

Proof Reconstruction in Classical Propositional Logic

(work in progress)

Jonathan Prieto-Cubides
(joint work with Andrés Sicard-Ramírez)

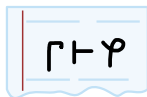
Master in Applied Mathematics
Universidad EAFIT
Medellín, Colombia

Agda Implementors' Meeting XXV
Teknikparken, Chalmers, Gothenburg, Sweden
11 May 2017



CHALMERS
UNIVERSITY OF TECHNOLOGY

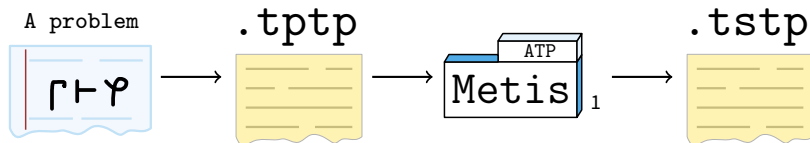
A problem



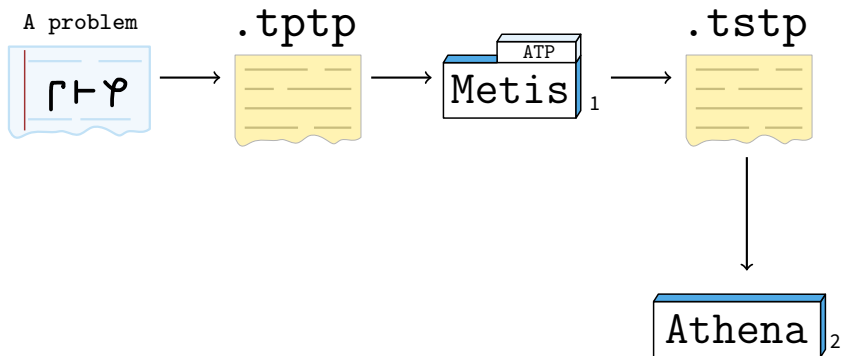




¹<http://www.gilith.com/software/metis>.

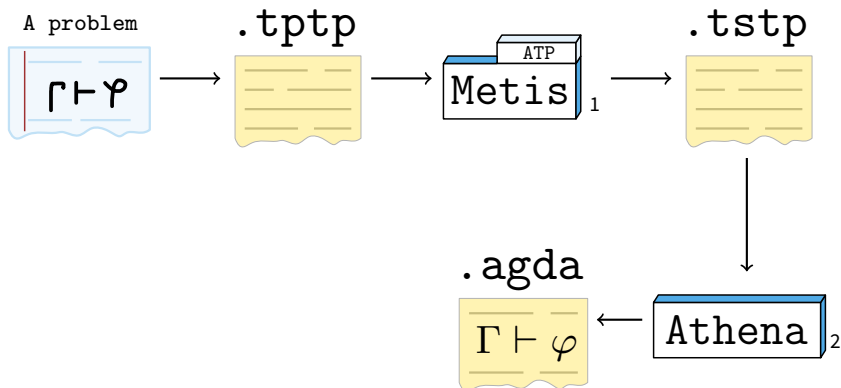


¹<http://www.gilith.com/software/metis>.



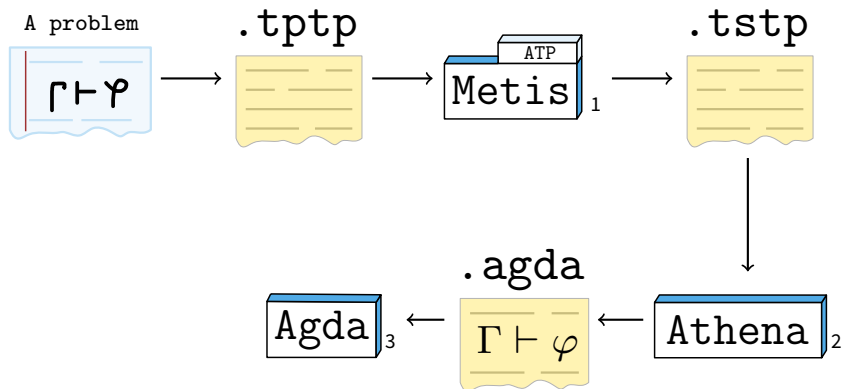
¹<http://www.gilith.com/software/metis>.

