Without Loss of Generality

John Harrison

Intel Corporation, JF1-13
2111 NE 25th Avenue, Hillsboro OR 97124, USA
johnh@ichips.intel.com

Abstract. One sometimes reads in a mathematical proof that a certain assumption can be made 'without loss of generality' (WLOG). In other words, it is claimed that considering what first appears only a special case does nevertheless suffice to prove the general result. Typically the intuitive justification for this is that one can exploit symmetry in the problem. We examine how to formalize such 'WLOG' arguments in a mechanical theorem prover. Geometric reasoning is particularly rich in examples and we pay special attention to this area.

1 Introduction

Mathematical proofs sometimes state that a certain assumption can be made 'without loss of generality', often abbreviated to 'WLOG'. The phase suggest that although making the assumption at first sight only proves the theorem in a more restricted case, this does nevertheless justify the theorem in full generality. What is the intuitive justification for this sort of reasoning? Occasionally the phrase covers situations where we neglect special cases that are obviously trivial for other reasons. But more usually it suggests the exploitation of symmetry in the problem. For example, consider Schur's inequality, which asserts that for any nonnegative real numbers a, b and c and integer $k \geq 0$ one has $0 \leq a^k(a-b)(a-c) + b^k(b-a)(b-c) + c^k(c-a)(c-b)$. A typical proof might begin:

> Without loss of generality, let $a \leq b \leq c$.

If asked to spell this out in more detail, we might say something like:

> Since $\leq$ is a total order, the three numbers must be ordered somehow, i.e. we must have (at least) one of $a \leq b \leq c$, $a \leq c \leq b$, $b \leq a \leq c$, $b \leq c \leq a$, $c \leq a \leq b$ or $c \leq b \leq a$. But the theorem is completely symmetric between a, b and c, so each of these cases is just a version of the other with a change of variables, and we may as well just consider one of them.

Suppose that we are interested in formalizing mathematics in a mechanical theorem prover. Generally speaking, for an experienced formalizer it's rather routine to take an existing proof and construct a formal counterpart, even though it may require a great deal of work to get things just right and encourage the proof assistant check all the details. But with such 'without loss of generality' constructs, it's not immediately obvious what the formal counterpart should be. We can plausibly suggest two possible formalizations:

– The phrase may be an informal shorthand saying ‘we should really do 6 very similar proofs here, but if we do one, all the others are exactly analogous and can be left to the reader’.
– The phrase may be asserting that ‘by a general logical principle, the apparently more general case and the special WLOG case are in fact equivalent (or at least the special case implies the general one)’.

The former point of view can be quite natural in a computer proof assistant. If we have a proof script covering one of the 6 cases, we might simply perform a 6-way case-split and for each case use a duplicate of the initial script, changing the names of variables systematically in an editor. Indeed, if we have a programmable proof assistant, it would be more elegant to write a general parametrized proof script that we could use for all 6 cases with different parameters. This sort of programming is exactly the kind of thing that LCF-style systems [3] like HOL [2] are designed to make easy via their ‘metalanguage’ ML, and sometimes its convenience makes it irresistible. However, this approach is open to criticism on at least three grounds:

– Ugly/clumsy
– Inefficient
– Not faithful to the informal proof.

Indeed, it seems unnatural, even with the improvement of using a parametrized script, to perform essentially the same proof 6 different times, and if each proof takes a while to run, it could waste computer resources. And it is arguably *not* what the phrase ‘without loss of generality’ is meant to conjure up. If the book had intended that interpretation, it would probably have said something like ‘the other cases are similar and are left to the reader’. So let us turn to how we might formalize and use a general logical principle.

2 A HOL Light proof of Schur’s inequality

In fact, in HOL Light there is already a standard theorem with an analogous principle for a property of *two* real numbers:

```
REAL_WLOG_LE =
  |- (∀x y. P x y ⇔ P y x) ∧
     (∀x y. x <= y ⇒ P x y)
     ⇒ (∀x y. P x y)
```

This asserts that for any property P of two real numbers, if the property is symmetric between those two numbers ($\forall x\ y.\ P\ x\ y \Leftrightarrow P\ y\ x$) and assuming $x \leq y$ the property holds ($\forall x\ y.\, x \leq y \Rightarrow P\ x\ y$), then we can conclude that it holds for all real numbers ($\forall x\ \ y.\ P\ \ x\ \ y$). In order to tackle the Schur inequality we will prove a version for *three* variables. Our chosen formulation is quite analogous, but using a more minimal formulation of symmetry between all three variables:

```
REAL_WLOG_3_LE =
  |- (∀x y z. P x y z ⇒ P y x z ∧ P x z y) ∧
     (∀x y z. x <= y ∧ y <= z ⇒ P x y z)
     ⇒ (∀x y z. P x y z)
```

