Translating HOL to Dedukti

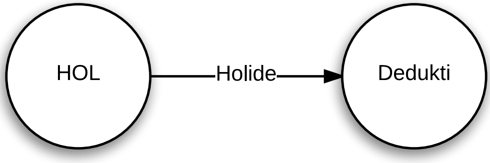
Ali Assaf[1,2] and Guillaume Burel[3]

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

PxTP 2015
Aug 3, 2015

Outline

This talk



Dedukti

- Type-checker for the $\lambda\Pi$-**calculus modulo rewriting**

  dependent types $+$ **rewriting**

- Logical framework
- Universal proof checker:
  - HOL: HOL Light, HOL4, ProofPower
  - CIC: Coq, Matita
  - First-order provers: Zenon (modulo), iProver (modulo)
  - SMT solvers: VeriT

$\lambda\Pi$-calculus modulo rewriting

Syntax

$$\begin{array}{lllll} \text{terms} & M, N, A, B & := & x \mid M\,N \mid \lambda x : A\,.\,M \mid \Pi x : A\,.\,B \mid \textbf{Type} \\ \text{contexts} & \Sigma, \Gamma & := & \varnothing \mid \Gamma, x : A \mid \Gamma, M \longmapsto N \end{array}$$

$\lambda\Pi$-calculus modulo rewriting

Syntax

$$\begin{array}{lllll} \text{terms} & M, N, A, B & \coloneqq & x \mid M\,N \mid \lambda x : A \,.\, M \mid \Pi x : A \,.\, B \mid \textbf{Type} \\ \text{contexts} & \Sigma, \Gamma & \coloneqq & \varnothing \mid \Gamma, x : A \mid \Gamma, \color{red}{M \longmapsto N} \end{array}$$

Main judgement

$$\Gamma \vdash M : A$$

$\lambda\Pi$-calculus modulo rewriting

Syntax

$$\begin{array}{llcl} \text{terms} & M, N, A, B & := & x \mid M\,N \mid \lambda x : A\,.\,M \mid \Pi x : A\,.\,B \mid \textbf{Type} \\ \text{contexts} & \Sigma, \Gamma & := & \varnothing \mid \Gamma, x : A \mid \Gamma, M \longmapsto N \end{array}$$

Main judgement

$$\Gamma \vdash M : A$$

Typing modulo computation

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : \textbf{Type} \qquad A \equiv_{\beta\Gamma} B}{\Gamma \vdash M : B}$$

Curry-Howard correspondence

Curry-Howard correspondence

$$\Gamma \vdash_{\mathcal{L}} A \iff [\![\Gamma]\!] \vdash_{\lambda_{\mathcal{L}}} M : [\![A]\!]$$

Curry-Howard correspondence

Curry-Howard correspondence

$$\Gamma \vdash_{\mathcal{L}} A \iff [\![\Gamma]\!] \vdash_{\lambda_{\mathcal{L}}} M : [\![A]\!]$$

Examples:

Propositional logic	Simply typed λ-calculus
First-order logic	$\lambda\Pi$-calculus
Second-order logic	System F
Classical logic	λ-calculus with call-cc

Logical framework

Logical framework approach

$$\Gamma \vdash_{\mathcal{L}} A \iff \Sigma_{\mathcal{L}}, [\![\Gamma]\!] \vdash_{\mathit{LF}} M : [\![A]\!]$$

Logical framework

Logical framework approach

$$\Gamma \vdash_{\mathcal{L}} A \iff \Sigma_{\mathcal{L}}, [\![\Gamma]\!] \vdash_{LF} M : [\![A]\!]$$

Examples:

- **Twelf**: $\lambda\Pi$-calculus ($\lambda\Pi$, LF, λP)
- **Dedukti**: $\lambda\Pi$-calculus modulo rewriting ($\lambda\Pi R$)

Why use a logical framework?

