Journal of Automated Reasoning
https://doi.org/10.1007/s10817-018-9498-9

Formalization of Geometric Algebra in HOL Light

Li-Ming Li[1,2] · Zhi-Ping Shi[1,3,4] · Yong Guan[1,3,4] · Qian-Ying Zhang[1,4] · Yong-Dong Li[4]

Received: 23 November 2016 / Accepted: 17 November 2018

Abstract

Although the theories of geometric algebra (GA) are widely applied in engineering design and analysis, the studies on their formalization have been scarcely conducted. This paper proposes a relatively complete formalization of GA in HOL Light. Both algebraic and geometric parts of the GA theories are formalized successively. For the algebraic part, a uniform abstract product is proposed to facilitate the formalization of the three basic products based on the formal definition of multivectors with three types of metrics. For the geometric part, the formal formulation is provided for the blades and versors and their relations at first. Then, several commonly used specific spaces are formally represented in the theoretical framework of GA. The novelty of the present paper lies in two aspects: (a) the multivector type, `(P,Q,R)geomalg`, is defined and the definition provides the most important foundation for the formalization of geometric algebra, and (b) a procedure is developed for automatically proving the properties of GA operations. The present work improves the function of HOL Light and makes the GA-based formal analysis and verification more convenient.

Keywords Formalization · Geometric algebra · Multivectors · Metrics · HOL Light

✉ Zhi-Ping Shi
shizp@cnu.edu.cn

Li-Ming Li
liliminga@cnu.edu.cn

Yong Guan
guanyong@cnu.edu.cn

Qian-Ying Zhang
qyzhang@cnu.edu.cn

Yong-Dong Li
LYDbeijing@163.com

1 College of Information Engineering, Capital Normal University, Beijing 100048, China

2 School of Mathematical Science, Capital Normal University, Beijing 100048, China

3 Beijing Key Laboratory of Electronic System Reliability Technology, Beijing 100048, China

4 Beijing Key Laboratory of Light Industrial Robot and Safety Verification, Beijing 100048, China

1 Introduction

1.1 Motivation

Geometric algebra (GA) is an advanced piece of mathematical theory widely used in geometric modeling. It provides a general framework that unifies most geometric theories, including the projective geometry, complex numbers, quaternions, etc [1]. Compared with the traditional vector algebra, it adopts multivectors to represent geometric objects and to implement geometric transforms. In this way, it formulates different geometric entities such as lines, planes and volumes in a unified form, which is more suitable for geometric modeling. Due to this reason, it becomes a versatile mathematical language for physics and engineering. In recent years, it has been widely applied in the fields of robotics [2,3], machine vision [4] and computer graphics [5]. Readers can refer to [6–8] for the detail of GA.

Traditionally, applications of the GA are performed by numerical computation and symbolic operations in software such as in Maple [9], Mathematica [10] and Matlab [11]. As is known, numerical computation cannot yield accurate results due to its approximation treatments. Symbolic operations occasionally neglect the prerequisites for mathematical expressions and as a result it sometimes leads to errors. For example, xy/x is frequently regarded as y by the symbolic procedure in Mathematica, even when $x = 0$. Although these inaccurate numerical computation and symbolic operations are acceptable in most engineering cases, the mistakes brought about by error accumulation should not be neglected in some safety-critical systems. Unfortunately, it is very difficult to find out such errors in the existing software by the conventional verification methods.

Comparatively speaking, formal verification is an effective method for finding the above-mentioned errors. At present, some formal software systems (or theorem provers) have been developed for this purpose, including HOL4 [12], HOL Light [13], Isabelle/HOL [14], Coq [15], PVS [16], ACL2 [17] and Mizar [18]. Before conducting formal verification, one generally has to formalize the corresponding mathematical theories in a prover. In the field of GA, some pioneering work has been done to implement its formalization. Arthan formalized $\mathcal{G}_{\infty,0,0}$, which comprises the union of $\mathcal{G}_{p,0,0}$ with $p \in \mathbb{N}$, in the ProofPower-HOL [19]. Fuchs and Théry [20] presented the formalization of GA products within Coq by using the method of binary trees. Harrison [21] formalized the GA products under the framework of Euclidean metric in the HOL Light prover. Based on Harrison's work, our team [22] implemented the formalization of CGA (conformal geometric algebra) in the HOL Light prover. All the existing GA formalizations are incomplete and not intuitively consistent with the GA in mathematics, which involves not only the Euclidean metric but also the non-Euclidean ones.

In the present work, we formalize a complete GA theory that can deal with problems in both Euclidean and non-Euclidean metrics. The proposed formalization of GA provides a practical tool for the verification of GA-based design and analyses in engineering.

1.2 HOL Light

HOL Light is one of the prevailing Higher Order Logic theorem provers [23,24]. It has a robust software architecture, which assures all reasoning work must go through checks by a small kernel. Such a design significantly boosts the reliability for the prover since only the kernel needs to be trusted [25]. HOL Light is suitable for formalizing GA because it has various theorem libraries that are important for GA. Particularly, Harrison's pioneering work,

Table 1 Some logic notations and their related HOL notations

Logic notation	HOL notation	Meaning
$\forall$	`!`	Universal quantifier
$\exists$	`?`	Existential quantifier
$\neg$	`~`	Not
$\wedge$	`/\`	And
$\vee$	`\/`	Or
$\Rightarrow$	`==>`	Implies
$\Leftrightarrow$	`<=>`	Equivalent
λ	`\`	Abstraction

Table 2 Some set notations and their related HOL notations

Set notation	HOL notation	Meaning
$x \in s$	`x IN s`	Set membership
$s \cup t$	`s UNION t`	Set union
$s \cap t$	`s INTER t`	Set intersection
$(s - t) \cup (t - s)$	`s SYMDIFF t`	Set symmetric difference
$s \subseteq t$	`s SUBSET t`	Subset
$s \cap t = \emptyset$	`DISJOINT s t`	Set disjoint
$\{x \mid m \leq x \leq n\}$	`m..n`	Natural number segment
$\lvert s \rvert$	`CARD s`	Cardinality of set

i.e., the formalization of the $\mathcal{G}_N$-based geometric algebra theory in HOL Light [26], offers us a crucial reference.

For the convenience of reading, the logic notations and set notations used in the present work are listed in Tables 1 and 2. Their corresponding HOL symbols and meanings are also given therein.

1.3 Related Work

In this section, the present work is compared with that of Harrison and with our prior work.

There are seven differences between the present work and that of Harrison [26]. (1) Difference in the multivector structures. This is the main difference between Harrisons work and ours. In Harrison's work, $\mathcal{G}_N$ is formalized and the multivectors are denoted by `(N)multivectors`. For quadratic forms with positive definite signatures, Harrison's method is very easy to use. In the present work, we define the multivector structure `real^(P,Q,R)geomalg` to formalize $\mathcal{G}_{p,q,r}$. This allows us to encode quadratic forms with arbitrary signatures. (2) Difference in the application range of the automatic evaluation conversion. The conversion for simplifying GA symbolic expressions developed by Harrison is applicable for only the outer products, but the conversion in the present work can be used to tackle the mixed operations of all three kinds of products. (3) Difference in the definition of abstract product. Harrison formalized the abstract product as `(Product mult op) x y`, but we formalize it as `(Productga sgn) x y`. Our intention was to slightly simplify the original formalization by using a short list of parameters. (4) Difference in the judging theorem of associativity. There is only a single `mult` variable in Harrison's formalization. As a result, associativity can only be expressed for a single type of product operation. In comparison, there are two different variables, `sgn1` and `sgn2`, in the present work. This enables the expression of associativity for two different types of product operations. (5) The formalization of inner product is different in Sect. 2.2. (6) A formal definition of the inverse of geometric product is provided in the present work in Sect. 2.3. (7) The present work provides the definitions and properties for the blade and versor that have explicit geometric meanings in Sect. 3.1. Finally, it should be emphasized again that for the positive definite quadratic forms, Harrison's multivector structure offers the most concise formalization. The previous work of Harrison in HOL Light gives us very important inspiration.

In addition, the present work is also different from the previous work of our research group [22]. In that work, we focused on the formal analysis of the inverse kinematics in CGA. For the purpose of formulating CGA $\mathcal{G}_{4,1}$, the formalization of $\mathcal{G}_{p,q}$ was introduced and the definitions of blade, invertible, dual and reversion were given therein. Comparatively speaking, the GA involved in the present work is more generalized, because it includes the quadratic forms with arbitrary signatures. Overall, the present work improves the formalization of GA in many aspects such as the formalization of the basic GA operations.

1.4 Outline

The GA theory formalized here includes two parts: the algebra and the geometry. The rest of the paper is organized as follows. Section 2 introduces the elements and operations of the algebra. Section 3 formulates geometric entities and their relations and transforms. Section 4 provides several frequently-used specific GAs. Finally, Sect. 5 gives some concluding remarks.

