Reconstructing Propositional Proofs in Type Theory

Jonathan Prieto-Cubides
Advisor: Andrés Sicard-Ramírez

Master in Applied Mathematics
Universidad EAFIT
Medellín, Colombia

November 17, 2017

UNIVERSIDAD
EAFIT ®

Goal
Formalization in type theory, classical propositional derivations generated
by the Metis theorem prover.

Research

Goal
Formalization in type theory, classical propositional derivations generated by the `Metis` theorem prover.

Topics

- Automatic reasoning using automatic theorem provers (ATPs) (e.g., `Metis`, `EProver`)
- Interactive proving using proof-assistants (e.g., `Agda`, `Coq`)
- Proof-reconstruction for proofs generated by ATPs in proof-assistants

Research Outcomes

Academic result: paper (work in progress)

Software related results:

- `Athena`[1]: a translator tool for `Metis` proofs to `Agda` in `Haskell`
- `Agda` libraries:
  - `Agda-Metis`[2]: `Metis` prover reasoning for propositional logic
  - `Agda-Prop`[3]: intuitionistic propositional logic with PEM
- Bugs found in `Metis`: see issues No. 2, No. 4, and commit 8a3f11e in `Metis` official repository[4]

In parallel, we develop:

- `Online-ATPs`[5]: a client for the TPTP world in `Haskell` This tool allowed us to use `Metis` without installing it
- `Prop-Pack`[6]: compendium of TPTP problems in classical propositional logic used to test `Athena`

---

[1] https://github.com/jonaprieto/athena.

[2] https://github.com/jonaprieto/agda-metis.

[3] https://github.com/jonaprieto/agda-prop.

[4] https://github.com/gilith/metis.

[5] https://github.com/jonaprieto/online-atps.

[6] https://github.com/jonaprieto/prop-pack.

Bug in the Printing of the Proof

$$\varphi := \neg p \wedge (\neg q \Leftrightarrow \neg r) \wedge (\neg p \Leftrightarrow (\neg q \Leftrightarrow \neg r))$$

Bug in the Printing of the Proof

Fixed in `Metis` v2.3 (release 20161108)

$$\varphi := \neg p \wedge (\neg q \Leftrightarrow \neg r) \wedge (\neg p \Leftrightarrow (\neg q \Leftrightarrow \neg r))$$

$$
\cfrac{\cfrac{\vdots}{\varphi} \text{ canonicalize}}{\neg p \Leftrightarrow (\neg q \Leftrightarrow \neg r)} \text{ conjunct} \qquad
\cfrac{\cfrac{\vdots}{\varphi} \text{ canonicalize}}{\neg q \Leftrightarrow \neg r} \text{ conjunct} \qquad
\cfrac{\cfrac{\vdots}{\varphi} \text{ canonicalize}}{\neg p} \text{ conjunct}
$$
$$\bot \quad \text{simplify}$$

The bug was caused by the conversion of `Xor` sets to `Iff` lists. After reporting this, `Metis` developer fixed the printing of `canonicalize` inference rule

$$\varphi := \neg p \wedge (\neg q \Leftrightarrow \neg r) \wedge (\neg p \Leftrightarrow (\neg q \Leftrightarrow r))$$

Soundness Bug in Splitting goals
Fixed in Metis v2.3 (release 20170810)

Consider this TPTP problem

```
$ cat issue.tptp
fof(goal, conjecture, (~ (p <=> q)) <=> ((p => ~ q) & (q => ~p))).
```

Metis found a proof when other ATPs do not. Indeed, the problem is not a tautology.

```
$ metis issue.tptp
SZS status Theorem for issue.tptp
```

