SAT Solvers and Configuration Management

Presentation for Mancoosi Project

Mate Soos

UPMC LIP6, PLANETE team INRIA, SALSA Team INRIA

2nd of November 2010

Table of Contents

Outline

1. Context
   - SAT solvers
   - The Mancoosi Project

2. CUDF and SAT solvers
   - Implementation ideas
   - Why would this work?
   - Why wouldn't this work?

3. Conclusions

Motivations and goals

Motivations

- Configuration management emerging problem
- SAT solvers refined tools
- Solve configuration management problems with SAT solvers

Goals

- Show how to use SAT solvers in config. management
- Draw attention advantages&disadvantages in this context

What is a SAT solver

Solves a problem in CNF

CNF is an "and of or-s"

$$(x_1 \vee \neg x_3) \quad \wedge \quad (\neg x_2 \vee x_3) \quad \wedge \quad (x_1 \vee x_2)$$

Uses DPLL(φ) algorithm

1. If formula φ is trivial, return SAT/UNSAT
2. ret = DPLL(φ with $v \leftarrow$ `true`)
3. if ret == SAT, return SAT
4. ret = DPLL(φ with $v \leftarrow$ `false`)
5. if ret == SAT, return SAT
6. return UNSAT

SAT solver internals

Conflict clauses

- Generated when current assignment doesn't satisfy a clause
- Collection of information leading to conflict
- Used to avoid similar wrong parts of the tree next time

Most important parts

- Lazy data structures
- Learning (and forgetting)
- How to pick a variable
- When to restart

Mancoosi

Package management in FLOSS

- Many packages
- Some conflict, some depend on others, some give same features
- Simplified to user: keep, install, upgrade, remove

Common Upgradeability Description Format (CUDF)

1. Preamble with distribution-specific properties
2. Set of packages: dependencies, conflicts, features, properties
3. User request

Solving CUDF

- Optimise for criteria: e.g. least no. changed packages
- Give best solution within time limit
- Result must satisfy dependencies, conflicts, user requests

Outline

1. Context
   - SAT solvers
   - The Mancoosi Project
2. CUDF and SAT solvers
   - Implementation ideas
   - Why would this work?
   - Why wouldn't this work?
3. Conclusions

A trivial implementation

Parser

- Parses up CUDF, optimisation criteria
- Clauses to represent *conflicts*
- Clauses to represent *dependencies*
- Clauses to express if package is real/virtual
- Clauses for user request: keep, install, upgrade, remove

SAT solver

- Gives a solution — correct, but not optimal
- Uses multi-threading
- Keeps track of found unitary and binary truths

A more refined implementation

Parser

- Binary adder for optimality criteria
- Cyclicly restricts adder to smaller values
- Solves until UNSAT — optimal for a criterion
- Solution is optimal for one criterion → backtrack to previous best and optimise for next criterion

SAT Solver

- Constant CNF file as input — contains static needs
- Plus a set of optimality constraints — changes over time
- Keeps state between SAT and SAT
- With help of Parser, some state between SAT and UNSAT

Why would this work?

Simplicity

- SAT solvers already optimised: no re-invent the wheel
- No need to manually multi-thread: it's in the solver
- Must express constraints simply: no repetitions

Right tool for the job: SAT solvers

- Good at binary clauses — conflicts&dependencies create these
- Binary adders are possible to represent natively — CryptoMS patch
- Can save state between runs, no need to solve repeatedly

Why wouldn't this work?

Optimisation&SAT solvers

- SAT solvers not very good at optimisation
- Binary adder could get very large
- Native adder could lead to less effective learnt clauses

Other problems

- No. variables could be huge — at least no. versioned packages
- Difficult to optimise for no repetitions: hash table expensive
- Might need to save more state than unitaries&binaries

Outline

1. Context
   - SAT solvers
   - The Mancoosi Project
2. CUDF and SAT solvers
   - Implementation ideas
   - Why would this work?
   - Why wouldn't this work?
3. Conclusions

Conclusions

Concluding remarks

- SAT is effective at many problems
- Configuration management could be one such problem
- But effort is needed

Future work

- CryManSolver is in preparation
- It will implement the above
- Will use CryptoMiniSat as back-end

Thank you for your time

Any questions?