²<http://github.com/jonaprieto/athena>.



¹<http://www.gilith.com/software/metis>.

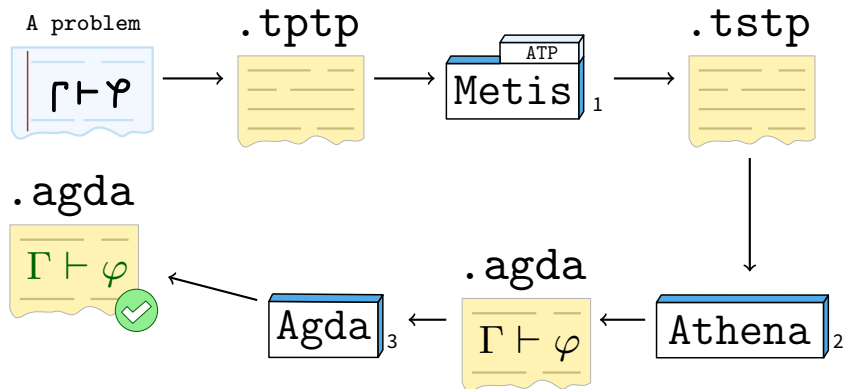
²<http://github.com/jonaprieto/athena>.



¹<http://www.gilith.com/software/metis>.

²<http://github.com/jonaprieto/athena>.

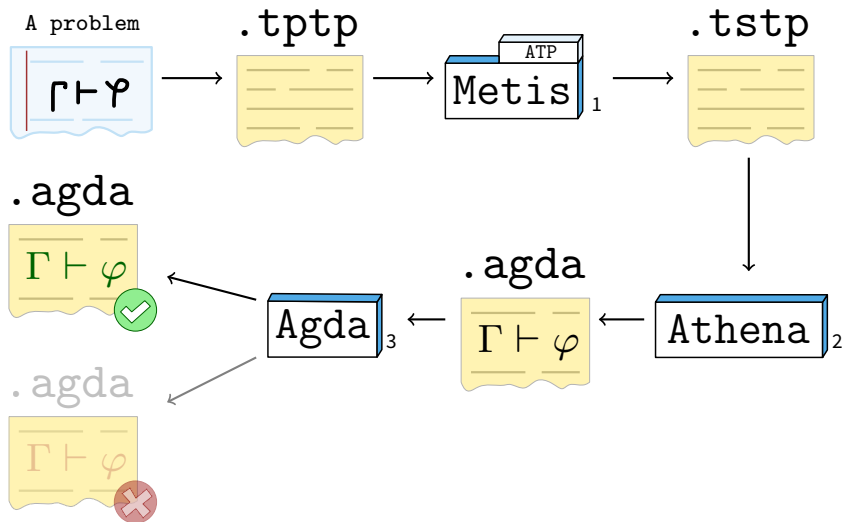
³<http://github.com/agda/agda>.



¹<http://www.gilith.com/software/metis>.

²<http://github.com/jonaprieto/athena>.

³<http://github.com/agda/agda>.