Testing with EProver with a client for SystemOnTPTP (Online-ATPs).

```
$ online-atps --atp=e issue.tptp
...
# No proof found!
# SZS status CounterSatisfiable
...
```
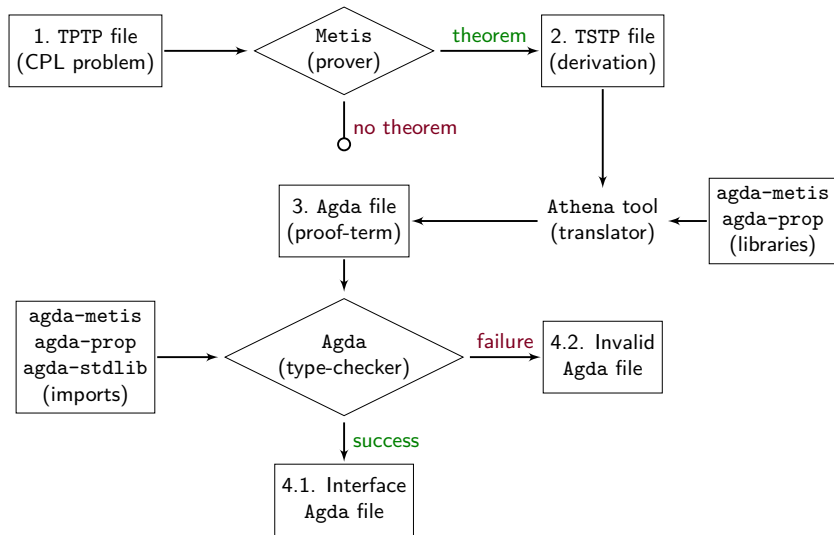
Soundness Bug in Splitting goals

The bug was in the `strip` inference rule:

$$\neg\,(p \Leftrightarrow q) \Leftrightarrow ((p \Rightarrow \neg\,q) \wedge (q \Rightarrow \neg\,p))$$

Solved with:

$$\neg\,(p \Leftrightarrow q) \Leftrightarrow ((p \Rightarrow \neg\,q) \wedge (\neg\,q \Rightarrow p))$$

Proof Reconstruction: Overview

Inference Rules of `Metis`

TSTP derivations by `Metis` exhibit the following inferences:

`Metis` rule	Purpose
`strip`	Strip a goal into subgoals
`conjunct`	Takes a formula from a conjunction
`resolve`	A general form of the resolution theorem
`canonicalize`	Normalization of the formula
`clausify`	Performs clausification
`simplify`	Simplify definitions and theorems

Proposition Type in `Agda`

A data type for formulas

```
data PropFormula : Set where
  Var : Fin n → Prop
  T   : Prop
  ⊥   : Prop
  _∧_ : (φ ψ : Prop) → Prop
  _∨_ : (φ ψ : Prop) → Prop
  _⇒_ : (φ ψ : Prop) → Prop
  _⇔_ : (φ ψ : Prop) → Prop
  ¬_  : (φ : Prop)   → Prop
```

Inference Rules For Propositional Logic I

Intuitionistic Propositional Logic + PEM ($\Gamma \vdash \varphi \vee \neg\varphi$)

$$\frac{}{\Gamma, \varphi \vdash \varphi} \text{ assume} \qquad \frac{}{\Gamma \vdash \top} \top\text{-intro} \qquad \frac{}{\Gamma \vdash \varphi \vee \neg\, \varphi} \text{ PEM}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi} \bot\text{-elim} \qquad \frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\, \varphi} \neg\text{-intro} \qquad \frac{\Gamma \vdash \neg\, \varphi \quad \Gamma \vdash \varphi}{\Gamma \vdash \bot} \neg\text{-elim}$$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} \wedge\text{-intro} \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} \wedge\text{-proj}_1 \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi} \wedge\text{-proj}_2$$

Inference Rules For Propositional Logic II

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} \ \vee\text{-intro}_1 \qquad\qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} \ \vee\text{-intro}_2$$

$$\frac{\Gamma, \varphi \vdash \gamma \qquad \Gamma, \psi \vdash \gamma}{\Gamma, \varphi \vee \psi \vdash \gamma} \ \vee\text{-elim}$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi} \ \Rightarrow\text{-intro} \qquad\qquad \frac{\Gamma \vdash \varphi \Rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi} \ \Rightarrow\text{-elim}$$

