A Universal Proof Framework

Ali Assaf

INRIA Paris-Rocquencourt, Deducteam

Outline

Introduction

A closer look

My work

Whack-A-Bug

```
def binary_search(A, x, left, right):
  middle = (left + right) / 2
  if A[middle] > x:
    return binary_search(A, x, left, middle)
  elif A[middle] < x:
    return binary_search(A, x, middle, right)
  else:
    return middle
```

Whack-A-Bug

```
def binary_search(A, x, left, right):
  middle = (left + right) / 2
  if A[middle] > x:
    return binary_search(A, x, left, middle)
  elif A[middle] < x:
    return binary_search(A, x, middle, right)
  else:
    return middle

A = [2, 3, 5, 7]
binary_search(A, 7, 0, 3) # Oops!
```

Whack-A-Bug

```
def binary_search(A, x, left, right):
  middle = (left + right) / 2
  if A[middle] > x:
    return binary_search(A, x, left, middle - 1)
  elif A[middle] < x:
    return binary_search(A, x, middle + 1, right)
  else:
    return middle

A = [2, 3, 5, 7]
binary_search(A, 7, 0, 3) # Oops!
```

Proof of correctness

Theorem

If $x \in A$ *and* $left \leq right$ *and* $A[left] \leq x \leq A[right]$ *then* $A[\text{binary\_search}(A, x, left, right)] = x$.

Proof.

By induction on $(right - left)$ ... □

Formal proof

Formal language: $\forall$, $\exists$, $\implies$, ...

Deductive system:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \implies B} \qquad \frac{\Gamma \vdash A \implies B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Proof systems

Theoretical systems

- First-order logic
- Higher-order logic
- Calculus of constructions

Proof systems

Theoretical systems

- First-order logic
- Higher-order logic
- Calculus of constructions

Implementations

- Twelf
- HOL
- Coq

Why use them?

In software engineering: eliminate more bugs

- CompCert project
- L4.verified project

In mathematics: prove harder theorems

- 4-color theorem
- Kepler theorem

Outline

Two problems

Proof checking:

- Given a proposition A and a proof $\mathcal{D}$, does $\mathcal{D}$ prove A?
- $\mathit{Check}(D, A) = \mathit{Yes}$ or No

Two problems

Proof checking:

- Given a proposition A and a proof $\mathcal{D}$, does $\mathcal{D}$ prove A?
- $Check(D, A) = Yes$ or No

Proof search:

- Given a proposition A, does there exist a proof $\mathcal{D}$ of A?
- $Search(A) = D$ or $None$

Writing proofs

```
Theorem foo: forall A B C : Prop,
  (A -> B) -> (B -> C) -> (A -> C) :=

fun A B C proof_of_A_B proof_of_B_C =>
fun proof_of_A =>

let proof_of_B := (proof_of_A_B proof_of_A) in
let proof_of_C := (proof_of_B_C proof_of_B) in
proof_of_C.
```

Writing proofs

```
Theorem foo: forall A B C : Prop,
  (A -> B) -> (B -> C) -> (A -> C) :=

fun A B C proof_of_A_B proof_of_B_C =>
fun proof_of_A =>

let proof_of_B := (proof_of_A_B proof_of_A) in
let proof_of_C := (proof_of_B_C proof_of_B) in
proof_of_C.
```

Writing proofs

```
Theorem foo: forall A B C : Prop,
  (A -> B) -> (B -> C) -> (A -> C) :=

fun A B C proof_of_A_B proof_of_B_C =>
fun proof_of_A =>

let proof_of_B := (proof_of_A_B proof_of_A) in
let proof_of_C := (proof_of_B_C proof_of_B) in
proof_of_C .
```

Proof development

1. Write your proof
2. Call the proof checker
3. Checker answers *Yes* or *No* and gives you an error
4. If answer is *No*, go back to step 1

Proof development

1. Write your proof
2. Call the proof checker
3. Checker answers *Yes* or *No* and gives you an error
4. If answer is *No*, go back to step 1

Sounds familiar?

Program development

1. Write your programs
2. Call the compiler
3. Compiler answers *Yes* or *No* and gives you an error
4. If answer is *No*, go back to step 1

Sounds familiar?

Proofs are programs!

Curry-Howard correspondence

Proof	Program
Proposition	Type
$A \implies B$	$A \to B$
Proof checking	Type checking

Proofs are programs!

Proof system = Programming language

Outline

A Zoology of proof systems

- Twelf, HOL, Coq, Isabelle, PVS, NuPRL, Mizar, Agda, ProofPower, Lego, ACL2, ...
- Different properties
  - Intuitionistic/Classical
  - Top-down/Bottom-up
  - Sequents/Proof terms
  - ...

