Representing First-Order Logic Sequent-Style Derivations in TPTP with SC-TPTP

13th TPTP Tea Party

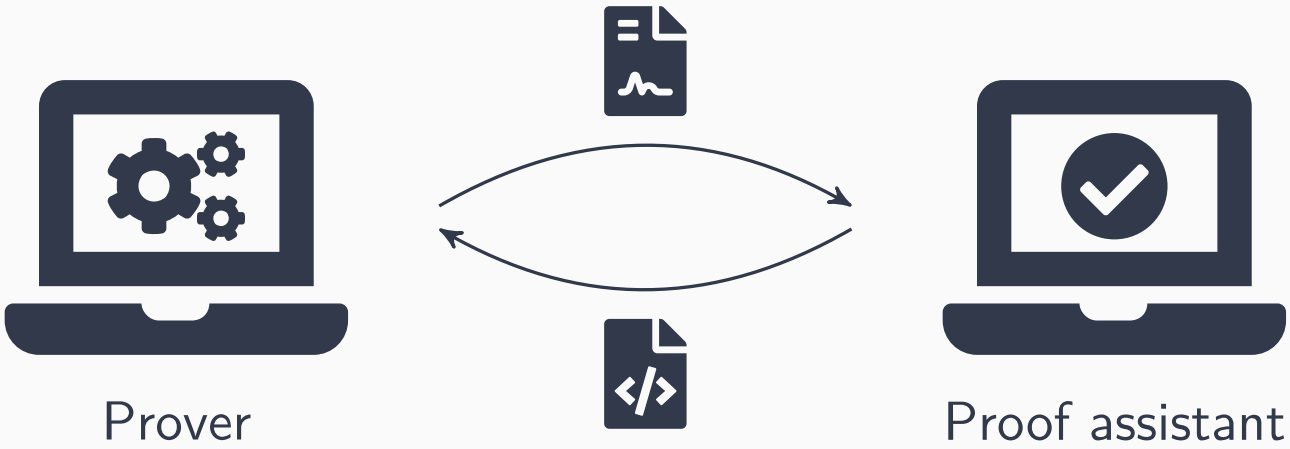
Julie Cailler and Simon Guilloud

Chair of Theoretical Computer Science
University of Regensburg
Germany

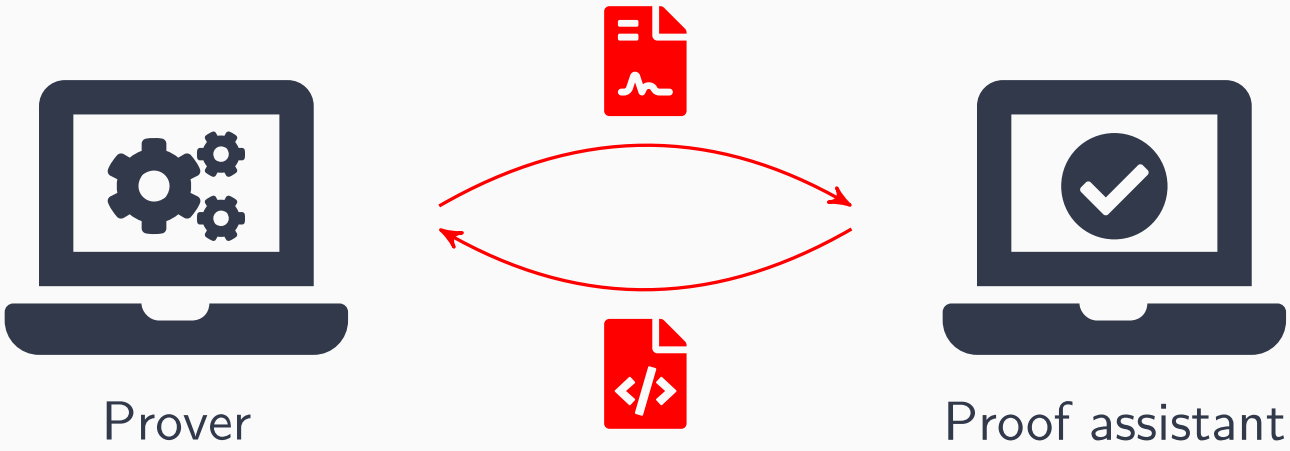
July 1, 2024

Universität Regensburg

Proof Transfers

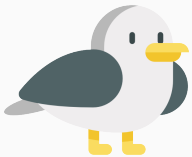


Proof Transfers

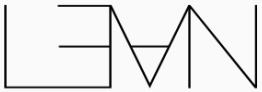


But...

Proof Transfers

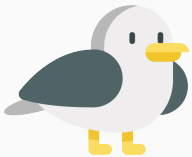


...

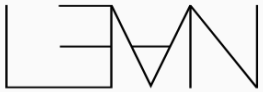


...

