Proof Reconstruction in Classical Propositional Logic

Jonathan Prieto-Cubides

Advisor: Andrés Sicard-Ramírez

Universidad EAFIT
Medellín, Colombia

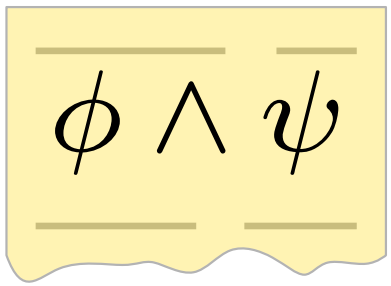
Agda Implementors' Meeting XXV
May 9-15th

Outline

Translation

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-Pragmas

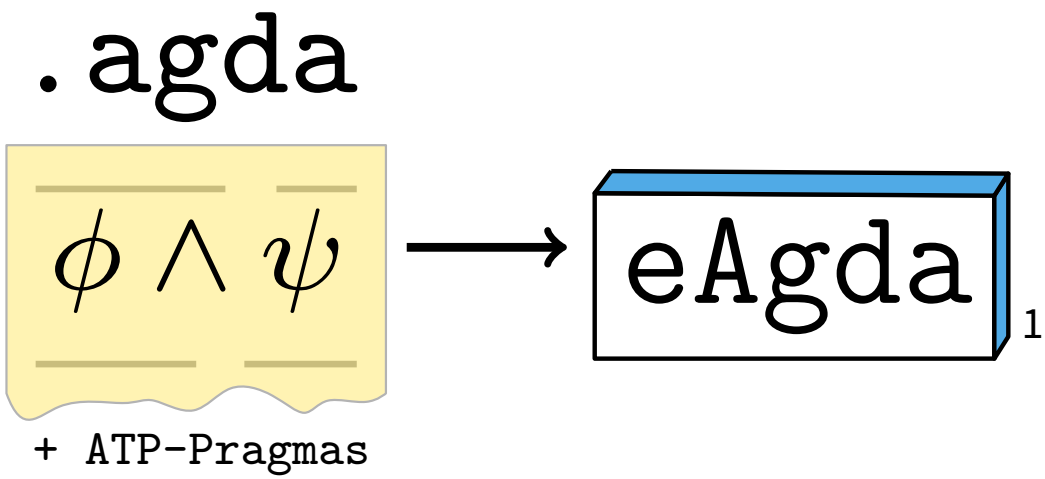
```
$ cat Or.agda
module Or where

data _or_ (A B : Set) : Set where
  inj1 : A -> A or B
  inj2 : B -> A or B

postulate
  A B    : Set
  or-comm : A or B -> B or A
{-# ATP prove or-comm #-}
```

