Bounded Model Generation for Isabelle/HOL

and Related Applications of SAT Solvers in Interactive Theorem Proving

Tjark Weber

webertj@in.tum.de

TECHNISCHE UNIVERSITÄT MÜNCHEN

Winterhütte, März 2005

Isabelle

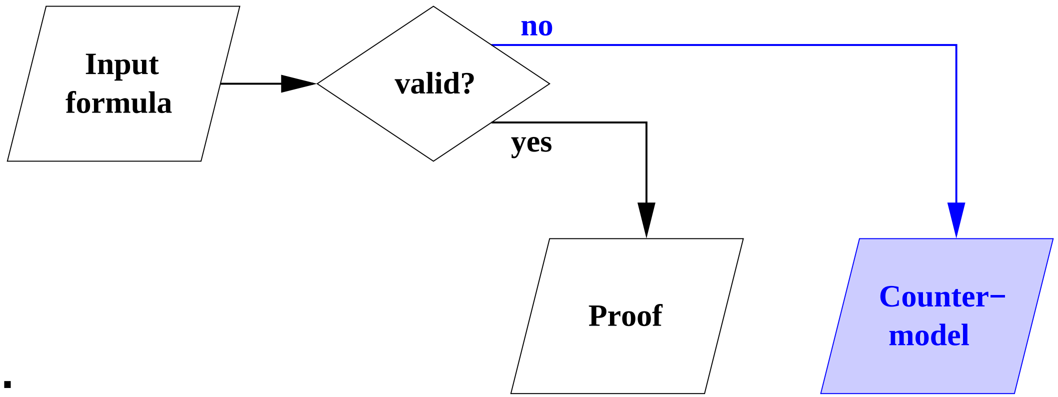
Isabelle is a generic proof assistant:

- Highly flexible
- Interactive
- Automatic proof procedures
- Advanced user interface
- Readable proofs
- Large theories of formal mathematics

Bounded Model Generation

Theorem proving: from formulae to proofs

Bounded model generation: *from formulae to models*



Applications:

- *Finding counterexamples to false conjectures*
- Showing the consistency of a specification
- Solving open mathematical problems
- Guiding resolution-based provers

Isabelle/HOL

HOL: higher-order logic based on Church's simple theory of types (1940)

Simply-typed λ-calculus:

- Types: $\sigma ::= \mathbb{B}$ | α | $\sigma \rightarrow \sigma$
- Terms: $t_\sigma ::= x_\sigma$ | $(t_{\sigma' \rightarrow \sigma}\, t_{\sigma'})_\sigma$ | $(\lambda x_{\sigma_1} .\, t_{\sigma_2})_{\sigma_1 \rightarrow \sigma_2}$

Two logical constants:

- $\Longrightarrow_{\mathbb{B} \rightarrow \mathbb{B} \rightarrow \mathbb{B}}$, $=_{\sigma \rightarrow \sigma \rightarrow \mathbb{B}}$

Isabelle/HOL

HOL: higher-order logic based on Church's simple theory of types (1940)

Simply-typed λ-calculus:

- Types: $\sigma ::= \mathbb{B} \mid \alpha \mid \sigma \rightarrow \sigma$
- Terms: $t_\sigma ::= x_\sigma \mid (t_{\sigma' \rightarrow \sigma}\, t_{\sigma'})_\sigma \mid (\lambda x_{\sigma_1}.\, t_{\sigma_2})_{\sigma_1 \rightarrow \sigma_2}$

Two logical constants:

- $\Longrightarrow_{\mathbb{B} \rightarrow \mathbb{B} \rightarrow \mathbb{B}}$, $=_{\sigma \rightarrow \sigma \rightarrow \mathbb{B}}$

Other constants, e.g.

$$\texttt{True} \mid \texttt{False} \mid \neg \mid \wedge \mid \vee \mid \forall \mid \exists \mid \exists!$$

are definable.

The Semantics of HOL

Set-theoretic semantics:

- Types denote certain sets.
- Terms denote elements of these sets.

