universität freiburg

How to Consume CaDiCaL Proofs?

Mathias Fleury
and the CaDiCaL authors

June 12, 2025
Dagstuhl 25231 – Certifying Algorithms for Automated Reasoning

Fix in 2.1.3;

> *Important (behavior changing) bug-fix #108: val now follows the IPASIR-UP interface. This changes the return value for negative literals.*

The semantics was off since the first CaDiCaL release, but nobody realized.

How?

Change in 2.2-rc2;

Without BVA : no change

With BVA : you have to ask the solver for a new variable by calling $\mathrm{vars}()$

CaDiCaL Getting Proofs

```
CaDiCaL::Solver *solver = new CaDiCaL::Solver;

bool with_antecedents = true; // antecedents
bool finalize_proofs = false; // useful for nobody
solver->connect_proof_tracer(my_tracer,
  with_antecedents, finalize_proofs);
```

Proof Formats

DRAT

```
-1 2 0
2 3 0
2 4 0
4 5 0

-1 4 0
```

And RAT additions

LRAT

LRAT
negate the clause and every
propagation is a conflict

```
1  -1 2 0
2  -1 3 0
3  -3 4 0
4  -4 5 0
5  -5 6 0

6  -1 4 0 1 2 3 4 5 0
```

(And RAT additions as `-4 5 7 0`
`-6 5 9 0`)

LRAT

LRAT
negate the clause and every propagation is a conflict

```
1  -1 2 0
2  -1 3 0
3  -3 4 0
4  -4 5 0
5  -5 6 0

6  -1 4 0 1 2 3 4 5 0
```

(And RAT additions as `-4 5 7 0` `-6 5 9 0`)

"Strict" LRAT
negate the clause, every propagation is a conflict, the last is a conflict, every literal is involved

```
1  -1 2 0
2  -1 3 0
3  -3 4 0
4  -4 5 0
5  -5 6 0

6  -1 4 0 2 3 0
```

LRAT

“Strict” LRAT = resolution if you read the it backwards

(Surprisingly tricky to make work for ELS)

Craig interpolation is relying on that property to even be able to reconstruct proofs

Proof Consumption

Getting the Clauses: `Learner`

`Learner` gives you only the learnt clauses.

Very useful when you know a solution but the solver claims UNSAT. (Trick from Sam Buss).

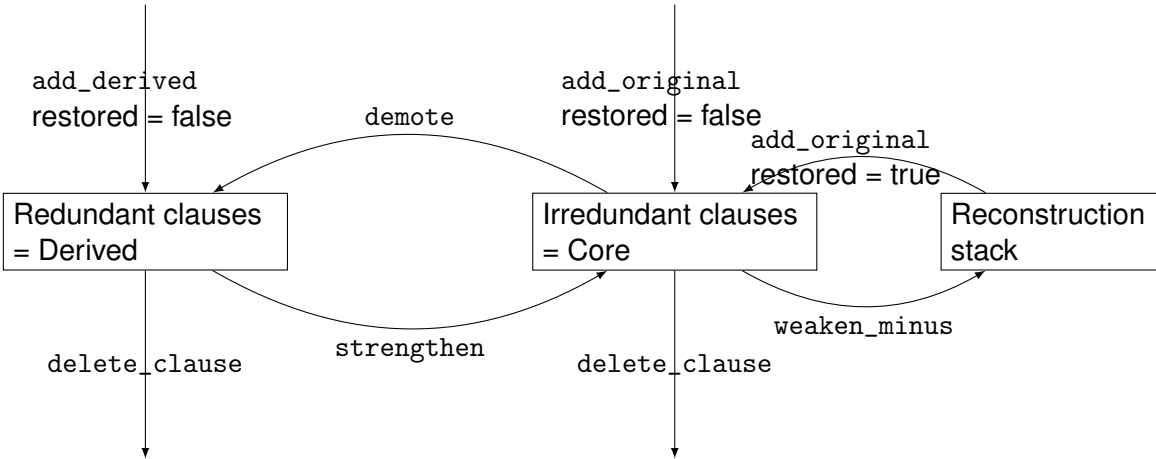
Proof Tracer

Gives you all information on what is happening:

- id based
- which clauses are learnt
- how? (if you ask for antecedents)

CaDiCaL's proof production uses the same interface!

Leaky Abstraction



Leaky Abstraction

- We don't justify deletions
- As user, you probably don't care about redundant/irredundant
- ... neither about reconstruction stack (important for IDRUP)
- You can also iterate over the clauses

Heavy Post-Processing

See for example [Berg, Bogaerts, Nordström, Oertel, Paxian, Vandesand, Certifying Without Loss of Generality Reasoning in Solution-Improving Maximum Satisfiability. CP 2024]

1. Renumber the literals / translate them back
2. Renumber the ids (veriPB 3.0 has explicit ids)

More Listeners

- `FixedAssignment`: for units
- `ExternalPropagator`: for IPASIR-UP interface
- Iterating over clauses
- (MaxHS relies on a CaDiCaL version with `EquivalentLiteralListener`)
- Anything else you need?

Conclusion

Conclusion

- Extracting proofs is actually easy
- Sadly still no standard (not expected to be in IPASIR2)

- If you need more information, just ask us...
- ... although we don't really want you to rely on the internals
- ... and report bugs and performance regressions