2 Formalizing Algebraic Elements and Operations

In the GA theory, the quadratic forms with arbitrary signatures are generally written as $\mathcal{G}_{p,q,r}$, whose detailed introduction can be found in [27–29]. $\mathcal{G}_{p,q,r}$ is generated by a real linear space $\mathbf{V}$ of vectors that have scalar products of type (p, q, r). This means that $\mathbf{V}$ has basis vectors $\mathbf{e}_{\{1\}}, \ldots, \mathbf{e}_{\{p+q+r\}}$ with $\mathbf{e}_{\{i\}} \cdot \mathbf{e}_{\{i\}}$ being 1 for $1 \leq i \leq p$, being -1 for $p < i \leq p+q$ and being 0 for $p+q < i \leq p+q+r$. If $i \neq j$, we have $\mathbf{e}_{\{i\}} \cdot \mathbf{e}_{\{j\}} = 0$. Thus, $\mathcal{G}_{p,q,r}$ is a real algebra that contains a copy of $\mathbb{R} = span\{\mathbf{e}_{\{\}}\}$ and a copy of $\mathbf{V} = span\{\mathbf{e}_{\{1\}}, \ldots, \mathbf{e}_{\{p+q+r\}}\}$ whose multiplication satisfies $\mathbf{e}_{\{i\}}^2 = \mathbf{e}_{\{i\}} \cdot \mathbf{e}_{\{i\}}$ and $\mathbf{e}_{\{i\}}\mathbf{e}_{\{j\}} = -\mathbf{e}_{\{j\}}\mathbf{e}_{\{i\}}$. The elements $\mathbf{e}_{\{i_1,i_2,\ldots,i_k\}} = \mathbf{e}_{\{i_1\}}\mathbf{e}_{\{i_2\}} \cdots \mathbf{e}_{\{i_k\}}$ with $i_1 < i_2 < \cdots < i_k$ form a basis of $\mathcal{G}_{p,q,r}$. So, $\mathcal{G}_{p,q,r}$ is a linear space that has dimension 2^{p+q+r}. In geometrical applications, we are usually concerned with the dimension of the space $\mathbf{V}$. Due to this reason, $\mathcal{G}_{p,q,r}$ is called as a $(p+q+r)$-D GA. In this section, we present the formalization of the algebraic elements and operations of a general GA space $\mathcal{G}_{p,q,r}$.

2.1 Multivectors

The GA theory is established in the multivector space. Its elements are multivectors, which have the form of weighted sums of basis blades. For example, a multivector in 3D GA can be represented as

$$\mathbf{x} = x_1\mathbf{e}_{\{\}} + x_2\mathbf{e}_{\{1\}} + x_3\mathbf{e}_{\{2\}} + x_4\mathbf{e}_{\{3\}} + x_5\mathbf{e}_{\{1,2\}} + x_6\mathbf{e}_{\{2,3\}} + x_7\mathbf{e}_{\{1,3\}} + x_8\mathbf{e}_{\{1,2,3\}} \tag{1}$$

where $\mathbf{e}_{\{\}}$ is the scalar basis, i.e. 1. $\mathbf{e}_{\{1\}}$, $\mathbf{e}_{\{2\}}$ and $\mathbf{e}_{\{3\}}$ are the 3D orthogonal basis vectors. $\mathbf{e}_{\{1,2\}}$, $\mathbf{e}_{\{1,3\}}$ and $\mathbf{e}_{\{2,3\}}$ are the basis bivectors. $\mathbf{e}_{\{1,2,3\}}$ is the basis trivector. $x_1, x_2, \ldots, x_8$ are weights. Using the set index, Eq. (1) can be rewritten as

$$\begin{aligned}\mathbf{x} = {} & x_{\{\}}\mathbf{e}_{\{\}} + x_{\{1\}}\mathbf{e}_{\{1\}} + x_{\{2\}}\mathbf{e}_{\{2\}} + x_{\{3\}}\mathbf{e}_{\{3\}} + x_{\{1,2\}}\mathbf{e}_{\{1,2\}} \\ & + x_{\{2,3\}}\mathbf{e}_{\{2,3\}} + x_{\{1,3\}}\mathbf{e}_{\{1,3\}} + x_{\{1,2,3\}}\mathbf{e}_{\{1,2,3\}}\end{aligned} \tag{2}$$

It can be observed from Eqs. (1) and (2) that a multivector is a vector in structure. That is to say, a real GA has an underlying structure of real vector space. Therefore, the existing formalization for vector spaces in HOL Light can be employed as a starting point to formalize multivectors.

2.1.1 Data Type of Multivector

In the existing vector theory of HOL Light, the type of the N-dimensional vector space $\mathbb{R}^N$ is formalized as `real^N`, where N denotes a type that is isomorphic to set $\{1, 2, \ldots, N\}$ [30]. Here, `real^N` is in bijection for the function space $N \to \mathbb{R}$. Note that the index type N is finite. Therefore, a multivector space $\mathbb{R}^{2^N}$ is isomorphic to a real vector space of dimension 2^N. The power set of $\{1, 2, \ldots, N\}$ ($\{s|s \subseteq \{1, 2, \ldots, N\}\}$, which has the dimension 2^N), is used to define a multivector type, which is denoted as `multivector`. Accordingly, the type of an N-dimensional multivector is written as `(N)multivector`, which is constructed by building the following isomorphism

$$\texttt{(N)multivector} \leftrightarrow \{s|s \subseteq \{1, 2, \ldots, N\}\} \tag{3}$$

Through this isomorphism, the multivector type had been established for a GA $\mathcal{G}_N$ that has positive definite signature in Harrison's work [26].

It would be possible to implement $\mathcal{G}_{p,q,r}$ as $\mathcal{G}_{p+3q+4r}$, a subalgebra of $\mathcal{G}_N$, via $\mathbf{e}_{\{i\}} \mapsto \mathbf{e}_{\{i\}}$ for $1 \le i \le p$, $\mathbf{e}_{\{p+i\}} \mapsto \mathbf{e}_{\{p+r+3i-2\}}\mathbf{e}_{\{p+r+3i-1\}}\mathbf{e}_{\{p+r+3i\}}$ for $1 \le i \le q$, and $\mathbf{e}_{\{p+q+i\}} \mapsto \mathbf{e}_{\{p+i\}} + \mathbf{e}_{\{p+3q+4i-2\}}\mathbf{e}_{\{p+3q+4i-1\}}\mathbf{e}_{\{p+3q+4i\}}$ for $1 \le i \le r$. The fact that under the mapping above the geometric product has the properties in the beginning of Sect. 2 can be formalized as follows, which we have proven using the existing library from [26].

```
|- !p q r i e.
   1 <= p + q + r /\ p + 3 * q + 4 * r <= dimindex(:N) /\
   (e i =
    if 1 <= i /\ i <= p then (mbasis{i}:real^(N)multivector)
   else if p + 1 <= i /\ i <= p + q then
    (mbasis{(3*i-2*p+r)-2}* mbasis{(3*i-2*p+r)-1}* mbasis{3*i-2*p+r})
   else if p + q + 1 <= i /\ i <= p + q + r then
    (mbasis{i-q}+
       mbasis{(4*i-3*p-q)-2}* mbasis{(4*i-3*p-q)-1}* mbasis{4*i-3*p-q})
   else vec 0) ==>
   e i * e i = if 1 <= i /\ i <= p then mbasis{}
               else if p + 1 <= i /\ i <= p + q then --mbasis{}
               else vec 0
```

However, when we use this embedding method to implement a multivector, its grade will be changed. For example, $\mathbf{e}_{\{5\}}$ in $\mathcal{G}_{4,1,0}$ is a vector, i.e., its grade is 1. If we use the embedding method, it will be denoted as $\mathbf{e}_{\{5\}}\mathbf{e}_{\{6\}}\mathbf{e}_{\{7\}}$, whose grade is 3. Therefore, the embedding method is unfavorable for uniformly formalizing the grade.

