SC-TPTP: an Extension of the TPTP Format for First-Order Sequent-Based Proofs

Julie Cailler (joint work with Simon Guilloud — EPFL)

VeriDis Team
University of Lorraine
CNRS, INRIA, LORIA

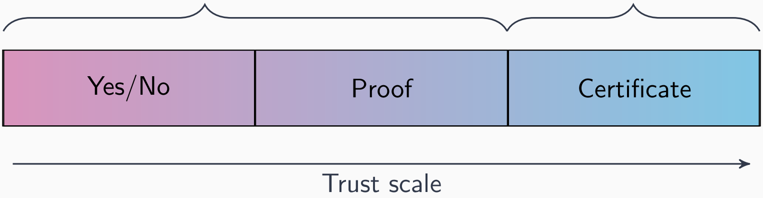
Proofs and Computers

Automated Theorem Proving

- Click-and-proof software
- Automatically search for a proof
- Output a statement or a proof-like trace

Interactive Theorem Proving

- Proof assistants
- Guide humans towards proofs
- Proof certificate



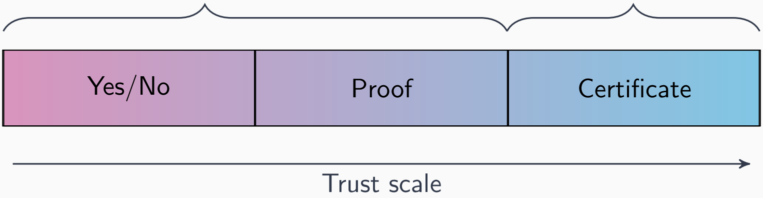
Proofs and Computers

Automated Theorem Proving

- Click-and-proof software
- Automatically search for a proof
- Output a statement or a proof-like trace

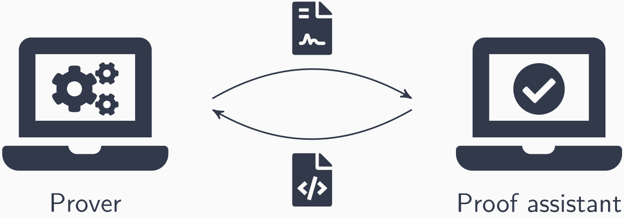
Interactive Theorem Proving

- Proof assistants
- Guide humans towards proofs
- Proof certificate

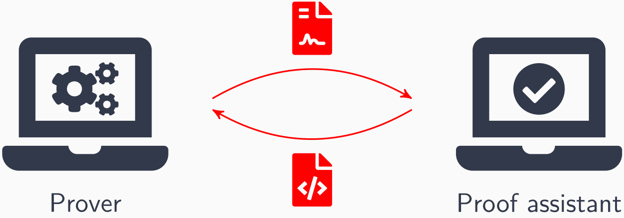


The best of both worlds: communication is the key!

Proof Transfers



Proof Transfers

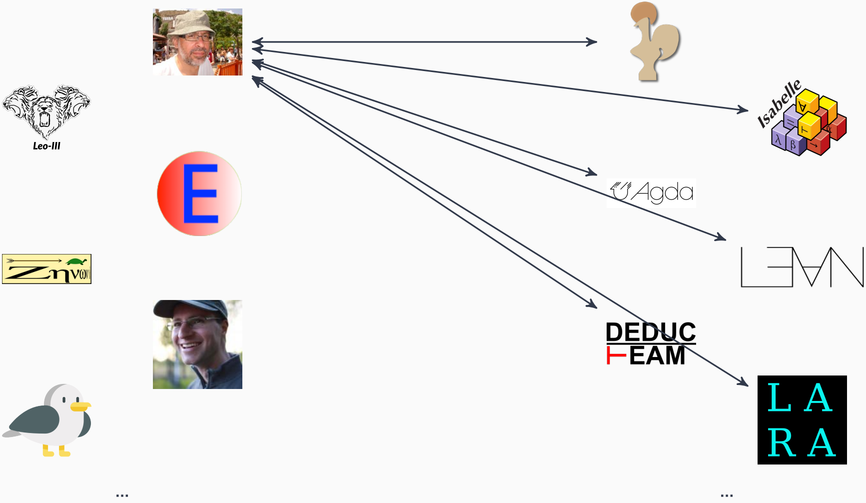


Proof Transfers (FOL)

