Adv. Appl. Clifford Algebras

DOI 10.1007/s00006-016-0650-5

Advances in
Applied Clifford Algebras

Higher-Order Logic Formalization of Conformal Geometric Algebra and its Application in Verifying a Robotic Manipulation Algorithm

Sha Ma, Zhiping Shi*, Zhenzhou Shao, Yong Guan, Liming Li and Yongdong Li

Abstract. Conformal geometric algebra (CGA) is an advanced geometric language used in solving three-dimensional Euclidean geometric problems due to its simple, compact and coordinate-free formulations. It promises to stimulate new methods and algorithms in all areas of science dealing with geometric properties, especially for engineering applications. This paper presents a higher-order logic formalization of CGA theories in the HOL-Light theorem prover. First, we formally define the classical algebraic operations and representations of geometric entities in the new framework. Second, we use these results to reason about the correctness of operation properties and geometric features such as the distance between the geometric entities and their rigid transformations in higher-order logic. Finally, in order to demonstrate the practical effectiveness and utilization of this formalization, we use it to formally model the grasping algorithm of a robot based on the conformal geometric control technique and verify the property that whether the robot can grasp firmly or not.

Keywords. Conformal geometric algebra, HOL-light, Formalization, Grasping algorithm, Robot.

1. Introduction

Conformal geometric algebra (CGA) [7] is a modern mathematical tool for geometric representation and computation, which provides a unified and compact homogeneous algebraic framework for classical geometries. "Conformality" comes from the fact that the angles can be kept unchanged easily during the conformal transformations [25]. The construction of a conformal model involves embedding Euclidean space in a higher dimensional one. A conformal

*Corresponding author.

Published online: 22 March 2016

group on the n-dimensional Euclidean vector space $\mathbb{R}^n$ has a natural representation in a space with two more dimensions, based on the *Minkowski signature* (1,1) [28]. CGA shows a great deal of advantages in geometric processing and calculation due to its geometric intuitiveness and a set of effective algorithms for expansion, elimination and simplification. CGA, as a mathematical tool, has the advantages and potential in solving the problems of robots. In robotics, a couple of diverse mathematical approaches like vector algebra, trigonometry, homogenous coordinates, quaternions or dual quaternions are used for different applications. Nevertheless, CGA can unify all these different approaches in a single mathematical system [18]. This is due to two reasons: the use of unified representations and the advantageous representation of the required transformations using versors. In addition, CGA has unique features in robotics since it can deal directly with geometric objects rather than with sequences of numbers. More precisely, an element of an expression has a clear meaning of being a geometric object or a transformation operator for algebraic manipulations in CGA framework. It has become one of the most important branches of geometric algebra (GA) currently and were highly valued due to the above-mentioned unique features and attractive properties. In the last few years, it has been widely used in robotics, graphics,animation, computer vision, cosmology, quantum mechanics and other high- tech fields [2,6,8,9,17,23,29,31,32].

Traditionally, CGA based analysis has been done using computer based numerical techniques or symbolic methods. However, both of these techniques cannot yield accurate results for safety-critical areas such as space travel, medicine and military. Numerical methods cannot guarantee an accurate value because of the limitations of floating-point numbers and computation resources. Symbolic methods, provided by computer algebraic systems (CASs) like Maple, CLUCalc [22] and Gaalop [21], offer relatively complete packages for CGA analysis. Despite being very efficient for calculating mathematical solutions symbolically, these methods cannot be considered 100 % reliable since the involvement of unverified huge symbolic algorithms in their cores, which are quite likely to contain bugs. For example, CASs are often open to doubt even for something as trivial as explicit calculation involving approximations like $\sin(0{:}7) = 0{:}6442176872$ and analytical derivation under default conditions like $x \neq 1$ in the derivation of $(x^2-1)/(x-1) = x+1$ [11]. A possible way to solve the above-mentioned accuracy problem is to conduct CGA analysis by the formal methods. In the last few decades, formal methods have turned out as a successful verification technique, which allows precise analysis and has been widely used in both hardware and software systems [10,24]. The main idea of formal methods is to develop a mathematical model for the given system or to design and analyze this system using mathematical reasoning, which in turn increases the chances for catching subtle but critical design errors that are often ignored by traditional techniques like paper-and-pencil based proofs or the above mentioned ones[35]. Higher-order logic theorem proving [4] is one of the two most commonly used formal methods. It is an interactive verification technique, which is more flexible and can handle a variety of systems, compared with the model checking technique [33]. To

overcome the above-mentioned limitation of inaccuracy, we present an initial formalization of CGA theories in a higher-order-logic theorem prover.

Our main idea is to use the abundant expressiveness of higher-order-logic to formalize the basic algebraic operations of CGA such as geometric product, inner product and outer product. Then, these results are used to construct conformal geometric entities and verify the geometric properties like distances and angles between them and their inner product null space (IPNS), outer product null space (OPNS) and transformations. The main benefit of these verified results is that they can greatly minimize the user intervention in CGA based analysis and further exploit their full potential for the formal analysis of many systems such as robots based on CGA methods. Finally, we use it to formally model a grasping algorithm of a robot in order to demonstrate the practical effectiveness and utilization of the formalization. It is found that the CGA geometric representation and its rich algebraic expressions offer great flexibility in the process of modeling mechanical objects. The work described in this paper is done using the HOL-Light theorem prover [15]. Here, we choose HOL-Light for the formalization of CGA due to the availability of some basic formalized GA theories on Euclidean space [13] by John Harrison. This paper has two main contributions: (1) the construction of a HOL theory of CGA for the first time; (2) the proposition of a novel CGA-based formal analysis method for the grasping algorithm of a manipulator.

The rest of the paper is organized as follows. Section 2 provides a brief introduction to the HOL-Light theorem prover and presents an overview on Harrison's formalization of Clifford algebra and our general framework of CGA Formalization. Section 3 introduces the formalization of basic operation theories in the framework of GA. The formalization of CGA theories including geometric representation and calculation are provided in Sect. 4. It is used to formally model the grasping algorithm of a manipulator in Sect. 5. Finally, Sect. 6 concludes the paper.

2. Preliminaries

2.1. GA and CGA

There are several different definitions of GA [30], suitable for different purposes. According to the most popular definition, it is a quotient of the tensor algebra when it has quadratic form on the vector space $V \subset \mathrm{Cl}_{p,q}$ [34]. In general, a geometric algebra $\mathrm{Cl}_{p,q}$ is a linear real vector space of 2^n dimensions, fulfilling $n = p + q$, with a subspace structure called blades. We will use $\mathbf{e}_i$ to denote the ith orthonormal basis, where $1 \leq i \leq n$. The geometric product of two bases in $\mathrm{Cl}_{p,q}$is defined as

$$\mathbf{e}_i\mathbf{e}_j = \begin{cases} 1 & for\, i = j \in 1, \ldots, p \\ -1 & for\, i = j \in p+1, \ldots, p+q \\ \mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i = \mathbf{e}_{ij} & for\, i \neq j \end{cases} \quad (1)$$

where p and q stand for the number of base vectors, which square to $+1$ and -1, respectively. The main construction of CGA is to embed the Euclidean

space $\mathbb{R}^3$ into the Minkowski space $\mathbb{R}^{3+1,1} = \mathbb{R}^3 \oplus \mathbb{R}^{1,1}$, whose vector space $\mathbb{R}^{1,1}$ has the orthonormal basis $\{\mathbf{e}_+,\mathbf{e}-\}$. Thus, a basis is given by $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_+, \mathbf{e}-\}$, which means two extra dimensions are added to the Euclidean space being modeled. Fortunately, the geometric algebra $\mathrm{Cl}_{4,1}$ can be utilized to treat conformal geometry in an very elegant way. Itcontains $2^5 = 32$ elements.

2.2. HOL-Light

HOL-Light is a relatively new version of the HOL theorem prover, which is an interactive theorem proving environment for the construction of mathematical proofs including an efficient set of inference rules and the usage of Objective CAML (OCaml). HOL-Light is one of a long line of HOL theorem provers that have been released into the public domain for applications in academia, industry and government organizations [12]. The first version of HOL called HOL88 was developed by Mike Gordon at the Cambridge University in 1980's and the later ones include HOL90 and HOL98 and HOL4. HOL-Light is intended to be a simpler and more elegant version as compared to other versions. It proves theorems in a system of classical higher order logic based on the polymorphic theory. There are two main types of interactive proof methods in HOL: forward and backward. The main idea of the forward method is that the user starts from the primitive inference rules and tries to realize the goal in a step-by-step style. However, it's often more convenient to prove theorems in a backwards method, starting from the goal and reducing it into various sub-goals so that they can be solved by applying the needed tactics such as REWRITE_TAC and MATCH_MP_TAC until the given goal is verified. The tactic mechanism of HOL-Light allows one to tackle proofs in a mixture of forward and backward steps. HOL-Light can maintain a high degree of reliability because all proofs in the environment proceeds by the application of low-level primitive inference rules. What's more, users can implement their own derived rules for proving various useful theorems and a number of useful mathematical theories such as real analysis [14] are already available.