In $\mathcal{G}_{p,q,r}$, it is possible for some of subspaces to be null space (For example, when $\mathcal{G}_{p,q}$ is used in GA, it is actually an abbreviation of $\mathcal{G}_{p,q,0}$). However, according to the existing definitions of finite Cartesian product types [31], the dimension of a subspace is written as `dimindex`, which is a natural number larger than zero. Hence, a more expressive definition is needed to accommodate for the representation dimension of the null space. For changes to the overall framework of vectors in HOL Light to be minimal, we define the pseudo-dimension function

```
|- !s. pdimindex s = dimindex s - 1,
```

i.e. by using Cartesian power of dimension $n + 1$ for spaces of pseudo-dimension n. We then have

```
|- (:A) HAS_SIZE n + 1 ==> pdimindex (:A) = n.
```

For the intuitiveness of the symbolism, we construct a function called `define_pseudo_finite_type`. We introduce the notation `'n` and define `(:'n)` to be the new type for vector spaces of pseudo-dimension `n`. For example, the command

```
define_pseudo_finite_type 0;;
```

give the theorem

```
|- (:'0) HAS_SIZE 0 + 1
```

and we immediately have

```
|- pdimindex (:'0) = 0.
```

Note that the vector `x:real^(N)` has dimension `dimindex(N)` and pseudo-dimension `pdimindex(N)`. In this paper, we only use `pdimindex` to represent the pseudo-dimension of a subspace that may be the null space. For a vector space, we still use `dimindex` to represent its dimension because its dimension is greater than or equal to 1.

With the series of definitions above, the dimension of the null space can be properly represented, while ensuring the compatibility with the existing framework for vectors in HOL Light.

In order to distinguish the arbitrary signatures of the quadratic forms in $\mathcal{G}_{p,q,r}$, three type variables, P, Q and R, are used to construct their vector spaces. P, Q and R are grouped into a finite triple type `:(P,Q,R)trip_fin_sum` through the following isomorphic relation

```
(P,Q,R)trip_fin_sum ↔
     1..(if 1<=pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)
         then pdimindex(:P)+pdimindex(:Q)+pdimindex(:R) else 1)
```
(4)

In the above, when the condition `1<=pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)` holds, we represent the dimension of `:(P,Q,R)trip_fin_sum` using `dimindex(:(P,Q,R)trip_fin_sum)`. This representation is chosen to facilitate the reuse of the existing libraries of HOL Light. When the condition `pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)=0` holds, `:('0,'0,'0)trip_fin_sum` can be used to represent a null space. We have `dimindex(:('0,'0,'0)trip_fin_sum)=1`, and `pdimindex(:('0,'0,'0)trip_fin_sum)=0`. Correspondingly, we say that the pseudo-dimension of `:('0,'0,'0)trip_fin_sum` is 0.

In this way, the real vector type $\mathbb{R}^{p,q,r}$ is formalized as `:real^(P,Q,R)trip_fin_sum`. Because $\mathcal{G}_{p,q,r}$ is the GA over $\mathbb{R}^{p,q,r}$, the multivector type in a $\mathcal{G}_{p,q,r}$ is denoted as `:real^(P,Q,R)geomalg`. Similarly, we have the following isomorphic relation

```
(P,Q,R)geomalg ↔ {s|s SUBSET 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)}
```
(5)

Between the types `:(P,Q,R)trip_fin_sum` and `:real^(P,Q,R)geomalg`, a function `multivec` is defined to transform a vector into a multivector as follow.

```
|- (multivec:real^(P,Q,R)trip_fin_sum->real^(P,Q,R)geomalg) x =
 vsum(1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R))(\i. x$i % mbasis{i})
```

where `vsum` is the summation function. The basis of multivectors `mbasis` is defined as

```
|- mbasis s = lambdas t. if s = t then &1 else &0
```

where `&` is the operator that converts the natural number type (`:num`) into the real number type (`:real`).

The equivalence property about the transformational function `multivec` is formalized as follows.

```
!x y:real^(P, Q, R)trip_fin_sum.
     1 <= pdimindex (:P) + pdimindex (:Q) + pdimindex (:R)
     ==> (x = y <=> multivec x = multivec y)
```

2.1.2 Index of Multivectors

It follows from the type `:real^(P,Q,R)geomalg` that $\{s|s \subseteq \{1, 2, \ldots, p+q+r\}\}$ is isomorphic to $\{1, 2, \ldots, 2^{p+q+r}\}$. Therefore, multivectors also are vectors, and theoretically speaking, they can be indexed by natural numbers. However, the widely used indexing method for multivectors is based on sets. According to Harrison [26], a one-to-one mapping relation between the natural number index and the set index can be constructed using the two functions `codeset` and `setcode`, i.e.,

$$codeset(s) = \sum_{i \in s} 2^{i-1} + 1 \tag{6}$$

$$setcode(n) = \left\{ i + 1 | ODD\left(\frac{n-1}{2^i}\right), i \in \mathbb{N} \right\} \tag{7}$$

Overloading the operators defined by Harrison, the binders of the natural number index `m$i` and the set index `m$$s` are `lambda` and `lambdas`, i.e.,

```
|- (lambda i. m$i) = m
|- (lambdas s. m$$s) = m
```

In addition, Table 3 also shows the mapping relations in a more direct way.

It should be noted that the set indexes introduced here are more convenient than the conventional natural number indexes for the set operations in higher order logic. Therefore, the set index method proposed by Harrison is applied in our formalization of the $\mathcal{G}_{p,q,r}$ theory.

2.1.3 Basic Operations of Multivectors

Since multivectors have two different indexing operators `$` and `$$`, it is inherited from the vector theory that the addition (+), negation (−), subtraction (-) and scalar multiplication (`%`) of multivectors can be computed componentwise using the following relations as

Table 3 The mapping relations between the two kinds of indexes

Basis blade	Number index	Bitmap	Set index
$1 = \mathbf{e}_{\{\}}$	1	000	{ }
$\mathbf{e}_{\{1\}}$	2	001	{1}
$\mathbf{e}_{\{2\}}$	3	010	{2}
$\mathbf{e}_{\{1\}} \wedge \mathbf{e}_{\{2\}} = \mathbf{e}_{\{1,2\}}$	4	011	{1,2}
$\mathbf{e}_{\{3\}}$	5	100	{3}
$\mathbf{e}_{\{1\}} \wedge \mathbf{e}_{\{3\}} = \mathbf{e}_{\{1,3\}}$	6	101	{1,3}
$\mathbf{e}_{\{2\}} \wedge \mathbf{e}_{\{3\}} = \mathbf{e}_{\{2,3\}}$	7	110	{2,3}
$\mathbf{e}_{\{1\}} \wedge \mathbf{e}_{\{2\}} \wedge \mathbf{e}_{\{3\}} = \mathbf{e}_{\{1,2,3\}}$	8	111	{1,2,3}

```
|- !x y:real^(P,Q,R)geomalg. x + y = lambdas s. x $$ s + y $$ s
|- !x:real^(P,Q,R)geomalg. -- x = lambdas s. -- x $$ s
|- !x y:real^(P,Q,R)geomalg. x - y = lambdas s. x $$ s - y $$ s
|- !x:real^(P,Q,R)geomalg c:real. c % x = lambdas s. c % x $$ s
```

2.1.4 Expansions of Multivectors

In general, one can expand multivectors according to their bases. For an arbitrary multivector $\mathbf{x}$, its expansion as $\mathbf{x} = \sum_{s \subseteq \{1,\ldots,p+q+r\}} x_s \mathbf{e}_s$ has the following formalization

```
|- !x:real^(P,Q,R)geomalg.
       vsum {s | s SUBSET 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)}
            (\s. x$$s % mbasis s) = x
```

Note that in the formalization of multivectors, we are free to fix an orthonormal base arbitrarily because all the constructions in GA are known to be invariant by orthogonal transformations. The invariance properties of the orthogonal transformations have been formalized in the theories on vectors and determinants in HOL Light [32,33]. Multivectors can inherit the properties of vectors therein.

2.2 The Product Operations of $\mathcal{G}_{p,q,r}$

There are three types of basic product operations in GA: *outer product*, *geometric product* and *inner product*. Generally, there are four different kinds of inner products: general inner product, left contraction product, right contraction product and scalar product [34]. The classical notations and HOL notations of all these products are compared in Table 4. In this section, we present a unified form for them. For this purpose, let's consider two arbitrary multivectors $\mathbf{x} = \sum_{s \subseteq \{1,\ldots,p+q+r\}} x_s \mathbf{e}_s$ and $\mathbf{y} = \sum_{t \subseteq \{1,\ldots,p+q+r\}} y_t \mathbf{e}_t$.

By using such linear decompositions of $\mathbf{x}$ and $\mathbf{y}$, the abovementioned products about them can be uniformly formulated as

$$\sum_{s \subseteq \{1,\ldots,p+q+r\}} \sum_{t \subseteq \{1,\ldots,p+q+r\}} sgn(s,t)(x_s y_t)\mathbf{e}_{(s-t)\cup(t-s)} \tag{8}$$

Table 4 The classical and HOL notations of products

Classical notation	HOL notation	Operator
$x \wedge y$	`x outer y`	Outer product
xy	`x * y`	Geometric product
$x \cdot y$	`x inner y`	General inner product
$x \rfloor y$	`x lcinner y`	Left contraction product
$x \lfloor y$	`x rcinner y`	Right contraction product
$x \star y$	`x scalar y`	Scalar product

where sgn is a function of the two subscript sets s and t and it decides the sign of the corresponding monomial, $x_s y_t$ is its coefficient, and $\mathbf{e}_{(s-t)\cup(t-s)}$ is its basis blade.