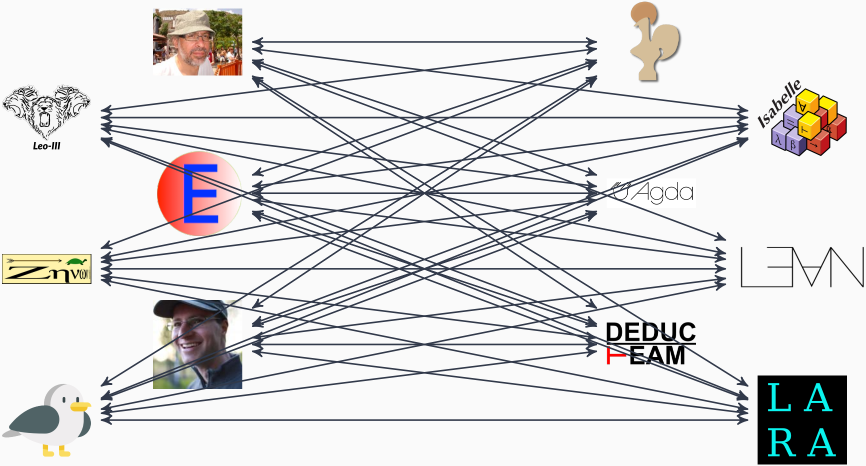
...

...

Proof Transfers (FOL)

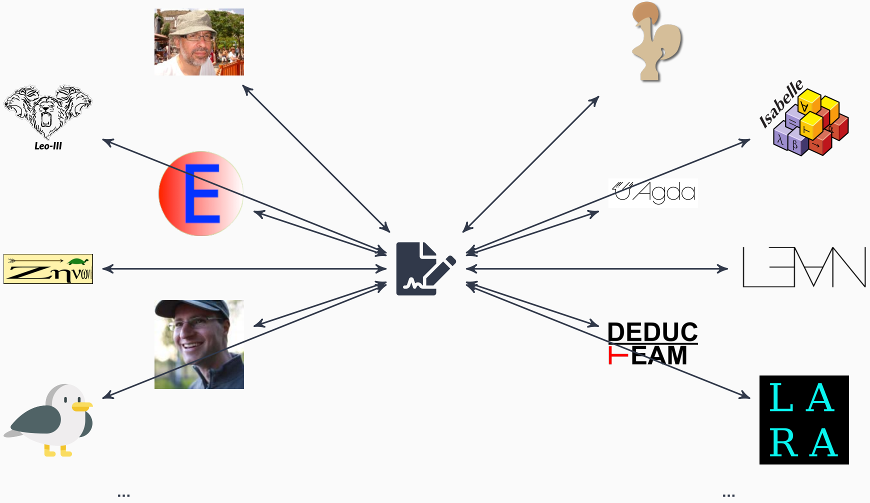


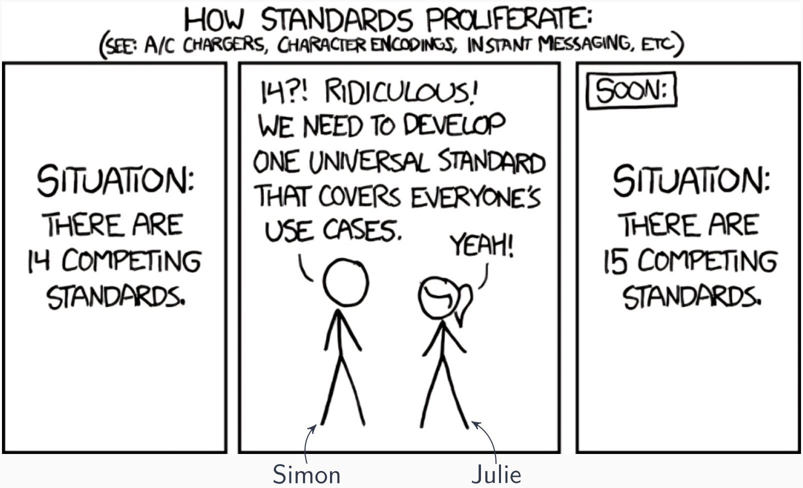
Proof Transfers (FOL)



... ...

