Towards HOL2000?

John Harrison

University of Cambridge

This talk will be a quick summary of some half-baked ideas that I've been thinking about and/or implementing recently.

- HOL2000
- An even lighter HOL Light
- Higher order set theory
- An approach to partial functions

HOL2000

How can we keep HOL's strengths while addressing some of its weaknesses?

- Start with a simple axiomatization of higher order set theory.
- Adhere to the LCF approach as far as possible.
- Use readable declarative proof scripts as the standard, with code-writing only for difficult cases.
- Address important issues in real mathematics, e.g. partial functions and 'subtypes'.
- Provide some of the convenience of simple type theory as an interface to set theory, and try to avoid a load of extra inferences.

I'm now experimenting with these ideas in theory and practice. This talk will be a quick discussion of some of the main points.

An even lighter HOL Light

Here are the results of some usage profiles for HOL Light's primitive rules.

Inference rule	Building	Tang
REFL	236,436	91,249,996
TRANS	111,325	59,700,503
EQ_MP	129,143	30,450,064
INST	124,337	30,456,796
MK_COMB	69,225	29,039,492
MP	87,170	761,637
INST_TYPE	38,867	277,365
BETA_CONV	28,428	225,223
DISCH	29,186	103,624
ASSUME	15,500	70,184
SYM	7,937	58,052
ABS	15,877	52,014
IMP_ANTISYM_RULE	923	3,730

Starting from equality

Generally speaking, it's equality reasoning that dominates. Also, about 80% of BETA_CONVs are trivial. So we define all logical constants in terms of equality.

$$\begin{aligned}
\top &= (\lambda x.\ x) = (\lambda x.\ x) \\
p \wedge q &= (\lambda f.\ f\ p\ q) = (\lambda f.\ f\ \top\ \top) \\
p \Rightarrow q &= p \wedge q = p \\
\forall\, P &= P = \lambda x.\ \top \\
\exists\, P &= \forall q.\ (\forall x.\ P\ x \Rightarrow q) \Rightarrow q \\
p \vee q &= \forall r.\ (p \Rightarrow r) \wedge (q \Rightarrow r) \Rightarrow r \\
\bot &= \forall (\lambda p.\ p) \\
\neg p &= p \Rightarrow \bot
\end{aligned}$$

Standard equality rules allow one to deduce from these all the usual intuitionistic rules of deduction.

New inference rules

We throw away: MP, DISCH and SYM.

We replace IMP_ANTISYM_RULE by the following slightly different one:

$$\frac{\Gamma, p \vdash q \qquad \Delta, q \vdash p}{\Gamma \cup \Delta \vdash p = q}$$

For efficiency in derived rules based on proforma theorems, it's convenient to make INST and INST_TYPE instantiate in assumptions, though this is not essential.

BETA_CONV now only works in the special case $(\lambda x.\, t[x])x = t[x]$, with the other cases derived via a separate INST.

Note: it is possible to derive TRANS reasonably easily, and INST rather less easily. But for efficiency reasons, we keep them as primitives.

Type theory vs set theory

Simple type theory is rather restrictive for many parts of mathematics. Richer type theories tend to be too complicated or poorly understood.

Types are useful: they organize work and avoid many explicit inferences. But we believe it's better to regard them as a front end to set theory. This is the approach of Mizar, at least in principle.

We can still run HOL-style type checkers over terms on input and get the convenience of simple type theory in most cases (anyway this is purely an interface issue).

It's mainly a matter of convenience which particular set theory to use: ZF, NBG, NFU ...

We suggest that a higher order axiomatization of ZF (Zermelo-Carnap-Gordon set theory) is a reasonable choice. Why?

Higher order set theory

Zermelo's original axioms were second order, so in a sense higher order set theory is a good formalization of what Zermelo had in mind.

Higher order logic, especially in a slimmed-down form as above, is in many ways simpler than first order logic. The ZF axioms are certainly much more directly expressible.

We can deal with definitions of sets (real), set operations ($\cup$), boolean operations ($\wedge$) etc. in a uniform way using object-level definitions; no separate notion of definitional/meta equality.

We can express many ideas involving classes in a very direct way, e.g. the inductive definition of the class of Conway numbers, or the use of Mod in first order model theory. Perhaps it would also be useful to formalize aspects of categorical reasoning?

Other ideas

If one wants to be more 'standard', one could follow Corella in restricting the ZF axioms to first order schemas. According to his thesis, this gives a conservative extension of first order ZF.

If one wants to be *less* standard, one could replace the Axiom of Foundation by a higher order Axiom of Restriction, i.e. say that the universe of sets is the smallest class closed under the standard generative principles.

This proves the Axiom of Foundation by induction — effectively the same as the usual proof of its relative consistency.

I think it also proves there aren't any inaccessible cardinals. In some ways it gives a much more precise picture of what the universe of sets is like.

Doing real mathematics

It's one thing to do elementary proofs in set theory, but much more important to make sure we have a satisfactory foundation for higher-level mathematical reasoning.

First, we can avoid much of the ugliness of subtyping, coercions, overloading etc. by actually making

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$$

Even if we build up the number systems step-by-step, we can easily embed the previous number system at each stage to make sure this holds.

Generally, mathematics is known to work well in set theory. The problem is partial functions. Here is an approach which seems to be reasonable.

Partial functions

For the purposes of embedding notions from simple type theory, it's convenient to have a strict notion of the domain and codomain of a function.

So rather than just the graph, we might represent $f : A \to B$ by the ordered pair $(graph(f), B)$. Let us now define the application operation:

$$f'(x) = \begin{cases} \varepsilon y.\ (x, y) \in graph(f) & \text{if } x \in dom(f) \\ cod(f) & \text{if } x \notin dom(f) \end{cases}$$

For example, we have $0^{-1} = \mathbb{R}$. The advantage of this approach is that we effectively have an undefined value $\perp$ for each function, so we get much of the flexibility of a logic of partial functions, without the complexity.

We could use a fixed element like $\emptyset$ for the undefined value, but it's always possible that it would be a permissible return from some function or other.

The point

Suppose, as usual, that we are in a 'simply typed' part of set theory. Then all the undefined elements 'line up' and we can read equality as 'either both sides are undefined or both are defined and equal'.

Moreover, the use of a fixed $\bot$ value for any undefined value is enough to ensure that many basic theorems can be extended automatically by the rewriting apparatus to the whole domain of sets, e.g.

$$\forall x \in \mathbb{C}, y \in \mathbb{C}.\ x + y = y + x$$

In cases where this is not true, e.g. $\forall x \in \mathbb{C}.\ x + 0 = x$, we can still have unconditional rewrites in context, e.g. $(x + 0) + y = x + y$.

In general, we can often "infer types" top-down using congruence rules in the rewriter: if it's outside the domain, we don't care anyway.