Other Rules

▶ Weakening: to extend the hypotheses with additional formulas

$$\frac{\Gamma \vdash \varphi}{\Gamma, \psi \vdash \varphi} \text{ weaken}$$

▶ The RAA rule is the formulation of the principle of proof by contradiction:

$$\frac{\Gamma, \neg \varphi \vdash \bot}{\Gamma \vdash \varphi} \text{ RAA}$$

Syntactical Consequence Relation in `Agda`

▶ Inductive family $\_ \vdash \_$ with two indexes: a set of propositions Γ (the premises) and a proposition φ (the conclusion)

Example

In [8] we define $\_ \vdash \_$ as follows

```
data _⊢_ : (Γ : Ctxt)(φ : Prop) → Set
  ...
  ∧-intro
    : ∀ {Γ} {φ ψ}
    → Γ ⊢ φ → Γ ⊢ ψ
    → Γ ⊢ φ ∧ ψ

  ∧-proj₁
    : ∀ {Γ} {φ ψ}
    → Γ ⊢ φ ∧ ψ
    → Γ ⊢ φ

  ∧-proj₂
    : ∀ {Γ} {φ ψ}
    → Γ ⊢ φ ∧ ψ
    → Γ ⊢ ψ
  ...
```

Reconstructing `Metis` Rules in Type Theory

Let $\mathrm{metisRule}$ be a `Metis` inference rule. We define the function metisRule in type theory which has the following pattern[7]:

$$\mathrm{metisRule} : \textsc{Premise} \to \textsc{Conclusion} \to \textsc{Prop}$$

$$\mathrm{metisRule} \ \varphi \ \psi \quad = \begin{cases} \psi, & \text{if metisRule built } \psi \text{ by applying inference} \\ & \text{rules to } \varphi; \\ \varphi, & \text{otherwise;} \end{cases}$$

To justify all transformations done by the metisRule rule, we prove its soundness with a theorem like the following:

> If $\Gamma \vdash \varphi$ then $\Gamma \vdash \mathrm{metisRule} \ \varphi \ \psi$, where $\psi : \textsc{Conclusion}$.

---

[7] $\textsc{Premise}$ and $\textsc{Conclusion}$ as synonyms of the $\textsc{Prop}$ type to describe in the function types the role of the arguments

Reconstructing a `Metis` **Inference Rule**

The `clausify` rule transforms a formula into its clausal normal form.

Example

In the following TSTP derivation by `Metis`, we see how `clausify` transforms the norm_0 formula to get norm_1 formula.

```
fof(norm_0, ¬ p ∨ (q ∧ r) ...
fof(norm_1, (¬ p ∨ q) ∧ (¬ p ∨ r), inf(clausify, norm_0)).
```

Theorem

Let ψ : CONCLUSION. If $\Gamma \vdash \varphi$ then $\Gamma \vdash$ clausify $\varphi\ \psi$, where

$$\text{clausify} : \text{PREMISE} \rightarrow \text{CONCLUSION} \rightarrow \text{PROP}$$

$$\text{clausify } \varphi\ \psi\ = \begin{cases} \psi, & \text{if } \varphi \equiv \psi; \\ \text{reorder}_{\wedge\vee} \ (\text{cnf } \varphi) \ \psi, & \text{otherwise.} \end{cases}$$

The Intuition behind the `Metis` Algorithm

Algorithm 1 `Metis` refutation strategy

---

    procedure METIS
    input: the goal and a set of *premises* $a_1, \cdots, a_n$
    output: maybe a derivation when $a_1, \cdots, a_n \vdash \text{goal}$, otherwise
    nothing.

