Translating proofs from automated theorem provers to Logipedia

First Logipedia meeting

Guillaume Burel[1,2]

Wednesday January 23rd, 2019

[1]Samovar, ENSIIE, Université Paris Saclay

[2]Inria and LSV, CNRS and ENS Paris Saclay, Université Paris Saclay

Why could Logipedia be interested in Automated Theorem Provers?

- Import proofs from databases of problems
  - TPTP yes, at least problems produced by humans
  - Proof obligations from program verification probably not
- Helping proof assistants
  - automated Logipedia tactic
- Transfer
  - cannot automated everything
  - but can definitively help (see [Cauderlier 17])

Families of automated theorem provers

- SAT solvers
  - propositional logic
- SMT solvers
  - combining a SAT solver with decision procedure for particular theories
- FO theorem provers
  - many based on resolution/superposition
- HO provers
  - TH1 of TPTP $\simeq$ classical STT$\forall$

Most of them are for classical logic

Output of ATPs

- proof term
  - the ATP produces directly a Dedukti file
  - Zenon modulo, iProverModulo, ArchSat
- proof script
  - tree (DAG) of formulas;
  - each formula is a logical consequence of its parents
  - TSTP format (partially), DRUP format
- proof trace
  - evolving set of formulas
  - satisfiability is preserved
  - TSTP format (Skolemization, splitting), DRAT format

Resolution proofs

$$\text{Res. } \frac{P \vee C \qquad \neg Q \vee D}{\sigma(C \vee D)} \sigma = mgu(P,Q) \qquad \text{Fact. } \frac{P \vee Q \vee D}{\sigma(P \vee D)} \sigma = mgu(P,Q)$$

Proof trace from e.g. Prover9:

- which rule?
- which premises?
- which literals?
- which derived clause?

[Cauderlier 18] Dedukti tactic using metadedukti

- a program written in Dedukti
- produce Dedukti proof terms for each inference step

```
def C3 := resolution.resolve 0 2 C1 C2.
thm c3 : resolution.qcproof C3
  := resolution.resolve_correct 0 2 C1 C2 c1 c2.
```

TSTP

Proof format of the CADE community

List of formulas

- each annotated by an inference tree whose leafs are other formulas

```
cnf(c_0_60,plain,
    ( join(X1,join(X2,X3)) = join(X2,join(X1,X3)) ),
    inference(rw,[status(thm)],
      [inference(spm,[status(thm)],[c_0_30,c_0_18]),
       c_0_30])).
```

TSTP

Proof format of the CADE community

List of formulas

- each annotated by an inference tree whose leafs are other formulas

```
cnf(c_0_60,plain,
    ( join(X1,join(X2,X3)) = join(X2,join(X1,X3)) ),
    inference(rw,[status(thm)],
      [inference(spm,[status(thm)],[c_0_30,c_0_18]),
       c_0_30])).
```

Independent of the proof calculus

Proof calculus of E

- $sel(\mathcal{C}) \subseteq \mathcal{C}$.
- If $sel(\mathcal{C}) \cap \mathcal{C}^- = \emptyset$, then $sel(\mathcal{C}) = \emptyset$.

We say that a literal $\mathcal{L}$ is *selected* (with respect to a given selection function) in a clause $\mathcal{C}$ if $\mathcal{L} \in sel(\mathcal{C})$. ◀

We will use two kinds of restrictions on deducing new clauses: One induced by ordering constraints and the other by selection functions. We combine these in the notion of *eligible literals*.

Definition 3.1.2 (Eligible literals)
Let $\mathcal{C} = \mathcal{L} \vee \mathcal{R}$ be a clause, let σ be a substitution and let sel be a selection function.

- We say $\sigma(\mathcal{L})$ is *eligible for resolution* if either
  - $sel(\mathcal{C}) = \emptyset$ and $\sigma(\mathcal{L})$ is $>_L$-maximal in $\sigma(\mathcal{C})$ or
  - $sel(\mathcal{C}) \neq \emptyset$ and $\sigma(\mathcal{L})$ is $>_L$-maximal in $\sigma(sel(\mathcal{C}) \cap \mathcal{C}^-)$ or
  - $sel(\mathcal{C}) \neq \emptyset$ and $\sigma(\mathcal{L})$ is $>_L$-maximal in $\sigma(sel(\mathcal{C}) \cap \mathcal{C}^+)$.
- $\sigma(\mathcal{L})$ is *eligible for paramodulation* if $\mathcal{L}$ is positive, $sel(\mathcal{C}) = \emptyset$ and $\sigma(\mathcal{L})$ is strictly $>_L$-maximal in $\sigma(\mathcal{C})$.