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

Outline

HOL

- Family of provers based on **simple type theory** (Church 1940):
  - HOL Light, HOL4, Isabelle, ProofPower, Hol Zero
- Large formalizations:
  - **Flyspeck**: proof of Kepler conjecture
  - **seL4**: verified OS kernel
- Similar implementations
  - LCF architecture

LCF architecture

- Programmed in ML
- A theorem is represented as an **abstract datatype** `thm`

LCF architecture

- Programmed in ML
- A theorem is represented as an **abstract datatype** `thm`
  - $\implies$ More safety ✓

LCF architecture

- Programmed in ML
- A theorem is represented as an **abstract datatype** `thm`
  - $\Longrightarrow$ More safety ✓
  - $\Longrightarrow$ Less memory footprint (no proof objects) ✓

LCF architecture

- Programmed in ML
- A theorem is represented as an **abstract datatype** `thm`
  - $\implies$ More safety $\checkmark$
  - $\implies$ Less memory footprint (no proof objects) $\checkmark$
  - $\implies$ Harder to communicate proofs *X*

LCF architecture

- Programmed in ML
- A theorem is represented as an **abstract datatype** `thm`
  - $\Longrightarrow$ More safety ✓
  - $\Longrightarrow$ Less memory footprint (no proof objects) ✓
  - $\Longrightarrow$ Harder to communicate proofs *X*
- Proof retrieval
  - **OpenTheory**
  - Kaliszyk & Krauss
  - (Common HOL)

OpenTheory (Hurd 2011)

- "Listen" to the kernel and record the proof trace
- Advantages of using OpenTheory:
  - Supports multiple provers (HOL Light, HOL4, ProofPower)
  - Well-documented proof format
  - Standard library of theorems

Outline

Embedding HOL types in $\lambda\Pi$

In HOL:

$$\text{types} \quad := \quad \text{bool} \mid A \to B$$

Embedding HOL types in $\lambda\Pi$

In HOL:

$$\text{types} \quad := \quad \text{bool} \mid A \to B$$

In Dedukti:

$$\begin{aligned} \text{htype} &: \mathbf{Type} \\ \text{bool} &: \text{htype} \\ \text{arrow} &: \text{htype} \to \text{htype} \to \text{htype} \end{aligned}$$

Embedding HOL types in $\lambda\Pi$

In HOL:

$$\text{types} \quad := \quad \text{bool} \mid A \to B$$

In Dedukti:

$$\begin{array}{lcl} \text{htype} & : & \textbf{Type} \\ \text{bool} & : & \text{htype} \\ \text{arrow} & : & \text{htype} \to \text{htype} \to \text{htype} \end{array}$$

Translation:

$$\begin{array}{lcl} [\text{bool}] & = & \text{bool} \\ [A \to B] & = & \text{arrow}\,[A]\ [B] \end{array}$$

Embedding HOL terms in $\lambda\Pi$

In HOL:

$$\text{terms} \quad := \quad x \mid \lambda x : A \,.\, M \mid M\,N \mid (=_A)$$

Embedding HOL terms in $\lambda\Pi$

In HOL:

$$\text{terms} \quad := \quad x \mid \lambda x : A \,.\, M \mid M\, N \mid (=_A)$$

In Dedukti:

$$\begin{array}{lcl} \text{hterm} & : & \text{htype} \to \textbf{Type} \\ \text{lam} & : & \Pi a : \text{htype} \,.\, \Pi b : \text{htype} \,.\, (\text{hterm}\ a \to \text{hterm}\ b) \to \text{hterm}\ (\text{arrow}\ a\ b) \\ \text{app} & : & \Pi a : \text{htype} \,.\, \Pi b : \text{htype} \,.\, \text{hterm}\ (\text{arrow}\ a\ b) \to \text{hterm}\ a \to \text{hterm}\ b \\ \text{eq} & : & \Pi a : \text{htype} \,.\, \text{hterm}\ a \to \text{hterm}\ a \to \text{hterm}\ \text{bool} \end{array}$$

Embedding HOL terms in $\lambda\Pi$

In HOL:

$$\text{terms} \quad := \quad x \mid \lambda x : A \,.\, M \mid M\,N \mid (=_A)$$

