universität freiburg

Porting IsaSAT to LLVM (and Inprocessing)

Mathias Fleury and Peter Lammich

CADE'29

Introduction

How do We Make SAT Solvers Correct?

DRAT Proofs

- requires to check the proof for each file
- not all techniques can be represented by current proof formats

How do We Make SAT Solvers Correct?

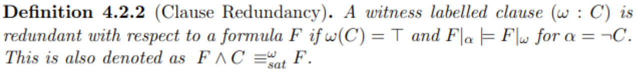
DRAT Proofs
- requires to check the proof for each file
- not all techniques can be represented by current proof formats

Program Verification This talk!
- works for every input, so *no overhead*
- does not crash even if program runs for a year

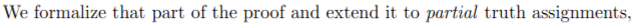
Why do We Care?

- We have abstract CDCL, we have inprocessing, but can we combine them?
- Corner cases easy to miss known papers have flaws!
- Metatheory, study variants See session 1

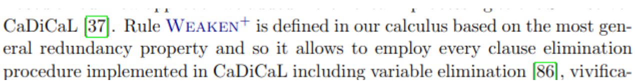
Why do We Care?



Definition 4.2.2 (Clause Redundancy)**.** *A witness labelled clause* $(\omega : C)$ *is redundant with respect to a formula* F *if* $\omega(C) = \top$ *and* $F|_\alpha \models F|_\omega$ *for* $\alpha = \neg C$. *This is also denoted as* $F \wedge C \equiv^{\omega}_{sat} F$.

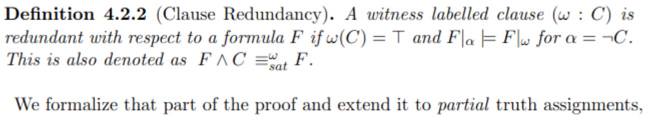


We formalize that part of the proof and extend it to *partial* truth assignments,

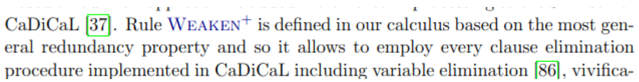


CaDiCaL [37]. Rule WEAKEN$^+$ is defined in our calculus based on the most general redundancy property and so it allows to employ every clause elimination procedure implemented in CaDiCaL including variable elimination [86], vivifica-

Why do We Care?


Definition 4.2.2 (Clause Redundancy). *A witness labelled clause* $(\omega : C)$ *is redundant with respect to a formula* F *if* $\omega(C) = \top$ *and* $F|_\alpha \models F|_\omega$ *for* $\alpha = \neg C$. *This is also denoted as* $F \wedge C \equiv^{\omega}_{sat} F$.

We formalize that part of the proof and extend it to *partial* truth assignments,

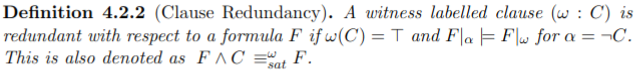

CaDiCaL [37]. Rule WEAKEN$^+$ is defined in our calculus based on the most general redundancy property and so it allows to employ every clause elimination procedure implemented in CaDiCaL including variable elimination [86], vivifica-

Either partial models or VE (Master thesis by Katharina Wagner)

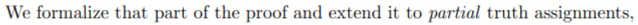
I did not realize that either before Isabelle refused a proof

Implementation heavily tested... on total modals

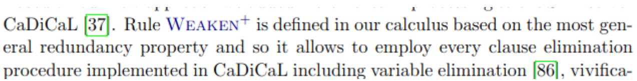
Why do We Care?



Definition 4.2.2 (Clause Redundancy). *A witness labelled clause* $(\omega : C)$ *is redundant with respect to a formula* F *if* $\omega(C) = \top$ *and* $F|_\alpha \models F|_\omega$ *for* $\alpha = \neg C$. *This is also denoted as* $F \wedge C \equiv_{sat}^{\omega} F$.