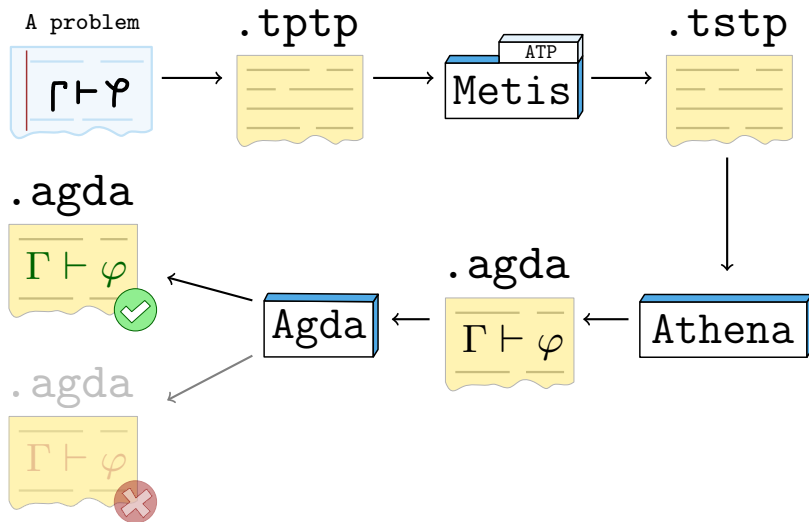


¹<http://www.gilith.com/software/metis>.

²<http://github.com/jonaprieto/athena>.

³<http://github.com/agda/agda>.

- ▶ *Ambiguities*: their output typically omits crucial information, such as which term is affected by rewriting.
- ▶ *Lack of standards*: automatic provers generate different output formats and employ a variety of inference systems
- ▶ *Complexity*: a single automatic prover may use numerous inference rules with complicated behaviors
- ▶ *Problem transformations*: ATPs re-order literals and make other changes to the clauses they are given



.tptp



- ▶ Is a language⁴ 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

formula formula in TPTP format

⁴<http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html>.

► $p \vdash p$

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

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

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



Metis is an automatic theorem prover for First-Order Logic with Equality (Hurd, 2003)

Why Metis?

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

⁵<http://www.gilith.com/software/metis/>.

TSTP derivations by Metis exhibit these inferences⁶

Rule	Purpose
canonicalize	transforms formulas to CNF, DNF or NNF
clausify	performs clausification
conjunct	takes 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

⁶Inference rules found in proofs of Propositional Logic theorems.

.tstp



A TSTP derivation⁷

- ▶ 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

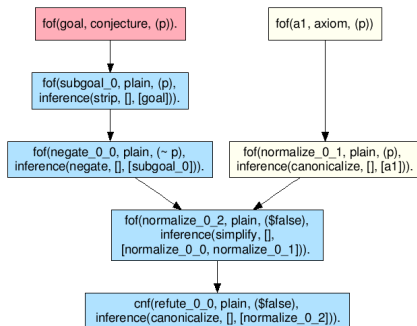
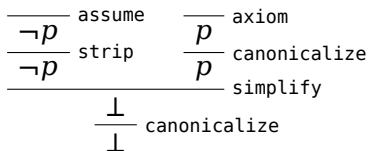
```
inference(rule, useful info, parents).
```

⁷<http://www.cs.miami.edu/~tptp/TPTP/QuickGuide/Derivations.html>.

- 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])).
```

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



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

- ▶ Parsing of TSTP language^{8,9}
- ▶ Creation⁸ 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

⁸<https://github.com/agomezl/tstp2agda>.

⁹<https://github.com/jonaprieto/tstp2agda>.

¹⁰<http://github.com/jonaprieto/athena>.

- ▶ Intuitionistic Propositional Logic + PEM ($\Gamma \vdash \phi \vee \neg \phi$)
- ▶ A data type for formulas

```
data Prop : 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
```

¹¹<https://github.com/jonaprieto/agda-prop>.

- ▶ A data type for theorems

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

- ▶ Constructors

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

- ▶ Natural deduction proofs for more than 71 theorems