        strip the goal into a list of *subgoals* s_i
        for each subgoal s_i **do**
            try to find by a refutation for $\neg s_i$:
                apply clausification for the negated subgoal $\neg s_i$
            if a premise a_j is relevant **then**
                apply clausification to a_j
            end if
                application of `Metis` inference rules
            if a contradiction can be derived from the assumptions **then**
                keep the refutation and continue with the others subgoals
            else
                exit without a proof.
            end if
        end for
        print the conjecture and the premises
        print each refutation for each negated subgoal
    end procedure

Some Challenges

- Formalization
  - Understanding the `Metis` reasoning without a proper documentation or description from the `Metis` author
  - Terminating of functions that reconstruct `Metis` inference rules
  - Intuitionistic logic implementation
- Software related
  - Parsing of `TSTP` derivations
  - Printing valid `Agda` files

Complete Example

The problem[8]:

$$(p \Rightarrow q) \wedge (q \Rightarrow p) \vdash (p \vee q) \Rightarrow (p \wedge q)$$

In TPTP syntax:

```
fof(a₁, axiom, (p ⇒ q) ∧ (q ⇒ p)).
fof(goal, conjecture, (p ∨ q) ⇒ (p ∧ q)).
```

Its TSTP solution using Metis:

```
fof(a₁, axiom, (p ⇒ q) ∧ (q ⇒ p)).
fof(goal, conjecture, (p ∨ q) ⇒ (p ∧ q)).
fof(s₁, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(s₂, ((p ∨ q) ∧ p) ⇒ q, inf(strip, goal)).
...
```

[8]Problem No. 13 in Disjunction Section in [7]

```
fof(s₁, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(s₂, ((p ∨ q) ∧ p) ⇒ q, inf(strip, goal)).
fof(neg₁, ¬ ((p ∨ q) ⇒ p), inf(negate, s₁)).
fof(n00, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a₁)).
fof(n01, ¬ q ∨ p, inf(conjunct, n00)).
fof(n02, ¬ p ∧ (p ∨ q), inf(canonicalize, neg₁)).
fof(n03, p ∨ q, inf(conjunct, n02)).
fof(n04, ¬ p, inf(conjunct, n02)).
fof(n05, q, inf(simplify,[n03, n04])).
cnf(r00, ¬ q ∨ p, inf(canonicalize, n01)).
cnf(r01, q, inf(canonicalize, n05)).
cnf(r02, p, inf(resolve, q, [r01, r00])).
cnf(r03, ¬ p, inf(canonicalize, n04)).
cnf(r04, ⊥, inf(resolve, p, [r02, r03])).
fof(neg₂, ¬ (((p ∨ q) ∧ p) ⇒ q), inf(negate, s2)).
fof(n10, ¬ q ∧ p ∧ (p ∨ q), inf(canonicalize, neg₂)).
fof(n11, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a₁)).
fof(n12, ¬ p ∨ q, inf(conjunct, n11)).
fof(n13, ⊥, inf(simplify,[n10, n12])).
cnf(r10, ⊥, inf(canonicalize, n13)).
```

```
fof(s₁, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(neg₁, ¬ ((p ∨ q) ⇒ p), inf(negate, s₁)).
fof(n00, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a₁)).
fof(n01, ¬ q ∨ p, inf(conjunct, n00)).
fof(n02, ¬ p ∧ (p ∨ q), inf(canonicalize, neg₁)).
fof(n03, p ∨ q, inf(conjunct, n02)).
fof(n04, ¬ p, inf(conjunct, n02)).
fof(n05, q, inf(simplify,[n03, n04])).
cnf(r00, ¬ q ∨ p, inf(canonicalize, n01)).
cnf(r01, q, inf(canonicalize, n05)).
cnf(r02, p, inf(resolve, q, [r01, r00])).
cnf(r03, ¬ p, inf(canonicalize, n04)).
cnf(r04, ⊥, inf(resolve, p, [r02, r03])).
```

