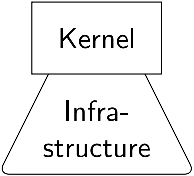
Isabelle/HOL and SMT

Sascha Böhme

Technische Universität München

September 10, 2009

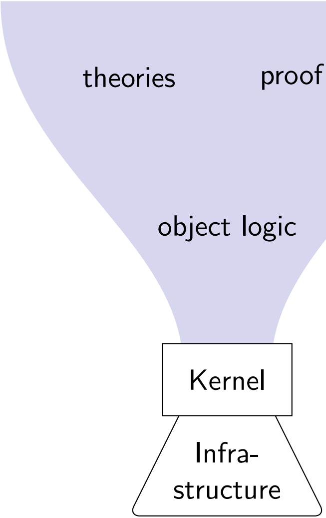
Isabelle – A Generic Theorem Prover



- theorems: abstract type
- inference rules: intuitionistic higher-order logic

terms, types, . . .

Isabelle – A Generic Theorem Prover



- theorems: abstract type
- inference rules: intuitionistic higher-order logic

terms, types, . . .

Isabelle's Meta-Logic

Terms:

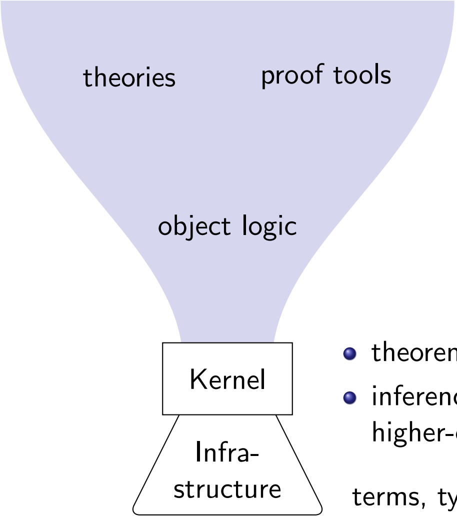
- constants ($\bigwedge$, $\Longrightarrow$, $\equiv$)
- variables
- λ-abstraction
- application

Theorems: $H \vdash P$

Rules:

- assumption
- introduction and elimination of $\bigwedge$ and $\Longrightarrow$ and $\equiv$
- reflexivity, symmetry, transitivity, congruence
- generalization, instantiation
- higher-order resolution

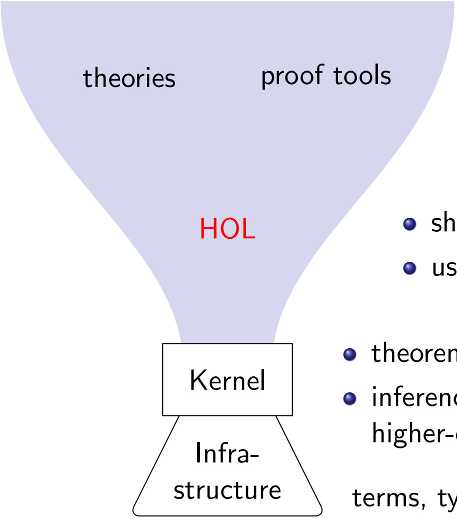
Isabelle/HOL – Higher-Order Logic in Isabelle



- theorems: abstract type
- inference rules: intuitionistic higher-order logic

terms, types, . . .

Isabelle/HOL – Higher-Order Logic in Isabelle

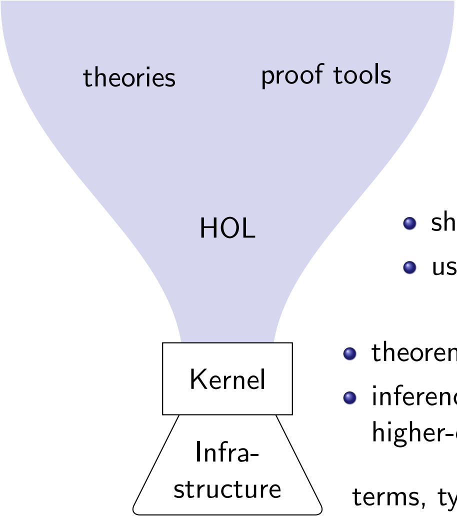


- shallow embedding in meta logic
- usual connectives and functions

- theorems: abstract type
- inference rules: intuitionistic higher-order logic

terms, types, ...