◀

The calculus is represented in the form of inference rules. For convenience, we distinguish two types of inference rules. For *generating* inference rules, written with a single line separating preconditions and results, the result is added to the set of all clauses. For *contracting* inference rules, written with a double line, the result clauses are substituted for the clauses in the precondition. In the following, u, v, s and t are terms, σ is a substitution and R, S and T are (partial) clauses. p is a position in a term and λ is the empty or top-position. $D \subseteq F$ is a set of unused constant predicate symbols. Different clauses are assumed to not share any common variables.

Definition 3.1.3 (The inference system SP)
Let $>$ be a total simplification ordering (extended to orderings $>_L$ and $>_C$ on literals and clauses), let sel be a selection function, and let D be a set of fresh propositional constants. The inference system **SP** consists of the following inference rules:

- *Equality Resolution*:

$$\text{(ER)} \quad \frac{u \not\simeq v \vee R}{\sigma(R)}$$

if $\sigma = mgu(u, v)$ and $\sigma(u \not\simeq v)$ is eligible for resolution.

8

- *Superposition into negative literals*:

$$\text{(SN)} \quad \frac{s \simeq t \vee S \quad u \not\simeq v \vee R}{\sigma(u[p \leftarrow t] \not\simeq v \vee S \vee R)}$$

if $\sigma = mgu(u|_p, s)$, $\sigma(s) \not< \sigma(t)$, $\sigma(u) \not< \sigma(v)$, $\sigma(s \simeq t)$ is eligible for paramodulation, $\sigma(u \not\simeq v)$ is eligible for resolution, and $u|_p \notin V$.

- *Superposition into positive literals*:

$$\text{(SP)} \quad \frac{s \simeq t \vee S \quad u \simeq v \vee R}{\sigma(u[p \leftarrow t] \simeq v \vee S \vee R)}$$

if $\sigma = mgu(u|_p, s)$, $\sigma(s) \not< \sigma(t)$, $\sigma(u) \not< \sigma(v)$, $\sigma(s \simeq t)$ is eligible for paramodulation, $\sigma(u \simeq v)$ is eligible for resolution, and $u|_p \notin V$.

- *Simultaneous superposition into negative literals*

$$\text{(SSN)} \quad \frac{s \simeq t \vee S \quad u \not\simeq v \vee R}{\sigma(S \vee (u \not\simeq v \vee R)[u|_p \leftarrow t])}$$

if $\sigma = mgu(u|_p, s)$, $\sigma(s) \not< \sigma(t)$, $\sigma(u) \not< \sigma(v)$, $\sigma(s \simeq t)$ is eligible for paramodulation, $\sigma(u \not\simeq v)$ is eligible for resolution, and $u|_p \notin V$.

This inference rule is an alternative to (SN) that performs better in practice.

- *Simultaneous superposition into positive literals*

$$\text{(SSP)} \quad \frac{s \simeq t \vee S \quad u \simeq v \vee R}{\sigma(S \vee (u \simeq v \vee R)[u|_p \leftarrow t])}$$

if $\sigma = mgu(u|_p, s)$, $\sigma(s) \not< \sigma(t)$, $\sigma(u) \not< \sigma(v)$, $\sigma(s \simeq t)$ is eligible for paramodulation, $\sigma(u \not\simeq v)$ is eligible for resolution, and $u|_p \notin V$.

This inference rule is an alternative to (SP) that performs better in practice.

- *Equality factoring*:

$$\text{(EF)} \quad \frac{s \simeq t \vee u \simeq v \vee R}{\sigma(t \not\simeq v \vee u \simeq v \vee R)}$$

if $\sigma = mgu(s, u)$, $\sigma(t) \not> \sigma(s)$ and $\sigma(s \simeq t)$ eligible for paramodulation.

- *Rewriting of negative literals*:

$$\text{(RN)} \quad \frac{s \simeq t \quad u \not\simeq v \vee R}{s \simeq t \quad u[p \leftarrow \sigma(t)] \not\simeq v \vee R}$$

if $u|_p = \sigma(s)$ and $\sigma(s) > \sigma(t)$.

9

- *Rewriting of positive literals*[2]:

$$\text{(RP)} \quad \frac{s \simeq t \quad u \simeq v \vee R}{s \simeq t \quad u[p \leftarrow \sigma(t)] \simeq v \vee R}$$

if $u|_p = \sigma(s)$, $\sigma(s) > \sigma(t)$, and if $u \simeq v$ is not eligible for paramodulation or $v > u$ or $p \neq \lambda$.