Tree for the Subgoal No. 1: $(p \vee q) \Rightarrow p$

```
fof(a₁, axiom, (p ⇒ q) ∧ (q ⇒ p)).
...
fof(n00, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a₁)).
fof(n01, ¬ q ∨ p, inf(conjunct, n00)).
...
```

$$(\mathcal{D}_1) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash (p \Rightarrow q) \wedge (q \Rightarrow p)} \text{ axiom } a_1}{\Gamma, \neg s_1 \vdash (p \Rightarrow q) \wedge (q \Rightarrow p)} \text{ weaken}}{\Gamma, \neg s_1 \vdash (\neg p \vee q) \wedge (\neg q \vee p)} \text{ canonicalize}}{\Gamma, \neg s_1 \vdash \neg q \vee p} \text{ conjunct}$$

```
...
fof(s_1, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(neg_1, ¬ ((p ∨ q) ⇒ p), inf(negate, s_1)).
...
fof(n02, ¬ p ∧ (p ∨ q), inf(canonicalize, neg_1)).
fof(n03, p ∨ q, inf(conjunct, n02)).
fof(n04, ¬ p, inf(conjunct, n02)).
...
```

$$(\mathcal{D}_2) \qquad \cfrac{\cfrac{\cfrac{}{\Gamma, \neg s_1 \vdash \neg s_1}\ \text{assume}}{\Gamma, \neg s_1 \vdash \neg p \wedge (p \vee q)}\ \text{canonicalize}}{\Gamma, \neg s_1 \vdash p \vee q}\ \text{conjunct}$$

$$(\mathcal{D}_3) \qquad \cfrac{\cfrac{\cfrac{}{\Gamma, \neg s_1 \vdash \neg s_1}\ \text{assume } \neg s_1}{\Gamma, \neg s_1 \vdash \neg p \wedge (p \vee q)}\ \text{canonicalize}}{\Gamma, \neg s_1 \vdash \neg p}\ \text{conjunct}$$

$$(\mathcal{D}_4) \qquad \dfrac{\dfrac{\mathcal{D}_2}{\Gamma, \neg s_1 \vdash p \vee q} \qquad \dfrac{\mathcal{D}_3}{\Gamma, \neg s_1 \vdash \neg p}}{\Gamma, \neg s_1 \vdash q} \text{ simplify}$$

$$(\mathcal{R}_1) \qquad \dfrac{\dfrac{\dfrac{\mathcal{D}_1}{\Gamma, \neg s_1 \vdash \neg q \vee p} \quad \dfrac{\mathcal{D}_4}{\Gamma, \neg s_1 \vdash q}}{\Gamma, \neg s_1 \vdash p} \text{ resolve } q \qquad \dfrac{\mathcal{D}_3}{\Gamma, \neg s_1 \vdash \neg p}}{\dfrac{\Gamma, \neg s_1 \vdash \bot}{\Gamma \vdash s_1} \text{ RAA}} \text{ resolve } p$$

Tree for the Subgoal No. 2: $((p \vee q) \wedge p) \Rightarrow q$

```
fof(s₂, ((p ∨ q) ∧ p) ⇒ q, inf(strip, goal)).
fof(neg₂, ¬ (((p ∨ q) ∧ p) ⇒ q), inf(negate, s2)).
fof(n10, ¬ q ∧ p ∧ (p ∨ q), inf(canonicalize, neg₂)).
fof(n11, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a₁)).
fof(n12, ¬ p ∨ q, inf(conjunct, n11)).
fof(n13, ⊥, inf(simplify,[n10, n12])).
cnf(r10, ⊥, inf(canonicalize, n13)).
```

