Course Proposal

Instructor(s): Jean H. Gallier, Rajeev Alur, Val Tannen

Prerequisites: CSE260

Course Description: This is an introduction to logic and its applications to solving problems in computer science. While teaching logic in a rigorous way, the course will also demonstrate that logical systems are helpful in formalizing correctness of software systems as well as for encoding computational problems. Every logic comprises of three components: a language to make logical assertions, a precise method to interpret the meaning or truth of these assertions, and a proof system or a decision algorithm to prove or disprove the truth of assertions. These issues will be discussed for three different logics: Propositional logic, first-order logic and temporal logic. Sample topics include soundness and completeness of proof systems, efficient search strategies for proofs, resolution, efficient algorithms for satisfiability of propositional logic, logic programming, and model checking for temporal logics. The projects will focus on the importance of these techniques in solving problems in circuit design, specification and analysis of protocols, and query evaluation in databases.


Additional recommended text: Logic in Computer Science: Modelling and reasoning about systems. Huth and Ryan, Cambridge University Press.

Brief description:

(1) Propositional logic
   
   (a) Syntax and Semantics
   
   (b) Proof theory
      
      i. Gentzen systems, sequent calculi and natural deduction
      
      ii. Soundness of the Sequent calculus $G_0$
      
      iii. Completeness of the Sequent calculus $G_0$
   
   (c) Satisfiability and NP-completeness
   
   (d) Encoding optimization problems in SAT

(2) First-order logic
(a) Syntax and Semantics
(b) Quantifiers, first-order structures
(c) Proof Theory
  i. Gentzen systems, sequent calculi and natural deduction
  ii. Soundness of the Sequent calculus $G$ (statement only, no proof)
  iii. Hintikka sets
  iv. Completeness of the Sequent calculus $G$ (statement only, no proof)
(d) First-order logic with equality

(3) The Resolution Method
(a) Clauses
(b) Propositional resolution
(c) Completeness of resolution
(d) Modern resolution-based SAT solvers and implementation issues
(e) Unification a la Robinson
(f) First-order resolution
(g) Refutational completeness
(h) Horn clauses
(i) SLD-resolution
(j) Logic programming
(k) Query evaluation using logic programming

(4) Temporal logic
(a) Syntax and Semantics
(b) Model checking
(c) Binary decision diagrams and efficient implementations
(d) Specification and verification of concurrent systems

The final grade in the course grade will be determined based on homework assignments (4), programming projects (2), and final exam. Sample projects are:

1. Implement a resolution-based satisfiability checker for propositional logic

2. Write a program to solve an NP-hard optimization problem (e.g. scheduling of airline flights) by translating into propositional satisfiability, and using a modern SAT solver (e.g. CHAFF)

3. Specify and verify a distributed algorithm using a model checker (e.g. SPIN).