



DEDUKTI: A Universal Proof Checker

Mathieu Boespflug Quentin Carbonneaux Gilles Dowek
Olivier Hermant [Ronan Saillard](#)

Deducteam INRIA

MINES ParisTech

January 9, 2013

WHAT IS DEDUKTI?

hal-00833992, version 1 - 14 Jun 2013

What is DEDUKTI?

- ▶ A type-checker for the $\lambda\Pi$ -calculus modulo
- ▶ A theory-independent proof-checker
- ▶ With CoqInE a tool to recheck proofs from Coq
- ▶ With HOLidE a tool to recheck proofs from HOL/OpenTheory
- ▶ A logical framework with rewrite rules
- ▶ A framework to study interoperability
- ▶ A type-checker with new implementation techniques

TABLE OF CONTENTS

hal-00833992, version 1 - 14 Jun 2013

WHAT IS DEDUKTI?

THE $\lambda\Pi$ -CALCULUS MODULO

AN EXAMPLE

AN EFFICIENT AND VERSATILE IMPLEMENTATION

- The big picture

- The type-checking algorithm

- Versatility

CONCLUSION

The $\lambda\Pi$ -calculus modulo

- ▶ type theory: the $\lambda\Pi$ -calculus (dependent types):

$$\text{array} : \text{nat} \rightarrow \text{Type}$$

- ▶ enriched with **rewrite rules**

$$\text{head} (hd :: tl) \longrightarrow hd$$

- ▶ used to generalize the **conversion** rule
- ▶ can encode all the Functional PTSes [[Cousineau & Dowek, 2007](#)]

TYPING RULES

hal-00833992, version 1 - 14 Jun 2013

$s \in \{\text{Type}, \text{Kind}\}$

$$(sort) \frac{\Gamma \text{ Well-Formed}}{\Gamma \vdash \text{Type} : \text{Kind}} \quad (var) \frac{\Gamma \text{ Well-Formed} \quad x:A \in \Gamma}{\Gamma \vdash x : A}$$

$$(prod) \frac{\Gamma \vdash A : \text{Type} \quad \Gamma, x:A \vdash B : s}{\Gamma \vdash \Pi x:A. B : s}$$

$$(abs) \frac{\Gamma \vdash A : \text{Type} \quad \Gamma, x:A \vdash B : s \quad \Gamma, x:A \vdash M : B}{\Gamma \vdash \lambda x:A. M : \Pi x:A. B}$$

$$(app) \frac{\Gamma \vdash M : \Pi x:A. B \quad \Gamma \vdash N : A}{\Gamma \vdash M N : \{N/x\}B}$$

$$(conv) \frac{\Gamma \vdash M : A \quad \Gamma \vdash A : s \quad \Gamma \vdash B : s \quad A \equiv_{\beta\mathcal{R}} B}{\Gamma \vdash M : B}$$

FIGURE: Typing rules for the $\lambda\Pi$ -calculus modulo

An Example

AN EXAMPLE

hal-00833992, version 1 - 14 Jun 2013

example.dk

```
Nat : Type.
Z   : Nat.
S   : Nat -> Nat.
two : Nat := S (S Z).

Bool : Type.
true  : Bool.
false : Bool.
prf   : Bool -> Type.
tt    : prf true.

eq : Nat -> Nat -> Bool.
[ ] eq Z Z --> true
[n:Nat, m:Nat] eq (S n) (S m) --> eq n m
[n:Nat] eq (S n) Z --> false
[n:Nat] eq Z (S n) --> false.

List : n:Nat -> Type.
Nil   : List Z.
Cons  : n:Nat -> x:Nat -> List n -> List (S n).

head : n:Nat -> List (S n) -> Nat.
[n:Nat, x:Nat, tl:List n] head n (Cons {n} x tl) --> x.

theorem0 : prf (eq (head Z (Cons Z two Nil)) two) := tt.
```


Implementation

The big picture

DEDUKTI: GOALS AND WEAPONS

hal-00833992, version 1 - 14 Jun 2013

Goals:

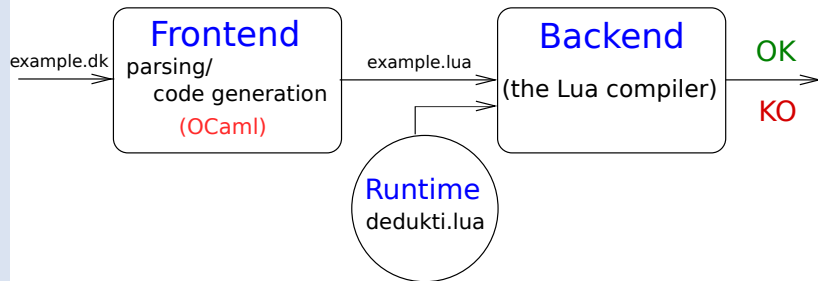
- ▶ Fast type checking.
- ▶ Versatility: efficient for any (embedded) logic.

