Proof Reconstruction: Translating Proofs

Alejandro Gómez-Londoño

Advisor - Andrés Sicard-Ramírez

EAFIT University

August 18, 2015

Introduction

A (very) general idea of the context

Proof assistant

ATP

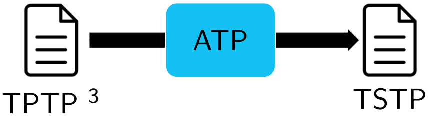
An interactive prover is a software tool aiding the development of formal proofs by man-machine collaboration.[1]

[1] Matita development team, Matita website, http://matita.cs.unibo.it/index.shtml

Deals with the development of computer programs that show that some statement (the conjecture) is a logical consequence of a set of statements (the axioms and hypotheses).[2]

[2] http://www.cs.miami.edu/~tptp/OverviewOfATP.html

Introduction
ATPs input/output



[3]Sutcliffe, G. The TPTP Problem Library and Associated In-frastructure: The FOF and CNF Parts. 2009.

Introduction
Examples

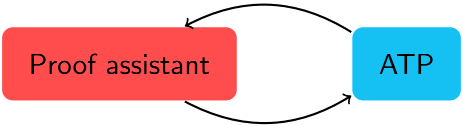
ATPs:

- Vampire
- E
- Metis
- SPASS
- Equinox

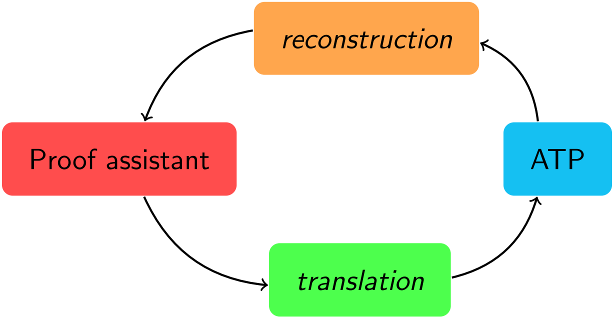
Proof assistants:

- Coq
- Agda
- Isabelle
- Mizar
- NuPRL

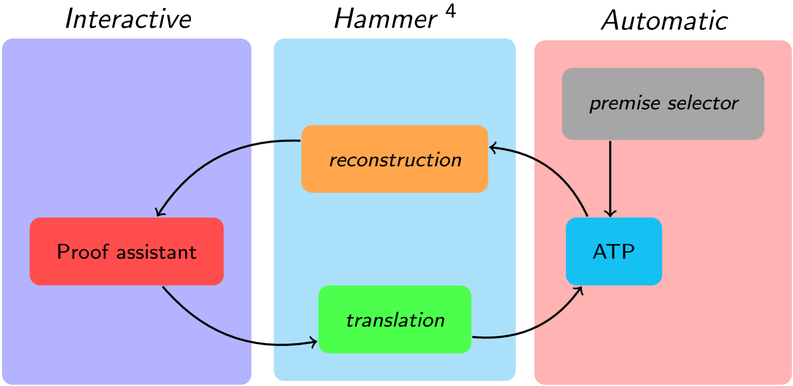
Introduction



Introduction



Introduction



[4] Jasmin C. Blanchette, Cezary Kaliszyk and Lawrence C. Paulson, Hammering towards QED. 2014.

Proof reconstruction

Example

- Hand written proof

$$\dfrac{\dfrac{x \qquad y}{x \wedge y} \qquad x \wedge y \Rightarrow z}{z}$$

- TPTP problem

```
fof(a_0,axiom,x).
fof(a_1,axiom,y).
fof(a_2,axiom, ((x & y) => z)).
fof(c_0,conjecture, z).
```

Proof reconstruction

Example

- TSTP proof

```
fof(s_0,plain,(x & y),
    inference(conjunction,[],[a_0,a_1])).

fof(s_1,plain,(z),
    inference(modus_ponens,[],[a_2,s_0])).

fof(r_0,plain,($true),
    inference(simplify,[],[s_1,c_0])).
```

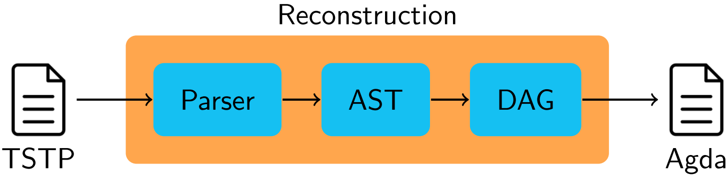
Proof reconstruction
Example

- Agda proof

```
--conjunction
data _∧_ (P : Set) (Q : Set) : Set where
    ∧-intro : P ∧ Q ∧ (P ∧ Q)

proof : { X Y Z : Set} →
        X → Y → ( X ∧ Y → Z) → Z
proof x y f = f ( ∧-intro x y )
```

Proof reconstruction
Implementation



Proof reconstruction

Parser and AST construction

```
fof(a_0,axiom,x).
fof(a_1,axiom,y).
fof(a_2,axiom, ((x & y) => z)).
fof(c_0,conjecture, z).

fof(s_0,plain,(x & y),
    inference(conjunction,[],[a_0,a_1])).

fof(s_1,plain,(z),
    inference(modus_ponens,[],[a_2,s_0])).

fof(r_0,plain,($true),
    inference(simplify,[],[s_1,c_0])).
```

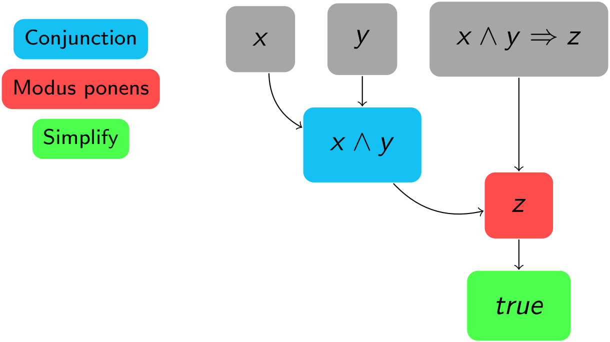
Proof reconstruction

Parser and AST construction

```
[
  F {name    = "s_0",
     role    = Plain,
     formula = "x" (:&:) "y",
     annotations = Conjunction ["a_0","a_1"]
    },
  F {name    = "s_1",
     role    = Plain,
     formula = "z",
     annotations = ModusPonens ["a_2","s_0"]
    },
  F {name    = "r_0",
     role    = Plain,
     formula = "$True",
     annotations = Simplify ["s_1", "c_0"]
    }
]
```

Proof reconstruction

DAG



Past results

Haskell and Agda were chosen as the programing languages for the implementation.

- Haskell was used for parsing and AST construction
- In Agda we will create and analyze the DAG.

Past results

Metis[5] was chosen as our ATP

- Uses TPTP as input format.
- Outputs proofs in TSTP format.
- Each refutation step is one of 6 rules.
- Has respectable performance.

[5] Joe Hurd. First-Order Proof Tactics in Higher-Order Logic Theorem Provers. 2003.

Past results

A modified version of the `logic-tptp`[6] Haskell library was used to implement a TSTP parser capable of analyze Metis proofs.

- This project is freely available on github[7].

[6] https://hackage.haskell.org/package/logic-TPTP

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

Objectives

Translate into idiomatic Agda code the AST resulting from the parsing of an ATP-generated proof.

- Translate to Agda code the Haskell AST data type.
- Build an Agda library that implements the logical kernel of the ATP.
- Reconstruct the proof in Agda using the aforementioned library.