Proof Transfers (FOL)





Credit: xkcd (`https://xkcd.com/`)

A “Good” Format?

Requirements

- Simple
- Human-readable
- Based on established format
- Well documented and specified
- Extendable
- Efficiently verifiable

Challenges

- Different foundations
- Multiple techniques
- Granularity

State of the Art

Other Communities

- SAT: DRAT
- SMT: LFSC, Z3, Alethe

State of the Art

Other Communities

- SAT: DRAT
- SMT: LFSC, Z3, Alethe

And for FOL?

- Dedukti/LambdaPi
  - Handle any foundation
  - Outputs toward multiple proof assistants
  - Hard to parse/import
  - Not widely adopted (yet!)
- TPTP/TSTP derivation format
  - Standard well-established input format
  - Easy syntax
  - Annotations for specific cases
  - ... No formally defined rules for sequent-based calculus

State of the Art

Other Communities

- SAT: DRAT
- SMT: LFSC, Z3, Alethe

And for FOL?

- Dedukti/LambdaPi
  - Handle any foundation
  - Outputs toward multiple proof assistants
  - Hard to parse/import
  - Not widely adopted (yet!)
- TPTP/TSTP derivation format
  - Standard well-established input format
  - Easy syntax
  - Annotations for specific cases
  - ... No formally defined rules for sequent-based calculus

TPTP

The TPTP World

- Geoff Sutcliffe
- Automated Theorem Proving (ATP) systems
- Problem library
- Solution library
- Language
- Ontologies
- Online services: problems generator, ATP host, ...
- Events: CASC, TPTP Tea Party, World Tour, ...
- More about the TPTP world: `https://www.tptp.org/`

The TPTP World

- Geoff Sutcliffe
- Automated Theorem Proving (ATP) systems
- Problem library
- Solution library
- Language
- Ontologies
- Online services: problems generator, ATP host, ...
- Events: CASC, TPTP Tea Party, World Tour, ...
- More about the TPTP world: `https://www.tptp.org/`

TPTP Language

Annotated Formulas

- First order (fof), higher order (thf), clausal form (cnf), typed (tff), ...
- Scheme:
  - name
  - role
  - formula
  - annotation

```
fof(<name>, <formula_role>, <fof_formula>, <annotations>).

fof(f1, conjecture, ((a => b) => (~a | b)), source_file).
```

Proofs in TPTP

Derivation

- List of *annotated formulas*
- Annotations are `<inferences>`

Inference

- Annotation of the formula
- Information about the rule applied
- Reference to the parent(s)

```
inference(<inference_rule>, <useful_info>, <inference_parents>)

    fof(f2, negated_conjecture, (~((a => b) => (~a | b))),
        inference(negated_conjecture, [status(cth)], [f1])).
    fof(f1, conjecture, ((a => b) => (~a | b)), source_file).
```

Example (Resolution)

Prove This!

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

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

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

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

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

$$\neg a \vee b \qquad a \qquad \neg b$$

$$b$$

$$\square$$

Example (TPTP)

Prove This!

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

```
fof(f9, plain, ($false), inference(subsumption_resolution, [status(thm)], [f7, f8])).
fof(f8, plain, (b), inference(subsumption_resolution, [status(thm)], [f5, f6])).
fof(f7, plain, (~b), inference(cnf_transformation, [status(esa)], [f4])).
fof(f6, plain, (a), inference(cnf_transformation, [status(esa)], [f4])).
fof(f5, plain, (~a | b), inference(cnf_transformation, [status(esa)], [f4])).
fof(f4, plain, ((~a | b) & a & ~b), inference(flattening, [status(thm)], [f3])).
fof(f3, plain, ((~a | b) & (a & ~b)),
    inference(NNF_transformation, [status(esa)], [f2])).
fof(f2, negated_conjecture, (~((a => b) => (~a | b))),
    inference(negated_conjecture, [status(cth)], [f1])).
fof(f1, conjecture, ((a => b) => (~a | b)), source_file).
```

SC-TPTP

Why Sequents?

