Interoperability between arithmetic proofs using Dedukti

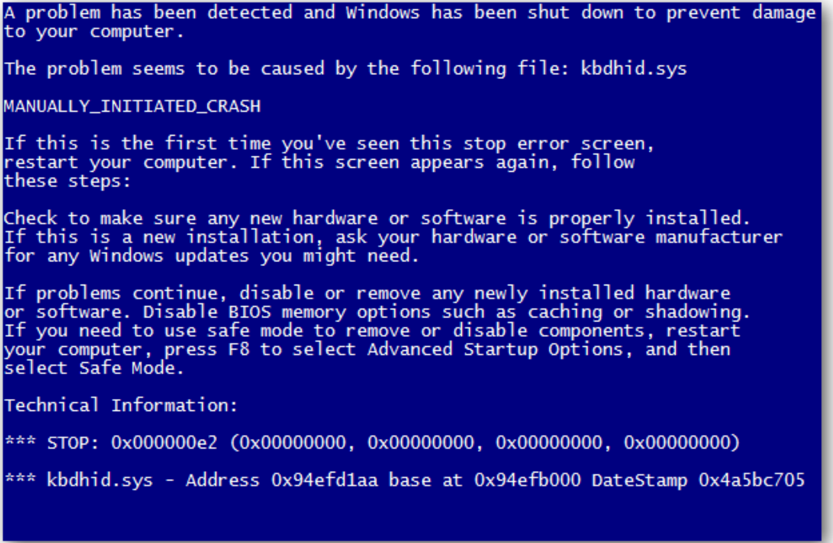
PhD supervised by Gilles Dowek & Stéphane Graham-Lengrand

François Thiré

LSV & LIX

November 22, 2018

Bugs are everywhere!



```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: kbdhid.sys

MANUALLY_INITIATED_CRASH

If this is the first time you've seen this stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use safe mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical Information:

*** STOP: 0x000000e2 (0x00000000, 0x00000000, 0x00000000, 0x00000000)

*** kbdhid.sys - Address 0x94efd1aa base at 0x94efb000 DateStamp 0x4a5bc705
```

Bugs are everywhere!

Would you trust windows to:

- drive your car
- be used in a nuclear power station
- perform surgery on you

Proof Assistants (1/2)

Coq

```
Theorem proj1 : ∀
(A B:Prop), A ∧ B
→ A.
Proof.
  intros A B H ;
destruct H ;
assumption.
Qed.
```

Proof Assistants (1/2)

Coq

Theorem proj1 : $\forall$ (A B:Prop), $A \land B \rightarrow A$.
Proof.
  intros A B H ;
destruct H ;
assumption.
Qed.

$\xrightarrow{\textit{produces}}$

Definition proj1' : $\forall$ A B:Prop, $A \land B \rightarrow A$:=
  fun (A B : Prop) (H : $A \land B$) $\Rightarrow$
    match H with
    | conj $H0$ _ $\Rightarrow$ $H0$
    end.

Proof Assistants (2/2)
HOL

```
  val AND1_THM = save_thm(
"AND1_THM",
let val t12 = mk_conj(t1b, t2b)
    val th2 = RIGHT_BETA(AP_THM (RIGHT_BETA(AP_THM AND_DEF t1b)) t2b)
    val th3 = SPEC t1b (EQ_MP th2 (ASSUME t12))
    val th4 = DISCH t1b (DISCH t2b (ADD_ASSUM t2b (ASSUME t1b)))
in
  GEN t1b (GEN t2b (DISCH t12 (MP th3 th4)))
end);
```

Proof checkers & automatic provers

- Automath
- Metamath
- Mizar
- PVS
- HOL family
- LF
- Coq
- Isabelle
- Agda
- Dedukti
- ProofCert
- CubicalTT

Interoperability problem

- There is a lot a proof checkers & logics
- Different logics might have different expressivity
- Each system has its own library of proofs
- There is not standard