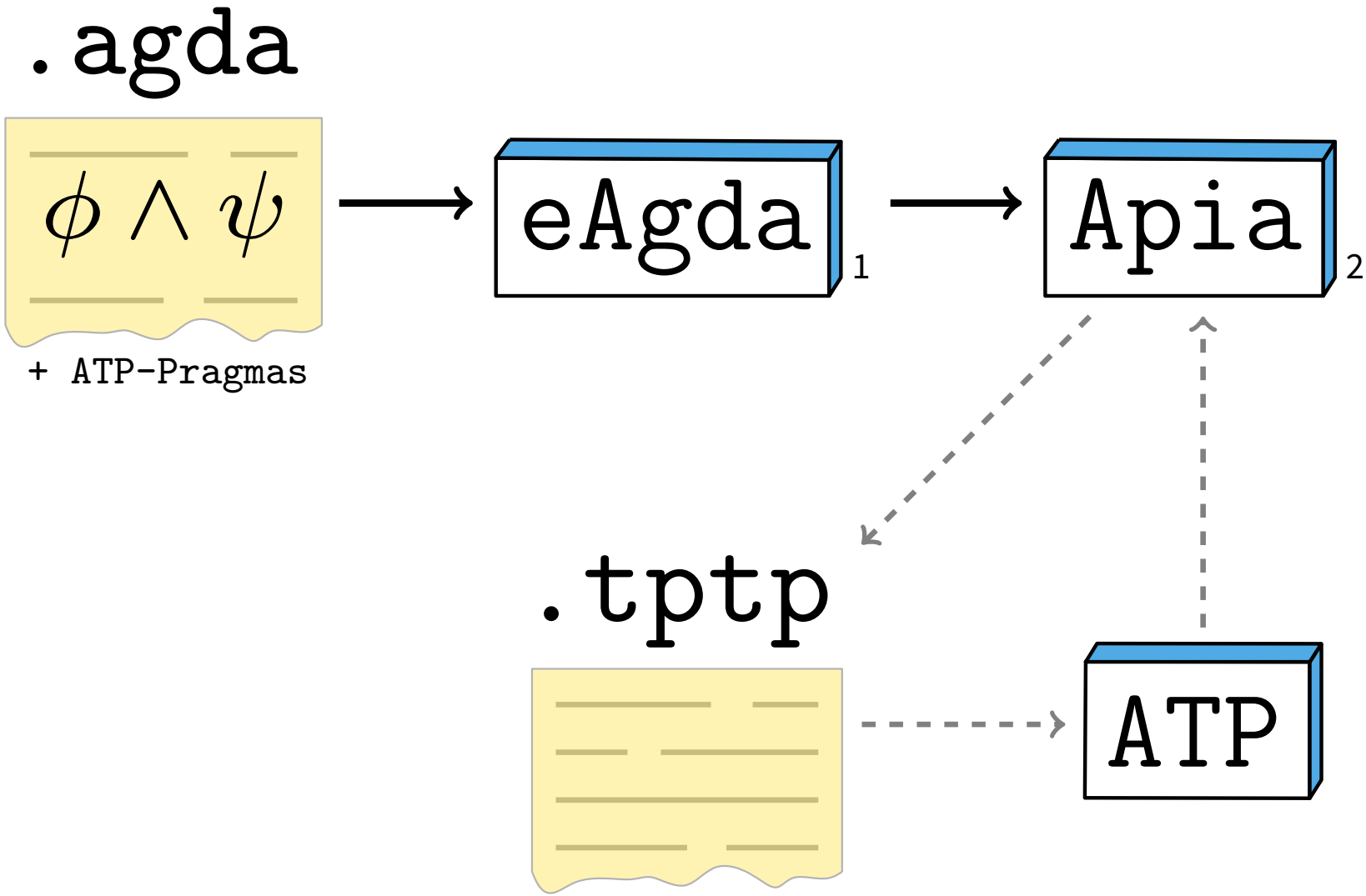
Translation

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



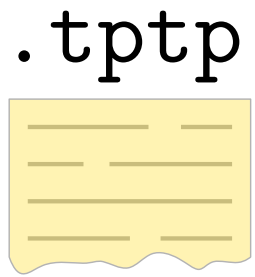
[1] Development version of Agda in order to handle a new built-in ATP-pragma. `https://github.com/asr/eagda`

Translation



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

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



- Is a language[3] to encode problems in text files
- Is the input of the ATPs
- his problems contains formulas with the form

```
language(name, role, formula).
```

`language`	THF, TFF, FOF, or CNF
`name`	to identify the formula within the problem
`role`	axiom, definition, hypothesis, conjecture, among others
`formula`	the logic formula in the language

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

Problems in Propositional Logic:

- $p \vdash p$

```
$ cat basic-4.tptp
fof(a, axiom, p).
fof(goal, conjecture, p).
```

- $p \wedge q \vdash q \wedge p$

```
$ cat conj-3.tptp
fof(a, axiom, p & q).
fof(goal, conjecture, q & p).
```

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

```
$ cat neg-7.tptp
fof(goal, conjecture, ~ ((p & ~ p) | (q & ~ q))).
```

.tstp

A TSTP derivation [4]

- Is a Directed Acyclic Graph where

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

- Is a list of annotated formulae:

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

`source` typically is a inference record:

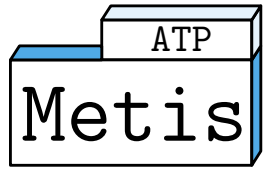
```
inference(rule, [], [])
```

- The inference rule name,
- a list of useful inference information, and
- a list of references to its parent formulae.