The Semantics of HOL

Set-theoretic semantics:

- Types denote certain sets.
- Terms denote elements of these sets.

An *environment* D assigns to each type variable α a non-empty set D_α.

Semantics of types:

- $D(\mathbb{B}) = \{\top, \bot\}$
- $D(\alpha) = D_\alpha$
- $D(\sigma_1 \to \sigma_2) = D(\sigma_2)^{D(\sigma_1)}$

The Semantics of HOL (2)

A *variable assignment* A maps each variable x_σ to an element $A(x_\sigma)$ in $D(\sigma)$.

Semantics of terms:

- $[\![x_\sigma]\!]_D^A = A(x_\sigma)$
- $[\![(t_{\sigma'\to\sigma}\, t_{\sigma'})_\sigma]\!]_D^A = [\![t_{\sigma'\to\sigma}]\!]_D^A([\![t_{\sigma'}]\!]_D^A)$
- $[\![(\lambda x_{\sigma_1}.\, t_{\sigma_2})_{\sigma_1\to\sigma_2}]\!]_D^A$ is the function that sends each d in $D(\sigma_1)$ to $[\![t_{\sigma_2}]\!]_D^{A[x_{\sigma_1}\mapsto d]}$
- $\Longrightarrow_{\mathbb{B}\to\mathbb{B}\to\mathbb{B}}, =_{\sigma\to\sigma\to\mathbb{B}}$: implication, equality

Hence the semantics of a term is an element of the set denoted by the term's type.

Overview

Input: HOL formula ϕ

Output: either a model for ϕ, or “no model found”

Overview

Input: HOL formula ϕ

1. Fix a finite environment D.
2. Translate ϕ into a Boolean formula that is satisfiable iff $[\![\phi]\!]_D^A = \top$ for some variable assignment A.
3. Use a SAT solver to search for a satisfying assignment.
4. If a satisfying assignment was found, compute from it the variable assignment A. Otherwise repeat for a larger environment.

Output: either a model for ϕ, or “no model found”

Fixing a Finite Environment

Fix a positive integer for every type variable that occurs in the typing of ϕ.

Every type then has a finite size:

- $|\mathbb{B}| = 2$
- $|\alpha|$ is given by the environment
- $|\sigma_1 \to \sigma_2| = |\sigma_2|^{|\sigma_1|}$

Finite model generation is a generalization of satisfiability checking, where the search tree is not necessarily binary.

The SAT Solver

Several *external* SAT solvers (zChaff, BerkMin, Jerusat, . . .) are supported.

- Efficiency
- Advances in SAT solver technology are “for free”

The SAT Solver

Several *external* SAT solvers (zChaff, BerkMin, Jerusat, . . .) are supported.

- Efficiency
- Advances in SAT solver technology are “for free”

Simple *internal* solvers are available as well.

- Easy installation
- Compatibility
- Fast enough for small examples

Some Extensions

Sets are interpreted as characteristic functions.

- $\sigma\ \texttt{set} \cong \sigma \to \mathbb{B}$
- $x \in P \cong P\,x$
- $\{x.\,P\,x\} \cong P$

Non-recursive datatypes can be interpreted in a finite model.

- $(\alpha_1, \ldots, \alpha_n)\sigma ::= C_1\,\sigma_1^1 \ldots \sigma_{m_1}^1 | \ldots | C_k\,\sigma_1^k \ldots \sigma_{m_k}^k$
- $|(\alpha_1, \ldots, \alpha_n)\sigma| = \sum_{i=1}^{k} \prod_{j=1}^{m_i} |\sigma_j^i|$
- Examples: *option*, *sum*, *product* types

Some Extensions

Recursive datatypes are restricted to initial fragments.

- Examples: `nat`, σ `list`, `lambdaterm`
- $\mathtt{nat}^1 = \{0\}$, $\mathtt{nat}^2 = \{0, 1\}$, $\mathtt{nat}^3 = \{0, 1, 2\}, \ldots$
- This works for datatypes that occur only positively.

