

# Proof Reconstruction in Classical Propositional Logic

(Work in Progress)

Jonathan Prieto-Cubides  
Joint work with Andrés Sicard-Ramírez

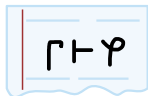
Universidad EAFIT  
Medellín, Colombia

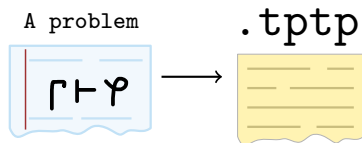
Agda Implementors' Meeting XXV  
May 9-15th



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

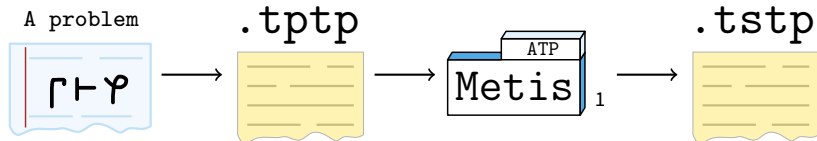
A problem



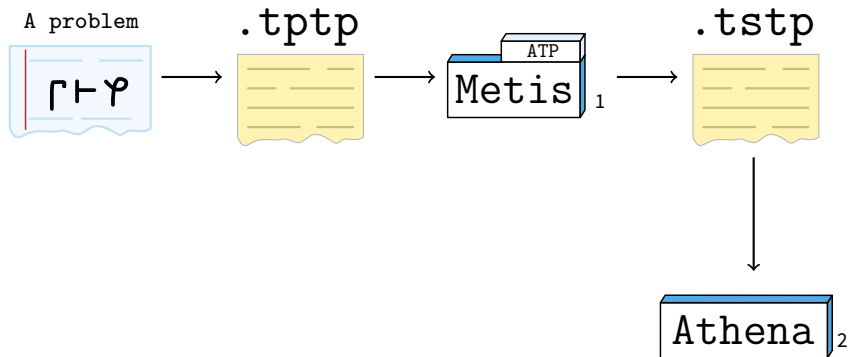




<sup>1</sup>It is available at <http://www.gilith.com/software/metis>

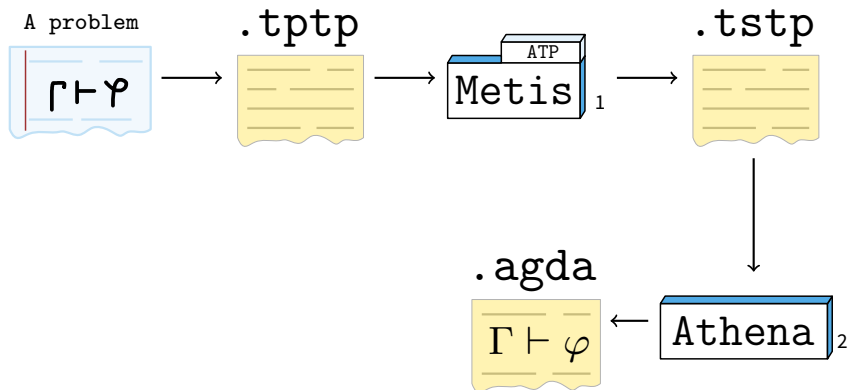


<sup>1</sup>It is available at <http://www.gilith.com/software/metis>



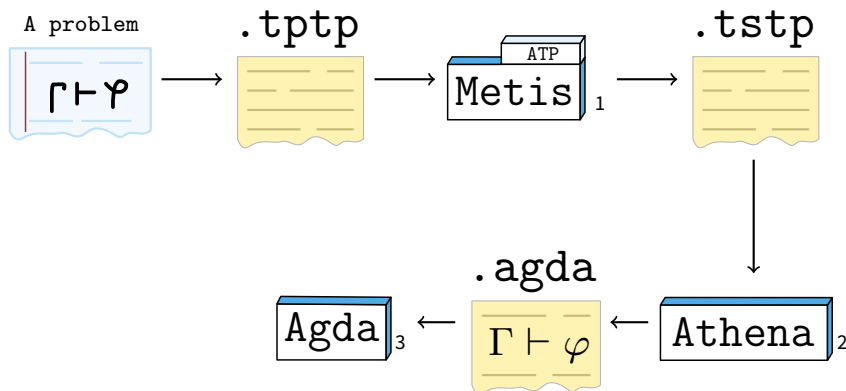
<sup>1</sup> It is available at <http://www.gilith.com/software/metis>