```
⇔-equiv, ⇔-assoc, ⇔-comm, ⇒-⇔-∨, ⇔-∨-to-∨,
¬⇔-to-∨, ∨-equiv, ⇒-⇔-∧, ⇔-trans, ∧-assoc,
∧-comm, ∧-dist, ¬∧-to-∨, ¬∨-to-∧, ∨-∧-⇔-∧,
subst-∧1, subst-∧2, ∨-assoc, ∨-comm, ∨-dist,
∨-equiv, ¬∨-to-∧, ¬∧-to-∨, ∨-dmorgan,
¬∨-∧-to-∨, cnf, nnf, dnf, RAA, ...
```

¹²<https://github.com/jonaprieto/agda-prop>.

Rule	Purpose	Theorem
<code>canonicalize</code>	transforms formulas to CNF, DNF or NNF	<code>atp-canonicalize</code>
<code>clausify</code>	performs clausification	<code>atp-clausify</code>
<code>conjunct</code>	takes a formula from a conjunction	<code>atp-conjunct</code>
<code>negate</code>	applies negation to the formula	<code>atp-negate</code>
<code>resolve</code>	applies theorems of resolution	<code>atp-resolve</code>
<code>simplify</code>	applies over a list of formula to simplify them	<code>atp-simplify</code>
<code>strip</code>	splits a formula into subgoals	<code>atp-strip</code>

¹³<https://github.com/jonaprieto/agda-metis>.

► Definition

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

► 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
```

¹⁴<https://github.com/jonaprieto/agda-metis>.


```
atp-conjunct
  :  $\forall \{\Gamma\} \{\varphi\}$ 
   $\rightarrow (\omega : \mathbf{Prop})$ 
   $\rightarrow \Gamma \vdash \varphi$ 
   $\rightarrow \Gamma \vdash \mathbf{conjunct} \ \varphi \ \omega$ 

atp-conjunct  $\{\Gamma\} \{\varphi \wedge \psi\} \ \omega \ \Gamma \vdash \varphi$ 
  with | eq  $\varphi \ \omega$  | | | eq  $\psi \ \omega$  |
... | true |  $\_$  =  $\wedge$ -proj1  $\Gamma \vdash \varphi$ 
... | false | true =  $\wedge$ -proj2  $\Gamma \vdash \varphi$ 
... | false | false =
  atp-conjunct  $\{\Gamma = \Gamma\} \{\varphi = \varphi\} \ \omega \ (\wedge\text{-proj}_1 \ \Gamma \vdash \varphi)$ 
atp-conjunct  $\{\_ \} \{\mathbf{Var} \ x\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\top\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\perp\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\varphi \vee \psi\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\varphi \Rightarrow \psi\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\varphi \Leftrightarrow \psi\} \ \_ = \mathbf{id}$ 
atp-conjunct  $\{\_ \} \{\neg \varphi\} \ \_ = \mathbf{id}$ 
```

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

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

- ▶ A natural deduction proof

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



```

fof(a, axiom, p & q).
fof(goal, conjecture, q & p).
fof(subgoal_0, plain, q,
    inference(strip, [], [goal])).
fof(subgoal_1, plain, q => p,
    inference(strip, [], [goal])).
fof(negate_0_0, plain, ~ q,
    inference(negate, [], [subgoal_0])).
fof(normalize_0_0, plain, (~ q),
    inference(canonicalize, [], [negate_0_0])).
fof(normalize_0_1, plain, p & q,
    inference(canonicalize, [], [a])).
fof(normalize_0_2, plain, q,
    inference(conjunct, [], [normalize_0_1])).
fof(normalize_0_3, plain, $false,
    inference(simplify, [],
        [normalize_0_0, normalize_0_2])).
cnf(refute_0_0, plain, $false,
    inference(canonicalize, [], [normalize_0_3])).
fof(negate_1_0, plain, ~ (q => p),
    inference(negate, [], [subgoal_1])).
fof(normalize_1_0, plain, ~ p & q,