Datatype *constructors* and *recursive functions* can be interpreted as partial functions.

- Examples: $\mathrm{Suc}_{\mathrm{nat}\to\mathrm{nat}}$, $+_{\mathrm{nat}\to\mathrm{nat}\to\mathrm{nat}}$, $@_{\sigma\mathrm{list}\to\sigma\mathrm{list}\to\sigma\mathrm{list}}$
- 3-valued logic: true, false, unknown

Axiomatic type classes introduce additional axioms that must be satisfied by the model.

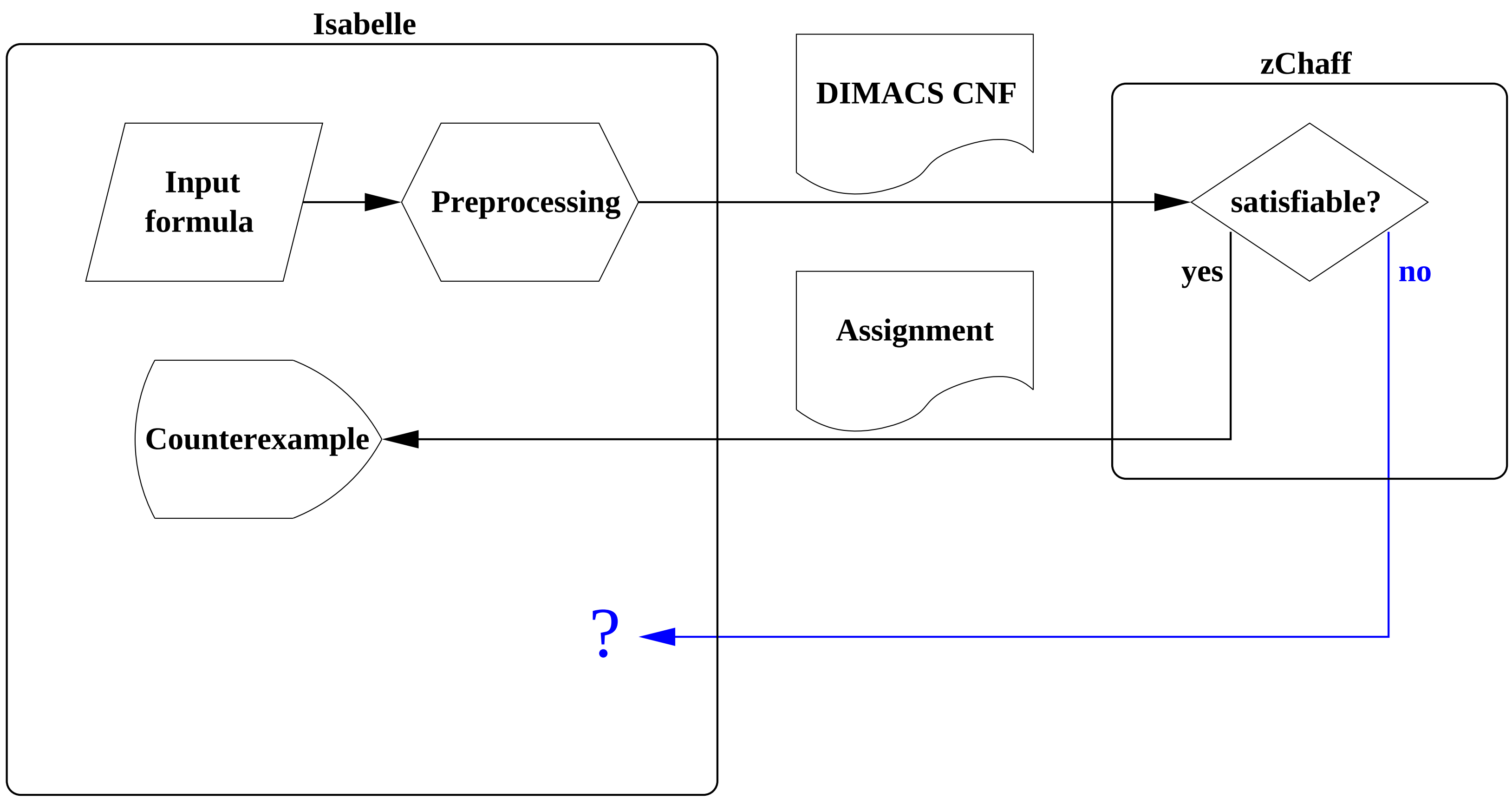
Records and *inductively defined sets* can be treated as well.