Philosophy, be lazy:

- ▶ Do not reimplement long-standing features.
- ▶ Reuse existing tools.

THE BIG PICTURE

hal-00833992, version 1 - 14 Jun 2013



- ▶ Dedukti is a proof-checker **generator**

ADVANTAGES

hal-00833992, version 1 - 14 Jun 2013

Use of the source language capability:

- ▶ **Higher Order Abstract Syntax** (HOAS).
- ▶ **Normalisation by Evaluation** (NbE).

| Dedukti | Lua |
|--|---|
| $(\lambda x.M)N$ | <code>(function (x) M end)(N)</code> |
| $head (hd :: tl) \longrightarrow hd$ $head nil \longrightarrow default$ | <pre>function head (x) if x.id == Cons and then return x.args[1] elseif x.id = Nil return default end end</pre> |

Implementation

Type-checking algorithm

BIDIRECTIONAL/CONTEXT-FREE TYPE CHECKING

hal-00833992, version 1 - 14 Jun 2013

Bidirectional type checking:

- ▶ Mix of type-checking and type-inference.
- ▶ Smaller terms (Curry-style).

Context-Free type checking:

- ▶ No search in contexts.
- ▶ Type annotation for free variables.

THE RESULTING SYSTEM

$\boxed{\vdash M \Rightarrow A}$ the term M synthesizes type A

$$(sort) \frac{}{\vdash \text{Type} \Rightarrow \text{Kind}}$$

$$(var) \frac{}{\vdash [x : A] \Rightarrow A}$$

$$(app) \frac{\vdash M \Rightarrow C \quad C \xrightarrow{*}_w \Pi x:A. B \quad \vdash N \Leftarrow A}{\vdash M N : \{N/x\}B}$$

$\boxed{\vdash M \Leftarrow A}$ the terms M verifies type A

$$(abs) \frac{C \xrightarrow{*}_w \Pi x:A. B \quad \vdash \{[y : A]/x\}M \Leftarrow \{y/x\}B}{\vdash \lambda x.M \Leftarrow \Pi x:A. B}$$

$$(prod) \frac{\vdash A \Leftarrow \text{Type} \quad \vdash \{[y : A]/x\}B \Leftarrow s}{\vdash \Pi x:A. B \Leftarrow s} \quad (s \in \{\text{Type}, \text{Kind}\})$$

$$(conv) \frac{\vdash N \Rightarrow B \quad A \equiv B}{\vdash N \Leftarrow A}$$

FIGURE: An implementation of the $\lambda\Pi$ -calculus modulo

Implementation

Versatility

The conversion test: how to normalize ?

- ▶ Proofs terms with few reductions (ex: from HOL).
 - ▶ Best technique: **interpretation** of the λ -terms.
- ▶ Proofs terms with a long reduction sequence (ex: proof by reflection (Coq)).
 - ▶ Best technique: **compilation** and execution of the λ -terms.

How to choose the correct strategy?

THE JIT COMPROMISE

- ▶ **compile** the computational parts, **interpret** the rest
- ▶ **choice delegated** to a cutting edge JIT: luajit

| File | Time to process |
|---------------------------------|--------------------|
| <code>fact.hs</code> | 0.7 sec + 0.04 sec |
| <code>fact.lua</code> | 0.7 sec |
| <code>fact.vo</code> | 3.3 sec |
| <code>Coq_Init_Logic.hs</code> | 45 sec + 0.4 sec |
| <code>Coq_Init_Logic.lua</code> | 0.4 sec |
| <code>Coq_Init_Logic.vo</code> | 0.14 sec |

FIGURE: **Compilation** vs **JIT** vs **Interpretation**

`Coq_Init_Logic` is a module in Coq's prelude, `fact` typechecks the identity with the type `vec 8! → vec 8!`.

Conclusion

CONCLUSION

hal-00833992, version 1 - 14 Jun 2013

CONTRIBUTIONS

- ▶ Proof-checker generator architecture.
- ▶ A algorithm mixing bidirectional and context-free type checking.
- ▶ Use of a JIT compiler.

FURTHER WORK

- ▶ Non-linear rewrite rules.
- ▶ Check Coq' Standard Library.
- ▶ LuaJIT limits.
- ▶ More optimisations (convertibility test/normalisation).

DEDUKTI'S WEBSITE

<https://www.rocq.inria.fr/deducteam/Dedukti/>

QUESTIONS ?

hal-00833992, version 1 - 14 Jun 2013

QUESTIONS ?