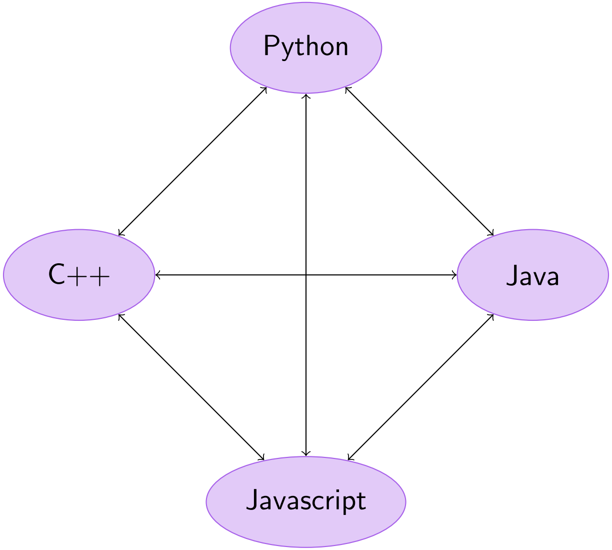
Interoperability problem

- There is a lot a proof checkers & logics
- Different logics might have different expressivity
- Each system has its own library of proofs
- There is not standard

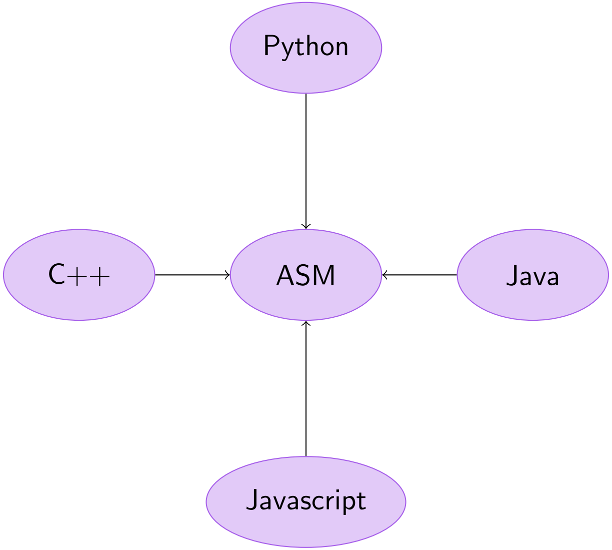
Our goal:

Take a small library (e.g. an arithmetic library) that could be exported to a bunch of proof checkers

