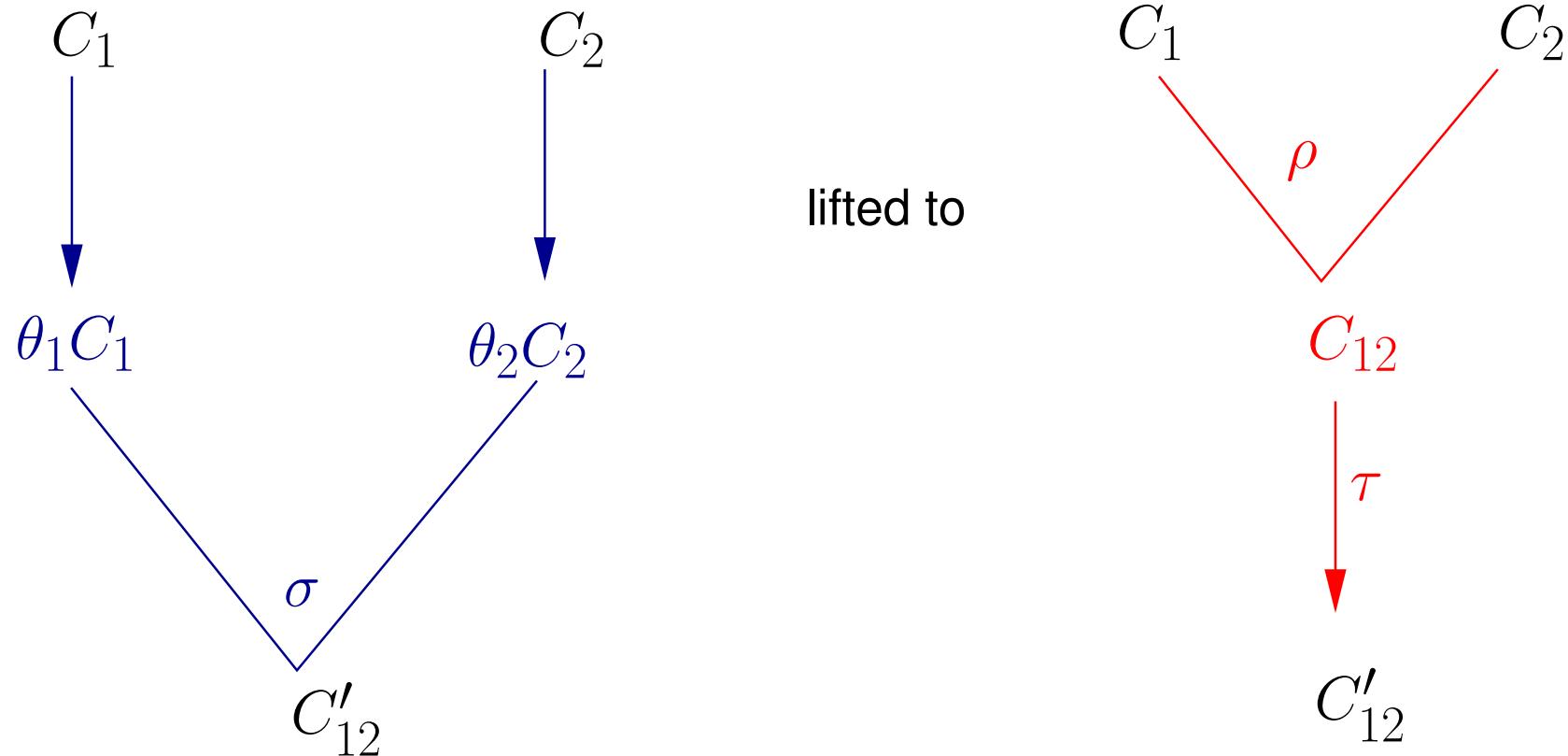
Lemma 32.5 (Lifting Lemma). (see *figure*) Let C_1 and C_2 be clauses and let $\theta_1, \theta_2, \sigma$ be substitutions such that

- $FV(C_1) \cap FV(C_2) = \emptyset$,
- $FV(\theta_1 C_1) \cap FV(\theta_2 C_2) = \emptyset$ 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 32.5

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

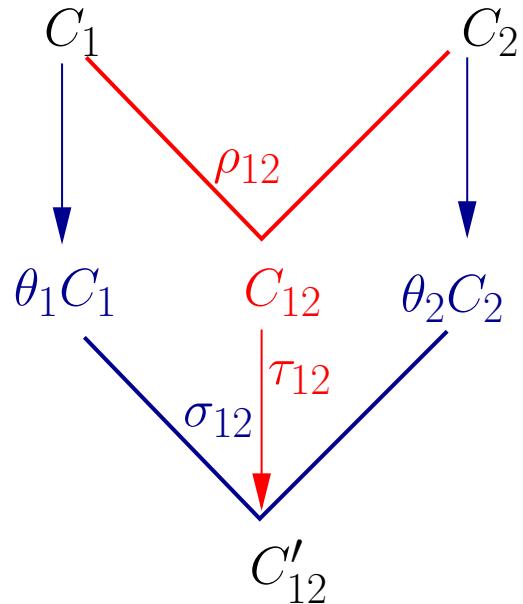
$$\tau \circ \rho = \sigma \circ (\theta_1 \cup \theta_2)$$

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 **32.5**) 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 32.5** 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 **27.7** 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 32.6 (Completeness of Resolution Refutation). *If a set Φ of clauses is unsatisfiable then the empty clause is derivable from Φ .*

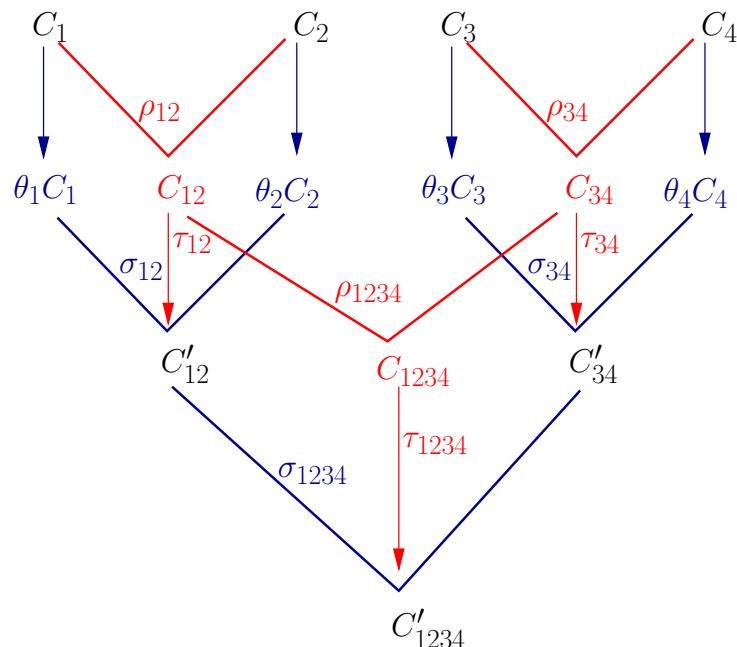


Proof of theorem 32.6

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 27.7 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

33: Resolution and Tableaux

1. FOL: Tableaux
2. FOL: Tableaux Rules
3. FOL Tableaux: Example 1
4. FOL Tableaux: Example 1 Contd
5. First-Order Tableaux
6. FOL Tableaux: Example 2
7. FOL Tableaux: Example 2 Contd

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.

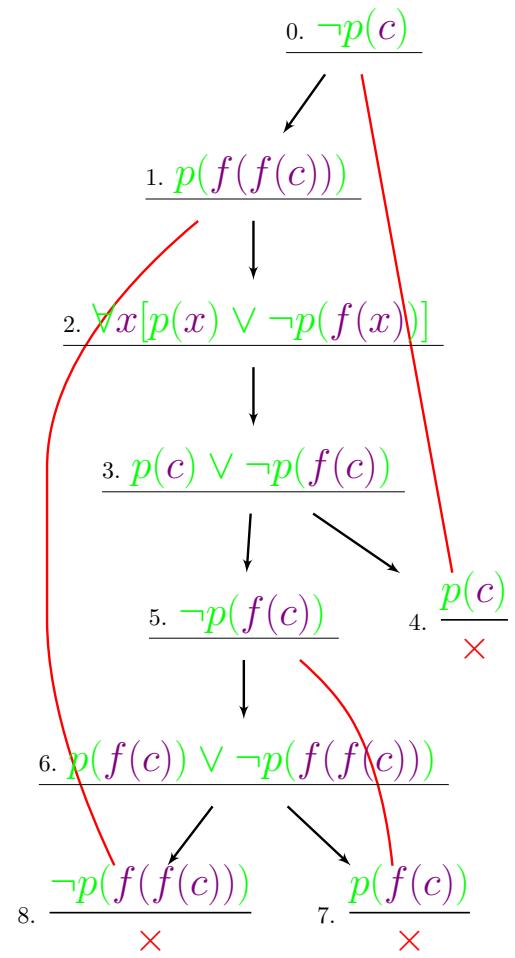
$\forall.$ $\frac{\forall x[\phi]}{\{t/x\}\phi}$	$\neg\forall.$ $\frac{\neg\forall x[\phi]}{\neg\{a/x\}\phi}$
$\exists.$ $\frac{\exists x[\phi]}{\{a/x\}\phi}$	$\neg\exists.$ $\frac{\neg\exists x[\phi]}{\neg\{t/x\}\phi}$

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 33.1 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.

Notice the **two applications** of rule $\forall E$. “ \times ” 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 $\forall x[p(x) \rightarrow q(x)] \models \forall x[p(x)] \rightarrow \forall x[q(x)]$

0. $\forall x[p(x) \rightarrow q(x)]$



1. $\neg(\forall x[p(x)] \rightarrow \forall x[q(x)])$



2. $\forall x[p(x)]$



3. $\neg\forall x[q(x)]$



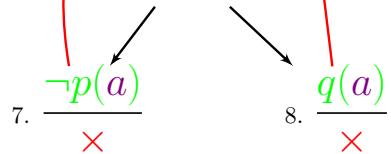
4. $\neg q(a)$



5. $p(a)$



6. $p(a) \rightarrow q(a)$



7. $\frac{\neg p(a)}{\times}$

8. $\frac{q(a)}{\times}$

34: 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

First-order Hintikka Sets

Definition 34.1 A finite or infinite set Γ is a **first-order Hintikka set** with respect to $\mathcal{P}_1(\Sigma)$ if

- 0-3. Γ is a (propositional) Hintikka set (definition 10.7) such that all the atomic propositions are ground.
- 4. For every $t \in \mathbb{T}_\Sigma$ (ground term).
 - $\forall x[\phi] \in \Gamma$ implies $\{t/x\}\phi \in \Gamma$,
 - $\neg\exists x[\phi] \in \Gamma$ implies $\neg\{t/x\}\phi \in \Gamma$
- 5. For at least one $t \in \mathbb{T}_\Sigma$ (ground term).
 - $\exists x[\phi] \in \Gamma$ implies $\{t/x\}\phi \in \Gamma$,
 - $\neg\forall x[\phi] \in \Gamma$ implies $\neg\{t/x\}\phi \in \Gamma$

Hintikka's Lemma for FOL

Lemma 34.2 *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 34.3 *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 34.4 (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 34.5 (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

35: Completeness of the Hilbert System

1. Deductive Consistency
2. Models of Deductively Consistent Sets
3. Deductive Completeness
4. The Completeness Theorem

Deductive Consistency

Definition 35.1 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 35.2 *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.



35.1. 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 12.1) 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 21.2. Notice that definition 10.1 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 35.1) 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 35.3 *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 35.4 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 35.5 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 35.4

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 14.1) $\Phi \cup \{\neg\forall[\phi]\} \vdash_{\mathcal{H}_1} \phi$ 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 35.2) 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 24.4 (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 35.4.

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 35.6 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 (theorem 12.10). Even the proof of the theorem mirrors the [alternative proof](#) of Lindenbaum's theorem.

Theorem 35.7 (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

$$\psi_1, \psi_2, \psi_3, \dots \quad (59)$$

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.

\vdash 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 35.4 since $\Phi_i \not\vdash_{\mathcal{H}_1} \psi_i$ and ψ is a closed formula, Φ_{i+1} must be deductively consistent. \dashv

Then $\Phi_\infty = \bigcup_{i \geq 0} \Phi_i$ is the desired set.

Claim. Φ_∞ is deductively consistent.

\vdash Suppose not. Then by lemma 35.2, 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. \dashv

Claim. Φ_∞ is deductively complete.

⊤ For any arbitrary closed formula ψ , ψ occurs in the enumeration 59 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 35.5 Φ_∞ is deductively complete. ⊢

QED

The Completeness Theorem

Theorem 35.8 (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 .



Proof: Assume $\Phi \models \phi$ and suppose $\Phi \not\vdash_{\mathcal{H}_1} \phi$. Then by lemma 35.4 $\Phi \cup \{\neg \forall[\phi]\}$ 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



36: 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
15. The Principle of Mathematical Induction
16. First-order Arithmetic
17. Extending First-order Arithmetic

Recall that a *sentence* in first-order logic is a closed formula (one with no free variables).

Definition 36.1 For any Σ -structure \mathbf{A} the **first-order theory of \mathbf{A}** , denoted $\text{Th}\mathbf{A}$ is the set of all sentences true in \mathbf{A} .

$$\text{Th } \mathbf{A} = \{\phi \in \mathcal{L}_1(\Sigma) \mid FV(\phi) = \emptyset, \mathbf{A} \models \phi\}$$

For any class \mathfrak{A} of Σ -structures,

$$\text{Th } \mathfrak{A} = \{\phi \in \mathcal{L}_1(\Sigma) \mid FV(\phi) = \emptyset, \mathfrak{A} \models \phi\}$$

Note.

1. There is always a *smallest* first-order theory consisting of all the logically valid sentences of first-order logic.
2. The *largest* first-order theory consisting of all the sentences in the language is unsatisfiable since ϕ and $\neg\phi$ both belong to the theory (for any sentence ϕ).

Definition 36.2 For any set Γ of sentences in $\mathcal{L}_1(\Sigma)$ the set

$$\text{LC } \Gamma = \{\phi \in \mathcal{L}_1(\Sigma) \mid FV(\phi) = \emptyset, \Gamma \models \phi\}$$

is the set of all **logical consequences** of Γ .

Definition 36.3 Given a deduction system \mathcal{D} , (which may or may not be complete) and a set Γ of sentences, we may define the set

$$\mathcal{DC} \Gamma = \{\phi \in \mathcal{L}_1(\Sigma) \mid FV(\phi = \emptyset, \Gamma \vdash_{\mathcal{D}} \phi)\}$$

of all deductive consequences of Γ .

Fact 36.4 For any set of closed Σ -formulae Γ ,

1. $\Gamma \subseteq LC \Gamma$
2. $\Gamma \subseteq DC \Gamma$

Definition 36.5 For any set Γ of Σ -sentences, the set of **models of** Γ is defined as the set

$$Mod \Gamma = \{A \mid A \Vdash \phi, \text{ for every } \phi \in \Gamma\}$$

We then have the following lemma which connects up the various sets that we have defined above.

Lemma 36.6

$$Th (Mod \Gamma) = LC \Gamma$$

Proof: For any $\phi \in \text{Th}(\text{Mod } \Gamma)$, we have $\text{Mod } \Gamma \Vdash \phi$. Hence for any $\mathbf{A} \in \text{Mod } \Gamma$ we have $\mathbf{A} \Vdash \phi$. If $\phi \in \Gamma \subseteq \text{LC } \Gamma$ there is nothing to prove. Suppose $\phi \notin \Gamma$, then since ϕ is true in every model of Γ , we have $\Gamma \models \phi$ and hence $\phi \in \text{LC } \Gamma$.

This proves that $\text{Th}(\text{Mod } \Gamma) \subseteq \text{LC } \Gamma$. The second part of the containment may be proven using the same argument backwards. QED

Hence it suffices to specify a (finite or infinite) set of first-order sentences in order to capture the models (which may well be a figment of our imagination) of interest to us. The first-order theory of these models would be the set of theorems that can be deduced from these axioms under a sound and complete deductive system. This first-order theory will in turn capture all the logical consequences of the set Γ of axioms.

Very often we may be interested in only a single structure \mathbf{A} and we may define axioms that we consider are fundamental properties of \mathbf{A} . However it may turn out that there may be other structures (with the same signature) that we never dreamed of, which satisfy the same set of axioms under a suitable interpretation of the signature. These other structures need not be isomorphic to the structure \mathbf{A} . However they would also satisfy all properties that are logical consequences of the axioms too. What we envisaged as a first-order theory of \mathbf{A} might very well turn out to be a first order theory of a class of structures \mathfrak{A} of which \mathbf{A} is just one member.

On the other hand if the set Γ is a set of axioms which lead to a contradiction, there would be no models at all. In this case every first-order sentence over the signature would be a logical consequence of the axioms and the set of theorems would include every first-order sentence too.

(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 21.3.12) 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

$$\begin{aligned}\Sigma &= \{\cdot; \langle\} \\ \phi_{<-irreflexivity} &\stackrel{df}{=} \forall x[\neg(x < x)] \\ \phi_{<-transitivity} &\stackrel{df}{=} \forall x, y, z[((x < y) \wedge (y < z)) \rightarrow (x < z)] \\ \phi_{<-trichotomy} &\stackrel{df}{=} \forall x, y[(x < y) \vee (x = y) \vee (y < x)] \\ \Phi_{ILO} &\stackrel{df}{=} \{\phi_{<-irreflexivity}, \phi_{<-transitivity}, \\ &\quad \phi_{<-trichotomy}\}\end{aligned}$$

(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

$$\begin{aligned}\Sigma_S &= \{0 : \rightarrow s, \ x^{+1} : s \rightarrow s; = : s^2\} \quad \textbf{P1, P2} \\ \phi_{0-not successor} &\stackrel{df}{=} \forall x[\neg(x^{+1} = 0)] \quad \textbf{P3} \\ \phi_{+1-injective} &\stackrel{df}{=} \forall x \forall y[x^{+1} = y^{+1} \rightarrow x = y] \quad \textbf{P4} \\ \phi_{\neq 0 \rightarrow successor} &\stackrel{df}{=} \forall y[\neg(y = 0) \rightarrow \exists x[y = x^{+1}]] \\ \phi_{(+1)^n-distinct} &\stackrel{df}{=} \forall x[\neg(x^{(+1)^n} = x)] \\ \Phi_{(+1)^\infty-distinct} &\stackrel{df}{=} \{\phi_{(+1)^n-distinct} \mid n > 0\}\end{aligned}$$

$$\begin{aligned}\Phi_S &\stackrel{df}{=} \{\phi_{0-not successor}, \phi_{+1-injective}, \phi_{\neq 0 \rightarrow successor}\} \\ &\cup \Phi_{(+1)^\infty-distinct}\end{aligned}$$

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*.

The Principle of Mathematical Induction

The principle of induction is really an inference rule (sometimes expressed as an axiom schema)

$$\text{Ind1. } \frac{\{0/x\}X, \forall x[X \rightarrow \{x^{+1}/x\}X]}{\forall x[X]}$$

where

- $\{0/x\}X$ is the basis of the induction,
- $\forall x[X \rightarrow \{x^{+1}/x\}X]$ is the induction step and
- X the antecedent in the induction step, is the induction hypothesis

Note.

- Notice that we have used syntactic substitutions to express the rule within our notation. But this rule is exactly the first-order rendering of the **Principle of Mathematical Induction – Version 1**.
- X in rule **Ind1** is a meta-variable on first-order predicates in **Peano arithmetic**. Rule **Ind1** by its very syntactic structure therefore is restricted to only the first-order properties of numbers whereas the principle **PMI1'** (and each of its equivalents) holds generally for properties of any order.