The proof is relatively straightforward following the informal intuition: we observe that one of the six possible ordering sequences must occur, and in each case we can deduce the general case from the more limited one and symmetry. The following is the tactic script to prove `REAL_WLOG_3_LE`:

```
REPEAT STRIP_TAC THEN (STRIP_ASSUME_TAC o REAL_ARITH)
 `x <= y ∧ y <= z ∨ x <= z ∧ z <= y ∨ y <= x ∧ x <= z ∨
  y <= z ∧ z <= x ∨ z <= x ∧ x <= y ∨ z <= y ∧ y <= x` THEN
ASM_MESON_TAC[]
```

Now let us see how to use this to prove Schur's inequality in HOL Light, which we formulate as follows:

```
|- ∀k a b c. &0 <= a ∧ &0 <= b ∧ &0 <= c
             ⇒ &0 <= a pow k * (a - b) * (a - c) +
                     b pow k * (b - a) * (b - c) +
                     c pow k * (c - a) * (c - b)
```

The first step in the proof is to strip off the additional variable k (which will not play a role in the symmetry argument), use backwards chaining with the WLOG theorem `REAL_WLOG_3_LE`, and then break the resulting goal into two subgoals, one corresponding to the symmetry and the other to the special case.

```
GEN_TAC THEN MATCH_MP_TAC REAL_WLOG_3_LE THEN CONJ_TAC
```

The first subgoal, corresponding to symmetry of the problem, is the following:

```
`∀a b c. (&0 <= a ∧ &0 <= b ∧ &0 <= c
          ⇒ &0 <= a pow k * (a - b) * (a - c) +
                  b pow k * (b - a) * (b - c) +
                  c pow k * (c - a) * (c - b))
         ⇒ (&0 <= b ∧ &0 <= a ∧ &0 <= c
             ⇒ &0 <= b pow k * (b - a) * (b - c) +
                     a pow k * (a - b) * (a - c) +
                     c pow k * (c - b) * (c - a)) ∧
            (&0 <= a ∧ &0 <= c ∧ &0 <= b
             ⇒ &0 <= a pow k * (a - c) * (a - b) +
                     c pow k * (c - a) * (c - b) +
                     b pow k * (b - a) * (b - c))`
```

Although this looks rather large, the proof simply exploits the fact that addition and multiplication are associative and commutative via routine logical reasoning, so we can solve it by:

```
MESON_TAC[REAL_ADD_AC; REAL_MUL_AC]
```

We have now succeeded in reducing the original goal to the special case:

```
`∀a b c. a <= b ∧ b <= c
         ⇒ &0 <= a ∧ &0 <= b ∧ &0 <= c
            ⇒ &0 <= a pow k * (a - b) * (a - c) +
                     b pow k * (b - a) * (b - c) +
                     c pow k * (c - a) * (c - b)`
```

and so we can claim that the foregoing proof steps correspond almost exactly to the informal WLOG principle. We now rewrite the expression into a more convenient form:

```
REPEAT STRIP_TAC THEN ONCE_REWRITE_TAC[REAL_ARITH
 `a pow k * (a - b) * (a - c) +
  b pow k * (b - a) * (b - c) +
  c pow k * (c - a) * (c - b) =
  (c - b) * (c pow k * (c - a) - b pow k * (b - a)) +
  a pow k * (c - a) * (b - a)`]
```

The form of this expression is now congenial, so we can simply proceed by repeatedly chaining through various monotonicity theorems and then use linear arithmetic reasoning to finish the proof:

```
REPEAT(FIRST(map MATCH_MP_TAC
         [REAL_LE_ADD; REAL_LE_MUL; REAL_LE_MUL2]) THEN
       ASM_SIMP_TAC[REAL_POW_LE2; REAL_POW_LE; REAL_SUB_LE] THEN
       REPEAT CONJ_TAC) THEN
ASM_REAL_ARITH_TAC
```

We have therefore succeeded in deploying WLOG reasoning in a natural way and following a standard textbook proof quite closely. However, a remaining weak spot is the proof of the required symmetry for the particular problem. In this case, we were just able to use standard first-order automation (`MESON_TAC`) to deduce this symmetry from the associativity and commutativity of the two main operations involved (real addition and multiplication). However, we can well imagine that in more complicated situations, this kind of crude method might be tedious or impractical. We will investigate how to approach this more systematically using reasoning from a somewhat different domain.

3 WLOG reasoning in geometry

Geometry is particularly rich in WLOG principles, perhaps reflecting the fundamental importance in geometry of property-preserving transformations. The modern view of geometry has been heavily influenced by Klein's "Erlanger Programm" [7], which emphasizes the role of transformations and invariance under classes of transformations, while modern physical theories usually regard conservation laws as manifestations of invariance properties: the conservation of angular momentum arises from invariance under rotations, while conservation of energy arises from invariance under shifts in time, and so on [8].

