Reconstructing Propositional Proofs in Type Theory

Jonathan Prieto-Cubides
Advisor: Andrés Sicard-Ramírez

Master in Applied Mathematics
Universidad EAFIT
Medellín, Colombia

Novembe

Research

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`)
- Formal methods to verify outputs of ATPs in proof-assistants

Outcomes of the Research

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.

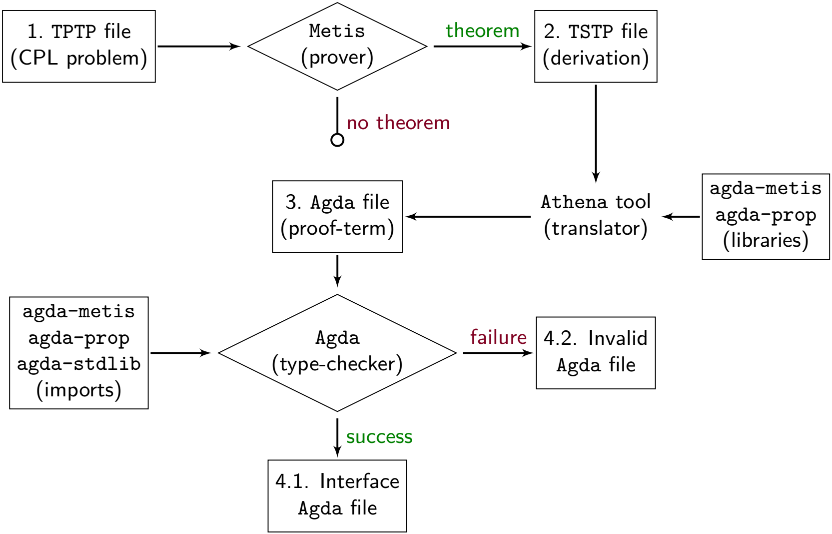
[3] https://github.com/jonaprieto/agda-prop.

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

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

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

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

A data type for formulas

```
data Prop : Set where
  Var : Fin n → Prop
  ⊤   : 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 ($\Gamma \vdash \varphi \vee \neg\varphi$)

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

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

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

Inference Rules For Propositional Logic II

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

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

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

Useful Rules

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

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

Syntactical Consequence Relation in Agda

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

```
data _⊢_ : (Γ : Ctxt)(φ : Prop) → Set
```

- Constructors (inference rules)

```
assume, ⊤-intro, ⊥-elim, ¬-intro,
¬-elim, ∧-intro, ∧-proj₁, ∧-proj₂, ∨-intro₁,
∨-intro₂, ∨-elim, ⇒-intro, ⇒-elim, ⇔-intro,
⇔-elim₁, ⇔-elim₂.
```

For example, the introduction rule for conjunction ∧-intro is the constructor:

```
  ∧-intro
    :   {Γ} {φ ψ}
    → Γ ⊢ φ → Γ ⊢ ψ
    → Γ ⊢ φ ∧ ψ
```

- In [**AgdaProp**] we can find more than 99 theorems to reasoning in classical propositional logic

```
∧-assoc, ∧-comm, ∧-dist, ¬∧-to-¬∨¬, ∨-assoc, ∨-comm, ∨-dist, ∨-equiv
¬¬∨¬¬-to-∨, cnf, nnf, dnf, ...
```

Reconstructing Metis Rules in Type Theory

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

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

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

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

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

[7] Premise and Conclusion as synonyms of the Prop type to describe in the function types the role of the arguments

Reconstructing Example

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 $\texttt{norm}_0$ formula to get $\texttt{norm}_1$ formula.

```
fof(norm₀, ¬ p ∨ (q ∧ r) ...
fof(norm₁, (¬ p ∨ q) ∧ (¬ p ∨ r), inf(clausify, norm₀)).
```

Theorem

Let $\psi : \textsc{Conclusion}$*. If* $\Gamma \vdash \varphi$ *then* $\Gamma \vdash \mathsf{clausify}\ \varphi\ \psi$*, where*

$$\mathsf{clausify} : \textsc{Premise} \to \textsc{Conclusion} \to \textsc{Prop}$$

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

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 the assumptions **then**
keep the refutation and continue with the others subgoals
else
exit without a proof. The conjecture can not be derived from the premises
end if
end for
print the conjecture and the premises
print each refutation for each negated subgoal
end procedure

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
  - Testing

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 [**Prieto-Cubides2017**]

```
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)).
```

TSTP Refutation of Subgoal No. 1

```
fof(s_1, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(neg_1, ¬ ((p ∨ q) ⇒ p), inf(negate, s_1)).
fof(n00, (¬ p ∨ q) ∧ (¬ q ∨ p), inf(canonicalize, a_1)).
fof(n01, ¬ q ∨ p, inf(conjunct, n00)).
fof(n02, ¬ p ∧ (p ∨ q), inf(canonicalize, neg_1)).
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(s₁, (p ∨ q) ⇒ p, inf(strip, goal)).
fof(neg₁, ¬ ((p ∨ q) ⇒ p), inf(negate, s₁)).
...
fof(n02, ¬ p ∧ (p ∨ q), inf(canonicalize, neg₁)).
fof(n03, p ∨ q, inf(conjunct, n02)).
fof(n04, ¬ p, inf(conjunct, n02)).
...
```

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

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

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

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

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)).
```

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

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:

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

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 [**Kanso2012**]
  - support `Metis` first-order proofs

Related Work

In type theory:

- **Kanso2012** in [**Kanso2012**] reconstructs in `Agda` propositional proofs generated by `EProver` and `Z3`
- **foster2011integrating** in [**foster2011integrating**] describe proof-reconstruction in `Agda` for equational logic of `Waldmeister` prover
- **Bezem2002** in [**Bezem2002**] transform a proof produced by the first-order prover `Bliksem` in a `Coq` proof-term

In classical logic:

- **paulson2007source** in [**paulson2007source**] introduce `SledgeHammer`, a tool ables to reconstructs proofs of well-known ATPs: `EProver`, `Vampire`, among others using `SystemOnTPTP` server
- **Hurd1999** in [**Hurd1999**] integrates the first-order resolution prover `Gandalf` prover for `HOL` proof-assistant
- **kaliszyk2013** in [**kaliszyk2013**] reconstruct proofs of different ATPs for `HOL Light`

References I

BONUS SLIDES

TPTP Syntax

Thousands of Problems for Theorem Provers

- 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:

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

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

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

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).
```

[10] http://www.cs.miami.edu/~tptp/TPTP/QuickGuide/Derivations.html.

Another TSTP Example

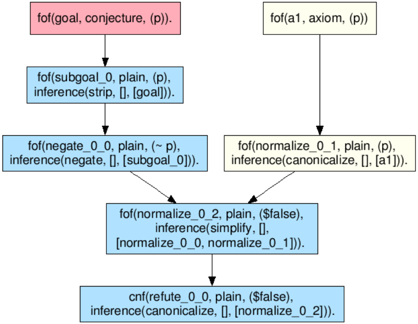
- Proof found by Metis for the problem $p \vdash p$

```
$ 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 $p \vdash p$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg p}\,\text{assume}}{\neg p}\,\text{strip} \qquad \dfrac{\dfrac{}{p}\,\text{axiom}}{p}\,\text{canonicalize}}{\bot}\,\text{simplify}}{\bot}\,\text{canonicalize}$$



Athena tool

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

- Parsing of TSTP language[11]
- 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

[11] https://github.com/agomezl/tstp2agda.