This uniform formulation has the following formalization

```
|- Product sgn x y:real^(P,Q,R)geomalg =
        vsum {s | s SUBSET 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)}
        (\s. vsum
             {s | s SUBSET 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)}
               (\t. (sgn s t * x$$s * y$$t) % mbasis(s SYMDIFF t)))
```

where the function `sgn` decides the kind of product.

Specifically, for the *outer product*, the function `sgn` is defined by

$$sgn(s,t) = \begin{cases} 0, & s \cap t \neq \emptyset \\ (-1)^n, & otherwise \end{cases}$$

where n is the number of reverse order pairs according to the Grassmann algebra [35], i.e., $n = |\{(i,j)|i,j \in \{1,\ldots,p+q+r\} \wedge i \in s \wedge j \in t \wedge i > j\}|$. In this case, it is simply the sign of the associated permutation. Therefore, the *outer product* is formalized as

```
|- x outer y:real^(P,Q,R)geomalg =
 Product (\s t. if ~(DISJOINT s t) then &0
  else -- &1 pow CARD {i,j | i IN 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R) /\
                             j IN 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R) /\
                             i IN s /\ j IN t /\ i > j})
        x y
```

where `pow` represents the power operation.

For the geometric product, the function `sgn` depends on the signatures of quadratic forms of $\mathcal{G}_{p,q,r}$, i.e.,

$$sgn(s,t) = (-1)^n(-1)^{m_1}0^{m_2}$$

where $m_1 = |\{p+1,\ldots,p+q\} \cap s \cap t|$ and $m_2 = |\{p+q+1,\ldots,p+q+r\} \cap s \cap t|$. Thus, the geometric product is formalized as

```
|- x * y:real^(P,Q,R)geomalg =
  Product (\s t.
    --(&1) pow CARD{i,j | i IN 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R) /\
                          j IN 1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R) /\
                          i IN s /\ j IN t /\ i > j} *
    --(&1) pow CARD((pdimindex(:P)+1..pdimindex(:P)+pdimindex(:Q))INTER s INTER t) *
       &0 pow CARD((pdimindex(:P)+pdimindex(:Q)+1..
                   pdimindex(:P)+pdimindex(:Q)+pdimindex(:R))INTER s INTER t))
       x y
```

Table 5 Four kinds of inner product operators and their sgn functions

Operator	Function `sgn`
General inner product	$sgn(s,t) = \begin{cases} 0, & s = \emptyset \vee t = \emptyset \vee \neg(s \subset t) \vee \neg(t \subset s) \\ (-1)^n(-1)^{m_1}0^{m_2}, & otherwise \end{cases}$
Left contraction product	$sgn(s,t) = \begin{cases} 0, & s = \emptyset \vee t = \emptyset \vee \neg(s \subset t) \\ (-1)^n(-1)^{m_1}0^{m_2}, & otherwise \end{cases}$
Right contraction product	$sgn(s,t) = \begin{cases} 0, & s = \emptyset \vee t = \emptyset \vee \neg(t \subset s) \\ (-1)^n(-1)^{m_1}0^{m_2}, & otherwise \end{cases}$
Scalar product	$sgn(s,t) = \begin{cases} 0, & s = \emptyset \vee t = \emptyset \vee \neg(s = t) \\ (-1)^n(-1)^{m_1}0^{m_2}, & otherwise \end{cases}$

The general *inner product*, the *left contraction product*, *right contraction product* and *scalar product* can be easily formalized under the present framework. The main difference lies in the definitions of the function `sgn` for these operations, as shown in Table 5.

2.3 The Invertibility of Geometric Product

According to the definition of invertibility, an operation is invertible, if it has equal left and right inverses. In this section, a predicate is defined as follows, which can be used to judge whether a multivector is invertible or not in terms of the geometric product

```
|- mvinvertible (x:real^(P,Q,R)geomalg) <=>
    (?x'. x' * x = mbasis{} /\ x * x' = mbasis{})
```

where `mbasis{}` denotes scalar 1.

The inverse of `x`, if exists, is also a multivector, whose formal definition is

```
|- !x. mvinverse x =(@x'. x' * x = mbasis{} /\ x * x' = mbasis{})
```

where "`@`" represents Hilbert's choice operator. `@x. P x` denotes an `x` such that `P x`. So, it can be inferred that

```
|- !x:real^(P,Q,R)geomalg.
    mvinvertible x <=> mvinverse x * x = mbasis{} /\ x * mvinverse x = mbasis{}
```

Based on the definition of inverse multivectors, the formal expression of geometric division are obtained as below

```
|- x / y:real^(P,Q,R)geomalg = x * mvinverse y
```

In $\mathcal{G}_{p,q,r}$, a multivector will have a left-inverse if and only if it has a right-inverse, and the two inverses are equal. This property can be formalized as

```
|- !x y:real^(P,Q,R)geomalg. x * y = mbasis{} <=> y * x = mbasis{}
```

Therefore, a multivector is invertible, if and only if it is left or right invertible.

```
|- mvinvertible (x:real^(P,Q,R)geomalg) <=> (?x'. x' * x = mbasis{})
|- mvinvertible (x:real^(P,Q,R)geomalg) <=> (?x'. x * x' = mbasis{})
```

For the degenerate case of a vector, the condition for its invertibility can be further refined. A vector is invertible if and only if it is not null, i.e.,

```
|- !x:real^(P,Q,R)trip_fin_sum.
       ~(is_null (multivec x)) <=> mvinvertible (multivec x)
```

where `is_null` is the predicate that judges whether a multivector is null or not (that is applied to vectors in the above). Its definition is formalized as follows

```
|- is_null (x:real^(P,Q,R)geomalg) <=> x inner x = vec 0
```

Note that '`is_null v`' is different from '`v = vec 0`', '`v = vec 0 ==> is_null v`' holds, but the opposite direction does not hold. The equivalence of '`is_null v`' and '`v = vec 0`' holds when the signature is definite (i.e. positive definite or negative definite).

2.4 The Operations Properties and Automatic Proof Procedures of $\mathcal{G}_{p,q,r}$

There are many properties for the operations in GA. Generally, it is annoying, tedious and time-consuming to prove them in the interactive way. A procedure is developed here to automatically perform the proving. For this purpose, the three basic properties, including bilinearity, associativity and outermorphism, are proved at first as follows.

2.4.1 Bilinearity

The *outer product*, *inner product* and *geometric product* of $\mathcal{G}_{p,q,r}$ simultaneously have the property of bilinearity. Based on the judging theorem of bilinearity in HOL Light, it is easy to prove that the unified product form proposed in Sect. 2.2 also has the property of bilinearity, i.e.,

```
|- !sgn. bilinear(Product sgn)
```

Therefore, it can be inferred that all these products satisfy the requirement of bilinearity. In addition, many other theorems can be derived from the bilinearity theorem. For example,

```
|- !x y z. Product sgn (x+y) z = (Product sgn x z)+(Product sgn y z)
|- !x y z. Product sgn x (y+z) = (Product sgn x y)+(Product sgn x z)
```

2.4.2 Associativity

Each monomial in Eq. (8) consists of three parts: the function `sgn`, a real number product and a basis blade. Since the real product satisfies the associativity surely, the symmetry difference of sets satisfies the associativity too.

```
|- !s t u. s SYMDIFF (t SYMDIFF u) = (s SYMDIFF t) SYMDIFF u
```

The associativity of the product operations only depends on the associativity of the function `sgn`. Generally, only the same operations have the associativity law. However, the associativity law may also be applicable among the different kinds of products of GA under a certain condition. For example, $s \cap u = \emptyset \Rightarrow e_s(e_t \wedge e_u) = (e_s e_t) \wedge e_u$ and $s \cap u = \emptyset \Rightarrow (e_s \wedge e_t)e_u = e_s \wedge (e_t e_u)$.

Here, two functions `sgn1` and `sgn2` are adopted to formalize the judgement theorem of the associativity between different products.

```
|- !sgn1 sgn2.
       (!s t u. sgn1 t u * sgn2 s (t SYMDIFF u) =
            sgn2 s t * sgn1 (s SYMDIFF t) u)
       ==> (!x y z. Product sgn2 x (Product sgn1 y z) =
              Product sgn1 (Product sgn2 x y) z)
```

When `sgn1` and `sgn2` are identical, this theorem can be used to judge the associativity of the same product. In this way, it can be verified that both the geometric product and outer product have the property of associativity. When `sgn1` and `sgn2` are different, the associativity between different products can be judged, for example, we can get the following theorems.

```
|-!s t u. DISJOINT s u ==>
       (mbasis s):real^(P,Q,R)geomalg * (mbasis t outer mbasis u) =
                 (mbasis s * mbasis t) outer mbasis u
|-!s t u. DISJOINT s u ==>
        (mbasis s outer mbasis t) * (mbasis u):real^(P,Q,R)geomalg =
                  mbasis s outer (mbasis t * mbasis u)
```

2.4.3 Outermorphism

The mathematical expression of the outermorphism of k blade is

$$\underline{f}(\mathbf{v}_1 \wedge \mathbf{v}_2 \wedge \cdots \wedge \mathbf{v}_k) = f(\mathbf{v}_1) \wedge f(\mathbf{v}_2) \wedge \cdots \wedge f(\mathbf{v}_k) \tag{9}$$

where $\underline{f}$ is the outermorphism of function f.