One of the most important ways in which such invariances are used in proofs is to make a convenient choice of coordinate system. In our formulation of Euclidean space in HOL Light [6], geometric concepts are all defined in analytic terms using vectors, which in turn are expressed with respect to a standard coordinate basis. For example, the angle formed by three points is defined in terms of the angle between two vectors:

```
|- angle(a,b,c) = vector_angle (a - b) (c - b)
```

which is defined in terms of norms and dot products using the inverse cosine function `acs` (degenerating to $\pi/2$ if either vector is zero):

```
|- vector_angle x y =
       if x = vec 0 ∨ y = vec 0 then pi / &2
       else acs((x dot y) / (norm x * norm y))
```

where norms are defined in terms of dot products:

```
|- ∀x. norm x = sqrt(x dot x)
```

and finally dot products in $\mathbb{R}^N$ are defined in terms of the N components in the usual way as $x \cdot y = \sum_{i=1}^{N} x_i y_i$, or in HOL Light:

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

This means that whenever we state geometric theorems, most of the concepts ultimately rest on a *particular* choice of coordinate system and standard basis vectors. When we are performing high-level reasoning, we can often reason about geometric concepts directly using lemmas established earlier without ever dropping down to the ultimate representation with respect to the standard basis. But when we *do* need to reason algebraically in terms of coordinates, we often find that a different choice of coordinate system would make the reasoning much more tractable.

The simplest example is probably choosing the origin of the coordinate system. If a proposition $\forall x.\ P[x]$ is invariant under spatial translation, i.e. changing x to any $a + x$, then it suffices to prove the special case $P[0]$, or in other words, to assume without loss of generality that x is the origin. The reasoning is essentially trivial: if we have $P[0]$ and also $\forall a\ x.\ P[x] \Rightarrow P[a + x]$, then we can deduce $P[x + 0]$ and so $P[x]$. In HOL Light we can state this as the following general theorem, asserting that if P is invariant under translation and we have the special case $P[0]$, then we can conclude $\forall x.\ P[x]$:

```
WLOG_ORIGIN =
  |- (∀a x. P(a + x) ⇔ P x) ∧ P(vec 0) ⇒ (∀x. P x)
```

Thus, when confronted with a goal, we can simply rearrange the universally quantified variables so that the one we want to take as the origin is at the outside, then apply this theorem, giving us the special case $P[0]$ together with the invariance of the goal under translation. For example, suppose we want to prove that the angles of a triangle that is not completely degenerate all add up to π radians (180 degrees):

```
`∀A B C. ~(A = B ∧ B = C ∧ A = C)
         ⇒ angle(B,A,C) + angle(A,B,C) + angle(B,C,A) = pi`
```

If we apply our theorem by `MATCH_MP_TAC WLOG_ORIGIN` and split the resulting goal into two conjuncts, we get one subgoal corresponding to the special case when A is the origin:

```
`∀B C.
   ~(vec 0 = B ∧ B = C ∧ vec 0 = C)
   ⇒ angle(B,vec 0,C) + angle(vec 0,B,C) + angle(B,C,vec 0) =
       pi`
```

and another goal for the invariance of the property under translation by a:

```
`∀a A. (∀B C.
            ~(a + A = B ∧ B = C ∧ a + A = C)
            ⇒ angle(B,a + A,C) +
               angle(a + A,B,C) + angle(B,C,a + A) = pi) ⇔
       (∀B C.
            ~(A = B ∧ B = C ∧ A = C)
            ⇒ angle(B,A,C) + angle(A,B,C) + angle(B,C,A) = pi)`
```

We will not dwell more on the detailed proof of the theorem in the special case where A is the origin, but will instead focus on the invariance proof. In contrast to the case of Schur's inequality, this is somewhat less easy and can't obviously be deferred to basic first-order automation. So how do we prove it?

At first sight, things don't look right: it seems that we ought to have translated not just A but *all* the variables A, B and C together. However, note that for any given a the translation mapping $x \mapsto a + x$ is surjective: for any y there is an x such that $a + x = y$ (namely $x = y - a$). That means that we can replace universal quantifiers over vectors, and even existential ones too, by translated versions. This general principle can be embodied in the following HOL theorem, easily proven automatically by `MESON_TAC`:

```
QUANTIFY_SURJECTION_THM =
 |- ∀f:A->B.
        (∀y. ∃x. f x = y)
        ⇒ (∀P. (∀x. P x) ⇔ (∀x. P (f x))) ∧
           (∀P. (∃x. P x) ⇔ (∃x. P (f x))) ∧
```

We can apply it with a bit of instantiation and higher-order rewriting to all the universally quantified variables on the left-hand side of the equivalence in the goal and obtain:

```
`∀a A.
   (∀B C.
        ~(a + A = a + B ∧ a + B = a + C ∧ a + A = a + C)
        ⇒ angle(a + B,a + A,a + C) +
           angle(a + A,a + B,a + C) +
           angle(a + B,a + C,a + A) = pi) ⇔
   (∀B C.
        ~(A = B ∧ B = C ∧ A = C)
        ⇒ angle(B,A,C) + angle(A,B,C) + angle(B,C,A) = pi)`
```

Now things are becoming better. First of all, it is clear that $a + A = a + B \Leftrightarrow A = B$ etc. just by general properties of vector addition. As for angles, recall that `angle(x,y,z)` is defined as the vector angle between the two differences $x - y$ and $z - y$. Because it is defined in terms of such differences, it again follows from basic properties of vector addition that, for example, $(a + B) - (a + A) = A - B$, and so we can deduce the invariance property that we seek.