Top 100 theorems

HOL Light	86
Mizar	57
Isabelle	51
Coq	49
ProofPower	42
nqthm/ACL2	18
PVS	16
NuPRL/MetaPRL	8

http://www.cs.ru.nl/~freek/100/

The need for interoperabilty

- CompCert project : 50 000 lines (Coq)
- Four-color theorem: 60 000 lines (Coq)
- Jordan curve theorem: 75 000 lines (HOL)
- Odd order theorem: 170 000 lines (Coq)
- L4.verified project: 200 000 lines (Isabelle)

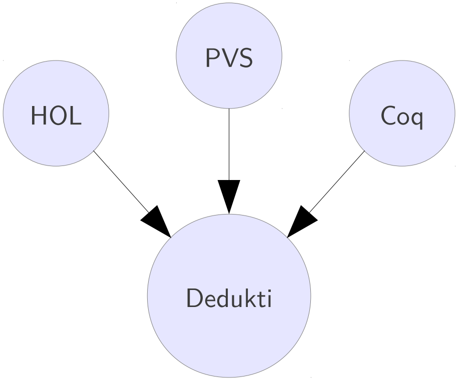
The need for interoperabilty

- CompCert project : 50 000 lines (Coq)
- Four-color theorem: 60 000 lines (Coq)
- Jordan curve theorem: 75 000 lines (HOL)
- Odd order theorem: 170 000 lines (Coq)
- L4.verified project: 200 000 lines (Isabelle)

Can we reuse them?

Logical Framework

Idea: Express all these proofs in a common logical framework.



Dedukti

- Dedukti: means “to deduce” in Esperanto
- Minimal formalism: $\lambda\Pi$-calculus modulo = First-order logic + rewriting

Without rewriting

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{4 = 4}}{4 + 1 = 5}}{4 + 2 = 6}}{4 + 3 = 7}}{4 + 4 = 8}$$

With rewriting

$$\begin{aligned} x + 0 &\longrightarrow x \\ x + (y + 1) &\longrightarrow (x + 1) + y \end{aligned}$$

With rewriting

$$
\begin{aligned}
x + 0 &\longrightarrow x \\
x + (y + 1) &\longrightarrow (x + 1) + y
\end{aligned}
$$

$$
4 + 4 \longrightarrow^{*} 8
$$

With rewriting

$$\begin{array}{rcl} x + 0 & \longrightarrow & x \\ x + (y + 1) & \longrightarrow & (x + 1) + y \end{array}$$

$$4 + 4 \longrightarrow^{*} 8$$

$$\frac{\overline{8 = 8}}{4 + 4 = 8}$$

Encodings

- Adding rewrite rules extends the logic.

Encodings

- Adding rewrite rules extends the logic.
- Can express the proofs of a system P in Dedukti.

$$\Gamma \vdash_P A \quad \Longrightarrow \quad \phi(\Gamma) \vdash_{\lambda\Pi_R} \phi(A)$$

Encodings

- Adding rewrite rules extends the logic.
- Can express the proofs of a system P in Dedukti.

$$
\begin{aligned}
\Gamma \vdash_P A &\implies \phi(\Gamma) \vdash_{\lambda\Pi_R} \phi(A) \\
\Gamma \vdash_P A &\impliedby \phi(\Gamma) \vdash_{\lambda\Pi_R} \phi(A)
\end{aligned}
$$

Encodings

- Adding rewrite rules extends the logic.
- Can express the proofs of a system P in Dedukti.

$$
\begin{array}{lcll}
\Gamma \vdash_P A & \Longrightarrow & \phi(\Gamma) \vdash_{\lambda\Pi_R} \phi(A) & \text{(Completeness)} \\
\Gamma \vdash_P A & \Longleftarrow & \phi(\Gamma) \vdash_{\lambda\Pi_R} \phi(A) & \text{(Soundness)}
\end{array}
$$

Encodings

- Need to find a careful balance between *expressivity* and *consistency*.

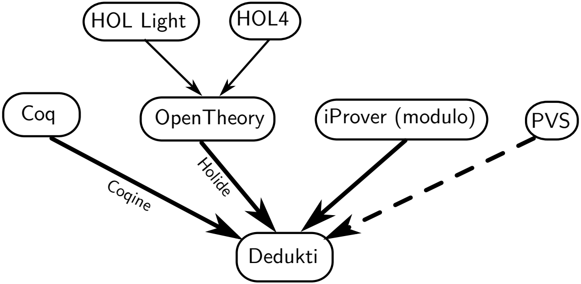
Theorem (Assaf, Cousineau, Dowek)

Any pure type system (PTS) can be encoded in the $\lambda\Pi$*-calculus modulo in a way that is sound and complete.*

Implementations

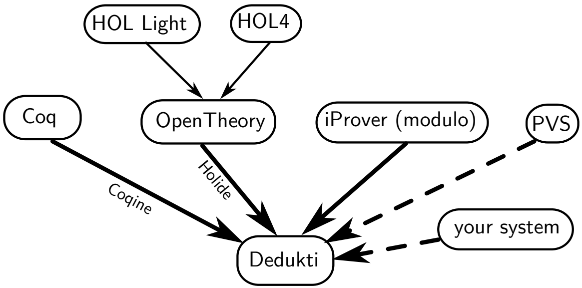
- Holide: HOL in Dedukti (Assaf, Burel)
- Coqine: Coq in Dedukti (Assaf, Boespflug, Burel)
- Focalide: Focalize in Dedukti (Cauderlier, Dubois)

Thank you!



https://www.rocq.inria.fr/deducteam/software.html

Thank you!



https://www.rocq.inria.fr/deducteam/software.html