Wishes for the VeriPB proof format

Daniel Le Berre

SLOPPY '24 - November 7, 2024

Université d'Artois and CNRS

SAT4J is 20 years old ...

- SAT, MAXSAT, PBO solvers
- Main development between 2004 and 2011
- Specific work by Emmanuel Lonca (Multi-Objective Optimization in 2015) and Romain Wallon (PB proof systems in 2020)
- Contains PB solvers with either Resolution-based or Cutting-Planes-based proof systems

Proof logging in Sat4j

- 2013: DRUP UNSAT proof (Daniel)
- 2021: VeriPB 1 UNSAT proof (Antony Blomme and Romain Wallon)
- 2024: iDRUP incremental proof and VeriPB 2 optimal and UNSAT (Daniel)

UNSAT vs incremental vs Optimal proofs in Sat4j

- UNSAT: one call to the solver (can have restarts)
- Incremental: multiple calls to the SAT solver, log calls under assumptions.
- Optimal proof: multiple calls to the SAT solver

Sat4j solvers emit events, which can be used for proof logging

DRUP proof logging: easy

```
public void end(Lbool result) {
    if (result == Lbool.FALSE) {
        out.println("0"); out.close(); }}

public void learn(IConstr c) { printConstr(c);}

public void delete(IConstr c) {
        out.print("d "); printConstr(c); }

private void printConstr(IConstr c) {
    for (var i = 0; i < c.size(); i++) {
        out.print(LiteralsUtils.toDimacs(c.get(i)));
        out.print(" ");
    } out.println("0");}

public void learnUnit(int p) {
    out.print(p); out.println(" 0");}
```

VeriPB 1 proof logging: ”worked” on crafted examples

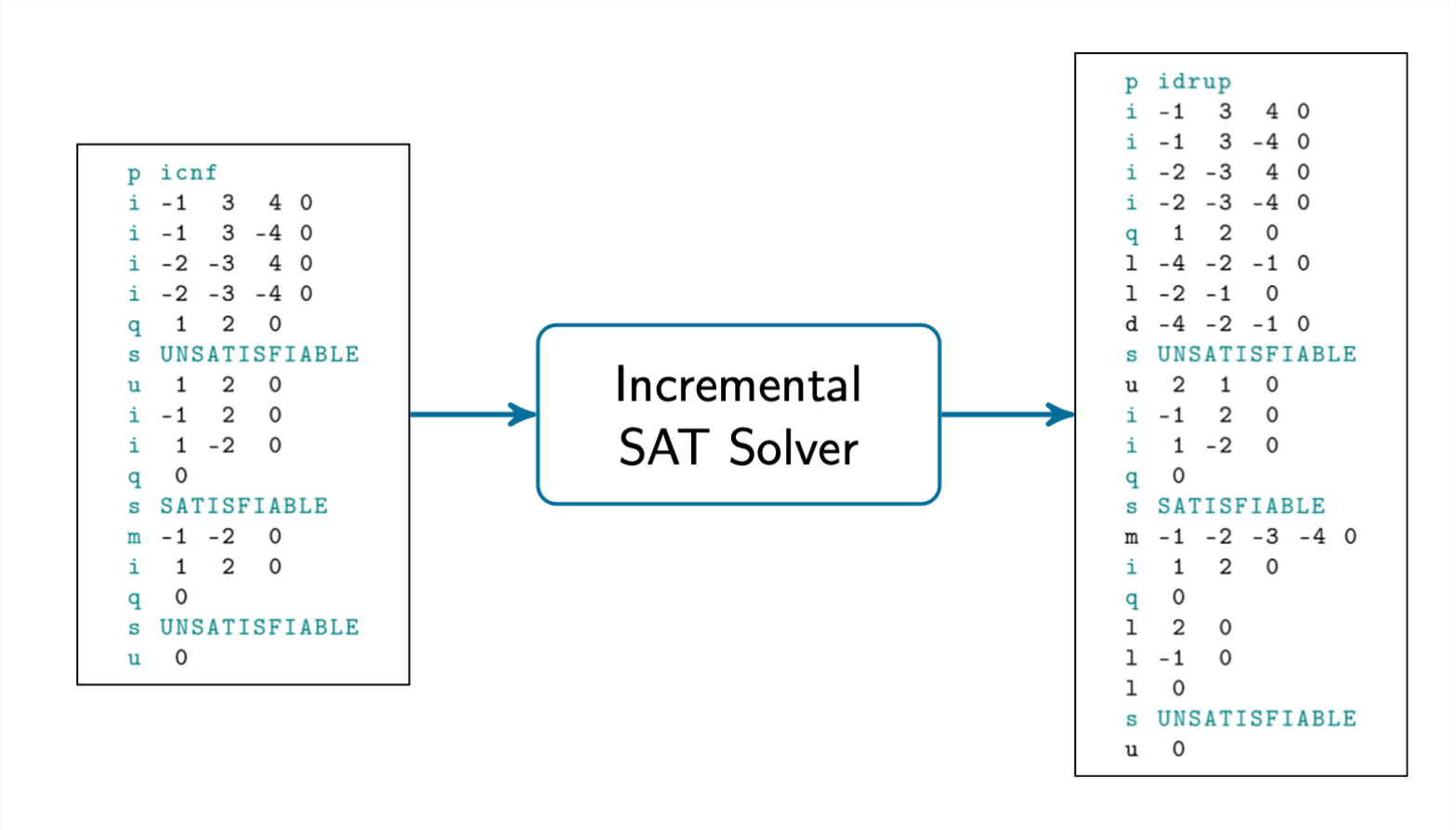
```
public void withReason(PBConstr constr) {
    if (constr != null) {
        this.reason = new StringBuilder("" + constr.getId());}}

public void weakenOnReason(int p) {
    this.reason.append(" x" + Math.abs(p)).append(" w");}

public void weakenOnReason(BigInteger coeff, int p) {
reason.append(" " + coeff).append(" x" + Math.abs(p))
            .append(" W");}

public void weakenOnConflict(int p) {
    this.conflict.append(" x" + Math.abs(p)).append(" w");}

public void weakenOnConflict(BigInteger coeff, int p) {
    this.conflict.append(" " + coeff).append(" x" + Math.abs(p))
            .append(" W");}
```