Isabelle/HOL – Higher-Order Logic in Isabelle



- term rewriting
- tableaux prover
- arithmetic

- shallow embedding in meta logic
- usual connectives and functions

- theorems: abstract type
- inference rules: intuitionistic higher-order logic

terms, types, . . .

Satisfiability Modulo Theories (SMT)

Many-sorted first-order logic

Theories:

- equality and uninterpreted functions
- linear (integer/real) arithmetic
- arrays
- bitvectors
- algebraic datatypes

Combination: in general undecidable with high complexity

- necessary fragment still successful: program verification, model checking, ...

SMT solvers: CVC3, Yices, Z3, ...

Isabelle/HOL and SMT

Observation: many essentially first-order propositions:

- Sledgehammer: connection to first-order provers

With SMT:

- built-in support for additional theories (e.g. linear arithmetic)
- weaker on quantifiers

SMT cannot (directly) deal with:

- polymorphism
- λ-abstractions
- induction

Isabelle/HOL and SMT

Observation: many essentially first-order propositions:

- Sledgehammer: connection to first-order provers

With SMT:

- built-in support for additional theories (e.g. linear arithmetic)
- weaker on quantifiers

SMT cannot (directly) deal with:

- polymorphism: monomorphization, encoding of types in terms
- λ-abstractions
- induction

Isabelle/HOL and SMT

Observation: many essentially first-order propositions:

- Sledgehammer: connection to first-order provers

With SMT:

- built-in support for additional theories (e.g. linear arithmetic)
- weaker on quantifiers

SMT cannot (directly) deal with:

- polymorphism: monomorphization, encoding of types in terms
- λ-abstractions: combinatory logic (SKI), lifting
- induction

Isabelle/HOL and SMT

Observation: many essentially first-order propositions:

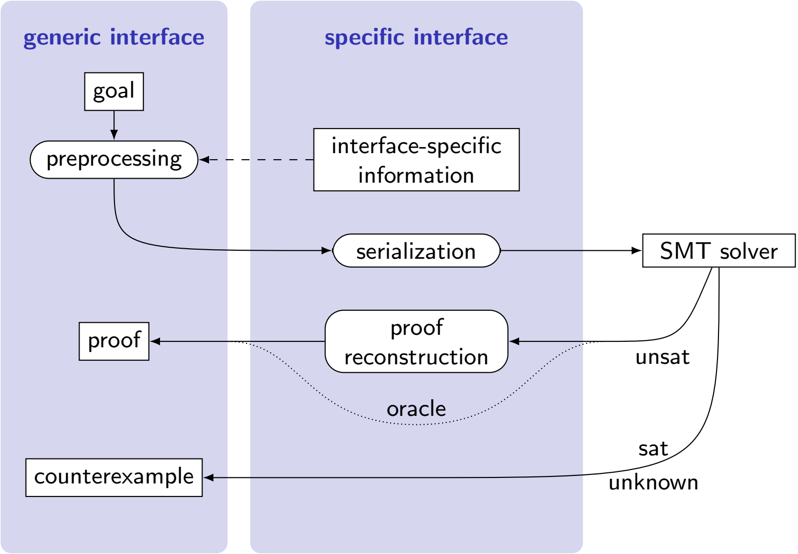
- Sledgehammer: connection to first-order provers

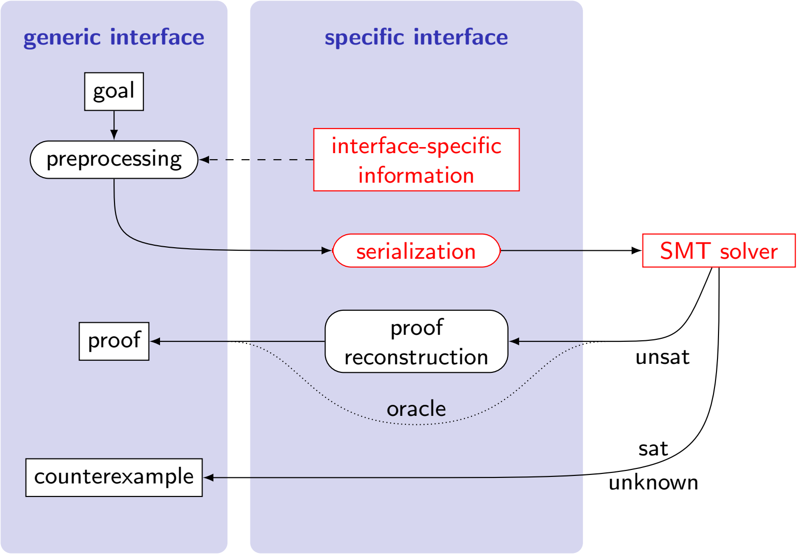
With SMT:

- built-in support for additional theories (e.g. linear arithmetic)
- weaker on quantifiers

SMT cannot (directly) deal with:

- polymorphism: monomorphization, encoding of types in terms
- λ-abstractions: combinatory logic (SKI), lifting
- induction (but partial unfolding of recursive functions)


generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
unknown
counterexample


generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
unknown
counterexample

Supported SMT Solvers and Formats

Generic approach:

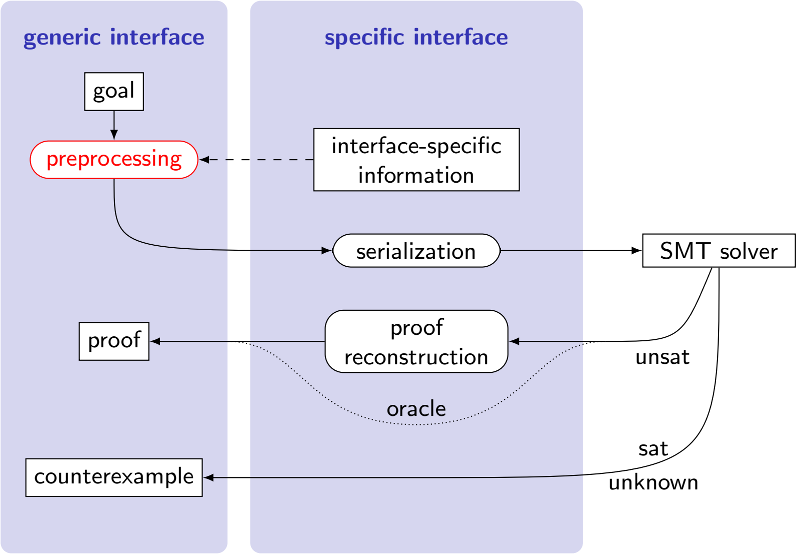
- low effort to integrate new solvers

SMT-LIB format:

- supported by practically all available solvers
- separates terms and formulas
- fixed logics (combination of theories)
- no polymorphism

Z3 low-level format:

- no separation between terms and formulas
- supports all theories and any combination
- restricted polymorphism

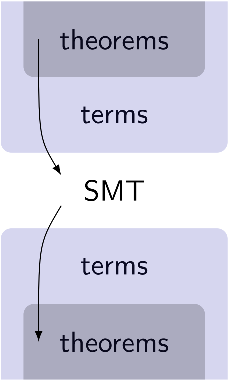

generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
unknown
counterexample

Preprocessing

SMT:

- requires transformations of essentially first-order HOL terms

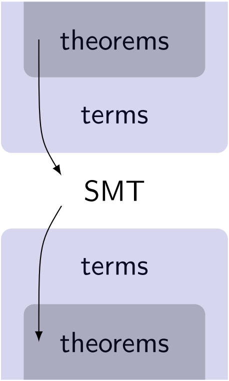
Preprocessing



SMT:

- requires transformations of essentially first-order HOL terms

Preprocessing



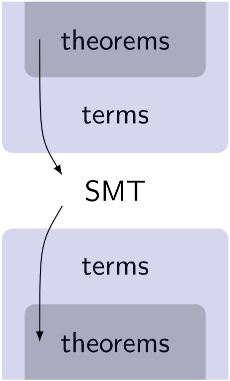
SMT:

- requires transformations of essentially first-order HOL terms

Rewriting of theorems (normalization):

- establish properties necessary for serialization and proof reconstruction

Preprocessing



SMT:

- requires transformations of essentially first-order HOL terms

Rewriting of theorems (normalization):

- establish properties necessary for serialization and proof reconstruction

Term transformations (decoration):

- prepare only serialization
- can use “dirty” tricks
- faster/simpler than theorem rewriting

Rewriting of Theorems (Normalization)