2.3. Harrison's Formalization of Clifford theories

Some of the basics of GA analysis in HOL-Light were laid down by John Harrison, which was named Clifford library and is fundamental to our work. The library contains some basic definitions and theories of GA on Euclidean space, mainly including multivector, base vector, geometric product, inner product and outer product. Overall, his formalization contains approximately 979 lines of proof scripts. His work provided us with a lot of inspirational ideas about the formalization of GA, especially the definition of geometric product. However, Harrison's Clifford library still has some inadequacies. Thus, we cannot use it to perform the formalization of CGA directly. Harrison's Clifford library has formalized Cl_n, which is the GA of n-dimensions in the Euclidean space. This is a simplified version of GA, which is able to represent basic geometric entities and 3D transformations in the Euclidean space. Whereas, the expression and transformation in the Harrison's Clifford library are non-homogeneous. Furthermore, it is difficult to deal with the non-Euclidean geometry such as the hyperbolic geometry and spherical

geometry based on the definition of GA. But the non-Euclidean geometry has now been found to be very necessary for the description of general objects in the physical world and other fields.

Relatively speaking, compared with the formula in Eq. (1), the geometric product of two base vectors in Harrison's Clifford library is defined as

$$\mathbf{e}_i\mathbf{e}_j = \begin{cases} 1 & for \quad i = j \in 1, \ldots, p \\ \mathbf{e}_i \wedge \mathbf{e}_j = \mathbf{e}_{ij} & for \quad i \neq j \end{cases} \tag{2}$$

The definition form $\mathrm{Cl_{p,q}}$ of GA has many advantages compared with the form $\mathrm{Cl_n}$. Different indexes p and q $\mathrm{Cl_{p,q}}$ can be selected to construct more optimal algebra framework or spaces for solving problems. For example, reference [5] shows that it is more superior to embed perceptron neurons into geometric algebra $\mathrm{Cl_{p,q}}$ than geometric algebra $\mathrm{Cl_n}$. In order to extend the modeling capability and application range of GA and meet the need of establishing CGA, we specify the geometric algebra $\mathrm{Cl_n}$ of the n-dimensional space by $\mathrm{Cl_{p,q}}$ and verify theorems in the framework of HOL-Light theorem prover based on Harrison's work. As is known, only the geometric product that is equal to 1 is considered in Harrison's definition of GA. Therefore, we still have to consider the case of geometric product that is equal to -1, and this is one of the major challenges we faced. Meanwhile, we also extend the other important operators such as the inverse, dual, norm of multivector in $\mathrm{Cl_{p,q}}$. The formalization of geometric algebra $\mathrm{Cl_{p,q}}$ is introduced in a very simple way in Sect. 3, and then p and q are assumed to be equal to 4 and 1, respectively, to construct the conformal model. The general framework of modeling and verification based on Harrison's Clifford library is depicted in Fig. 1, and the contents in the two dashed boxes are the main work of the paper.

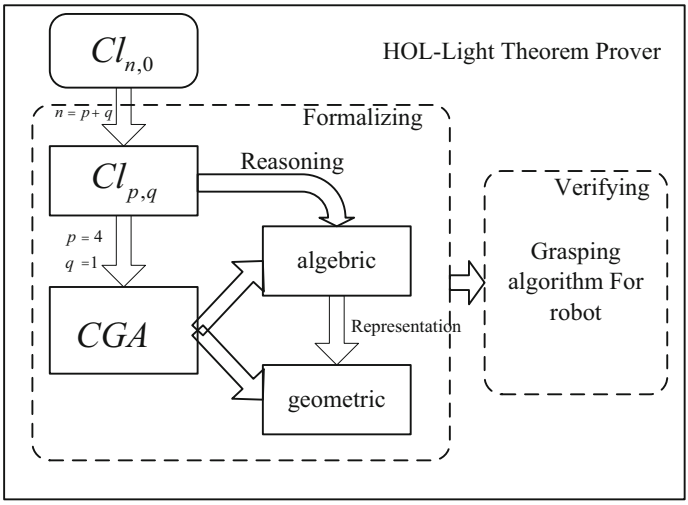


FIGURE 1. The general framework of modeling and verification based on the Clifford library

Table 1. Multivector-related definitions in Clifford Library

Definition	HOL formalization
Setcode	|−setcode s = 1 + binarysum (IMAGE PRE s)
Codeset	|−codeset n = IMAGE SUC (bitset(n - 1))
Sindex	|− x$$s = x$(setcode s)
Lambdas	|−(lambdas) g = (lambda i. g(codeset i))
Mbasis	|−mbasis i = lambdas s. if i = s then & 1 else & 0

In HOL-Light, an n-dimensional vector is represented as a R^n column matrix with the elements being real numbers, whose data-type is defined as **real^N**. Similarly, a n-dimensional multivector in a n-dimensional GA is defined as **real^(N)multivector** in the Clifford library. Harrison's Clifford library gives us some heuristic definitions related to multivector (see Table 1), which can help to understand the rest of paper.

Table 1 shows some functions in binary library. The **setcode** gives the function mapping from a set such as {1, 2, 3} to its corresponding number 8 (1 + 2^0 + 2^1 + 2^2 = 8), and the **codeset** is just its opposite function. The construction of these two definitions is intended to verify the bi-jection relationship between the set {1,2,...n} and the number 2^n. That is to say, the single basis bi-jection over the entire space in GA is provided in the Clifford library. The function **lambdas** and operator **$$** are defined to use the subspace size as a collection of elements to represent the entire space. For example, the notation (**p$$s:real**) represents the size of subspace $\mathbf{e}_s$ of the multivector p. The mbasis function defines the basis blades indexed by subsets of 1...N. For example, the basis vector $\mathbf{e}_1$can be represented as **mbasis{1}**.

2.4. General Framework of CGA Formalization

The formalization of CGA includes not only geometrical expression but also algebraic operation. They are intimately related together because the geometrical expression cannot be done without the support of algebraic operation. Specifically, the representations of geometric entities are based on inner product and outer product. These two products can be used to reveal the geometric meaning by verifying their OPNS or IPNS. Reasoning about the distance between two geometric entities is based upon its corresponding inner product, and the formalization of transformation is primarily based on the geometric product. In addition, the basic operators such as reverse, inverse and dual are utilized to describe and verify some important theorems. For example, the theorem that the two representations in CGA are duals of each other can be described and verified by the operator "dual". Figure 2 describes the relationship between the geometrical and algebraic parts of CGA.

It deserves noting that Li [26] created a new clifford algebraic model to mechanically prove some famous theorems, which are difficult to be proved. He combined the GA representation with Wu's method [38] to solve equations of multivectors. Further, he established the method of Clifford bracket algebra, whose foundation is a set of generalized Grassmann-Plücker relations.

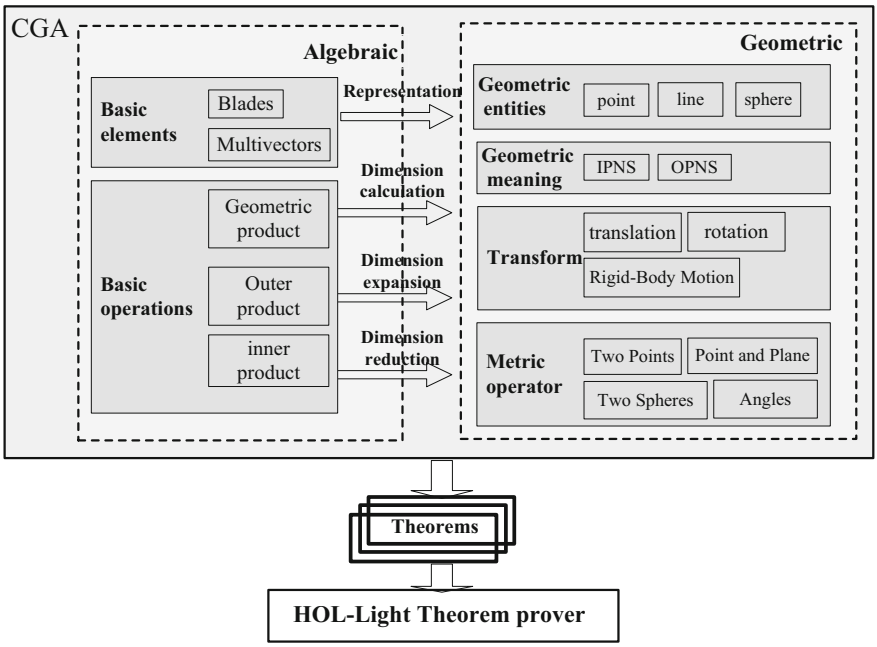


FIGURE 2. General framework of CGA formalization

The main difference between Li's work and ours is that our formalization of CGA is based on higher-order logic in a reliable environment. Here, we don't need to solve a group of equations, and the theorems can be verified just by using the primitive inference rules and formalized theorems in the theorem prover. Li's work greatly simplifies the complexity of symbolic geometric computation. In our work, we formally model CGA at first, and then use the model to verify that the design of a system or an algorithm meets the specified properties. Besides the property proving, the design defects of the system may also be found out timely. This is the superiority of our formal method over the traditional ones. Figure 3 describes the process of robotic verification using our CGA formalization.