- Original formula
- Proofs readily translatable into machine-checkable ones
- Works on non-classical logics
- Currently missing
- Your current speaker's favorite method :)

Sequent Calculus

Sequent Calculus

- $h_1, ..., h_n \vdash c_1, ..., c_m$
- Set on inference rules
- One- or two-sided
- Formulas stay in the branch

$$\frac{\Gamma, h'_1, \ldots, h'_{n'} \vdash c'_1, \ldots, c'_{m'} \Delta}{\Gamma, h_1, \ldots, h_n \vdash c_1, \ldots, c_m \Delta} \text{ Rule}$$

$$\frac{}{\Gamma, A \vdash A, \Delta} \text{ Axiom}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \text{ Left And}$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Sigma \vdash B, \Pi}{\Gamma, \Sigma \vdash A \wedge B, \Delta, \Pi} \text{ Right And}$$

Sequent Calculus

Sequent Calculus

- $h_1, ..., h_n \vdash c_1, ..., c_m$
- Set on inference rules
- One- or two-sided
- Formulas stay in the branch

Prove This!

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \Rightarrow b, \neg a \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\text{ Ax.} \qquad \dfrac{}{a \Rightarrow b, b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\text{ Ax.}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\text{ Left Imp.}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b}\text{ Right Or}}{\vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b)}\text{ Right Imp.}$$

Sequent in TPTP

FOFX

- Two lists of formulas (one per sequent side)
- Separated by `-->`
- First oder (FOFX) and typed first-order (TXF)
- "Not yet in use" ☹

```
fof(<name>, <formula_role>, [<fof_formula_list>] --> [<fof_formula_list>],
    <annotations>).

fof(f0, conjecture, [] --> [((a => b) => (~a | b))], source_file).
```

Inference Rules for Sequent Calculus (Level 1)

Level-1 Rules

- One- and two-sided sequent calculus (left and right)
- Basic unit step
- Premises and parameters

Example: Left Or

$$\frac{\Gamma, A \vdash \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \lor B \vdash \Delta, \Pi}$$

Rule Specifications

- 2 premises
- 2 parameters: `status(thm)` and index of $A \lor B$ on the left

```
fof(f2, plain, [a | b, b] --> [], ...).
fof(f1, plain, [a | b, a] --> [], ...).
fof(f0, plain, [a | b] --> [],
inference(leftOr, [status(thm), 0], [f1, f2])).
```

Inference Rules for Sequent Calculus (Level 1)

Example: Right Substitution

$$\frac{\Gamma, t = u \vdash P(t), \Delta}{\Gamma, t = u \vdash P(u), \Delta}$$

Rule Specifications

- 1 premise
- 4 parameters:
  - `status(thm)`
  - `i:Int`: index of $t = u$ on the left
  - `P(Z):Var`: shape of the predicate on the right
  - `Z:Var`: unifiable sub-term in the predicate

```
fof(f1, plain, [a = b] --> [P(a)], ...).
fof(f0, plain, [a = b] --> [P(b)],
inference(rightSubst, [status(thm), 0, P(X), X], [f1])).
```

Inference Rules for Sequent Calculus (Level 2)

Level-2 Rules

- More advanced reasoning steps
- Can be unfolded into level-1 rules
- Better interactions between tools (e.g., congruence, negated normal form, multiple substitutions)

Example: Congruence

$$\frac{}{\Gamma, P(u) \vdash P(t), \Delta}$$

Rule Specifications

- No premise
- 1 parameter: status(thm)
- Γ contains a set of equalities such that t and u are equals

```
fof(f0, assumption, [a = b, b = c, c = d, P(a)] --> [P(d)]
        inference(congruence, [status(thm)], [])).
```

Inference Rules for Sequent Calculus (Level 2)

Example: Multiple Right Substitutions

$$\frac{\Gamma \vdash P(t_1, ..., t_n), \Delta}{\Gamma \vdash P(u_1, ..., u_n), \Delta}$$

Rule Specifications

- 1 premise
- 4 parameters:
  - `status(thm)`
  - `[`$\mathtt{i}_1$`, ..., `$\mathtt{i}_n$`:Int]`: index of $t_j = u_j$ on the left
  - `P(`$\mathtt{Z}_1$`, ..., `$\mathtt{Z}_n$`):Term`: shape of the formula on the right
  - `[`$\mathtt{Z}_1$`, ..., `$\mathtt{Z}_n$`:Var]`: variables indicating where to substitute

```
fof(f1, plain, [a = b, c = d] --> [Q(a, c, d)], ...).
fof(f0, plain, [a = b, c = d] --> [Q(b, d, d)], inference(
rightSubstMulti, [status(thm), [0, 1], Q(X, Y, d), [X, Y]], [f1])).
```