Proof Transfers

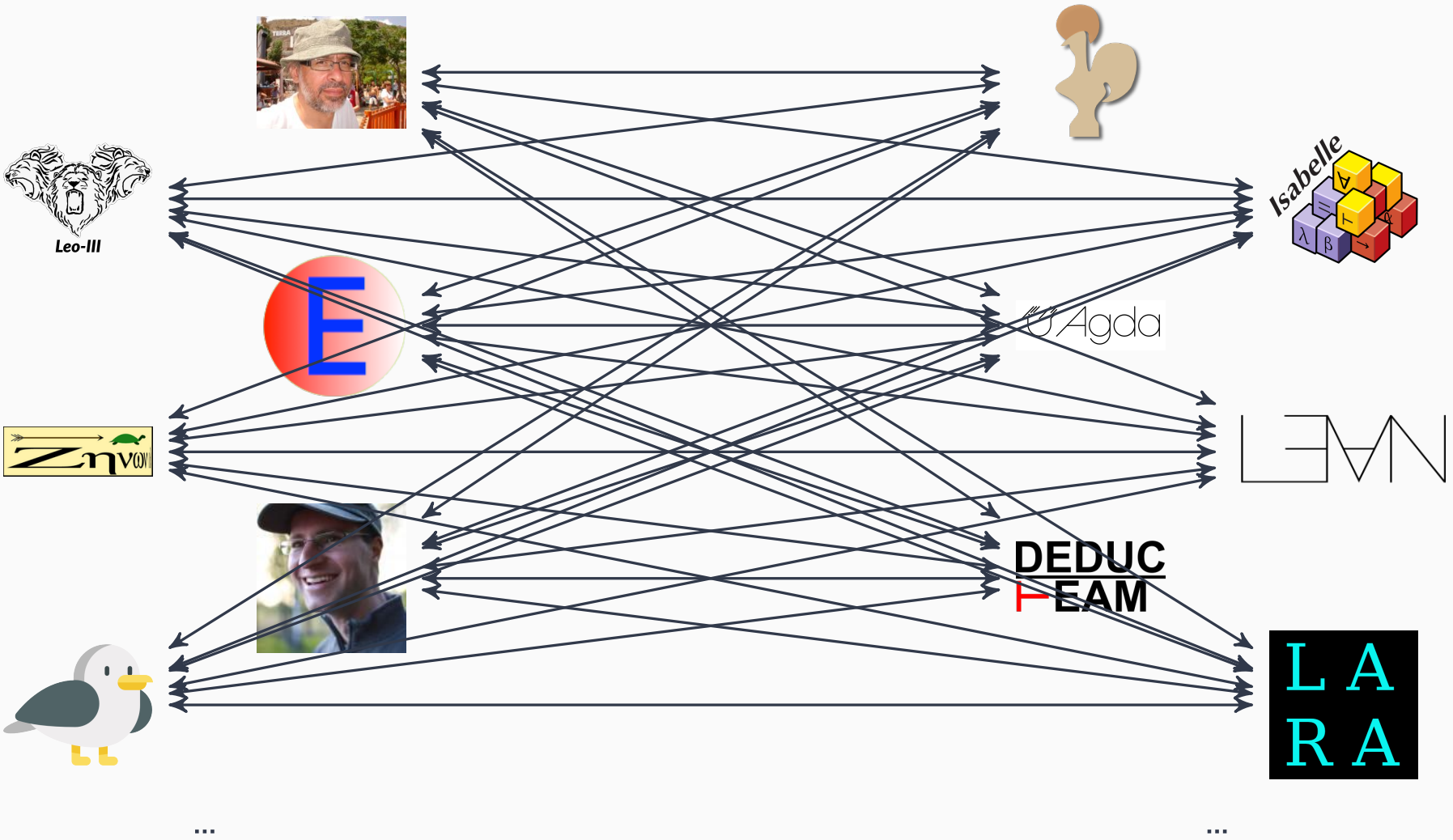


...