<sup>2</sup> It is available at <http://github.com/jonaprieto/athena>



<sup>1</sup> It is available at <http://www.gilith.com/software/metis>

<sup>2</sup> It is available at <http://github.com/jonaprieto/athena>

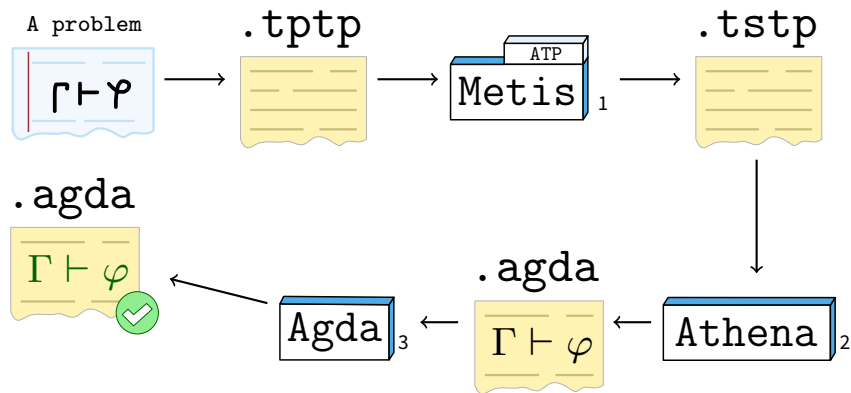


<sup>1</sup> It is available at <http://www.gilith.com/software/metis>

<sup>2</sup> It is available at <http://github.com/jonaprieto/athena>

<sup>3</sup> It is available at <http://github.com/agda/agda>



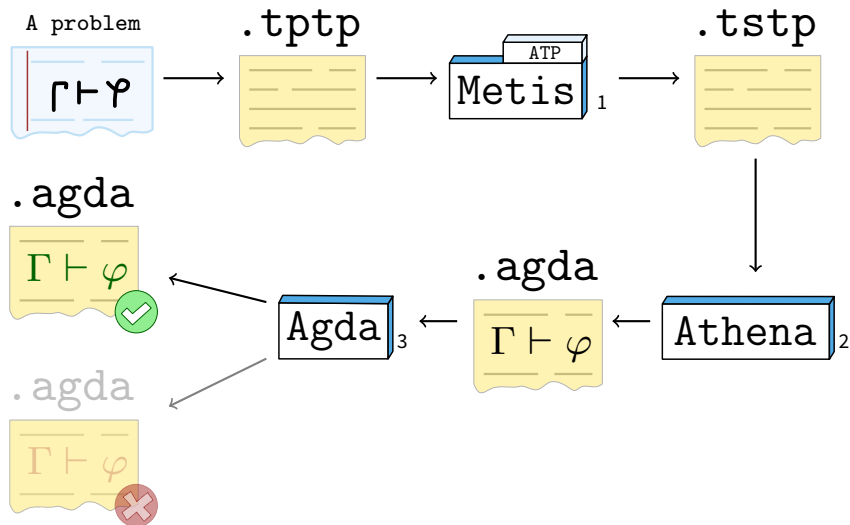


<sup>1</sup> It is available at <http://www.gilith.com/software/metis>

<sup>2</sup> It is available at <http://github.com/jonaprieto/athena>

<sup>3</sup> It is available at <http://github.com/agda/agda>

# Proof Reconstruction: Overview

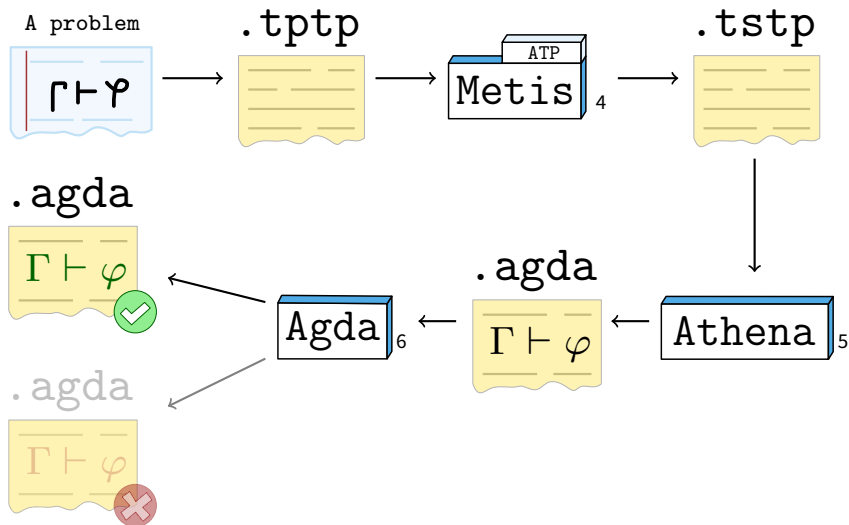


<sup>1</sup> It is available at <http://www.gilith.com/software/metis>

<sup>2</sup> It is available at <http://github.com/jonaprieto/athena>

<sup>3</sup> It is available at <http://github.com/agda/agda>

## Proof Reconstruction: Overview



<sup>4</sup> It is available at <http://www.gilith.com/software/metis>

<sup>5</sup> It is available at <http://github.com/jonaprieto/athena>

<sup>6</sup> It is available at <http://github.com/agda/agda>



.tptp



- ▶ Is a language<sup>7</sup> to encode problems (Sutcliffe, 2009)
- ▶ 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, among others

**formula** version in TPTTP format

<sup>7</sup>Is available at <http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html>

►  $p \vdash p$

```
fof(a, axiom, p).  
fof(goal, conjecture, p).
```

►  $\vdash \neg(p \wedge \neg p) \vee (q \wedge \neg q)$

```
fof(goal, conjecture, ~ ((p & ~ p) | (q & ~ q))).
```

---

<sup>8</sup>Is available at <http://github.com/jonaprieto/prop-pack>

.tstp



A TSTP derivation<sup>9</sup>

- ▶ Is a **Directed Acyclic Graph** 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:

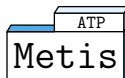
```
language(name, role, formula, source [,useful info]).
```

where **source** typically is an inference record:

```
inference(rule, useful info, parents)
```