First-order Arithmetic

In addition to **Peano's postulates** first order arithmetic contains the addition and multiplication operations, which are axiomatized as follows. The axioms clearly reflect the definitions by induction of these operations.

$$\begin{aligned}\Sigma_A &= \Sigma_S \cup \{+, . : s^2 \rightarrow s\} \\ \phi_{+0} &\stackrel{df}{=} \forall x[x + 0 = x] \\ \phi_{++1} &\stackrel{df}{=} \forall x \forall y [x + (y^{+1}) = (x + y)^{+1}] \\ \phi_{.0} &\stackrel{df}{=} \forall x[x.0 = 0] \\ \phi_{.+1} &\stackrel{df}{=} \forall x \forall y [x.(y^{+1}) = (x.y) + x]\end{aligned}$$

Extending First-order Arithmetic

The signature Σ_A may be extended by other operations and relations.

Linear order. Define the (infix) $< : s^2$ as

$$x < y \stackrel{df}{=} \exists z[\neg(z = 0) \wedge x + z = y]$$

Subtraction. Define the (infix) partial function $- : s^2 \rightarrow s$ as one satisfying the axiom

$$\forall x, y, z[(z = y - x) \leftrightarrow (x + z = y)]$$

Exercise 36.1

1. Express the associativity properties of addition and multiplication as first-order formulae.
2. Prove the associativity properties for addition and multiplication in First-order arithmetic from the non-logical axioms and the axioms of $\mathbb{P}\mathbb{C} =$.
3. Express the commutativity of addition and multiplication operations in First-order arithmetic.
4. Prove the commutativity properties of addition and multiplication.
5. Prove that $\forall x, y, z [(z = y - x) \rightarrow \neg(z = 0) \rightarrow (x < y) \wedge (z < y)]$
6. Express the quotient and remainder operations on the naturals as axioms in First order arithmetic.

37: Towards Logic Programming

Computer science is no more about computers than astronomy is about telescopes.

Edsger W. Dijkstra

1. Reversing the Arrow
2. Arrow Reversal
3. Horn Clauses
4. Program or Rule Clause
5. Goal clauses
6. Logic Programs
7. Sorting in Logic
8. Prolog: Sort
9. Prolog: Merge Sort
10. Prolog: Quick Sort
11. Prolog: SEND+MORE=MONEY
12. 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

Arrow Reversal

$$\begin{aligned} C &\Leftrightarrow \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \vee (\bigvee_{1 \leq j \leq n} \neg \nu_j)] \\ &\Leftrightarrow \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \vee \neg(\bigwedge_{1 \leq j \leq n} \nu_j)] \\ &\Leftrightarrow \vec{\forall}[(\bigwedge_{1 \leq j \leq n} \nu_j) \rightarrow (\bigvee_{1 \leq i \leq p} \pi_i)] \\ &\equiv \vec{\forall}[(\bigvee_{1 \leq i \leq p} \pi_i) \leftarrow (\bigwedge_{1 \leq j \leq n} \nu_j)] \\ &\stackrel{df}{=} \pi_1, \dots, \pi_p \leftarrow \nu_1, \dots, \nu_n \end{aligned}$$

Horn Clauses

Definition 37.1 *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.*

Program or Rule Clause

$$\begin{aligned} P &\stackrel{df}{=} \pi \leftarrow \nu_1, \dots, \nu_n \\ &\equiv \vec{\forall}[\pi \vee (\bigvee_{1 \leq j \leq n} \neg \nu_j)] \\ &\equiv \vec{\forall}[\pi \vee \neg(\bigwedge_{1 \leq j \leq n} \nu_j)] \end{aligned}$$

and is read as “ π if ν_1 and ν_2 and \dots and ν_n ”.

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 37.2 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

$sort(x, y)$	$\leftarrow perm(x, y), ordered(y)$
$ordered(nil)$	\leftarrow
$ordered(x.nil)$	\leftarrow
$ordered(x.y.z)$	$\leftarrow lesseq(x, y), ordered(y.z)$
$lesseq(x, x)$	\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: Sort

```
isort(X, Y) :- permutation(X, Y),  
            ordered(Y).
```

```
ordered([]).
```

```
ordered(H.[]).
```

```
ordered(F.S.T) :- lesseq(F, S),  
                ordered(S.T).
```

```
lesseq(F, S) :- F=S.
```

```
lesseq(F, S) :- F<S.
```

```
permutation([], []).
```

```
permutation(H.T, F.R) :- delete(F, H.T, Z),  
                      permutation(Z, R).
```

```
delete(H, H.T, T).
```

```
delete(X, H.T, H.U) :- delete(X, T, U).
```

```
/* isort([2,8,-1,10,4,2], Y). */
```

Prolog: Merge Sort

```
mergeSort([], []).
mergeSort([H|T], [H|T]).
mergeSort([F,S,T], [S1|T1]) :- split([F,S,T], [S1|T1]),
                                mergeSort([S1|T1], [S2|T2]),
                                mergeSort([T2|T], [T3|T4]),
                                merge([S2|T2], [T3|T4], [S1|T1]).  
split([], [], []).
split([H|T], [H|T], []).
split([F,S,T], [F1|S1|T1], [S2|T2]) :- split([S,T], [S1|T1]),
                                         split([T], [T1]),
                                         split([F|S1], [F1|S2], [S1|T1]).  
merge([], L, L).
merge(L, [], L).
merge([F|B], [H|T], [F|H|T]) :- F < H, merge(B, [H|T], T).
merge([F|B], [H|T], [H|F|T]) :- H < F, merge([F|B], T, [H|T]).
```

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)))))) */
```

Example 37.3 In this example we give a prolog implementation of double-ended queues of integers using constructors.

```
/* 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
*/
```

Prolog: Abstract Interpreter

Algorithm 3 A simple abstract interpreter for Prolog

Require: A Prolog program P and ground goal G

Ensure: *yes* if $P \vdash G$ else *no*

