Proof Reconstruction in Classical Propositional Logic

(Work in Progress)

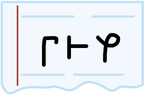
Jonathan Prieto-Cubides
Joint work with Andrés Sicard-Ramírez

Universidad EAFIT
Medellín, Colombia

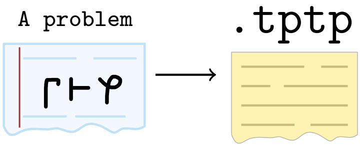
Agda Implementors' Meeting XXV
May 9-15th

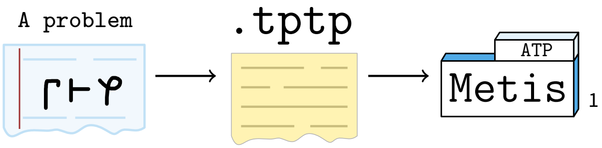
Proof Reconstruction: Overview

A problem

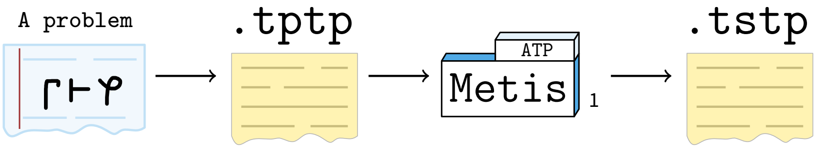


Proof Reconstruction: Overview

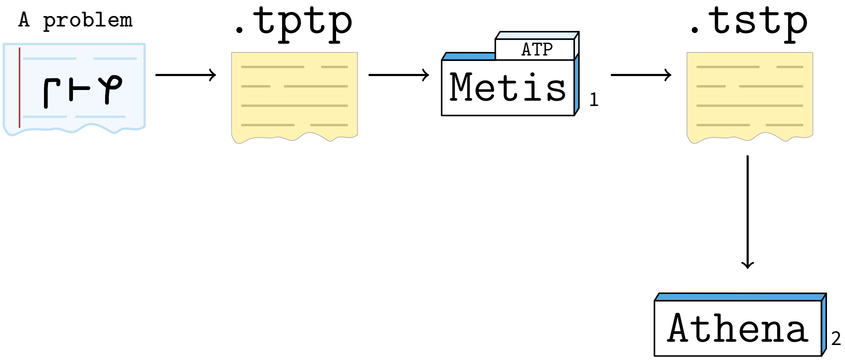




[1] It is available at `http://www.gilith.com/software/metis`



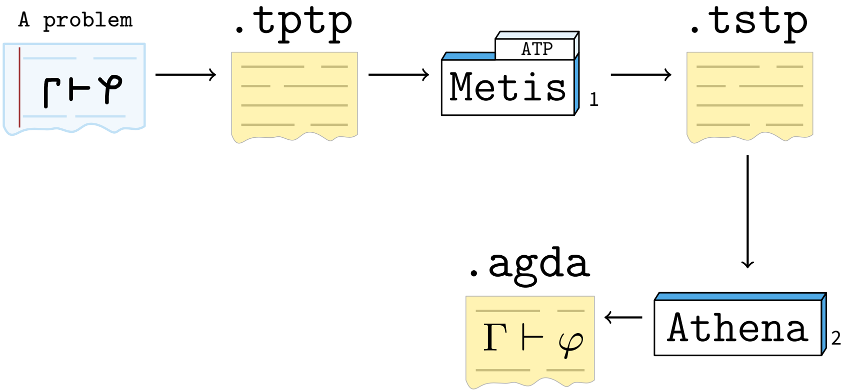
[1] It is available at `http://www.gilith.com/software/metis`



