Mixing HOL and Coq in Dedukti

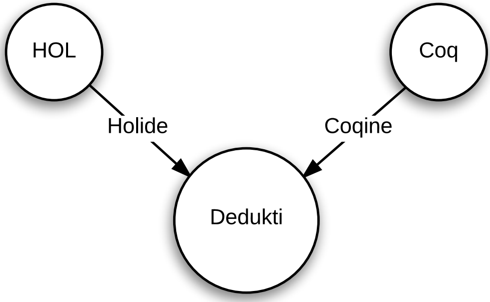
Ali Assaf[1,2] and Raphaël Cauderlier[1,3]

[1]Inria Paris-Rocquencourt
[2]Ecole Polytechnique
[3]CNAM/Cédric

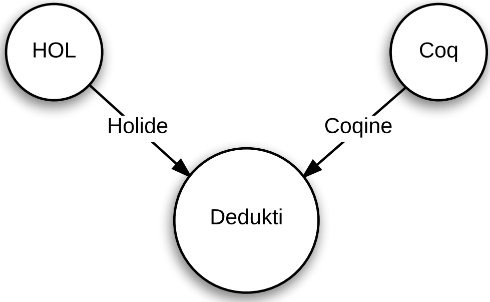
PxTP 2015
Aug 3, 2015

Outline

This talk



This talk



Warning: highly experimental!

Tools

Dedukti:

- Type-checker for the $\lambda\Pi$-*calculus modulo rewriting* ($\lambda\Pi R$)
- Logical framework (see previous talk)

Tools

Dedukti:

- Type-checker for the $\lambda\Pi$-*calculus modulo rewriting* ($\lambda\Pi R$)
- Logical framework (see previous talk)

HOL:

- Family of provers based on *higher-order logic* (HOL)
- **Holide**: translation of HOL to Dedukti through OpenTheory (see previous talk)

Tools

Dedukti:

- Type-checker for the $\lambda\Pi$-*calculus modulo rewriting* ($\lambda\Pi R$)
- Logical framework (see previous talk)

HOL:

- Family of provers based on *higher-order logic* (HOL)
- **Holide**: translation of HOL to Dedukti through OpenTheory (see previous talk)

Coq:

- Prover based on the *calculus of inductive constructions* (CIC)
- **Coqine**: translation of Coq to Dedukti

Coqine

Translation of Coq to Dedukti

- Version 1.0 by Boespflug and Burel (2012)
  - Inductive types ✓
  - Modules ✓
  - No universe hierarchy: **Type** : **Type** X

Coqine

Translation of Coq to Dedukti

- Version 1.0 by Boespflug and Burel (2012)
  - Inductive types ✓
  - Modules ✓
  - No universe hierarchy: **Type** : **Type** X
- Version 2.0 by Assaf (2015)
  - Universe hierarchy: $\textbf{Type}_i : \textbf{Type}_{i+1}$ ✓
  - Universe cumulativity: $\textbf{Type}_i \subseteq \textbf{Type}_{i+1}$ ✓
  - Work in progress

Why use a logical framework?

- Independent proof checking (see previous talk)
- Better understanding of logics
- Interoperability

Outline

Goal

Have:

$$\Gamma \vdash_{HOL} A \implies \Sigma_H, [\![\Gamma]\!]_H \vdash_{\lambda\Pi R} M : [\![A]\!]_H$$

$$\Delta \vdash_{Coq} B \implies \Sigma_C, [\![\Delta]\!]_C \vdash_{\lambda\Pi R} N : [\![B]\!]_C$$

Goal

Have:

$$\Gamma \vdash_{HOL} A \quad \Longrightarrow \quad \Sigma_H, [\![\Gamma]\!]_H \vdash_{\lambda\Pi R} M : [\![A]\!]_H$$

$$\Delta \vdash_{Coq} B \quad \Longrightarrow \quad \Sigma_C, [\![\Delta]\!]_C \vdash_{\lambda\Pi R} N : [\![B]\!]_C$$

Want:

$$\Sigma_{C+H}, [\![\Gamma]\!]_H, [\![\Delta]\!]_C \vdash_{\lambda\Pi R} (M, N) : [\![A]\!]_H \wedge [\![B]\!]_C$$

Challenges

- Propositions might be represented differently
- The logics might be incompatible
- Datatypes might be defined differently