<sup>9</sup><http://www.cs.miami.edu/~tptp/TPTP/QuickGuide/Derivations.html>

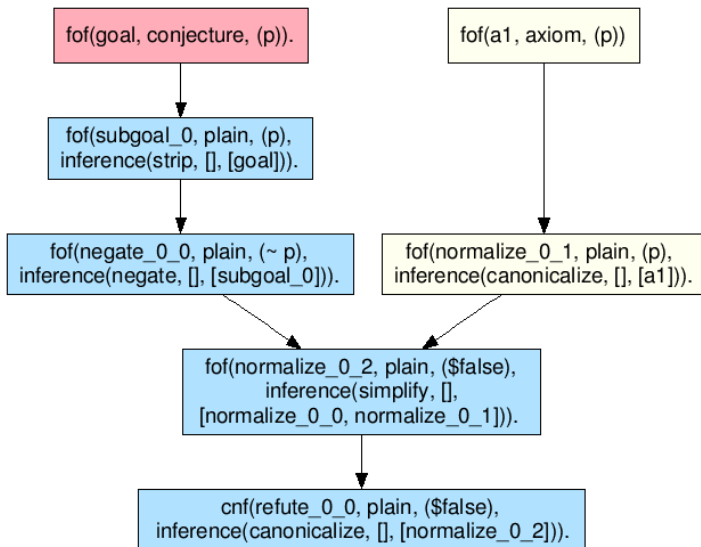
- Proof found by **Metis** ATP for the problem  $p \vdash p$



```
$ metis --show proof basic-4.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 for the previous TSTP derivation found by Meti's ATP

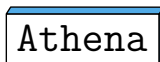


## Athena

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

It depends on:

- ▶ **agda-prop** Classical Logic within Agda: Axioms + Theorems
- ▶ **agda-metis** Theorems of the inference rules of Metis Prover



## Haskell

- ▶ Parsing
- ▶ AST construction
- ▶ Creation and analysis of DAG derivations
- ▶ Analysis of inference rules used
- ▶ Generation of Agda code of the proof



## Agda

Is a dependently typed functional programming language and it also a proof assistant.

We used it to type-check the proofs found by Metis Prover

- ▶ Agda-Prop Library: Logic framework for Classical Propositional Logic
- ▶ Agda-Metis Library: theorems based on the inference rules of Metis Prover



Metis is an automatic theorem prover for First-Order Logic with equality

## Why Metis?

- ▶ Open source implemented in Standard ML
- ▶ Reads problem in **TPTP** format
- ▶ Outputs detailed proofs in **TSTP** format
- ▶ Each refutation step is one of **6 simple rules**

TSTP derivations exhibit these inferences:

Rule	Task
<b>canonicalize</b>	transforms formulas to CNF, DNF or NNF
<b>clausify</b>	performs clausification
<b>conjoinct</b>	extracts a formula from a conjunction
<b>negate</b>	applies negation to the formula
<b>resolve</b>	applies theorems of resolution
<b>simplify</b>	applies over a list of formula to simplify them
<b>strip</b>	splits a formula into subgoals

## ► Definition

$$\text{conjunct}(\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_i \wedge \cdots \wedge \phi, \phi_i) \longrightarrow \phi_i$$

## ► Function<sup>10</sup>:

```
conjunct : Prop → Prop → Prop
conjunct (φ ∧ ψ) ω with [ eq φ ω ] | [ eq ψ ω ]
... | true   | _      = φ
... | false  | true   = ψ
... | false  | false  = conjunct φ ω
conjunct φ ω          = φ
```

## ► Theorem<sup>10</sup>

```
atp-conjunct
  : ∀ {Γ} {φ}
  → (ω : Prop)
  → Γ ⊢ φ
  → Γ ⊢ conjunct φ ω
```

<sup>10</sup>It is available in `agda-metis` library in `ATP.Metis.Rules.Conjunct` module.

- ▶ The problem is  $p \wedge q \vdash q \wedge p$
- ▶ TPTP derivation

```
fof(a, axiom, p & q).  
fof(goal, conjecture, q & p).
```

- ▶ A natural deduction proof

$$\frac{\frac{\phi \wedge \psi}{\phi} \wedge\text{-proj}_1 \quad \frac{\phi \wedge \psi}{\psi} \wedge\text{-proj}_2}{\psi \wedge \phi} \wedge\text{-intro}$$





```
proof0 :  $\Gamma \vdash \text{subgoal}_0$   
proof0 =  
  (RAA  
    (atp-canonicalize  
      (atp-simplify  
        (atp-canonicalize  
          (atp-strip  
            (assume { $\Gamma = \Gamma$ }  
              (atp-negate subgoal0))))))  
    (atp-conjunct (q)  
      (atp-canonicalize  
        (weaken (atp-negate subgoal0)  
          (assume { $\Gamma = \emptyset$ } a)))))))
```

```
proof1 :  $\Gamma \vdash$  subgoal1
proof1 =
  (RAA
    (atp-canonicalize
      (atp-simplify
        (atp-conjunct (q)
          (atp-canonicalize
            (weaken (atp-negate subgoal1)
              (assume { $\Gamma = \emptyset$ } a))))))
      (atp-simplify
        (atp-canonicalize
          (atp-strip
            (assume { $\Gamma = \Gamma$ }
              (atp-negate subgoal1))))))
        (atp-conjunct (p)
          (atp-canonicalize
            (weaken (atp-negate subgoal1)
              (assume { $\Gamma = \emptyset$ } a))))))))))
```

Failure Example goes here

### SledgeHammer

- ▶ Isabelle/HOL tool
- ▶ Metis ported within Isabelle
- ▶ Reconstruct proofs of well-known ATPs: EProver, Vampire, among others

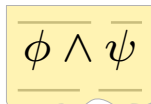
### Integrating Waldmeister and Agda

- ▶ Source code not available
- ▶ Equational Logic
- ▶ Reflection Layers



At the moment, the communication between Agda and the ATPs is unidirectional because the ATPs are being used as oracles (Sicard-Ramírez, 2015).

. agda



+ ATP-Pragma