We formalize that part of the proof and extend it to *partial* truth assignments,



CaDiCaL [37]. Rule WEAKEN$^+$ is defined in our calculus based on the most general redundancy property and so it allows to employ every clause elimination procedure implemented in CaDiCaL including variable elimination [86], vivifica-

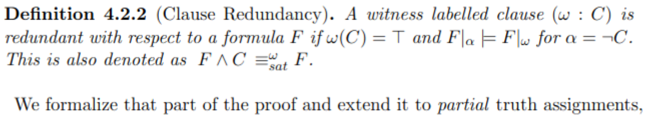
Either partial models or VE (Master thesis by Katharina Wagner)

I did not realize that either before Isabelle refused a proof

Implementation heavily tested... on total modals

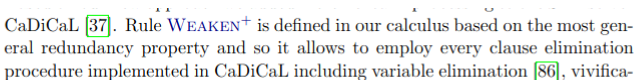
NB: this is not (yet) in IsaSAT

Why do We Care?



Definition 4.2.2 (Clause Redundancy). *A witness labelled clause* $(\omega : C)$ *is redundant with respect to a formula* F *if* $\omega(C) = \top$ *and* $F|_\alpha \models F|_\omega$ *for* $\alpha = \neg C$. *This is also denoted as* $F \wedge C \equiv^{\omega}_{sat} F$.

We formalize that part of the proof and extend it to *partial* truth assignments,



CaDiCaL [37]. Rule WEAKEN$^+$ is defined in our calculus based on the most general redundancy property and so it allows to employ every clause elimination procedure implemented in CaDiCaL including variable elimination [86], vivifica-

Either partial models or VE (Master thesis by Katharina Wagner)

I did not realize that either before Isabelle refused a proof

Implementation heavily tested... on total modals

NB: this is not (yet) in IsaSAT

See session 12, talk by Nikolaj Bjorner

Why do We Care?

- We have abstract CDCL, we have inprocessing, but can we combine them?
- Corner cases easy to miss — known papers have flaws!
- Metatheory, study variants — See session 1
- No gap between specification and implementation

most papers: “we can combine”

Porting to LLVM

Correctness Theorems

$$\begin{tikzcd}
\boxed{\text{Definition}} \arrow[d, bend right, "\text{fulfills}"'] \\
\boxed{\text{Specification}}
\end{tikzcd}$$

Correctness Theorems

$$\begin{tikzcd}\boxed{\text{Definition}} \arrow[d, bend right, "\text{fulfills}"] \\ \boxed{\text{Specification}}\end{tikzcd}$$

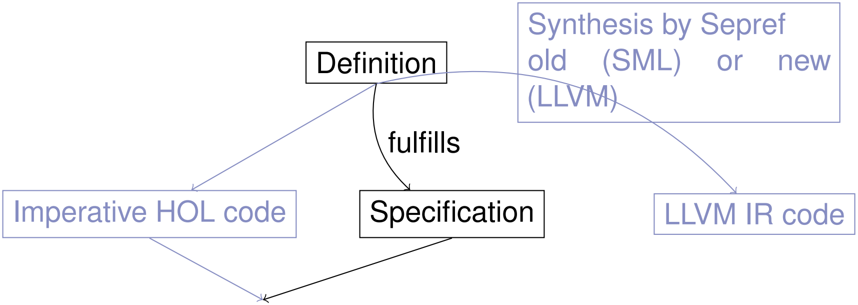
Synthesis by Sepref
old (SML) or new (LLVM)

Correctness Theorems

Synthesis by Sepref old (SML) or new (LLVM)