In Dedukti:

$$\begin{array}{lcl}
\text{hterm} & : & \text{htype} \to \textbf{Type} \\
\text{lam} & : & \Pi a : \text{htype} \,.\, \Pi b : \text{htype} \,.\, (\text{hterm}\ a \to \text{hterm}\ b) \to \text{hterm}\ (\text{arrow}\ a\ b) \\
\text{app} & : & \Pi a : \text{htype} \,.\, \Pi b : \text{htype} \,.\, \text{hterm}\ (\text{arrow}\ a\ b) \to \text{hterm}\ a \to \text{hterm}\ b \\
\text{eq} & : & \Pi a : \text{htype} \,.\, \text{hterm}\ a \to \text{hterm}\ a \to \text{hterm}\ \text{bool}
\end{array}$$

Translation:

$$\begin{array}{lcl}
[x] & = & x \\
[\lambda x : A \,.\, M] & = & \text{lam}\ [A]\ [B]\ (\lambda x \,.\, [M]) \\
[M\,N] & = & \text{app}\ [A]\ [B]\ [M]\ [N] \\
[(=_A)] & = & \text{eq}\ [A]
\end{array}$$

Term translation example

Example

The term

$$(\lambda x : \mathsf{bool}\,.\,x)\ y$$

is translated to

$$\mathsf{app\,bool\,bool}\,(\mathsf{lam\,bool\,bool}\,(\lambda x\,.\,y))\ y$$

Term translation example

Example

The term

$$(\lambda x : \mathsf{bool} \,.\, x)\ y$$

is translated to

$$\mathsf{app}\,\mathsf{bool}\,\mathsf{bool}\,(\mathsf{lam}\,\mathsf{bool}\,\mathsf{bool}\,(\lambda x \,.\, y))\ y$$

which is not equivalent to y!

Embeddings in $\lambda\Pi$

- Representing types as objects allows higher-order reasonning...

$$\Pi a : \mathsf{htype}\,.\,\mathsf{hterm}\, a \to \mathsf{hterm}\, a \to \mathsf{hterm}\,\mathsf{bool}$$

- ... but does not preserve computation:

$$[(\lambda x : \mathsf{bool}\,.\,x)\ y] \not\equiv_{\beta} [y]$$

With rewriting

Add a rewrite rule:

$$\mathsf{hterm}\,(\mathsf{arrow}\;a\;b) \;\longmapsto\; \mathsf{hterm}\;a \to \mathsf{hterm}\;b$$

With rewriting

Add a rewrite rule:

$$\mathsf{hterm}\,(\mathsf{arrow}\; a\; b) \;\equiv\; \mathsf{hterm}\; a \to \mathsf{hterm}\; b$$

With rewriting

Add a rewrite rule:

$$\mathsf{hterm}\,(\mathsf{arrow}\;a\;b) \;\equiv\; \mathsf{hterm}\;a \to \mathsf{hterm}\;b$$

$$\begin{array}{lcl}
[x] & = & x \\
[\lambda x : A\,.\,M] & = & \lambda x : \mathsf{hterm}\,[A]\,.\,[M] \\
[M\;N] & = & [M]\;[N] \\
[(=_A)] & = & \mathsf{eq}\,[A]
\end{array}$$

With rewriting

Example

The term

$$(\lambda x : \mathsf{bool} \,.\, x)\ y$$

is translated to

$$(\lambda x : \mathsf{hterm}\,\mathsf{bool} \,.\, x)\ y$$

With rewriting

Example

The term

$$(\lambda x : \mathsf{bool} \,.\, x)\ y$$

is translated to

$$(\lambda x : \mathsf{hterm}\,\mathsf{bool} \,.\, x)\ y$$

which reduces to y!

With rewriting

Example

The term

$$(\lambda x : \mathsf{bool} \,.\, x)\ y$$

is translated to

$$(\lambda x : \mathsf{hterm}\,\mathsf{bool} \,.\, x)\ y$$

which reduces to y!