3. Formalization of Basic Operations of GA

Blades and multivectors are basic algebraic elements of GA. An (p + q)-dimensional GA consists of blades with grades 0, 1, 2,. . . , (p + q), where a scalar is a 0-blade, the base vectors are the 1-blades, $\mathbf{e}_i \wedge \mathbf{e}_j$ $(= \mathbf{e}_{ij})$ are the 2-blades spanned by two 1-blades, and so on,we call the outer product of k vectors a k-blade. There are 2^{p+q} blades in an (p + q)-dimensional GA. Only one element of the maximum grade p + q is called the pseudoscalar. This leads to a basis for the entire algebra $\mathrm{Cl}_{p,q}$:

$$\{1\}, \{e_i\}, \{e_i \wedge e_j\}, \{e_i \wedge e_j \wedge e_k\}, \ldots, \{e_1 \wedge e_2 \wedge \ldots \wedge e_{p+q}\} \qquad (3)$$

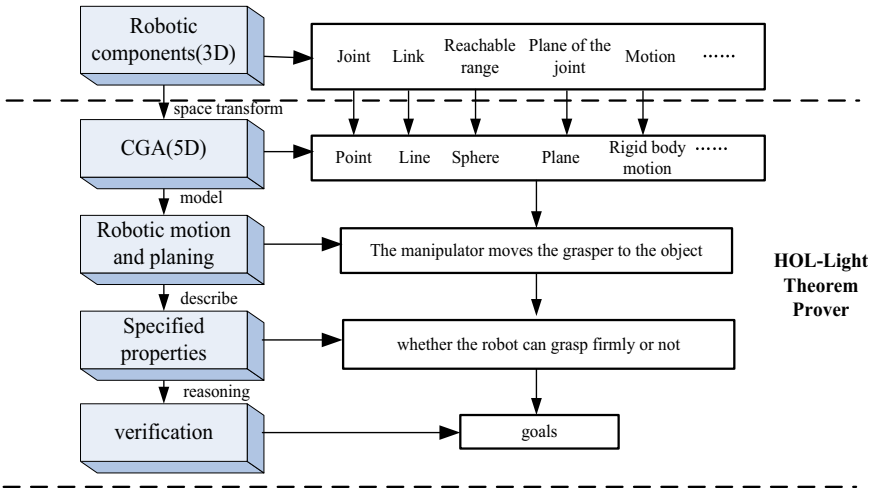


FIGURE 3. The process of robotic verification based on CGA formalization

Any multivector can be expressed in terms of orthonormal basis . A multivector in $\mathrm{Cl}_{3,0}$ can be defined with $2^3 = 8$ real numbers and will look like this:

$$\mathbf{A} = \underbrace{\alpha_1}_{scalar\ part} + \underbrace{\alpha_2\mathbf{e}_1 + \alpha_3\mathbf{e}_2 + \alpha_4\mathbf{e}_3}_{vector\ part} + \underbrace{\alpha_5\mathbf{e}_{12} + \alpha_6\mathbf{e}_{13} + \alpha_7\mathbf{e}_{23}}_{bivector\ part} \underbrace{\alpha_8\mathbf{e}_{123}}_{trivector\ part} \tag{4}$$

The data-type of a multivector in geometric algebra $\mathrm{Cl}_{\mathrm{p,q}}$ is defined as **real^(p,q)multivector**, where **(p,q:num#num)** refers to the number of two bases equal to +1 and −1, respectively. In order to facilitate the understanding of the rest of the paper, some of the frequently used operations in $\mathrm{Cl}_{\mathrm{p,q}}$ are formalized below:

Definition 1. K-vectors
| − ∀k p. grade k p = lambdas s**. if** s HAS_SIZE k **then** p**$$**s **else &**0

The function **grade** receives a number and a multivector, and returns the part with grade k of the multivector. The **(s:num → bool)** is the subset of 1...(p+q), which represents the index of the subspace.

Definition 2. K-blade
|− blade k f ↔ ∀k f. independent (IMAGE f (1...k)) → (blade k f = iterate (outer) (1...k) (multivec o f))

The function **blade** receives a number k and a function f, and returns a k-blade, where (k:num) refers to the dimension of the subspace the blade spans. The (*f:num → real ^(p+q)*) refers to a number i to the corresponding vectors $\mathrm{v_{i,}}$ which are mutually linearly independent. We use the function *independent* in Vector Library to describe. The *iterate op (s:A− >bool) f* is a generic iteration of operation over set with finite support, which uses k

independent vectors v_i to construct the computational element $v_1^{\wedge}v_2^{\wedge}\ldots^{\wedge}v_k$. The *multivec* is a function maps from a regular vector to the corresponding multivector and defined in Sect. 4.

Definition 3. General product construct

```
|-∀x y. Product mult op x y =
      vsum {s | s SUBSET 1..(dimindex(:p) + dimindex(:q))}
        (λs. vsum {s | s SUBSET 1..(dimindex(:p) + dimindex(:q))}
          (λt. (x$$s * y$$t * mult s t) % mbasis (op s t)))
```

The function **Product** receives two multivectors and returns a multivector that was obtained by them. The **vsum s f** is a summation operator over vectors, and the distinct symbol % represents scalar-vector multiplication. The **mult** and **op** are abstract functions operating on two sets that represent the subspace index of x and y. The **mult** function is used to determine the sign of the abstract product result of the subspace of x and y, and thus the returned value of **(mult s t)** is $+1$ or -1 or 0. We can use the definition to construct the geometric product, outer product and inner product. Geometric product embodies both the inner and outer products, and it combines the notions of orthogonality and collinearity into one operation.

Definition 4. Geometric product

```
|-∀x y. x* y =
      Product (λs t. --(&1) pow  CARD {i,j | i IN 1..(dimindex(:p) + dimindex(:q))  /\
                                            j IN 1..(dimindex(:p) + dimindex(:q))  /\
                                            i IN s /\ j IN t /\ i > j} *
                     --(&1) pow CARD{i,j|iIN(dimindex(:p) + 1)..(dimindex(:p) + dimindex(:q)) /\
                                            j IN (dimindex(:p) + 1)..(dimindex(:p) + dimindex(:q))
                                            /\ i IN s /\ j IN t /\ i = j})
           (λs t. (s DIFF t) UNION (t DIFF s))
            x y
```

While the operator symbol * looks like a real multiplication operator, it represents geometric product when x and y are multivectors with data-types **real^(p,q)multivector**. The **mult** function in geometric product means when i and j are in the range of 1 and p + q with i being greater than j, the result provides a negative sign, and if i and j are equal and in the range of p + 1 and p + q, the result provides a negative sign as well. The op function represents the basis blades indexed by a certain set that is obtained by two disjoint parts necessarily and satisfies the operation rules of geometric product.

Definition 5. Outer product

```
|- ∀x y. x outer y =
       Product (λs t. if ~(s INTER t = {}) then &0
                      else --(&1) pow CARD {i,j |i IN 1..(dimindex(:p) + dimindex(:q))  /\
                                                j IN 1..(dimindex(:p) + dimindex(:q)) /\
                                                i IN s /\ j IN t /\ i > j})
            (λs t. (s DIFF t) UNION (t DIFF s))
             x y
```

The meaning of **s INTER t** = { } in **mult** function means x and y contain equal subspaces. If **mult s t** returns zero value, it means the outer product of two parallel vectors is 0.

Definition 6. Inner product

```
|- ∀x y. x inner y =
      Product (λs t. if ~(s SUBSET t) then &0
                  else --(&1) pow CARD {i,j | i IN 1..(dimindex(:p) + dimindex(:q)) /\
                                              j IN 1..(dimindex(:p) + dimindex(:q)) /\
                                              i IN s /\ j IN t /\ i > j} *

                       --(&1)powCARD{i,j|i IN(dimindex(:p)+1)..(dimindex(:p)+dimindex(:q))/\
                                         j IN(dimindex(:p)+1)..(dimindex(:p)+ dimindex(:q))/\
                                         i IN s /\ j IN t /\ i = j})
              (λs t. (s DIFF t) UNION (t DIFF s))
               x y
```

where, **if ~(s SUBSET t) then & 0** represents that the inner product is always zero when the grade of the RHS operand is lower than that of the LHS one. This is the property of left contraction. Here, we define the inner product as the contraction inner product since it is more useful in computer science. If **~(s SUBSET t)** then the two sets must meet the condition **~ (s INTER t = {})** and their outer product must be equal to zero. Then, the inner product between their subspace equals to their geometric product.

