Scalable fine-grained proofs for formula processing

Haniel Barbosa, Jasmin Christian Blanchette,
Simon Cruanes, Daniel El Ouraoui, Pascal Fontaine

Univ. de Lorraine, CNRS, Inria, LORIA
Univ. Federal do Rio Grande do Norte
Vrije Universiteit Amsterdam

PxTP 2017

2017–09–23, Brasília, Brasil

Why proofs?

- check results for unsatisfiable/valid formulas
- export proofs into other tools

- debugging
- evaluate algorithms

- take part to the reasoning framework (e.g. conflict clauses)
- extract cores
- compute interpolants

But what kind of proofs?

Challenges for proofs in automated reasoning

- Collecting and storing proof information efficiently
  still an issue, lots of work in progress
  [KBT+16; HBR+15; KV13; Sch13; BODF09; WDF+09; Mos08; MB08; SZS04]
- Producing proofs for sophisticated processing techniques
  proofs with holes or too coarse
- Extract proofs for some decision procedures (e.g. CAD)
  still a research subject
- Standardizing a proof format
  open

Challenges for proofs in automated reasoning

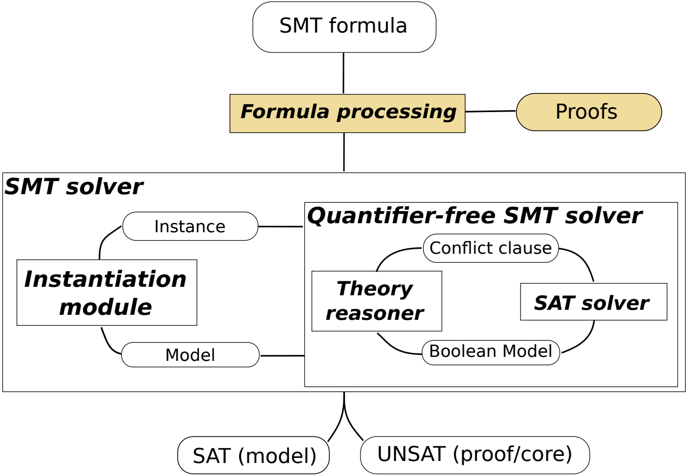
- Collecting and storing proof information efficiently
  still an issue, lots of work in progress
  [KBT+16; HBR+15; KV13; Sch13; BODF09; WDF+09; Mos08; MB08; SZS04]

- Producing proofs for sophisticated processing techniques
  ~~proofs with holes or too coarse~~ **scalable fine-grained proofs**

- Extract proofs for some decision procedures (e.g. CAD)
  still a research subject

- Standardizing a proof format
  open

SMT: the big picture



Proofs in SMT: main reasoning steps

Resolution chains, input formulas, tautologies for theory and quantifier reasoning

- SAT solver: resolution

$$\frac{A \vee \ell \qquad B \vee \bar{\ell}}{A \vee B}$$

Antecedents: $A \vee \ell$, $B \vee \bar{\ell}$
Pivot: ℓ or $\bar{\ell}$
Resolvent: $A \vee B = (A \vee \ell) \diamond (B \vee \bar{\ell})$

- theory solvers: theory lemmas

$$\neg(a \simeq c) \vee \neg(c \simeq b) \vee a \simeq b \qquad \neg(a \simeq b) \vee f(a) \simeq f(b)$$

$$\neg(y > 1) \vee \neg(x < 1) \vee y > x$$

- instantiation module: instantiation lemmas

$$\neg(\forall x.\, \psi[x]) \vee \psi[t]$$

Proving formula processing

- ⊖ Resolution does not capture all transformations
- ⊖ Some transformations do not preserve logical equivalence
- ⊖ Processing code is lengthy and deals with many cases
- ⊖ Difficult to manipulate binders soundly and efficiently

Proving formula processing

- ~~Resolution does not capture all transformations~~
- ~~Some transformations do not preserve logical equivalence~~
- ~~Processing code is lengthy and deals with many cases~~
- ~~Difficult to manipulate binders soundly and efficiently~~