This is all very well, but the process is quite laborious. We have to carefully apply translation to all the quantified variables just once so that we don't get into an infinite loop, and then we have to appeal to suitable basic invariance theorems for pretty much *all* the concepts that appear in our theorems. Even in this case, doing so is not entirely trivial, and for more involved theorems it can be worse, as Hales [5] notes:

> [. . .] formal proofs by symmetry are much harder than anticipated. It was necessary to give a total of nearly a hundred lemmas, showing that the symmetries preserve all of the relevant structures, all the way back to the foundations.

Indeed, this process seems unpleasant enough that we should consider automating it, and for geometric invariants this is just what we have done.

4 Tactics using invariance under translation

Our WLOG tactic for choosing the origin is based on a list of theorems asserting invariance under translation for various geometric concepts, stored in a reference variable `invariant_under_translation`. The vision is that each time a new geometric concept (angle, collinear, etc.) is defined, one proves a corresponding invariance theorem and adds it to this list, so that thereafter the invariance will be exploitable automatically by the WLOG tactic. For example, the entry corresponding to `angle` is

```
|- ∀a b c d. angle(a + b,a + c,a + d) = angle(b,c,d)
```

While we usually aim to prove that numerical functions of vectors (e.g. distances or angles) or predicates on vectors (e.g. collinearity) are completely invariant under translation, for operations returning more vectors, we normally want to prove that the translation can be 'pulled outside', e.g.

```
|- ∀a x y. midpoint(a + x,a + y) = a + midpoint (x,y)
```

Then a translated formula can be systematically mapped into its untranslated form by applying these transformations in a bottom-up fashion, pulling the translation up through vector-producing functions like `midpoint` and then systematically eliminating them when they reach the level of predicates or numerical functions of vectors.

Our setup is somewhat more ambitious in that it applies not only to properties of vectors but also to properties of *sets* of vectors, many of which are also invariant under translation. For example, recall that a set is convex if whenever it contains the points x and y it also contains each intermediate point between x and y, i.e. each $ux + vy$ where $0 \leq u, v$ and $u + v = 1$:

```
|- ∀s. convex s ⇔
       (∀x y u v.
            x IN s ∧ y IN s ∧ &0 <= u ∧ &0 <= v ∧ u + v = &1
            ⇒ u % x + v % y IN s)
```

This is invariant under translation in the following sense:

```
|- ∀a s. convex (IMAGE (λx. a + x) s) ⇔ convex s
```

as are many other geometric or topological predicates (bounded, closed, compact, path-connected, ...) and numerical functions on sets such as measure (area, volume etc. depending on dimension):

```
|- ∀a s. measure (IMAGE (λx. a + x) s) = measure s
```

As with points, for functions that return other sets of vectors, our theorems state rather that the 'image under translation' operation can be pulled up through the function, e.g.

```
|- ∀a s. convex hull IMAGE (λx. a + x) s =
         IMAGE (λx. a + x) (convex hull s)
```

We include in the list other theorems of the same type for the basic set operations, so that they can be handled as well, e.g.

```
|- ∀a s t. IMAGE (λy. a + y) s UNION IMAGE (λy. a + y) t =
           IMAGE (λy. a + y) (s UNION t)

|- ∀a s t. IMAGE (λy. a + y) s SUBSET IMAGE (λy. a + y) t ⇔
           s SUBSET t
```

Our conversion (`GEOM_ORIGIN_CONV`) and corresponding tactic that is defined in terms of it (`GEOM_ORIGIN_TAC`) work by automating the process sketched in the special case in the previous section. First, they apply the basic reduction so that we need to prove equivalence when one nominated variable is translated. Then the other quantifiers are modified to apply similar translation to the other variables, even if quantification is nested in a complicated way. We use an enhanced version of the theorem

QUANTIFY_SURJECTION_THM which applies a similarly systematic modification to quantifiers over sets of vectors and set comprehensions such as $\{x \mid \text{angle}(a, b, x) = \pi/3\}$:

```
|- ∀f. (∀y. ∃x. f x = y)
       ⇒ (∀P. (∀x. P x) ⇔ (∀x. P (f x))) ∧
          (∀P. (∃x. P x) ⇔ (∃x. P (f x))) ∧
          (∀Q. (∀s. Q s) ⇔ (∀s. Q (IMAGE f s))) ∧
          (∀Q. (∃s. Q s) ⇔ (∃s. Q (IMAGE f s))) ∧
          (∀P. {x | P x} = IMAGE f {x | P (f x)})
```