```
1: resolvent := { $G$ }
2: while ~ empty(resolvent) do
3:   Choose goal  $A$  from resolvent
4:   Choose a ground instance of some clause  $A' \leftarrow B_1, \dots, B_k$  from  $P$  such that  $A \equiv A'$ 
5:   if  $A'$  does not exist then
6:     exit loop
7:   end if
8:   resolvent := (resolvent - { $A$ })  $\cup$  { $B_1, \dots, B_k$ }
9: end while
10: if empty(resolvent) then
11:   return yes
12: else
13:   return no
14: end if
```

HOME PAGE

◀◀

◀

▶

▶▶

LCS November 4, 2018

Go BACK

FULL SCREEN

CLOSE

900 OF 1040

QUIT

38: Verification of Imperative Programs

Program testing can be used to show the presence of bugs, but never to show their absence!

Edsger W. Dijkstra

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

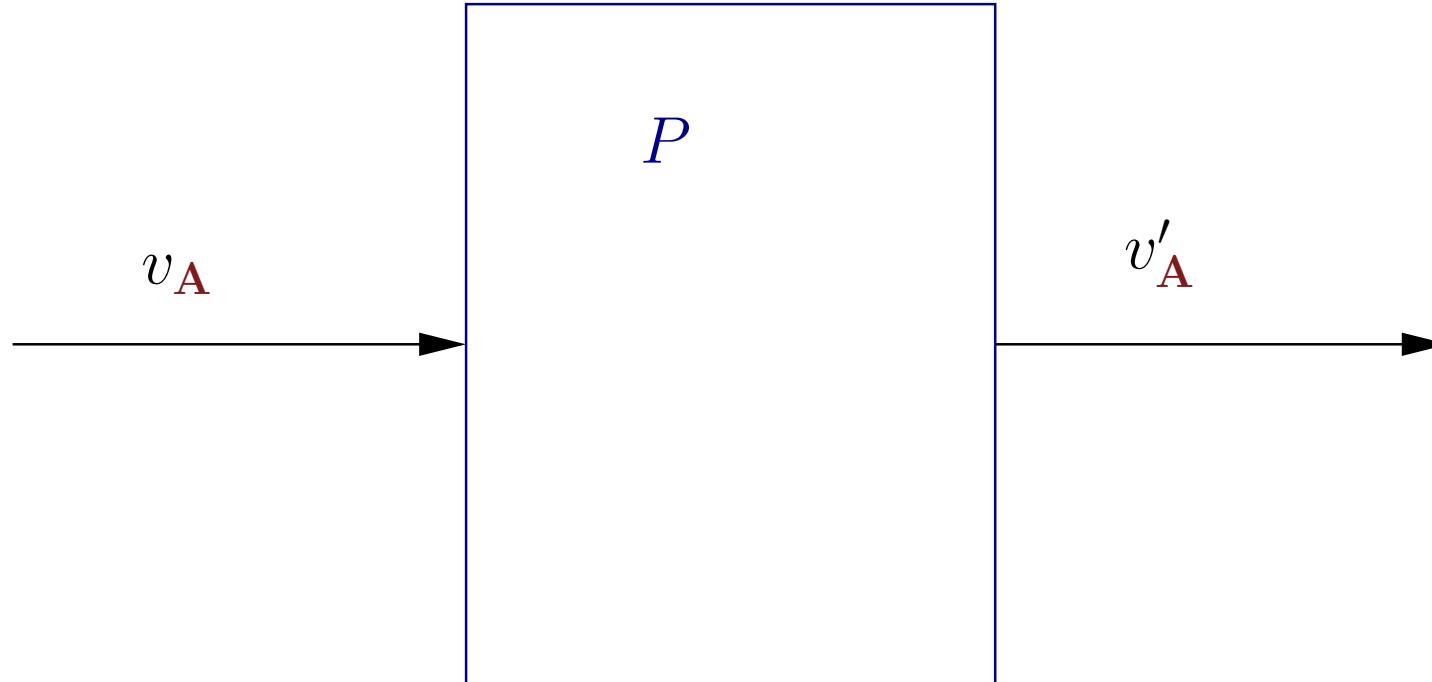
Let Σ be any signature. Then the programming language $\mathcal{WH}(\Sigma)$ is defined by the following BNF.

$$\begin{array}{ll} P, Q ::= \epsilon & (\textit{Skip}) \\ | \quad x := t & (\textit{Assignment}) \\ | \quad P; Q & (\textit{Composition}) \\ | \quad [P] & (\textit{Block}) \\ | \quad \chi?P : Q & (\textit{Conditional}) \\ | \quad \{\chi?P\} & (\textit{While}) \end{array}$$

where $\chi \in \mathcal{QF}_1(\Sigma)$.

Programs As State Transformers

A program is merely a *state transformer* which takes a valuation (also called a **state**) $v_{\mathbf{A}}$ of variables and yields another valuation $v'_{\mathbf{A}}$.



The Semantics of WHILE

Let \mathbf{A} be a Σ -algebra. Let $V_{\mathbf{A}} = \{v_{\mathbf{A}} \mid v_{\mathbf{A}} : V \rightarrow |\mathbf{A}|\}$ be the set of all valuations. The meaning of a program P is given by $\mathcal{M}_{\mathbf{A}}[P] : V_{\mathbf{A}} \rightarrow V_{\mathbf{A}}$

$$\begin{aligned}\mathcal{M}_{\mathbf{A}}[\epsilon]_{v_{\mathbf{A}}} &\stackrel{df}{=} v_{\mathbf{A}} \\ \mathcal{M}_{\mathbf{A}}[x := t]_{v_{\mathbf{A}}} &\stackrel{df}{=} v_{\mathbf{A}}[x := \mathcal{V}_{\mathbf{A}}[t]_{v_{\mathbf{A}}}] \\ \mathcal{M}_{\mathbf{A}}[P]_{v_{\mathbf{A}}} &\stackrel{df}{=} \mathcal{M}_{\mathbf{A}}[P]_{v_{\mathbf{A}}} \\ \mathcal{M}_{\mathbf{A}}[P; Q]_{v_{\mathbf{A}}} &\stackrel{df}{=} (\mathcal{M}_{\mathbf{A}}[Q] \circ \mathcal{M}_{\mathbf{A}}[P])_{v_{\mathbf{A}}} \\ \mathcal{M}_{\mathbf{A}}[\chi?P : Q]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} \mathcal{M}_{\mathbf{A}}[P]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 1 \\ \mathcal{M}_{\mathbf{A}}[Q]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 0 \end{cases} \\ \mathcal{M}_{\mathbf{A}}[\{\chi?P\}]_{v_{\mathbf{A}}} &\stackrel{df}{=} \begin{cases} \mathcal{M}_{\mathbf{A}}[P; \{\chi?P\}]_{v_{\mathbf{A}}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 1 \\ v_{\mathbf{A}} & \text{if } \mathcal{T}_{\mathbf{A}}[\chi]_{v_{\mathbf{A}}} = 0 \end{cases}\end{aligned}$$

Notes:

1. Except for the *While* command, all other commands are well-defined.
2. Notice that the last construct viz the *While* command with its meaning defined in terms of the meaning of itself and the *Composition* command inherently makes the meaning function partial (definition 0.1).
3. The meaning of the *While* command corresponds to the usual meaning associated with a (possibly indefinite) “*loop-unrolling*”.
4. There is no guarantee that this definition is inductive. Hence any program employing this command is potentially undefined at least for certain valuations.
5. Hence the correctness of any program employing this construct needs to establish its “*well-definedness*” by proving an appropriate property of the initial state v_A and all subsequent states (obtained through *Composition* that the program may traverse.

Exercise 38.1

1. We may define an equivalence relation on programs in $\mathcal{WH}(\Sigma)$ as follows:

Definition 38.1 $P =_{WH} Q$ iff for each Σ -algebra \mathbf{A} and each valuation $v_{\mathbf{A}}$, $\mathcal{M}_{\mathbf{A}}[\![P]\!]_{v_{\mathbf{A}}} = \mathcal{M}_{\mathbf{A}}[\![Q]\!]_{v_{\mathbf{A}}}$

(a) For $t, u \in T(\Sigma)$, $t = u$ implies for all variables x , $x := t =_{WH} x := u$

(b) For $\chi, \theta \in \mathcal{QF}_1(\Sigma)$, $\chi \Leftrightarrow \theta$ implies $\chi?P : Q =_{WH} \theta?P : Q$

(c) Prove the following for all programs P, Q and R in $\mathcal{WH}(\Sigma)$.

i. $P; \epsilon =_{WH} P =_{WH} \epsilon; P$

ii. $P; [Q; R] =_{WH} [P; Q]; R$

iii. $\chi?P : Q =_{WH} \neg\chi?Q : P$

iv. $\{\chi?P\} =_{WH} \chi? [P; \{\chi?P\}] : \epsilon$

(d) Prove that $=_{WH}$ is a congruence relation on $\mathcal{WH}(\Sigma)$.

Programs on Sets of States

Since $\mathcal{M}_A[\![P]\!]$ for any program P could be undefined (it is **partial**) we may make it total by defining the function over sets of source states yielding target states. Of course, in the process, we lose the exact correspondence between input and output states.

$$\mathcal{M}'_A[\![P]\!] : \mathbb{Z}^{V_A} \longrightarrow \mathbb{Z}^{V_A}$$

such that for any set $S_A \subseteq V_A$

$$\mathcal{M}'_A[\![P]\!]_{S_A} \stackrel{df}{=} T_A \quad (60)$$

where

$$T_A = \{v'_A \mid v'_A = \mathcal{M}_A[\![P]\!]_{v_A}, v_A \in S_A\}$$

Notes:

- V_A could be uncountable since V is countable and $|A|$ could be countable. However, since a program P has only a finite number of variables, the set of tuples of values that represent a *state* of the program is at most countable if $|A|$ is countable.
- For any program P let ϕ be any predicate whose free variables are (some superset of) the variables of P . We may then regard the predicate ϕ as defining a property which relates the values

of the variables in the program. Indeed ϕ is a property of certain states of the program P . By abuse of notation we write $\{\phi\}$ to denote the set of all states that satisfy the predicate. In other words,

$$\{\phi\} = \{v_A \mid (A, v_A) \Vdash \phi\} \quad (61)$$

and we refer to $\{v_A \mid (A, v_A) \Vdash \phi\}$ as the **characteristic set** of ϕ .

- In general, even if V_A is countable there is no guarantee that every subset of V_A can be defined by a predicate. In fact, since the number of possible subsets of a countable set V_A is uncountable, there are only at most a countable number of them which can be characterised by predicates. However, every predicate ϕ whose free variables are drawn from only a finite superset of the variables of the program, does have a unique characteristic set $\{\phi\}$ given by equation (61).
- In particular if $\{\phi\} = S_A$ and $\{\psi\} = T_A$ in equation (60) then we may claim that P transforms the predicate ϕ into the predicate ψ and denote this fact by the identity.

$$\{\phi\}P = \{\psi\} \quad (62)$$

- More generally the identity (62) may be regarded as a relation that connects the program P with its **pre-condition** (ϕ) and its **post-condition** (ψ).

$$\{\phi\} P \{\psi\} \quad (63)$$

where ϕ is the **pre-condition**,

Program Specification

There are several different ways in which we could regard the relationships that exist between the three components of the triple (63).

State transformer. P transforms a state in $\{\phi\}$ into a state in $\{\psi\}$.

Predicate transformer. The triple (63) allows us to regard the program P as one that given an *input* state v_A satisfying the predicate ϕ may produce an *output* state v'_A that satisfies the predicate ψ . That is, P transforms the predicate ϕ into the predicate ψ . That is,

$$P : \mathcal{P}_1(\Sigma) \longrightarrow \mathcal{P}_1(\Sigma) \quad (64)$$

and $P(\{\phi\}) = \{\psi\}$.

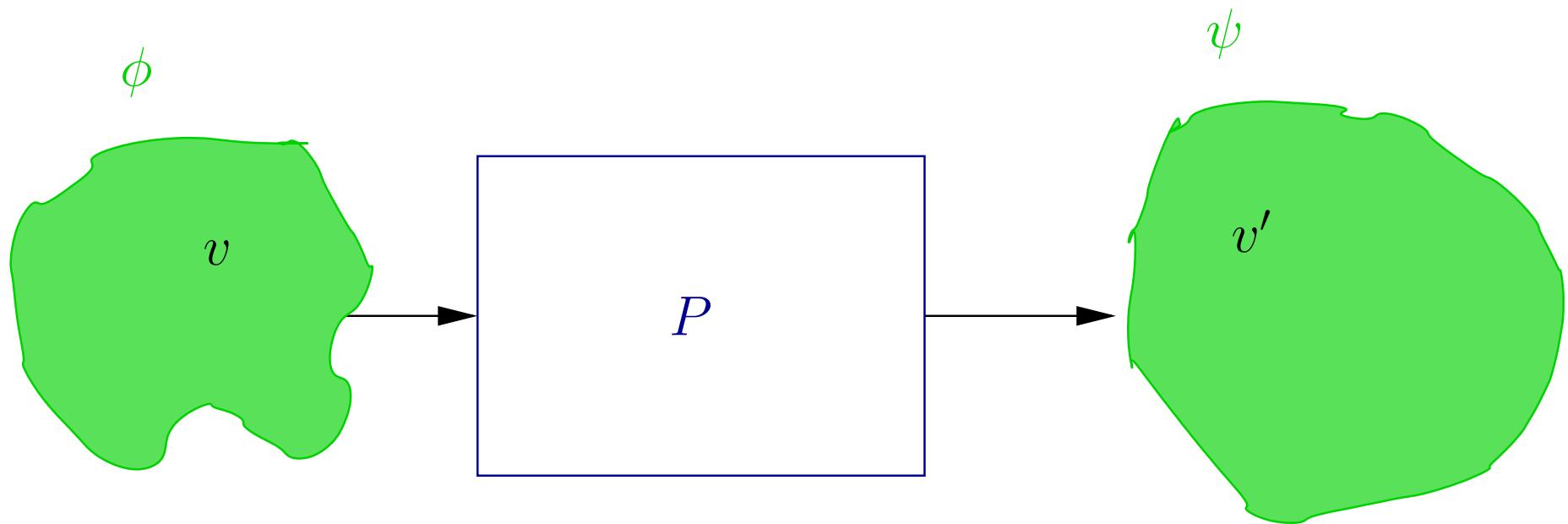
Specification and Implementation. We may also think of the pair $\langle \phi, \psi \rangle$ as a **specification** that requires a program P to transform the predicate ϕ into ψ . Any such program then is an **implementation** of the specification. We may also say that P **realises** the specification $\langle \phi, \psi \rangle$.

Correctness Assertion. The triple (63) may be regarded as a correctness assertion which states that P is correct with respect to the specification $\langle \phi, \psi \rangle$, and hence potentially requires a proof of

correctness. This also implies that one requires a collection of axioms and inference rules from which the triple $\{\phi\} \textcolor{blue}{P} \{\psi\}$ may be deduced.

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 38.2 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 38.3

- $\{\phi\} P \{\psi\}$ 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 $A \Vdash \{\phi\} P \{\psi\}$) if $(A, v_A) \Vdash \{\phi\} P \{\psi\}$ for every state v_A .

Total Correctness of Programs

Definition 38.4 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 38.5

- $\{\phi\} P \{\psi\}$ 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 $A \Vdash \{\phi\} P \{\psi\}$) if $(A, v_A) \Vdash \{\phi\} P \{\psi\}$ for every state v_A .

Examples: Factorial 1

Let $\Sigma \supseteq \mathbb{Z} \cup \{! : s \rightarrow s, +, -, * : s^2 \rightarrow s; =, > : s^2\}$.

Example 38.6 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. $\mathbf{Z} \Vdash \{x = x_0\} P_1 \{p = x_0!\}$
2. However $\mathbf{Z} \not\Vdash \{[x = x_0]\} P_1 \{[p = x_0!]\}$ since P_1 will not terminate for negative values of x .

Examples: Factorial 2

Example 38.7 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. $\mathbf{Z} \Vdash \{x = x_0 \geq 0\} P \{p = x_0!\}$
2. $\mathbf{Z} \Vdash \{\boxed{x = x_0 \geq 0}\} P \{\boxed{p = x_0!}\}$
3. A more “technically complete” specification is
 $\mathbf{Z} \Vdash \{\boxed{x = x_0}\} P \{\{(x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)\}\}$

Exercise 38.2

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?

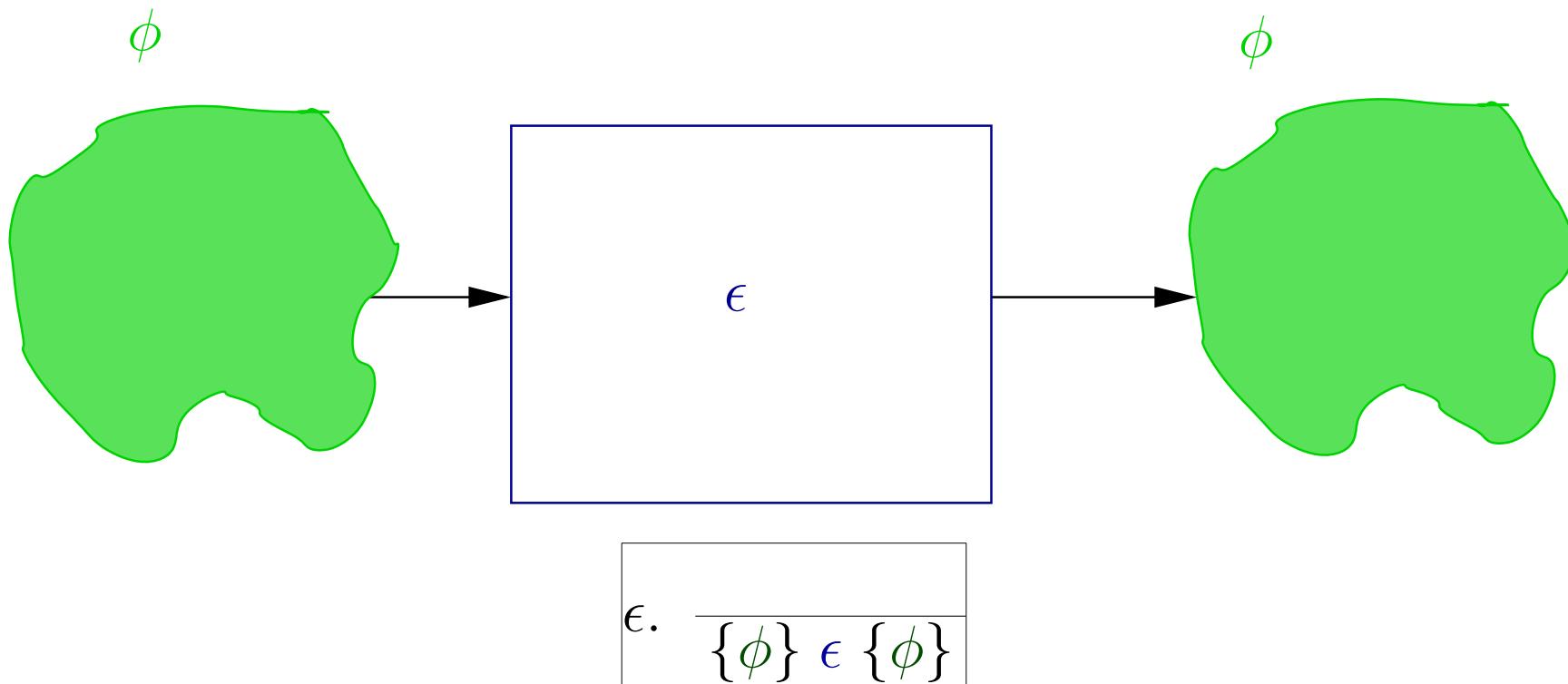
39: Verification of WHILE Programs

If debugging is the process of removing software bugs, then programming must be the process of putting them in.

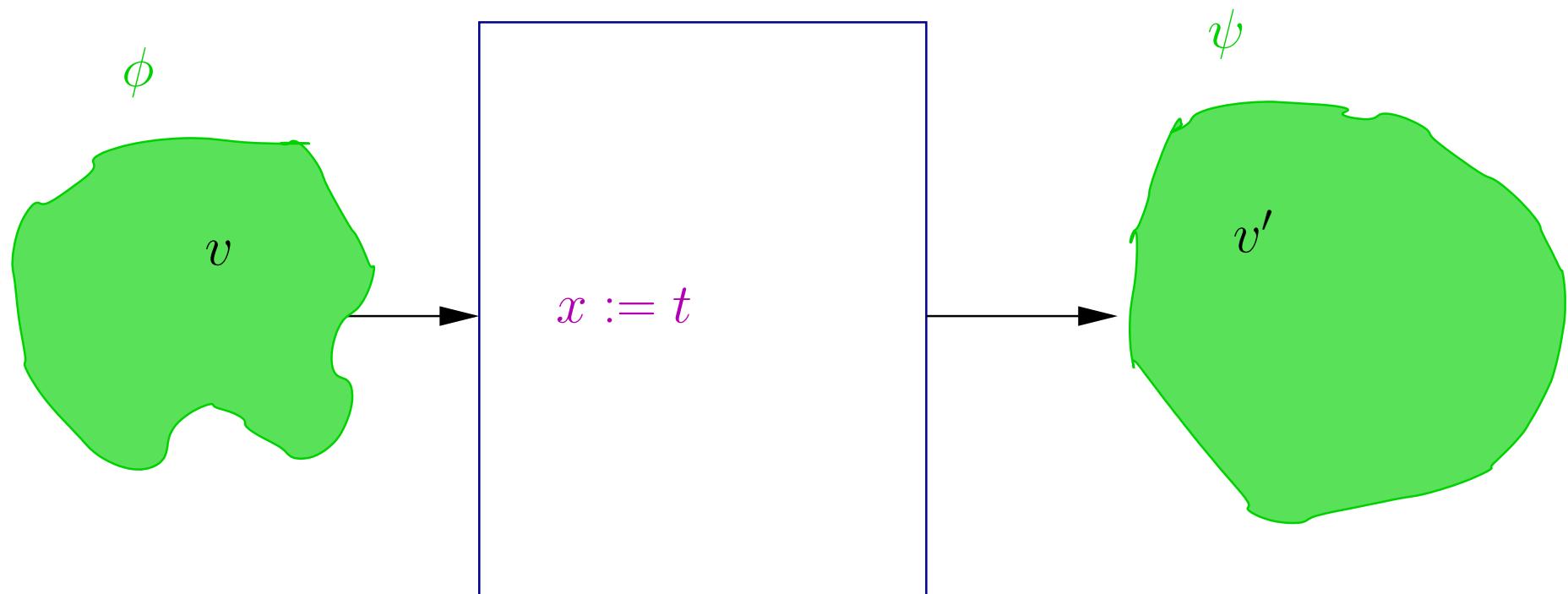
Edsger W. Dijkstra

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


$$:= \cdot \frac{\{\phi\} \ x := t \ \{\psi\}}{} \quad (\phi \equiv \{t/x\}\psi)$$

Swap Programs

Example 39.1 Swap1. Consider the following program which may be used to swap the values of a pair of variables x and y . It uses a temporary variable to “remember” one of the values which gets “over-written”.

$$\text{Swap1} \stackrel{df}{=} t := x; x := y; y := t$$

This program is not specific to any particular signature.

Proof of Swap1:0

We begin by specifying the precondition and the post-condition.

$\vdash \{x = x_0 \wedge y = y_0\}$

$t := x;$

$x := y;$

$y := t \quad \{x = y_0 \wedge y = x_0\}$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap1:1

We begin by specifying the precondition and the post-condition.

$\vdash \{x = x_0 \wedge y = y_0\}$

$t := x;$

$x := y; \quad \{x = y_0 \wedge t = x_0\}$

$y := t \quad \{x = y_0 \wedge y = x_0\}$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap1:2

We begin by specifying the precondition and the post-condition.

$$\begin{array}{ll} \vdash & \{x = x_0 \wedge y = y_0\} \\ t := x; & \{y = y_0 \wedge t = x_0\} \\ x := y; & \{x = y_0 \wedge t = x_0\} \\ y := t & \{x = y_0 \wedge y = x_0\} \end{array}$$

The **assignment rule** is most convenient when applied “backwards”.

Swap2

Example 39.2 swap2 However the following program specifically uses operations that are specific to a data type that allows for (operations akin to) addition and subtraction on the integers.

$$\text{Swap2} \stackrel{df}{=} u := u + v; v := u - v; u := u - v$$

Proof of Swap2:0

Again we specify the precondition and the postcondition. However we restrict our interpretation of the operations $+$ and $-$ to the corresponding operations on integers.

$\mathbf{Z} \Vdash \{u = u_0 \wedge v = v_0\}$

$u := u + v;$

$v := u - v;$

$u := u - v \quad \{u = v_0 \wedge v = u_0\}$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap2:1

Again we specify the precondition and the postcondition. However we restrict our interpretation of the operations $+$ and $-$ to the corresponding operations on integers.

$\mathbf{Z} \Vdash \{u = u_0 \wedge v = v_0\}$

$u := u + v;$

$v := u - v; \quad \{u - v = v_0 \wedge v = u_0\}$

$u := u - v \quad \{u = v_0 \wedge v = u_0\}$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap2:2

Again we specify the precondition and the postcondition. However we restrict our interpretation of the operations $+$ and $-$ to the corresponding operations on integers.

$$\begin{array}{ll} \mathbf{Z} \Vdash & \{u = u_0 \wedge v = v_0\} \\ u := u + v; & \{u - (u - v) = v_0 \wedge u - v = u_0\} \\ v := u - v; & \{u - v = v_0 \wedge v = u_0\} \\ u := u - v & \{u = v_0 \wedge v = u_0\} \end{array}$$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap2:3

Again we specify the precondition and the postcondition. However we restrict our interpretation of the operations $+$ and $-$ to the corresponding operations on integers.

$$\begin{array}{ll} \mathbf{Z} \Vdash & \{u = u_0 \wedge v = v_0\} \\ & = \{(u + v) - ((u + v) - v) = v_0 \wedge (u + v) - v = u_0\} \\ u := u + v; & \{u - (u - v) = v_0 \wedge u - v = u_0\} \\ v := u - v; & \{u - v = v_0 \wedge v = u_0\} \\ u := u - v & \{u = v_0 \wedge v = u_0\} \end{array}$$

The **assignment rule** is most convenient when applied “backwards”.

Proof of Swap2:4

Again we specify the precondition and the postcondition. However we restrict our interpretation of the operations $+$ and $-$ to the corresponding operations on integers.

$$\begin{aligned} \mathbf{Z} \Vdash & \quad \{u = u_0 \wedge v = v_0\} \\ &= \{v = v_0 \wedge u = u_0\} \\ &= \{(u + v) - ((u + v) - v) = v_0 \wedge (u + v) - v = u_0\} \\ u := u + v; & \quad \{u - (u - v) = v_0 \wedge u - v = u_0\} \\ v := u - v; & \quad \{u - v = v_0 \wedge v = u_0\} \\ u := u - v & \quad \{u = v_0 \wedge v = u_0\} \end{aligned}$$

The **assignment rule** is most convenient when applied “backwards”.

Swap1 vs. Swap2

- The proof of Swap1 (example 39.1) required no properties particular to any signature. Hence we have

$$\Vdash \{x = x_0 \wedge y = y_0\} Swap1 \{x = y_0 \wedge y = x_0\}$$

- The proof of Swap2 (example 39.2) clearly requires certain operations like $+$ and $-$ as part of the signature. These operations need to be total. Moreover

$$- \{v = v_0 \wedge u = u_0\} = \{(u + v) - ((u + v) - v) = v_0 \wedge (u + v) - v = u_0\}$$

would hold only if the two operations $+$ and $-$ provably satisfy properties such as

$$- (u + v) - v = u$$

$$- (u + v) - ((u + v) - v) = v$$

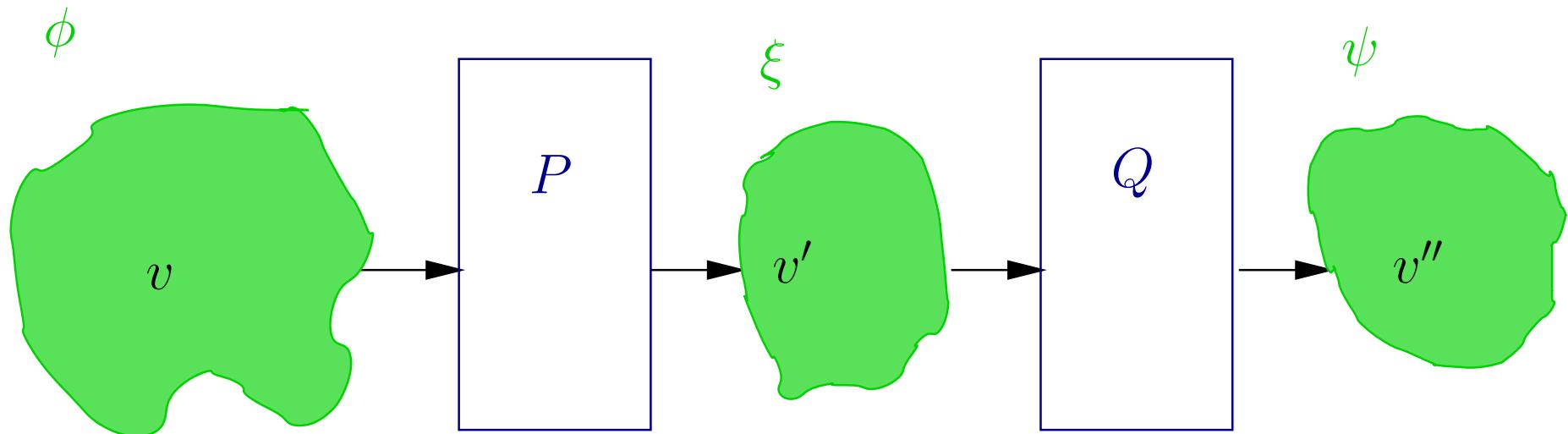
However the first equality $\{u = u_0 \wedge v = v_0\} = \{v = v_0 \wedge u = u_0\}$ is a purely logical one (commutativity of \wedge) and is independent of interpretation. Hence

$$\mathbf{Z} \Vdash \{x = x_0 \wedge y = y_0\} Swap2 \{x = y_0 \wedge y = x_0\}$$

But

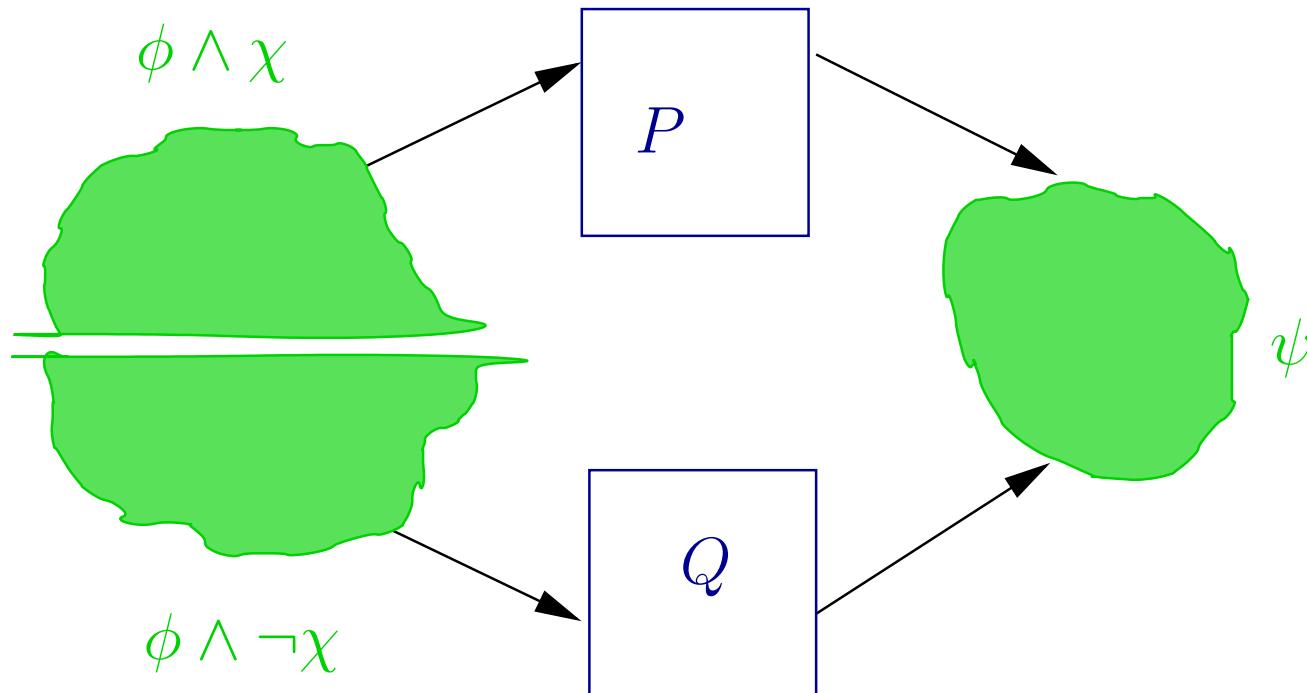
$$\nVdash \{x = x_0 \wedge y = y_0\} Swap2 \{x = y_0 \wedge y = x_0\}$$

Proof Rule: Composition



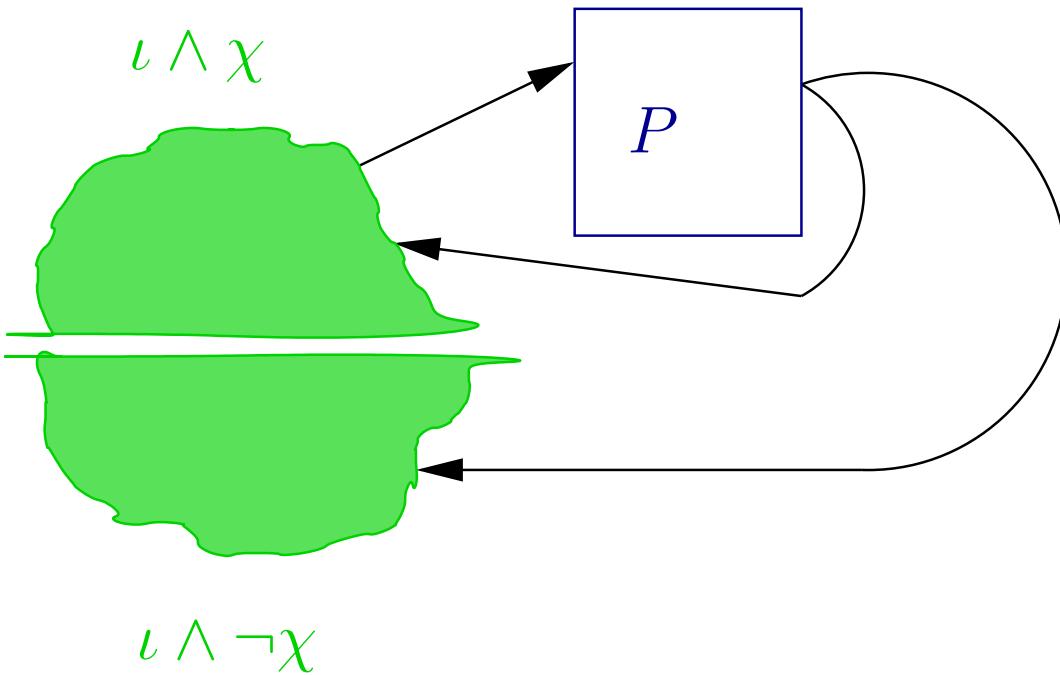
$$\frac{\begin{array}{c} \{\phi\} P \{\xi\} \\ ; . \quad \{\xi\} Q \{\psi\} \end{array}}{\{\phi\} P; Q \{\psi\}}$$

Proof Rule: The Conditional



$$\boxed{\frac{\begin{array}{c} \{\phi \wedge \chi\} P \{\psi\} \\ ? : . \quad \{\phi \wedge \neg\chi\} Q \{\psi\} \end{array}}{\{\phi\} \chi?P : Q \{\psi\}}}$$

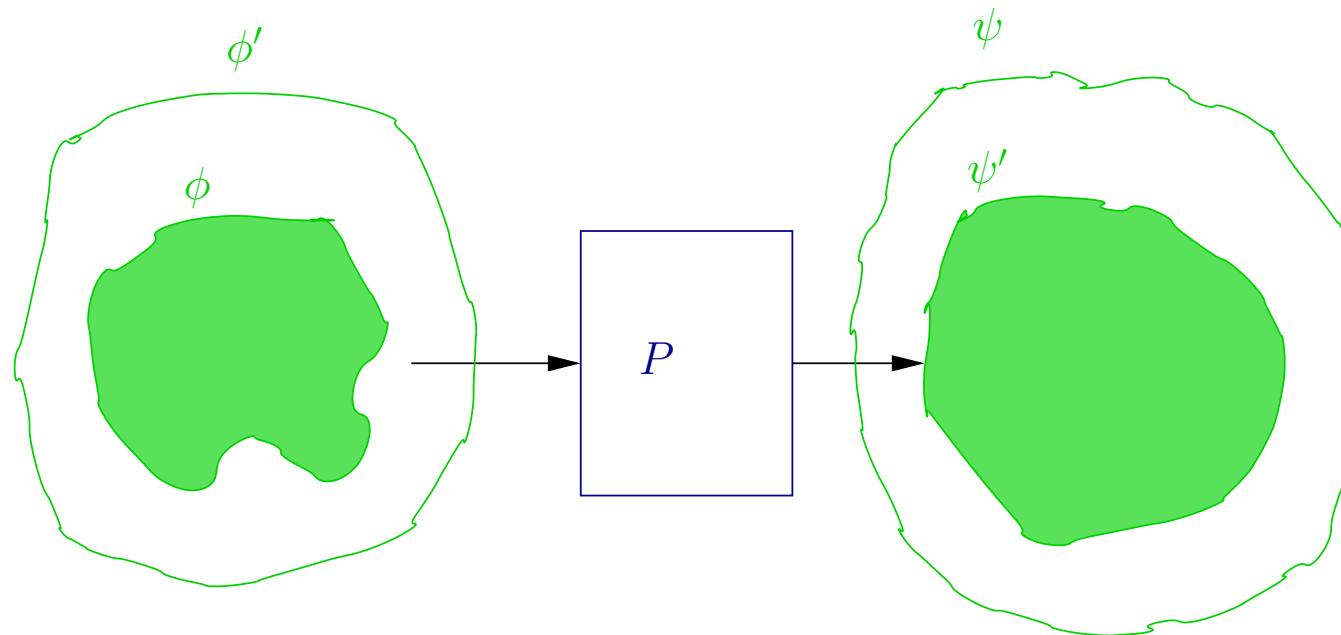
Proof Rule: The While Loop



$$\{\} \cdot \frac{\{\iota \wedge \chi\} \ P \ \{\iota\}}{\{\iota\} \ \{\chi?P\} \ \{\iota \wedge \neg\chi\}}$$

ι in the while rule is called the **loop invariant**.

The Consequence Rule



$$\Rightarrow \ . \quad \frac{\begin{array}{c} \{\phi\} \subseteq \{\phi'\} \\ \{\phi'\} P \{\psi'\} \\ \{\psi'\} \subseteq \{\psi\} \end{array}}{\{\phi\} P \{\psi\}}$$

Proof Rules for Partial Correctness

$$\epsilon. \quad \{\phi\} \in \{\phi\}$$

$$:= . \quad \{\{t/x\}\psi\} \ x := t \ \{\psi\}$$

$$[] . \quad \begin{array}{c} \{\phi\} \ P \ \{\psi\} \\ \{\phi\} [P] \ \{\psi\} \end{array}$$

$$\{\}. \quad \frac{\{\iota \wedge \chi\} \ P \ \{\iota\}}{\{\iota\} \ \{\chi?P\} \ \{\iota \wedge \neg \chi\}}$$

$$\vdots . \quad \frac{\{\phi\} \ P \ \{\xi\} \quad \{\xi\} \ Q \ \{\psi\}}{\{\phi\} \ P; Q \ \{\psi\}}$$

$$? \ddots \quad \frac{\{\phi \wedge \chi\} \ P \ \{\psi\} \quad \{\phi \wedge \neg \chi\} \ Q \ \{\psi\}}{\{\phi\} \ \chi?P : Q \ \{\psi\}}$$

$$\Rightarrow . \quad \frac{\begin{array}{c} \{\phi\} \subseteq \{\phi'\} \\ \{\phi'\} \ P \ \{\psi'\} \\ \{\psi'\} \subseteq \{\psi\} \end{array}}{\{\phi\} \ P \ \{\psi\}}$$

ι in the while rule is called the **loop invariant**.

Example: Factorial 1

$$\begin{aligned} & \{x = x_0\} \\ p := 1; & \subseteq \{x = x_0 \wedge p = 1\} \\ & \subseteq \{(x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!)\} \\ \{\neg(x = 0)? & \subseteq \{\phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!) \equiv \{\iota\} \\ [p := p * x; & \subseteq \{\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \neg(x = 0)\} \\ x := x - 1] & \subseteq \{\phi_0 \wedge (x > 0) \wedge (p * x! = x_0!)\} \\ & \subseteq \{\phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \\ & \quad \{\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!)\} \\ & \subseteq \{\iota\} \\ \} & \subseteq \{\iota \wedge (x = 0)\} \\ & \subseteq \{\phi_0 \wedge (x = 0) \wedge (p * x! = x_0!)\} \\ & \subseteq \{(x_0 \geq 0 \rightarrow p = x_0!)\} \end{aligned}$$

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

$$\epsilon! \quad \frac{}{\{\phi\} \epsilon \{\phi\}}$$

$$:=! \quad \frac{}{\{\{t/x\}\psi\}} x := t \{\psi\}$$

$$[]! \quad \frac{\{\phi\} P \{\phi\}}{\{\phi\} [P] \{\phi\}}$$

$$; ! \quad \frac{\{\phi\} P \{\xi\} \quad \{\xi\} Q \{\psi\}}{\{\phi\} P; Q \{\psi\}}$$

$$? : ! \quad \frac{\{\phi \wedge \chi\} P \{\psi\} \quad \{\phi \wedge \neg \chi\} Q \{\psi\}}{\{\phi\} \chi?P : Q \{\psi\}}$$

$$\Rightarrow ! \quad \frac{\{\phi\} \subseteq \{\phi'\} \quad \{\phi'\} P \{\psi'\} \quad \{\psi'\} \subseteq \{\psi\}}{\{\phi\} P \{\psi\}}$$

$$\{\}! \quad \frac{\{\iota \wedge \chi \wedge \beta(\vec{v}) = b_0 > 0\} P \{\iota \wedge \beta(\vec{v}) < b_0\}}{\{\iota\} \{\chi?P\} \{\iota \wedge \neg \chi\}}$$

where

- β is a **bound function** and
- ι is the **loop invariant**.

c.f. Proof Rules for Partial Correctness

Example: Factorial 2

