Learning Goals of CS245 Logic and Computation

Alice Gao

April 27, 2018

Contents

1 Propositional Logic

Introduction to logic

- Give a one-sentence high-level definition of logic.

- Give examples of applications of logic in computer science.

Propositions

- Define a proposition.

- Define an atomic proposition and a compound proposition.

Translations

- Determine if an English sentence is a proposition.

- Determine if an English sentence is an atomic proposition.

- For an English sentence with no logical ambiguity, translate the sentence into a propositional formula.

- For an English sentence with logical ambiguity, translate the sentence into multiple propositional formulas and show that the propositional formulas are not logically equivalent using a truth table.

Well-formed formulas

- Describe the three types of symbols in propositional logic.

- Give the inductive definition of well-formed formulas.

- Determine and justify whether a given formula is well formed.

- Write the parse tree for a well-formed formula.

Structural induction

- Prove properties of well-formed propositional formulas using structural induction.

- Prove properties of a recursively defined concept using structural induction.

Truth valuation, truth table, and valuation tree

- Define a truth valuation.

- Determine the truth value of a formula given a truth valuation.

- Give a truth valuation under which a formula is true or false.

- Draw a truth table given a formula.

- Draw the valuation tree given a formula.

The meanings of connectives

- Define the meaning of the connectives: negation, conjunction, disjunction, conditional, and bi-conditional, using truth tables.

- Understand the subtleties of the implication: Explain the relationship between the truth values of the premise, the conclusion, and the implication.

Properties of formulas

- Define tautology, contradiction, and satisfiable formula.

- Determine if a given formula is a tautology, a contradiction, and/or a satisfiable formula.

Logical equivalence

- Prove that two formulas are logically equivalent using truth tables.

- Prove that two formulas are logically equivalent using logical identities.

- Translate a condition in a block of code into a propositional formula. Simplify an if statement. Determine whether a piece of code is live or dead.

Adequate set of connectives

- Describe what it means for one connective to be definable in terms of a set of connectives.

- Describe what it means for a set of connectives to be adequate for propositional logic.

- Prove that a set of connectives is adequate.

- Prove that a set of connectives is not adequate.

Semantic entailment

- Determine if a set of formulas is satisfiable.

- Define semantic entailment.

- Explain subtleties of semantic entailment.

- Prove that a semantic entailment holds/does not hold by using the definition of semantic entailment, and/or truth tables.

Natural deduction

- Describe rules of inference for natural deduction.

- Prove that a conclusion follows from a set of premises using natural deduction inference rules.

Soundness and completeness of natural deduction

- Define soundness and completeness.

- Prove that a semantic entailment holds using natural deduction and the soundness of natural deduction.

- Show that no natural deduction proof exists using the contrapositive of the soundness of natural deduction.

2 Predicate Logic

Introduction to Predicate Logic

- Give examples of English sentences that can be modeled using predicate logic but cannot be modeled using propositional logic.

Translations

- Translate an English sentence into a predicate formula.

Syntax of predicate logic

- Define term.

- Define formula.

- Define free and bound variables.

- Determine whether a variable in a formula is free or bound.

- Determine the scope of a quantifier in a formula.

- Describe the problem when a variable is captured in a substitution.

- Perform substitution in a formula to avoid capture.

Semantics of predicate logic

- Define interpretation.

- Define environment.

- Determine the truth value of a formula given an interpretation and an environment.

- Give an interpretation and an environment that make a formula true.

- Given an interpretation and an environment that make a formula false.

- Determine and justify whether a formula is valid, satisfiable, and/or unsatisfiable.

Semantic entailment for predicate logic

- Define semantic entailment for predicate logic.

- Prove that a semantic entailment holds.

- Prove that a semantic entailment does not hold.

Natural deduction for predicate logic

- Describe the rules of inference for natural deduction.

- Prove that a conclusion follows from a set of premises using natural deduction inference rules.

Soundness and completeness of natural deduction

- Define soundness and completeness.

- Prove that a semantic entailment holds using natural deduction and the soundness of natural deduction.

- Show that no natural deduction proof exists using semantic entailment and the soundness natural deduction.

Axioms of equality

- Describe the two axioms of equality.

- Prove the symmetry of equality.

- Prove the transitivity of equality.

- Describe the derived inference rules EQsubs and EQtrans.

3 Program Verification

Introduction

- Give reasons for performing formal verification rather than testing.

- Define a Hoare triple.

- Define partial correctness.

- Define total correctness.

Total correctness for assignment and conditional statements

- Prove that a Hoare triple is satisfied under total correctness for a program containing assignment and conditional statements.

Partial correctness for while loops

- Determine whether a given formula is an invariant for a while loop.

- Find an invariant for a given while loop.

- Prove that a Hoare triple is satisfied under partial correctness for a program containing while loops.

Total correctness for while loops

- Determine whether a given formula is a variant for a while loop.

- Find a variant for a given while loop.

- Prove that a Hoare triple is satisfied under total correctness for a program containing while loops.

Partial correctness for array assignments

- Prove that a Hoare triple is satisfied under partial correctness for a program containing array assignment statements.

4 Undecidability

Introduction to undecidability

- Define decision problem.

- Define decidable problem.

- Define undecidable problem.

- Prove that a decision problem is decidable by giving an algorithm to solve it.

The halting problem

- Describe the halting problem.

- Prove that the halting problem is undecidable.

Proving that a problem is undecidable by a reduction from the halting problem

- Define reduction.

- Describe at a high level how we can use reduction to prove that a decision problem is undecidable.

- Prove that a decision problem is undecidable by using a reduction from the halting problem.