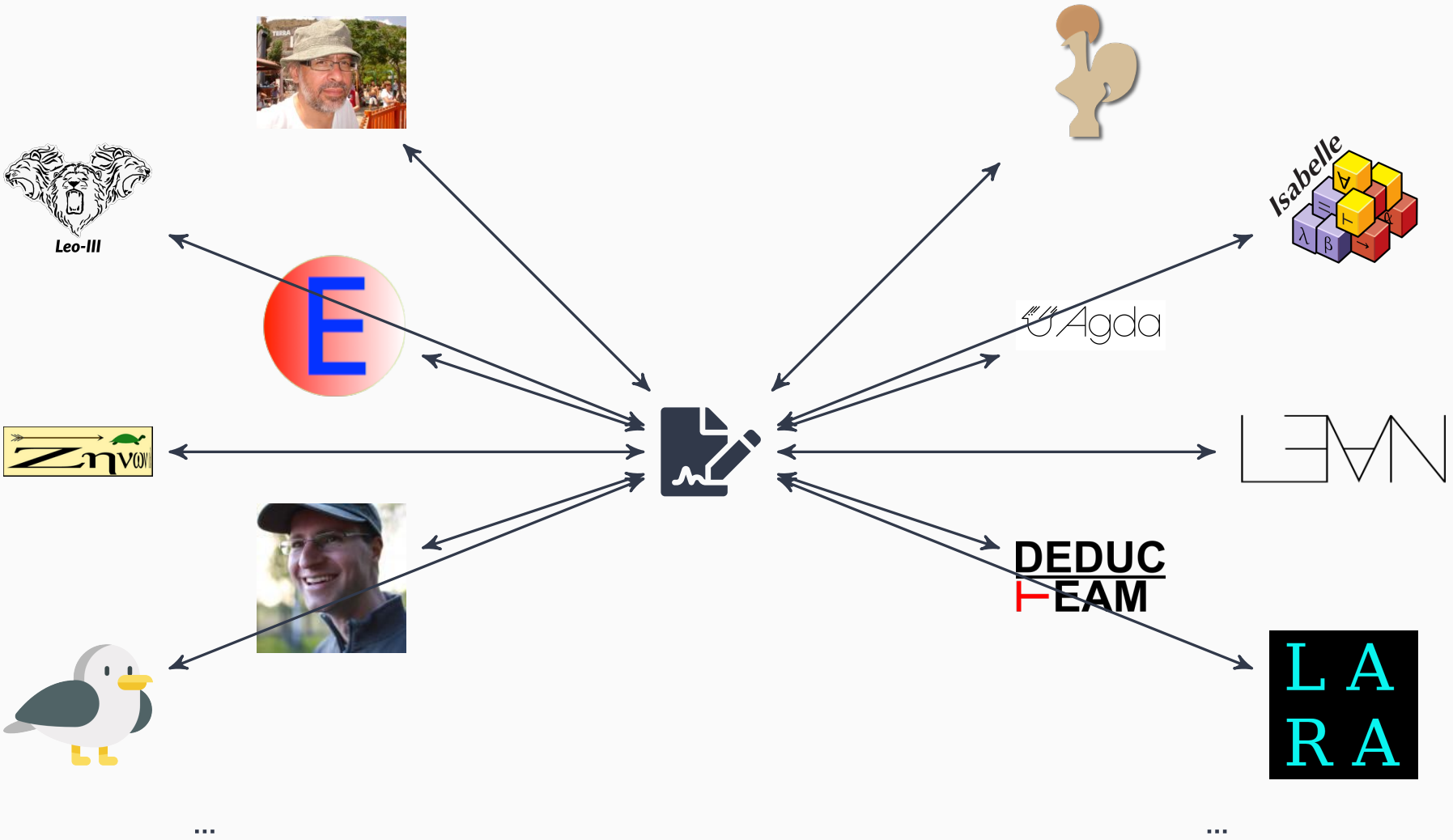


