HOL Constant Definition Done Right

Rob Arthan

Lemma 1 Ltd. / Queen Mary University of London, UK

ITP, Vienna, Austria

15 July 2014

In the beginning

- HOL (Mike Gordon c. 1984):
  - simply typed λ-calculus:

$$(\lambda f : \mathbb{N} \to \mathbb{N} \cdot \lambda x : \mathbb{N} \cdot f\ x) : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N}$$

  - + minimalist polymorphism:
    - free type variables:

$$(\lambda f \cdot \lambda x \cdot f\ x) : (\alpha \to \beta) \to \alpha \to \beta$$

    - polymorphic constants:

$$\mathbf{first}(\mathbf{head}[1; 2; 3], "a") = \mathbf{first}(\mathbf{head}[(1, 2)])$$

  - + principle for defining new types
  - + principle for defining new constants.
- HOL is a great compromise between simplicity and expressiveness.
- (At least) 6 current implementations and many users 30 years on.
- This talk describes an improved principle for defining new constants.

`new_definition`

- Input to `new_definition` is an equation:

$$c\, v_1 \ \ldots \ v_n = t$$

- Result is a new constant **c** with defining property:

$$\vdash \forall v_1 \ \ldots \ v_n \cdot \mathbf{c}\, v_1 \ \ldots v_n = t$$

- Side-conditions:
  1. c and v_i distinct variables
  2. $\mathbf{frees}(t) \subseteq \{v_1, \ldots, v_n\}$
  3. $\mathbf{tyvars}(t) \subseteq \mathbf{tyvars}(c)$

  Condition 3 fixes an inconsistency found by Roger Jones c. 1988
- Means for specifying constant names like **c** immaterial in this talk.

A little later: a feature request

- Roger Jones (c. 1988) made an observation:

  `new_definition` **doesn't support implicit definitions.**

- You can't give an implicit definition of **min**:

$$\mathbf{min}(x, y) \in \{x, y\} \wedge \mathbf{min}(x, y) \leq x \wedge \mathbf{min}(x, y) \leq y$$

- or define **Pre** in terms of **Suc**:

$$\mathbf{Pre}(\mathbf{Suc}(n)) = n$$

- or give an approximate specification of a number:

$$\mathbf{c_1} \leq 10.$$

Work-arounds

- Can work around using specific circumlocutions:

$$\mathbf{min}(x, y) = \mathbf{if}\ x \leq y\ \mathbf{then}\ x\ \mathbf{else}\ y$$

  and then prove the desired defining property as a theorem.
- General purpose "work-around" with the Hilbert choice operator:
  - E.g., to define $\mathbf{c_1}$ such that $\mathbf{c_1} \leq 10$, use `new_definition` to define:

$$\mathbf{c_1} = (\varepsilon x \cdot x \leq 10)$$

  - This is a hack: it introduces unintended identities.
  - E.g., the naive way of now defining $\mathbf{c_2}$ such that $\mathbf{c_2} \leq 10$ leads to:

$$\mathbf{c_1} = (\varepsilon x \cdot x \leq 10) = \mathbf{c_2}$$

  - More devious hacks mitigate the problem but don't eliminate it.

The feature request implemented: `new_specification`

- Roger's observation was addressed by adding a new definitional principle called `new_specification`.
- `new_specification` takes as input a theorem of the form:

$$\vdash \exists v_1 \ldots v_n \cdot p$$

- Results in new constants $\mathbf{c}_1, \ldots, \mathbf{c}_n$ with defining property:

$$\vdash p[\mathbf{c}_1/v_1, \ldots, \mathbf{c}_n/v_n]$$

- Side conditions:
  1. $\textbf{frees}(p) \subseteq \{v_1 \ldots v_n\}$
  2. $\textbf{tyvars}(p) \subseteq \textbf{tyvars}(v_i), i = 1 \ldots n$.
- E.g.,
  - You prove: $\exists v_1\, v_2 \cdot v_1 \leq 10 \wedge v_2 \leq 10$
  - and you get $\mathbf{c}_1$ and $\mathbf{c}_2$ such that $\mathbf{c}_1 \leq 10 \wedge \mathbf{c}_2 \leq 10$.
  - And that is *all* you know about $\mathbf{c}_1$ and $\mathbf{c}_2$.

Further Observations

- `new_specification` provides the abstraction Roger wanted.
- I observed (c. 1992) that it is annoying that:
  - `new_specification` supersedes `new_definition`, but
  - `new_definition` is required for bootstrapping, to define $\exists$.
- John Harrison observed (*HOL Done Right*, 1995) that the polymorphic typed λ-calculus is extremely expressive:
  - The HOL logic can be defined using equality alone.
  - **T**, **F**, $\neg$, $\wedge$, $\vee$, $\exists$, $\forall$ are all definable.
  - Full strength of HOL may then be obtained from three axioms.
  - HOL Light follows this approach (as does OpenTheory).
- `new_specification` was replaced by a form of the ε hack in HOL Light c. 2006, to simplify work on self-verification.
- I and others (c. 1992 – 2014) observed that the constraint on the use of type variables is rather restrictive for some purposes.