Extensible framework to produce proofs for processing techniques involving *locally replacing equals by equals* in the presence of *binders*

Some instances:

Skolemization: $(\neg\,\forall x.\ \mathsf{p}(x)) \simeq \neg\,\mathsf{p}(\varepsilon x.\ \neg\,\mathsf{p}(x))$

let elimination: $(\mathrm{let}\ x \simeq \mathsf{a}\ \mathrm{in}\ \mathsf{p}(x,\ x)) \simeq \mathsf{p}(\mathsf{a},\ \mathsf{a})$

theory simplifications: $(\mathsf{k} + 1 \times 0 < \mathsf{k}) \simeq (\mathsf{k} < \mathsf{k})$

Inference system

A context Γ fixes a set of variables and specifies a substitution

substitution

$$\Gamma ::= \varnothing \mid \Gamma, x \mid \Gamma, \bar{x}_n \mapsto \bar{s}_n$$

bound variable

Inference system

A context Γ fixes a set of variables and specifies a substitution

$$\Gamma ::= \varnothing \mid \Gamma, x \mid \Gamma, \bar{x}_n \mapsto \bar{s}_n$$

substitution

bound variable

Rules have the form

derivations of premises

$$\frac{\mathcal{D}_1 \quad \cdots \quad \mathcal{D}_n}{\Gamma \rhd t \simeq u}\ \mathrm{R}$$

assumptions

transformation

- ▷ Semantically, the judgement expresses the equality of the terms $\Gamma(t)$ and u for all variables fixed by Γ

Inserting processing proofs into resolution proof

Now assertions may have different justifications other than tautologies and resolution:

$$\dfrac{\varphi \qquad \dfrac{\mathcal{D}}{\varphi \simeq \psi} \qquad \dfrac{}{\neg\,(\varphi \simeq \psi) \vee \neg\,\varphi \vee \psi}\ \textsc{Taut}_{\simeq}}{\psi}\ \textsc{Resolve}$$

in which $\mathcal{D}$ is the derivation of the processing of φ into ψ, which yields the conclusion $\varphi \simeq \psi$

Example of theory simplification

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\rhd\ \mathsf{k} \simeq \mathsf{k}}\ \textsc{Cong} \qquad \dfrac{}{\rhd\ 1\times 0 \simeq 0}\ \textsc{Taut}_\times}{\rhd\ \mathsf{k}+1\times 0 \simeq \mathsf{k}+0}\ \textsc{Cong} \qquad \dfrac{}{\rhd\ \mathsf{k}+0 \simeq \mathsf{k}}\ \textsc{Taut}_+}{\rhd\ \mathsf{k}+1\times 0 \simeq \mathsf{k}}\ \textsc{Trans} \qquad \dfrac{}{\rhd\ \mathsf{k} \simeq \mathsf{k}}\ \textsc{Cong}}{\rhd\ (\mathsf{k}+1\times 0 < \mathsf{k}) \simeq (\mathsf{k} < \mathsf{k})}\ \textsc{Cong}$$

Output skolemization

The skolemization proof of the formula $\neg \forall x.\, \mathsf{p}(x)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto \varepsilon x.\, \neg \mathsf{p}(x) \;\rhd\; x \simeq \varepsilon x.\, \neg \mathsf{p}(x)}\ \textsc{Refl}}{x \mapsto \varepsilon x.\, \neg \mathsf{p}(x) \;\rhd\; \mathsf{p}(x) \simeq \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x))}\ \textsc{Cong}}{\rhd\; (\forall x.\, \mathsf{p}(x)) \simeq \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x))}\ \textsc{Sko}_\forall}{\rhd\; (\neg \forall x.\, \mathsf{p}(x)) \simeq \neg \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x))}\ \textsc{Cong}$$

veriT syntax:

$$\begin{array}{l}
(.\mathrm{c0}\ (\mathsf{Sko_All}\ \ {:}\mathrm{conclusion}\ ((\forall x.\, \mathsf{p}(x)) \simeq \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x)))\\
\qquad\quad {:}\mathrm{args}\ (x \mapsto (\varepsilon x.\, \neg \mathsf{p}(x)))\\
\qquad\quad {:}\mathrm{subproof}\ ((.\mathrm{c1}\ (\mathsf{Refl}\ \ {:}\mathrm{conclusion}\ (x \simeq (\varepsilon x.\, \neg \mathsf{p}(x)))))\\
\qquad\qquad\qquad\quad (.\mathrm{c2}\ (\mathsf{Cong}\ \ {:}\mathrm{clauses}\ (.\mathrm{c1})\\
\qquad\qquad\qquad\qquad\qquad\quad {:}\mathrm{conclusion}\ (\mathsf{p}(x) \simeq \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x))))))))\\
(.\mathrm{c3}\ (\mathsf{Cong}\ \ {:}\mathrm{clauses}\ (.\mathrm{c0})\ \ {:}\mathrm{conclusion}\ ((\neg \forall x.\, \mathsf{p}(x)) \simeq \neg \mathsf{p}(\varepsilon x.\, \neg \mathsf{p}(x)))))
\end{array}$$

Example of ‘let’ expansion

$$
\dfrac{\dfrac{}{\rhd\ \mathsf{a} \simeq \mathsf{a}}\ \textsc{Cong} \qquad \dfrac{\dfrac{}{x \mapsto \mathsf{a}\ \rhd\ x \simeq \mathsf{a}}\ \textsc{Refl} \qquad \dfrac{}{x \mapsto \mathsf{a}\ \rhd\ x \simeq \mathsf{a}}\ \textsc{Refl}}{x \mapsto \mathsf{a}\ \rhd\ \mathsf{p}(x, x) \simeq \mathsf{p}(\mathsf{a}, \mathsf{a})}\ \textsc{Cong}}{\rhd\ (\text{let } x \simeq \mathsf{a} \text{ in } \mathsf{p}(x, x)) \simeq \mathsf{p}(\mathsf{a}, \mathsf{a})}\ \textsc{Let}
$$

matryoshka

SMT for HOL

Example: beta-reduction (1/2)

$$
\dfrac{
  \dfrac{}{\rhd\ \mathsf{a} \simeq \mathsf{a}}\ \textsc{Cong}
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{}{x \mapsto \mathsf{a}\ \rhd\ \mathsf{p} \simeq \mathsf{p}}\ \textsc{Refl}
      \quad
      \dfrac{}{x \mapsto \mathsf{a}\ \rhd\ x \simeq \mathsf{a}}\ \textsc{Refl}
    }{x \mapsto \mathsf{a}\ \rhd\ \mathsf{p}\,x \simeq \mathsf{p}\,\mathsf{a}}\ \textsc{Cong}
    \quad
    \dfrac{}{x \mapsto \mathsf{a}\ \rhd\ x \simeq \mathsf{a}}\ \textsc{Refl}
  }{x \mapsto \mathsf{a}\ \rhd\ \mathsf{p}\,x\,x \simeq \mathsf{p}\,\mathsf{a}\,\mathsf{a}}\ \textsc{Cong}
}{\rhd\ (\lambda x.\, \mathsf{p}\,x\,x)\,\mathsf{a}\ \simeq \mathsf{p}\,\mathsf{a}\,\mathsf{a}}\ \textsc{Beta}
$$

