The HOL Light formalization of Euclidean space

John Harrison

Intel Corporation

JMM special session on Formal Mathematics for Mathematicians

January 8th, 2011 (09:30–10:00)

Summary

- History of this formalization
- Encoding trick for $\mathbb{R}^n$
- Vectors, matrices and linear algebra
- Topology, convexity and polyhedra
- Analysis, integration and measure
- Complex analysis

History of this formalization

This formalization began following a discussion at NYU in February 2004, in an attempt to answer the question

How to formalize Euclidean space in a natural and streamlined way in HOL Light, with the goal of supporting the Flyspeck project?

Although Flyspeck is mainly concerned with $\mathbb{R}^3$, we try to work in the more general setting of $\mathbb{R}^N$ where possible.

The general theory has also been applied to the special case of $\mathbb{R}^2$ and used in a further development of complex analysis.

Authorship and acknowledgements

As well as the present speaker, others have contributed substantially:

- Tom Hales and Flyspeck group: the further development and application to the Flyspeck project, as well as many lemmas here and much of the motivation.
- Lars Schewe: several results on affine dependence and convex sets, Radon's theorem.
- Marco Maggesi, Graziano Gentili and Gianni Ciolli: further development of complex analysis including higher derivatives and the first and second Cartan theorems.
- Valentina Bruno: Cauchy's inequality, analytic continuation, maximum modulus principle, open mapping theorem, Schwarz's lemma.

The problem with simple type theory

Can work over abstract spaces but then parametrization is heavy.

We would like each $\mathbb{R}^n$ to be a *type* in **simple type theory**.

For any *fixed* n we can use n-tuples, e.g. $\mathbb{R} \times \mathbb{R}$ for $\mathbb{R}^2$.

For general n, using a *set/predicate* is OK, but then the type system isn't helping us much.

Yet we have no dependent types so we can't have a *type* $\mathbb{R}^n$ depend on a *term* n.

A parochial problem

Defining spaces such as $\mathbb{R}^n$ presents no problem for many foundational systems.

- Untyped systems such as set theory (ACL2, B prover, Mizar, . . .)
- Richer dependent type theories (Coq, MetaPRL, PVS, . . .)

However, there are reasons to stick to simple type theory.

Several highly developed provers based on simple type theory (HOL4, HOL Light, IMPS, Isabelle/HOL, . . .)

Our solution

For $\mathbb{R}^n$ use the function space $\tau \to \mathbb{R}$ where $|\tau| = n$.

With some technical groundwork, this gives quite a nice solution:

- Operations can be defined generically with no parametrization
- Use polymorphic type variables in place of numeric parameters
- Use constructors like disjoint sum for ”arithmetic” on indices
- Theorems about $\mathbb{R}^2$ etc. are really *instances* of results for $\mathbb{R}^\alpha$

Main downside: types are still not completely ‘first class’, so can’t trivially do induction on dimension etc.

Gory details

Define a binary type constructor ‘`^`’.

Second argument is coerced to size 1 if infinite.

Indexing function `($):A^N->num->A`.

Components are `x$1`, `x$2`, `x$3` etc.

Special notion of lambda-binding for componentwise expressions so that `(lambda i. t[i])$j = t[j]`.

Basic definitions

```
 |- x + y = lambda i. x$i + y$i
```

```
 |- c % x = lambda i. c * x$i
```

```
 |- vec n = lambda i. &n
```

For summations, looks similar to $x \cdot y = \sum_{i=1}^{n} x_i y_i$:

```
|- (x:real^N) dot (y:real^N) =
       sum(1..dimindex(:N)) (λi. x$i * y$i)
```

Norms etc.

Define some of the usual vector notions:

```
|- norm x = sqrt(x dot x)

|- dist(x,y) = norm(x - y)

|- orthogonal x y ⇔ (x dot y = &0)
```

and linear functions:

```
|- linear (f:real^M->real^N) ⇔
       (∀x y. f(x + y) = f(x) + f(y)) ∧
       (∀c x. f(c % x) = c % f(x))
```

Matrices

Encode $M \times N$ matrices by $(\mathbb{R}^N)^M$. Multiplication:

```
|- (A:real^N^M) ** (B:real^P^N) =
     lambda i j. sum (1..dimindex(UNIV:N->bool))
                     (λk. A$i$k * B$k$j)
```

Types give a natural way of enforcing dimensional compatibility in matrix multiplication.

```
|- ∀A:real^N^M. linear(λx. A ** x)
```

```
|- ∀f:real^M->real^N.
        linear f ⇒ ∀x. matrix f ** x = f(x)
```

```
|- ∀f g. linear f ∧ linear g
         ⇒ (matrix(g o f) = matrix g ** matrix f)
```