According to the property of outermorphism, each basis of a multivector can be expanded as an ordered outer product of all basis vectors. The outermorphic expansion of multivector `x` is formalized as

```
|- outermorphism (f:real^(P,Q,R)geomalg->real^N)
     (x:real^(P,Q,R)geomalg) =
        vsum {s | s SUBSET 1..(pdimindex(:P)+pdimindex(:Q)+pdimindex(:R))}
            (\s. x$$s % seqiterate(outer) s (multivec o f o basis))
```

where `seqiterate` denotes the ordered iteration, `basis` represents the basis vector in the space, and `o` denotes the combine operator of the functions.

2.4.4 An Automatic Evaluation Procedure

In practice, it is usually necessary to determine the results of the three product operations of multivectors. Under this condition, the automatic evaluation in the theorem prover is very useful for the users. Harrison implemented the conversion of automatic evaluation of *outer product*s through the `OUTER_CANON_CONV` [26]. Inspired by Harrison's work, we develop a conversion of automatic evaluation of the mixed operation about geometric, outer and inner products.

The *outer* and *inner product*s can be easily expressed by *geometric product*, their relations can be formalized as

```
MBASIS_OUTER_GEOM:
|- !s t. (mbasis s) outer (mbasis t):real^(P,Q,R)geomalg =
    if DISJOINT s t then mbasis s * mbasis t else vec 0
```

```
MBASIS_INNER_GEOM:
|- !s t. (mbasis s) inner (mbasis t):real^(P,Q,R)geomalg =
    if ~(s = {}) /\ ~(t = {}) /\ (s SUBSET t \/ t SUBSET s) then
     mbasis s * mbasis t else vec 0
```

We develop a conversion, GA_GEOM_CONV, to convert mixed product form to canonical form of geometric products via rewriting the two abovementioned theorems, and simplifying logical and set operations.

```
let GA_GEOM_CONV =
  PURE_REWRITE_CONV[MBASIS_OUTER_GEOM; MBASIS_INNER_GEOM] THENC
  SIMP_CONV[EXTENSION; IN_INTER; IN_INSERT; SUBSET; NOT_IN_EMPTY;
            NOT_FORALL_THM; GSYM NOT_EXISTS_THM; EXISTS_REFL;
            ARITH_RULE `!a b x:num. ~(a=b) ==> ~(x=a /\ x=b)`;
            EXISTS_OR_THM; ARITH_EQ];;
```

For example:

```
# GA_GEOM_CONV `mbasis{1} *(mbasis{1} inner mbasis{1,3} +
                       mbasis{1} outer mbasis{2}:real^('4,'1,'1)geomalg)`;;
val it : thm =
|- mbasis{1} *(mbasis{1} inner mbasis{1,3} + mbasis{1} outer mbasis{2}) =
     mbasis{1} *(mbasis{1} * mbasis{1,3} + mbasis{1} * mbasis{2})
```

Therefore, if a conversion of automatic evaluation of *geometric product*s is implemented, all the operations of the three products can be automatically evaluated.

The automatic evaluation procedure for geometric products includes the following steps:

- Step 1: Split each basis blade of the multivectors as the geometric product of ordered single basis vectors. For example, decompose `` `mbasis{1,2}` `` as `` `mbasis{1} * mbasis{2}` ``. Here, the decomposition is based on the theorem

  ```
  MVBASIS_SPLIT_GEOM:
  |- !a s. (!x. x IN s ==> a < x)
                  ==> mbasis (a INSERT s) = mbasis{a} * mbasis s
  ```

- Step 2: Rewrite the bilinearity property of the geometric operations and expand the expressions as the summation of polynomials. For example, expand `` `mbasis{1} * (mbasis{2} + mbasis{3})` `` as `` `mbasis{1} * mbasis{2} + mbasis{1} * mbasis{3}` ``.
- Step 3: Evaluate the geometric product of a single basis vector based on the following related basic properties:

  ```
  |- (!x y z:real^(P,Q,R)geomalg. (x * y) * z = x * (y * z)) /\
     (!i j. i > j ==> mbasis{i}:real^(P,Q,R)geomalg * mbasis{j} =
            --(&1) % (mbasis{j} * mbasis{i})) /\
     (!i j x:real^(P,Q,R)geomalg. i > j ==>
         mbasis{i} * mbasis{j} * x = --(&1) % (mbasis{j} * mbasis{i} * x)) /\
     (!i. mbasis{i}:real^(P,Q,R)geomalg * mbasis{i} =
            if i IN 1..pdimindex(:P) then mbasis{}
             else if i IN pdimindex(:P)+1..pdimindex(:P)+pdimindex(:Q) then
                --(&1) % mbasis{} else vec 0) /\
     (!i x:real^(P,Q,R)geomalg. mbasis{i} * mbasis{i} * x =
           if i IN 1..pdimindex(:P) then x
  ```

```
              else if i IN pdimindex(:P)+1..pdimindex(:P)+pdimindex(:Q) then
                 --(&1) % x else vec 0) /\
      (!x:real^(P,Q,R)geomalg. mbasis{} * x = x) /\
      (!x:real^(P,Q,R)geomalg. x * mbasis{} = x) /\
      (!p x y z:real^(P,Q,R)geomalg. x * (if p then y else z) =
         if p then x * y else x * z)
```

– Step 4: Obtain the pseudo-dimension through rewriting with the theorems on pseudo-dimensionality. Then, calculate the natural numbers by invoking the `NUM_REDUCE_CONV` in HOL Light.
– Step 5: Merge geometric products of single basis vectors as basis blades. This step is the inverse process of step 1 and it is also based on the theorem `MBASIS_SPLIT_GEOM` too.
– Step 6: Arrange all monomials in the multivectors according to the order of the `setcode` of the corresponding basis. For example, we can convert
  `` `&2 % mbasis{1,2} + &3 % mbasis{2} + &2 % mbasis{1,3} + -- (&3) % mbasis{1}` ``
  as
  `` `&3 % mbasis{2} + -- (&3) % mbasis{1} + &2 % mbasis{1,2} + &2 % mbasis{1,3}` ``
  This step can be implemented by adopting the `MBASIS_GROUP_CONV` of Harrison's formalization.

This conversion is named as `GEOM_CANON_CONV`. For example, we run the following command

```
GEOM_CANON_CONV `(mbasis{2,3}:real^('3,'0,'1)geomalg) * mbasis{1,2,3,4}`;;
```

and then a new theorem is obtained

```
|- mbasis{2,3} * mbasis{1,2,3,4} = -- &1 % mbasis{1,4}
```

Using `GA_GEOM_CONV` and `GEOM_CANON_CONV`, we can figure out mixed product operations. For example:

```
# (GA_GEOM_CONV THENC GEOM_CANON_CONV)
  `mbasis{1} *(mbasis{1} inner mbasis{1,3} +
       mbasis{1} outer mbasis{2}:real^('4,'1,'1)geomalg)`;;
val it : thm =
  |- mbasis{1} *(mbasis{1} inner mbasis{1,3} + mbasis{1} outer mbasis{2}) =
     &1 % mbasis{2} + &1 % mbasis{1,3}
```

Using `GEOM_CANON_CONV`, an automatic proof procedure `GEOM_ARITH` is constructed

```
let GEOM_ARITH tm =
      let l,r = dest_eq tm in
      let th,th' = GEOM_CANON_CONV l, GEOM_CANON_CONV r in
      TRANS th (SYM th');;
```

This procedure can be used to prove the properties of some operations that involve the geometric product. It will be applied in the automatically proving the properties of some specific GAs in the present work.

3 Formalizing Geometric Elements, Relations and Transforms

Based on the formalization of the algebra part in Sect. 2, this section proposes that of the geometric part, including geometric elements and transformations.

3.1 Blade and Versor

Vectors are the most fundamental geometric elements. Spanned by the outer products of linearly independent vectors, blades represent geometric entities, including points, lines and planes. Yielded by the geometric products of non-null vectors, versors formulate geometric transforms such as reflections and rotations.

