Chapter 2, Adam P. and Josh F.

Theories

- What is a Theory?
  - Uses First Order Logic(Variables, Logical symbols, Nonlogical symbols, Syntax) as a generic syntactic framework
  - Each theory has its own restriction on Nonlogical symbols
  - INTERPRETATION of Nonlogical symbols is also important
  - Most of the theories we will see in this book are “quantifier-free”, and the logical axioms(restrictions on the interpretation of logical symbols) are built-in: common to all FO theories

Theory 1: Propositional Logic

- Simple syntax: Or, Not, (), atom
- Three atoms: Boolean-identifiers, TRUE, FALSE
- Many applications: database queries, planning problems in AI, automated reasoning, circuit design, etc.

Motivation

- *n* radio stations
- *k* transmission frequencies, $k < n$
- Two stations that are too close together cannot have the same frequency(call this set *E*)

Motivation

- Every station is assigned at least one frequency
- Every station is assigned not more than one frequency
- Close stations are not assigned the same frequency

Motivation - Discussion

- **Example** Consider three persons, *A*, *B*, and *C* who must be seated in a row, with the following constraints: *A* won't sit next to *C*, *A* won't sit it the left chair and *B* won't sit to the right of *C*. Write a propositional formula that is satisfiable *iff* there is a seat assignment which satisfies all constraints.
- **Example 2** Given the following programs, show they are equivalent:
  $!(a||b)?h :!(a == b)?f : g$
  $!(!a||!b)?g : (!a\&\&!b)?h : f$

SAT Solvers

- Given a Boolean formula β, a SAT solver decides whether β is satisfiable. If so, reports satisfying assignment
- REMEMBER: For this chapter, all inputs are in CNF
- Vast improvement of SAT solvers in recent years: learn from wrong assignments, prune large search spaces quickly, and focusing on ”important” variables first

SAT Solvers

- DPLL framework
  - Traversing and backtracking on a binary tree
  - Internal nodes represent partial assignments, leaves represent full assignments
  - Complete(terminates AND returns “Valid” when input formula is valid)
- Stochastic search framework
  - Solver guesses a full assignment
  - If the formula is evaluated to FALSE under this assignment, starts to flip values of variables according to some (greedy) heuristic
  - Most are incomplete

DPLL-Definitions

- **decision-** assign a value to a variable
- **decision level-** depth in the binary decision tree in which a decision is made, starting from 1.
- **ground level-** decision level 0; clauses with a single literal
- **satisfied clause-** one or more of its literals are satisfied
- **conflicting clause-** all of its literals are assigned but not satisfied
- **unit clause-** not satisfied and all but one of its literals are assigned
- **unresolved clause-** otherwise

DPLL-Definitions

- **unit clause rule-** Given a partial assignment under which a clause becomes unit, it **must** be extended so that it satisfies the unassigned literal
- For a given unit clause C with an unassigned literal l, we say that "C implies l" and that C is the antecedent clause of l, denoted by $Antecedent(l)$.
- If more than one unit clause implies l, we refer to the clause that the **SAT solver used** in order to imply l as its antecedent
- High level diagrams on pg. 30

BCP

- Repeated application of **unit clause rule** until either a conflict or no more implications possible
- Best visualized(and modeled) with an **implication graph**
- A **partial implication graph** is a subgraph which illustrates the BCP at a specific decision level

BCP

- After reaching a **conflict node** k, the ANALYZE-CONFLICT function chooses a “smart” **conflict clause** to add to the list of formula constraints
- Most competitive solvers design ANALYZE-CONFLICT to generate **asserting clauses** only.
- ANALYZE-CONFLICT also choses the decision level to backtrack to. According to **conflict-driven backtracking** strategy, choose the **second most recent decision level in the conflict clause**

BCP

- Is this process guaranteed to terminate?
- Yes. It is never the case that the solver enters decision level DL again with the same partial assignment (See node x1 in figures on pg. 33)

Chapter 3, Adam P. and Paul K.

Theory 2: Theory of Equality(Equality Logic)

- Syntax similar to *Propositional Logic*
- **BUT** : atoms are equalities between variables(or between variables and constants) over some infinite domain

Propositional VS Equality Logic

- Both *Equality Logic* and *Propositional Logic* are NP-Complete: They can model the same decision problems
- Why study both?
    - Convenience of modeling: Some problems are more naturally expressed in Equality Logic
    - Efficiency: The high level structure of the input Equality Logic formula can potentially make the decision procedure realize shortcuts(This info may be lost if the problem is modeled directly in Propositional Logic)

Uninterpreted Functions

- **Uninterpreted Functions**: Function placeholders, used for abstracting, or generalizing, theorems.
  - Essentially, we are ignoring the semantics of the function (except for one general behavior: **functional consistency**)
  - Different from Interpreted Functions, which have interpretation-specific semantic properties
- **Functional consistency(congruence)**: Instances of the same function return the same value if given equal arguments.
- SHOW formal definition??(symbols)

General Scheme for using Uninterpreted Functions

- Assume a formula of interest has interpreted functions, and a validity check is too hard(computationally), or even impossible
- Assign an uninterpreted function to each interpreted function
- Check the validity of the new formula. If valid, then interpreted formula is valid(by thm. 3.3)

Aside: Removing Boolean Variables and Constants

- Method to remove Boolean Variables (in short): replace each variable with an equality between two new variables.
- Method to remove Constants (in short): replace each constant with a new variable, and add inequality constraints to the new formula such that distinct constants(now variables) cannot equal one another.

Problems with Interpreted Functions

- Large or infinite state spaces(multiplication of 32-bit integers, unbounded memory usage, etc.)
- Complicated semantics of interpreted functions can clutter(and slow down) the reasoning surrounding their use in formulas

Problems with **Un**interpreted functions

- Consider: $x_1 + y_1, x_2 + y_2$
- Assume $x_1 = y_2$ and $y_1 = x_2$
- Replace ‘+’ with F

Partially Interpreted Functions

- **partially interpreted functions** - Adding additional constraints that restore **some** semantic information about the original(interpreted) function
- $((x_1 = x_2 \wedge y_1 = y_2) \vee (x_1 = y_2 \wedge y_1 = x_2)) \implies f1 = f2$
- Multiplication argument of 0 $\implies$ result is 0

References

D. Kroening and O. Strichman, Decision Procedures: An Algorithmic Point of View, Springer, 2008. ISBN: 978-3-540-74104-6