$$\begin{aligned} & \{\ x = x_0 \ \} \\ p := 1; & \subseteq \{\ x = x_0 \wedge p = 1 \ \} \\ & \subseteq \{\ (x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!) \wedge (x_0 < 0 \rightarrow p = 1) \ \} \\ \{x > 0? & \subseteq \{\ \phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!) \ \} = \{\ \iota \ \} \\ [p := p * x; & \{ \phi_0 \wedge (x > 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0 \ \} \\ x := x - 1] & \{ \phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \\ & \quad \beta(x, p) = x = \beta_0 > 0 \ \} \\ \} & \{ \phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x < \beta_0 \ \} \\ & \subseteq \{\ \iota \ \} \\ & \subseteq \{\ \phi_0 \wedge (x = 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = 0 \ \} \\ & \subseteq \{\ (x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1) \ \} = \{\ \iota \ \} \end{aligned}$$

where $\phi_0 = (x_0 \geq 0) \wedge (x_0 < 0 \rightarrow p = 1)$

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.

$$\begin{aligned} & \{\{x = x_0\}\} = \{\{\phi_0\}\} \\ \subseteq & \{\{x = x_0 \wedge 1 = 1\}\} = \{\{\{1/p\}\phi_1\}\} \\ p := 1; & \{\{x = x_0 \wedge p = 1\}\} = \{\{\phi_1\}\} \\ \subseteq & \{\{(x_0 = x) \wedge (x \geq 0 \rightarrow p * x! = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)\}\} \\ \subseteq & \{\{\phi_0 \wedge (x \geq 0 \rightarrow p * x! = x_0!)\}\} = \{\{\iota\}\} \\ \{x > 0? & \{\{\phi_0 \wedge (x > 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0\}\} = \{\{\phi'_1\}\} \\ \subseteq & \{\{p * x / p\}\psi_1\} \\ [p := p * x; & \{\{\phi_0 \wedge (x > 0) \wedge (p * (x - 1)! = x_0!) \wedge \beta(x, p) = x = \beta_0 > 0\}\} = \{\{\psi_1\}\} \\ \subseteq & \{\{x - 1 / x\}\psi_2\} \\ x := x - 1 & \{\{\phi_0 \wedge (x \geq 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x < \beta_0 > 0\}\} = \{\{\psi_2\}\} \\] & [\psi_2] \\ \subseteq & \{\{\iota\}\} \\ \} & \{\{\iota \wedge \neg(x > 0) \wedge \beta(x, p) = x = 0\}\} \\ \subseteq & \{\{\phi_0 \wedge (x = 0) \wedge (p * x! = x_0!) \wedge \beta(x, p) = x = 0\}\} \\ \subseteq & \{\{(x_0 \geq 0 \rightarrow p = x_0!) \wedge (x_0 < 0 \rightarrow p = 1)\}\} = \{\{\iota\}\} \end{aligned}$$

An Open Problem: Collatz

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

$$\mathbf{Z} \Vdash \{\textcolor{violet}{x} > 0\} \textit{Collatz} \{\textcolor{violet}{x} = 1\}$$

where

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

Example 39.3 For any two integers $n, k > 1$ write an $O(\log_2 n)$ program along with its proof of correctness in Hoare Logic, which computes the number $\lfloor \sqrt[k]{n} \rfloor$. You may assume that there exists a predefined binary operator ' k ', which computes m^k for any integers m and $k > 0$.

We observe that $m = \lfloor \sqrt[k]{n} \rfloor$ really stands for the predicate $m^k \leq n < (m + 1)^k$. We use this as the post-condition. Effectively it requires determining the closed interval $[m, m + 1]$ within which the k -th root of n lies.

The invariant is simply obtained from the post-condition by weakening it, which effectively means extending the post-condition to a larger interval $[l, u]$ and bounding the values of l and u to ensure that the interval does not expand arbitrarily. Hence we have the constraint $1 \leq l < u \leq n$ from which the bound function is obtained as either the length of the interval $[l, u]$ which is bounded below by 2 or equivalently the measure $\beta = u - l$ which will be bounded below by 1.

The algorithmic strategy employs binary search starting with the initial interval $[1, n]$ which satisfies the invariant property trivially and proceeds to halve the interval in each iteration.

	$\{[n > 1 \wedge k > 1]\}$
$l := 1; u := n;$	$\{[u = n > 1 = l \wedge k > 1]\}$
	$\subseteq \{[\iota \equiv (1 \leq l < u \leq n) \wedge (l^k \leq n < u^k)]\} \cap \{[\beta : u - l \geq 1]\}$
$\{u > l + 1?\}$	$\{[\iota \wedge u > l + 1]\}$
	$\subseteq \{[(1 \leq l < l + 1 < u \leq n) \wedge (l^k \leq n < u^k)]\} \cap \{[\beta_0 : u - l > 1]\}$
$[m := (l + u) \text{ div } 2;$	$\{[(1 \leq l < l + 1 < u \leq n) \wedge (l^k \leq n < u^k) \wedge l \leq m < u]\}$
$m^k > n?$	$\{[(1 \leq l < l + 1 < u \leq n) \wedge (l^k \leq n < m^k < u^k) \wedge l \leq m < u]\}$
	$\subseteq \{[(1 \leq l < m \leq n) \wedge (l^k \leq n < m^k)]\}$
$u := m$	$\{[(1 \leq l < u \leq n) \wedge (l^k \leq n < u^k) \equiv \iota]\} \cap \{[\beta : u - l = m - l < \beta_0]\}$
:	$\{[\iota \wedge m^k \leq n]\}$
	$\subseteq \{[(1 \leq m < u \leq n) \wedge (m^k \leq n < u^k)]\}$
$l := m$	$\{[(1 \leq l < u \leq n) \wedge (l^k \leq n < u^k) \equiv \iota]\} \cap \{[\beta : u - l = u - m < \beta_0]\}$
]	$\{[\iota]\} \cap \{[\beta < \beta_0]\}$
}	$\{[\iota \wedge u \leq l + 1]\}$
	$\subseteq \{[(1 \leq l < u \leq n) \wedge u \leq l + 1 \wedge (l^k \leq n < u^k)]\} \cap \{[\beta : u - l = 1]\}$
	$\subseteq \{[(1 \leq l < u = l + 1 \leq n) \wedge (l^k \leq n < u^k)]\} \cap \{[\beta : u - l = 1]\}$
	$\subseteq \{[(1 \leq l < u = l + 1 \leq n) \wedge (l^k \leq n < (l + 1)^k)]\} \cap \{[\beta : u - l = 1]\}$
	$\subseteq \{[(1 \leq l < n) \wedge l = \lfloor \sqrt[k]{n} \rfloor]\} \cap \{[\beta : u - l = 1]\}$

40: Many-sorted FOL

1. Sortedness
2. Many-Sorted Logic: Symbols
3. Many-Sorted Signatures
4. Many-Sorted Signature: Terms
5. Many-Sorted Predicate Logic
6. Reductions: Guardedness
7. Reductions: Bounded Quantification

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 40.1 *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 40.2 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 40.3 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
- Second-Order logic may simply be considered a 2-sorted logic with predicates parameterised over both individuals and sets of individuals.

Reductions: Guardedness

- An S -sorted Predicate Logic may be reduced to a 1-sorted Predicate logic by introducing a fresh set of unary postfix predicates $: s_i$ (one for each sort s_i)
- Every quantified formula of the form $\forall_i x_i[\phi]$ is replaced by the 1-sorted formula $\forall x[(x : s_i) \rightarrow \phi]$ and recursively for each quantifier.
- Every quantified formula of the form $\exists_i x_i[\phi]$ is replaced by the 1-sorted formula $\exists x[(x : s_i) \wedge \phi]$ and recursively for each quantifier,

Reductions: Bounded Quantification

Quantified formulae are often abbreviated as follows:

$$\begin{array}{ll} \forall x : s_i[\phi] & \text{for } \forall x[(x : s_i) \rightarrow \phi] \\ \exists x : s_i[\phi] & \text{for } \exists x[(x : s_i) \wedge \phi] \end{array}$$

Notice that

$$\begin{array}{l} \neg \forall x : s_i[\phi] \Leftrightarrow \exists x : s_i[\neg \phi] \\ \neg \exists x : s_i[\phi] \Leftrightarrow \forall x : s_i[\neg \phi] \end{array}$$

HOME PAGE

◀◀

◀

▶

▶▶

LCS November 4, 2018

Go BACK

FULL SCREEN

CLOSE

961 OF 1040

QUIT

41: Abstract Data Types

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

C. A. R. Hoare

1. Pairs
2. The Array Structure: Signature
3. The Array Structure: Axioms
4. The List Structure: Signature
5. The List Structure Axioms: 1
6. The List Structure Axioms: 2
7. The List Structure: Induction
8. The List Structure: Acyclicity 1
9. The List Structure: Acyclicity 2
10. The Stack Structure: Signature
11. The Stack Structure Axioms: 1
12. The Stack Structure Axioms: 2
13. The Stack Structure: Induction
14. The Queue Structure: Signature
15. The Queue Structure Axioms: 1
16. The Queue Structure Axioms: 2
17. The Queue Structure: Induction
18. Binary Trees: Signature
19. The Binary Tree Axioms: 1

- 20. The Binary Tree Axioms: 2
- 21. The Binary Tree Axioms: 3
- 22. The Binary Tree Induction Rule

41.1. Introduction: Compound Data Structures

We have so far considered only the first-order theory of elementary data such as numbers and characters. In each case the data have a predefined equality relation. In this section we consider the first-order theory (with equality) of compound data structures encountered in computer science books on programming, data structures and algorithms.

Compound data structures such as strings, lists and trees are built using some more elementary data such as numbers and characters. However it is also possible to construct compound data structures over other compound data structures. For example, we could construct lists of trees of numbers, lists of strings, or even trees over lists of strings, trees over trees, lists of lists etc. There is literally no limit to such complex constructions. But in each case we would require to consider at least two sorts. Hence lists of numbers would take the numbers as a parameter. Similarly lists of strings would take the strings over an alphabet as a parameter.

Many compound data structures may be built on more than one more elementary data structure. This is especially true of many user-defined data structures such as tuples, records and variant records. In each of these cases there exists a first-order theory (with equality) which depends upon the nature of construction of these structures. The equality relations on the individual components induces a

naturally defined equality relation (component-wise equality) on the compound structure too.

Since record structures are specifically user-defined and not particularly interesting structures, we will not describe them in great detail. They are not interesting, mainly because they allow for just constructors to build them and deconstructors (or projection functions) which allow us to extract the individual components. The following example of tuples illustrates what we mean by the above.

Example 41.1 Let \mathbb{A} and \mathbb{B} be carrier sets of \mathbf{A} and \mathbf{B} , whose signatures are Σ_1 and Σ_2 respectively. Further let $=_1$ and $=_2$ be respective equality relations in Σ_1 and Σ_2 respectively. $\mathbb{A} \times \mathbb{B}$ is the carrier set of the cartesian product of the two sets i.e.

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

Think of $(,)$ as a “mixfix” constructor which constructs ordered pairs of elements belonging to the two sets respectively. We define the projection functions $\#1$ and $\#2$, to extract the components of any ordered pair $(a, b) \in \mathbb{A} \times \mathbb{B}$. That is, $\#1(a, b) = a$ and $\#2(a, b) = b$. These projection functions are the deconstructors of this pairing data structure. If s_1 denotes the sort of \mathbf{A} and s_2 denotes the sort of \mathbf{B} then we have the structure

$$\text{Pairs}[\mathbf{A}, \mathbf{B}] = \langle \mathbb{A} \times \mathbb{B}; (,), \#1, \#2; =_{12} \rangle$$

where the signature of $\text{Pairs}[\mathbf{A}, \mathbf{B}]$ is given by

$$\begin{aligned} (,) &: s_1, s_2 \longrightarrow (s_1, s_2) \\ \#1 &: (s_1, s_2) \longrightarrow s_1 \\ \#2 &: (s_1, s_2) \longrightarrow s_1 \\ =_{12} &: (s_1, s_2)^2 \end{aligned}$$

where (s_1, s_2) is the sort of the pairs whose components have sorts s_1 and s_2 respectively and $=_{12}$ is the component-wise equality predicate induced by the equality predicates $=_1$ and $=_2$ respectively.

In the absence of any knowledge of the properties of the individual component structures, the following axiom illustrates a common thread in the axiomatization of every complex data structure.

$$\phi_{\text{Pairs}} \stackrel{\text{df}}{=} \forall x: (s_1, s_2)[x =_{12} (\#1 x, \#2 x)]$$

i.e. any pair x may be re-constructed by applying the constructor $(,)$ to the pair of elements obtained by applying the deconstructors $\#1$ and $\#2$ to the pair respectively.

Besides the above axiom, the equality of pairs would depend on having equality as a predicate in the signatures Σ_1 and Σ_2 as well. We assume that $=$ is available in both Σ_1 and Σ_2 . Of course, the three equality relations all operate on different signatures (and hence structures).

We disambiguate the three equality relations as follows. Assume $=_1 : s_1^2$, $=_2 : s_2^2$ and $=_{12} : (s_1, s_2)^2$ are the respective equality predicates. The second axiom lifts equality on the individual components to equality on pairs.

$$\phi_{Pairs-equality} \stackrel{df}{=} \forall x, y: (s_1, s_2) [x =_{12} y \leftrightarrow (\#1 x =_1 \#1 y \wedge \#2 x =_2 \#2 y)]$$

In summary we have the following formalization of Pairs.

Pairs

Given sorts s_1 and s_2 with the respective equality predicates $=_1 : s_1^2$ and $=_2 : s_2^2$, the theory of pairs is given by

$$\Sigma = \{ (,) : s_1 \times s_2 \longrightarrow (s_1, s_2), \\ \#1 : (s_1, s_2) \longrightarrow s_1, \#2 : (s_1, s_2) \longrightarrow s_1; \\ =_{12} : (s_1, s_2)^2 \}$$

$$\phi_{Pairing} \stackrel{df}{=} \forall x: (s_1, s_2)[x =_{12} (\#1 x, \#2 x)]$$

$$\phi_{Pairs-=} \stackrel{df}{=} \forall x, y: (s_1, s_2) \\ [x =_{12} y \leftrightarrow ((\#1 x =_1 \#1 y) \wedge (\#2 x =_2 \#2 y))]$$

$$\Phi_{Pairs} \stackrel{df}{=} \{\phi_{Pairing}, \phi_{Pairs-=}\}$$

3-sorted logic of pairs In a first-order logic of pairing, with the signature $\Sigma_1 \cup \Sigma_2 \cup \Sigma$ we would require that every bound variable used in a predicate be “guarded” by its sort membership predicate.

Exercise 41.1

Let **A**, **B** and **C** be structures with carrier sets \mathbb{A} , \mathbb{B} and \mathbb{C} respectively.

1. From the theory of **pairs** derive first-order axioms for the following structures.
 - (a) **Pairs[A, Pairs[B, C]]**
 - (b) **Pairs[Pairs[A, B], C]**
2. Define the first-order theory of triples $\text{Triples}[\mathbf{A}, \mathbf{B}, \mathbf{C}] = \langle \mathbb{A} \times \mathbb{B} \times \mathbb{C}; (\textcolor{violet}{, , }), \#1, \#2, \#3; = \rangle$ over structures **A**, **B** and **C** respectively.
3. Note that $\mathbb{A} \times \mathbb{B} \times \mathbb{C} \neq \mathbb{A} \times (\mathbb{B} \times \mathbb{C}) \neq (\mathbb{A} \times \mathbb{B}) \times \mathbb{C} \neq \mathbb{A} \times \mathbb{B} \times \mathbb{C}$ since the binary cartesian product operation is neither commutative nor associative. However there are obvious and trivial isomorphisms which associate each triple (a, b, c) with each of the pairs $(a, (b, c))$ and $((a, b), c)$ respectively. Identify the obvious association between the axioms of the first order theory of **Triples[A, B, C]** and the axioms and derived rules of **Pairs[A, Pairs[B, C]]** and **Pairs[Pairs[A, B], C]** respectively

41.2. Arrays as functions

Arrays may be regarded as valuations from a non-zero finite ordinal \underline{n} (the indexing set of sort \underline{n}) to a set of values (drawn from a sort s). We use the usual notation $A[i]$ to denote the array element with index i . We denote array updates by $A\langle i := v \rangle$ to denote a new array which is point-wise equal to the array A except (possibly) for the index i where it has the value v .

For each non-zero finite ordinal \underline{n} , we assume the structure

$$\underline{\mathbf{n}} = \langle \underline{n}, ;=_{\underline{n}}, <_{\underline{n}} \rangle$$

where $\underline{n} = \{0, \dots, n - 1\}$.

Our arrays deal with the axiomatization of what are known as “functional arrays” in the sense that the update operation $A\langle i := v \rangle$ results in a different array than the original than actually updating only the index element of the array. This is different from the usual array update operation in imperative programming where array updates are actually side-effects. While this may sound idealistic it is actually useful when trying to prove programs, since “side-effects” are complicated to reason about and to keep track of.

The Array Structure: Signature

Let $\mathbf{A} = \langle \mathbb{A}, \Sigma_{\textcolor{blue}{s}} \rangle$ be a structure with sort $\textcolor{blue}{s}$ such that $=_{\textcolor{blue}{s}}$ is the equality predicate in $\Sigma_{\textcolor{blue}{s}}$. Let \underline{n} be any non-zero finite ordinal (see section 21.1).

$$\mathbf{Arrays}[\mathbf{A}, \underline{n}] = \langle \mathbb{A}_{\text{arrays}}, \Sigma_{s_n_array} \rangle$$

is the structure of arrays of n elements of \mathbb{A} indexed by the elements $\{0, \dots, n - 1\}$ where

$$\begin{aligned}\Sigma_{s_n_array} = & \{ [\] : s_n_array, \underline{n} \longrightarrow s \\ & \langle := \rangle : s_n_array, \underline{n}, s \longrightarrow s_n_array \\ & =_{s_n_array} : s_n_array^2 \}\end{aligned}$$

The Array Structure: Axioms

$$\phi[] \stackrel{df}{=} \forall A: s_n_array \forall i: \underline{n} [A[i]: s]$$

$$\phi[] \stackrel{df}{=} \forall A: s_n_array \forall i, j: \underline{n} [i =_n j \rightarrow A[i] =_s A[j]]$$

$$\phi_{\langle := \rangle} \stackrel{df}{=} \forall A: s_n_array \forall i: \underline{n} \forall v: s [A\langle i := v \rangle[j]: s_n_array]$$

$$\begin{aligned} \phi_{= \langle := \rangle} \stackrel{df}{=} & \forall A: s_n_array \forall i, j: \underline{n} \forall v: s \\ & [i =_n j \rightarrow A\langle i := v \rangle[j] =_s v] \end{aligned}$$

$$\begin{aligned} \phi_{\neq \langle := \rangle} \stackrel{df}{=} & \forall A: s_n_array \forall i, j: \underline{n} \forall v: s \\ & [i \neq_n j \rightarrow A\langle i := v \rangle[j] =_s A[j]] \end{aligned}$$

$$\begin{aligned} \phi_{=s_n_array} \stackrel{df}{=} & \forall A, B: s_n_array [A =_{s_n_array} B \leftrightarrow \\ & \forall i: \underline{n} [A[i] =_s B[i]]] \end{aligned}$$

41.3. Inductively Defined Data-types

The axiomatizations of the above structures was rather simple. What is more interesting are inductively defined (often called recursive) data structures and the axioms of their first-order theories. We devote the rest of our treatment to such inductively defined data structures viz. lists, stacks, queues and binary trees. Inductively defined data-types are characterised by two other important considerations (besides all that characterises the non-inductively defined structures).

Induction rule(s). Any inductively defined data-type needs to have a basis and induction step. Consequently to prove general properties of the data-type we need to be able to use the inductive construction process to prove many properties.

Acyclicity axiom(s). Any inductively defined data-type also needs to have distinctness axioms. One way of understanding the need for this is to hark back to the **inductive definition** of the naturals which require an infinite number of **distinctness axioms** $\Phi_{(+1)^{\infty}-\text{distinct}}$ to ensure that the construction does indeed yield a countably infinite number of distinct elements as the “**intended interpretation**” (points 4 and 5). We define the distinctness axioms for lists only and leave the the distinctness axioms for the other data-types as exercises (see problem 4 in exercise 41.3, problem 5 in 41.4 and problem 1 in 41.5).

41.4. Lists: The First-order Theory

A list in ML-like languages conforms to the following data-type definition.