The above mentioned three major products have a series of linear properties such as distributive addition, commutative scalar multiplication and associative scalar multiplication, etc. It should be noted that the inner product does not meet the property of associativity because of its left direction property. In addition, the outer product has anti-commutative property exclusively. We spent plenty of time and energy to reason about the correctness of these properties in HOL-Light. In particular, the associativity property of the geometric product is the main bottleneck in the proof. At first, we need to break the goal into several sub-goals to prove that many complex point-pair sets like {**i,j** |**...** } are finite, and then we still need to simplify its expressions. Next, some important operators in GA are given below, which will be useful in the formalization of CGA.

Definition 7. Reversion

| − ∀x. reversion x = lambdas s**.** −(**&**1) pow ((CARD(s) ***** (CARD(s) **-** 1)) DIV 2) ***** x**$$**s

Definition 8. Invertible
| − ∀x. invertible_mult x = ∃x'**.** (x ***** x' = mbasis{})/\ (x' ***** x = mbasis{})

Definition 9. The inverse of k-blade
| − ∀k p. multiv_inv (blade k p) = **if** invertible_mult (blade k p) **then** reversion (blade k p)/(reversion(blade k p) ***** (blade k p)) **else** vec 0

Definition 10. Dual
| − ∀x. DUAL x = x * (multiv_inv pseudo)

Definition 11. Norm of multivector
|− ∀x. mult_norm x = sqrt((reversion x * x)$${})

Definition 8 presents the definition of an invertible multivector because not all multivectors are invertible. In Definition 9, the operator/represents the division of a geometric product, which is defined as **x/y = @z. x = y * z**. In Definition 10, **pseudo** means a pseudoscalar in $\mathrm{Cl}_{p,q}$ that is defined as **pseudo = mbasis(1..(dimindex(:p)+dimindex(:q)))** in HOL-Light. The symbol $${} in Definition 11 is the function get the scalar part from a multivector. Theoretical knowledge of the above mentioned operations can be found in reference [36]. We will use these definitions to formalize the CGA theories in the next section.

4. Formalization of CGA

The data-type of all multivectors in CGA can be defined as **real^(4,1)multivector** for its corresponding algebra $\mathrm{Cl}_{4,1}$. The formal verification of CGA not only ensures the correctness of our definition in Sect. 3 but also plays a vital role in minimizing the user intervention in reasoning about CGA based analysis of applications, as will be depicted in Sect. 5 of this paper.

4.1. The Extended Basis

The two additional base vectors, $\mathbf{e}_+$ and $\mathbf{e}_-$, have positive and negative signs, respectively, which means that $\mathbf{e}_+$ squares to $+1$ but $\mathbf{e}_-$ to -1. They are used to define two null vectors as follows.

$$\boldsymbol{e}_0 = \frac{1}{2}(\boldsymbol{e}_- - \boldsymbol{e}_+) \quad \boldsymbol{e}_\infty = \boldsymbol{e}_- + \boldsymbol{e}_+ \quad \mathbf{e}_0^2 = \mathbf{e}_\infty^2 = 0 \tag{5}$$

where $\mathbf{e}_0$ is the origin of the coordinate system and $\mathbf{e}_\infty$ is the point at infinity. Their inner product results in

$$\mathrm{e}_\infty \cdot \mathrm{e}_0 = -1 \tag{6}$$

The base vectors $\mathbf{e}_0$ and $\mathbf{e}_\infty$ allow for a more geometrically meaningful vectors than $\mathbf{e}_+$ and $\mathbf{e}_-$ do. We use **mbasis{4}** and **mbasis{5}** to represent $\mathbf{e}_+$ and $\mathbf{e}_-$ in HOL-Light, which describe the fourth and fifth base vectors in CGA. According to Eq. (5), the null vectors $\mathbf{e}_0$ and $\mathbf{e}_\infty$ can be formalized as follows:

Definition 12. Null vectors
|−null_zero= (&1 / &2)%(mbasis{5} - mbasis{4}) /\null_inf = mbasis{5}+ mbasis{4}

Some critical properties of the null vectors in CGA are formalized as shown in Table 2. They are very useful in simplifying the subsequent calculation in this paper. The proofs of the goals in Table 2 are primarily based on the products of the basis and singleton basis in $\mathrm{Cl}_{p,q}$, which have been formally verified in our work. For example,

TABLE 2. The basic operation properties of null basis

Property	HOL Formalization	Mathematical
GEOM_NULL	|-null_inf * null_inf = vec 0 /\ null_zero* null_zero= vec 0	$e_\infty{}^2 = e_0{}^2 = 0$
OUTER_NULL	|-null_inf outer null_zero= mbasis{4} outer mbasis{5}	$e_\infty \wedge e_0 = e_+ \wedge e_-$
INNER_NULL_INF_ZERO	|-null_inf inner null_inf = vec 0 /\ null_zero inner null_zero = vec 0	$e_\infty \cdot e_\infty = e_0 \cdot e_0 = 0$
INNER_INF_ZERO	|-null_inf inner null_zero= --mbasis{}	$e_\infty \cdot e_0 = -1$
INNER_INF_ZERO_SYM	|-null_zero inner null_inf = null_inf inner null_zero	$e_0 \cdot e_\infty = e_\infty \cdot e_0$
INNER_BASIS_INF	|-∀i.mbasis{i} inner null_inf = **if** i IN 1..3 **then** vec 0 **else if** i = 4 **then** mbasis{} **else if** i = 5 **then** --mbasis{} **else** vec 0	$e_i \cdot e_\infty = \begin{cases} 0 & i \in 1..3 \\ 1 & i = 4 \\ -1 & i = 5 \end{cases}$
INNER_INF_SYM	|-∀i. null_inf inner mbasis{i} = mbasis{i} inner null_inf	$e_\infty \cdot e_i = e_i \cdot e_\infty$
INNER_BASIS_ZERO	|-∀i. mbasis{i} inner null_zero = **if** i IN 1..3 **then** vec 0 **else if** i IN 4..5 **then** --(&1 / &2) % mbasis{} **else** vec 0	$e_i \cdot e_0 = \begin{cases} 0 & i \in 1..3 \\ -\frac{1}{2} & i = 4,5 \end{cases}$
INNER_ZERO_SYM	|-∀i. null_zero inner mbasis{i} = mbasis{i} inner null_zero	$e_0 \cdot e_i = e_i \cdot e_0$

Theorem 1. *The geometric product on singleton basis in* $Cl_{p,q}$

```
|- ∀i j. mbasis{i} * mbasis{j} =
      if i IN 1..(dimindex(:p) + dimindex(:q)) /\ j IN 1..(dimindex(:p) + dimindex(:q))
       then if i = j then if i IN 1..dimindex(:p) /\ j IN 1..dimindex(:p)then mbasis{}
                          else --mbasis{}
            else if i < j then mbasis{i,j}
                 else --(mbasis{i,j})
      else vec 0
```

Theorem 2. *The inner product on singleton basis in* $Cl_{p,q}$

```
|- ∀i j. mbasis{i} inner mbasis{j} =
      if i IN 1..(dimindex(:p) + dimindex(:q)) /\
        j IN 1..(dimindex(:p) + dimindex(:q))
       then if i = j then if i IN 1..dimindex(:p) /\j IN 1..dimindex(:p) then mbasis{}
                          else --mbasis{}
            else vec 0
       else vec 0
```

During the verification process, we need to break the current goal into several sub-goals to simplify the complex point-pair sets such as **{i,j |...}** based on the current assumptions. For example, during the process of verifying theorem 2, we met the complex set in the current goal as follows:

```
{i',j' | (dimindex (:p) + 1 <= i' /\ i' <= dimindex (:p) + dimindex (:q)) /\
      (dimindex (:p) + 1 <= j' /\ j' <= dimindex (:p) + dimindex (:q)) /\
       i' = j /\j' = j /\i' = j'}
```

TABLE 3. List of the conformal geometric entities

Entity	IPNS representation	OPNS representation
Point	$P = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathbf{e}_\infty + \mathbf{e}_0$	
Sphere	$S = P - \frac{1}{2}\mathrm{r}^2\mathrm{e}_\infty$	$S^* = P_1 \wedge P_2 \wedge P_3 \wedge P_4$
Plane	$\pi = \mathbf{n} + d\mathbf{e}_\infty$	$\pi^* = P_1 \wedge P_2 \wedge P_3 \wedge \mathbf{e}_\infty$
Circle	$Z = S_1 \wedge S_2$	$Z^* = P_1 \wedge P_2 \wedge P_3$
Line	$L = \pi_1 \wedge \pi_2$	$L^* = P_1 \wedge P_2 \wedge \mathbf{e}_\infty$
Pointpair	$P_P = S_1 \wedge S_2 \wedge S_3$	$P_P^* = P_1 \wedge P_2$

While the current assumption list is

```
0 [`(1 <= i /\ i <= dimindex (:p) + dimindex (:q)) /\
   1 <= j /\ j <= dimindex (:p) + dimindex (:q)`]
1 [`i = j`]
2 [`~(j <= dimindex (:p))`]
```

Then, we can build the sub-goal to verify the complex set, which equals to the simple set {**(j,j)**}. Some theorems about sets are used in the verification of the sub-goal. In the proof, when we meet a term, we need to judge whether it is true or not. Then, the ASM_CASES_TAC can be employed to perform the proving.

