A Generalization of SAT and #SAT for Probabilistic Inference and Verification

Alternations and #∃-SAT

\[ \Phi: \{T,F\}^n \to \{T,F\} \]

- \[ Q \in \{∃, ∀, Σ,...\}^n \]
- \[ Q: \{∃, ∀, Σ,...\}^n \to \{T,F\} \]
- \[ x: \{T,F\}^n \]

- \[ Q=∃^n \text{ can encode decision making (SAT)} \]
- \[ Q=Σ^n \text{ can encode probabilities (#SAT)} \]
- \[ \text{With } Q=Σ^m \exists^{n-m} \text{ we can express #∃-SAT, a new type of problem for robust verification} \]

BDD

1. Build a BDD for the formula \( \Phi \) in stratified variable order
2. Eliminate \( ∃ \)-nodes and simplify
3. Count remaining nodes

Only useful if base BDD can be constructed cheaply

DPLL

- \( ∃ \) and \( Σ \)
- Two-level DPLL search
- With or without component detection and caching

Heavy-weight component caching does not seem useful

POPS

- Prune with pessimistic and optimistic SAT subproblems
- Uses a 4-valued logic: lucky, true, false, unlucky

Can exploit a type of structure other solvers cannot

Sample experiment: Job shop scheduling

- Length bits
- Machines
- Jobs
- Time steps

Experiments

- P = 0.75
- P = 0.5
- P = 0.25