# Proof Complexity and Solving LAB

## New Year Recap

Dr. Joshua Blinkhorn

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

# Goals

- Implemenatation of SAT solving algorithms

  (a) 2-SAT (polynomial time)

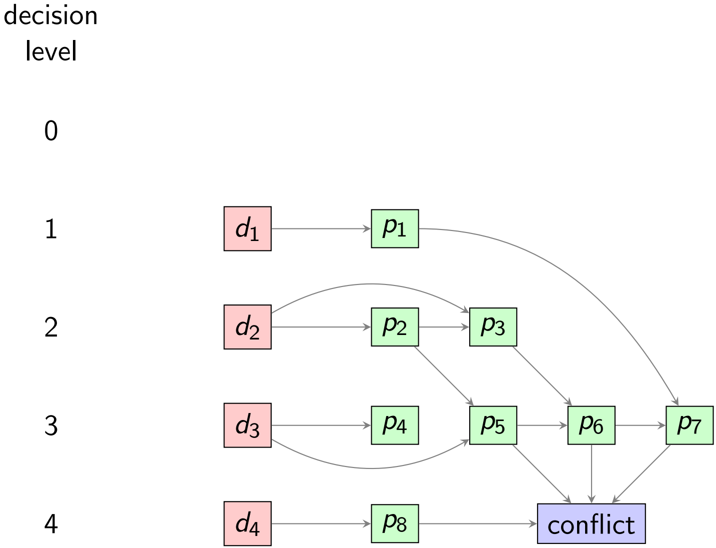
  (b) DPLL

  (c) CDCL
    - watched literals
    - clause learning
    - decision heuristics
    - restart strategy

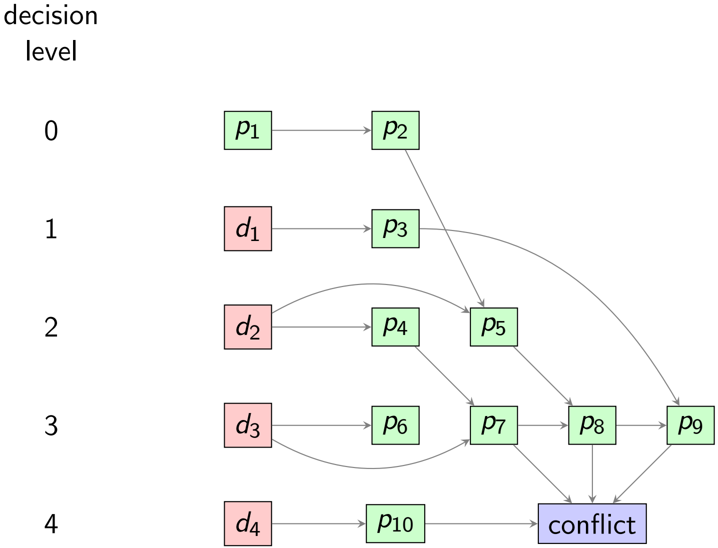
  (d) QBF expansion..

- Practical programming experience
  - use your favourite language (Python, C, C++, Java, ..)
  - recommended: Python

# Reaching the First Conflict



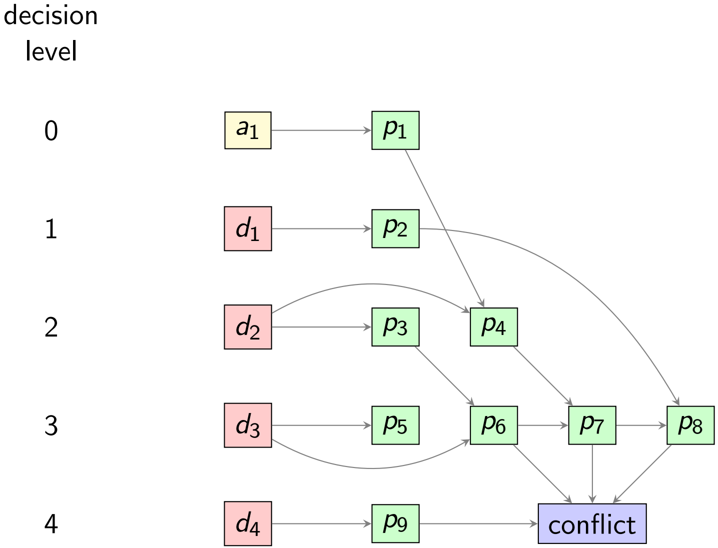
# Reaching Later Conflicts



# Learning Unit Clauses (!?)

- a learned unit clause should propagate:
  (a) immediately
  (b) forever
- result: adding learned unit clauses is unnecessary ..
- .. just make the assignment at decision level 0 instead
- decision level 0 is never undone
- watched literals do not work with unit clauses