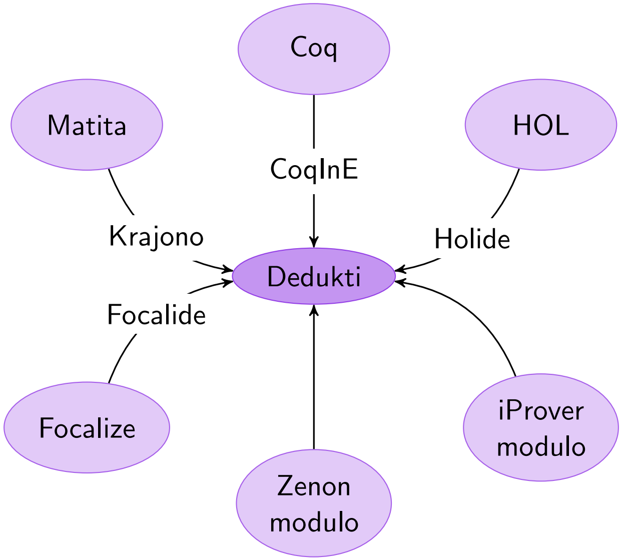
Methodology



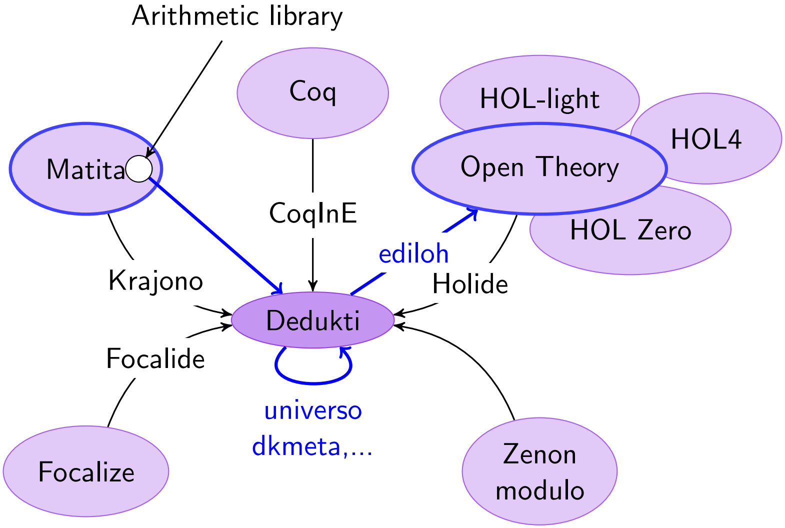
Methodology



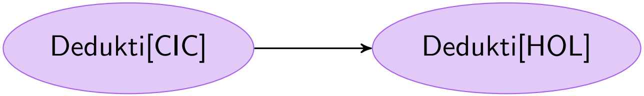
Dedukti


Coq
Matita
HOL
CoqInE
Krajono
Holide
Dedukti
Focalide
Focalize
Zenon modulo
iProver modulo

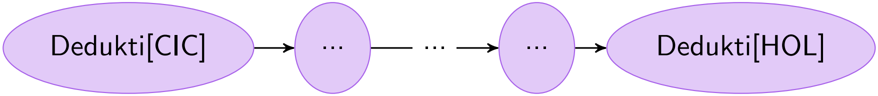
From Matita to HOL



Dedukti[CIC] to Dedukti[HOL]



Dedukti[CIC] to Dedukti[HOL]



Dedukti[CIC] to Dedukti[HOL]

What to do?

- Remove useless stuff (proofs not used)
- Remove universes
- Remove dependent products
- Remove rewrite rules

Dedukti[CIC] to Dedukti[HOL]

What to do?

- Remove useless stuff (proofs not used)
- Remove universes
- Remove dependent products
- Remove rewrite rules

How?

- Dedukti rewrite system
- OCaml

Dedukti[CIC] to Dedukti[HOL]

What to do?

- Remove useless stuff (proofs not used)
- Remove universes
- Remove dependent products
- Remove rewrite rules

How?

- Dedukti rewrite system
- OCaml

Problems:

- All at the same time?
- What is the best order?

Dedukti[CIC] to Dedukti[HOL]

What to do?

- Remove useless stuff (proofs not used)
- Remove universes
- Remove dependent products
- Remove rewrite rules

How?

- Dedukti rewrite system
- OCaml

Problems:

- All at the same time?
- What is the best order?

Remove rewrite rules

```
nat : Type.

0 : nat.
S : nat -> nat.

odd : nat -> Prop.
pi : odd (S 0).

[] one --> S 0.

def pi1 : (odd one) := pi.
```

Remove rewrite rules

```
nat : Type.

0 : nat.
S : nat -> nat.

odd : nat -> Prop.
pi : odd (S 0).

def eq : nat -> nat -> Prop :=
    x:nat =>
    y:nat =>
    forall (P:(nat -> Prop) =>
      impl (P x) (P y)).
```

Remove rewrite rules

```
nat : Type.

0 : nat.
S : nat -> nat.

odd : nat -> Prop.
pi : odd (S 0).

def eq : nat -> nat -> Prop :=
    x:nat =>
    y:nat =>
    forall (P:(nat -> Prop) =>
      impl (P x) (P y)).

eq_one : eq (S 0) one.
```

Remove rewrite rules

```
nat : Type.

0 : nat.
S : nat -> nat.

odd : nat -> Prop.
pi : odd (S 0).


def eq : nat -> nat -> Prop.
[x,y] eq x y -->
      forall (P:(nat -> Prop) =>
        impl (P x) (P y)).

eq_one : eq (S 0) one.
def pi1 : (odd one) :=
    eq_one (ctx => odd ctx) pi.
```

Dedukti[HOL] to OpenTheory (OT)

Should be *easy*, right? But...

Dedukti[HOL] to OpenTheory (OT)

Should be *easy*, right? But...

- Polymorphism in OT is not the handle the same way as in Dedukti[HOL]
- Dedukti is modulo delta (constants unfolding), OT is not
- Dedukti is modulo beta, OT is not!
- Dedukti using De Bruijn indices, OT uses names!

In practice

Dedukti to Dedukti (done)

- 6 months
- 3 tools
- about 1500 lines of OCaml

Dedukti to Dedukti (done)

- 5 months
- 1 tool
- 3000 lines of OCaml

Conclusion & Future Work

- Find (or invent) a better language to write these compilers (a futur work with Prof. Brigitte Pientka)
- OT has some good ideas for interoperability that could be reuse
- Extend the compiler to other systems such as ProofCert, Coq or PVS