With this done, it remains only to rewrite with the invariance theorems taken from the list invariant_under_translation in a bottom-up sweep. If the intended result uses only these properties in a suitable fashion, then this should automatically reduce the invariance goal to triviality. The user does not even see it, but is presented instead with the special case. (If the process of rewriting does not solve the invariance goal, then that is returned as an additional subgoal so that the user can either help the proof along manually or perhaps observe that a concept is used for which no invariance theorem has yet been stored.) For example, if we set out to prove the formula for the volume of a ball:

```
`∀z:real^3 r. &0 <= r
              ⇒ measure(cball(z,r)) = &4 / &3 * pi * r pow 3`
```

a simple application of GEOM_ORIGIN_TAC `z:real^3` reduces it to the special case when the ball is centered at the origin:

```
`∀r. &0 <= r
     ⇒ measure(cball(vec 0,r)) = &4 / &3 * pi * r pow 3`
```

Here is an example with a more complicated quantifier structure and a mix of sets and points. We want to prove that for any point a and nonempty closed set s there is a closest point of s to a. (A set is closed if it contains all its limit points, i.e. all points that can be approached arbitrarily closely by a member of the set.) We set up the goal:

```
g `∀s a:real^N.
     closed s ∧ ~(s = {})
     ⇒ ∃x. x IN s ∧
           (∀y. y IN s ⇒ dist(a,x) <= dist(a,y))`;;
```

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

```
# e(GEOM_ORIGIN_TAC `a:real^N`);;
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)))`
```

5 Tactics using invariance under orthogonal transformations

This is just one of several analogous tactics that we have defined. Many other tactics also exploit the invariance of many properties under *orthogonal transformations*. These are essentially maps $f : \mathbb{R}^N \to \mathbb{R}^N$ that are linear and preserve dot products:

```
|- ∀f. orthogonal_transformation f ⇔
       linear f ∧ (∀v w. f v dot f w = v dot w)
```

where linearity of a function $f : \mathbb{R}^M \to \mathbb{R}^N$ is defined as

```
|- ∀f. linear f ⇔
       (∀x y. f(x + y) = f x + f y) ∧
       (∀c x. f(c % x) = c % f x)
```

Orthogonal transformations can be characterized in various other ways. For example, a linear map f is an orthogonal transformation iff its corresponding matrix is an orthogonal matrix:

```
|- orthogonal_transformation f ⇔
   linear f ∧ orthogonal_matrix(matrix f)
```

where an $N \times N$ matrix Q is orthogonal if its transpose is also its inverse, i.e. $Q^T Q = QQ^T = 1$:

```
|- orthogonal_matrix(Q) ⇔
      transp(Q) ** Q = mat 1 ∧ Q ** transp(Q) = mat 1`;;
```

It is easy to prove that the determinant of an orthogonal matrix is either 1 or -1, and this gives a classification of orthogonal transformations into 'rotations', where the matrix has determinant 1 and 'rotoinversions' where the matrix has determinant -1. Intuitively, rotations do indeed correspond to rotation about the origin in n-dimensional space, while rotoinversions involve additional reflections. For example, in two dimensions, each rotation matrix is of the form

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

where θ is the (anticlockwise) angle of rotation. Invariance under orthogonal transformation is used in several tactics that allow us to transform a particular nonzero vector into another more convenient one of the same magnitude. The following theorem guarantees us that given any two vectors a and b in $\mathbb{R}^N$ of the same magnitude, there exists an orthogonal transformation that maps one into the other:

```
|- ∀a b:real^N.
        norm(a) = norm(b)
        ⇒ ∃f. orthogonal_transformation f ∧ f a = b
```

If we furthermore want $f : \mathbb{R}^N \to \mathbb{R}^N$ be a rotation, then *almost* the same theorem is true, except that we need the dimension to be at least 2. (An orthogonal transformation taking a vector into its negation in $\mathbb{R}^1$ must have a matrix with determinant -1.)

```
|- ∀a b:real^N.
       2 <= dimindex(:N) ∧ norm(a) = norm(b)
       ⇒ ∃f. orthogonal_transformation f ∧
              det(matrix f) = &1 ∧
              f a = b
```

Just as a reference variable `invariant_under_translation` is used to store theorems asserting the invariance of various concepts under translation, we use a second reference variable `invariant_under_linear` to store analogous theorems for invariance under linear transformations. These in general apply to slightly different classes of linear transformations, almost all of which are more general than orthogonal transformations. For each concept we try to use the most general natural class of linear mappings. Some theorems apply to all linear maps, e.g. the one for convex hulls:

```
|- ∀f s. linear f
        ⇒ convex hull IMAGE f s = IMAGE f (convex hull s)
```