4.2. Geometric Entities

CGA provides a variety of basic geometric entities such as points, spheres, planes, circles, lines and point pairs as listed in Table 3. They have two algebraic representations: the IPNS and OPNS (or 'standard' and 'direct'). These two representations are duals of each other (an asterisk in the superscript denotes the dualization operator). See Sect. 4.3 for more details of the IPNS and OPNS.

In Table 3, $\mathbf{x}$ and $\mathbf{n}$ are in bold type, which indicates that they represent 3D entities obtained by linear combinations of the base vectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$:

$$\mathbf{x} = x_1 e_1 + x_2 e_2 + x_3 e_3 \tag{7}$$

So the data-type of $\mathbf{x}$ is defined as **real^3**. It deserves noting that $\mathbf{x}^2$ is the well-known scalar product in the Euclidean space and we use **dot** function to describe $\mathbf{x}^2$. The outer product "∧" in the OPNS indicates the construction of a geometric object with the help of the points $\{P_i\}$ lying on it. However, the outer product in the IPNS represents the intersection of geometric entities. First, we define these geometric entities in the IPNS representation as follows:

Representation 1. *Points*

```
|- ∀s. point_CGA s = s$1 % mbasis{1} + s$2 % mbasis{2} + s$3 % mbasis{3}
+ (&1 / &2 * (s dot s)) % null_inf + null_zero
```

The function **point_CGA:real^3 → real^(4,1)multivector** maps a vector standing for a point in $\mathbb{R}^3$to its corresponding multivector representing the expression of the point embedded in CGA. The notation **s$n** represents the nth component of a vector s. We can define a function **f: real^N →**

real^(p,q)multivector that maps from a regular vector to the corresponding multivector as follows:

multivec x = vsum (1..(dimindex(:p) + dimindex(:q))) (λi. x$i % mbasis{i})

The **vsum (1.. (dimindex(:p) + dimindex(:q)) f** is the vector summation of the (p + q) terms from 1 to (p + q) of function f . Thus, **s$1 % mbasis{1} + s$2 % mbasis{2} + s$3 % mbasis{3}** can be replaced by **multivec x** in Representation 1. In the same way, more geometric entities can be defined.

Representation 2. *Spheres*
| − ∀p r. sphere_CGA p r = point_CGA p − **((&1/&2) * (**r pow 2**)) %** null_inf

The function **sphere_CGA: real^3→real→ real^(4,1) multivector** describes a sphere characterized by its center point P and its radius r. It can be easily verified that the representation of a point is simply equal to that of a sphere with zero radius.

Representation 3. *Planes*
| − ∀n d .plane_CGA n d = multivec n + d % null_inf

where **(n:real^3)** refers to the 3D normal vector of the plane and **(d:real)** is its distance from the origin. Note that a plane is a sphere of infinite radius. A particular nice representation of the mid-plane between points **a** and **b** can be described as A–B in the IPNS representation.

Representation 4. *Midplane*
| − ∀a b. mid_plane_CGA a b = point_CGA a - point_CGA b

The other basic geometric entities in IPNS and all the basic geometric entities in OPNS can be easily defined by using the outer product (Definition 5). For example, the plane represented by two points lying on it can be defined as follows:
| − ∀a b c. plane_direct_CGA a b c = a outer b outer c outer null_inf

4.3. IPNS and OPNS

The entities of CGA are given geometric meaning by referring to their OPNS or IPNS. The outer product of a geometric expression mainly reflects different levels of mutual construction relations among geometric entities, while the inner product representation builds the corresponding parametric equations through the available measurement indexes of distance and angles. Here, we are only interested in the set of Euclidean vectors embedded in the conformal space PK^{n}, which is represented by $\mathbb{R}^{n+1,1}$. Hence, the Euclidean IPNS and OPNS for PK^{n} can be defined as

$$\begin{aligned} NO_E(A \in \alpha(PK^n)) &:= \{\mathbf{x} \in E^n : X \wedge A = 0\} \\ NI_E(A \in \alpha(PK^n)) &:= \{\mathbf{x} \in E^n : X \cdot A = 0\} \end{aligned} \tag{8}$$

For the blade $\mathrm{A_k}$, the relationship between IPNS and OPNS does indeed exist and it is called dualty:

$$NO_E(A_k) := NI_E(A_k^*) \tag{9}$$

When n is equal to 3, Eq. (8) can be redefined as:

TABLE 4. Formal verification of Euclidean IPNS

Entity	Euclidean IPNS	Formal verification
Point	$NI_E(A) := \{\mathbf{x} \in E^3 : X \cdot A = 0\} = \mathbf{a}$	| − ∀ x. IPNS (point_CGA x) ={x}
Spheres	$NI_E(S = P - \frac{1}{2}\mathrm{r}^2 e_\infty) := \left\{\mathbf{x} \in E^3 : \|\mathbf{x} - \mathbf{a}\|^2 = \rho^2\right\}$	| − ∀ p r. IPNS (sphere_CGA p r) = {x| dist(p,x) pow 2 = r pow 2}
Planes	$NI_E(\pi = \mathbf{n} + d\mathbf{e}_\infty) := \{\mathbf{x} \in E^3 : \mathbf{x} \cdot \mathbf{n} = d\}$	| − ∀ n d. IPNS (plane_CGA n d) = {x| x dot n = d}
Midplane	$NI_E(\pi = A - B) := \{\mathbf{x} \in E^3 : \|\mathbf{x} - \mathbf{a}\| = \|\mathbf{x} - \mathbf{b}\|\}$	| − ∀ a b. IPNS(mid_plane_CGA a b) = {x| dist(x,a) = dist(x,b)}

Definition 13. Euclidean IPNS
| − ∀k s. isblade k s→IPNS s = {x | point_CGA x inner s = vec 0}

Definition 14. Euclidean OPNS
| − ∀k s. isblade k s→OPNS s = {x | point_CGA x outer s = vec 0}

where the function receives a**(s: real^(4,1)multivector)** and returns a (**set: real^3→bool)** that represents the set of its corresponding Euclidean vectors. First we define the function *isblade* to verify whether an (s: real^(4,1)multivector) is k-blade using the Definition 2:

| − ∀k s.isblade k s ↔ (@f. s = blade k f)

and then we can build some goals to verify the representations of geometric entity are all blades by using the EXISTS_TAC. Table 4 shows the corresponding Euclidean IPNSs of the basic geometric entities and their mapping relations verified by us.

A rich set of formalized operations and functions on the vectors in $\mathbb{R}^n$ such as the usual **dot** product and **dist** functions can be utilized to simplify the description of the above theorems. The methods to prove these theorems about Euclidean IPNS are similar. These methods are mainly based on Table 2, Theorem 2 and some simple reasoning on the n-dimensional vectors in the Euclidean space. The theorems successfully proved in Table 4 reveal their geometric meanings and the relationship between the 3D Euclidean space and the 5D conformal space. For instance, it can be found that the Euclidean IPNS of $S(= P - \frac{1}{2}\mathrm{r}^2\mathbf{e}_\infty)$ is a sphere represented by its center point p and radius $r$in the three-dimensional Euclidean space. In addition, they also verifies the correctness of our definitions of geometric entities. In order to express the geometric meaning of OPNS vividly, next we formalize the **on** function, which represents a point lying on a geometric object. It is known that x ∧ A =0 if and only if x is linearly dependent on A.