- *Clause subsumption*:

$$\text{(CS)} \quad \frac{C \quad \sigma(C \vee R)}{C}$$

where C and R are arbitrary (partial) clauses and σ is a substitution.

- *Equality subsumption*:

$$\text{(ES)} \quad \frac{s \simeq t \quad u[p \leftarrow \sigma(s)] \simeq u[p \leftarrow \sigma(t)] \vee R}{s \simeq t}$$

- *Positive simplify-reflect*[3]:

$$\text{(PS)} \quad \frac{s \simeq t \quad u[p \leftarrow \sigma(s)] \not\simeq u[p \leftarrow \sigma(t)] \vee R}{s \simeq t \quad R}$$

- *Negative simplify-reflect*

$$\text{(NS)} \quad \frac{s \not\simeq t \quad \sigma(s) \not\simeq \sigma(t) \vee R}{s \not\simeq t \quad R}$$

- *Tautology deletion*:

$$\text{(TD)} \quad \frac{C}{}$$

if C is a tautology[4]

[2] A stronger version of (RP) is proven to maintain completeness for Unit and Horn problems and is generally believed to maintain completeness for the general case as well [Bac98]. However, the proof of completeness for the general case seems to be rather involved, as it requires a very different clause ordering than the one introduced [BG94], and we are not aware of any existing proof in the literature. The variant rule allows rewriting of maximal terms of maximal literals under certain circumstances:

$$\text{(RP')} \quad \frac{s \simeq t \quad u \simeq v \vee R}{s \simeq t \quad u[p \leftarrow \sigma(t)] \simeq v \vee R}$$

if $u|_p = \sigma(s)$, $\sigma(s) > \sigma(t)$ and if $u \simeq v$ is not eligible for paramdulation or $u \not> v$ or $p \neq \lambda$ or σ is not a variable renaming.

This stronger rule is implemented successfully by both E and SPASS [Wei99].

[3] In practice, this rule is only applied if $\sigma(s)$ and $\sigma(t)$ are $>$-incomparable – in all other cases this rule is subsumed by (RN) and the deletion of resolved literals (DR).

10

- *Deletion of duplicate literals*:

$$\text{(DD)} \quad \frac{s \simeq t \vee s \simeq t \vee R}{s \simeq t \vee R}$$

- *Deletion of resolved literals*:

$$\text{(DR)} \quad \frac{s \not\simeq s \vee R}{R}$$

- *Destructive equality resolution*:

$$\text{(DE)} \quad \frac{x \not\simeq y \vee R}{\sigma(R)}$$

if $x, y \in V, \sigma = mgu(x, y)$

- *Contextual literal cutting*:

$$\text{(CLC)} \quad \frac{\sigma(C \vee R \vee s \dot{\simeq} t) \qquad C \vee \overline{s \dot{\simeq} t}}{\sigma(C \vee R) \qquad C \vee \overline{s \dot{\simeq} t}}$$

where $\overline{s \dot{\simeq} t}$ is the negation of $s \dot{\simeq} t$ and σ is a substitution

This rule is also known as *subsumption resolution* or *clausal simplification*.

- *Condensing*:

$$\text{(CON)} \quad \frac{l_1 \vee l_2 \vee R}{\sigma(l_1 \vee R)}$$

if $\sigma(l_1) = \sigma(l_2)$ and $\sigma(l_1 \vee R)$ subsumes $l_1 \vee l_2 \vee R$

- *Introduce definition*[5]

$$\text{(ID)} \quad \frac{R \vee S}{d \vee R \quad \neg d \vee S}$$

if R and S do not share any variables, $d \in D$ has not been used in a previous definition and R does not contain any symbol from D

- *Apply definition*

$$\text{(AD)} \quad \frac{\sigma(d \vee R) \quad R \vee S}{\sigma(d \vee R) \quad \neg d \vee S}$$

if σ is a variable renaming, R and S do not share any variables, $d \in D$ and R does not contain any symbol from D

[4] This rule can only be implemented approximately, as the problem of recognizing tautologies is only semi-decidable in equational logic. Current versions of E try to detect tautologies by checking if the ground-completed negative literals imply at least one of the positive literals, as suggested in [NN93].

[5] This rule is always exhaustively applied to any clause, leaving n split-off clauses and one final link clause of all negative propositions.