Some apply to all injective linear maps, e.g. those for closedness of a set:

```
|- ∀f s. linear f ∧ (∀x y. f x = f y ⇒ x = y)
        ⇒ (closed (IMAGE f s) ⇔ closed s)
```

Some apply to all bijective (injective and surjective) linear maps, e.g. those for openness of a set:

```
|- ∀f s. linear f ∧
        (∀x y. f x = f y ⇒ x = y) ∧ (∀y. ∃x. f x = y)
        ⇒ (open (IMAGE f s) ⇔ open s)
```

Some apply to all norm-preserving linear maps, e.g. those for angles:

```
|- ∀f a b c. linear f ∧ (∀x. norm(f x) = norm x)
            ⇒ angle(f a,f b,f c) = angle(a,b,c)
```

Note that a norm-preserving linear map is also injective, so this property also suffices for all those requiring injectivity. For a function $f : \mathbb{R}^N \to \mathbb{R}^N$ this property is precisely equivalent to being an orthogonal transformation:

```
|- ∀f:real^N->real^N.
       orthogonal_transformation f ⇔
       linear f ∧ (∀v. norm(f v) = norm v)
```

However, it is important for some other related applications (an example is below) that we make theorems applicable to maps where the dimensions of the domain and codomain spaces are not necessarily the same.

Finally, the most restrictive requirement applies to just one theorem, the one for the vector cross product. This has a kind of chirality, so may have its sign changed by a general orthogonal transformation. Its invariance theorem requires a *rotation* of type $\mathbb{R}^3 \to \mathbb{R}^3$:

```
|- ∀f x y. linear f ∧
           (∀x. norm(f x) = norm x) ∧ det(matrix f) = &1
           ⇒ (f x) cross (f y) = f(x cross y)
```

We actually store the theorem in a slightly peculiar form, which makes it easier to apply uniformly in a framework where we can assume a transformation is a rotation except in dimension 1:

```
|- ∀f x y. linear f ∧ (∀x. norm(f x) = norm x) ∧
           (2 <= dimindex(:3) ⇒ det(matrix f) = &1)
           ⇒ (f x) cross (f y) = f(x cross y)
```

We can implement various tactics that exploit our invariance theorems to make various simplifying transformations without loss of generality:

– GEOM_BASIS_MULTIPLE_TAC chooses an orthogonal transformation or rotation to transform a vector into a nonnegative multiple of a chosen basis vector.
– GEOM_HORIZONTAL_PLANE_TAC chooses a combination of a translation and orthogonal transformation to transform a plane p in $\mathbb{R}^3$ into a 'horizontal' one $\{(x, y, z) \mid z = 0\}$.
– PAD2D3D_TAC transforms a problem in $\mathbb{R}^3$ where all points have zero third coordinate into a corresponding problem in $\mathbb{R}^2$.

The first two work in much the same way as the earlier tactic for choosing the origin. We apply the general theorem, modify all the other quantified variables and then rewrite with invariance theorems. We can profitably think of the basic processes in such cases as instances of general HOL theorems, though this is not actually how they are implemented. For example, we might say that if for each x we can find a 'transform' (e.g. translation, or orthogonal transformation) f such that $f(x)$ is 'nice' (e.g. is zero, or a multiple of some basis vector), and can also deduce for any 'transform' that $P(f(x)) \Leftrightarrow P(x)$, then proving $P(x)$ for all x is equivalent to proving it for 'nice' x. (The theorem that follows is automatically proved by MESON.)

```
|- ∀P. (∀x. ∃f. transform(f) ∧ nice(f x)) ∧
       (∀f x. transform(f) ⇒ (P(f x) ⇔ P x))
       ⇒ ((∀x. P x) ⇔ (∀x. nice(x) ⇒ P(x)))
```

However, in some more general situations we don't exactly want to show that $P(f(x)) \Leftrightarrow P(x)$, but rather that $P(f(x)) \Leftrightarrow P'(x)$ for some related but not identical property P', for example if we want to transfer a property to a different type. For this reason, it is actually more convenient to observe that we can choose a 'transform' *from* a 'nice' value rather than *to* it, i.e. rely on the following:

```
|- ∀P P'. (∀x. ∃f y. transform(f) ∧ nice(y) ∧ f y = x) ∧
          (∀f x. transform(f) ∧ nice x ⇒ (P(f x) ⇔ P' x))
          ⇒ ((∀x. P x) ⇔ (∀y. nice(y) ⇒ P'(y)))`
```

The advantage of this is that in our approach based on rewriting by applying invariance theorems, the new property P' can emerge naturally from the rewriting of $P(f(x))$, instead of requiring extra code for its computation. Even in cases where the generality is not needed, we typically use this structure, i.e. choose our mapping *from* a 'nice' value.