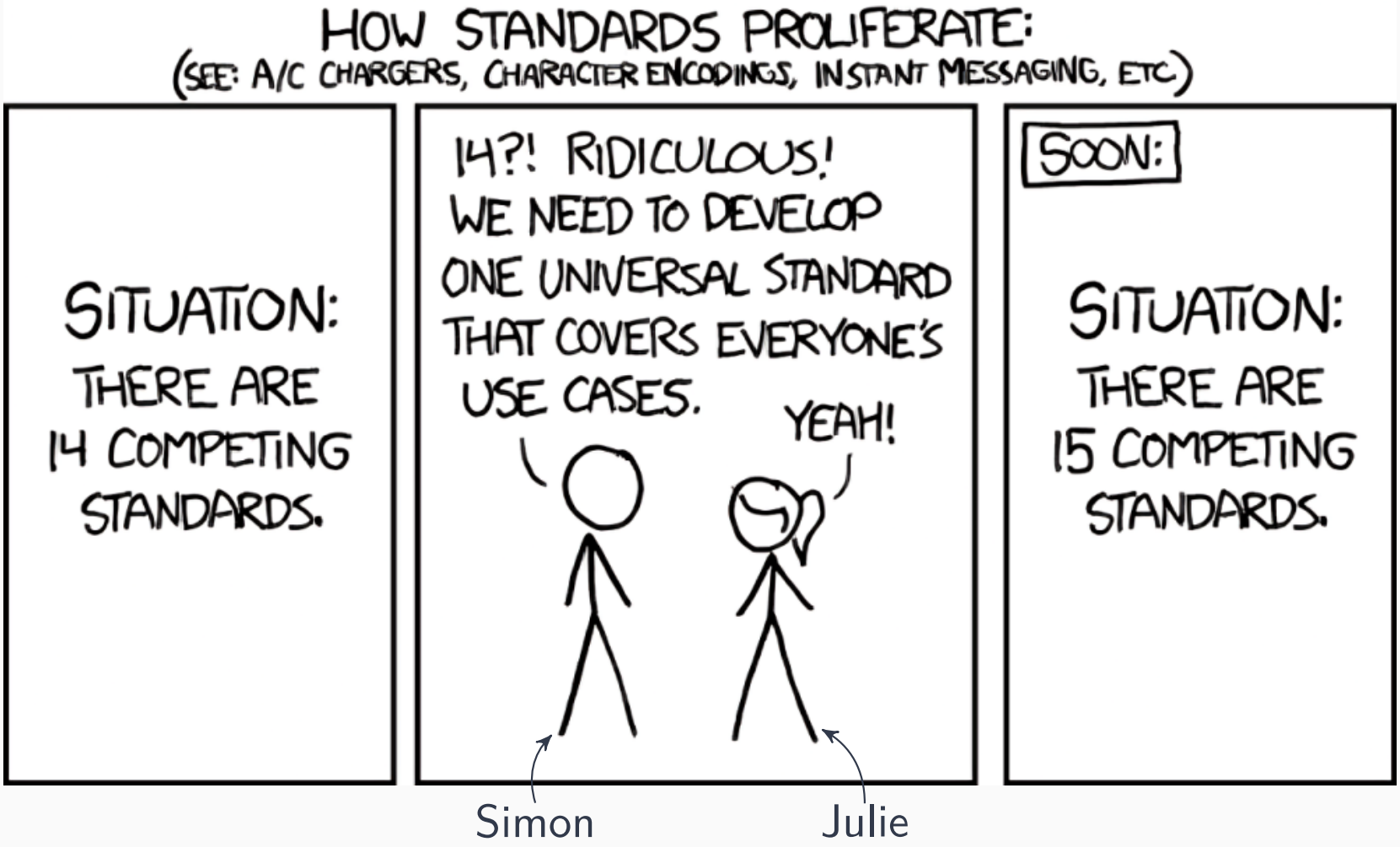
...