```
datatype 'a list = []
            | :: of 'a * 'a list -> 'a list
infix ::
```

In effect, an '`'a list` of elements of type '`'a` is the smallest set (of terms) containing the empty list (`[]`) and closed under the (infix) cons `::` operation. This is the “functional” view of a list viz. as a term of an expression language. This view of a list is very unlike the notions of singly linked lists and doubly linked lists commonly treated in textbooks on data structures. To understand the relative differences refer to figure 13.

We may regard singly-linked lists and doubly-linked lists as particular implementations of the abstract data-type (ADT) of lists defined as terms of an expression language. In fact, in the terminology of logic, these implementations would be models of the axiomatic definition of the ADT lists defined as expressions, provided they satisfy all the axioms of the ADT.

Generally speaking, we could define list structures **Lists[A]** which refers to the lists of elements over **A**, where **A** could itself be a simple or compound structure. In other words, we treat this matter through a form of “parametric polymorphism” by making no assumptions about the first order theory

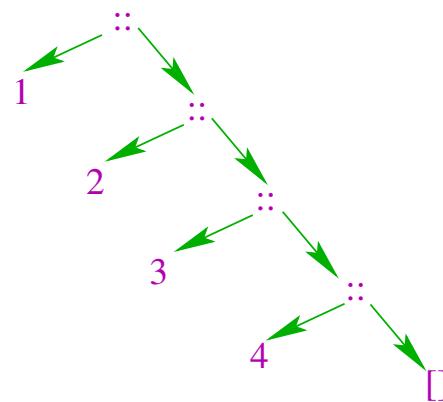
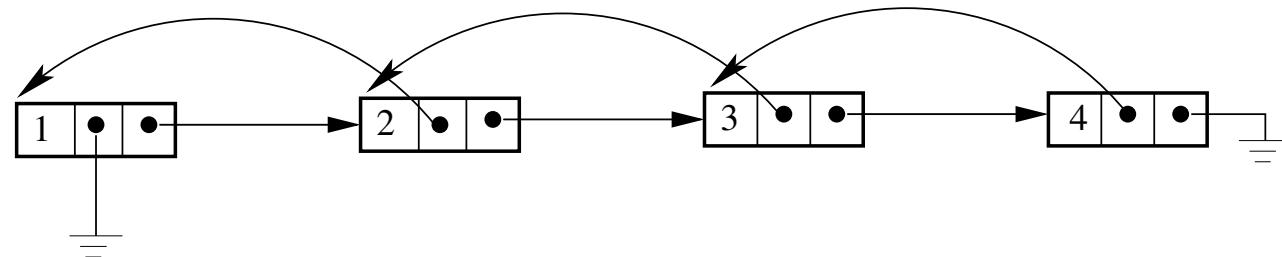


Figure 13: Schematic of a singly-linked list, a doubly-linked list and a list as term of an expression language

(with equality) of the structure **A**, except that an equality relation does exist on **A** and may be used to induce or define an equality relation on **Lists[A]** as well. Further, being a many-sorted first-order logic, not all functions and predicates are total. Hence it is necessary to make several predicates conditional by guarding them with a type-predicate. Lastly, for any sort *s* of elements which make up a list, we use *s_list* as the sort of elements of $|\text{Lists}[A]|$ and use this sort name for the infix type predicate $: s_list$. Similar remarks also apply to other inductively defined data-types such as **BinTrees[A]** that we define in the sequel. In that sense our treatment of these structures follows the rigorous type-safe approach used in the ML-type systems as opposed to the more loosely defined type systems of Lisp, C and other languages, old and new.

A signature for such a structure would consist of

Constructors. [] and ::

Deconstructors. *hd* and *tl*

Distinguished functions and predicates. *null* which defines when a list is empty.

Extension functions and predicates. Open the ML structure **List** to see the various useful functions and predicates which are present to make the structure useful and easy to work with.

A first-order axiomatization of the theory of such structures would consist typically of

Typing axioms. These define the types of the various constants functions and predicates. Since the constructors hd and tl are only partial functions it is important to introduce a “guard” for predicates involving these functions. For example the defining equation 65 would then be a guarded first-order predicate as shown in the axiom ϕ_{decomp} .

Structural axioms. These define the inter-relationships between the constructors and deconstructors. In the case of lists we have the following

$$L = [] \vee L = hd(L) :: tl(L) \quad (65)$$

Distinguished predicates. $null$ to check whether a list is empty. This is especially useful to guard the application of the hd and tl functions.

Equality axioms. Equality of compound structures is often “induced” (usually point-wise as we have seen in the case of pairs) by the equality relation on the parameters of the compound structure and needs to be defined through axioms.

Other predicates. Usually the extension of the signature may require other axioms to define the behaviours of the extension functions and predicates.

We know from the theory of lists in ML-like languages, (due to problems relating to undecidability or representation) that not all structures \mathbf{A} may have an equality relation defined on them. But in principle, an equality relation always exists. Further many structures do admit an equality relation and the corresponding equality relation on $\text{Lists}[\mathbf{A}]$ is induced by the equality relation on \mathbf{A} .

As a matter of notational convenience we deviate from the rigid **notational formalities** defined earlier and instead use more intuitively obvious variable names such as L, M etc. to range over $\text{Lists}[\mathbf{A}]$ while retaining variable symbols x, y to denote elements of \mathbf{A} . Further as a matter of naming convenience we introduce sort names which are derived from the names of sort parameters suffixed by the name so of the structure itself. Hence the sort of $\text{Lists}[\mathbf{A}]$ would be s_list where s is the sort of elements of \mathbf{A} . This allows for an easy extension of sort names (analogous to the type names in ML-like languages). Hence the sort of $\text{Lists}[\text{Lists}[\mathbf{A}]]$ would be s_list_list and the sort of $\text{Stacks}[\text{Lists}[\mathbf{A}]]$ would be named s_list_stack and so on.

The List Structure: Signature

Let $\mathbf{A} = \langle \mathbb{A}, \Sigma_s \rangle$ be a structure with sort s such that $=_s$ is the equality predicate in Σ_s . The structure

$$\text{Lists}[\mathbf{A}] = \langle \text{Lists}[\mathbb{A}], \Sigma_{s_list} \rangle$$

is the structure of lists of elements of \mathbb{A} where

$$\begin{aligned}\Sigma_{s_list} = \{ & [] : s_list, \\ & :: : s, s_list \longrightarrow s_list, \\ & hd : s_list \rightarrow s, \\ & tl : s_list \rightarrow s_list; \\ & null : s_list, \\ & =_{s_list} : s_list^2 \}\end{aligned}$$

Note.

1. Even though we are considering a single structure $\text{Lists}[\mathbf{A}]$, the fact that it is built up from \mathbf{A} implies that any axiomatization of such a structure has to respect sorts. For instance, the equality predicate $=_{s_list}$ is “induced” on $\text{Lists}[\mathbf{A}]$ by the equality predicate $=_s$ on \mathbf{A} .
2. In general, any predicate on a compound structure that is defined using predicates on the components would necessarily require that sorts be respected. Hence the first-order theory of compound structures is in general, a many-sorted one.
3. While respecting sorts, sometimes it becomes rather cumbersome to mention the sort at each stage. We will often be guilty of omitting the sort where the context makes it obvious and unambiguous. For instance, equality predicates will often be “overloaded” in this fashion.
4. It frequently happens that our signature consists of operations or constructors or functions which are only partially defined. It is necessary in such cases to specify axioms which clearly delimit the conditions under which the operation is defined. We will refer to these axioms as “typing axioms”. For example the functions hd and tl in the signature Σ_{s_list} are not total. However the signature itself does not specify for what values of the domain these functions are defined. We need to specify in first order logic the sub-domain on which these functions are well-defined.

The List Structure Axioms: 1

$$\phi_{[]} : \stackrel{df}{=} [] : s_list$$

$$\phi_{null} : \stackrel{df}{=} null([])$$

$$\phi_{cons} : \stackrel{df}{=} \forall x: s \ \forall L: s_list [(x :: L) : s_list]$$

$$\phi_{\neg null} : \stackrel{df}{=} \forall x: s \ \forall L: s_list [\neg null(x :: L)]$$

$$\phi_{hd} : \stackrel{df}{=} \forall L: s_list [\neg null(L) \rightarrow (hd(L) : s)]$$

$$\phi_{tl} : \stackrel{df}{=} \forall L: s_list [\neg null(L) \rightarrow (tl(L) : s_list)]$$

The List Structure Axioms: 2

$$\begin{aligned}\phi_{hd} &\stackrel{df}{=} \forall x: s \ \forall L: s_list [hd(x :: L) =_s x] \\ \phi_{tl} &\stackrel{df}{=} \forall x: s \forall L: s_list [tl(x :: L) =_{s_list} L] \\ \phi_{decomp} &\stackrel{df}{=} \forall L: s_list [\neg null(L) \rightarrow (L =_{s_list} hd(L) :: tl(L))] \\ \phi_{=}-inj &\stackrel{df}{=} \forall x, y: s \ \forall L, M: s_list [(x :: L =_{s_list} y :: M) \leftrightarrow \\ &\quad (x =_s y \wedge L =_{s_list} M)]\end{aligned}$$

Explanations.

1. The formulae $\phi_{[]}$, $\phi_{cons:}$, $\phi_{hd:}$ and $\phi_{tl:}$ are the typing axioms. They are analogous to Peano's postulates **P1** and **P2** which are the typing axioms for the naturals in Peano arithmetic.
2. The axioms ϕ_{null} and $\phi_{\neg null}$ distinguish between empty and non-empty lists and clearly show that the empty list is unique in the sort *s_list*.
3. The axiom ϕ_{decomp} actually defines the relation between the constructors and the deconstructor operations on lists.
4. The axiom $\phi_{=_{-inj}}$ defines the equality relation on lists as being induced by the point-wise equality relation on the underlying elements.
5. Lastly ϕ_{hd} and ϕ_{tl} define the partial functions *hd* and *tl* respectively.

The List Structure: Induction

Analogous to the **induction principle** in Peano Arithmetic we may define an induction principle for List structures in general.

$$\text{IndList. } \frac{\{[]/L\}X, \forall x: s \forall L: s_list[X \rightarrow \{(x :: L)/L\}X]}{\forall L: s_list[X]}$$

where

- L is the only free variable of X ,
- $\{[]/L\}X$ is the basis of the induction,
- $\forall x: s \forall L: s_list[X \rightarrow \{(x :: L)/L\}X]$ is the induction step and
- X the antecedent in the induction step, is the induction hypothesis

The List Structure: Acyclicity 1

Our intended interpretation requires of the “cons” operation that each such operation yields a list that is different from the list on which the operation was performed.

$$\begin{aligned}\phi_{(:)^1\text{-}distinct} &\stackrel{df}{=} \forall L: s_list \forall x_1: s [L \neq x_1 :: L] \\ \phi_{(:)^2\text{-}distinct} &\stackrel{df}{=} \forall L: s_list \forall x_1, x_2: s [L \neq x_1 :: (x_2 :: L)] \\ &\vdots \quad \vdots \quad \vdots \\ \phi_{(:)^n\text{-}distinct} &\stackrel{df}{=} \forall L: s_list \forall x_1, \dots, x_n: s [L \neq x_1 :: (\dots :: (x_n :: L))] \\ &\vdots \quad \vdots \quad \vdots\end{aligned}$$

The List Structure: Acyclicity 2

Our intended interpretation requires that no list equals its tail, or the tail of its tail or the tail of the tail of its tail, and so on.

$$\phi_{(tl)^1\text{-}distinct} \stackrel{df}{=} \forall L: s_list[\neg null(L) \rightarrow L \neq tl(L)]$$

$$\begin{aligned}\phi_{(tl)^2\text{-}distinct} &\stackrel{df}{=} \forall L: s_list[\neg null(L) \rightarrow \neg null(tl(L)) \rightarrow \\ &\quad L \neq tl(tl(L))]\\ &\vdots \quad \vdots \quad \vdots\end{aligned}$$

$$\begin{aligned}\phi_{(tl)^n\text{-}distinct} &\stackrel{df}{=} \forall L: s_list[\neg null(L) \rightarrow \dots \neg null(tl^n(L)) \rightarrow \\ &\quad L \neq tl^n(L)]\\ &\vdots \quad \vdots \quad \vdots\end{aligned}$$

Exercise 41.2

1. We have specified certain axioms and an induction principle for ML-like lists via the *signature* Σ_s and the *axioms* and an *inference rule*. The “intention” is that $\text{Lists}[\mathbf{A}]$ is the (smallest) inductively defined set of terms containing $[]$ and closed under the infix binary operation $::$.
 - (a) Which axioms actually support the above claim?
 - (b) Are there other sets (analogous to the non-standard models of Peano arithmetic) which also satisfy the same set of axioms and inference rule? If so find examples.
 - (c) Consider a set containing all finite length lists as well as infinite length lists. Does this set satisfy all the *axioms* and the *inference rule*? Justify your answer.
2. The (infix) append operation $@$ on ML-like lists is defined programmatically as follows.