```

```
    inference(canonicalize, [], [negate_1_0])).  
fof(normalize_1_1, plain, p & q,  
    inference(canonicalize, [], [a])).  
fof(normalize_1_2, plain, p,  
    inference(conjunct, [], [normalize_1_1])).  
fof(normalize_1_3, plain, q,  
    inference(conjunct, [], [normalize_1_1])).  
fof(normalize_1_4, plain, $false,  
    inference(simplify, [],  
        [normalize_1_0, normalize_1_2, normalize_1_3])).  
cnf(refute_1_0, plain, ($false),  
    inference(canonicalize, [], [normalize_1_4])).
```

```
p, q, a, goal, subgoal0, subgoal1 : Prop

-- Axiom.
a = (p ∧ q)

-- Premise.
Γ : Ctxt
Γ = [ a ]

-- Conjecture.
goal = (q ∧ p)

-- Subgoals.
subgoal0 = q
subgoal1 = (q ⇒ p)
```

```
a : Prop
a = (p ∧ q)

subgoal₀ : Prop
subgoal₀ = q

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))))))))))
```

```
subgoal1 : Prop
subgoal1 = (q ⇒ p)

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))))))))))
```

```
-- Premise.  
 $\Gamma = [ a ]$   
  
-- Conjecture.  
goal = (q  $\wedge$  p)  
  
-- Subgoals.  
subgoal0 = q  
subgoal1 = (q  $\Rightarrow$  p)  
  
-- Proof  
proof0 :  $\Gamma \vdash$  subgoal0  
proof1 :  $\Gamma \vdash$  subgoal1  
  
proof :  $\Gamma \vdash$  goal  
proof =  
   $\Rightarrow$ -elim  
    atp-splitGoal -- q  $\wedge$  (q  $\Rightarrow$  p)  $\Rightarrow$  p  
    ( $\wedge$ -intro proof0 proof1)
```



```
$ cat problem.tptp
fof(goal, conjecture,
  ((p <=> q) <=> r) <=> (p <=> (q <=> r))).
```

```
$ metis --show proof problem.tptp
...
fof(normalize_2_0, plain,
  (~ p & (~ q <=> ~ r) & (~ p <=> (~ q <=> ~ r))),
  inference(canonicalize, [], [negate_2_0])).
fof(normalize_2_1, plain, ~ p <=> (~ q <=> ~ r),
  inference(conjunct, [], [normalize_2_0])).
fof(normalize_2_2, plain, ~ q <=> ~ r,
  inference(conjunct, [], [normalize_2_0])).
fof(normalize_2_3, plain, ~ p,
  inference(conjunct, [], [normalize_2_0])).
fof(normalize_2_4, plain, $false,
  inference(simplify, [],
    [normalize_2_1, normalize_2_2, normalize_2_3])).
...
```

¹⁵<https://github.com/gilith/metis/issues/2>.

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

$$\frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg p \Leftrightarrow (\neg q \Leftrightarrow \neg r)} \text{ conjunct} \quad \frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg q \Leftrightarrow \neg r} \text{ conjunct} \quad \frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg p} \text{ conjunct}}{\neg p} \text{ simplify}}{\perp}$$

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

$$\frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg p \Leftrightarrow (\neg q \Leftrightarrow \neg r)} \text{ conjunct} \quad \frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg q \Leftrightarrow \neg r} \text{ conjunct} \quad \frac{\frac{\frac{\vdots}{\varphi} \text{ canonicalize}}{\neg p} \text{ conjunct}}{\neg p} \text{ simplify}}{\perp}$$

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

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

SledgeHammer

(Paulson and Susanto, 2007)

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

Integrating Waldmeister into Agda

(Foster and Struth, 2011)

- ▶ Framework for a integration between Agda and ATPs
 - ▶ Equational Logic
 - ▶ Reflection Layers
- ▶ Source code is not available¹⁶

¹⁶<http://simon-foster.staff.shef.ac.uk/agdaatp>.

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

```
module Or where
```

```
data _v_ (A B : Set) : Set where
```

```
  inj₁ : A → A v B
```

```
  inj₂ : B → A v B
```

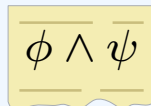
```
postulate
```

```
  A B      : Set
```

```
  v-comm : A v B → B v A
```

```
{-# ATP prove v-comm #-}
```