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

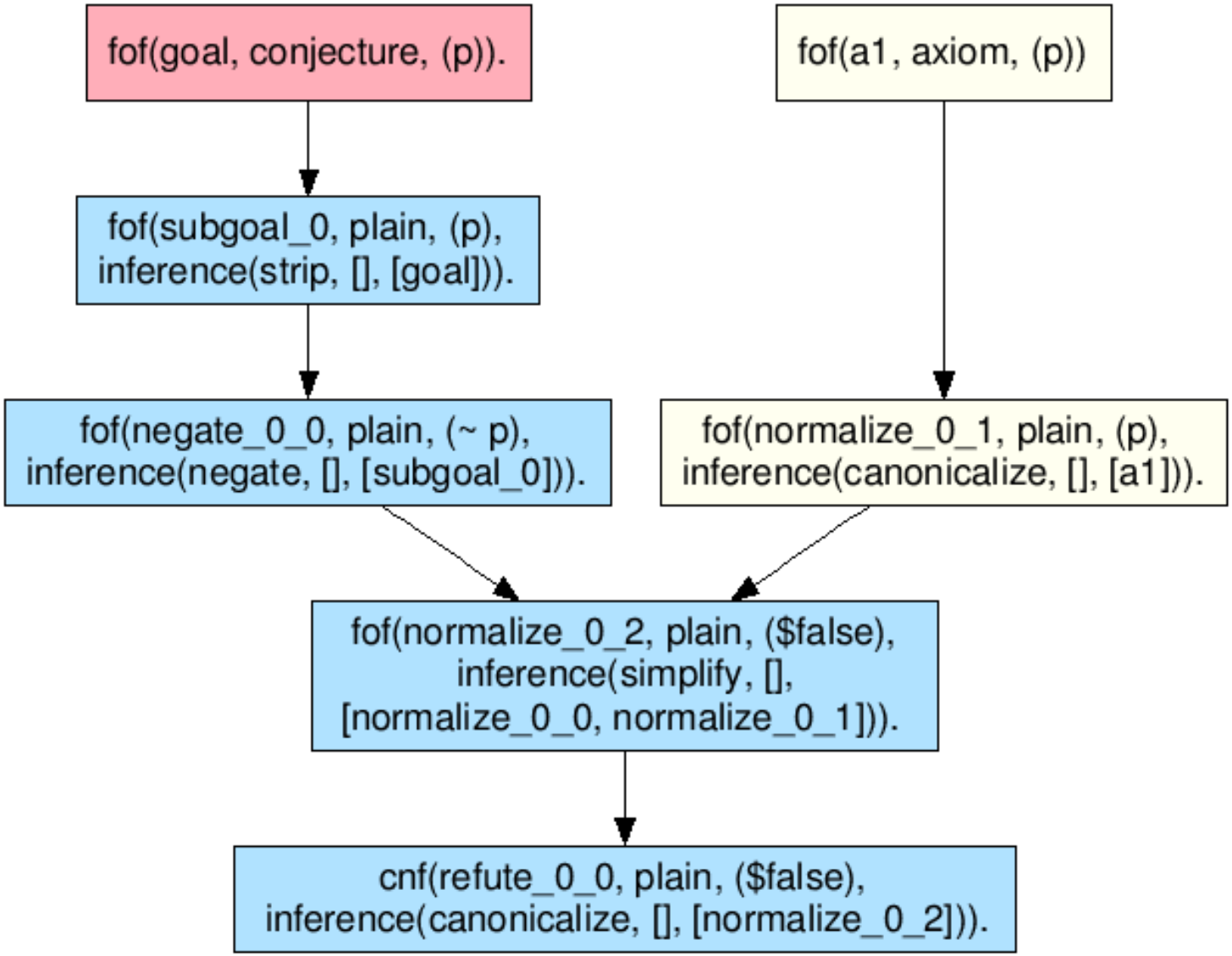
TSTP Example

- 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 `Metis` ATP



Programming Languages

- **`Haskell`** is a standardized, general-purpose purely functional programming language.
  Our usages:
  - Parsing
  - AST construction
  - Creation and analysis of DAG derivations
  - Analysis of inference rules used
  - Generation of Agda code of the proof
- **`Agda`** is dependently typed functional programming language and it also a proof assistant.
  Our usages:
  - Logic of Classical Propositional Logic
  - Porting of Metis' inference rules

References

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