```
fun [] @ M = M  
| (a :: L) @ M = a :: (L @ M)
```

- (a) Define the infix append operation on $\text{Lists}[\mathbf{A}]$.
- (b) Prove that $\langle \text{Lists}[\mathbf{A}]; [], @; = \rangle$ is a monoid. That is, you need to prove the following.

i. $[] @ L =_{s_list} L =_{s_list} L @ []$

ii. $L @ (M @ N) =_{s_list} (L @ M) @ N$

(c) Write an iterative program in $\mathcal{WH}(\Sigma_{s_list})$ which takes as input a list L and produces its reverse as list M . Prove the correctness of your program.

41.5. Stacks: The First-order Theory

Fundamentally stacks are a different abstract data-type than lists. So we have used symbols that are distinct for the operations of each data-type. Using different symbols ensures that there is no confusion when the operations of one data type are expressed (“implemented”) in terms of the operations of another abstract data type.

1. $[>]$ denotes the empty stack
2. \downarrow denotes the (infix) “push” operation on stacks. Often when we are interested in the “values” in the stack we could abbreviate for example, the complex expression

$$(((>\downarrow 1) \downarrow 2) \downarrow 3) \downarrow 4)$$

by the simpler notation (see problem 3 in exercise 41.3)

$$[1, 2, 3, 4 >$$

3. \wedge denotes the (prefix) “top of stack” function. Hence $\wedge[1, 2, 3, 4 >$ is 4.
4. \uparrow denotes the (postfix) “pop” operation. Its effect is to remove the “top of stack” element from the stack. Hence $[1, 2, 3, 4 >\uparrow$ yields the stack $[1, 2, 3 >$.

While it is regarded good programming practice to use names that are self-explanatory in nature, it is also at the cost of brevity and compactness of notation. We have chosen to use compact symbols to denote the various constructors, functions and operators to allow ease of composition and readability of compound expressions involving them.

The Stack Structure: Signature

Let $\mathbf{A} = \langle \mathbb{A}, \Sigma_s \rangle$ be a structure with sort s such that $=_s$ is the equality predicate in Σ_s . The structure

$$\text{Stacks}[\mathbf{A}] = \langle \text{Stacks}[\mathbb{A}], \Sigma_{s_stack} \rangle$$

is the structure of stacks of elements of \mathbb{A} where

$$\begin{aligned}\Sigma_{s_stack} = \{ &[> : s_stack, \\ &\downarrow : s_stack, s \longrightarrow s_stack, \\ &\uparrow : s_stack \rightharpoonup s_stack, \\ &\wedge : s_stack \rightharpoonup s; \\ &nulls : s_stack, \\ &=_s : s_stack \times s_stack\}\end{aligned}$$

The Stack Structure Axioms: 1

$$\phi[>:] \stackrel{df}{=} [>: s_stack]$$

$$\phi_{nulls} \stackrel{df}{=} nulls([>])$$

$$\phi_{\downarrow} \stackrel{df}{=} \forall x: s \ \forall S: s_stack [(S \downarrow x): s_stack]$$

$$\phi_{\neg nulls} \stackrel{df}{=} \forall x: s \ \forall S: s_stack [\neg nulls(S \downarrow x)]$$

$$\phi^{\wedge} \stackrel{df}{=} \forall S: s_stack [\neg nulls(S) \rightarrow (\wedge S): s]$$

$$\phi^{\wedge} \stackrel{df}{=} \forall x: s \ \forall S: s_stack [\wedge (S \downarrow x) =_s x]$$

The Stack Structure Axioms: 2

$$\phi_{\uparrow} \stackrel{df}{=} \forall S: s_stack [\neg nulls(S) \rightarrow (S \uparrow): s_stack]$$

$$\phi_{\uparrow} \stackrel{df}{=} \forall x: s \forall S: s_stack [(S \downarrow x) \uparrow =_{s_stack} S]$$

$$\phi_{decomp} \stackrel{df}{=} \forall S: s_stack [\neg nulls(S) \rightarrow S =_{s_stack} (S \uparrow) \downarrow (\wedge S)]$$

$$\phi_{= -inj} \stackrel{df}{=} \forall x, y: s \forall S, T: s_stack [(S \downarrow x =_{s_stack} T \downarrow y) \leftrightarrow x =_s y \wedge S =_{s_stack} T]$$

The Stack Structure: Induction

Analogous to the **induction principle** for lists we have

$$\text{IndStack. } \frac{\{[>/S]X, \forall x: s \forall S: s_stack[X \rightarrow \{(S \downarrow x)/S\}X]}{\forall S: s_stack[X]}$$

where

- S is the only free variable of X ,
- $\{[>/S]X$ is the basis of the induction,
- $\forall x: s \forall S: s_stack[X \rightarrow \{(S \downarrow x)/S\}X]$ is the induction step and
- X the antecedent in the induction step, is the induction hypothesis.

Exercise 41.3

1. Use the *list structure* to define a *stack structure* in terms of lists.
2. Use the *first-order theory of lists* to prove that your stack structure does indeed satisfy all the *axioms* of the first-order theory of stacks.
3. Prove that for every stack expression $S: s_stack$, $S =_{s_stack} T$ where $T: s_stack$ is a stack expression with no occurrences of operations \uparrow or \wedge .
4. State the acyclicity axioms for stacks.

41.6. Queues: The First Order Theory

We proceed in a manner analogous to stacks here.

1. $<<$ denotes the empty queue.
2. \swarrow is the infix binary operation to “enqueue” an element to a given queue. Often when we are interested only in the “sequence of values” in a queue, we could abbreviate (for example) the complex expression

$$(((<<\swarrow 1) \swarrow 2) \swarrow 3) \swarrow 4)$$

by the simpler notation (see problem 1 in exercise 41.4)

$$< 1, 2, 3, 4 <$$

3. \curvearrowleft denotes the prefix “head of the queue” function. The head of the queue $< 1, 2, 3, 4 <$ is the value 1.
4. \nwarrow denotes the (prefix) “dequeue” operation. Hence $\nwarrow < 1, 2, 3, 4 <$ yields the queue $< 2, 3, 4 <$. The effect of a dequeue operation is to “remove” the head of the queue.

The Queue Structure: Signature

Let $\mathbf{A} = \langle \mathbb{A}, \Sigma_s \rangle$ be a structure with sort s such that $=_s$ is the equality predicate in Σ_s . The structure

$$\text{Queues}[\mathbf{A}] = \langle \text{Queues}[\mathbb{A}], \Sigma_{s_queue} \rangle$$

is the structure of queues of elements of \mathbb{A} where

$$\begin{aligned}\Sigma_{s_queue} = \{ & \ll : s_queue, \\ & \leftarrow : s_queue, s \rightarrow s_queue, \\ & \rightarrow : s_queue \rightarrow s_queue, \\ & \circlearrowleft : s_queue \rightarrow s; \\ & nullq : s_queue, \\ & =_{s_queue} : s_queue^2 \}\end{aligned}$$

The Queue Structure Axioms: 1

$$\phi_{<<} \stackrel{df}{=} << : s_queue$$

$$\phi_{nullq} \stackrel{df}{=} nullq(<<)$$

$$\phi_{\swarrow} \stackrel{df}{=} \forall x: s \ \forall Q: s_queue [(Q \swarrow x) : s_queue]$$

$$\phi_{\neg nullq} \stackrel{df}{=} \forall x: s \ \forall Q: s_queue [\neg nullq(Q \swarrow x)]$$

$$\phi_{\curvearrowright} \stackrel{df}{=} \forall Q: s_queue [\neg nullq(Q) \rightarrow (\curvearrowright Q) : s]$$

$$\phi_{\curvearrowright <<} \stackrel{df}{=} \forall x: s [\curvearrowright (<< \swarrow x) =_s x]$$

$$\begin{aligned} \phi_{\curvearrowright \neg nullq} \stackrel{df}{=} & \forall x: s \forall Q: s_queue \\ & [\neg nullq(Q) \rightarrow (\curvearrowright (Q \swarrow x) =_s \curvearrowright Q)] \end{aligned}$$

The Queue Structure Axioms: 2

$$\phi_{\leftarrow} \stackrel{df}{=} \forall Q: s_queue [\neg nullq(Q) \rightarrow ((\leftarrow Q): s_queue)]$$

$$\begin{aligned}\phi_{\leftarrow \neg nullq} &\stackrel{df}{=} \forall x: s \forall Q: s_queue \\ &[\neg nullq(Q) \rightarrow (\leftarrow (Q \swarrow x) =_{s_queue} (\leftarrow Q) \swarrow x)]\end{aligned}$$

$$\begin{aligned}\phi_{= inj} &\stackrel{df}{=} \forall x, x': s \forall Q, Q': s_queue \\ &[((Q \swarrow x) =_{s_queue} (Q' \swarrow x')) \leftrightarrow \\ &(Q =_{s_queue} Q' \wedge x =_s x')]\end{aligned}$$

The Queue Structure: Induction

$$\text{IndQueue. } \frac{\{ \ll /Q \} X, \forall x: s \forall Q: s_queue[X \rightarrow \{ (Q \swarrow x) / Q \} X]}{\forall Q: s_queue[X]}$$

where

- Q is the only free variable of X ,
- $\{ \ll /Q \} X$ is the basis of the induction,
- $\forall x: s \forall Q: s_queue[X \rightarrow \{ (Q \swarrow x) / Q \} X]$ is the induction step and
- X the antecedent in the induction step, is the induction hypothesis.

Exercise 41.4

1. Prove that for every queue expression $Q: s_queue$, $Q =_{s_queue} R$ where $R: s_queue$ is a queue expression with no occurrences of operations ↪ or ↤.
2. The abstract data type for “double-ended queues” $\text{DEQueues[A]} = \langle \text{DEQueues[A]}, \Sigma_{s_deq} \rangle$ of values drawn from a structure A is given by

$$\begin{aligned}\Sigma_{s_deq} = & \{ >< : s_deq, \\ & \searrow : s, s_deq \rightarrow s_deq, \swarrow : s_deq, s \rightarrow s_deq, \\ & \nwarrow : s_queue \rightarrow s_deq, \nearrow : s_deq \rightarrow s_queue, \\ & \curvearrowleft : s_deq \rightarrow s, \curvearrowright : s_deq \rightarrow s; \\ & \text{nullq} : s_deq, =_{s_deq} : s_deq^2 \} \end{aligned}$$

The various functions and constructors are briefly described below

- $><$: the empty double-ended queue.
- $x \searrow D$: the infix operation to enqueue element x at the front of the double-ended queue D .
- $D \swarrow y$: the infix operation to enqueue element y at the rear of the double-ended queue D .

- $\curvearrowleft D$: the prefix function which yields the first element of the double-ended queue D .
- $D \curvearrowright$: the postfix function which yields the last element of the double-ended queue D .
- $\nwarrow D$: the prefix operation which dequeues the first element from the double-ended queue D .
- $D \nearrow$: the postfix operation which dequeues the last element from the double-ended queue D .
- $\text{nulldeq}(D)$: test of emptiness of the double-ended queue D .
- $D =_{s_deq} D'$: equality predicate for double-ended queues.

Write a first-order axiomatization of the structure **DEQueues[A]** (Hint: Also see example 37.3).

3. Let **A** be any structure. Express the operations of **Queues[A]** in terms of the operations of **Lists[A]** and prove that the queue axioms are satisfied.
4. We may implement a queue as a pair of stacks or as a pair of lists. This is often useful since it allows both enqueueing and dequeuing to be performed as amortised constant time operations. Prove the correctness of your implementation.
5. State the acyclicity axioms for queues.

41.7. Binary Trees: The First-order Theory

We define an inductive user-defined data-type of binary trees in ML-like languages conforming to the following data-type definition.