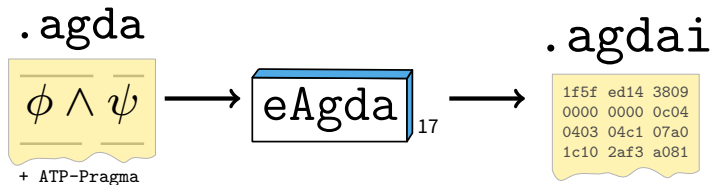
.agda



+ ATP-Pragma

Related Work: Apia

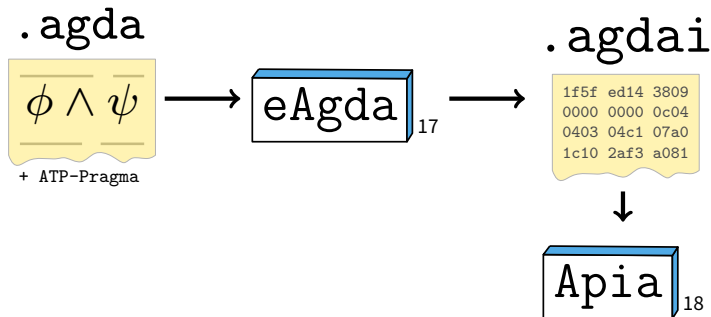
Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic



¹⁷<https://github.com/asr/eagda>.

Related Work: Apia

Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic

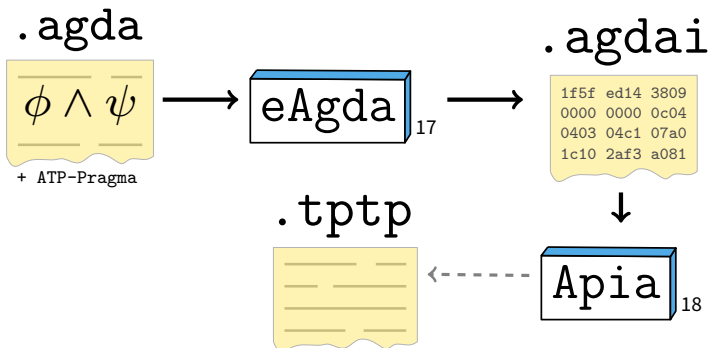


¹⁷<https://github.com/asr/eagda>.

¹⁸<https://github.com/asr/apia>.

Related Work: Apia

Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic

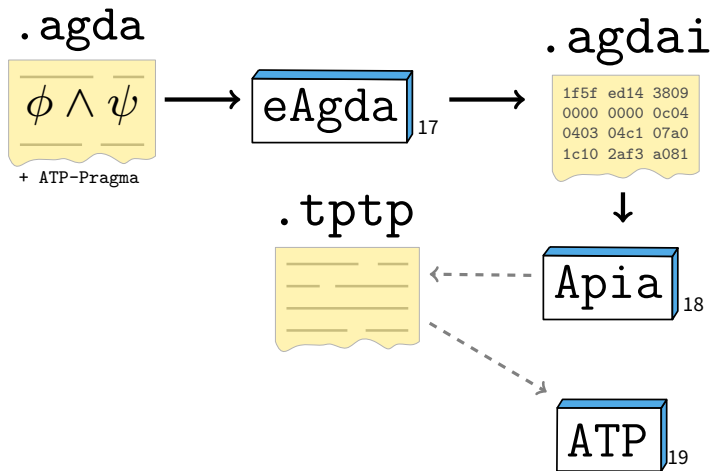


¹⁷<https://github.com/asr/eagda>.

¹⁸<https://github.com/asr/apia>.

Related Work: Apia

Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic



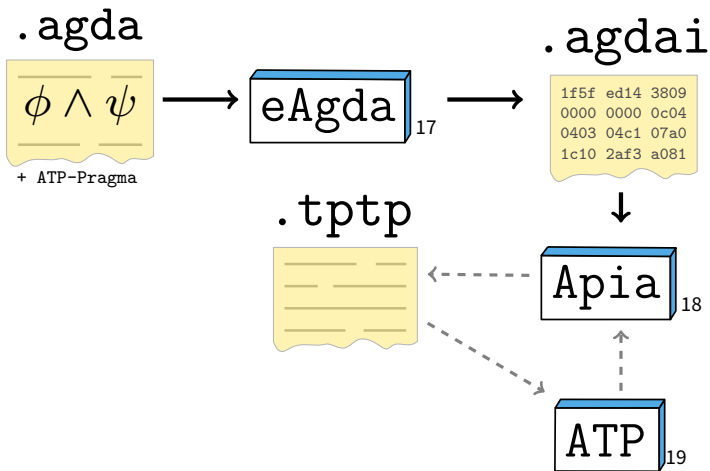
¹⁷<https://github.com/asr/eagda>.