A Proposed Enhancement

- In 2012, I proposed `gen_new_specification`.
- Takes as input a theorem of the form

$$v_1 = t_1, \ldots, v_n = t_n \vdash p$$

- Results in new constants $\mathbf{c}_1, \ldots, \mathbf{c}_n$ with defining property:

$$\vdash p[\mathbf{c}_1/v_1, \ldots, \mathbf{c}_n/v_n]$$

- Subject to the following restrictions:
  1. the v_i must be pairwise distinct variables;
  2. $\mathbf{frees}(t_i) = \emptyset$
  3. $\mathbf{tyvars}(t_i) \subseteq \mathbf{tyvars}(v_i)$.
  4. $\mathbf{frees}(p) \subseteq \{v_1, \ldots v_n\}$;
- There is no restriction on the type variables appearing in p.

Example

- For example, it is easy to prove:

$$n = 0, f = \lambda y \cdot 1 \vdash \forall x \cdot \neg f\, x = n$$

- This meets the requirements of `gen_new_specification`
- Hence can define constants **f** and **n** such that:

$$\forall x \cdot \neg \mathbf{f}\, x = \mathbf{n}.$$

- Note **f** has type $\alpha \to \mathbb{N}$ and **n** has type $\mathbb{N}$:
  - This would be impossible with `new_specification`.
  - `new_specification` always gives $\mathbf{tyvars}(\mathbf{c_i}) = \mathbf{tyvars}(\mathbf{c_j})$.

Soundness

Claim

`gen_new_specification` *is conservative and hence sound.*

- The informal proof is really quite simple (simpler than for `new_specification`):
  - It is easy to derive the theorem $\vdash p[t_1/v_1, \ldots, t_n/v_n]$ from the theorem that is input to `new_specification`.
  - Hence replacing each instance of a $\mathbf{c}_i$ with the corresponding instance of $\mathbf{t}_i$ will transform a proof whose conclusion doesn't involve the $\mathbf{c}_i$ into a proof that doesn't involve the $\mathbf{c}_i$ at all.
- As reported in Ramana's talk yesterday, Ramana Kumar, Magnus Myreen and Scott Owens have now formalised this proof (and much, much more) in HOL4.

Backwards Compatibility

Claim

`gen_new_specification` *subsumes* `new_definition`.

- The proof is easy:
  - to simulate `new_definition` on input:

$$c\ v_1\ \ldots\ v_n = t$$

  - apply `gen_new_specification` to the easily proved theorem:

$$c = \lambda v_1\ \ldots\ v_n \cdot t \vdash \forall v_1\ \ldots\ v_n \cdot c\ v_1\ \ldots v_n = t$$

Backwards Compatibility (2)

Claim

`gen_new_specification` *subsumes* `new_specification`.

- The proof requires a little boot-strapping:
  - For the proof in the special case of `new_specification` on input

$$\vdash \exists c \cdot\ p$$

  - apply `gen_new_specification` to the following (easily derived from the above input theorem):

$$c = \varepsilon v \cdot\ p \vdash p.$$

  - Use this to define the constructor and destructors for binary products.
  - Once you have these, the precise behaviour of `new_specification` in general is easily simulated.
- See paper for details.

Assessment

- The proposal solves my concern about bootstrapping:
  - `gen_new_specification` subsumes both `new_definition` and `new_specification`.
- The proposal satisfies John Harrison's criterion:
  - `gen_new_specification` involves no constants other than equality.
- The proposal has now been proved sound:
  - No further need for the ε hack in HOL Light.
- The proposal is much more liberal about type variables:
  - We have seen a simple, but not useless example.
  - See the paper for more significant examples.

Current Status

- HOL implementors were awaiting a formalised correctness proof before adopting the proposal.
- Now thanks to the hard work of Ramana Kumar et al., we have a correctness proof in HOL4.
- Ramana has a branch of HOL4 including `gen_new_specification`
- `gen_new_specification` is in ProofPower working snapshot 3.1w1 and later.
  - Includes a version of `new_specification` implementing the subsumption proof sketched above.
- ProofPower and HOL4 both implement `gen_new_specification` as a replacement for `new_definition`:
  - Keep the old `new_specification` as a built-in for pragmatic reasons.
- Joe Hurd has included `gen_new_specification` in draft version 6 of the OpenTheory article file format.