- Negative numerals: rewrite into negated positive numerals

Rewriting of Theorems (Normalization)

- Negative numerals: rewrite into negated positive numerals
- Natural numbers: embed into integers
  - add axiomatization of *nat* and *int*

Example

$$P\ (2 + x) \ \rightsquigarrow\ P\ (\mathit{nat}\ (2 + \mathit{int}\ x))$$

Rewriting of Theorems (Normalization)

- Negative numerals: rewrite into negated positive numerals
- Natural numbers: embed into integers
  - add axiomatization of *nat* and *int*

Example

$$P\ (2 + x) \ \rightsquigarrow\ P\ (\mathit{nat}\ (2 + \mathit{int}\ x))$$

- Lambda terms: lift

Example

$$\mathit{map}\ (\lambda x.\ x + 1)\ [1, 2] = [2, 3] \ \rightsquigarrow\ \begin{cases} \forall x.\ f\ x = x + 1 \\ \mathit{map}\ f\ [1, 2] = [2, 3] \end{cases}$$

Rewriting of Theorems (Normalization)

- Negative numerals: rewrite into negated positive numerals
- Natural numbers: embed into integers
  - add axiomatization of *nat* and *int*

Example

$$P\ (2 + x) \ \rightsquigarrow\ P\ (\mathit{nat}\ (2 + \mathit{int}\ x))$$

- Lambda terms: lift

Example

$$\mathit{map}\ (\lambda x.\ x + 1)\ [1, 2] = [2, 3] \ \rightsquigarrow \begin{cases} \forall x.\ f\ x = x + 1 \\ \mathit{map}\ f\ [1, 2] = [2, 3] \end{cases}$$

- Axiomatization for *abs*, *min*, *max*, and pairs

Term Transformations (Decoration)

Monomorphization:

- compute necessary instances of polymorphic constants
- copy and instantiate polymorphic assumptions
- enforce termination: upper limit on generated copies
- simple, but can cause blow-up of formulas

Term Transformations (Decoration)

Monomorphization:

- compute necessary instances of polymorphic constants
- copy and instantiate polymorphic assumptions
- enforce termination: upper limit on generated copies
- simple, but can cause blow-up of formulas

Identification of built-in symbols

Term Transformations (Decoration)

Monomorphization:

- compute necessary instances of polymorphic constants
- copy and instantiate polymorphic assumptions
- enforce termination: upper limit on generated copies
- simple, but can cause blow-up of formulas

Identification of built-in symbols

Separation between formulas and terms:

- insert marker symbol
- add axiomatization for term-level occurrences of $\wedge$, $\vee$, $\leq$, ...

Term Transformations (Decoration)

Monomorphization:

- compute necessary instances of polymorphic constants
- copy and instantiate polymorphic assumptions
- enforce termination: upper limit on generated copies
- simple, but can cause blow-up of formulas