[1] It is available at `http://www.gilith.com/software/metis`

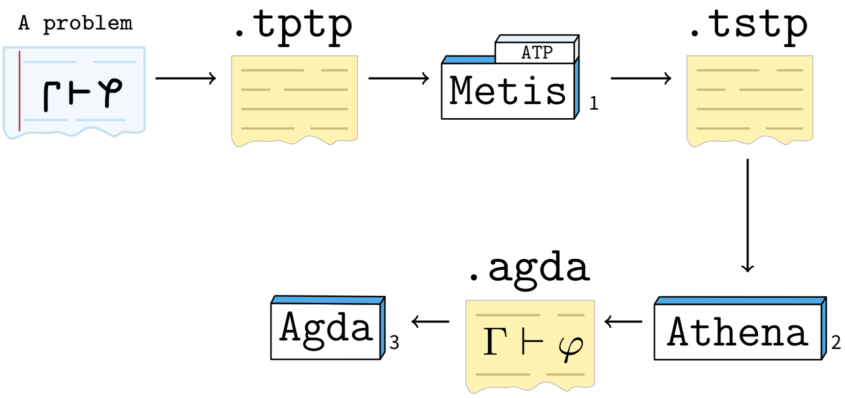
[2] It is available at `http://github.com/jonaprieto/athena`

Proof Reconstruction: Overview



[1] It is available at `http://www.gilith.com/software/metis`

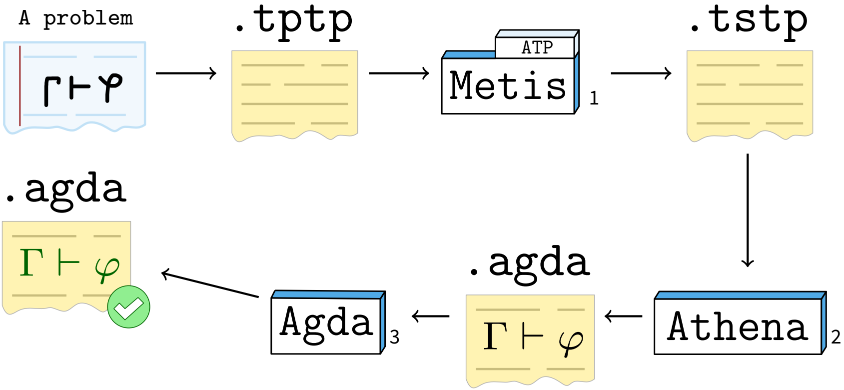
[2] It is available at `http://github.com/jonaprieto/athena`



[1] It is available at **http://www.gilith.com/software/metis**

[2] It is available at **http://github.com/jonaprieto/athena**

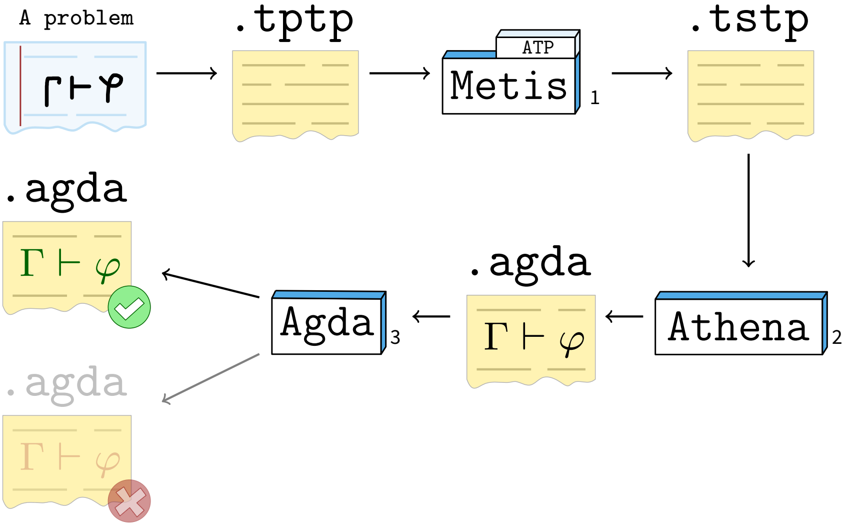
[3] It is available at **http://github.com/agda/agda**