Topology

Induction over dimension in Heine-Borel and Brouwer are OK:

```
|- compact s ⇔
     ∀f:num->realˆN.
       (∀n. f(n) IN s)
       ⇒ ∃l r. l IN s ∧ (∀m n. m < n ⇒ r(m) < r(n))
                ((f o r) --> l) sequentially

|- compact s ⇔ bounded s ∧ closed s

|- ∀ f:realˆN->realˆN.
       compact s ∧ convex s ∧ ¬(s = {}) ∧
       f continuous_on s ∧ IMAGE f s SUBSET s
       ⇒ ∃x. x IN s ∧ f x = x
```

Convex sets and polyhedra

Classic properties of convex sets such as Radon’s theorem

```
|- ∀c. affine_dependent c
       ⇒ ∃m p. m SUBSET c ∧ p SUBSET c ∧ DISJOINT m
            ¬(DISJOINT (convex hull m) (convex hull p
```

and results about polytopes and polyhedra, their faces etc.

```
|- polytope s <=> ∃v. FINITE v ∧ s = convex hull v

|- polyhedron s <=>
     ∃f. FINITE f ∧ s = INTERS f ∧
         ∀h. h IN f ⇒ ∃a b. ¬(a = vec 0) ∧
                             h = {x | a dot x <= b}

|- ∀s. polytope s <=> polyhedron s ∧ bounded s
```

Analysis

Usual Fréchet derivative:

```
|- (f has_derivative f’) (at x) ⇔
        linear f’ ∧
        ((λy. inv(norm(y - x)) % (f(y) - (f(x) + f’(y -
         --> vec 0)
        (at x)
```

and typical theorems:

```
|- (f has_derivative f’) (at x) ∧
   (g has_derivative g’) (at (f x))
   ⇒ ((g o f) has_derivative (g’ o f’)) (at x)
```

Integration and measure

Kurzweil-Henstock gauge integral for functions $\mathbb{R}^M \to \mathbb{R}^N$:

```
|- ∀f g h s.
     (∀k. (f k) integrable_on s) ∧ h integrable_on s
     (∀k x. x IN s ⇒ norm(f k x) <= drop(h x)) ∧
     (∀x. x IN s ⇒ ((λk. f k x) --> g x) sequentially
     ⇒ g integrable_on s ∧
        ((λk. integral s (f k)) --> integral s g)
         sequentially
```

and Lebesgue measure as integral of characteristic function:

```
|- ∀f:realˆN->realˆN.
        linear f ∧ measurable s
        ⇒ (IMAGE f s) has_measure
            (abs(det(matrix f)) * measure s)
```

Complex analysis (1)

Define complex derivatives and analytic functions, and relate it to general differentiability in Euclidean space:

```
|- ∀f z. f complex_differentiable at z <=>
         f differentiable at z  ∧
         (jacobian f (at z))$1$1 =
         (jacobian f (at z))$2$2 ∧
         (jacobian f (at z))$1$2 =
         --((jacobian f (at z))$2$1)
```

Many other analytic theorems are proved.

Complex analysis (2)

More interesting theorems involve contour integrals, like Cauchy’s integral formula:

```
|- ∀f s k g z.
     convex s ∧ FINITE k ∧ f continuous_on s ∧
     (∀x. x IN interior(s) DIFF k
          ⇒ f complex_differentiable at x) ∧
     z IN interior(s) DIFF k ∧
     valid_path g ∧
     (path_image g) SUBSET (s DELETE z) ∧
     pathfinish g = pathstart g
     ⇒ ((λw. f(w) / (w - z)) has_path_integral
       (Cx(&2 * pi) * ii * winding_number(g,z) * f(z)
        g
```

Automated reasoning routines

As well as *theorems*, we have also developed a few convenient automated proof tools:

- Simple routine for automatically proving universally quantified vector formulas ‘componentwise’
- More sophisticated quantifier elimination for vectors, based on an idea of Solovay.
- ‘Without loss of generality’ tactics for exploiting symmetries and invariances, especially in geometry.

WLOG example

```
g ‘∀s a:realˆN.
     closed s ∧ ¬(s = {})
     ⇒ ∃x. x IN s ∧
           ∀y. y IN s ⇒ dist(a,x) <= dist(a,y)‘;;
```

With a single application of our tactic, we can suppose the point in question is the origin:

```
# e(GEOM_ORIGIN_TAC ‘a:realˆN‘);;
val it : goalstack = 1 subgoal (1 total)

‘∀s. closed s ∧ ¬(s = {})
     ⇒ ∃x. x IN s ∧
           ∀y. y IN s ⇒ dist(vec 0,x) <= dist(vec 0,y
```

Summary

- Simple but apparently effective representational trick
- Many definitions and theorems have a very natural formulation
- Some potential difficulties over induction on dimension etc.
- Has been developed into a substantial library with many classic theorems
- Supplemented with some convenient proof tools
- Seems to provide a good foundation for Flyspeck work
- Also used for a significant development of complex analysis