```
module Or where

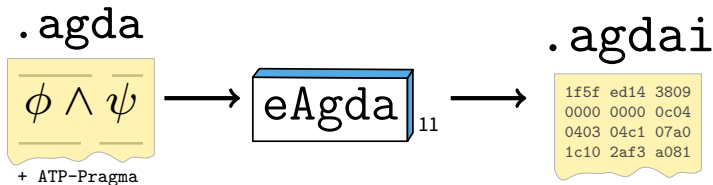
data _∨_ (A B : Set) : Set where
  inj1 : A → A ∨ B
  inj2 : B → A ∨ B

postulate
  A B      : Set
  ∨-comm   : A ∨ B → B ∨ A
  {-# ATP prove ∨-comm #-}
```

## Related Work: Apia

*Proving first-order theorems written in Agda using automatic theorem provers for first-order logic*

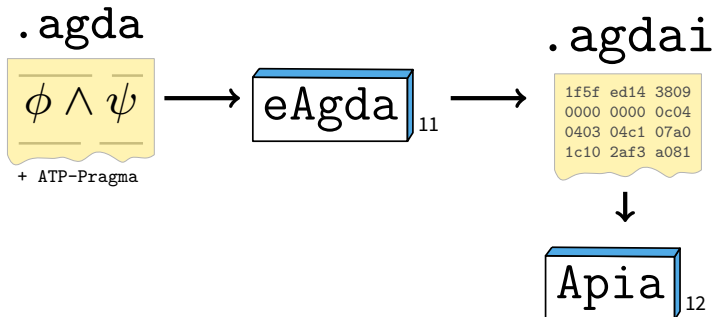
At the moment, the communication between Agda and the ATPs is unidirectional because the ATPs are being used as oracles (ibid.).



<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

## Related Work: Apia

*Proving first-order theorems written in Agda using automatic theorem provers for first-order logic*



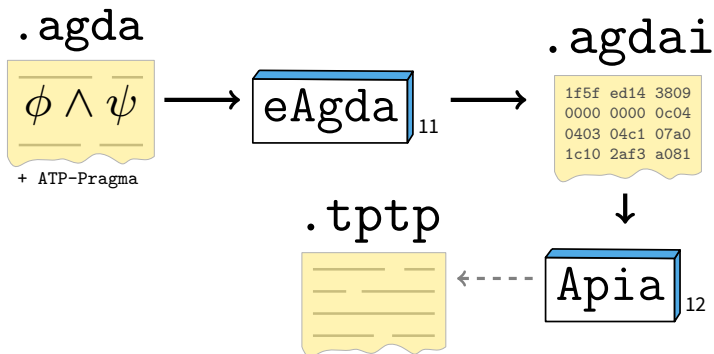
<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

<sup>12</sup>Haskell program for proving first-order theorems written in Agda using ATPs. <https://github.com/asr/apia>



## Related Work: Apia

*Proving first-order theorems written in Agda using automatic theorem provers for first-order logic*

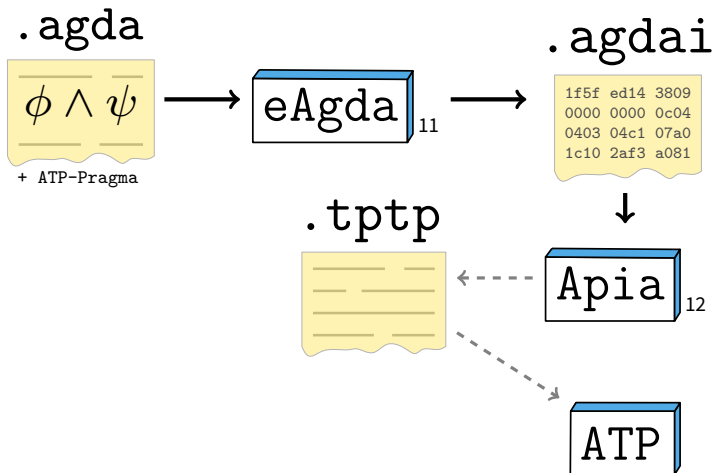


<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

<sup>12</sup>Haskell program for proving first-order theorems written in Agda using ATPs. <https://github.com/asr/apia>

## Related Work: Apia

*Proving first-order theorems written in Agda using automatic theorem provers for first-order logic*