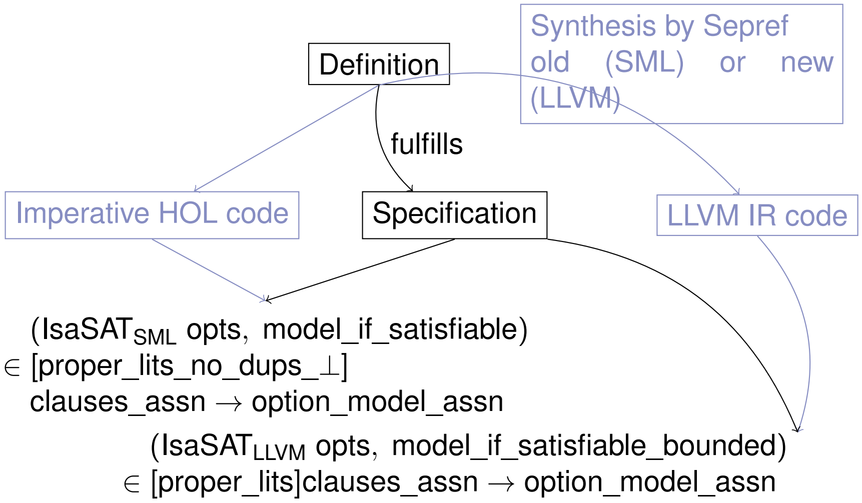
$$\begin{tikzcd}
 & \boxed{\text{Definition}} \arrow[dl] \arrow[dr, bend right, "\text{fulfills}"] & \\
\boxed{\text{Imperative HOL code}} \arrow[dr] & & \boxed{\text{Specification}} \arrow[dl] \\
 & \begin{array}{ll} & (\text{IsaSAT}_{\text{SML}}\ \text{opts},\ \text{model\_if\_satisfiable}) \\ \in & [\text{proper\_lits\_no\_dups\_}\bot] \\ & \text{clauses\_assn} \rightarrow \text{option\_model\_assn} \end{array} &
\end{tikzcd}$$

Correctness Theorems



$$
\begin{aligned}
&(\mathsf{IsaSAT}_{\mathsf{SML}}\ \mathsf{opts},\ \mathsf{model\_if\_satisfiable})\\
\in\ &[\mathsf{proper\_lits\_no\_dups\_}\bot]\\
&\mathsf{clauses\_assn} \rightarrow \mathsf{option\_model\_assn}
\end{aligned}
$$

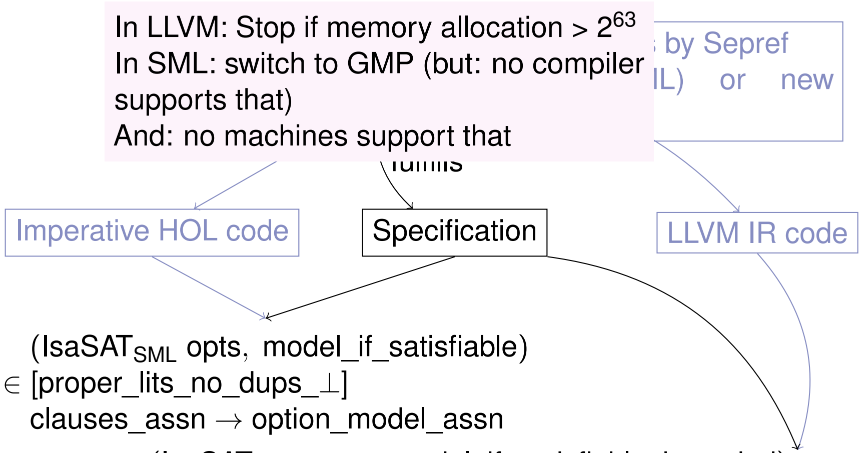
Correctness Theorems



$$(\text{IsaSAT}_{\text{SML}}\ \text{opts},\ \text{model\_if\_satisfiable}) \in [\text{proper\_lits\_no\_dups\_}\bot]\ \text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

$$(\text{IsaSAT}_{\text{LLVM}}\ \text{opts},\ \text{model\_if\_satisfiable\_bounded}) \in [\text{proper\_lits}]\text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

Correctness Theorems



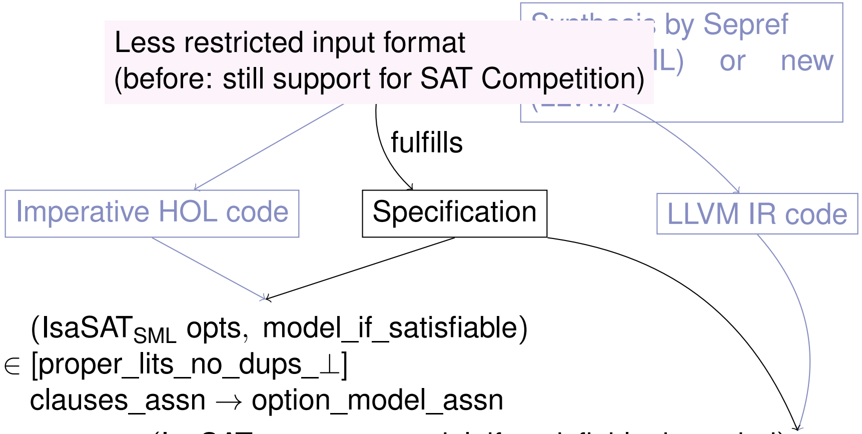
In LLVM: Stop if memory allocation > 2^{63}
In SML: switch to GMP (but: no compiler supports that)
And: no machines support that