```
datatype 'a bintree = Empty | Node of 'a * 'a bintree * 'a bintree
```

where each node is labelled by a value from the sort parameter. In this case a leaf-node is actually a (sub-)tree with empty sub-trees as children (nodes 4,5, and 6 in figure 14). Further any internal node with only one child (either left or right) has an empty tree as the other child underneath it (node 2 in figure 14).

Since we require a linear representation we prefer to grow our trees from left to right.

A brief explanation of the signature of this data-type is given below:

1. \triangleleft denotes the empty binary-tree.
2. $\begin{pmatrix} x & \nearrow T_R \\ & \searrow T_L \end{pmatrix}$ denotes a binary tree with root labelled x and left subtree T_L and right subtree T_R .
3. $rt(T)$, $lst(T)$ and $rst(T)$ denote respectively the root, the left subtree and the right subtree of the non-empty binary tree T .

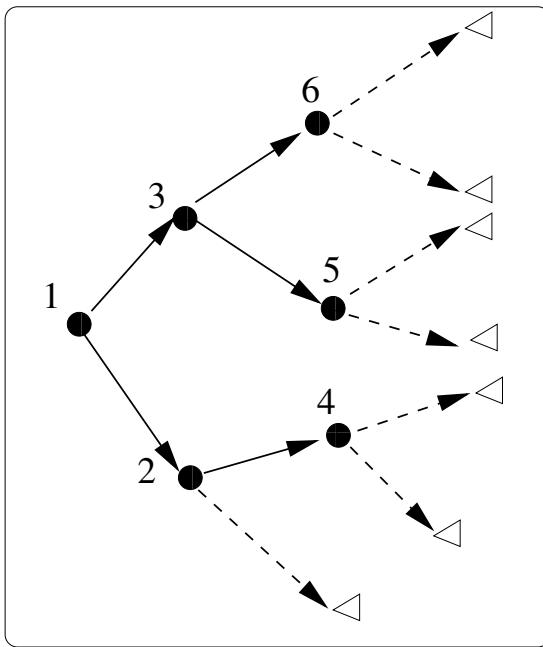
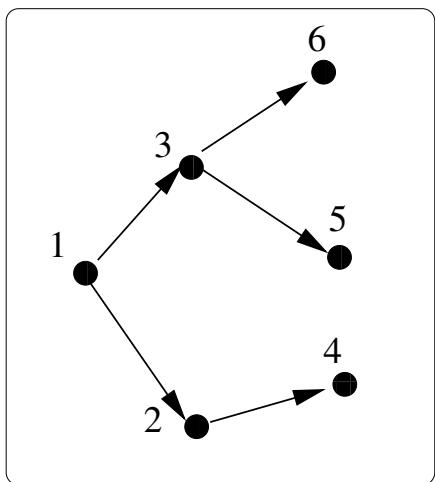


Figure 14: Example binary tree with 6 nodes drawn left to right

Binary Trees: Signature

Let $\mathbf{A} = \langle \mathbb{A}, \Sigma_{\textcolor{blue}{s}} \rangle$ be a structure with sort $\textcolor{blue}{s}$ such that $=_{\textcolor{blue}{s}}$ is the equality predicate in $\Sigma_{\textcolor{blue}{s}}$. The structure

$$\mathbf{BinTrees}[\mathbf{A}] = \langle \textcolor{brown}{\mathbb{B}\text{in}\text{T}\text{rees}}[\mathbb{A}], \Sigma_{\textcolor{blue}{s}_bt} \rangle$$

is the structure of binary trees of elements of \mathbb{A} where

$$\begin{aligned}\Sigma_{\textcolor{blue}{s}_bt} = \{ & \textcolor{purple}{\triangleleft} : s_bt, \\ & \textcolor{purple}{lst} : s_bt \rightarrow s_bt, \\ & \textcolor{purple}{rst} : s_bt \rightarrow s_bt, \\ & \textcolor{purple}{rt} : s_bt \rightarrow s; \\ & \textcolor{green}{nullbt} : s_bt, \\ & =_{\textcolor{blue}{s}_bt} : s_bt^2 \}\end{aligned}$$

The Binary Tree Axioms: 1

$$\phi_{\triangleleft} \stackrel{df}{=} \triangleleft : s_bt$$

$$\phi_{nullbt} \stackrel{df}{=} nullbt(\triangleleft)$$

$$\phi_{node} \stackrel{df}{=} \forall x: s \forall T_L, T_R: s_bt [\begin{pmatrix} & \nearrow^{T_R} \\ x & \searrow^{T_L} \end{pmatrix} : s_bt]$$

$$\phi_{\neg nullbt} \stackrel{df}{=} \forall x: s \forall T_L, T_R s_bt [\neg nullbt(\begin{pmatrix} & \nearrow^{T_R} \\ x & \searrow^{T_L} \end{pmatrix})]$$

The Binary Tree Axioms: 2

$$\phi_{rt} \stackrel{df}{=} \forall T: s_bt [\neg \text{nullbt}(T) \rightarrow rt(T): s]$$

$$\phi_{lst} \stackrel{df}{=} \forall T: s_bt [\neg \text{nullbt}(T) \rightarrow lst(T): s_bt]$$

$$\phi_{rst} \stackrel{df}{=} \forall T: s_bt [\neg \text{nullbt}(T) \rightarrow rst(T): s_bt]$$

$$\phi_{rt} \stackrel{df}{=} \forall x: s \forall T_L, T_R: s_bt [rt(\begin{array}{c} \nearrow T_R \\ x \\ \searrow T_L \end{array}) =_s x]$$

$$\phi_{lst} \stackrel{df}{=} \forall x: s \forall T_L, T_R: s_bt [lst(\begin{array}{c} \nearrow T_R \\ x \\ \searrow T_L \end{array}) =_{s_bt} T_L]$$

$$\phi_{rst} \stackrel{df}{=} \forall x: s \forall T_L, T_R: s_bt [rst(\begin{array}{c} \nearrow T_R \\ x \\ \searrow T_L \end{array}) =_{s_bt} T_R]$$

The Binary Tree Axioms: 3

$$\begin{aligned}\phi_{decomp} &\stackrel{df}{=} \forall T: s_bt [\neg nullbt(T) \rightarrow (T =_{s_bt} \left(rt(T) \begin{array}{c} \nearrow rst(T) \\ \searrow lst(T) \end{array} \right))] \\ \phi_{=_{s_bt}} &\stackrel{df}{=} \forall x, x': s \forall T_L, T'_L, T_R, T'_R: s_bt \\ &[(\left(x \begin{array}{c} \nearrow T_R \\ \searrow T_L \end{array} \right) =_{s_bt} \left(x' \begin{array}{c} \nearrow T'_R \\ \searrow T'_L \end{array} \right)) \rightarrow \\ &(x =_s x' \wedge T_L =_{s_bt} T'_L \wedge T_R =_{s_bt} T'_R)]\end{aligned}$$

The Binary Tree Induction Rule

We may define an induction principle for binary trees.

$$\frac{\begin{array}{c} \{ \triangleleft / T \} X, \\ \forall x: s \forall T_L, T_R: s_bt [(\{ T_L / T \} X \wedge \{ T_R / T \} X) \rightarrow \\ \text{IndBinTrees.} \end{array}}{\forall T: s_bt [X]}$$

$\{ \left(\begin{array}{c} \nearrow T_R \\ x \\ \searrow T_L \end{array} \right) / T \} X$

Explanation. In the induction rule

- T is the only free variable of X ,
- $\{\triangleleft/T\}X$ is the basis of the induction,
- $\forall x: s \forall T_L, T_R: s_bt[(\{T_L/T\}X \wedge \{T_R/T\}X) \rightarrow \{ \left(x \begin{array}{c} \nearrow^{T_R} \\ \searrow^{T_L} \end{array} \right) / T \} X]$ is the induction step and
- $(\{T_L/T\}X \wedge \{T_R/T\}X)$ the antecedent in the induction step, is the induction hypothesis.

Exercise 41.5

1. State the acyclicity axioms for binary trees (Hint: You need to specify that no binary tree equals any of its proper sub-trees).
2. It is possible to define a different data-type for binary trees which is more in keeping with the requirements of algorithms on binary-trees. In the usual construction, leaves are treated separately from interior nodes of the tree and there are no empty trees. Such a binary tree may be defined in ML-like languages as