[1] It is available at `http://www.gilith.com/software/metis`

[2] It is available at `http://github.com/jonaprieto/athena`

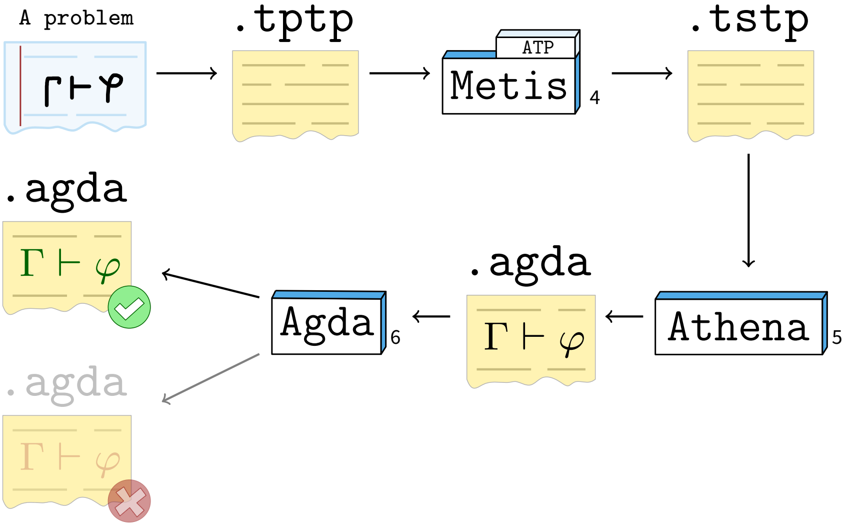
[3] It is available at `http://github.com/agda/agda`



[1] It is available at `http://www.gilith.com/software/metis`

[2] It is available at `http://github.com/jonaprieto/athena`

[3] It is available at `http://github.com/agda/agda`

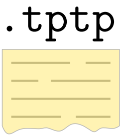


[4] It is available at `http://www.gilith.com/software/metis`

[5] It is available at `http://github.com/jonaprieto/athena`

[6] It is available at `http://github.com/agda/agda`

Bonus Slides



- Is a language[7] 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

[7] Is available at `http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html`

Go Back

- $p \vdash p$

```
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))).
```

[8] Is available at `http://github.com/jonaprieto/prop-pack`

Go Back

`.tstp`

A TSTP derivation [9]

- 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)
```

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

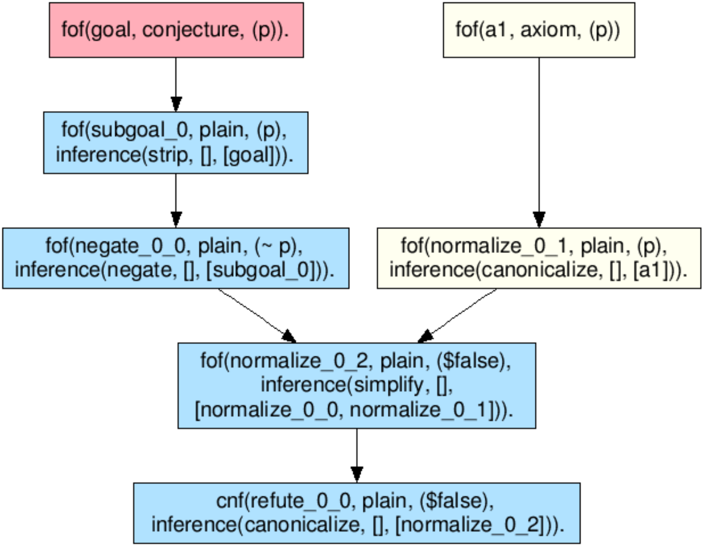
Go Back

- 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])).
```

Go Back

DAG for the previous TSTP derivation found by `Metis` ATP



Go Back

Athena

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

Go Back

Athena

Haskell

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

Go Back

Agda

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

Go Back

Metis Theorem Prover

http://www.gilith.com/software/metis/

ATP Metis

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
`canonicalize`	transforms formulas to CNF, DNF or NNF
`clausify`	performs clausification
`conjunct`	extracts a formula from a conjunction
`negate`	applies negation to the formula
`resolve`	applies theorems of resolution
`simplify`	applies over a list of formula to simplify them
`strip`	splits a formula into subgoals

- Definition

$$conjunct(\phi_1 \wedge \phi_2 \wedge \cdots \wedge \boldsymbol{\varphi_i} \wedge \cdots \wedge \phi, \boldsymbol{\varphi_i}) \longrightarrow \boldsymbol{\varphi_i}$$