Soundness and Completeness

If the SAT solver is sound/complete, we have ...

- *Soundness*: The algorithm returns “model found” only if the given formula has a finite model.
- *Completeness*: If the given formula has a finite model, the algorithm will find it (given enough time).
- *Minimality*: The model found is a smallest model for the given formula.

“No Model Found”

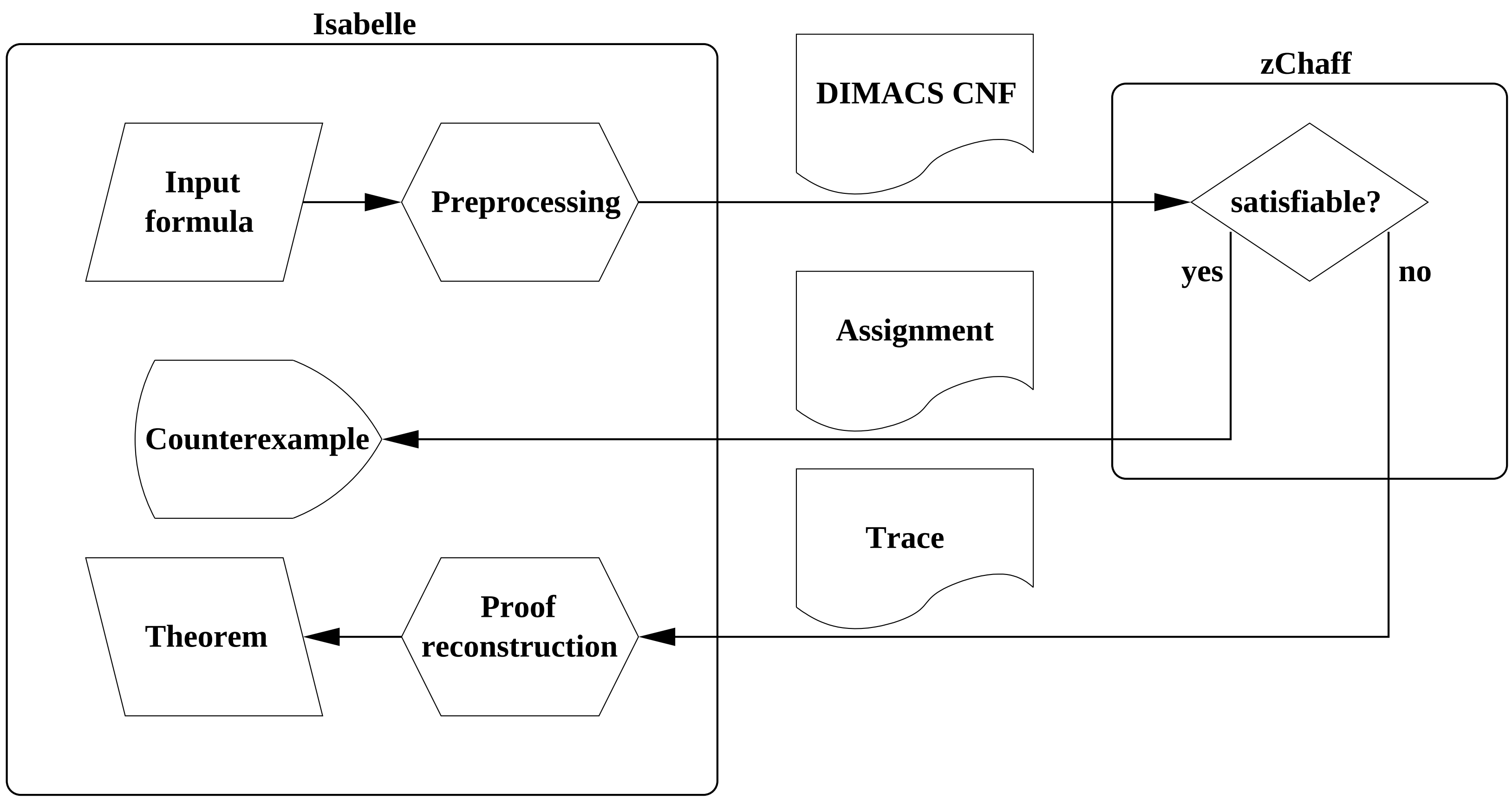


Unsatisfiability – Helpful at All?

- If the Boolean formula is unsatisfiable, the HOL formula ϕ does not have a model of a certain size.
- If ϕ has the finite model property, we can test all models up to the required size.
- If no model is found, $\neg\phi$ must be provable.

Difficult to implement . . . let's only look at *Boolean formulae* for now.

Deciding Boolean Formulae with zChaff



The Algorithm

Preprocessing:

- No conversion from HOL is necessary, only from Boolean logic into CNF.
- But the conversion must be *proof-generating*, i.e. return a theorem $\phi = \phi_{\mathrm{CNF}}$.

The Algorithm

Preprocessing:

- No conversion from HOL is necessary, only from Boolean logic into CNF.
- But the conversion must be *proof-generating*, i.e. return a theorem $\phi = \phi_{\mathrm{CNF}}$.

Proof reconstruction:

- zChaff returns a *resolution-style proof* of unsatisfiability.