Inference Rules for Sequent Calculus (Level 3 & 4)

Level-3 Rules

- Steps that we can be verified (with an implemented function)
- Translation or external tool

Level-4 Rules

- Unknown/trusted steps

Example (Sequent Calculus)

Prove This!

$$(a \Rightarrow b) \Rightarrow (\neg a \lor b)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \Rightarrow b, \neg a \vdash (a \Rightarrow b) \Rightarrow (\neg a \lor b), \neg a \lor b, \neg a, b}\text{Ax.} \qquad \dfrac{}{a \Rightarrow b, b \vdash (a \Rightarrow b) \Rightarrow (\neg a \lor b), \neg a \lor b, \neg a, b}\text{Ax.}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \lor b), \neg a \lor b, \neg a, b}\text{Left Imp.}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \lor b), \neg a \lor b}\text{Right Or}}{\vdash (a \Rightarrow b) \Rightarrow (\neg a \lor b)}\text{Right Imp.}$$

Example (SC-TPTP)

Prove This!

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

```
fof(f4, assumption, [(a => b), b] --> [((a => b) => (~a | b)), (~a | b), ~a, b],
    inference(hyp, [status(thm), 1, 3], [])).
fof(f3, assumption, [(a => b), ~a] --> [((a => b) => (~a | b)), (~a | b), ~a, b],
    inference(hyp, [status(thm), 1, 2], [])).
fof(f2, plain, [(a => b)] --> [((a => b) => (~a | b)), (~a | b), ~a, b],
    inference(leftImp, [status(thm), 0], [f3, f4])).
fof(f1, plain, [(a => b)] --> [((a => b) => (~a | b)), (~a | b)],
    inference(rightOr, [status(thm), 1], [f2])).
fof(f0, plain, [] --> [((a => b) => (~a | b))],
    inference(rightImp, [status(thm), 0], [f1])).
fof(my_conjecture, conjecture, ((a => b) => (~a | b))).
```

Utilities and Use Case

SC-TPTP Utilities

Proof Checker

Check the correctness of the proof steps w.r.t. the SC-TPTP format.

Level-2 Steps Unfold

Proof improvement by unfolding level-2 proof steps (congruence with e-graph, multiple substitutions, ...)

`Coq` Output

Provide verified proofs in `Coq` (lemmas file, context, ...)

`Egg` Elaboration Steps

Equality steps are explained by `Egg`, translated into SC-TPTP, and added to the proof.

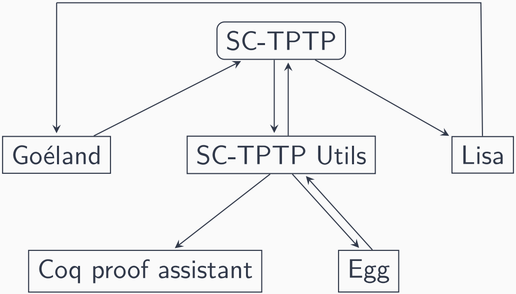
Use Case: Interactions between Goéland and Lisa

Goéland

- Automated theorem prover
- First-order logic
- Method of analytics tableaux
- Concurrent proof-search procedure

Lisa

- Proof assistant
- First-order logic
- Set-theoretic foundations
- Sequent-based proof system



Work in Progress and More

Calculus

- Resolution/superposition
- Connection calculus

New Compatible Tools

- Prover9
- Connect++
- Princess
- Isabelle
- LambdaPi

Proof Transformation Steps

- Clausification (Tseitin)
- Skolemization

Conclusion

SC-TPTP

- An extension of the TPTP derivation format to handle sequent-based calculus (LJ, LK, Tableaux, GS3, ...)
- Library of utilities

Future Work

- Add compatible ATP/ITP/Outputs
- Extension to Typed eXtended first-order Form (TXF)
- Shorter proofs (via `Let`)
- Theory management

A Standard Output Format!

- Verify CASC solutions
- New verification competition
- Make research and life easier
- Other communities have done it!

Thank you! ☺

https://github.com/SC-TPTP/sc-tptp