avoiding state with infinite contexts

Freek Wiedijk
Radboud University Nijmegen
The Netherlands

10th Asian Logic Conference
special session: model theory and proof theory
Kobe, Japan

2008 09 05, 15:40
proof checking of mathematics

fully verified for correctness by a computer:

- Gödel’s first incompleteness theorem
- Jordan curve theorem
  HOL Light (2005), Mizar (2005)
- prime number theorem
  Isabelle (2004), HOL Light (2008)
- four color theorem
  Coq (2004)
the de Bruijn criterion

all software has bugs . . .
why trust a proof checker?

N.G. de Bruijn, 1968:

[ . . . ] This is one of the reasons for keeping AUTOMATH as primitive as possible. [ . . . ]

two approaches:

• small independent checker(s)
e.g.: Ivy system for Otter/Prover9

• small proof checking kernel inside the system
LCF architecture

Robin Milner, 1972
## systems and kernels

### source sizes in $10^3$ lines of code

<table>
<thead>
<tr>
<th>System</th>
<th>Language</th>
<th>Kernel Size</th>
<th>System Size</th>
<th>Theorems</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOL Light</td>
<td>ocaml</td>
<td>0.7</td>
<td>30</td>
<td>69</td>
</tr>
<tr>
<td>Isabelle</td>
<td>sml</td>
<td>5</td>
<td>160</td>
<td>40</td>
</tr>
<tr>
<td>Twelf</td>
<td>sml</td>
<td>6</td>
<td>70</td>
<td>—</td>
</tr>
<tr>
<td>ProofPower</td>
<td>sml</td>
<td>7</td>
<td>90</td>
<td>42</td>
</tr>
<tr>
<td>Coq</td>
<td>ocaml</td>
<td>14</td>
<td>180</td>
<td>39</td>
</tr>
<tr>
<td>Mizar</td>
<td>pascal</td>
<td></td>
<td>80</td>
<td>45</td>
</tr>
<tr>
<td>ACL2</td>
<td>lisp</td>
<td></td>
<td>170</td>
<td>12</td>
</tr>
<tr>
<td>PVS</td>
<td>lisp</td>
<td></td>
<td>280</td>
<td>15</td>
</tr>
</tbody>
</table>
proving the kernel correct

- **Coq in Coq**
  Bruno Barras, 1997–1999
  executable Coq model of the Coq logic
  two different systems (the real system versus the extracted code)
  extracted code not yet used for serious proof development

- **HOL in HOL**
  John Harrison, 2006
  HOL model of the actual HOL Light kernel source
  not yet the full code (no type polymorphism, no definitions)
  no systematic relation between HOL model and executable code
kernels and state

to make proving a kernel feasible:

- code should be as ‘mathematical’ as possible

- for current technology:
  kernel should be programmed in a purely functional language
  Lisp, ML, Haskell, Coq

current practice:

- kernels always have a state:
  definitions from the formalization that already have been processed

- corresponds to a context in the formal treatment of the logic

\[ \Gamma \vdash M : A \]
undo for HOL

abstract datatypes of the HOL Light kernel

type hol_type = private
| Tyvar of string
| Tyapp of string * hol_type list

type term = private
| Var of string * hol_type
| Const of string * hol_type
| Comb of term * term
| Abs of term * term

type thm = private
| Sequent of term list * term

# 'pi';;
val it : term = 'pi'
#
the problem with undoing definitions

_hypothetical HOL Light session:_

```plaintext
# let X0 = new_definition 'X = 0';;
val ( X0 ) : thm = |- X = 0
# undo_definition "X";;
val it : unit = ()
# let X1 = new_definition 'X = 1';;
val ( X1 ) : thm = |- X = 1
# TRANS (SYM X0) X1;;
val it : thm = |- 0 = 1
#
```

undoing definitions can change the meaning of existing thms inconsistent!

⇒ _HOL Light does not support ‘undo’_
putting the definitions in the names

- **current HOL Light**
  names of constants: pair of a string and a type

- **stateless HOL Light**
  names of constants: pair of the traditional name and the definition

comparing equal definitions by pointer comparison is cheap
nested kernels

current HOL Light

stateless HOL Light

about 10% slower
**type theory without explicit contexts**

**first order logic and contexts**

<table>
<thead>
<tr>
<th>First order logic</th>
<th>Type theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \vdash P(x)$</td>
<td>$H : A, x : D \vdash M_1 : P(x)$</td>
</tr>
<tr>
<td>$\frac{\forall I}{A \vdash \forall x P(x)}$</td>
<td>$\frac{\lambda}{H : A \vdash M_2 : \Pi x : D. P(x)}$</td>
</tr>
<tr>
<td>$\frac{\rightarrow I}{\vdash A \rightarrow \forall x P(x)}$</td>
<td>$\frac{\lambda}{\vdash M_3 : A \rightarrow \Pi x : D. P(x)}$</td>
</tr>
</tbody>
</table>