¹⁸<https://github.com/asr/apia>.

¹⁹<http://github.com/jonaprieto/online-atps>.

Related Work: Apia

Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic



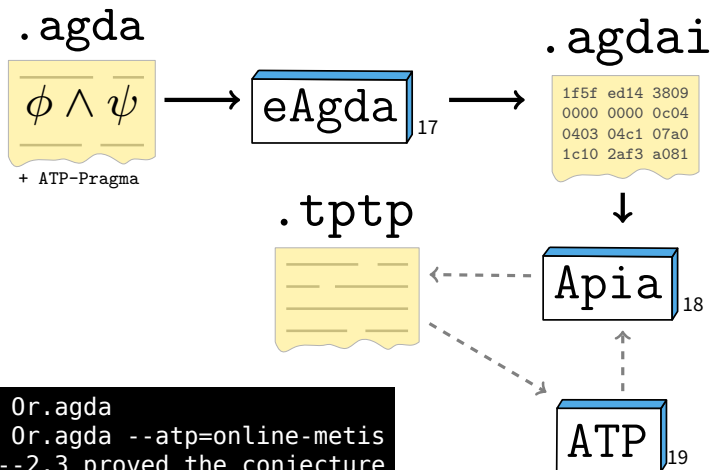
¹⁷<https://github.com/asr/eagda>.

¹⁸<https://github.com/asr/apia>.

¹⁹<http://github.com/jonaprieto/online-atps>.

Related Work: Apia

Proving First-Order theorems written in Agda using automatic theorem provers for First-Order Logic



¹⁷<https://github.com/asr/eagda>.

¹⁸<https://github.com/asr/apia>.

¹⁹<http://github.com/jonaprieto/online-atps>.

- ▶ Complete implementation for `simplify` inference²⁰
- ▶ Complete implementation for `canonicalize` inference: what normal form use to transform the formulas
- ▶ Complete implementation for Splitting a goal in a list of subgoals

²⁰<https://github.com/gilith/metis/issues/3>.

Name	Purpose
Agda-Metis	Implementation of Metis inference rules
Agda-Prop	Syntax and theorems of Classical Propositional Logic
Athena	Translator for Metis TSTP files to Agda
OnlineATPs	Client to use ATPs from SystemOnTPTP of TPTP World
Prop-Pack	Collection of TPTP problems to test Athena

- ▶ Integration with Apia
- ▶ Support First-Order Logic with Equality
- ▶ Support another prover like EProver or Vampire



Foster, Simon and Georg Struth (2011). “Integrating an Automated Theorem Prover into Agda”. In: *NASA Formal Methods: Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*. Ed. by Mihaela Bobaru et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 116–130.



Hurd, Joe (2003). “First-order proof tactics in higher-order logic theorem provers”. In: *Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports*, pp. 56–68.



Paulson, Lawrence C. and Kong Woei Susanto (2007). “Source-Level Proof Reconstruction for Interactive Theorem Proving”. In: *Theorem Proving in Higher Order Logics: 20th International Conference, TPHOLs 2007, Kaiserslautern, Germany, September 10-13, 2007. Proceedings*. Ed. by Klaus Schneider and Jens Brandt. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 232–245.



Sicard-Ramírez, Andrés, Ana Bove, and Peter Dybjer (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.

$$\frac{}{C} \text{ axiom}$$

$$\frac{}{L \vee \neg L} \text{ assume } L$$

$$\frac{C}{\sigma C} \text{ subst } \sigma$$

$$\frac{L \vee C \quad \neg L \vee D}{C \vee D} \text{ resolve } L$$

$$\frac{}{t = t} \text{ refl } t$$

$$\frac{}{\neg(L[p] = t) \vee \neg L \vee L[p \mapsto t]} \text{ eq } L \ p \ t$$