```
datatype 'a bintree' = Leaf of 'a | Node of 'a * 'a bintree' * 'a bintree'
```

Define the modified signature and axioms (and an induction rule) for the modified data-type

$$\text{BinTrees}'[\mathbf{A}] = \langle \text{BinTrees}'[\mathbf{A}], \Sigma_{s_bt} \rangle$$

3. Our construction of binary trees may be criticised for creating an “artificial” difference between the trees $\begin{pmatrix} x \\ \nearrow T \\ \searrow \square \end{pmatrix}$ and $\begin{pmatrix} x \\ \nearrow \square \\ \searrow T \end{pmatrix}$.

- (a) Define a new data-type definition to eliminate this distinction for nodes of out-degree 1.
- (b) Define the signature and the structure of this new data-type, $\text{BinTrees}''[\mathbf{A}]$ without including concepts like out-degree or numbers in any way.

(c) Define the modified set of first-order axioms and the induction rule for this new data-type.

(Hint: Define two different constructors for nodes of out-degree 1 and out-degree 2.)

HOME PAGE

◀◀

◀

▶

▶▶

LCS November 4, 2018

Go BACK

FULL SCREEN

CLOSE

1015 OF 1040

QUIT

42: Miscellaneous Examples

1. Random Examples

Random Examples

What follows is a motley collection of randomly chosen solved examples.

Example 42.1 We know that the set $\{\neg, \wedge, \vee\}$ is adequate and functionally complete for propositional logic. Prove that the set $\{\rightarrow, \perp\}$ is adequate for \mathcal{P}_0 , by expressing \neg , \wedge and \vee in terms of \rightarrow and \perp .

From the semantics of propositional logic it is easy to show that the following equivalences hold.

$$\begin{aligned}\neg\phi &\Leftrightarrow \phi \rightarrow \perp \\ \phi \vee \psi &\Leftrightarrow (\phi \rightarrow \perp) \rightarrow \psi \\ \phi \wedge \psi &\Leftrightarrow (\phi \rightarrow (\psi \rightarrow \perp)) \rightarrow \perp\end{aligned}$$

Since we know that $\{\wedge, \vee, \neg\}$ is an adequate set the above logical equivalences show that each operator expressible in terms of the operators $\{\wedge, \vee, \neg\}$ is also expressible by the set $\{\rightarrow, \perp\}$.

Example 42.2 Using the semantics of predicate logic prove that $\mathcal{O}x[\phi] \Leftrightarrow \mathcal{O}y[\{y/x\}\phi]$ for all $y \notin FV(\phi)$, $\mathcal{O} \in \{\forall, \exists\}$.

Since $y \notin FV(\phi)$, the substitution $\{y/x\}$ is admissible in ϕ . By the substitution lemma for formulae (lemma 19.7 of the slides) it follows that for any interpretation (A, v) , if $v(y) = a$ then $\mathcal{T}[\{y/x\}\phi]_v = \mathcal{T}[\phi]_{v[x:=a]} = \mathcal{T}[\phi]_{v[x:=v(y)]}$. It is then easy to see that for each value $a \in |A|$,

1. $\mathcal{T}[\{y/x\}\phi]_{v[y:=a]} = 1$ iff $\mathcal{T}[\phi]_{v[x:=a]} = 1$ and
2. $\mathcal{T}[\{y/x\}\phi]_{v[y:=a]} = 0$ iff $\mathcal{T}[\phi]_{v[x:=a]} = 0$.

Hence

$$\begin{aligned} & \mathcal{T}[\mathcal{O}y[\{y/x\}\phi]]_v \\ &= \bigcirc\{\mathcal{T}[\{y/x\}\phi]_{v'} \mid v' =_{\setminus y} v\} \\ &= \bigcirc\{\mathcal{T}[\phi]_{v'} \mid v' =_{\setminus x} v\} \\ &= \mathcal{T}[\mathcal{O}x[\phi]]_v \end{aligned}$$

where $\bigcirc = \prod$ or \sum according as \mathcal{O} is \forall or \exists .

Example 42.3 Let Σ be a signature containing at least one constant symbol c . Construct an example of a set Φ containing at least two closed first-order formulae such that the following conditions are satisfied.

1. Φ has a model (specify the model), but
2. Φ has no Herbrand model, unless there are at least two constant symbols.

Let p be a unary predicate. Now consider the set of formulae $\{p(c), \exists x[\neg p(x)]\}$. Let $\Sigma = \{c : \rightarrow s; p : s\}$. Clearly the Herbrand base consists of the singleton set $\{c\}$ and $\{c\} \not\models \Phi$ regardless of whether $p(c)$ is true or false. On the other hand consider the set \mathbb{Z} with c being interpreted as the number 0 and p being interpreted as the predicate “is zero”. Clearly $\mathbb{Z} \models \Phi$ since both $p(c)$ and $\exists x[\neg p(x)]$ are true in \mathbb{Z} .

Example 42.4 Consider the following statements.

All barbers in Jonesville are men. There are many men in Jonesville. Every barber in Jonesville shaves exactly those who do not shave themselves.

Which of the following statements yield models of Jonesville that are consistent with the above statements?

1. *There are only clean-shaven men in Jonesville.*
2. *There are only unshaven men in Jonesville.*
3. *There are both clean-shaven men and unshaven men in Jonesville.*
4. *There no barbers in Jonesville.*
5. *There are at least two barbers in Jonesville.*
6. *Every barber in Jonesville shaves every other barber.*

The statement “Every barber in Jonesville shaves exactly those who do not shave themselves” may be translated as

$$\phi_3 \stackrel{df}{=} \forall \mathbf{x} [b(\mathbf{x}) \rightarrow \forall \mathbf{y} [\neg s(\mathbf{y}, \mathbf{y}) \leftrightarrow s(\mathbf{x}, \mathbf{y})]] \quad (66)$$

where $b(\textcolor{violet}{x})$ stands for “ $\textcolor{violet}{x}$ is a barber in Joneville” and $s(\textcolor{violet}{x}, \textcolor{violet}{y})$ stands for “ $\textcolor{violet}{x}$ shaves $\textcolor{violet}{y}$ ”. We now prove that there is no barber (in Jonesville) by showing that

$$\forall \textcolor{violet}{x}[b(\textcolor{violet}{x}) \rightarrow \forall \textcolor{violet}{y}[\neg s(\textcolor{violet}{y}, \textcolor{violet}{y}) \rightarrow s(\textcolor{violet}{x}, \textcolor{violet}{y})]] \models \neg \exists x[b(\textcolor{violet}{x})] \quad (67)$$

Let $\psi \stackrel{\text{df}}{=} \exists x[b(\textcolor{violet}{x})]$ and $\Gamma = \{\phi_3, \psi\}$. We then have

$$\frac{\exists E \frac{\Gamma \vdash \exists x[b(\textcolor{violet}{x})]}{\Gamma \vdash_{\exists E} b(\textcolor{red}{a})} \quad \forall E \frac{\Gamma \vdash \forall \textcolor{violet}{x}[b(\textcolor{violet}{x}) \rightarrow \forall \textcolor{violet}{y}[\neg s(\textcolor{violet}{y}, \textcolor{violet}{y}) \leftrightarrow s(\textcolor{violet}{x}, \textcolor{violet}{y})]]}{\Gamma \vdash b(\textcolor{red}{a}) \rightarrow \forall \textcolor{violet}{y}[\neg s(\textcolor{violet}{y}, \textcolor{violet}{y}) \leftrightarrow s(\textcolor{red}{a}, \textcolor{violet}{y})]} \quad \forall E \frac{\forall \textcolor{violet}{y}[\neg s(\textcolor{violet}{y}, \textcolor{violet}{y}) \leftrightarrow s(\textcolor{red}{a}, \textcolor{violet}{y})]}{\neg s(\textcolor{red}{a}, \textcolor{red}{a}) \leftrightarrow s(\textcolor{red}{a}, \textcolor{red}{a})}}$$

We may prove from Exercise 14.3 of the notes that $\neg s(\textcolor{red}{a}, \textcolor{red}{a}) \leftrightarrow s(\textcolor{red}{a}, \textcolor{red}{a}) \Leftrightarrow \neg s(\textcolor{red}{a}, \textcolor{red}{a}) \wedge s(\textcolor{red}{a}, \textcolor{red}{a})$ from which by rule $\perp E$ the conclusion (67) follows. Having proved the above, for all of the statements (except statement (v)) we may construct a single model **MJ** of Men in Jonesville with a carrier set **MJ**, containing at least two elements m_1, m_2 , such that $s(\textcolor{violet}{x}, \textcolor{violet}{x})$ means “ $\textcolor{violet}{x}$ is clean-shaven” and $\neg s(\textcolor{violet}{x}, \textcolor{violet}{x})$ means “ $\textcolor{violet}{x}$ is unshaven” (though this can actually be derived if we pursue the more literal translation of “ $\textcolor{violet}{x}$ is clean-shaven” as $\exists y[s(y, \textcolor{violet}{x})]$ and “ $\textcolor{violet}{x}$ is unshaven” as $\neg \exists y[s(y, \textcolor{violet}{x})]$).

1. **MJ** $\Vdash \forall \textcolor{violet}{x}[s(\textcolor{violet}{x}, \textcolor{violet}{x})]$

2. **MJ** $\Vdash \forall \textcolor{violet}{x}[\neg s(\textcolor{violet}{x}, \textcolor{violet}{x})]$
3. **MJ** $\Vdash \exists \textcolor{violet}{x}[s(\textcolor{violet}{x}, \textcolor{violet}{x})] \wedge \exists \textcolor{violet}{x}[\neg s(\textcolor{violet}{x}, \textcolor{violet}{x})]$
4. **MJ** $\Vdash \neg \exists \textcolor{violet}{x}[b(\textcolor{violet}{x})]$
5. **MJ** $\not\Vdash \exists \textcolor{violet}{x}, \textcolor{violet}{y}[\neg(\textcolor{violet}{x} = \textcolor{violet}{y}) \wedge b(\textcolor{violet}{x}) \wedge b(\textcolor{violet}{y})]$ for any structure **MJ** satisfying $\{\phi_1, \phi_2, \phi_3\}$.
6. **MJ** $\Vdash \forall \textcolor{violet}{x}, \textcolor{violet}{y}[(b(\textcolor{violet}{x}) \wedge b(\textcolor{violet}{y}) \wedge \neg(\textcolor{violet}{x} = \textcolor{violet}{y})) \rightarrow s(\textcolor{violet}{x}, \textcolor{violet}{y})]$

Example 42.5 Use first-order resolution to prove the following argument where b is a unary predicate symbol and s is a binary predicate symbol.

$$\forall x[b(x) \rightarrow \forall y[\neg s(y, y) \leftrightarrow s(x, y)]] \models \neg \exists x[b(x)]$$

We need to show that the set $\{\forall x[b(x) \rightarrow \forall y[\neg s(y, y) \leftrightarrow s(x, y)]], \exists x[b(x)]\}$ is unsatisfiable. We first convert each formula into PCNF and then skolemize if necessary. $\exists x[b(x)]$ gets skolemized to $b(a)$. The PCNF of $\forall x[b(x) \rightarrow \forall y[\neg s(y, y) \leftrightarrow s(x, y)]]$ is obtained by moving the quantifier $\forall y$ and then converting the quantifier-free body of the formula into CNF. This gives us the logically equivalent formula $\forall x, y[\neg b(x) \vee ((\neg s(y, y) \vee \neg s(x, y)) \wedge (s(x, y) \vee s(x, y)))]$ which on distributing $\neg b(x)$ over the conjunction and standardizing the variables apart yields the following clauses.

$$\begin{aligned}C_1 &= \{\neg b(x_1), \neg s(y_1, y_1), \neg s(x_1, y_1)\} \\C_2 &= \{\neg b(x_2), s(x_2, y_2), s(y_2, y_2)\} \\C_3 &= \{b(a)\}\end{aligned}$$

Resolving on the atom b with the mgu $\{a/x_2, a/x_1\}$ yields

$$\begin{aligned}C_1 &= \{\neg s(y_1, y_1), \neg s(a, y_1)\} \\C_2 &= \{s(a, y_2), s(y_2, y_2)\}\end{aligned}$$

Again resolving the two clauses with the mgu $\{a/y_2, a/y_1\}$ yields the empty clause.

Example 42.6 Consider the following argument.

- ϕ_1 . For every set x there is a set y whose cardinality is greater than the cardinality of x .
- ϕ_2 . If x is contained in y , the cardinality of x is no more than the cardinality of y .
- ϕ_3 . Every set is contained in the universe.

Therefore

- ϕ_4 . The universe is not a set.

1. Define a minimal signature Σ to which the statements of the above argument apply.

$$\Sigma = \{\textcolor{violet}{U}; \textcolor{green}{s} : s; >_c, \subseteq : s^2\}$$

where $\textcolor{violet}{U}$ is a constant (representing the universe), $\textcolor{green}{s}$ is a unary predicate which is true whenever its argument is a set and $>_c$ is a binary predicate which is true whenever its first argument is a set whose cardinality is greater than the second argument which is also a set and \subseteq is the usual containment relation on sets.

2. Provide an intuitively faithful translation of the sentences in the above argument.

$$\phi_1 \equiv \forall x[s(x) \rightarrow \exists y[s(y) \wedge y >_c x]]$$

$$\phi_2 \equiv \forall x, y[s(x) \rightarrow s(y) \rightarrow x \subseteq y \rightarrow \neg(x >_c y)]$$

$$\phi_3 \equiv \forall x[s(x) \rightarrow x \subseteq U]$$

$$\phi_4 \equiv \neg s(U)$$

3. Prove the above argument using rule $\exists E$.

Let $\Gamma = \{\phi_1, \phi_2, \phi_3\}$. We prove the conclusion by assuming its negation and arriving at a contradiction. We have split the proof into several trees since the tree is too wide. $(\forall E)^2$ indicates two consecutive applications of $\forall E$.

Proof tree \mathcal{T}_1 .

$$\text{MP} \frac{\Gamma, s(U) \vdash s(U)}{\text{MP} \frac{\forall E \frac{}{\Gamma, s(U) \vdash \phi_1 \rightarrow (s(U) \rightarrow \exists y[s(y) \wedge y >_c U])}{\Gamma, s(U) \vdash \phi_1}}{\Gamma, s(U) \vdash s(U) \rightarrow \exists y[s(y) \wedge y >_c U]}}$$
$$\exists E \frac{\Gamma, s(U) \vdash \exists y[s(y) \wedge y >_c U]}{\text{MP} \frac{\Gamma, s(U) \vdash s(a) \wedge a >_c U}{\wedge E \frac{}{\Gamma, s(U) \vdash s(a), a >_c U}}}$$

Proof tree \mathcal{T}_2 .

$$\frac{\text{MP} \frac{\text{MP} \frac{\frac{(\forall E)^2 \frac{\Gamma, s(U) \vdash \phi_2}{\Gamma, s(U) \vdash s(a) \rightarrow s(U) \rightarrow a \subseteq U \rightarrow \neg(a >_c U)}}{\Gamma, s(U) \vdash s(U) \rightarrow a \subseteq U \rightarrow \neg(a >_c U)}}{\Gamma, s(U) \vdash a \subseteq U \rightarrow \neg(a >_c U)}}{\Gamma, s(U) \vdash s(U)}$$

Proof tree \mathcal{T}_3 .

$$\frac{\text{MP} \frac{\text{MP} \frac{\frac{\forall E \frac{\Gamma, s(U) \vdash \phi_3}{\Gamma, s(U) \vdash s(a) \rightarrow a \subseteq U}}{\Gamma, s(U) \vdash a \subseteq U}}{\Gamma, s(U) \vdash a \subseteq U \rightarrow \neg(a >_c U)}}{\Gamma, s(U) \vdash \neg(a >_c U)}}$$

Proof tree \mathcal{T}_4 .

$$\begin{array}{c}
 \wedge \vdash \frac{\Gamma, s(\textcolor{violet}{U}) \vdash \textcolor{red}{a} >_c \textcolor{violet}{U} \quad \Gamma, s(\textcolor{violet}{U}) \vdash \neg(\textcolor{red}{a} >_c \textcolor{violet}{U})}{\perp \vdash \frac{\Gamma, s(\textcolor{violet}{U}) \vdash \textcolor{red}{a} >_c \textcolor{violet}{U} \wedge \neg(\textcolor{red}{a} >_c \textcolor{violet}{U})}{\neg \vdash \frac{\Gamma, s(\textcolor{violet}{U}) \vdash \perp}{\Gamma \vdash \neg s(\textcolor{violet}{U})}}}
 \end{array}$$

4. Prove the above argument without using the rule $\exists E$.

We follow the steps in the proof of Theorem 24.3 (Existential-Elimination Elimination). We first introduce an assumption which replaces all occurrences of a by a variable z that does not occur anywhere in the argument. Let $\phi_5 \stackrel{df}{=} s(z) \wedge z >_c U$. Let $\Delta = \Gamma, s(U), \phi_5$. We then have the following proof tree.

Proof tree \mathcal{T}'_1 :

$$\text{MP} \frac{\Gamma, s(U) \vdash s(U)}{\Gamma, s(U) \vdash \exists y[s(y) \wedge y >_c U]} \quad \text{MP} \frac{\forall E \frac{\Gamma, s(U) \vdash \phi_1 \rightarrow (s(U) \rightarrow \exists y[s(y) \wedge y >_c U])}{\Gamma, s(U) \vdash \phi_1}}{\Gamma, s(U) \vdash s(U) \rightarrow \exists y[s(y) \wedge y >_c U]}$$

Proof tree \mathcal{T}'_2 :

$$\text{MP} \frac{(\forall E)^2 \frac{\Delta \vdash \phi_2}{\Delta \vdash s(z) \rightarrow s(U) \rightarrow z \subseteq U \rightarrow \neg(z >_c U)} \quad \Delta \vdash s(z)}{\text{MP} \frac{\Delta \vdash s(U) \rightarrow z \subseteq U \rightarrow \neg(z >_c U)}{\Delta \vdash z \subseteq U \rightarrow \neg(z >_c U)}} \quad \Delta \vdash s(U)$$

Proof tree \mathcal{T}'_3 .

$$\frac{\text{MP} \frac{\frac{\forall E \frac{\Delta \vdash \phi_3}{\Delta \vdash s(z) \rightarrow z \subseteq U}}{\Delta \vdash s(z)}}{\text{MP} \frac{\Delta \vdash z \subseteq U}{\Delta \vdash z \subseteq U \rightarrow \neg(z >_c U)}}}{\Delta \vdash \neg(z >_c U)}$$

Proof tree \mathcal{T}'_4 .

$$\begin{array}{c} \wedge I \frac{\Delta \vdash z >_c U \quad \Delta \vdash \neg(z >_c U)}{\perp I \frac{\Delta \vdash \perp}{\Delta \vdash z >_c U \wedge \neg(z >_c U)}} \\ \text{DT} \Rightarrow \frac{}{\forall I \frac{\Gamma, s(U) \vdash (s(z) \wedge z >_c U) \rightarrow \perp}{\Gamma, s(U) \vdash \forall y[(s(y) \wedge y >_c U) \rightarrow \perp]}} \\ \text{exercise 23.2(d)} \frac{\text{MP} \frac{\Gamma, s(U) \vdash \exists y[s(y) \wedge y >_c U] \rightarrow \perp}{\neg I \frac{\Gamma, s(U) \vdash \perp}{\Gamma \vdash \neg s(U)}}}{\mathcal{T}'_1 \frac{\Gamma, s(U) \vdash \exists y[s(y) \wedge y >_c U]}{\Gamma, s(U) \vdash \perp}} \end{array}$$

Example 42.7 Consider the first order theory of directed graphs. Let path be a binary predicate which refers to the transitive-closure of the edge relation on directed graphs. Prove that path is not definable in First-order logic.

Consider the class of directed graphs with an infinite number of nodes and an infinite number of edges. The transitive closure of the edge relation is defined as $e^+ = \bigcup_{n>0} e^n$ where $e^1 = e$ and $e^{n+1} = \{(a, b) \mid \text{for some } c, e(a, c) \text{ and } e^n(c, b)\}$. Let $\text{path}(x, y)$ be false for a given pair of nodes x and y . The falsehood of $\text{path}(x, y)$ cannot be established by any finite means in first-order logic since it requires the establishment of the (many-sorted) predicate $\forall n[\neg e^n(x, y)]$ which in the first order theory of directed graphs requires the negation of an infinite number of predicates of the form $\neg e^n(x, y)$, one for each positive integer n .

The above argument holds also for the first-order theory of finite directed graphs where there is no a priori bound on the number of nodes.

However for the class of finite directed graphs with at most N nodes for a given number N the notion of path reduces to the finite relation $\bigcup_{1 \leq n \leq N} e^n$ and then the predicate $\text{path}(x, y)$ becomes finitely refutable.

Example 42.8 Prove the following derived rules of inference in the system \mathcal{H}_0 which show that for every direct proof there is a corresponding proof by contradiction and vice-versa.

$$\text{I2D. } \frac{\Gamma, \neg\phi \vdash \neg(\psi \rightarrow \psi)}{\Gamma \vdash \phi}$$

$$\text{D2I. } \frac{\Gamma \vdash \phi}{\Gamma, \neg\phi \vdash \neg(\psi \rightarrow \psi)}$$

You are not allowed to use any of the rules of \mathcal{G}_0 , but you may use any of the derived rules of \mathcal{H}_0 (including those in the exercises preceding natural deduction) in the lectures.

1. I2D. Assume there exists a proof tree $\Gamma, \neg\phi \vdash \neg(\psi \rightarrow \psi)$. We extend it as follows.

$$\text{DT} \Rightarrow \frac{\text{MP} \frac{\Gamma, \neg\phi \vdash \neg(\psi \rightarrow \psi)}{\Gamma \vdash \neg\phi \rightarrow \neg(\psi \rightarrow \psi)}}{\text{N'} \frac{\text{MP} \frac{\Gamma \vdash (\psi \rightarrow \psi) \rightarrow \phi}{\Gamma \vdash \phi}}{\text{R} \rightarrow \frac{}{\Gamma \vdash \psi}}} \quad \text{R} \rightarrow \frac{}{\Gamma \vdash \psi}$$

2. D2I. Suppose there is a proof tree $\Gamma \vdash \phi$. Then we extend it as follows.

$$\begin{array}{c}
 \nwarrow \mathcal{T} \nearrow \\
 \Gamma \vdash \phi
 \end{array}
 \quad
 \text{R} \rightarrow \frac{}{\Gamma \vdash \psi \rightarrow \psi} \quad
 \text{K} \frac{\Gamma \vdash \phi \rightarrow ((\psi \rightarrow \psi) \rightarrow \phi)}{\Gamma \vdash (\psi \rightarrow \psi) \rightarrow \phi} \quad
 \text{N''} \frac{\Gamma \vdash ((\psi \rightarrow \psi) \rightarrow \phi) \rightarrow (\neg \phi \rightarrow \neg(\psi \rightarrow \psi))}{\Gamma \vdash \neg \phi \rightarrow \neg(\psi \rightarrow \psi)}$$

$$\text{MP} \frac{\Gamma \vdash \phi}{\Gamma \vdash \psi \rightarrow \psi} \quad
 \text{MP} \frac{\Gamma \vdash (\psi \rightarrow \psi) \rightarrow \phi}{\Gamma \vdash \neg \phi \rightarrow \neg(\psi \rightarrow \psi)} \quad
 \text{DT} \Leftarrow \frac{\Gamma \vdash \neg \phi \rightarrow \neg(\psi \rightarrow \psi)}{\Gamma, \neg \phi \vdash \neg(\psi \rightarrow \psi)}$$

Example 42.9 In the proof system \mathcal{H}_1 , prove the following theorem

$$\exists x[\forall y[\phi]] \rightarrow \forall y[\exists x[\phi]]$$

1. using the rule $\exists E$ and
2. without using the rule $\exists E$

$$1. \frac{\forall E}{\frac{\exists E}{\frac{\exists x \forall y[\phi] \vdash \exists x \forall y[\phi]}{\frac{\exists x \forall y[\phi] \vdash \forall y[\{a/x\}\phi]}{\frac{\exists E}{\frac{\exists x \forall y[\phi] \vdash \{a/x\}\phi}{\frac{\exists I}{\frac{\exists x \forall y[\phi] \vdash \exists x[\phi]}{\frac{\forall I}{\exists x \forall y[\phi] \vdash \forall y \exists x[\phi]}}}}}}}}$$

2. Let z be a new variable that does not anywhere in the formulae.

$$\begin{array}{c}
 \forall E \frac{\exists x \forall y[\phi], \forall y[\{z/x\}\phi] \vdash \forall y[\{z/x\}\phi]}{\exists I \frac{\exists x \forall y[\phi], \forall y[\{z/x\}\phi] \vdash \{z/x\}\phi}{\text{DT} \Rightarrow \frac{\exists x \forall y[\phi], \forall y[\{z/x\}\phi] \vdash \exists x[\phi]}{\forall I \frac{\exists x \forall y[\phi] \vdash \forall y[\{z/x\}\phi] \rightarrow \exists x[\phi]}{\text{exercise 23.1.2(d)} \frac{\text{MP} \frac{\exists x \forall y[\phi] \vdash \exists x[\forall y[\phi] \rightarrow \exists x[\phi]] \quad \exists x \forall y[\phi] \vdash \exists x \forall y[\phi]}{\forall I \frac{\exists x \forall y[\phi] \vdash \exists x[\phi]}{\text{DT} \Rightarrow \frac{\exists x \forall y[\phi] \vdash \forall y \exists x[\phi]}{\vdash \exists x \forall y[\phi] \rightarrow \forall y \exists x[\phi]}}}}}}}}}}}$$

Example 42.10 Assuming the following statements.

1. Everyone admires a hero.
2. Anyone who is not a hero is a failure.
3. A failure admires everyone.

Use first-order resolution to determine whether

1. there are two different persons who admire each other,
2. there is a person who admires himself or herself,
3. there is a hero who admires a failure.

We begin by translating the statements (both hypotheses and conclusions) into first-order logic sen-

tences and then negating the possible conclusions.

$$\begin{array}{ll}
 \phi_1 \stackrel{df}{=} \forall x, y [h(x) \rightarrow a(y, x)] & \phi_1 \Leftrightarrow \forall x, y [\neg h(x) \vee a(y, x)] \\
 \phi_2 \stackrel{df}{=} \forall x [\neg h(x) \rightarrow f(x)] & \phi_2 \Leftrightarrow \forall x [h(x) \vee f(x)] \\
 \phi_3 \stackrel{df}{=} \forall x [f(x) \rightarrow \forall y [a(x, y)]] & \phi_3 \Leftrightarrow \forall x, y [\neg f(x) \vee a(x, y)] \\
 \psi_1 \stackrel{df}{=} \exists x, y [\neg(x = y) \wedge a(x, y) \wedge a(y, x)] & \neg\psi_1 \Leftrightarrow \forall x, y [x = y \vee \neg a(x, y) \vee \neg a(y, x)] \\
 \psi_2 \stackrel{df}{=} \exists x [a(x, x)] & \neg\psi_2 \Leftrightarrow \forall x [\neg a(x, x)] \\
 \psi_3 \stackrel{df}{=} \exists x, y [h(x) \wedge f(y) \wedge a(x, y)] & \neg\psi_3 \Leftrightarrow \forall x, y [\neg h(x) \vee \neg f(y) \vee \neg a(x, y)]
 \end{array}$$

After transforming them into PCNF, Skolemization and standardizing the variables apart we get the following sets of clauses for each part.

$$1. \{\phi_1, \phi_2, \phi_3, \neg\psi_1, \} \equiv \{\{\neg h(x_1), a(y_1, x_1)\}, \{h(x_2), f(x_2)\}, \{\neg f(x_3), a(x_3, y_3)\}, \{x_4 = y_4, \neg a(x_4, y_4), \neg a(y_4, x_4)\}\}$$

$$\frac{\{\{\neg h(x_1), a(y_1, x_1)\}, \{h(x_2), f(x_2)\}, \{\neg f(x_3), a(x_3, y_3)\}, \{x_4 = y_4, \neg a(x_4, y_4), \neg a(y_4, x_4)\}\}}{\{\{a(y_1, x_2), f(x_2)\}, \{\neg f(x_3), a(x_3, y_3)\}, \{x_4 = y_4, \neg a(x_4, y_4), \neg a(y_4, x_4)\}\}} \frac{x_2/x_1}{x_3/x_2} \\
 \frac{\{\{a(y_1, x_3), a(x_3, y_3)\}, \{x_4 = y_4, \neg a(x_4, y_4), \neg a(y_4, x_4)\}\}}{\{\{x_4 = y_4\}\}} \frac{x_4/y_1, y_4/x_3, x_4/y_3}{}$$

Since the empty clause could not be derived, ψ_1 is not a logical consequence of the statements ϕ_1 , ϕ_2 and ϕ_3 .

2. Now consider the set $\{\phi_1, \phi_2, \phi_3, \neg\psi_2, \}\equiv\{\{\neg h(x_1), a(y_1, x_1)\}, \{h(x_2), f(x_2)\}, \{\neg f(x_3), a(x_3, y_3)\}, \{\neg a(x_5, x_5)\}\}$
 The proof here remains the same as in the previous part except in the last steps. So we write out only the last steps.

$$\frac{\{\{a(y_1, x_3), a(x_3, y_3)\}, \{\neg a(x_5, x_5)\}\}}{\{\{\}\}} \{x_5/y_1, x_5/x_3, x_5/y_3\}$$

ψ_2 is therefore a logical consequence of ϕ_1 , ϕ_2 and ϕ_3 .

3. The set $\{\phi_1, \phi_2, \phi_3, \psi_3, \}\equiv\{\{\neg h(x_1), a(y_1, x_1)\}, \{h(x_2), f(x_2)\}, \{\neg f(x_3), a(x_3, y_3)\}, \{\neg h(x_6), \neg f(y_6), \neg a(x_6, y_6)\}\}$
 requires a similar treatment and yields

$$\frac{\{\{a(y_1, x_3), a(x_3, y_3)\}, \{\neg h(x_6), \neg f(y_6), \neg a(x_6, y_6)\}\}}{\{a(y_1, x_6), \neg h(x_6), \neg f(y_6)\}} \{x_6/x_3, y_6/y_3\}$$

Hence ψ_3 is not a logical consequence of ϕ_1 , ϕ_2 and ϕ_3 .

Example 42.11 Inadequacy of $\{\neg, \leftrightarrow\}$

It is relatively easy to show that any set of operators (such as $\{\wedge, \vee\}$) (exercise 5.1. 2) which does not explicitly or implicitly include the power of negation is going to be expressively inadequate for propositional logic. In particular, sets of operators like $\{\rightarrow, \neg\}$ and $\{\uparrow\}$ do include the power of negation and hence are expressively adequate. We prove a harder theorem here for a set which includes negation but is still inadequate.

Theorem 42.12 (Inadequacy of $\{\neg, \leftrightarrow\}$) *The set $\{\neg, \leftrightarrow\}$ is expressively inadequate for propositional logic. In particular, the operators \wedge and \vee cannot be expressed using only \neg and \leftrightarrow .*

By the 1-1 correspondence between the operators of \mathcal{P}_0 and the algebra $\mathbf{2}'$ it suffices to prove the following theorem.

Theorem 42.13 (Inadequacy of $\{\bar{,} \dot{=}\}$) *The set $\{\bar{,} \dot{=}\}$ is expressively inadequate for the algebra $\mathbf{2}'$. In particular, the operators $.$ and $+$ cannot be expressed using only $\bar{}$ and $\dot{=}$.*

Proof: Notice that in the **truth table** definition of the binary operators both the columns for $a.b$ and $a + b$ have an odd number of occurrences of 1. Hence if we can show that every expression made up

of only the variables a , b and operators \neg and \doteq always yields truth tables with an even number of occurrences of 1 it would imply that $a \cdot b$ and $a + b$ cannot be expressed using only operators \neg and \doteq . We prove this in the following lemma (lemma 42.14). QED

Lemma 42.14 Consider the subset of boolean expressions generated by the BNF

$$e, f ::= a \mid b \mid \bar{e} \mid (e \doteq f)$$

The truth table of every boolean expression which has occurrences of both a and b has an even number of occurrences of 1 in the last column.

Proof: Let $e(a, b)$ denote an arbitrary expression in this language containing both variables. Every such boolean expression has a truth table of exactly 4 rows. Let $\#0e$ and $\#1e$ denote respectively the number of occurrences of 0 and 1 in the truth table of e . In any such a truth table either both $\#0e$ and $\#1e$ are even or both are odd.

We prove this lemma by induction on the structure of expressions. Notice that the basis consists only of the boolean expressions a and b neither of which has occurrences of both variables and hence the claim holds vacuously. In fact, the smallest expression containing both variables is $a \doteq b$ whose truth table contains an even number of occurrences of 1 .

Assume the induction hypothesis that $\#1f(a, b)$ and $\#1g(a, b)$ are even. The induction step has the following cases.

Case $e(a, b) \equiv \overline{f(a, b)}$. By the induction hypothesis since both $\#0f$ and $\#1f$ are even it follows that both $\#0e$ and $\#1e$ are also even.

Case $e(a, b) \equiv (f \doteq g)$. Here we have the following sub-cases.

Sub-case $e(a, b) \equiv f(a) \doteq g(b)$. In this sub-case, $f(a)$ (regardless of how complicated its expression may be) has a truth table with only 2 rows corresponding to $a = 0$ and $a = 1$. Similarly with $g(b)$. In fact $f(a)$ and $g(b)$ being unary functions can represent only the four possible unary functions

$$\begin{aligned} 0 &\stackrel{df}{=} \lambda x[0], \\ 1 &\stackrel{df}{=} \lambda x[1], \\ I &\stackrel{df}{=} \lambda x[x], \\ N &\stackrel{df}{=} \lambda x[\bar{x}] \end{aligned}$$

Hence there are sixteen possibilities for the expression $e(a, b) \equiv (f(a) \doteq g(b))$ and the following matrix specifies the values of $\#1e$ for each possibility.

	$0(a)$	$1(a)$	$\mathbf{l}(a)$	$\mathbf{N}(a)$
$0(b)$	4	0	2	2
$1(b)$	0	4	2	2
$\mathbf{l}(b)$	2	2	2	2
$\mathbf{N}(b)$	2	2	2	2

In all the sixteen cases, $\#1e$ is even.

Sub-case $e(a, b) \equiv (f(a) \doteq g(b))$. This sub-case is analogous to the previous sub-case with the previous matrix being transposed to obtain the matrix for this sub-case.

Sub-case $e(a, b) \equiv (f(a, b) \doteq g(a, b))$. By the induction hypothesis we have that $\#1f \in \{0, 2, 4\}$ and $\#1g \in \{0, 2, 4\}$. The following matrix (with the centre square being left blank) gives the values of $\#1e$ in eight of the nine cases.

	0	2	4
0	4	2	0
2	2		2
4	0	2	4

In all of the eight cases we see that $\#1e$ is even. The only case that has been left out is the case when $\#1f = 2 = \#1g$. Here again we do a case analysis as follows:

Sub-sub-case [Same rows]. The two 1s of $f(a, b)$ occur in the same rows as the two 1s of $g(a, b)$. Then clearly $\#1e = 4$ which is even.

Sub-sub-case [Disjoint rows]. The two 1s of $f(a, b)$ occur in rows that are disjoint from the rows in which the two 1s of $g(a, b)$ occur. Then $\#1e = 0$ which is also even.

Sub-sub-case [Neither of the above]. Then there is exactly one row in which both f and g are 1 and exactly one row in which they are both 0. In the other two rows both f and g have different values. Then clearly $\#1e = 2$ which is again even. *QED*

References

- [1] A. R. Bradley and Z. Manna. *The Calculus of Computation*. Springer-Verlag, Berlin, Germany, 2007.
- [2] I. M. Copi. *Symbolic Logic*. Macmillan, London, UK, 1979.
- [3] H. D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, New York, USA, 1994.
- [4] H. B. Enderton. *A Mathematical Introduction to Logic*. Elsevier India, New Delhi, India, 2001.
- [5] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, USA, 1990.
- [6] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, UK, 2000.
- [7] John Kelly. *The Essence of Logic*. Prentice-Hall India, New Delhi, India, 1997.
- [8] S. C. Kleene. *Mathematical Logic*. Dover Publications Inc., New York, USA, 1967.
- [9] Z. Manna and R. Waldinger. *The Logical Basis for Computer Programming, Vol. I*. Addison-Wesley Publishing Company, U. S. A., 1985.
- [10] Z. Manna and R. Waldinger. *The Logical Basis for Computer Programming, Vol. II*. Addison-Wesley Publishing Company, U. S. A., 1990.
- [11] E. Mendelson. *Introduction to Mathematical Logic*. D. Van Nostrand Co. Inc., Princeton, New Jersey, USA, 1963.
- [12] Anil Nerode and R. Shore. *Logic for Applications*. Springer-Verlag, New York, USA, 1993.
- [13] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, Germany, 1968.
- [14] V. Sperschneider and G. Antoniou. *Logic: A Foundation for Computer Science*. Addison-Wesley Publishing Company, Reading, UK, 1991.

Thank You!

Any Questions?