Incremental DRUP

- Proposed in Katalin Fazekas, Florian Pollitt, Mathias Fleury, Armin Biere: Certifying Incremental SAT Solving. LPAR 2024: 321-340
- Implemented in Sat4j during the master thesis of Iris Parruca studying API fuzzing in Sat4j, advised by Mathias Fleury and Tobias Paxian
- Log all events of incremental SAT: learning clauses, queries with assumptions, models, unsat assumptions.
- Log several search spaces, either SAT or UNSAT
- icnf format and idrup-checker available

`https://github.com/arminbiere/idrup-check`

icnf and idrup example [Fazekas et al 2024]



https://kfazekas.github.io/talks/mbmv2024_talk.pdf

Main issue while implementing iDRUP

- Some solvers in Sat4j can derive clauses but not "watch" them: MiniLearning in 2004
- Those clauses may be used in conflict analysis, but do not propagate nor cause conflicts
- Those clauses needed to be logged in the proof
- New "ignore" event created

Main issue while implementing iDRUP

- Some solvers in Sat4j can derive clauses but not "watch" them: MiniLearning in 2004
- Those clauses may be used in conflict analysis, but do not propagate nor cause conflicts
- Those clauses needed to be logged in the proof
- New "ignore" event created

That problem would have been hard to notice without API fuzzing

VeriPB 2 proof logging: ”worked” on crafted examples

- Benefit from iDRUP implementation for handling multiple calls to the SAT solver
- Passes the test cases from VeriPB 2.0 repository
- VeriPB 1.0 to 2.0 changes were under-estimated

VeriPB 2 proof logging: first (small) difficulty, the output

☆ OR **Olivier Roussel**

[PB24] output DERIVABLE IMPLICIT

À : Daniel Le Berre

Salut Daniel,

Pourquoi utilises-tu dans tes preuves "output DERIVABLE IMPLICIT" et pas "output NONE" ?

