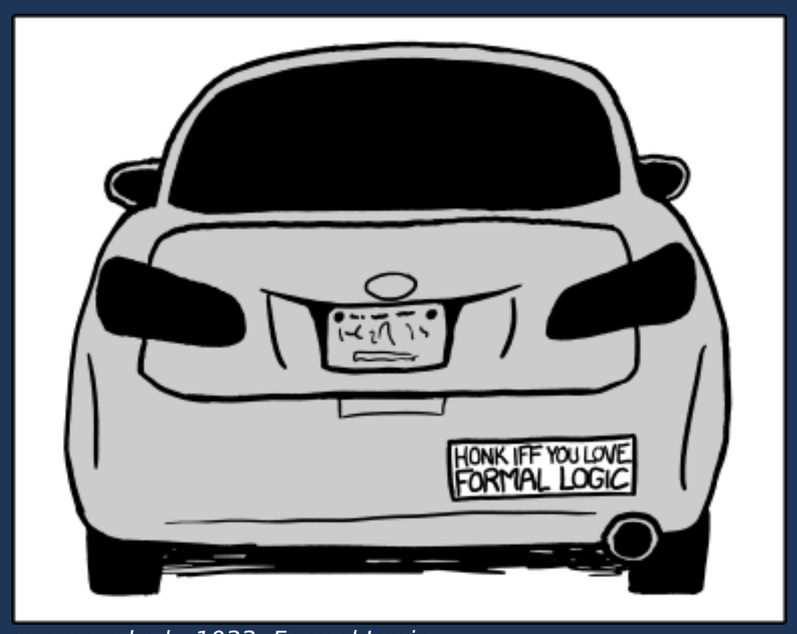
Incremental Inprocessing in SAT Solving

Katalin Fazekas, TU Wien, Vienna, Austria
Armin Biere, Albert-Ludwigs-University, Freiburg, Germany
Christoph Scholl, Albert-Ludwigs-University, Freiburg, Germany



source: *xkcd - 1033: Formal Logic*
Title text: *Note that this implies you should NOT honk solely because I stopped for a pedestrian and you're behind me.*

Motivation - Automated Formal Verification

Modern cars contain ~150 Electronic Control Units (ECUs) running software from different suppliers. How can we be sure that these software components are correct?

> *"The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness."*
> *E.W. Dijkstra, 1972*

Is our system correct? ⇆ **Is formula F satisfiable?**

We need efficient tools to answer this question!

Background - SAT

Is this formula satisfiabile?

$$(a \vee \neg b) \wedge (a \vee b) \wedge (\neg a \vee \neg b)$$

- Yes, a possible model:

$$\{a \leftarrow True, b \leftarrow False\}$$

- NP-complete decision problem
- SAT solvers find a model or prove unsatisfiability

Shape of Verification Problems

Is formula $\mathcal{F}_0$ satisfiable?
Is formula $\mathcal{F}_0 \wedge \mathcal{F}_1$ satisfiable?
Is formula $\mathcal{F}_0 \wedge \mathcal{F}_1 \wedge \mathcal{F}_2$ satisfiable?

- Sequence of decision problems
- Each problem is an extension of the previous

Our Contributions

- Introduced a novel technique to combine incremental reasoning with inprocessing during SAT solving
- Provided a simple but efficient algorithm to implement it
- Improved performance in a hardware verification application
- Formally proved the correctness of our approach
- Simplified the use of incremental solvers, less work expected from user

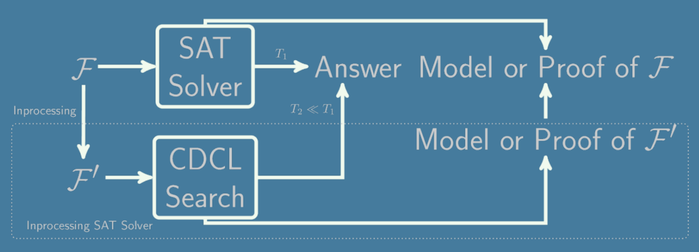
Our Method

1. Allow full inprocessing as in non-incremental solvers
2. When a new formula is added, identify the potentially problematic previous simplification steps
3. Undo each problematic simplification step

Formula Pre- and Inprocessing

- Formula simplifications before and during search

$$\mathcal{F} = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5$$
$$\mathcal{F}' = C_1 \wedge \cancel{C_2} \wedge C_3 \wedge \cancel{C_4} \wedge C_5 \wedge C_6$$
$$\mathcal{F} \equiv_{sat} \mathcal{F}'$$



Non-Incremental vs. Incremental Reasoning

- Solve each formula with the exact same solver
- Can reuse reasoning steps instead of repeating them
- Leads to significant speed-up in practice