As is known, k linearly independent vectors can span as a k-blade. So, the following formal definition is proposed to judge whether a multivector is a k-blade or not.

```
|- k blade (A:real^(P,Q,R)geomalg) <=>
     (?a.
       independent {a i | i IN 1..k} /\
       {a i | i IN 1..k} HAS_SIZE k /\
       A = seqiterate (outer) (1..k) (multivec o a))`;;
```

where `independent` is a formal predicate for the judgment whether a vectors set is linear independent or not, `s HAS_SIZE k` means that the size of set s is k.

The predicate that judges whether or not a multivector is a blade can be formally defined as

```
|- is_blade (A:real^(P,Q,R)geomalg) <=> ?k. k blade A
```

In $\mathcal{G}_{p,q,r}$, the top-grade unit blade is the unit pseudoscalar $\mathbf{I}$, which is defined formally as

```
|- pseudoscalar:real^(P,Q,R)geomalg =
        mbasis(1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R))
```

Similarly, the geometric product of several non-null vectors yields a versor. So, whether a multivector is a versor or not can be judged by using the following formal definition

```
|- is_versor A:real^(P,Q,R)geomalg <=>
   ?k a.
     (!i. i IN 1..k ==> ~(is_null(multivec (a i)))) /\
     (A = seqiterate(*)(1..k)(multivec o a))
```

Because all non-null vectors are invertible, all versors are invertible. This is formalized as

```
|- !x:real^(P,Q,R)geomalg. is_versor x ==> invertible x
```

3.2 Geometric Relations

3.2.1 Linear Dependency

A set of mutually distinct vectors are linearly dependent if and only if their outer product is zero. For a specific vector set, its linear dependency can be easily decided using the following conversion.

```
let OUTER_VECTOR_CONV =
  REWRITE_CONV[REWRITE_RULE[LINEAR_MULTIVEC]
  (ISPEC `multivec:real^(P,Q,R)geomalg->real^(P,Q,R)geomalg` linear);
   MULTIVEC_BASIS; VECTOR_SUB; VECTOR_NEG_MINUS1] THENC OUTER_CANON_CONV;;
```

where `OUTERGA_CANON_CONV` is the conversion of automatic evaluation of outer products, and the theorems `LINEAR_MULTIVEC` and `MULTIVEC_BASIS` are

```
|- linear multivec
|- !i. multivec(basis i) = mbasis{i}
```

For example, running the command

```
OUTER_VECTOR_CONV
`(multivec(basis 1 + basis 2)) outer (multivec(basis 2 + basis 3)) outer
  (multivec(basis 1 - basis 3))`;;
```

we get the following result

```
val it : thm =
  |- multivec (basis 1 + basis 2) outer
     multivec (basis 2 + basis 3) outer
     multivec (basis 1 - basis 3) =
     &0 % mbasis {1, 2, 3}
```

From this result, it is easy to decide that $\mathbf{e}_1 + \mathbf{e}_2$, $\mathbf{e}_2 + \mathbf{e}_3$ and $\mathbf{e}_1 - \mathbf{e}_3$ are linearly dependent.

This is essentially the same thing as computing the determinant. For example,

```
OUTERGA_VECTOR_CONV
`(multivec (basis 1 + basis 2)) outer (multivec (basis 2 + basis 3)) outer
  (multivec (basis 1 + basis 3))`;;
```

```
|- multivec (basis 1 + basis 2) outer
   multivec (basis 2 + basis 3) outer
   multivec (basis 1 + basis 3) =
   &2 % mbasis {1, 2, 3}
```

The result indicates that the determinant $\begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix}$ is evaluated to 2.

3.2.2 Reversion and Conjugation

In GA, if a blade $\mathbf{A}_k$ is spanned by k vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$, i.e. $\mathbf{A}_k = \mathbf{v}_1 \wedge \mathbf{v}_2 \wedge \cdots \wedge \mathbf{v}_k$, its reversion is $\widetilde{\mathbf{A}_k} = \mathbf{v}_k \wedge \mathbf{v}_{k-1} \wedge \cdots \wedge \mathbf{v}_1$. The reversion of a multivector can be obtained by reversing all the basis blades. For example, $\tilde{\mathbf{A}} = a^s \widetilde{\mathbf{e}_s}$, where $\widetilde{\mathbf{e}_s} = (-1)^{|s|(|s|-1)}\mathbf{e}_s$. The reversion of a multivector can be defined formally as

```
|-(reversion:real^(P,Q,R)geomalg->real^(P,Q,R)geomalg) x =
    lambdas s. --1 pow ((CARD(s) * (CARD(s) - 1)) DIV 2) * x$$s
```

The conjugation of a multivector $\mathbf{A}^\dagger = a^s \mathbf{e}_s^\dagger$, where $\mathbf{e}_s^\dagger = (-1)^{|s \cap \{p+1,\ldots,p+q\}|}\widetilde{\mathbf{e}_s}$. Similarly, the conjugation of a multivector can be formalized as

```
|- (conjugation:real^(P,Q,R)geomalg->real^(P,Q,R)geomalg) x =
    lambdas s. --(&1) pow ((CARD(s) * (CARD(s) - 1)) DIV 2 +
      CARD(s INTER (pdimindex(:P)+1..pdimindex(:P)+pdimindex(:Q))))* x$$s
```

3.2.3 Dual

In non-degenerated GA, the dual of a blade $\mathbf{B}$ is $\mathbf{B}^* = \mathbf{B} \cdot \mathbf{I}^{-1}$, where $\mathbf{I}^{-1}$ is the inverse of the unit pseudoscalar. The dual operation can also be generalized to arbitrary multivectors. In this case, the dual is applied to every basis blade contained in the multivector. In other words, the dual operation is distributive. Such a relation has the following formalization

Table 6 The classical and HOL notations of geometric relations

Relation	Classical notation	HOL notation	Involved operations
Reversion	$\tilde{\mathbf{A}}$	`reversion A`	Outer product
Conjugation	$\mathbf{A}^{\dagger}$	`conjugation A`	Outer product
Dual	$\mathbf{B}^{*}$	`dual B`	Inner product, mvinverse
Horizontal projection	$P_{\|\mathbf{B}}\mathbf{A}$	`project B A`	Outer product, mvinverse product
Vertical projection	$P_{\perp\mathbf{B}}\mathbf{A}$	`reject B A`	Inner product, mvinverse

```
|- (dual:real^(P,Q,'0)geomalg->real^(P,Q,'0)geomalg) x =
      vsum {s | s SUBSET 1..pdimindex(:P) + pdimindex(:Q)}
        (\s.x$$s % mbasis s inner (mvinverse pseudoscalar))
```

3.2.4 Projection

The project operation stands for the projection of a geometric object on another one. For example, the horizontal and vertical projections of geometric object $\mathbf{A}$ on geometric object $\mathbf{B}$ are $P_{\|\mathbf{B}}\mathbf{A} = (\mathbf{A}\wedge\mathbf{B})\mathbf{B}^{-1}$ and $P_{\perp\mathbf{B}}\mathbf{A} = (\mathbf{A}\cdot\mathbf{B})\mathbf{B}^{-1}$. The formalization of these two projections are

```
|- project B A = (A outer B) * (mvinverse B)
|- reject B A = (A inner B) * (mvinverse B)
```

As a summary, the notations and the involved basic operation of the above-mentioned geometric relation are shown in Table 6.

3.3 Geometric Transformations

In the GA theory, the geometric transforms are uniformly written as $\mathbf{y} = \mathbf{V}\mathbf{x}\mathbf{V}^{-1}$, where $\mathbf{V}$ is a versor and $\mathbf{x}$ is a blade which represents a geometric object.

The corresponding formal definition is

```
|- transform v x = v * x * mvinverse v
```

where `transform v` is a rotation operator if `v` is a *rotor*. It represents a rigid-body transform instead if `v` is a *motor*.

The superposition of two transforms can be formalized as

```
|- !x a b.
    is_versor a /\ is_versor b ==>
         a * (b * x * mvinverse b) * mvinverse a =
         (a * b) * x * mvinverse (a * b)
```

The above theorem holds for any invertible `a` and `b` but we only need this result for versors. This can be proved by using the associativity theorem of geometric products and the following theorem

```
|- !a b.
    mvinvertible a /\ mvinvertible b ==>
       mvinverse b * mvinverse a = mvinverse(a * b)
```

where `mvinverse` is an implicit definition based on Hilbert's choice operator of the inverse of multivectors, because it is difficult to directly describe the corresponding explicit form. A versor can always be represented as the geometric product of blades and the inverse of a versor can be transformed into the geometric product of the inverse of blades. The inverse of any non-null blade $\mathbf{x}$ always has the explicit form $\frac{\tilde{\mathbf{x}}}{\tilde{\mathbf{x}}\mathbf{x}}$, which can be formalized as

```
|- !x:real^(P,Q,R)geomalg.
   ~(is_null x) /\ is_blade x ==>
   mvinverse x = inv((reversion x * x) $$ {}) % reversion x
```

where `(reversion x) * x` is a scalar, which can be transformed into a real number by extracting the component corresponding to the empty set, and `inv` is a function that represents the reciprocal of a real number.