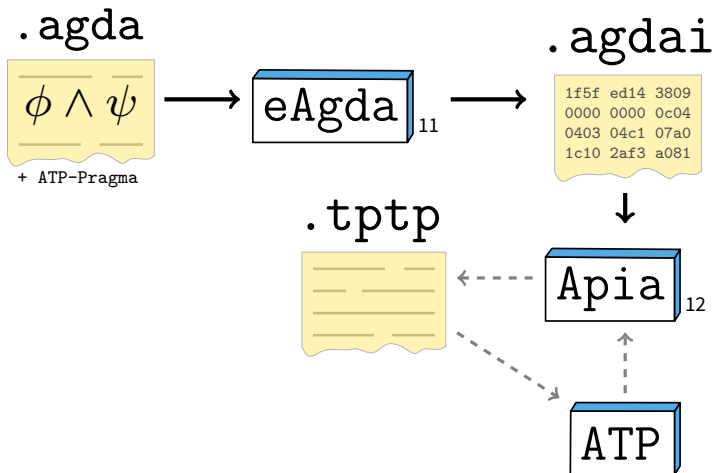


<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

<sup>12</sup>Haskell program for proving first-order theorems written in Agda using ATPs. <https://github.com/asr/apia>

## Related Work: Apia

*Proving first-order theorems written in Agda using automatic theorem provers for first-order logic*

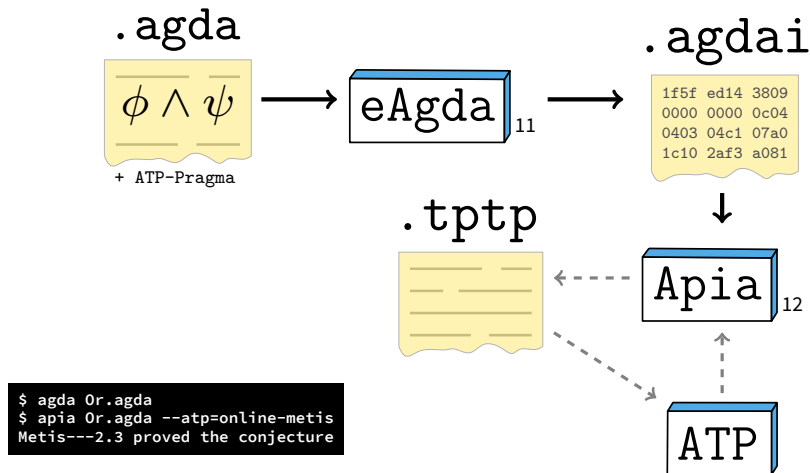


<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

<sup>12</sup>Haskell program for proving first-order theorems written in Agda using ATPs. <https://github.com/asr/apia>

## Related Work: Apia

Proving first-order theorems written in Agda using automatic theorem provers for first-order logic



<sup>11</sup>Development version of Agda in order to handle a new built-in ATP-pragma. <https://github.com/asr/eagda>

<sup>12</sup>Haskell program for proving first-order theorems written in Agda using ATPs. <https://github.com/asr/apia>

- ▶ Add shallow embedding in order to work with **Apia**
- ▶ Support First-Order Logic with Equality
- ▶ Support another prover like **EProver**



Sicard-Ramírez, Andrés (2015). *Reasoning about functional programs by combining interactive and automatic proofs*. PEDECIBA Informática, Universidad de la República.



Sutcliffe, G. (2009). “The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0”. In: *Journal of Automated Reasoning* 43.4, pp. 337–362.