$$(\mathcal{R}_2) \quad \cfrac{\cfrac{\cfrac{\cfrac{}{\Gamma, \neg s_2 \vdash \neg s_2} \text{ assume } (\neg s_2)}{\Gamma, \neg s_2 \vdash \neg q \wedge p \wedge (p \vee q)} \text{ canonicalize} \quad \cfrac{\cfrac{\cfrac{\cfrac{}{\Gamma \vdash (p \Rightarrow q) \wedge (q \Rightarrow p)} \text{ axiom } a_1}{\Gamma, \neg s_2 \vdash (p \Rightarrow q) \wedge (q \Rightarrow p)} \text{ weaken}}{\Gamma, \neg s_2 \vdash (\neg p \vee q) \wedge (\neg q \vee p)} \text{ canonicalize}}{\Gamma, \neg s_2 \vdash \neg p \vee q} \text{ conjunct}}{\Gamma, \neg s_2 \vdash \bot} \text{ simplify}}{\Gamma \vdash s_2} \text{ RAA}$$

Summarizing the Example

The problem was:

$$(p \Rightarrow q) \wedge (q \Rightarrow p) \vdash (p \vee q) \Rightarrow (p \wedge q)$$

Its TSTP solution using `Metis` was:

```
fof(a₁, axiom, (p ⇒ q) ∧ (q ⇒ p)).
fof(goal, conjecture, (p ∨ q) ⇒ (p ∧ q))).
fof(s₁, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(s₂, ((p ∨ q) ∧ p) ⇒ q, inf(strip, goal)).
...
```

The proof is:

$$
\cfrac{
  \cfrac{}{\Gamma \vdash (s_1 \wedge s_2) \Rightarrow \mathsf{goal}} \text{ strip}
  \qquad
  \cfrac{
    \cfrac{\mathcal{R}_1}{\Gamma \vdash s_1} \qquad \cfrac{\mathcal{R}_2}{\Gamma \vdash s_2}
  }{\Gamma \vdash s_1 \wedge s_2} \text{ } \wedge\text{-intro}
}{\Gamma \vdash \mathsf{goal}} \Rightarrow\text{-elim}
$$

(Live example using Agda and Athena)

Future Work

Further research directions include, but are not limited to:

- ▶ improve the performance of the `canonicalize` rule
- ▶ extend the proof-reconstruction presented in this paper to
  - ▶ support the proposition logic with equality of `Metis`
  - ▶ support other ATPS for propositional logic like `EProver` or `Z3`.
    See Kanso's Ph.D. thesis [5]
  - ▶ support `Metis` first-order proofs

Related Work

In type theory:

- ▶ Kanso in [5] reconstructs in `Agda` propositional proofs generated by `EProver` and `Z3`
- ▶ Foster and Struth in [2] describe proof-reconstruction in `Agda` for equational logic of `Waldmeister` prover
- ▶ Bezem, Hendriks, and Nivelle in [1] transform a proof produced by the first-order prover `Bliksem` in a `Coq` proof-term

In classical logic:

- ▶ Paulson and Susanto in [6] introduce `SledgeHammer`, a tool ables to reconstructs proofs of well-known ATPS: `EProver`, `Vampire`, among others using `SystemOnTPTP` server
- ▶ Hurd in [3] integrates the first-order resolution prover `Gandalf` prover for `HOL` proof-assistant
- ▶ Kaliszyk and Urban in [4] reconstruct proofs of different ATPS for `HOL Light`

References I

📄 Marc Bezem, Dimitri Hendriks, and Hans de Nivelle. Automated Proof Construction in Type Theory Using Resolution. Journal of Automated Reasoning 29.3-4 (2002), pp. 253–275. DOI: 10.1023/A:1021939521172 (cit. on p. 29).

📄 Simon Foster and Georg Struth. Integrating an Automated Theorem Prover in Agda. In: NASA Formal Methods (NFM 2011). Ed. by Mihael Bobaru et al. Vol. 6617. Lecture Notes in Computer Science. Springer, 2011, pp. 116–130. DOI: 10.1007/978-3-642-20398-5_10 (cit. on p. 29).