- More compact
- Preserves computation

$$\frac{}{\vdash M = M}\ \textsf{Refl} \qquad \frac{}{\{\phi\} \vdash \phi}\ \textsf{Assume} \qquad \frac{}{\vdash (\lambda x : A \,.\, M)\ x = M}\ \textsf{Beta}$$

$$\frac{\Gamma \vdash M = N}{\Gamma \vdash \lambda x : A \,.\, M = \lambda x : A \,.\, N}\ \textsf{AbsThm}$$

$$\frac{\Gamma \vdash F = G \qquad \Delta \vdash M = N}{\Gamma \cup \Delta \vdash F\ M = G\ N}\ \textsf{AppThm}$$

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{(\Gamma - \{\psi\}) \cup (\Delta - \{\phi\}) \vdash \phi = \psi}\ \textsf{DeductAntiSym}$$

$$\frac{\Gamma \vdash \phi = \psi \qquad \Delta \vdash \phi}{\Gamma \cup \Delta \vdash \psi}\ \textsf{EqMp} \qquad \frac{\Gamma \vdash \phi}{\sigma(\Gamma) \vdash \sigma(\phi)}\ \textsf{Subst}$$

Derivation rules in Dedukti

$$
\begin{array}{lcl}
\mathsf{Refl} & : & \Pi\alpha : \mathsf{htype}\,.\,\Pi x : \mathsf{hterm}\,\alpha\,.\,\mathsf{proof}\,(\mathsf{eq}\,\alpha\,x\,x) \\
\mathsf{AbsThm} & : & \Pi\alpha,\beta : \mathsf{htype}\,.\,\Pi f,g : \mathsf{hterm}\,(\mathsf{arrow}\,\alpha\,\beta)\,. \\
 & & (\Pi x : \mathsf{hterm}\,\alpha\,.\,\mathsf{proof}\,(\mathsf{eq}\,\beta\,(f\ x)\ (g\,x))) \to \\
 & & \mathsf{proof}\,(\mathsf{eq}\,(\mathsf{arrow}\,\alpha\,\beta)\ f\ g) \\
\mathsf{AppThm} & : & \Pi\alpha,\beta : \mathsf{htype}\,.\,\Pi f,g : \mathsf{hterm}\,(\mathsf{arrow}\,\alpha\,\beta)\,.\,\Pi x,y : \mathsf{hterm}\,\alpha\,. \\
 & & \mathsf{proof}\,(\mathsf{eq}\,(\mathsf{arrow}\,\alpha\,\beta)\ f\ g) \to \mathsf{proof}\,(\mathsf{eq}\,\alpha\,x\,y) \to \\
 & & \mathsf{proof}\,(\mathsf{eq}\,\beta\,(f\ x)\ (g\ y)) \\
\mathsf{EqMp} & : & \ldots \\
\ldots & &
\end{array}
$$

Soundness and completeness

Theorem

For any Γ, A*,*

$$\Gamma \vdash_{HOL} A \iff \exists M.\ \Sigma_{HOL}, [\![\Gamma]\!] \vdash_{\lambda\Pi R} M : [\![A]\!]\ .$$

where $[\![A]\!] = hterm\,[A]$*.*

Proof.

- $\Longrightarrow$ by Cousineau and Dowek (TLCA 2007),
- $\Longleftarrow$ by Assaf (TLCA 2015).

$\square$

Outline

Holide

- Implemented in OCaml ($\sim$1000 lines)
- Optimizations:
  - Proof and term sharing (similar to Kaliszyk & Krauss 2013)
  - Free variable elimination

Proof size

Example

A proof of $t + t = u + u$:

```
let p = ... (* A very large proof of t = u *) in
appThm (appThm (refl (+)) p p)
```

$$
\dfrac{\dfrac{\dfrac{}{(+) = (+)}\ \text{Refl} \qquad \dfrac{\pi}{t = u}}{(+)\,t = (+)\,u}\ \text{AppThm} \qquad \dfrac{\pi}{t = u}}{(+)\,t\,t = (+)\,u\,u}\ \text{AppThm}
$$

Need proof sharing!