- Function:

```
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

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

- The problem is $p \wedge q \vdash q \wedge p$
- TSTP derivation

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

- A natural deduction proof

$$\dfrac{\dfrac{\phi \wedge \psi}{\phi}\ \wedge\text{-}\mathtt{proj}_1 \qquad \dfrac{\phi \wedge \psi}{\psi}\ \wedge\text{-}\mathtt{proj}_2}{\psi \wedge \phi}\ \wedge\text{-}\mathtt{intro}$$

Go Back

Type-checked Proof

```
proof₀ : Γ ⊢ subgoal₀
proof₀ =
  (RAA
    (atp-canonicalize
      (atp-simplify
        (atp-canonicalize
          (atp-strip
            (assume {Γ = Γ}
              (atp-negate subgoal₀))))
        (atp-conjunct (q)
          (atp-canonicalize
            (weaken (atp-negate subgoal₀)
              (assume {Γ = ∅} a)))))))
```

Go Back

Type-checked Proof

```
proof₁ : Γ ⊢ subgoal₁
proof₁ =
  (RAA
    (atp-canonicalize
      (atp-simplify
        (atp-conjunct (q)
          (atp-canonicalize
            (weaken (atp-negate subgoal₁)
              (assume {Γ = ∅} a))))
        (atp-simplify
          (atp-canonicalize
            (atp-strip
              (assume {Γ = Γ}
                (atp-negate subgoal₁))))
          (atp-conjunct (p)
            (atp-canonicalize
              (weaken (atp-negate subgoal₁)
                (assume {Γ = ∅} a))))))))
```

Go Back

Type-checked Proof

Failure Example goes here

Go Back

SledgeHammer

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

Integrating Waldmeister and Agda

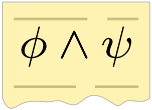
- Source code not available
- Equational Logic
- Reflection Layers

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 (Sicard-Ramírez, 2015).

.agda



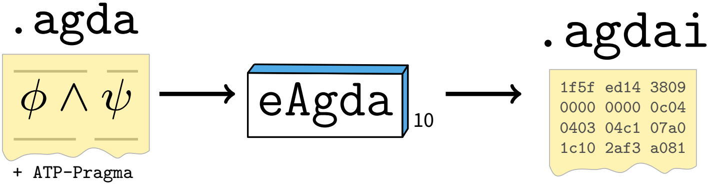
+ ATP-Pragma