Definition 15. A point that lie on a geometric object
| − ∀x s. (point_CGA x) ON s ↔ (point_CGA x) outer s= vec 0

In this way, we can verify that $NO_E(A \wedge B \wedge C \wedge D)$ is a sphere passing through points **a**, **b**, **c** and **d** in the 3D Euclidean space. Next, the dual relationship between these two representations can then be verified because the geometric entities are all blades that we have verified. Based on Definition 10, the goal can be described as follows:

Theorem 3. *Dual relationship* $| - \forall k\ f$ *.IPNS (blade k f) = OPNS (DUAL (blade k f))*

4.4. Distances and Angles

In CGA, points, planes and spheres can be represented as vectors. Thus, their inner product is a scalar and can be used as a measure of distances or angles. We can investigate the inner product of conformal vectors and understand its geometric meaning. In this section, we verify the distances between them based on Definition 6. The computing process of distances and angles can be found in reference [18].

A vector in CGA can be written as

$$V = v_1e_1 + v_2e_2 + v_3e_3 + v_4e_\infty + v_5e_0 \tag{10}$$

Based on some properties shown in Table 2, the inner product between a conformal vector U and a conformal vector V results in

$$U \cdot V = \mathbf{u} \cdot \mathbf{v} - u_5v_4 - u_4v_5 \tag{11}$$

Based on this observation we can investigate the inner product between points, spheres and planes. so first we build a goal to verification the Eq. (11)

| − ∀ u v u_4 u_5 v_4 v_5. (multivec u + u_4 % null_inf + u_5 % null_zero) inner (multivec v + v_4 % null_inf + v_5 % null_zero) =(u dot v - u_5 * v_4- u_4 * v_5) % mbasis{}

Then apply it in the case of U and V being the various geometric entities. For instance the distance between points:

| − ∀ a b. (point_CGA a) inner (point_CGA b) = (−(&1 / &2) * dist(a,b) pow 2) % mbasis{}

According to Eq. (11) we get the square of the Euclidean distance between two Euclidean vectors corresponds to the inner product of the two conformal points multiplied by −2. Similarly, we can compute the inner product of two spheres:

| − ∀ p1 r1 p2 r2. (sphere_CGA p1 r1) inner (sphere_CGA p2 r2) = (&1 / &2 * ((r1 pow 2 + r2 pow 2) - dist(p2,p1) pow 2)) % mbasis{}

The proofs of the distance between two objects are primarily based on Representation 1, Theorem 2 and some linearity properties of the inner product. There will be many conditional statements emerging in the above goals when Theorem 2 is applied, and most of these conditions are simple inequalities. Thus, the automatic tactics NUM_LE_CONV can be used to simplify the current goal greatly.

In addition, Kinematics calculations often need the computation of angles between primitives that are the same grade like a pair of lines or a pair of planes.The angle can be calculated based on the inner product of their normalized OPNS representations:

$$\theta = \angle(o_1, o_2) = \arccos \frac{o_1^* \cdot o_2^*}{|o_1^*|\,|o_2^*|} \tag{12}$$

According to Definition 11, we can define this formula as follows, and using the definition we can build goals to reasoning the angle between two planes or two lines.

Definition 16. Angles between two primitives with same grades

```
|- ∀ o1 o2 k. vector_angles_CGA o1 o2 ↔
     (isblade k o1)/\( isblade k o2) → vector_angles_CGA o1 o2 = acs((o1 inner o2)$${} /
     (mult_norm o1 * mult_norm o2))
```

4.5. Rigid Transformations

Arbitrary rigid body motion and transformation in the Euclidean space can be represented and calculated in CGA in a very simple way. Motion transforms based on CGA have the property of associativeness, which is of great significance for the expression of complex movements. It can achieve not only a simple transformation such as translation and rotation but also a complex one unified by a simple combination. This method has been widely used in various fields [27,37].

Transformations can be easily described in CGA using the geometric product. All kinds of transformations of an object o can be defined in such a way:

$$o_{transformed} = V o \tilde{V} \tag{13}$$

where V is a versor mainly including rotor, translator and motor. $\tilde{\boldsymbol{V}}$represents its reverse which has been defined in Sect. 3.

4.5.1. Rotation. The operator called rotor are defined by

$$R = \cos\left(\frac{\phi}{2}\right) - L \sin\left(\frac{\phi}{2}\right) \tag{14}$$

where L is the rotation axis represented by a normalized bivector and $\phi$is the rotation angle around this axis. Here, L can be an arbitrary line not necessarily going through the origin.

Definition 17. Rotor
| − ∀ t l. rotation_CGA t l = cos(t / &2) % mbasis{} - l * (sin(t / &2) % mbasis{})

Next, we define the pure rotation of an object according to Eq. (13) as follows:

Definition 18. Pure rotation of an object x
| − ∀ x t l. pure_rotationed_CGA x t l = (rotation_CGA t l) * x * (reversion(rotation_CGA t l))

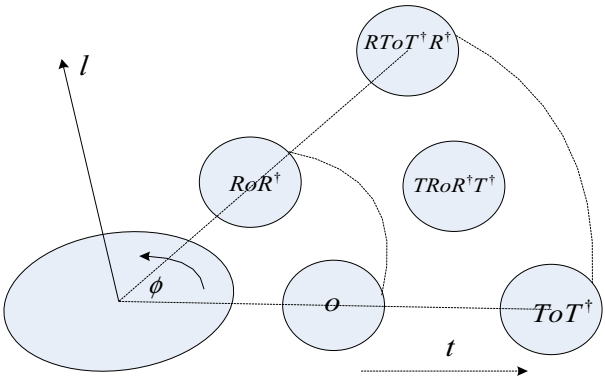


FIGURE 4. The rigid body motion of a sphere

The function **rotation_CGA** receives a real number and a multivector which represent the rotation angle and rotation axis respectively. The function **pure_rotationed_CGA** is based on Definitions 4, 7 and 17.

4.5.2. Translation. The operator called translator can be described according to the property $\mathbf{e}_{\infty}^2 = 0$results in

$$T = 1 - \frac{1}{2}\mathbf{t}e_{\infty} \tag{15}$$

where **t** is a 3D vector representing the translation direction an-d length.

Definition 19. Translator
| − ∀ t. Translation_CGA t = mbasis{} - (&1 / &2) % (t * null_inf)

Definition 20. Pure translation of an object x
| − ∀ t. pure_translationed_CGA x t = (Translation_CGA t) * x * (reversion(Translation_CGA t))

4.5.3. Rigid body motion. In CGA, 3D motions include both rotation and translation. A rigid body motion is described by an operator M called motor

$$M = TR \tag{16}$$

In order to describe the rigid body motions in CGA more clearly, Fig. 4 shows the rigid body motion of a sphere as follows.

Definition 21. Motor
| − ∀ a l t. motor_CGA a l t = Translation_CGA t * rotation_CGA a l

Definition 22. Rigid body motion of an object x
| − ∀ x a l t. rigid_body_motion x a l t = (motor_CGA a l t) * x * (reversion(motor_CGA a l t))

The formalization, presented in Sects. 3 and 4, has to be done in an interactive way due to the uncertainty of higher-order logic. One of the major challenges during this formalization is the non-availability of the detailed proof steps of the algebraic operations in the framework of geometric algebra

$Cl_{p,q}$. The mathematical theories of the algebraic operation rules of $Cl_{p,q}$ is relatively complex, and so the definitions of the three main products are abstract and complicate as well. In our formalization, many multivector-related theorems and some lemmas in CGA have to be formally verified, and such verifications are unavailable in the current version of HOL-Light. Some of these results of common interest are given below.

Lemma 1. *Componentwise operations on multivectors*

```
|- ∀ x y s c. s SUBSET 1..dimindex (:p) + dimindex (:q) ==> (c % x - c % y) $$ s = (c % x) $$ s -(c
   % y) $$ s
```

Lemma 2. *Geometric products on basis in* $Cl_{p,q}$

```
|-∀s t. mbasis s * mbasis t =
    if s SUBSET 1..(dimindex(:p) + dimindex(:q)) /\
       t SUBSET 1..(dimindex(:p) + dimindex(:q))
     then (--(&1) pow CARD {i,j | i IN s /\ j IN t /\ i > j} *
           --(&1) pow CARD {i,j | i IN (dimindex(:p) + 1)..(dimindex(:p) + dimindex(:q))/\
                                  j IN (dimindex(:p) + 1)..(dimindex(:p) + dimindex(:q)) /\
                                  i IN s /\ j IN t /\ i = j})
          % mbasis((s DIFF t) UNION (t DIFF s))
    else vec 0
```

Lemma 3. *The dual of the dual of a mutivector in CGA* $|-\forall\ x.\ DUAL\ (DUAL\ x) = -\ x$

Lemma 4. *sum of the numbers of diagonal elements at the top right in a square matrix*
$|-\forall$ N. CARD {i,j | i IN *1...*N $\bigwedge$ j IN *1...*N $\bigwedge$ i > j} = (N*(N-1)) DIV 2

5. Application: Formal Analysis of Robot Manipulation

In this section, the conformal geometric control technique is employed to verify a robotic grasping algorithm [3,19]. The goal of the algorithm is to move a gripper to an object. Both the gripper and the object can be described by a circle. Here, the reference circles instead of the points are used to compute in the 3D rigid transformation, which is efficient using CGA transformation. First, we compute the grasping circle in the object. Next, the griper circle can be estimated. Finally, the translation and rotation is calculated. In order to present all the steps clearly, the methodology is specially described in Algorithm 1 as follows. In addition, the main steps of the computing process are also shown in Fig. 5.