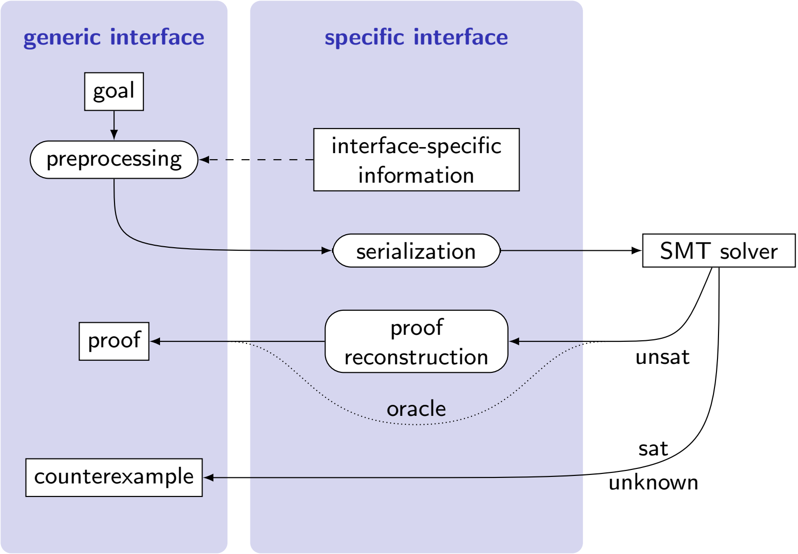
Identification of built-in symbols

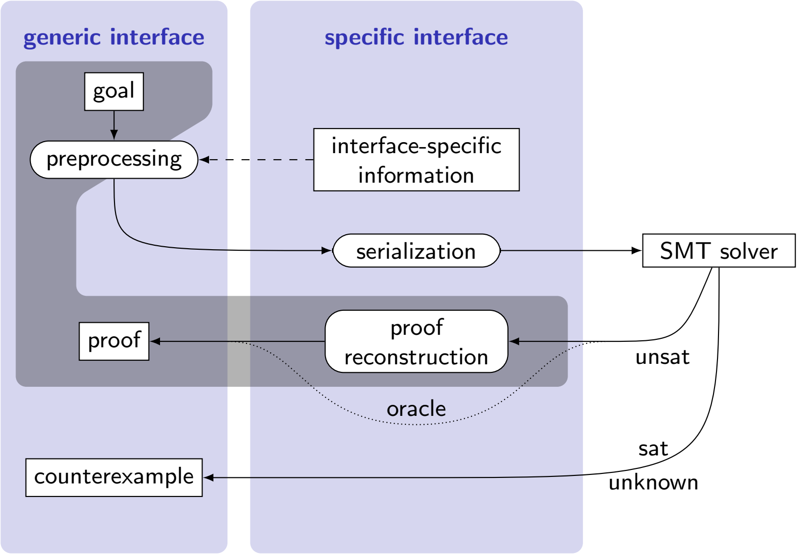
Separation between formulas and terms:

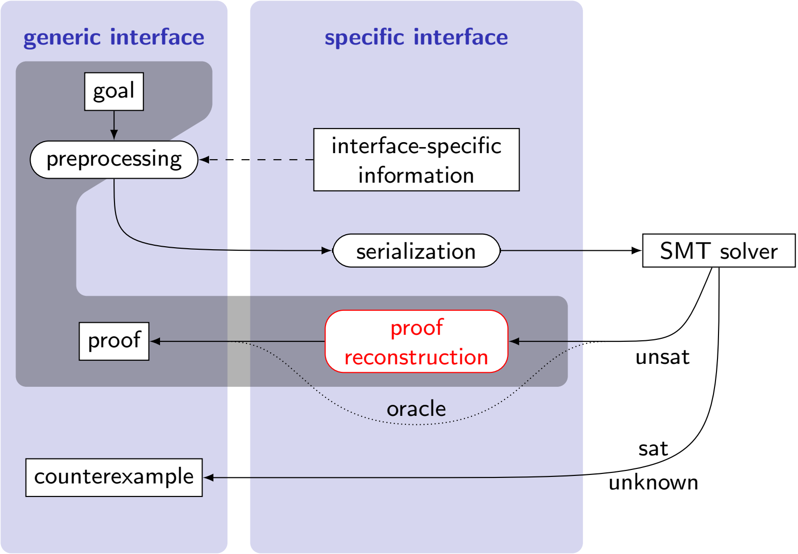
- insert marker symbol
- add axiomatization for term-level occurrences of $\wedge$, $\vee$, $\leq$, ...

Transformation of partially-applied functions:

- additional symbol: make application explicit


generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
unknown
counterexample


generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
unknown
counterexample


generic interface
specific interface
goal
preprocessing
interface-specific information
serialization
SMT solver
proof
proof reconstruction
unsat
oracle
sat
counterexample
unknown

Z3 Terms

Signature:

- types: basic types (*int*, *real*) and user-defined types (nullary type constructors)
- function symbols: fixed arity, no polymorphism

Terms:

- variables: x, y
- applications: $f\ t_1 \ldots t_n$
- quantifiers (triggers are ignored)

Formulas (terms of sort *bool*): P, Q

Natural mapping into HOL term structure

Equisatisfiability

Example

$$(\neg x \vee \mathit{false}) \sim (\neg y)$$

Semantics: existential closure

Example

$$(\exists x.\ \neg x \vee \mathit{false}) \leftrightarrow (\exists y.\ \neg y)$$

Representation in HOL:

- equivalence without existential closure
- exception: Skolemization

Z3 Proofs

Natural deduction style:

Example

$$\dfrac{\dfrac{}{\neg true \ \vdash \ \neg true}\ \textbf{asserted} \qquad \dfrac{}{\vdash \ \neg true \leftrightarrow false}\ \textbf{rewrite}}{\neg true \ \vdash \ false}\ \textbf{mp}_{\leftrightarrow}$$