```
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 #-}
```

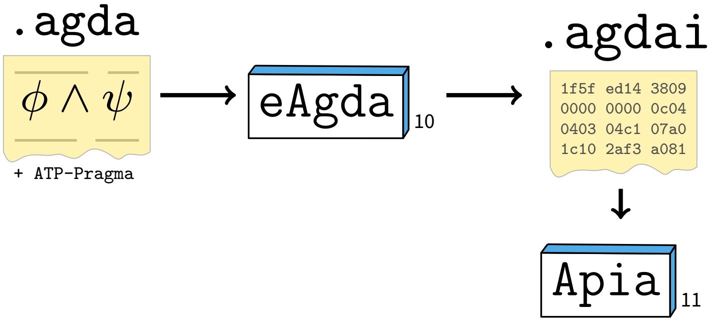
At the moment, the communication between Agda and the ATPs is unidirectional because the ATPs are being used as oracles (ibid.).



[10] 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

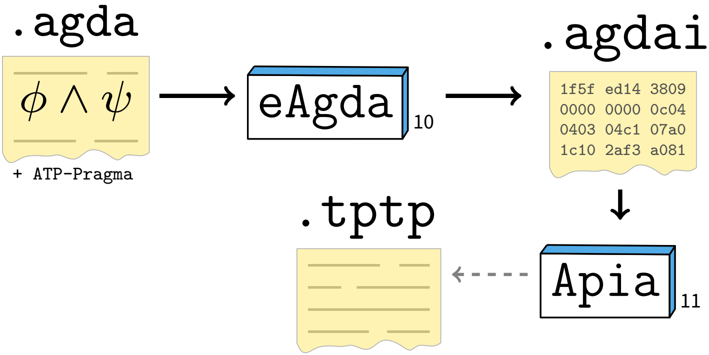


[10] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

[11] Haskell program for proving first-order theorems written in Agda using ATPs. `https://github.com/asr/apia`

Related Work: Apia

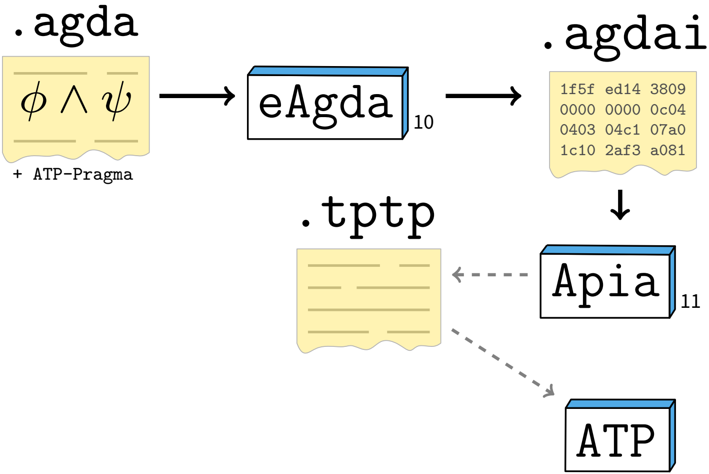
Proving first-order theorems written in Agda using automatic theorem provers for first-order logic



[10] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

[11] Haskell program for proving first-order theorems written in Agda using ATPs. `https://github.com/asr/apia`

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

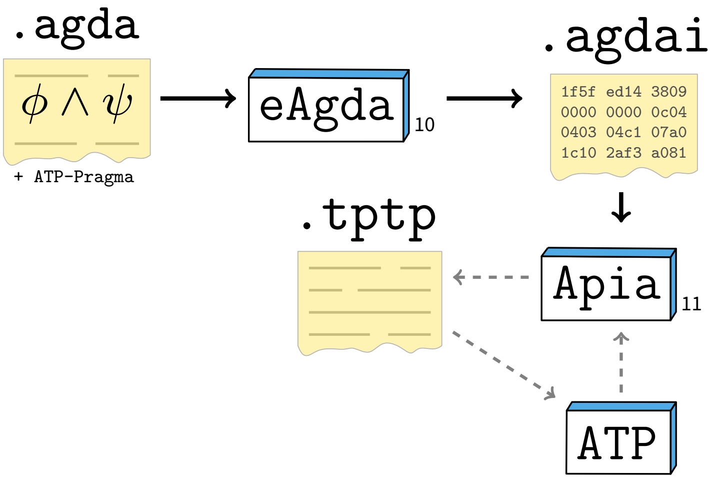


[10] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

[11] 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

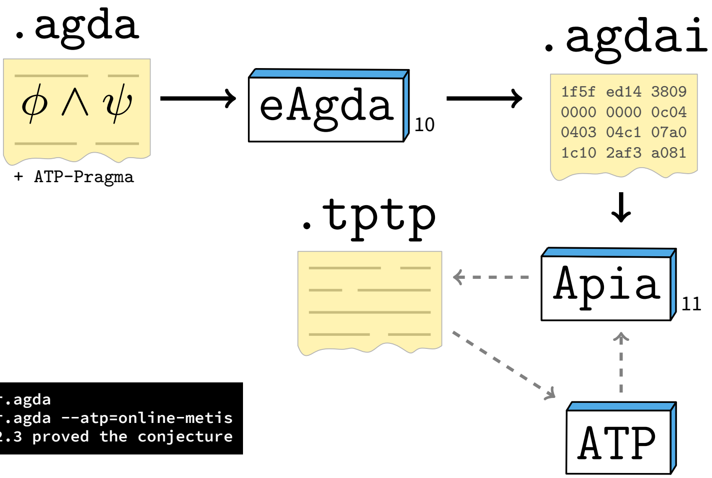


[10] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

[11] 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




```
$ agda Or.agda
$ apia Or.agda --atp=online-metis
Metis---2.3 proved the conjecture
```

[10] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

[11] Haskell program for proving first-order theorems written in Agda using ATPs. `https://github.com/asr/apia`

Future Work

- 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.