Using the formalization of conformal geometric algebra theories, these steps in Algorithm 1 can be formally modeled in HOL-Light depicted in Fig. 6.

The representations of the basic geometric entities, their combinations, intersections, distance and transformations in CGA are used in Fig. 6. The OPNS representation is utilized to deal with the computation of the base circle Z_b^*, the auxiliary plane π_h^* and the translation axis l_T^*. The IPNS representation of sphere is applied to define the auxiliary sphere S_h. Two kinds of

Algorithm 1. Rapid prototyping of the robotic grasping algorithm

Input: Four points (x_1,x_2,x_3,x_4) to identify the object to be grasped. (x_1,x_2,x_3)are in the base and one more (x_4) is at the top of the object
The griper circle with center at p_h and radius ρ
Two points a and b on the griper

Output: New position of the griper circle.

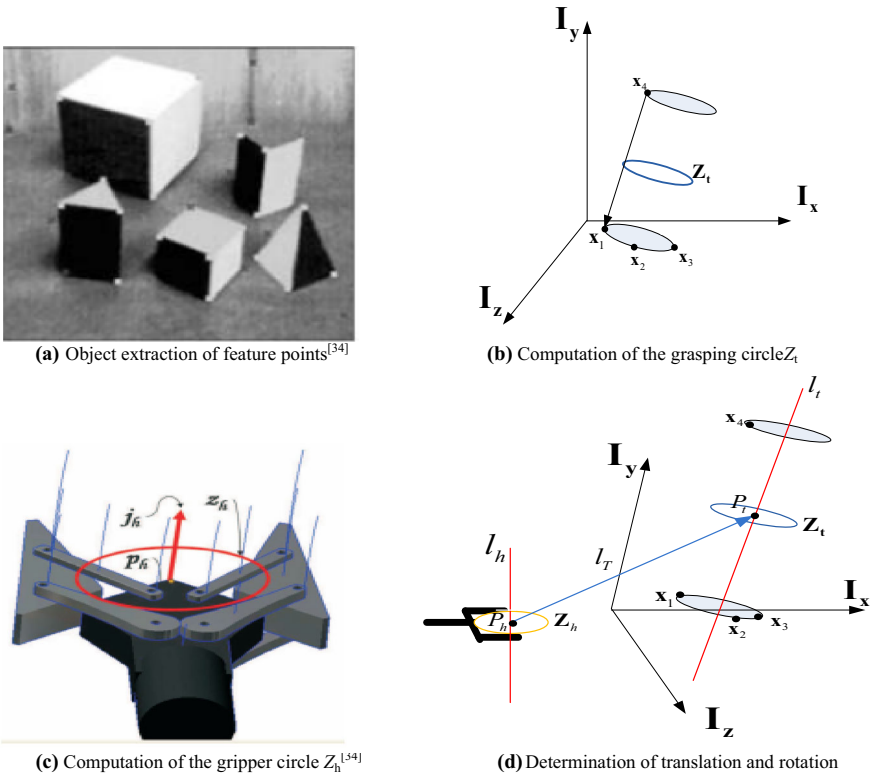
1.1 % Compute the grasping circle
1.2 $Z_b^* = x_1 \wedge x_2 \wedge x_3$ *% the base circle using these point*
1.3 $\pi_b = (Z_b^* \wedge e_\infty) I_c$ *% the base plane*
1.4 $T = 1 + \frac{1}{4}(\pi_b \cdot x_4)\pi_b \boldsymbol{e}_\infty$ *% the object grasped in the middle*
1.5 $\mathrm{Z}_t = T\mathrm{Z}_b\tilde{T}$ *% the grasping circle*
1.6 % Compute the griper circle
1.7 $S_h = P_h - \frac{1}{2}\rho^2 e_\infty$ *% creating a sphere with center at p_h and radius ρ*
1.8 $\pi_h^* = P_h \wedge a \wedge b \wedge e_\infty$ *% the auxiliary plane*
1.9 $\mathrm{Z}_h = S_h \wedge \pi_h$ *% the griper circle*
1.10 % Determination of translation
1.11 $P_t = \mathrm{Z}_t e_\infty \mathrm{Z}_t$ *% the center point P_t of a circle Z_t*
1.12 $l_T^* = P_h \wedge P_t \wedge e_\infty$ *% the translation axis*
1.13 $\mathrm{d} = |l_T^*| = dist(P_h, P_t)$ *% the distance d between the circles*
1.14 % Determination of rotation
1.15 $l_h^* = Z_h \wedge e_\infty$ *% the auxiliary axes of griper circle*
1.16 $l_t^* = Z_t \wedge e_\infty$ *% the auxiliary axes of grasping circle*
1.17 $\pi_{th}^* = l_t^* \wedge \left(l_h^*(e_0 \wedge e_\infty)\right)$ *% axes l_h and l_t yield in the plane π_{th}^**
1.18 $l_r^* = P_h \wedge \pi_{th} \wedge e_\infty$ *% the rotation axis*
1.19 $\theta = \arccos \frac{l_t^* \cdot l_h^*}{|l_t^*||l_h^*|}$ *% the angle θ between circles*
1.20 % Calculate the new position of the griper
1.21 $R = \cos\left(\frac{\theta}{2}\right) - l_r \sin\left(\frac{\theta}{2}\right)$ *% the rotor*
1.22 $T = 1 - \frac{1}{2} d l_T e_\infty$ *% the translator*
1.23 $\mathrm{Z}_h' = TR\mathrm{Z}_h\tilde{R}\tilde{T}$ *% the new position of the griper*

representation can be transformed using the dual operator, which have been used to reason about the correctness of the dual relationship in Theorem 3 and Lemma 3. The distance between the circles is calculated based on that between two points (see Theorem 6). Definition 16 is employed to calculate the rotation angles. In addition, not only the simple transformations such as translation and rotation but also the combined ones in Sect. 4 are used to determine the grasping circle Z_t and the new position of the griper$\mathrm{Z}_h^{'}$.

Finally, we are able to move the gripper circle $\mathrm{Z_h}$ step by step towards the grasping circle $\mathrm{Z_t}$. The manipulator moves the grasper to the object and they should fulfill some conditions to grasp the object firmly. According to the above algorithm, we can verify the condition in HOL-light that the new position of the griper $\mathrm{Z_h}$should be equal to that of the grasping circle $\mathrm{Z_t}$. Such a goal can be described as follows:

∀ x1 x2 x3 x4 ph r a b. circle_zh_new x1 x2 x3 x4 ph r a b = circle_zt x1 x2 x3 x4

Substituting the above formal formulations into the goal, we can obtain about 100 lines of codes in HOL-Light. Then, after rewriting



(a) Object extraction of feature points[34]

(b) Computation of the grasping circle Z_t

(c) Computation of the gripper circle Z_h[34]