6 An extended example

Let us see a variety of our tactics at work on a problem that was, in fact, the original motivation for most of the work described here.

```
`∀u1:real^3 u2 p a b.
     ~(u1 = u2) ∧
     plane p ∧
     {u1,u2} SUBSET p ∧
     dist(u1,u2) <= a + b ∧
     abs(a - b) < dist(u1,u2) ∧
     &0 <= a ∧
     &0 <= b
     ⇒ (∃d1 d2. {d1,d2} SUBSET p ∧
                &1 / &2 % (d1 + d2) IN affine hull {u1, u2} ∧
                dist(d1,u1) = a ∧
                dist(d1,u2) = b ∧
                dist(d2,u1) = a ∧
                dist(d2,u2) = b)`
```

The first step is to assume without loss of generality that the plane p is $\{(x,y,z) \mid z = 0\}$, i.e. the set of points whose third coordinate is zero, following which we manually massage the goal so that the quantifiers over u_1, u_2, d_1 and d_2 carry explicit restrictions:

```
# e(GEOM_HORIZONTAL_PLANE_TAC `p:real^3->bool` THEN
    ONCE_REWRITE_TAC[TAUT
     `a ∧ b ∧ c ∧ d ⇒ e ⇔ c ∧ a ∧ b ∧ d ⇒ e`] THEN
    REWRITE_TAC[INSERT_SUBSET; EMPTY_SUBSET] THEN
    REWRITE_TAC[IMP_CONJ; RIGHT_FORALL_IMP_THM] THEN
    REWRITE_TAC[GSYM CONJ_ASSOC; RIGHT_EXISTS_AND_THM] THEN
    REWRITE_TAC[IN_ELIM_THM]);;
```

which produces the result:

```
`∀u1. u1$3 = &0
      ⇒ (∀u2. u2$3 = &0
                ⇒ ~(u1 = u2)
                ⇒ plane {z | z$3 = &0}
                ⇒ (∀a b.
                          dist(u1,u2) <= a + b
                          ⇒ abs(a - b) < dist(u1,u2)
                          ⇒ &0 <= a
                          ⇒ &0 <= b
                          ⇒ (∃d1. d1$3 = &0 ∧
                                  (∃d2. d2$3 = &0 ∧
                                        &1 / &2 % (d1 + d2) IN
                                        affine hull {u1, u2} ∧
                                        dist(d1,u1) = a ∧
                                        dist(d1,u2) = b ∧
                                        dist(d2,u1) = a ∧
                                        dist(d2,u2) = b))))`
```

Now we apply another WLOG tactic to reduce the problem from $\mathbb{R}^3$ to $\mathbb{R}^2$, and again make a few superficial rearrangements:

```
# e(PAD2D3D_TAC THEN
    SIMP_TAC[RIGHT_IMP_FORALL_THM; IMP_IMP; GSYM CONJ_ASSOC]);;
```

resulting in:

```
`∀u1 u2 a b.
     ~(u1 = u2) ∧
     plane {z | z$3 = &0} ∧
     dist(u1,u2) <= a + b ∧
     abs(a - b) < dist(u1,u2) ∧
     &0 <= a ∧
     &0 <= b
     ⇒ (∃d1 d2.
              &1 / &2 % (d1 + d2) IN affine hull {u1, u2} ∧
              dist(d1,u1) = a ∧
              dist(d1,u2) = b ∧
              dist(d2,u1) = a ∧
              dist(d2,u2) = b)`
```

Although HOL Light does not by default show the types, all the vector variables are now in $\mathbb{R}^2$ instead of $\mathbb{R}^3$ (except for the bound variable z in the residual planarity hypothesis, which is no longer useful anyway). Having collapsed the problem from 3 dimensions to 2 in this way, we finally choose u_1 as the origin:

```
# e(GEOM_ORIGIN_TAC `u1:real^2`);;
val it : goalstack = 1 subgoal (1 total)

`∀u2 a b.
     ~(vec 0 = u2) ∧
     plane {z | z$3 = &0} ∧
     dist(vec 0,u2) <= a + b ∧
     abs(a - b) < dist(vec 0,u2) ∧
     &0 <= a ∧
     &0 <= b
     ⇒ (∃d1 d2.
              &1 / &2 % (d1 + d2) IN affine hull {vec 0, u2} ∧
              dist(d1,vec 0) = a ∧
              dist(d1,u2) = b ∧
              dist(d2,vec 0) = a ∧
              dist(d2,u2) = b)`
```

and now u_2 as a multiple of the first standard basis vector:

```
# e(GEOM_BASIS_MULTIPLE_TAC 1 `u2:real^2`);;
val it : goalstack = 1 subgoal (1 total)

`∀u2. &0 <= u2
      ⇒ (∀a b.
              ~(vec 0 = u2 % basis 1) ∧
              plane {z | z$3 = &0} ∧
              dist(vec 0,u2 % basis 1) <= a + b ∧
              abs(a - b) < dist(vec 0,u2 % basis 1) ∧
              &0 <= a ∧
              &0 <= b
              ⇒ (∃d1 d2.
                       &1 / &2 % (d1 + d2) IN
                       affine hull {vec 0, u2 % basis 1} ∧
                       dist(d1,vec 0) = a ∧
                       dist(d1,u2 % basis 1) = b ∧
                       dist(d2,vec 0) = a ∧
                       dist(d2,u2 % basis 1) = b))`
```

We have thus reduced the original problem to a nicely oriented situation where the points we consider live in 2-dimensional space and are of the form $(0, 0)$ and $(0, u_2)$. The final coordinate geometry is now relatively straightforward.