Proof Transfers



Proof Transfers





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

State of the Art

Other Communities

- SAT: DRAT
- SMT: LFSC, Z3, Alethe (work in progress)

State of the Art

Other Communities

- SAT: DRAT
- SMT: LFSC, Z3, Alethe (work in progress)

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 (work in progress)

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

Sequent Calculus

Sequent Calculus

- $h_1, ..., h_n \vdash c_1, ..., c_m$
- Set on inference rules
- One- or two-sided
- Proofs readily translatable toward ITP
- Works on non-classical logics
- Your current speaker's favorite method :)

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

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

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

Sequent 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_sequent> ::= <fof_formula_tuple> --> <fof_formula_tuple>
                          | (<fof_sequent>)
    <fof_formula_tuple> ::= []
                          | [<fof_formula_tuple_list>]
<fof_formula_tuple_list> ::= <fof_logic_formula>
                          | <fof_logic_formula>, <fof_formula_tuple_list>
```

Proofs in TPTP

Derivation

List of *annotated formulas*

```
<fof_annotated> ::= fof(<name>, <formula_role>, <fof_formula>, <annotations>).
 <formula_role> ::= assumption | axiom | conjecture | plain
  <fof_formula> ::= <fof_logical_formula> | <fof_sequent>
```

Inference

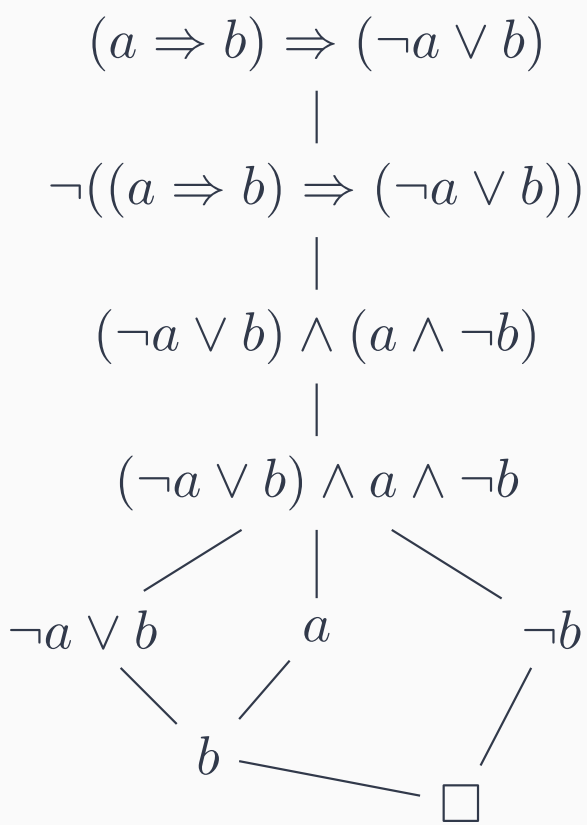
- Annotation of the formula
- Information about the rule applied
- Reference to the parent(s)

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

Example (Resolution)

Prove This!

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



Example (TSTP)

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).
```

