SAT Solver - Variant 2

Progress Slides 2

Next Steps from Progress Slides 3

- Begin Coding!!!
- I want to see how DPLL works in practice, so I can start implementing and experimenting with heuristics and search tactics
- Get a basic sat solver near completed/completed

Current SAT Solver

- I have built a basic SAT solver
- Can handle small CNF formulas quickly
- Struggles with larger CNF formulas

Decision Heuristics

- Pure Literals
  - Makes Decisions on pure literals first
- Dynamic Largest Individual Sum
  - After pure literals, makes decision on literals will highest occurrence
- Unit Clauses
  - Still not fully functional, but am currently getting decisions for unit clauses to work

Current Design (Storage)

- Literals
  - A dictionary of literals with literal as key, and occurrence as the value
- Clauses
  - A list of clauses
- Assignment
  - A dictionary of literals with literal as key, and its boolean assignment as the value
- Clause_Status
  - An list of strings (“Unresolved, Resolved, Unit”)
  - Thinking about changing this to dictionary with clause as key for faster retrieval of data

Current Design (Algorithm/Functions)

- DPLL Class
  - **add_clause(clause)**
    - Adds clause and deals with literal occurrence
  - **dpll()**
    - Recursive dpll algorithm
  - **check_satisfiability()**
    - Checks if assignment is SAT, and edits clause_status for each clause when necessary
  - **choose_literal()**
    - Returns literal with next highest occurrence
  - **find_pure_literal()**
    - Returns a pure literal if not assignment and exists
  - **check_unit_clause()**
    - Checks of a clause is a unit clause based on current assignment
  - **find_unit_clause_literal()**
    - Returns a literal from first unit clause if exists, false otherwise

CNF DataSets

- I have found several benchmark DIMACS CNF formulas online that I am going to use to Benchmark my solver
- Currently, the solver is not very fast, so it struggles with some of these Benchmarks
- Tried using a python library (cnfgen), but can’t get it to work
- Is there any other places I can find smaller formulas to test my solver in its initial stages?
- Where I can find DIMACS CNF formulas for imbalance instances

Next Steps

- Keep Coding!
- Get Unit Propagation fully functional
- Start integrating other decision heuristics into my solver that are more tailored to my variant of unbalanced instances
- In the end, have a strong and efficient solver for large CNF formula benchmarks, tailored to my variant