11

Proof reconstruction

Use information from the proof trace to guide proof building

Inspired by Sledgehammer and PRocH

Two approaches:

- premises selector
- trace steps reconstruction

Premises selector

Problems can contain many axioms

- (especially if they come from ITP in a huge development)

Proofs found by ATP only use a few of them

Use the trace to know which axioms are actually needed
Reconstruct the proof from scratch using only these axioms

- In a Dedukti-producing ATP

Premises selection, experimental results

[Pham 2016]:
Fork of Zenon modulo, reads a TSTP file and keep only needed axioms

On 12467 FO problems of the TPTP library:

	Zenon modulo (alone)	E prover	Premises selection + Zenon modulo
#Problems solved	2274	8901	3249
%	18.2	71.3	26.0

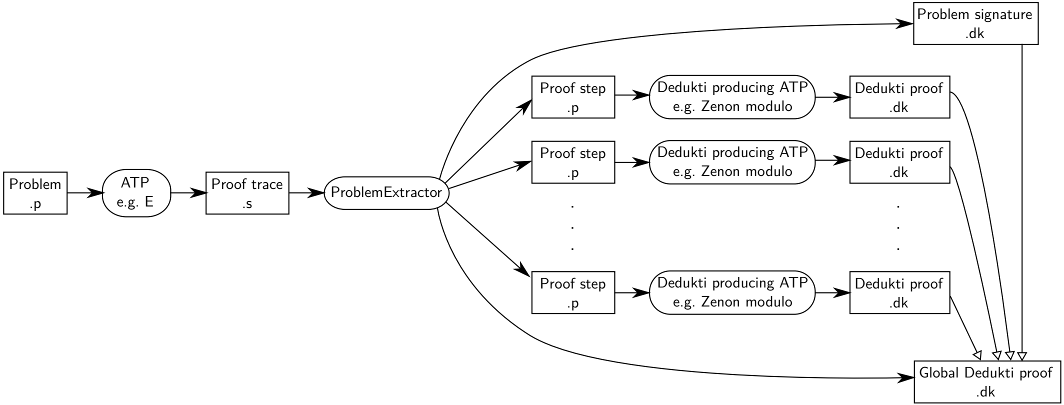
Proof step reconstruction

Axiom selection not enough, need to rebuild each proof steps

Part of Yacine El Haddad PhD thesis (ongoing work)

- agnostic wrt the proof calculus
- agnostic wrt the proof-producing reconstructor

Architecture



Remark

The structure of the original trace is kept in the global Dedukti proof:

```
cnf(c_0_60,plain,
    ( join(X1,join(X2,X3)) = join(X2,join(X1,X3)) ),
    inference(rw,[status(thm)],
      [inference(spm,[status(thm)],[c_0_30,c_0_18]),
       c_0_30])).
```

```
let c_0_18 : ... = ...
let c_0_30 : ... = ...
...
let c_0_60 : P (eq(join(X1,join(X2,X3)), join(X2,join(X1,X3)))) =
    c_0_40.goal c_0_30 c_0_18 c_0_30 in x...
```

Proofs Modulo Theory

SMT solver VeriT

Proof traces:

- logical steps
- theory "axioms"
  - formulas valid in the theory
  - generated by the theory reasoner (learned lemma)

Verine: translation to Dedukti [Gilbert 15]

- Logical steps can be easily translated
- Needs theory specific Dedukti-proof producing solver
  - ArchSat? Coq(Omega) + translation?

SAT solving

De facto standard for SAT solvers: DRAT

List of clauses

- each new clause preserves satisfiability of preceding ones
    - using a criterion called Reverse Asymmetric Tautology
- Deletion : indicates which clauses are no longer needed

New clauses may not be logical consequences of preceding ones!

- think of Skolemization in FOL

Proof transformation

Satisfiability preservation:

Γ has a model $\Rightarrow \Gamma, C$ has a model

Provability preservation:

$\Gamma, C \vdash \bot$ has a proof $\Rightarrow \Gamma \vdash \bot$ has a proof

1. Start from $\overline{\Gamma, \bot \vdash \bot}$
2. Transform proof until $Axioms \vdash \bot$

RAT criterion leads to a algorithm to transform proofs

- using auxiliary clauses

Limits of proof transformation

Start from the end of the trace

- Cannot benefit from deletion information

Can be adapted to follow the trace in the right order,

but produces too many unneeded auxiliary clauses

Extended Resolution

Fortunately, [Kiesl et al. 2018]:

- Extended resolution simulates DRAT

Extended resolution [Tseitin 1968]:

- resolution + definitions of new propositional variables
- Easily expressible in Dedukti

The translation from DRAT to what we need of Extended resolution can be performed in quadratic time (better in practice)

Questions

1. Constructivization
2. Which automatically found proofs do we want in Logipedia?
3. Is it possible to present them so that export out of Logipedia look nice?
4. How much can ATPs help in concept alignment?
   - see also nitpick