Obstacle I: Type inhabitation

- In HOL, all types are inhabited

$$\forall A.\ \ \mathsf{select}\, A\, (\lambda x\, .\, \top) : A$$

- In Coq, some types are empty

$$\nexists M.\ \ \vdash M : \bot$$

Obstacle I: Type inhabitation

- In HOL, all types are inhabited

$$\forall A.\ \ \mathsf{select}\, A\, (\lambda x\, .\, \top) : A$$

- In Coq, some types are empty

$$\nexists M.\ \vdash M : \bot$$

- Union is inconsistent! X

$$(\vdash_{HOL} \exists x : \alpha . \top)\ \wedge\ (\vdash_{COQ} \neg \forall \alpha . \exists x : \alpha . \top)$$

Type inhabitation

Solution (Keller and Werner 2010): interpret HOL types as *inhabited* Coq types

$$\mathsf{htype} \quad := \quad \Sigma \underbrace{\alpha : \mathsf{ctype}}_{\text{carrier}} . \underbrace{\alpha}_{\text{witness}}$$

Type inhabitation

Solution (Keller and Werner 2010): interpret HOL types as *inhabited* Coq types

$$\mathsf{htype} \quad := \quad \Sigma \underbrace{\alpha : \mathsf{ctype}}_{\mathsf{carrier}} . \underbrace{\alpha}_{\mathsf{witness}}$$

Can be lifted to arrow types:

$$\mathsf{harrow}\, a\, b \quad := \quad (\mathsf{carrow}\,(\mathsf{carrier}\, a)\ (\mathsf{carrier}\, b)\,, \lambda x\,.\,\mathsf{witness}\, b)$$

Type inhabitation

Solution (Keller and Werner 2010): interpret HOL types as *inhabited* Coq types

$$\mathsf{htype} \quad := \quad \Sigma\, \underbrace{\alpha : \mathsf{ctype}}_{\mathsf{carrier}} .\ \underbrace{\alpha}_{\mathsf{witness}}$$

Can be lifted to arrow types:

$$\mathsf{harrow}\, a\, b \quad := \quad (\mathsf{carrow}\,(\mathsf{carrier}\, a)\ (\mathsf{carrier}\, b)\,, \lambda x\,.\,\mathsf{witness}\, b)$$

Consistent union: ✓

$$\mathsf{hterm}\, a \quad := \quad \mathsf{cterm}\,(\mathsf{carrier}\, a)$$

Obstacle II: Bool vs Prop

In HOL:

- Propositions are the terms of type bool
- No difference between propositions and booleans
- Classical system:

$$\forall p.\ (p = \top) \lor (p = \bot)$$

Obstacle II: Bool vs Prop

In HOL:

- Propositions are the terms of type bool
- No difference between propositions and booleans
- Classical system:

$$\forall p.\ (p = \top) \vee (p = \bot)$$

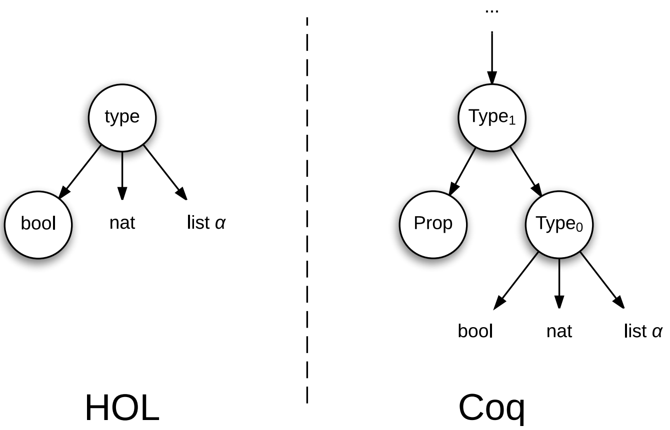
In Coq:

- Propositions are the terms of type **Prop** (which is in $\mathbf{Type}_1$)
- Booleans are the terms of the inductive type bool (which is in $\mathbf{Type}_0$):

```
Inductive bool := true | false.
```

- Intuitionistic system

Bool vs Prop



Bool vs Prop

2 solutions:

- Place HOL types in $\mathbf{Type}_0$ and reflect HOL booleans into Coq propositions:

$$\mathsf{hproof}\, b \quad := \quad \mathsf{cproof}\, (\mathsf{istrue}\, b)$$

Bool vs Prop

2 solutions:

- Place HOL types in $\mathbf{Type}_0$ and reflect HOL booleans into Coq propositions:

$$\mathrm{hproof}\, b \quad := \quad \mathrm{cproof}\,(\mathrm{istrue}\, b)$$

- Place HOL types in $\mathbf{Type}_1$ and identify HOL booleans with Coq propositions
  - Law of excluded middle...
  - ... vs. **Prop** elimination?