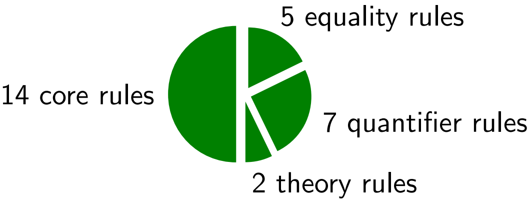
Z3 Proofs

Natural deduction style:

Example

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

28 proof rules:



Proof Reconstruction

Follows the proof structure:

$$\dfrac{\dfrac{}{\neg true \vdash \neg true}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg true \leftrightarrow false}\ \textbf{rewrite}}{\neg true \vdash false}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule

$$\dfrac{\dfrac{}{\neg true \vdash \neg true}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg true \leftrightarrow false}\ \textbf{rewrite}}{\neg true \vdash false}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule
- all inferences certified by Isabelle kernel

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule
- all inferences certified by Isabelle kernel
- global check at the end

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

Proof Reconstruction

Follows the proof structure:

- bottom-up
- one method for every rule
- all inferences certified by Isabelle kernel
- global check at the end
- local checks for debugging

$$\dfrac{\dfrac{}{\neg \mathit{true} \vdash \neg \mathit{true}}\ \textbf{asserted} \qquad \dfrac{}{\vdash \neg \mathit{true} \leftrightarrow \mathit{false}}\ \textbf{rewrite}}{\neg \mathit{true} \vdash \mathit{false}}\ \textbf{mp}_{\leftrightarrow}$$

Reconstruction Methods

Reconstruction Methods

- Direct representation or basic inference rule: (3 rules)

Examples

$$\frac{}{\vdash\ \mathit{true}}\ \textbf{true-prop} \qquad \frac{}{P \vdash P}\ \textbf{asserted}$$

Reconstruction Methods

- Direct representation or basic inference rule: (3 rules)

Examples

$$\frac{}{\vdash \; true}\ \textbf{true-prop} \qquad \frac{}{P \;\vdash\; P}\ \textbf{asserted}$$

- Theorem or inference rule, and resolution: (9 rules)

Example

$$\frac{\Gamma_1 \;\vdash\; P_1 \qquad \Gamma_2 \;\vdash\; P_1 \leftrightarrow P_2}{\Gamma_1 \cup \Gamma_2 \;\vdash\; P_2}\ \mathbf{mp}_{\leftrightarrow}$$

in Isabelle: $P_1 \Longrightarrow P_1 \leftrightarrow P_2 \Longrightarrow P_2$

Reconstruction Methods

- Direct representation or basic inference rule: (3 rules)

Examples

$$\frac{}{\vdash \ true}\ \textbf{true-prop} \qquad \frac{}{P \vdash P}\ \textbf{asserted}$$

- Theorem or inference rule, and resolution: (9 rules)

Example

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_1 \leftrightarrow P_2}{\Gamma_1 \cup \Gamma_2 \vdash P_2}\ \textbf{mp}_{\leftrightarrow}$$

in Isabelle: $P_1 \Longrightarrow P_1 \leftrightarrow P_2 \Longrightarrow P_2$

- Isabelle proof tools (7 rules)

Reconstruction Methods: The Remaining 9 Rules

Special treatment due to:

- no available proof tools
- optimizations for central proof rules

Optimizations:

- meta-equality instead of HOL equality
- cheap inference rules of Isabelle kernel
- memoize intermediate steps
- reduce number of resolution steps, prepare suitable theorems

Unit Resolution

Example

$$\frac{P_1 \vee \neg P_2 \vee \neg P_3 \qquad P_2}{P_1 \vee \neg P_3}$$

Unit Resolution

Example

$$\frac{P_1 \vee (\neg P_2 \vee \neg P_3) \qquad P_2}{P_1 \vee \neg P_3}$$

Unit Resolution

Example

$$\frac{P_1 \vee (\neg P_2 \vee \neg P_3) \qquad P_2}{P_1 \vee \neg P_3}$$

Idea: combine resolution with rewriting

Example with rewriting

$$\frac{P_1 \vee (\neg P_2 \vee \neg P_3) \qquad \dfrac{P_2}{P_1 \vee (\neg P_2 \vee \neg P_3) \equiv P_1 \vee \neg P_3}}{P_1 \vee \neg P_3}$$