# Reaching Later Conflicts (Revisited)



# CDCL Pseudocode

```
function CDCL-solver(Φ)                                  #assuming Φ is preprocessed
    decision-level ← 0
    while there are unassigned variables
        decision-level++
        decide()                                         #adds assignment to trail
        C_conflict ← propagate()                         #returns conflict clause or null

        while C_conflict is not null
            if decision-level = 0 return UNSAT
            C_learned ← analyse-conflict(C_conflict)
            if C_conflict is unit
                backtrack(0)
                assign unit literal
            else
                backtrack(asserting-level(C_learned))    #changes trail and DL
                Φ ← Φ ∧ C_learned
            C_conflict ← propagate()

        apply-restart-policy()
    return SAT
```

# 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:

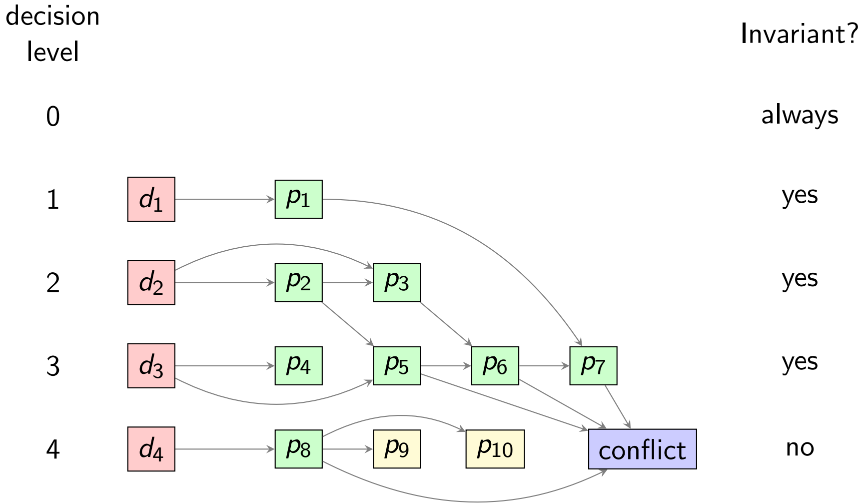
(a) maintain a list of 'watched clauses' for each literal

(b) process a variable assignment by:

    1. visit watched clauses for the falsified literal 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



# Swapping the Watch

$\boxed{x_1 \mapsto 0}$ $\boxed{x_2 \mapsto 1}$ $\boxed{x_3 \mapsto 0}$ $\boxed{x_4 \mapsto 1}$ $\boxed{x_5 \mapsto 1}$

trivial case: $(x_6 \vee \overline{x_7} \vee \overline{x_4}) \quad \to \quad (x_6 \vee \overline{x_7} \vee \overline{x_4})$

sat case: $(\overline{x_1} \vee \overline{x_4} \vee \overline{x_2}) \quad \to \quad (\overline{x_1} \vee \overline{x_4} \vee \overline{x_2})$

sat-swap case: $(\overline{x_4} \vee \overline{x_6} \vee x_3 \vee x_5) \quad \to \quad (\overline{x_4} \vee \overline{x_6} \vee x_3 \vee x_5)$

swap case: $(\overline{x_4} \vee \overline{x_6} \vee x_3 \vee \overline{x_4} \vee x_6) \quad \to \quad (\overline{x_4} \vee \overline{x_6} \vee x_3 \vee \overline{x_4} \vee x_6)$

# Unit Clauses and Conflicts

$\boxed{x_1 \mapsto 0}$ $\boxed{x_2 \mapsto 1}$ $\boxed{x_3 \mapsto 0}$ $\boxed{x_4 \mapsto 1}$ $\boxed{x_5 \mapsto 1}$

unit case: $(\overline{x_4} \vee \overline{x_6} \vee x_3 \vee \overline{x_4}) \quad \rightarrow \quad (\overline{x_4} \vee \overline{x_6} \vee x_3 \vee \overline{x_4})$

add assignment: $x_6 \mapsto 0$

conflict case: $(\overline{x_4} \vee \overline{x_5} \vee x_3) \quad \rightarrow \quad (\overline{x_4} \vee \overline{x_5} \vee x_3)$

add assignment: $x_5 \mapsto 0$

$\Longrightarrow$ CONFLICT

# Watched Literals Task

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

# Clause Learning - Cutting the Implication Graph