Example: beta-reduction (2/2)

$$
\dfrac{
  \dfrac{
    \dfrac{}{\Gamma_1 \;\rhd \mathsf{f} \simeq \mathsf{f}}\,\textsc{Refl}
    \quad
    \dfrac{}{\Gamma_1 \;\rhd x \simeq w}\,\textsc{Refl}
  }{\Gamma_1 \;\rhd \mathsf{f}\, x \simeq \mathsf{f}\, w}\,\textsc{Cong}
  \qquad
  \dfrac{
    \dfrac{}{\Gamma_2 \;\rhd y \simeq \mathsf{f}\, w}\,\textsc{Refl}
    \quad
    \dfrac{}{\Gamma_3 \;\rhd \mathsf{p}\, z \simeq \mathsf{p}\,(\mathsf{f}\, w)}\,\textsc{Refl}
  }{\Gamma_2 \;\rhd (\lambda z.\, \mathsf{p}\, z)\, y \simeq \mathsf{p}\,(\mathsf{f}\, w)}\,\textsc{Beta}
}{
  \dfrac{\Gamma_1 \;\rhd (\lambda y.\, (\lambda z.\, \mathsf{p}\, z)\, y)\,(\mathsf{f}\, x) \simeq \mathsf{p}\,(\mathsf{f}\, w)}{\rhd (\lambda x.\, (\lambda y.\, (\lambda z.\, \mathsf{p}\, z)\, y)\,(\mathsf{f}\, x)) \simeq (\lambda w.\, \mathsf{p}\,(\mathsf{f}\, w))}\,\textsc{Bind}
}\,\textsc{Beta}
$$

where

$$\Gamma_1 = w,\, x \mapsto w;\;\; \Gamma_2 = \Gamma_1,\, y \mapsto \mathsf{f}\, w;\text{ and }\; \Gamma_3 = \Gamma_2,\, z \mapsto \mathsf{f}\, w:$$

Proof-producing contextual recursion

function *process*(Δ, t)
  match t
    case x:
      return *build_var*(Δ, x)
    case $\mathsf{f}(\bar{t}_n)$:
      $\bar{\Delta}'_n \leftarrow (\mathit{ctx\_app}(\Delta, \mathsf{f}, \bar{t}_n, i))_{i=1}^n$
      return *build_app*$\big(\Delta, \bar{\Delta}'_n, \mathsf{f}, \bar{t}_n, (\mathit{process}(\Delta'_i, t_i))_{i=1}^n\big)$
    case $Qx.\, \varphi$:
      $\Delta' \leftarrow \mathit{ctx\_quant}(\Delta, Q, x, \varphi)$
      return *build_quant*(Δ, Δ', Q, x, φ, *process*(Δ', φ))
    case $\text{let } \bar{x}_n \simeq \bar{r}_n \text{ in } t'$:
      $\Delta' \leftarrow \mathit{ctx\_let}(\Delta, \bar{x}_n, \bar{r}_n, t')$
      return *build_let*(Δ, Δ', $\bar{x}_n$, $\bar{r}_n$, t', *process*(Δ', t'))

▷ Parameterized by a notion of context and plugin functions

Example of plugin functions for theory simplification

For simplifying $u + 0$ and $0 + u$ to u:

Context Γ is a list of variables

function *ctx_quant*(Γ, Q, x, φ)
  return Γ, x

function *build_quant*$(\Gamma, \Gamma', Q, x, \varphi, \psi)$
  apply$(\textsc{Bind}, 1, \Gamma, Qx.\,\varphi, Qx.\,\psi)$
  return $Qx.\,\psi$

function *build_app*$(\Gamma, \bar{\Gamma}'_n, \mathsf{f}, \bar{t}_n, \bar{u}_n)$
  apply$(\textsc{Cong}, n, \Gamma, \mathsf{f}(\bar{t}_n), \mathsf{f}(\bar{u}_n))$
  if $\mathsf{f}(\bar{u}_n)$ *has form* $u + 0$ *or* $0 + u$ **then**
    apply$(\textsc{Taut}_+, 0, \Gamma, \mathsf{f}(\bar{u}_n), u)$
    apply$(\textsc{Trans}, 2, \Gamma, \mathsf{f}(\bar{t}_n), u)$
    return u
  else
    return $\mathsf{f}(\bar{u}_n)$