Unit Resolution

$$\frac{}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}$$

Unit Resolution

$$\dfrac{\overline{P_1 \equiv P_1}}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}$$

Unit Resolution

$$\dfrac{\dfrac{}{P_1 \equiv P_1} \qquad \dfrac{}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}$$

Unit Resolution

$$\dfrac{\dfrac{}{P_1 \equiv P_1} \qquad \dfrac{\dfrac{P_2}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}$$

Unit Resolution

$$\dfrac{\dfrac{}{P_1 \equiv P_1} \qquad \dfrac{\dfrac{P_2}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}\, E_1}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}$$

$$E_1 : \quad \dfrac{P_2 \qquad Q_1 \Longrightarrow \neg Q_1 \vee Q_2 \equiv Q_2}{\neg P_2 \vee Q_2 \equiv Q_2}$$

Unit Resolution

$$
\dfrac{\dfrac{}{P_1 \equiv P_1} \qquad \dfrac{\dfrac{P_2}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}\,E_1 \qquad \dfrac{}{\neg P_3 \equiv \neg P_3}}{\neg P_2 \vee \neg P_3 \equiv \neg P_3}}{P_1 \vee \neg P_2 \vee \neg P_3 \equiv P_1 \vee \neg P_3}
$$

$$
E_1 : \quad \dfrac{P_2 \qquad Q_1 \Longrightarrow \neg Q_1 \vee Q_2 \equiv Q_2}{\neg P_2 \vee Q_2 \equiv Q_2}
$$

Congruence

Natural choice: use Isabelle's simplifier

But: custom-made procedure provides much better performance

Idea: combine reflexivity and congruence of basic inference rules

Example

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f \equiv f} \qquad a \equiv b}{f\ a \equiv f\ b} \qquad \dfrac{}{c \equiv c}}{f\ a\ c \equiv f\ b\ c} \qquad d \equiv e}{f\ a\ c\ d \equiv f\ b\ c\ e}$$

Memoization for Conjunction Elimination

Example

$$\frac{P_1 \wedge P_2 \wedge P_3}{P_2}$$

Similar: conclude P_1 or P_3

Idea:

1. explode $P_1 \wedge P_2 \wedge P_3$ once into literals
2. memoize literals
3. pick required literal on demand

Dually for negated disjunction elimination

Skolemization

Example

$$\frac{}{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

Skolemization

Example

$$\frac{}{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

With Hilbert choice operator ε

$$\frac{}{f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash (\exists x.\, P\ x\ y) \leftrightarrow P\ (f\ y)\ y}$$

Skolemization

Example

$$\overline{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

With Hilbert choice operator ε

$$\overline{f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash (\exists x.\, P\ x\ y) \leftrightarrow P\ (f\ y)\ y}$$

At the end of reconstruction:

$$\Gamma, f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash \mathit{false}$$

Skolemization

Example

$$\frac{}{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

With Hilbert choice operator ε

$$\frac{}{f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash (\exists x.\, P\ x\ y) \leftrightarrow P\ (f\ y)\ y}$$

At the end of reconstruction:

$$\frac{\Gamma, f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash \mathit{false}}{\Gamma \vdash f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \Longrightarrow \mathit{false}}$$

Skolemization

Example

$$\frac{}{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

With Hilbert choice operator ε

$$\frac{}{f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash (\exists x.\, P\ x\ y) \leftrightarrow P\ (f\ y)\ y}$$

At the end of reconstruction:

$$\cfrac{\cfrac{\Gamma, f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash \mathit{false}}{\Gamma \vdash f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \Longrightarrow \mathit{false}}}{\Gamma \vdash (\lambda y.\, \varepsilon x.\, P\ x\ y) \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \Longrightarrow \mathit{false}}$$

Skolemization

Example

$$\frac{}{\vdash (\exists x.\, P\ x\ y) \sim P\ (f\ y)\ y}$$

With Hilbert choice operator ε

$$\frac{}{f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash (\exists x.\, P\ x\ y) \leftrightarrow P\ (f\ y)\ y}$$

At the end of reconstruction:

$$\cfrac{\cfrac{\cfrac{\Gamma, f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \vdash \mathit{false}}{\Gamma \vdash f \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \Longrightarrow \mathit{false}}}{\Gamma \vdash (\lambda y.\, \varepsilon x.\, P\ x\ y) \equiv (\lambda y.\, \varepsilon x.\, P\ x\ y) \Longrightarrow \mathit{false}}}{\Gamma \vdash \mathit{false}}$$

Rewrite

"The head function symbol of the left-hand side is interpreted."