📄 Joe Hurd. Integrating Gandalf and HOL. In: Theorem Proving in Higher Order Logics (TPHOLs 2001). Ed. by Yves Bertot, Gilles Dowek, Laurent Théry, and Christine Paulin. Vol. 1690. Lecture Notes in Computer Science. Springer, 2001, pp. 311–321. DOI: 10.1007/3-540-48256-3_21 (cit. on p. 29).

References II

📄 Cezary Kaliszyk and Josef Urban. PRocH: Proof Reconstruction for HOL Light. In: Automated Deduction (CADE-24). Ed. by Maria Paola Bonacina. Vol. 7898. Lecture Notes in Artifical Intellingence. Springer, 2013, pp. 267–274. DOI: 10.1007/978-3-642-38574-2_18 (cit. on p. 29).

📄 Karim Kanso. Agda as a Platform for the Development of Verified Railway Interlocking Systems. PhD thesis. Department of Computer Science. Swansea University, 2012. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.310.1502 (cit. on pp. 28, 29).

📄 Lawrence C. Paulson and Kong Woei Susanto. Source-level Proof Reconstruction For Interactive Theorem Proving. In: TPHOLs. Vol. 4732. Springer. 2007, pp. 232–245 (cit. on p. 29).

📄 Jonathan Prieto-Cubides. A Collection of Propositional Problems in TPTP Format. June 2017. DOI: 10.5281/ZENODO.817997 (cit. on p. 20).

Jonathan Prieto-Cubides. A Library for Classical Propositional Logic in Agda. 2017. DOI: 10.5281/zenodo.398852 (cit. on p. 15).

- ▶ Is a language[9] to encode problems
- ▶ Is the input of the ATPS
- ▶ Annotated formulas with the form

  language(name, role, formula).

language FOF or CNF
   name to identify the formula within the problem
   role axiom, definition, hypothesis, conjecture
 formula formula in TPTP format

---

[9]http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html.

`Metis` **Theorem Prover**

`Metis` is an automatic theorem prover for first-order logic with equality.

- ▶ Open source implemented
- ▶ Reads problems in TPTP format
- ▶ Outputs *detailed* proofs in TSTP format
- ▶ For the propositional logic, `Metis` has only three inference rules:

$$\frac{}{\Gamma \vdash \varphi_1 \vee \cdots \vee \varphi_n} \text{ axiom } \varphi_1, \cdots, \varphi_n$$

$$\frac{}{\Gamma \vdash \varphi \vee \neg\varphi} \text{ assume } \varphi$$

$$\frac{\Gamma \vdash \varphi_1 \vee \cdots \vee l \vee \cdots \vee \varphi_n \qquad \Gamma \vdash \psi_1 \vee \cdots \vee \neg l \vee \cdots \vee \psi_m}{\Gamma \vdash \varphi_1 \vee \cdots \vee \varphi_n \vee \psi_1 \vee \cdots \vee \psi_m} \text{ resolve } l$$

TSTP **Syntax**

A TSTP derivation[10]

- ▶ Is a **D**irected **A**cyclic **G**raph where

  leaf is a formula from the TPTP input
  node is a formula inferred from parent formula
  root the final derived formula

- ▶ Is a list of annotated formulas with the form

  language(name, role, formula, source [,useful info]).

where source typically is an inference record

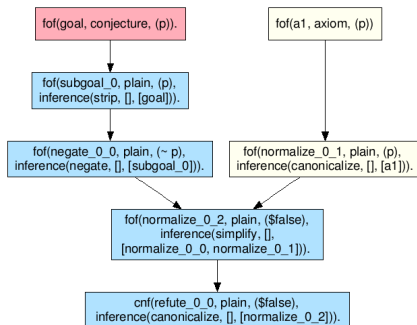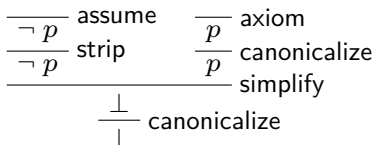 inference(rule, useful info, parents).