Inference Rules for Sequent Calculus

Level-1 Rules

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

Example: Left Or

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

Rule Specifications

- 2 premises
- 2 parameters: `status(thm)` and index of $A \vee B$ 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

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 Rules

- More advanced reasoning steps
- Can be unfolded into level-1 rules
- Better interactions with proof assistants (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

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)`
  - `[i`$_1$`, ..., i`$_n$`:Int]`: Index of $t_j = u_j$ on the left
  - `P(Z`$_1$`, ..., Z`$_n$`):Term`: Shape of the formula on the right
  - `[Z`$_1$`, ..., 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])).
```

Example (Sequent Calculus)

Prove This!

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \Rightarrow b, \neg a \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\,\text{ax} \quad \dfrac{}{a \Rightarrow b, b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\,\text{ax}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b, \neg a, b}\,\Rightarrow_{\text{left}}}{a \Rightarrow b \vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b), \neg a \vee b}\,\vee_{\text{right}}}{\vdash (a \Rightarrow b) \Rightarrow (\neg a \vee b)}\,\Rightarrow_{\text{right}}$$

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))).
```

SC-TPTP Utilitaries

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, ...)

SC-TPTP Utilitaries

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, ...)

More about that tomorrow at 11:30 at PAAR!

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

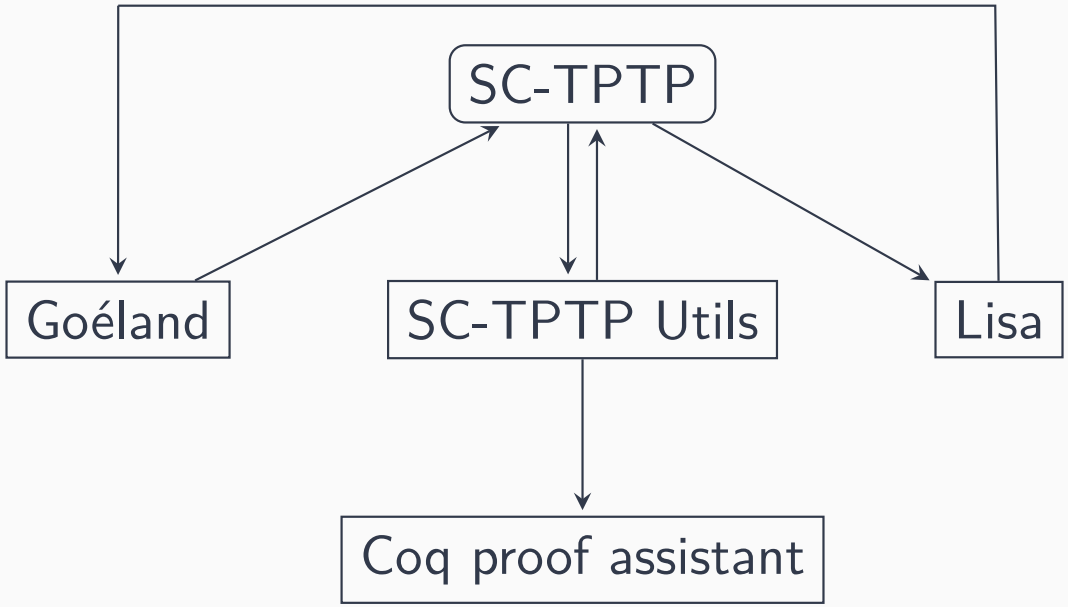
Use Case: Interactions between Goéland and Lisa

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



Conclusion

SC-TPTP

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

Future Work

- Extension to Typed eXtended first-order Form (TXF)
- Add compatible ATP (Princess, Zenon, Connect++)
- Expand proof output formats (LambdaPi, Isabelle)
- Add new tools to the library (desoklemization, ...)
- Connection calculus, theory management (level-3 rules?)

A Standard Output Format!

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

Thank you! ☺

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