7 Future work

Our battery of tactics so far is already a great help in proving geometric theorems. There are several possible avenues for improvement and further development.

One is to make use of still broader classes of transformations when handling theorems about correspondingly narrower classes of concepts. For example, some geometric properties, e.g. those involving collinearity and incidence but not distances and angles, are invariant under still broader classes of transformations, such as *shearing*, and this can be of use in choosing an even more convenient coordinate system — see for example the proof of Pappus's theorem given by Chou [1]. Other classes of theorems behave nicely under scaling, so we can freely turn some point $(0, a) \neq (0, 0)$ into just $(0, 1)$ and so eliminate another variable. Indeed, for still more restricted propositions, e.g. those involving only topological properties, we can consider continuous maps that may not be linear.

It would also be potentially interesting to extend the process to additional 'higher-order' properties. To some extent, we already do this with our support for sets of vectors, but we could take it much further, e.g. considering properties of sequences and series and their limits. A nice example where we would like to exploit a higher-order invariance arises in proving that every polygon has a triangulation. The proof given in [4] says: 'Pick the coordinate axis so that no two vertices have the same y coordinate'. It should not be difficult to extend the methods here to prove invariance of notions like 'triangulation of', and we could then pick a suitable orthogonal transformation to force the required property (there are only finitely many vertices but uncountably many angles of rotation to choose).

Another interesting idea would be to reformulate the process in a more 'metalogical' or 'reflective' fashion, by formalizing the class of problems for which our transformations suffice once and for all, instead of rewriting with the current selection of theorems and then either succeeding or failing. From a practical point of view, we think our current approach is usually better. It is actually appealing *not* to delimit the class of permissible geometric properties, but have that class expand automatically as new invariance theorems are added. Moreover, to use the reflective approach we would need to map into some formal syntax, which needs similar transformations anyway. However, there may be some situations where it would be easier to prove general properties in a metatheoretic fashion. For example, a first-order assertion over vectors with M vector variables, even if the pattern of quantification is involved, can be reduced to spaces of dimension $\leq M$ [9]. It should be feasible to handle important special cases (e.g. purely universal formulas) within our existing framework, but exploiting the full result might be a good use for metatheory.

Acknowledgements

The author is grateful to Truong Nguyen, whose stimulating questions on the Flyspeck project mailing list were the inspiration for most of this work.

References

1. S.-C. Chou. Proving elementary geometry theorems using Wu's algorithm. In W. W. Bledsoe and D. W. Loveland, editors, *Automated Theorem Proving: After 25 Years*, volume 29 of *Contemporary Mathematics*, pages 243–286. American Mathematical Society, 1984.

2. M. J. C. Gordon and T. F. Melham. *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press, 1993.
3. M. J. C. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.
4. T. C. Hales. Easy pieces in geometry. Available at `http://www.math.pitt.edu/~thales/papers/`, 2007.
5. T. C. Hales. The Jordan curve theorem, formally and informally. *The American Mathematical Monthly*, 114:882–894, 2007.
6. J. Harrison. A HOL theory of Euclidean space. In J. Hurd and T. Melham, editors, *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005*, volume 3603 of *Lecture Notes in Computer Science*, pages 114–129, Oxford, UK, 2005. Springer-Verlag.
7. F. Klein. Vergleichende Betrachtungen ber neuere geometrische Forschungen. *Mathematische Annalen*, 43:63–100, 1893. Based on the speech given on admission to the faculty of the Univerity of Erlang in 1872. English translation "A comparative review of recent researches in geometry" in Bulletin of the New York Mathematical Society, vol. 2, (1892-3), pp. 460–497.
8. E. Noether. Invariante Variationsprobleme. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen: Mathematisch-physikalische Klasse*, pages 235–257, 1918. English translation "Invariant variation problems" by M.A. Travel in 'Transport Theory and Statistical Physics', vol. 1, pp. 183–207, 1971.
9. R. M. Solovay, R. Arthan, and J. Harrison. Some new results on decidability for elementary algebra and geometry. ArXiV preprint 0904.3482; submitted to Annals of Pure and Applied Logic. Available at `http://arxiv.org/PS_cache/arxiv/pdf/0904/0904.3482v1.pdf`, 2009.