Another TSTP Example

▶ Proof found by `Metis` for the problem $p \vdash p$

```
$ metis --show proof problem.tptp
fof(a, axiom, p).
fof(goal, conjecture, p).
fof(subgoal_0, plain, p),
  inference(strip, [], [goal])).
fof(negate_0_0, plain, ~ p,
  inference(negate, [], [subgoal_0])).
fof(normalize_0_0, plain, ~ p,
  inference(canonicalize, [], [negate_0_0])).
fof(normalize_0_1, plain, p,
  inference(canonicalize, [], [a])).
fof(normalize_0_2, plain, $false,
  inference(simplify, [],
    [normalize_0_0, normalize_0_1])).
cnf(refute_0_0, plain, $false,
    inference(canonicalize, [], [normalize_0_2])).
```

DAG Example

By refutation, we proved $p \vdash p$:

$$\frac{}{\neg p} \text{ assume}$$
$$\frac{\neg p}{\neg p} \text{ strip}$$

$$\frac{}{p} \text{ axiom}$$
$$\frac{p}{p} \text{ canonicalize}$$

$$\frac{}{\bot} \text{ simplify}$$
$$\frac{\bot}{\bot} \text{ canonicalize}$$

fof(goal, conjecture, (p)).

fof(a1, axiom, (p))

fof(subgoal_0, plain, (p), inference(strip, [], [goal])).

fof(negate_0_0, plain, (~ p), inference(negate, [], [subgoal_0])).

fof(normalize_0_1, plain, (p), inference(canonicalize, [], [a1])).

fof(normalize_0_2, plain, ($false), inference(simplify, [], [normalize_0_0, normalize_0_1])).

cnf(refute_0_0, plain, ($false), inference(canonicalize, [], [normalize_0_2])).

Athena **tool**

Is a `Haskell` program that translates proofs given by `Metis` in TSTP format to `Agda` code

- ▶ Parsing of TSTP language
- ▶ Creation and analysis of **DAG** derivations
- ▶ Analysis of inference rules used in the TSTP derivation
- ▶ `Agda` code generation

Library	Purpose
`Agda-Prop`	axioms and theorems of classical propositional logic
`Agda-Metis`	versions of the inference rules used by `Metis`

▶ Definition

$$\mathsf{conjunct}(\overbrace{\varphi_1 \wedge \cdots \wedge \varphi_n}^{\varphi}, \psi) = \begin{cases} \varphi_i & \text{if } \psi \text{ is equal to some } \varphi_i \\ \varphi & \text{otherwise} \end{cases}$$

Agda-Metis: **Conjunct Inference** [11]

▶ Definition

$$\text{conjunct}(\overbrace{\varphi_1 \wedge \cdots \wedge \varphi_n}^{\varphi}, \psi) = \begin{cases} \varphi_i & \text{if } \psi \text{ is equal to some } \varphi_i \\ \varphi & \text{otherwise} \end{cases}$$

▶ Inference rules involved

$$\frac{\varphi_1 \wedge \varphi_2}{\varphi_1} \text{ } \wedge\text{-proj}_1 \qquad\qquad \frac{\varphi_1 \wedge \varphi_2}{\varphi_2} \text{ } \wedge\text{-proj}_2$$
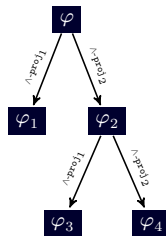
▶ Example

$$\varphi := \varphi_1 \wedge \overbrace{(\varphi_3 \wedge \varphi_4)}^{\varphi_2}$$

  ▶ conjunct $(\varphi, \varphi_3 \wedge \varphi_1) \equiv \varphi$

  ▶ conjunct $(\varphi, \varphi_3) \equiv \varphi_3$

  ▶ conjunct $(\varphi, \varphi_2) \equiv \varphi_2$