Problem with Inprocessing in Incremental Reasoning - Example

$$\mathcal{F}_0 \wedge \mathcal{F}_1 \not\equiv_{sat} \mathcal{F}'_0 \wedge \mathcal{F}_1$$

	Unsound solution		Our sound solution	
What we would like to solve:	$\mathcal{F}_0 = (a \vee b)$	$\mathcal{F}_0 \wedge \mathcal{F}_1 = \mathcal{F}_0 \wedge (\neg a) \wedge (\neg b)$	$\mathcal{F}_0 = (a \vee b)$	$\mathcal{F}_0 \wedge \mathcal{F}_1 = \mathcal{F}_0 \wedge (\neg a) \wedge (\neg b)$
Inprocessing	Eliminate $(a \vee b)$	Do nothing	Eliminate $(a \vee b)$	Restore $(a \vee b)$
What the solver actually solves:	$\{\}$	$\{\} \wedge (\neg a) \wedge (\neg b)$	$\{\}$	$(a \vee b) \wedge (\neg a) \wedge (\neg b)$
Answer of the solver:	SAT	SAT	SAT	UNSAT

Sound Incremental Inprocessing Rules

$$\frac{\varphi\ [\ \rho\ ]\ \sigma}{\varphi\ [\ \rho \wedge C\ ]\ \sigma}\ \natural \quad \text{Learn}^-$$

$$\frac{\varphi\ [\ \rho \wedge C\ ]\ \sigma}{\varphi\ [\ \rho\ ]\ \sigma} \quad \text{Forget}$$

$$\frac{\varphi\ [\ \rho \wedge C\ ]\ \sigma}{\varphi \wedge C\ [\ \rho\ ]\ \sigma} \quad \text{Strengthen}$$

$$\frac{\varphi\ [\ \rho\ ]\ \sigma}{\varphi \wedge \Delta\ [\ \rho\ ]\ \sigma}\ \mathcal{I} \quad \text{AddClauses}$$

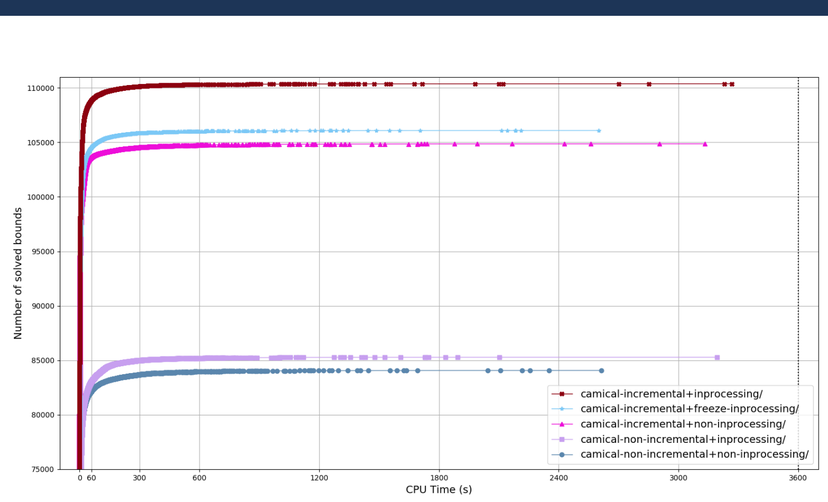
$$\frac{\varphi \wedge C\ [\ \rho\ ]\ \sigma}{\varphi\ [\ \rho\ ]\ \sigma \cdot (\omega : C)}\ \flat \quad \text{Weaken}^+$$

$$\frac{\varphi \wedge C\ [\ \rho\ ]\ \sigma}{\varphi\ [\ \rho\ ]\ \sigma}\ \varnothing \quad \text{Drop}$$

$$\frac{\varphi\ [\ \rho\ ]\ \sigma \cdot (\omega : C) \cdot \sigma'}{\varphi \wedge C\ [\ \rho\ ]\ \sigma \cdot \sigma'}\ \partial \quad \text{Restore}$$

where $\natural$ is $\varphi \wedge \rho \models C$, $\flat$ is $\varphi \wedge C \equiv^{\omega}_{sat} \varphi$, $\varnothing$ is $\varphi \models C$,
∂ is C is clean w.r.t. σ' and $\mathcal{I}$ is that each clause in Δ is clean w.r.t. σ

Experimental Results



Results of CaMiCaL on the 300 instances of the single safety track of HWMCC'17, using CaDiCaL as a back-end SAT solver with different configurations. y-axis: all bounds solved over all problems sorted by the time needed for the SAT call for each bound.

TECHNISCHE UNIVERSITÄT WIEN