Examples

$$\overline{P_1 \wedge P_2 \wedge \mathit{true} = P_2 \wedge P_1} \qquad \overline{(x < y) = (y + (-1 * x) > 0)}$$

Several possible simplification steps:

- ACI rewriting of $\wedge$ and $\vee$
- AC rewriting of non-idempotent functions (e.g. $+$)
- arithmetic: polynomial normal-form
- array: application of access/update-rules
- quantifier elimination: $(\exists x.\ 1 \leq x \wedge x < y) = (1 < y)$

Rewrite

Approach 1: try

1. identified simplication rules
2. custom-made ACI rewriting for $\wedge$ and $\vee$
3. simplifier (arrays) and arithmetic decision procedures

Approach 2:

- choose the appropriate method
- based on the head symbol of the left-hand side

Overall difference negligible:

- Isabelle's arithmetic DPs take much longer

Evaluation

Recurrence relation $x_{i+2} = |x_{i+1}| - x_i$ has period 9:

- with Isabelle's arithmetic: 4 minutes
- with Z3: 15 seconds

SMT-LIB benchmarks:

- industrial problems: huge formulas
- Z3 proofs: around 100KB, up to several MB
- reconstruction: around 20 times slower than proof finding

Some Quirks in Z3's Proof Generation

$$\frac{}{\vdash\ P \wedge (\forall x : int.\, x > 0) \ \leftrightarrow\ false \wedge P}\ \textbf{rewrite}$$

Some Quirks in Z3's Proof Generation

$$\frac{}{\vdash\ P \wedge (\forall x : int.\, x > 0)\ \leftrightarrow\ false \wedge P}\ \textbf{rewrite}$$

$$\frac{\Gamma_1 \vdash\ P_1 \vee P_2 \vee P_1 \qquad \Gamma_2 \vdash\ \neg P_2}{\Gamma_1 \cup \Gamma_2 \vdash\ P_1}\ \textbf{unit}$$

Some Quirks in Z3's Proof Generation

$$\frac{}{\vdash\ P \wedge (\forall x : \mathit{int}.\, x > 0)\ \leftrightarrow\ \mathit{false} \wedge P}\ \textbf{rewrite}$$

$$\frac{\Gamma_1\ \vdash\ P_1 \vee P_2 \vee P_1 \qquad \Gamma_2\ \vdash\ \neg P_2}{\Gamma_1 \cup \Gamma_2\ \vdash\ P_1}\ \textbf{unit}$$

$$\frac{\Gamma_1\ \vdash\ s = t \qquad \Gamma_2\ \vdash\ u = t}{\Gamma_1 \cup \Gamma_2\ \vdash\ s = u}\ \textbf{trans}$$

Some Quirks in Z3's Proof Generation

$$\frac{}{\vdash\ P \wedge (\forall x : \mathit{int}.\, x > 0)\ \leftrightarrow\ \mathit{false} \wedge P}\ \textbf{rewrite}$$

$$\frac{\Gamma_1\ \vdash\ P_1 \vee P_2 \vee P_1 \qquad \Gamma_2\ \vdash\ \neg P_2}{\Gamma_1 \cup \Gamma_2\ \vdash\ P_1}\ \textbf{unit}$$

$$\frac{\Gamma_1\ \vdash\ s = t \qquad \Gamma_2\ \vdash\ u = t}{\Gamma_1 \cup \Gamma_2\ \vdash\ s = u}\ \textbf{trans}$$

$$\frac{}{f\ x = 1 + x + g\ x}\ \textbf{rewrite}*$$

Conclusion

Generic connection of SMT solvers with Isabelle/HOL:

- can solve many essentially first-order formulas
- can cope (to some extent) with polymorphism, λ-expressions, and recursive functions

Proof reconstruction for Z3:

- certifying connection of Z3 with Isabelle/HOL
- several optimizations
- helped to improve Z3 proof generation