$$(\text{IsaSAT}_{\text{SML}}\ \text{opts},\ \text{model\_if\_satisfiable}) \in [\text{proper\_lits\_no\_dups\_}\bot]\ \text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

$$(\text{IsaSAT}_{\text{LLVM}}\ \text{opts},\ \text{model\_if\_satisfiable\_bounded}) \in [\text{proper\_lits}]\text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

Correctness Theorems

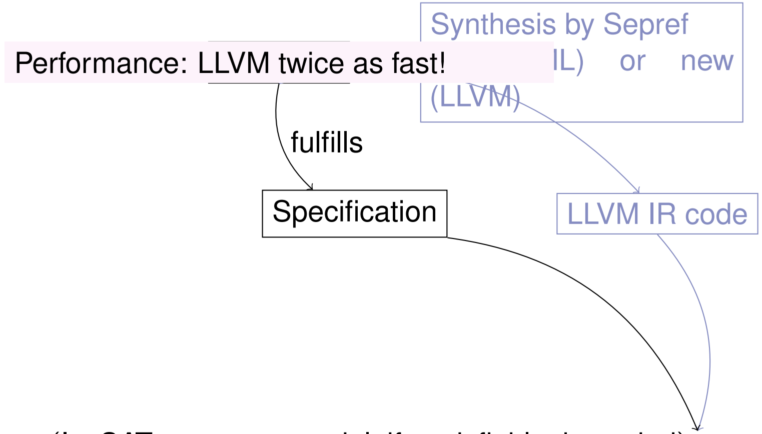
Less restricted input format
(before: still support for SAT Competition)



$$(\text{IsaSAT}_{\text{SML}}\ \text{opts},\ \text{model\_if\_satisfiable}) \in [\text{proper\_lits\_no\_dups\_}\bot]\ \text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

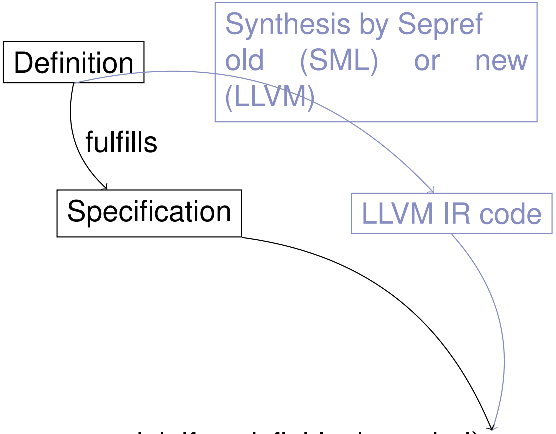
$$(\text{IsaSAT}_{\text{LLVM}}\ \text{opts},\ \text{model\_if\_satisfiable\_bounded}) \in [\text{proper\_lits}]\text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

Correctness Theorems



$$(\text{IsaSAT}_{\text{LLVM}}\ \text{opts},\ \text{model\_if\_satisfiable\_bounded})$$
$$\in [\text{proper\_lits}]\text{clauses\_assn} \rightarrow \text{option\_model\_assn}$$

Correctness Theorems



$$(\mathrm{IsaSAT}_{\mathrm{LLVM}}\ \mathrm{opts},\ \mathrm{model\_if\_satisfiable\_bounded}) \in [\mathrm{proper\_lits}]\mathrm{clauses\_assn} \rightarrow \mathrm{option\_model\_assn}$$