Theoretical properties

- ▷ Soundness of inference rules proven through an encoding into simply typed λ-calculus

$$M \ ::= \ \boxed{t} \mid \lambda x.\, M \mid (\lambda \bar{x}_n.\, M)\, \bar{t}_n$$

$$\frac{\mathcal{D}_1 \qquad \cdots \qquad \mathcal{D}_n}{M \simeq N}\,\mathsf{R}$$

- ▷ Correctness of proof-producing contextual recursion algorithm
- ▷ Cost of proof production is linear and of proof checking is (almost) linear*

  * assuming suitable data structures

Implementation

Proof output for veriT

Framework implemented with a proof-producing contextual recursion algorithm

- ⊕ fine-grained proofs for most processing transformations
- ⊕ No negative impact on performance
- ⊕ More transformations in proof producing mode
- ⊕ Dramatic simplification of the code base

Prototype checker in Isabelle/HOL

Maps proofs into Isabelle theorems

- ⊕ Judgements encoded in λ-calculus

Conclusions

- ▷ Centralizes manipulation of bound variables and substitutions
- ▷ Accommodates many transformations (e.g. Skolemization)
- ▷ Proof checking is (almost) linear
- ▷ Implementation and integration within veriT

Conclusions

- ▷ Centralizes manipulation of bound variables and substitutions
- ▷ Accommodates many transformations (e.g. Skolemization)
- ▷ Proof checking is (almost) linear
- ▷ Implementation and integration within veriT

Future work

- ▷ Support global rewritings within the framework
- ▷ Implement mechanized proof reconstruction

References

Guy Katz, Clark W. Barrett, Cesare Tinelli, Andrew Reynolds, and Liana Hadarean. “Lazy proofs for DPLL(T)-based SMT solvers”. In: Formal Methods In Computer-Aided Design (FMCAD). Ed. by Ruzica Piskac and Muralidhar Talupur. IEEE, 2016, pp. 93–100.

Liana Hadarean, Clark W. Barrett, Andrew Reynolds, Cesare Tinelli, and Morgan Deters. “Fine Grained SMT Proofs for the Theory of Fixed-Width Bit-Vectors”. In: Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Ed. by Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov. Vol. 9450. Lecture Notes in Computer Science. Springer, 2015, pp. 340–355.

Laura Kovács and Andrei Voronkov. “First-Order Theorem Proving and Vampire”. English. In: Computer Aided Verification (CAV). Ed. by Natasha Sharygina and Helmut Veith. Vol. 8044. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 1–35.

Stephan Schulz. “System Description: E 1.8”. English. In: Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Ed. by Ken McMillan, Aart Middeldorp, and Andrei Voronkov. Vol. 8312. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 735–743.

References

Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. "veriT: An Open, Trustable and Efficient SMT-Solver". In: Proc. Conference on Automated Deduction (CADE). Ed. by Renate A. Schmidt. Vol. 5663. Lecture Notes in Computer Science. Springer, 2009, pp. 151–156.

Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. "SPASS Version 3.5". English. In: Proc. Conference on Automated Deduction (CADE). Ed. by RenateA. Schmidt. Vol. 5663. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 140–145.

Michał Moskal. "Rocket-Fast Proof Checking for SMT Solvers". In: Tools and Algorithms for Construction and Analysis of Systems (TACAS). Ed. by C. R. Ramakrishnan and Jakob Rehof. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 486–500.

Leonardo Mendonça de Moura and Nikolaj Bjørner. "Proofs and Refutations, and Z3". In: Logic for Programming, Artificial Intelligence, and Reasoning (LPAR) Workshops. Ed. by Piotr Rudnicki, Geoff Sutcliffe, Boris Konev, Renate A. Schmidt, and Stephan Schulz. Vol. 418. CEUR Workshop Proceedings. CEUR-WS.org, 2008.

References

Geoff Sutcliffe, Jürgen Zimmer, and Stephan Schulz. "TSTP Data-Exchange Formats for Automated Theorem Proving Tools". In: Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems. Ed. by Weixiong Zhang and Volker Sorge. Vol. 112. Frontiers in Artificial Intelligence and Applications. IOS Press, 2004, pp. 201–215.