Bool vs Prop

2 solutions:

- Place HOL types in $\mathbf{Type}_0$ and reflect HOL booleans into Coq propositions:

$$\text{hproof}\, b \quad := \quad \text{cproof}\, (\text{istrue}\, b)$$

- Place HOL types in $\mathbf{Type}_1$ and identify HOL booleans with Coq propositions
  - Law of excluded middle...
  - ... vs. **Prop** elimination?

- In our work: option 1 ✓

Outline

Case study

In HOL:

- Natural numbers
- Partial order $\leq$ (with proofs of reflexivity, transitivity, etc.)

Case study

In HOL:

- Natural numbers
- Partial order $\leq$ (with proofs of reflexivity, transitivity, etc.)

In Coq:

- Polymorphic lists
- Insertion sort algorithm parametrized by a partial order
- Proof of correctness

```
Theorem sorted_insertion_sort:
  forall l, sorted (insertion_sort l).
Theorem perm_insertion_sort:
  forall l, permutation l (insertion_sort l).
```

In Dedukti

1. Translate the two developments to Dedukti.
2. Link the results together.
3. ???
4. Profit!!!

Linking

Linking consists of writing a file (`interop.dk`):

- Instanciating the Coq development with HOL natural numbers
- Interfacing the proofs of the two systems
- Proving that the theorems needed by the Coq proofs are indeed those given by HOL (e.g. HOL comparison is total w.r.t. Coq)

Linking

Linking consists of writing a file (`interop.dk`):

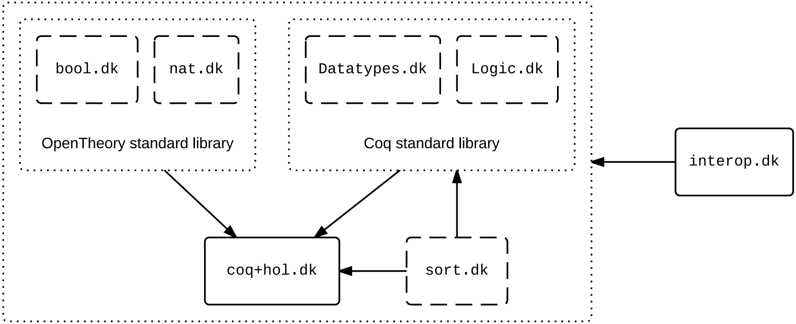
- Instanciating the Coq development with HOL natural numbers
- Interfacing the proofs of the two systems
- Proving that the theorems needed by the Coq proofs are indeed those given by HOL (e.g. HOL comparison is total w.r.t. Coq)

Result:

$$\Pi l : \mathrm{cterm}_1\,(\mathrm{clist}\,\mathrm{hnat})\,.\,\mathrm{cproof}\,(\mathrm{sorted}\,(\mathrm{insertion\_sort}\,\mathrm{compare}\,l))$$

$$\Pi l : \mathrm{cterm}_1\,(\mathrm{clist}\,\mathrm{hnat})\,.\,\mathrm{cproof}\,(\mathrm{permutation}\,l\,(\mathrm{insertion\_sort}\,\mathrm{compare}\,l))$$

Components



bool.dk
nat.dk
OpenTheory standard library
Datatypes.dk
Logic.dk
Coq standard library
interop.dk
coq+hol.dk
sort.dk

Limitations

- A lot of manual work needed for linking
  - Need tools for automation
- Developments largely orthogonal (except for bool).
  - How to mix HOL natural numbers with Coq natural numbers?
- Sorting “algorithm” freezes because HOL is not computational
  - Importance of having computational embeddings

Conclusion

- Using Dedukti as a platform for **interoperability**
- **Case study** of sorting Coq lists of HOL natural numbers
- Lots of future work perspectives

http://dedukti-interop.gforge.inria.fr/

Conclusion

- Using Dedukti as a platform for **interoperability**
- **Case study** of sorting Coq lists of HOL natural numbers
- Lots of future work perspectives

http://dedukti-interop.gforge.inria.fr/

Thank you!

for real this time :-)