A+
OR

VeriPB 2 proof logging: first (small) difficulty, the output

OR

Olivier Roussel

[PB24] output DERIVABLE IMPLICIT

À : Daniel Le Berre

Salut Daniel,

Pourquoi utilises-tu dans tes preuves "output DERIVABLE IMPLICIT" et pas "output NONE" ?

A+
OR

Certificates were accepted by VeriPB but not by CakePB

VeriPB 2 proof logging: the output in Sat4j

```
private void displayConclusion(String answer) throws IOException {
    fw.write("output DERIVABLE IMPLICIT\n");
    fw.write("conclusion " + answer + "\n");
}

public void endWhole(ExitCode status) {
    // ...
        switch (status) {
        case UNSATISFIABLE:
            if (!this.foundContradiction) {
                fw.write("rup >= 1 \n");
                this.nConstraints++;
            }
            if (((IPBSolverService) solverService)
                    .getObjectiveFunction() != null) {
                displayConclusion("BOUNDS INF INF");
            } else {
                displayConclusion("UNSAT");
            }
            break;
        case OPTIMUM_FOUND:
            displayConclusion("BOUNDS " + lowerBound + " " + upperBound);
            break;
        case SATISFIABLE:
            displayConclusion("SAT");
            break;
        default:
            fw.write("output NONE\n");
            fw.write("conclusion NONE\n");
        }
        fw.write("end pseudo-Boolean proof");
        fw.close();
    // ...
}
```

VeriPB 2 proof logging: second (real) difficulty, simplifications

From toy/crafted benchmarks to competition benchmarks

- PB competition benchmarks contain "unit clauses"
- PB competition benchmarks contain PB constraints that can be simplified as cardinality constraints or clauses

Sat4j Cutting Planes VeriPB 2.0 certificates are incorrect on those benchmarks

VeriPB 2 proof logging: second (real) difficulty, simplifications

From toy/crafted benchmarks to competition benchmarks

- PB competition benchmarks contain "unit clauses"
- PB competition benchmarks contain PB constraints that can be simplified as cardinality constraints or clauses

Sat4j Cutting Planes VeriPB 2.0 certificates are incorrect on those benchmarks

Sat4j Resolution VeriPB 2.0 certificates are correct on those benchmarks

Why is it a problem in Sat4j?

- "Unit clauses" are propagated directly when parsing the benchmark
- The simplification is performed to represent the constraint in the most appropriate way in the solver, again while parsing the benchmark
- This is true for both Sat4j Resolution and Sat4j Cutting Planes

There is no "event" in that case, especially because the simplifications can be done in many places

Why is it a problem in Sat4j?

- "Unit clauses" are propagated directly when parsing the benchmark
- The simplification is performed to represent the constraint in the most appropriate way in the solver, again while parsing the benchmark
- This is true for both Sat4j Resolution and Sat4j Cutting Planes

There is no "event" in that case, especially because the simplifications can be done in many places

VeriPB 2.0 is friendly to Resolution proof system, unfriendly with Cutting Planes one

Why is it a problem in Sat4j?

- "Unit clauses" are propagated directly when parsing the benchmark
- The simplification is performed to represent the constraint in the most appropriate way in the solver, again while parsing the benchmark
- This is true for both Sat4j Resolution and Sat4j Cutting Planes

There is no "event" in that case, especially because the simplifications can be done in many places

VeriPB 2.0 is friendly to Resolution proof system, unfriendly with Cutting Planes one

VeriPB 2.0 support does not ship with Sat4j, it lives in a specific branch (VERIPB2)

How to fix this?

On Sat4j side:

- create new events for all simplifications occurring before the search?
- not so easy on 20 years old code (47k LOC)
- API fuzz testing can help (not available for PB yet in Sat4j)

On VeriPB side:

- Could VeriPB be more friendly with equivalent transformations?
- Could VeriPB focus on what we derive, not how we derive it?