Proof sharing

- Share common subproofs

```
step1 : proof (t = u) := ...
step2 : proof ((+) = (+)) := refl q.
step3 : proof ((+) t = (+) u) :=
  appThm step2 step1.
step4 : proof ((+) t t = (+) u u) :=
  appThm step3 step1.
```

  - Already provided by OpenTheory
  - Going further: our own factorization
- Needs lambda lifting

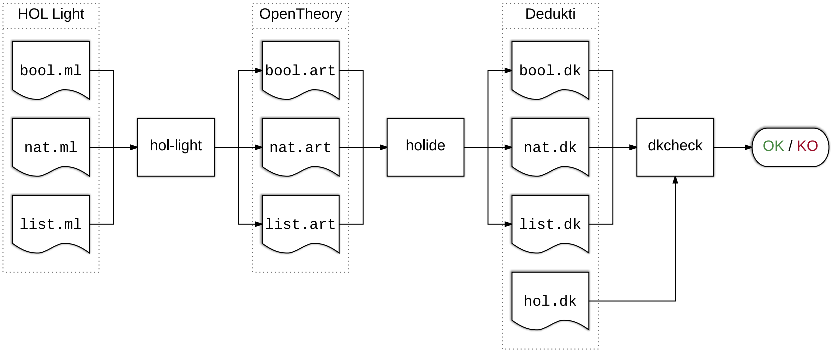
Proof sharing

- Share common subproofs

```
step1 : proof (t = u) := ...
step2 : proof ((+) = (+)) := refl q.
step3 : proof ((+) t = (+) u) :=
  appThm step2 step1.
step4 : proof ((+) t t = (+) u u) :=
  appThm step3 step1.
```

  - Already provided by OpenTheory
  - Going further: our own factorization
- Needs lambda lifting
- Can be extended to terms and types

Translation process



Results

	Size (kB)		Time (s)	
Package	OpenTheory	Dedukti	Translation	Verification
`unit`	5	13	0.2	0
`function`	16	53	0.3	0.2
`pair`	38	121	0.8	0.5
`bool`	49	154	0.9	0.5
`sum`	84	296	2.1	1.1
`option`	93	320	2.2	1.2
`relation`	161	620	4.6	2.8
`list`	239	827	5.7	3.2
`real`	286	945	6.5	3.1
`natural`	343	1065	6.8	3.2
`set`	389	1462	10.2	5.8
Total	1702	4877	40.3	21.6

Conclusion

- **Holide**: scalable translation of **HOL to Dedukti**
- Using **OpenTheory as frontend**
- Translation of **standard library**

https://www.rocq.inria.fr/deducteam/Holide/

Conclusion

- **Holide**: scalable translation of **HOL to Dedukti**
- Using **OpenTheory as frontend**
- Translation of **standard library**

https://www.rocq.inria.fr/deducteam/Holide/

Thank you!

to be continued ;-)

Computation embeddings

- Compress proofs of conversion:

$$\dfrac{\dfrac{}{\vdash f = f}\,\text{Refl} \qquad \dfrac{\dfrac{}{\vdash g = g}\,\text{Refl} \qquad \dfrac{}{\vdash (\lambda x : A\,.\,x)\ x = x}\,\text{Beta}}{\vdash g\,((\lambda x : A\,.\,x)\ x) = g\,x}\,\text{AppThm}}{\vdash f\,(g\,((\lambda x : A\,.\,x)\ x)) = f\,(g\,x)}\,\text{AppThm}$$

  can be translated simply as Refl $B(f\,(g\,x))$.
- Give computational meaning to HOL:
  - Intuitionistic version ($\Rightarrow$ and $\forall$ instead of $=$)
  - Better interoperability with Coq (see next talk)

Related work

	Target	Translation tool	Scalable	Type theory
Appel	Twelf			✓
Naumov et al. (2001)	NuPRL	✓		✓
Schürmann & Stehr (2006)	NuPRL	✓		✓
Obua & Skalberg (2006)	Isabelle/HOL	✓		
Rabe (2010)	LF			✓
Keller & Werner (2010)	Coq	✓		✓
Kaliszyk & Krauss (2013)	Isabelle/HOL	✓	✓	
Assaf & Burel (2015)	Dedukti	✓	✓	✓