For an arbitrary multivector, if $\tilde{\mathbf{x}}\mathbf{x}$ is a non-null scalar, $\mathbf{x}$ will be a non-null blade. This can be formalized as

```
|- !x:real^(P,Q,R)geomalg.
    0 multivectors reversion x * x /\ ~(reversion x * x = vec 0)
     <==> ~( is_null x) /\ is_blade x
```

where `k multivectors x` denotes that the multivector $\mathbf{x}$ is a k-vector. Its formal definition is

```
|- k multivectors (p:real^(P,Q,R)geomalg) <=>
      !s. s SUBSET(1..pdimindex(:P)+pdimindex(:Q)+pdimindex(:R)) /\ ~(p$$s = &0)
          ==> s HAS_SIZE k
```

Thus, we have

```
|- !x:real^(P,Q,R)geomalg.
    0 multivectors reversion x * x /\ ~(reversion x * x = vec 0)
    ==> mvinvertible x
```

The `GEOM_CANON_CONV` can be used to judge whether a specific multivector is invertible or not. If it is invertible, one can invoke the `GEOM_CANON_CONV` to determine its inverse based on the expression $\frac{\tilde{\mathbf{x}}}{\tilde{\mathbf{x}}\mathbf{x}}$.

4 Several Specific GAs

4.1 Gibbs's Vector Algebra

Gibbs's vector algebra can be represented by $\mathcal{G}_{3,0,0}$. The scalar product of two vectors is identical to the *inner product* in $\mathcal{G}_{3,0,0}$. The *cross product* of two vectors $\mathbf{u}$ and $\mathbf{v}$ is the dual of their outer product. Therefore, it can be written in $\mathcal{G}_{3,0,0}$ as $\mathbf{u} \times \mathbf{v} = (\mathbf{u} \wedge \mathbf{v})^*$, which is formalized as

```
|- (u:real^('3,'0,'0)trip_fin_sum) cross_product v =
        dual (multivec u outer multivec v)
```

4.2 Complex Numbers

Complex numbers have been implemented using the type $\mathbb{R}^2$(`:real^2`) in HOL Light by Harrison [36]. The algebra of complex numbers can also be regarded as a GA where the real and imaginary parts of a complex number are interpreted as the two coordinates of a point in a 2D space. We have $\mathbb{C} \cong \mathcal{G}_{0,1,0}$ via the real unit $1 \mapsto \mathbf{e}_{\{\}}$ and the imaginary unit $\mathrm{i} \mapsto \mathbf{e}_{\{1\}}$. Since the product of two complex numbers corresponds to the geometric product, complex numbers can be formulated in 1-D GA $\mathcal{G}_{0,1,0}$. It can be formally proved that

```
|- dimindex(:('0,'1,'0)geomalg) = dimindex(:2)
```

where `real^('0,'1,'0)geomalg` is the specification of the real multivector type `real^(P,Q,R)geomalg`.

Based on this isomorphism, each complex number $a + b\mathrm{i}$ can be represented as a linear combination $a\mathbf{e}_{\{\}} + b\mathbf{e}_{\{1\}}$. The product of two complex numbers corresponds to the geometric product. It is formalized as

```
|- !x y:real^('0,'1,'0)geomalg.
  x * y =
     (x $$ {} * y $$ {} - x $$ {1} * y $$ {1}) % mbasis {} +
         (x $$ {} * y $$ {1} + x $$ {1} * y $$ {}) % mbasis {1}
```

So, the theorem $\mathrm{i}^2 = -1$ can be proved formally by the automatic proof procedure in Sect. 2.4.4.

```
GEOM_ARITH `mbasis{1}:real^('0,'1,'0)geomalg * mbasis{1} = --mbasis{}`;;
```

It follows that

```
|- (mbasis{1}:real^('0,'1,'0)geomalg * mbasis{1} = --mbasis{}
```

Alternatively, for geometric appeal, the complex numbers can be described by $\mathcal{G}^{+}_{2,0,0}$, which is an even subalgebra of $\mathcal{G}_{2,0,0}$. Here, the subspace $\{\mathbf{e}_{\{\}}, \mathbf{e}_{\{1,2\}}\}$ is isomorphic with the ring of complex numbers. Under the isomorphism, each complex number $a + b\mathrm{i}$ can be represented as a linear combination $a\mathbf{e}_{\{\}} + b\mathbf{e}_{\{1,2\}}$ too. Then, the product of two complex numbers can be denoted by the geometric product. So, the theorem $\mathrm{i}^2 = -1$ is proved via

```
GEOM_ARITH `mbasis{1,2}:real^('2,'0,'0)geomalg * mbasis{1,2} = --mbasis{}`;;
```

Because the complex numbers are embedded in GA, the complex theory based analyses and applications can be formalized by GA.

4.3 Quaternions

The quaternions ($\mathbb{H}$) are presented here as consisting of a scalar component and three imaginary components. The imaginary components are typically denoted by i, j and k. Quaternions have been implemented in the type (`:real^4`) in HOL Light by Gabrielli and Maggesi [37].

We have the following theorem stating that the quaternions can be represented in the form of $\mathcal{G}_{0,2,0}$.

```
|- dimindex(:('0,'2,'0)geomalg) = dimindex(:4)
```

The scalar component 1 can be expressed as $\mathbf{e}_{\{\}}$ and the three imaginary components i, j and k can be denoted by $\mathbf{e}_{\{1\}}$, $\mathbf{e}_{\{2\}}$ and $\mathbf{e}_{\{1,2\}}$, respectively. The product of two quaternions corresponds to the geometric product in $\mathcal{G}_{0,2,0}$. This correspondence is formalized as

```
|- !x y:real^('0,'2,'0)geomalg.
  x * y =
  (x$${}*y$${}-x$${1}*y$${1}-x$${2}*y$${2}-x$${3}*y$${3})% mbasis{} +
  (x$${}*y$${1}+x$${1}*y$${}+x$${2}*y$${1,2}-x$${1,2}*y$${2})% mbasis{1} +
  (x$${}*y$${2}-x$${1}*y$${1,2}+x$${2}*y$${}+x$${1,2}*y$${1})% mbasis{2} +
  (x$${}*y$${1,2}+x$${1}*y$${2}-x$${2}*y$${1}+x$${1,2}*y$${})% mbasis{1,2}
```

Moreover, we are able to automatically prove all of the following properties of quaternions with the single decision procedure `GEOM_ARITH` from Sect. 2.4.

$$\begin{aligned} &\mathrm{i}^2 = -1, \mathrm{j}^2 = -1, \mathrm{k}^2 = -1, \mathrm{ij} = \mathrm{k}, \mathrm{jk} = \mathrm{i}, \mathrm{ki} = \mathrm{j}, \\ &\mathrm{ij} = -\mathrm{ji}, \mathrm{jk} = -\mathrm{kj}, \mathrm{ki} = -\mathrm{ik}, \mathrm{ijk} = -1 \end{aligned} \tag{10}$$

In addition, it is geometrically convenient to describe the quaternions as $\mathcal{G}^+_{3,0,0}$, which is an even subalgebra of $\mathcal{G}_{3,0,0}$. Then, a *quaternion* space can be constructed based on the subspace $\{\mathbf{e}_{\{\}}, \mathbf{e}_{\{2,3\}}, \mathbf{e}_{\{1,2\}}, \mathbf{e}_{\{1,3\}}\}$, where $\mathbf{e}_{\{\}}, \mathbf{e}_{\{2,3\}}, \mathbf{e}_{\{1,2\}}$ and $-\mathbf{e}_{\{1,3\}}$ represent 1, i, j and k, respectively.

In such a framework, all the properties of *quaternion*s in Eq. (10) can be verified using the automatic decision procedure `GEOM_ARITH` in Sect. 2.4.

4.4 Dual Quaternions

A dual quaternion can be represented in the form of $p + \varepsilon q$ where p and q are ordinary quaternions, but ε is a dual unit (i.e., $\varepsilon\varepsilon = 0$) that commutes with each element of the algebra. Dual quaternions are isomorphic with $\mathcal{G}_{0,2,1}$, so they can be represented as $\mathcal{G}_{0,2,1}$. For geometric appeal, they can also be described as $\mathcal{G}^+_{3,0,1}$, which is an even subalgebra of $\mathcal{G}_{3,0,1}$. In this way, a *dual quaternion* space is thus constructed based on the subspace $\{\mathbf{e}_{\{\}}, \mathbf{e}_{\{2,3\}}, \mathbf{e}_{\{1,2\}}, \mathbf{e}_{\{1,3\}}, \mathbf{e}_{\{1,4\}}, \mathbf{e}_{\{3,4\}}, \mathbf{e}_{\{2,4\}}, \mathbf{e}_{\{1,2,3,4\}}\}$, where $\mathbf{e}_{\{\}}$, $\mathbf{e}_{\{2,3\}}$, $\mathbf{e}_{\{1,2\}}$, $-\mathbf{e}_{\{1,3\}}$ and $\mathbf{e}_{\{1,2,3,4\}}$ stand for 1, i, j, k and ε respectively.

