Proof Complexity and Solving LAB Unit Propagation with Watched Literals

Dr. Joshua Blinkhorn

Friedrich-Schiller-Universität Jena https://github.com/JoshuaBlinkhorn/SAT-LAB

Goals

- Implemenatation of SAT solving algorithms
 - (a) 2-SAT (polynomial time)
 - (b) DPLL
 - (c) CDCL
 - watched literals
 - clause learning
 - decision heuristics
 - restart strategy
 - (d) QBF expansion..
- Practical programming experience
 - use your favourite language (Python, C, C++, Java, ..)
 - recommended: Python

Assignment Data Structure (An Apology)

- good scenario for storing the assignment:
 - store assignments in fixed location (fast access)
 - maintain a list of 'pointers' to them (the trail)
- decision level data can be stored in either

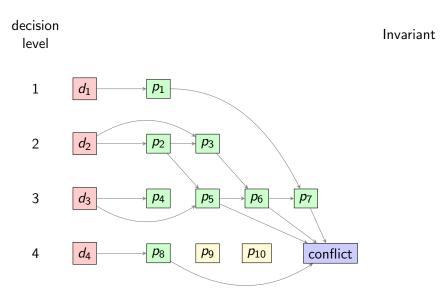
Watched Literals

- when searching for conflict, we only care about unit clauses
- a clause becomes unit when:
 - it has exactly one unassigned literal, and
 - all other literals are falsified
- sufficient to watch just two literals in every clause
- maintain this invariant for each clause:
 - either both watched literals are unassigned
 - or at least one watched literal is satisfied
- important: if both watched literals are assigned, and one is falsifed: its decision level should be no lower than the satisfied one

How is it done?

- many options here's one:
- (a) maintain a list of 'watched clauses' for each literal
- (b) process a variable assignment by:
 - 1. visit watched clauses in order
 - 2. make sure the invariant holds
 - you may need to 'swap the watch'
 - 3. if clause becomes unit, add unit assignment to trail
 - note: in this case, both watched literals have the same decision level
 - so there is no need to swap the watch
- (c) if the invariant cannot be maintained, we reach conflict

Conflicts and Backtracking



Watched Literals Task

- implement unit propagation with watched literals in your CDCL solver
- check correctness
- compare the solving time to naive propagation