Lessons Learned

Aggressive reorganization of theories... but not enough

- IsaSAT is the largest development based on Isabelle/LLVM (94kloc)
- Sepref enforces a non-parallel style (and is not very fast)
- Currently 1h 20 for the non-synthesis part, 2h with it

  same for (smaller) SML version
- Synthesis rather slow and (for technical reasons) single threaded

Lessons Learned
Local reasoning is better

Before

$$\begin{tikzcd}[column sep=small]
(\mathcal{A}, \arrow[dr] & \ldots, & W, \ldots) \arrow[dl] \\
 & W &
\end{tikzcd}$$

After

$$\begin{tikzcd}[column sep=small]
 & = & & \\
(\mathcal{A}, \ldots, \arrow[ur, bend left] & & (\mathcal{A}, W), \arrow[ul, bend right] \arrow[dr] & \ldots) \\
 & & & W
\end{tikzcd}$$

This is ongoing refactoring: the new things are ported, the olds whenever I change them

Lessons Learned
Performance bugs happen

A normal run:

```
c propagate                  : 83.48% (581.66 s)
c reduce                     : 0.12% (0.82 s)
c subsumption                : 0.06% (0.39 s)
c pure_lits                  : 0.05% (0.33 s)
c binary_simp                : 0.02% (0.15 s)
c GC                         : 0.16% (1.10 s)
```

(Hacked)Implementation by calling C function (via `sed/grep`)

optional, but very useful

But you have to trust the (unverified) statistics!

Performance Results

CDF on the SAT Competition 2022

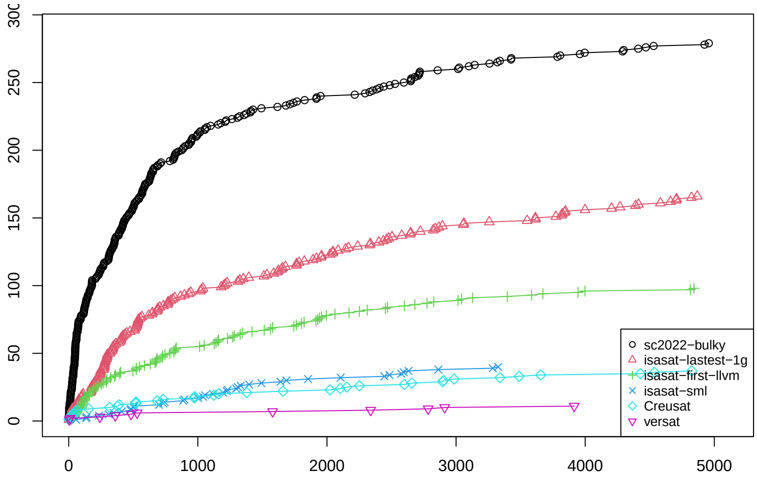


Figure: Higher is better (best: Kissat, unverified)

Conclusion

Summary

- IsaSAT is now much better

  Bug: after 2 years: GC and reduction swapped!

  I still suspect a heuristic bug

- We have now inprocessing with general rules specialized for our needs

  See paper for Subsumption-Resolution

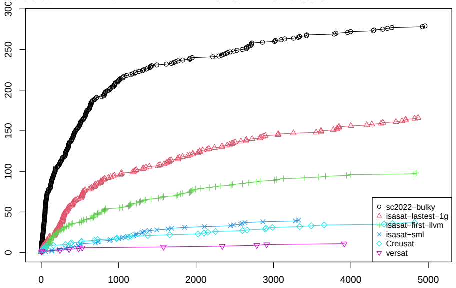
- We know how to improve the formalization but it won't solve all problems

- For Main track SAT Competition, DRAT proof (via sed to interface LLVM to C code)