In this framework, in addition to Eq. (10), the following properties can be automatically proved using `GEOM_ARITH` in Sect. 2.4.

$$\mathrm{i}\varepsilon = \varepsilon\mathrm{i}, \mathrm{j}\varepsilon = \varepsilon\mathrm{j}, \mathrm{k}\varepsilon = \varepsilon\mathrm{k}, \varepsilon^2 = 0$$

To sum up, the formalization of $\mathcal{G}_{p,q,r}$ can be used to formulate all specific GA spaces.

5 Conclusions

This paper provides a formalization of the GA theory in HOL Light. At first, the definition is proposed for multivectors with quadratic forms with arbitrary signatures. Then, a uniform abstract product is constructed to facilitate the formalization of three basic operations, including the outer product, inner product and geometric product. Next, some operation properties are formally verified and an effective automatic conversion is developed to simplify expressions and prove identities in GA. Based on the formalization of the multivector-based algebra, the geometric elements (i.e., blades and versors) and their relations are further formally formulated. At last, several specific spaces such as *complex numbers, quaternion*s and *dual quaternion*s are formally represented. The present work lays a foundation for the GA-based formal analyses and verifications in many engineering fields such as robotics, machine vision and computer graphics.

Finally, it deserves noting that there are still some other GA theories that need further formalization for the purpose of engineering applications. For example, the GA calculus is also widely used in many fields and its formalization is our future work.

Acknowledgements This work was supported by the National Key R & D Plan (2017YFC0806700), National Natural Science Foundation of China (61572331, 61876111, 61472468, 61602325), Capacity Building for Sci-Tech Innovation - Fundamental Scientific Research Funds (025185305000).

References

1. Hestenes, D.: A Unified Language for Mathematics and Physics. Springer, Netherlands (1986)
2. Kleppe, A.L., Egeland, O.: Inverse kinematics for industrial robots using conformal geometric algebra. Model. Identif. Control **37**(1), 63–75 (2016)
3. Stanway, M.J., Kinsey, J.C.: Rotation identification in geometric algebra: theory and application to the navigation of underwater robots in the field. J. Field Robot. **32**(5), 632–654 (2015)
4. Bernal-Marin, M., Bayro-Corrochano, E.: Integration of Hough transform of lines and planes in the framework of conformal geometric algebra for 2D and 3D robot vision. Pattern Recognit. Lett. **32**(16), 2213–2223 (2011)
5. Franchini, S., Gentile, A., Sorbello, F., Vassallo, G., Vitabile, S.: An embedded, FPGA-based computer graphics coprocessor with native geometric algebra support. Integr. VlSI J. **42**(3), 346–355 (2009)
6. Hestenes, D., Sobczyk, G., Marsh, J.S.: Clifford algebra to geometric calculus: a unified language for-mathematics and physics. Am. J. Phys. **53**(5), 510–511 (1985)
7. Dorst, L., Doran, C., Lasenby, J.: Applications of Geometric Algebra in Computer Science and Engineering. Birkhauser, Basel (2002)
8. Macdonald, A.: A survey of geometric algebra and geometric calculus. Adv. Appl. Clifford Algebras **27**(1), 1–39 (2016)
9. Abłamowicz, R., Fauser, B.: Clifford and Graßmann Hopf algebras via the BIGEBRA package for Maple. Comput. Phys. Commun. **170**(2), 115–130 (2002)
10. Aragoncamarasa, G., Aragongonzalez, G., Aragon, J.L., Rodriguezandrade, M.A.: Clifford algebra with mathematica. Physics (2008). Preprint. arXiv:0810.2412. October 2008
11. Mann, S., Dorst, L., Bouma, T.: The Making of GABLE: A Geometric Algebra Learning Environment in Matlab. Birkhüser, Boston (2001)
12. Gordon, M.J.C., Melham, T.F.: Introduction to HOL: a theorem proving environment for higher order logic. FEBS Lett. **89**(2), 317–320 (1993)
13. Harrison, J.: HOL light: a tutorial introduction. In Srivas, M., Camilleri, A., eds.: Proceedings of the First International Conference on Formal Methods in Computer-Aided Design (FMCAD'96). Volume 1166 of Lecture Notes in Computer Science., Springer, pp. 265–269 (1996)
14. Paulson, L.C.: Isabelle—A Generic Theorem Prover. LNCS, Heidelberg (1994)
15. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development. Springer, Berlin (2004)
16. Dutertre, B.: Elements of mathematical analysis in PVS. Lect. Notes Comput. Sci. **1125**, 141–156 (1999)
17. Kaufmann, M., Manolios, P., Moore, J.S., Kaufmann, M., Moore, J.S.: Acl2 Computer Aided Reasoning: An Approach (vol 1). World Scientific Publishing Company **81**(957), 1–27 (2002)
18. Rudnicki, P.: An overview of the Mizar project. Univ. Technol. Bastad **31**(3), 311–332 (1994)
19. Arthan, R.: Mathematical Case Studies: The Geometric Algebra (2006). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.1120. May 2012
20. Fuchs, L., Théry, L.: Implementing geometric algebra products with binary trees. Adv. Appl. Clifford Algebras **24**(2), 589–611 (2014)
21. Harrison, J.: The HOL Light theory of Euclidean space. J. Autom. Reason. **50**(2), 173–190 (2013)
22. Ma, S., Shi, Z., Shao, Z., Guan, Y., Li, L., Li, Y.: Higher-order logic formalization of conformal geometric algebra and its application in verifying a robotic manipulation algorithm. Adv. Appl. Clifford Algebras **26**(4), 1–26 (2016)
23. Harrison, J.: The HOL Light theorem prover. http://www.cl.cam.ac.uk/~jrh13/hol-light/. Accessed 6 Aug 2016
24. Harrison, J.: HOL Light: an overview. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2009. Volume 5674 of Lecture Notes in Computer Science., Munich, Germany, Springer-Verlag 60–66 (2009)

25. Hales, T., Adams, M., Bauer, G., Dang, D.T., Harrison, J., Hoang, T.L., Kaliszyk, C., Magron, V., Mclaughlin, S., Nguyen, T.T.: A formal proof of the Kepler conjecture. Mathematics **16**(3), 47–58 (2015)
26. Harrison, J.: Geometric algebra. https://github.com/jrh13/hol-light/blob/master/Multivariate/clifford.ml. Accessed 4 May 2016
27. Rivera-Rovelo, J., Bayro-Corrochano, E.: Medical image segmentation using a self-organizing neural network and Clifford geometric algebra. In: International Joint Conference on Neural Networks, pp. 3538–3545 (2006)
28. Hestenes, D., Li, H., Rockwood, A.: New algebraic tools for classical geometry. Geometric Computing with Clifford Algebras, pp. 3–26 (2001)
29. Hestenes, D.: New foundations for classical mechanics. Math. Gaz. **71**(458), 703–704 (2002)
30. Harrison, J.: A HOL theory of Euclidean space. In: Hurd, J., Melham, T. (eds.) Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005. Volume 3603 of Lecture Notes in Computer Science., Oxford, UK, Springer-Verlag, pp. 114–129 (2005)
31. Harrison, J.: Definition of finite Cartesian product types. https://github.com/jrh13/hol-light/blob/master/cart.ml. Accessed 16 May 2016
32. Harrison, J.: Real vectors in Euclidean space, and elementary linear algebra. https://github.com/jrh13/hol-light/blob/master/Multivariate/vectors.ml. Accessed 11 June 2016
33. Harrison, J.: Determinant and trace of a square matrix. https://github.com/jrh13/hol-light/blob/master/Multivariate/determinants.ml. Accessed 11 Jan 2016
34. Dorst, L.: The Inner Products of Geometric Algebra. Birkhäuser, Boston (2002)
35. Hartley, D., Tuckey, P.: Gröbner bases in Clifford and Grassmann algebras. J. Symb. Comput. **20**(2), 197–205 (1995)
36. Harrison, J.: Formalizing basic complex analysis. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec. Volume 10(23) of Studies in Logic, Grammar and Rhetoric, pp. 151–165. University of Białystok (2007)
37. Gabrielli, A., Maggesi, M.: In: Formalizing Basic Quaternionic Analysis. Springer, Cham pp. 225–240 (2017)