- The proof is replayed in Isabelle/HOL to derive $\neg\phi$.

Performance

- Isabelle is several orders of magnitude slower than zverify_df.
- However, zChaff vs. auto/blast/fast . . .
  - 42 propositional problems in TPTP, v2.6.0
    - 19 “easy” problems, solved in less than a second each by auto, blast, fast, and zchaff_tac
    - 23 harder problems

Performance

Problem	Status	auto	blast	fast	zChaff
MSC007-1.008	unsat.	x	x	x	726.5
NUM285-1	sat.	x	x	x	0.2
PUZ013-1	unsat.	0.5	x	5.0	0.1
PUZ014-1	unsat.	1.4	x	6.1	0.1
PUZ015-2.006	unsat.	x	x	x	10.5
PUZ016-2.004	sat.	x	x	x	0.3
PUZ016-2.005	unsat.	x	x	x	1.6
PUZ030-2	unsat.	x	x	x	0.7
PUZ033-1	unsat.	0.2	6.4	0.1	0.1
SYN001-1.005	unsat.	x	x	x	0.4
SYN003-1.006	unsat.	0.9	x	1.6	0.1
SYN004-1.007	unsat.	0.3	822.2	2.8	0.1
SYN010-1.005.005	unsat.	x	x	x	0.4
SYN086-1.003	sat.	x	x	x	0.1
SYN087-1.003	sat.	x	x	x	0.1
SYN090-1.008	unsat.	13.8	x	x	0.5
SYN091-1.003	sat.	x	x	x	0.1
SYN092-1.003	sat.	x	x	x	0.1
SYN093-1.002	unsat.	1290.8	16.2	1126.6	0.1
SYN094-1.005	unsat.	x	x	x	0.8
SYN097-1.002	unsat.	x	19.2	x	0.2
SYN098-1.002	unsat.	x	x	x	0.4
SYN302-1.003	sat.	x	x	x	0.4

Conclusions and Future Work

- Finite countermodels for HOL formulae
- A fast decision procedure for Boolean formulae

- Further optimizations, benchmarks
- A SAT-based decision procedure for a fragment of HOL
- Integration of external model generators
- …