  "officially", I am not aware of any bug during proof checking

Summary

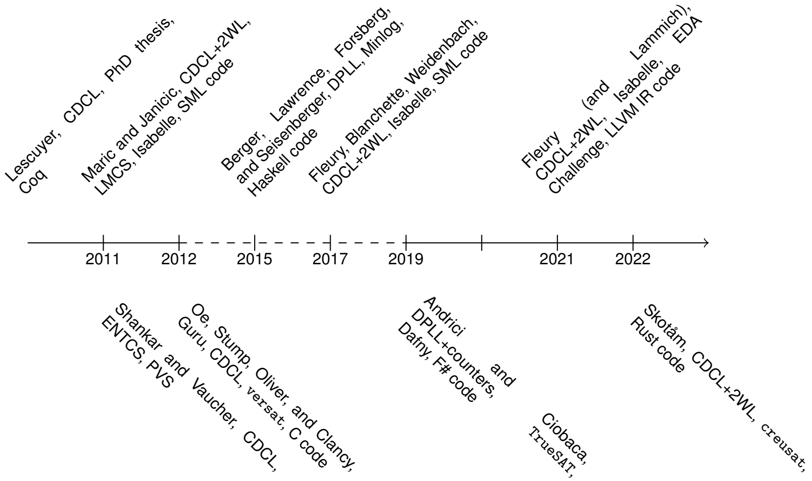
- IsaSAT is now much better



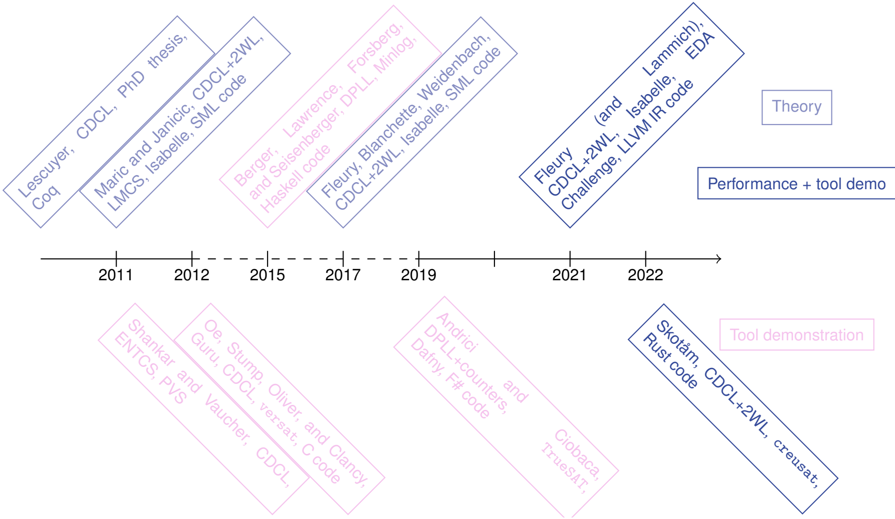
- We have now inprocessing
- We know how to improve the formalization but it won't solve all problems

Questions?

A Personal History of Solver Verification


Lescuyer, CDCL, PhD thesis, Coq
Maric and Janicic, CDCL+2WL, LMCS, Isabelle, SML code
Berger, Lawrence, Forsberg, and Seisenberger, DPLL, Minlog, Haskell code
Fleury, Blanchette, Weidenbach, CDCL+2WL, Isabelle, SML code
Fleury (and Lammich), CDCL+2WL, Isabelle, EDA Challenge, LLVM IR code
2011
2012
2015
2017
2019
2021
2022
Shankar and Vaucher, CDCL, ENTCS, PVS
Oe, Stump, Oliver, and Clancy, Guru, CDCL, versat, C code
Andrici and Ciobaca, DPLL+counters, TrueSAT, Dafny, F# code
Skotåm, CDCL+2WL, creusat, Rust code

A Personal History of Solver Verification


Lescuyer, CDCL, PhD thesis, Coq
Maric and Janicic, CDCL+2WL, LMCS, Isabelle, SML code
Berger, Lawrence, Forsberg, and Seisenberger, DPLL, Minlog, Haskell code
Fleury, Blanchette, Weidenbach, CDCL+2WL, Isabelle, SML code
Fleury (and Lammich), CDCL+2WL, Isabelle, EDA Challenge, LLVM IR code
Theory
Performance + tool demo
2011
2012
2015
2017
2019
2021
2022
Shankar and Vaucher, CDCL, ENTCS, PVS
Oe, Stump, Oliver, and Clancy, Guru, CDCL, versat, C code
Andrici and Ciobaca, DPLL+counters, TrueSAT, Dafny, F# code
Skotåm, CDCL+2WL, creusat, Rust code
Tool demonstration