Non-Restarting SAT Solvers With Simple Preprocessing Can Efficiently Simulate Resolution
Paul Beame* Ashish Sabharwal*
*Computer Science and Engineering, University of Washington, Seattle, WA, USA
*Allen Institute for Artificial Intelligence, Seattle, WA, USA

How Powerful are Modern SAT Solvers?
- SAT = propositional satisfiability testing
  - Is F = (a or b) and (-a or c) and (b or c) satisfiable?
- Modern SAT solvers scale to 1M+ variables and 10M+ clauses
- What are the inherent strengths and weaknesses of these solvers?
- If they had perfect heuristics, what could they solve in polynomial time?

Formal SAT Solver Models
- PD11: Most faithful to real solvers
  - 1 asserting clause per conflict
  - Backtracking to asserting level
- Pool Resolution [VanGelder 05]
  - Learns multiple clauses per conflict
  - No notion of asserting clause/level
- RTL systems [Buss et al, ’08]
  - Similar concerns as pool resolution
  - Weakening
- CL- proof system [Beame et al, ’04]
  - Selectively ignores propagation
- CL proof system [Beame et al, ’04]
  - Non-asserting clause learning

A Proof Complexity Perspective
- Trace of SAT solvers = Series of steps in the Resolution Proof System
  - Resolution rule: (a or c) and (-a or D) entails (C or D)
  - Problem Complexity Lower bounds => Solver Runtime lower bounds
- What about the other way?
  - If F has a short proof, can a “smart enough” SAT solver find it?
  - I.e., are SAT solvers as powerful as Resolution?

Previously Known Results
1. CL : with many Restarts, can simulate Resolution [2004]
2. CL : with non-asserting learning, exponentially more powerful than any natural, proper refinement of Resolution [2004]
3. Pool Resolution: without Restarts, with formula dependent Preprocessing, can simulate Resolution [2008]
4. PD11: with many restarts, can simulate Resolution [2011]
5. regWRTI: every known candidate formula for separating SAT solvers from Resolution has a short regWRTI proof [2012]

Main Findings
1. Restarts in modern SAT solvers can be simulated with Preprocessing
2. Resolution can be simulated by SAT solvers without Restarts

Construction and Correctness
Claim 1: CDCL branching order (u, y_{m-1}, y_{m-2}, ..., y_v, v) on G(m), assigning value False whenever possible, simulates an m-bit counter y_{m-1}y_{m-2}...y_0.
- Counter starts at 0
- Every 2 conflicts advance the counter by 1
- Note: with Phase Saving, this branching order simulates a Gray Code counter

Claim 2: Can simulate up to 2^{m-1} restarts on F(n) by interleaving solver executions over F(n) and G(m), branching on v=False to simulate each restart on F(n).

Distinguishing Features
- Unlike Beame et al: Solver does not selectively ignore unit propagations
- Unlike Hertel et al: Simpler counting formula G(m), disjoint vars. than F(n)
  - Compatible with learning 1 asserting clause per conflict
  - Depends only on n, not on the specifics of F(n)
- Unlike Pipatsrisawat and Darwiche: No restarts, tighter analysis
  - For UNSAT F(n), this produces the exact same proof as with Restarts
  - Compatible with techniques like Backjumping & Phase Saving

Figure 1: Conflict graphs for (a) 1st conflict (b) 2nd conflict
(c) conflict 3 for some j.