'sea' of free variables

<table>
<thead>
<tr>
<th>First order logic</th>
<th>Type theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \vdash {x} P(x)$</td>
<td>$H : A \vdash {x} M_1 : P(x)$</td>
</tr>
<tr>
<td>$\frac{\forall I}{A \vdash {} \forall x P(x)}$</td>
<td>$\frac{\lambda}{H : A \vdash {} M_2 : \Pi x : D. P(x)}$</td>
</tr>
<tr>
<td>$\frac{\rightarrow I}{\vdash {} A \rightarrow \forall x P(x)}$</td>
<td>$\frac{\lambda}{\vdash {} M_3 : A \rightarrow \Pi x : D. P(x)}$</td>
</tr>
</tbody>
</table>

\[
\vdash \{\} (\forall x P(x)) \rightarrow (\exists x P(x))
\]
merging all contexts into an infinite context

category of contexts for a given type theory

• objects:

\[ \Gamma \]

\( \Gamma \) is a finite or countably infinite context

• morphisms:

\[ \Gamma \xrightarrow{f} \Gamma' \]

\( f \) is an injection mapping the variables from \( \Gamma \) to variables from \( \Gamma' \)

this category has pushouts:

every two contexts can be combined into a bigger context

direct limit of all contexts:

\[ \Gamma_\infty \]
the system $\Gamma_\infty$

$\Gamma_\infty$: system equivalent to the PTS rules but without explicit contexts (we reuse the name of the infinite context for the system)

PTS = Pure Type System

we write

$$ M : A $$

to morally mean

$$ \Gamma_\infty \vdash M : A $$

$\Gamma_\infty$ preterms:

$$ A ::= s \mid {x_i}^A \mid x_i \mid AA \mid \lambda x_i : A.A \mid \Pi x_i : A.A $$

two kind of variables: free variables $x_i^A$ and bound variables $x_i$

superscript $A$ of $x_i^A$ may be any preterm
two of the six $\Gamma_\infty$ rules

**PTS rules**

- $\Gamma \vdash A : s$
- $\Gamma, x_i : A \vdash x_i : A$

- $\Gamma \vdash A : s_1 \quad \Gamma, x_i : A \vdash B : s_2$
- $\Gamma \vdash \Pi x_i : A.B : s_3$

**equivalent $\Gamma_\infty$ rules**

- $A : s$
- $x_i^A : A$

- $A : s_1 \quad B : s_2$
- $\Pi x_i : A.B[x_j^A := x_i] : s_3$

binding a variable in $\Gamma_\infty$: replace a free variable by a bound variable
correspondence theorems

\[
\text{derivable PTS judgment} \quad \leftrightarrow \quad \text{derivable } \Gamma_\infty \text{ judgment}
\]

from left to right:

\textbf{alpha convert} the judgement to separate free from bound variables
then: remove the context

from right to left:

\textbf{topological sort} of the free variables in the } \Gamma_\infty \text{ judgement
then: put them in that order in the context
implementing $\Gamma_\infty$ for LF

the datatypes of the kernel

type preterm =
  | Star
  | Ref of int
  | Var of string * preterm
  | Const of string * preterm * preterm list
  | App of preterm * preterm
  | Bind of int * preterm * preterm

  0 = \lambda, 1 = \Pi

axioms used

type term = private
  | Box
  | In of preterm * term

type red = private
  | Red of preterm * preterm
difference of this approach with other kernels

most purely functional kernels:

\[
\begin{align*}
\text{App} : & \text{preterm} \times \text{preterm} \rightarrow \text{preterm} \\
\text{typecheck} : & \text{state} \rightarrow (\text{preterm} \rightarrow \text{term}) \\
\text{extend\_state} : & \text{string} \times \text{preterm} \rightarrow \text{state} \rightarrow \text{state}
\end{align*}
\]

mutually inconsistent terms (from mutually inconsistent states) possible

internally inconsistent state not possible

our approach:

*function application in LCF style*

\[
\begin{align*}
\text{app} : & \text{term} \times \text{term} \rightarrow \text{term}
\end{align*}
\]

mutually inconsistent terms *not* possible
### comparing kernel sizes

<table>
<thead>
<tr>
<th></th>
<th>current HOL Light</th>
<th></th>
<th>all lines</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td></td>
<td>fusion.ml</td>
<td>669</td>
<td>394</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>stateless HOL Light</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td></td>
<td>core.ml</td>
<td>404</td>
<td>330</td>
</tr>
<tr>
<td>state</td>
<td></td>
<td>state.ml</td>
<td>95</td>
<td>71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\Gamma_\infty$ for LF</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td></td>
<td></td>
<td>214</td>
<td>166</td>
</tr>
<tr>
<td>convertibility</td>
<td></td>
<td></td>
<td>64</td>
<td>49</td>
</tr>
<tr>
<td>typechecker</td>
<td></td>
<td></td>
<td>29</td>
<td>25</td>
</tr>
</tbody>
</table>
outlook

future work

but does this scale?

experiment:

HOL Light

kernel

LF context for HOL

kernel

\( \Gamma_\infty \) for LF

how much slower than current HOL Light? 100 times? \( \infty \) times?