(d) Determination of translation and rotation

FIGURE 5. The main steps of the computing process

the above mentioned functions such as **sphere_CGA** (Representation 2), **pure_translationed_CGA** (Definition 20) and **DUAL** (Definition 10), we can continue to unfold the goal. In the process, there are a lot of computations about the geometric product, inner product and outer product on multivectors. So, first we need to use their linear properties such as distributive addition and commutative scalar multiplication to expand the current goal until the product of the basis and singleton is obtained. Then, the three products are required to simplify the goal. For example, Theorems 1 and 2, the Lemmas in Table 2 and Lemma 2 are needed in the simplification. In addition, some properties of the outer product in Harrison's Clifford library such as OUTER_ACI can help us to greatly simplify the computation of the outer product of the basis and singleton. During the process of verification, we need to prove many calculation related lemmas such as the dual of a base vector in CGA to simplify the goal. Meanwhile, a lot of simple arithmetic reasoning on vectors is also required by the proof of the goal. The example of the robotic grasping algorithm clearly verifies the usefulness of the CGA method in conducting formal analysis on real-world applications.

```
(*Compute the grasping circle *)
|- dual_circle_zb x1 x2 x3 = (circle_direct_CGA (point_CGA x1)(point_CGA x2)(point_CGA
x3))
|- plane_b x1 x2 x3 = ((dual_circle_zb x1 x2 x3) outer null_inf) * pseudo
|- circle_zt x1 x2 x3 x4 = pure_translationed_CGA ( -- DUAL (dual_circle_zb x1 x2 x3)) (--(&1
   / &2) % ((plane_b x1 x2 x3) inner (point_CGA x4)))

(*Compute the griper circle*)
|- sphere_Sh ph r = sphere_CGA ph r
|- dual_plane_pih ph a b = plane_direct_CGA (point_CGA ph)(point_CGA a)(point_CGA b)
|- circle_zh ph r a b = (sphere_Sh ph r) outer ( -- DUAL (dual_plane_pih ph a b))

(*Estimation of translation *)
|- center_point_pt x1 x2 x3 x4 = (circle_zt x1 x2 x3 x4) * null_inf * (circle_zt x1 x2 x3 x4)
|- dual_translation_axis ph x1 x2 x3 x4 = line_direct_CGA (point_CGA ph) (center_point_pt x1
  x2 x3 x4 )
|- distance ph x1 x2 x3 x4 = -- sqrt (&2 * ((point_CGA ph) inner (center_point_pt x1 x2 x3
   x4))$${})

(*Estimation of rotation *)
|- dual_lh ph r a b = (circle_zh ph r a b) outer null_inf
|- dual_lt x1 x2 x3 x4 = (circle_zt x1 x2 x3 x4) outer null_inf
|- dual_plane_th x1 x2 x3 x4 ph r a b = (dual_lt x1 x2 x3 x4) outer ((dual_lh ph r a b ) * (null_zero
   outer null_inf))
|- dual_rotation_axis x1 x2 x3 x4 ph r a b = (point_CGA ph) outer (-- DUAL(dual_plane_th x1 x2
   x3 x4 ph r a b)) outer null_inf
|- rotation_angle x1 x2 x3 x4 ph r a b = vector_angles_CGA (dual_lh ph r a b ) (dual_lt x1 x2 x3
   x4)

(*calculate the new position of the griper *)
|- circle_zh_new  x1 x2 x3 x4 ph r a b = pure_translationed_CGA
   (pure_rotationed_CGA (circle_zh ph r a b)(rotation_angle x1 x2 x3 x4 ph r a b) (-- DUAL
   (dual_rotation_axis x1 x2 x3 x4 ph r a b )))
   ((distance ph r a b x1 x2 x3 x4) % (-- DUAL (dual_translation_axis ph r a b x1 x2 x3 x4)))
```

FIGURE 6. Formalization of the Algorithm 1

6. Conclusions

This article presents a comprehensive methodology to reason about CGA in a theorem prover. As the first step, we provide the formalization of its classical algebraic operation theories. Using this formalization, we verify the CGA representation of geometric entities and its geometric features like IPNS, OPNS, distance and transformations in higher-order logic. The main concepts of CGA in our formalization include geometric product, outer product, inner product, dual, inverse, null vectors and motor. Based on this work, we continue to conduct formal analysis on the grasping algorithm of a robot, and the formal analysis reveals the practicability of our formalization of CGA.

Overall our formalization requires more than 4000 lines of codes. This shows the non-triviality of our work in simplifying the formal analysis of conformal geometric algebra systems. The main difficulties encountered are the enormous amount of user intervention required due to the uncertainty of the higher-order logic. We overcome this limitation by formalizing the

CGA theory in higher-order logic and minimizing the user guidance in the reasoning process by building the system on available results.

The reported formalization in the paper opens the doors to many novel and promising directions of related researches. The formalization of the new framework of CGA foundations can be directly used to model, represent and stimulate new efficient algorithms [1] in all areas of science dealing with geometric properties like robotics. What's more, our formalization method can also be used to formalize other mathematical language, because some mathematical systems such as complex numbers, Plücker coordinates, quaternions and dual quaternions can be identified in CGA [20]. In addition, our formalization of geometric algebra $\mathrm{Cl}_{p,q}$ is a more abstract algebraic framework, which is instructive for the further study of other algebra and geometry to some degree. For instance, spacetime algebra (STA) [16] can be written as $\mathrm{Cl}_{1,3}$.

Acknowledgments

This work was supported by the International Cooperation Program on Science and Technology (2011DFG13000), the National Natural Science Foundation of China (61170304, 61472468, 61572331), the Project of Beijing Municipal Science & Technology Commission (Z141100002014001), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (No.IDHT20150507), and the Scientific Research Base Development Program of the Beijing Municipal Commission of Education (TJSHG201310028014).

References

[1] Aristidou, A., Lasenby, J.: FABRIK: a fast, iterative solver for the inverse kinematics problem. Graph. Models **73**(4), 243–260 (2011)

[2] Bayro-Corrochano, E., Bernal-Marin, M.: Generalized hough transform and conformal geometric algebra to detect lines and planes for building 3D maps and robot navigation. In: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, pp. 810–815 (2010)

[3] Bayro-Corrochano, E., Zamora-Esquivel, J.: Differential and inverse kinematics of robot devices using conformal geometric algebra. Robotica **25**(1), 43–61 (2007)

[4] Blanchette, J.C., Nipkow, T.: Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder. Lecture Notes in Computer Science, pp. 131–146 (2010)

[5] Buchholz, S., Sommer, G.: Introduction to neural computation in Clifford algebra. In: Sommer, G. (ed.) Geometric Computing with Clifford Algebras, pp. 291–314. Springer-Verlag, Heidelberg (2001)

[6] Carbajal-Espinosa, O., Osuna-Gonzalez, G., Gonzalez-Jimenez, L. et al.: Visual servoing and robust object manipulation using symmetries and conformai geometric algebra. In: Humanoid Robots (Humanoids), IEEE-RAS International Conference on, pp. 1051–1056 (2014)

[7] Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object Oriented approach to Geometry. Morgan Kaufmann, San Francisco (2007)

[8] Franchini, S., Gentile, A., Sorbello, F. et al.: ConformalALU: a conformal geometric algebra coprocessor for medical image processing. IEEE Trans. Comput. **64**(4), 955–970 (2015)

[9] Gonzalez-Jimenez, L.E., Carbajal-Espinosa, O.E., Bayro-Corrochano, E.: Geometric techniques for the kinematic modeling and control of robotic manipulators. In: Proc. of IEEE ICRA, pp. 5831–5836 (2011)

[10] Han, D.S., Yang, Q.L., Xing, J.C.: UML-Based modeling and formal verification for software self-adaptation. J. Softw. **26**(4), 730–746 (2015)

[11] Harrison, J.: HOL Light: A Tutorial Introduction. Lecture Notes in C Computer Science, pp. 265–269 (1996)

[12] Harrison, J.: Towards Self-verification of HOL Light. Lecture Notes in Computer Science, pp. 177–191 (2006)

[13] Harrison, J.: https://code.google.com/p/hol-light/source/browse/trunk/Multivariate/clifford.ml (2010)

[14] Harrison, J.: The HOL light theory of Euclidean space. J. Autom. Reasoning **50**(2), 173–190 (2013)

[15] Hasan, O., Ahmad, M.: Formal analysis of steady state errors in feedback control systems using HOL-light. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 18–22 March, Grenoble, France, pp. 1423–1426 (2013)

[16] Hestenes, D.: Spacetime physics with geometric algebra. Am. J. Phys. **71**(7), 691–714 (2003)

[17] Hildenbrand, D., Bayro-Corrochano, E., Zamora, J.: Advanced geometric approach for graphics and visual guided robot object manipulation. In: Proc. of IEEE ICRA, pp. 4727–4732 (2005)

[18] Hildenbrand, D.: Geometric computing in Computer Graphics and Robotics using Conformal Geometric Algebra. PhD Thesis, Technische Universitaet Darmstadt (2007)

[19] Hildenbrand, D., Zamora, J., Bayro-Corrochano, E.: Inverse kinematics computation in computer graphics and robotics using conformal geometric algebra. Adv. Appl. Clifford Algebras **18**(3), 699–713 (2008)

[20] Hildenbrand, D.: Foundations of geometric algebra computing. Geom. Comput. **1479**(4), 27–30 (2012)

[21] Hildenbrand, D., Koch, A.: Gaalop: high performance computing based on conformal geometric algebra. Geom. Comput. (2013)

[22] Hitzer, E., Perwass, C.: Interactive 3D space group visualization with CLUCalc and the clifford geometric algebra description of space groups. Adv. Appl. Clifford Algebras **20**(4), 631–658 (2010)

[23] Kim, J.S., Jin, H.J., Park, J.H.: Inverse kinematics and geometric singularity analysis of a 3-SPS/S redundant motion mechanism using conformal geometric algebra. Mech. Mach. Theory **90**, 23–36 (2015)

[24] Klein, G., Elphinstone, K., Heiser, G., et al.: seL4: Formal Verification of an OS Kernel. Acm Symposium on Operating Systems Principles, pp. 207–220 (2009)

[25] Li, H.: Conformal geometric algebra and algebraic manipulations of geometric invariants. J. Comput. Aided Des. Comput. Graph. **18**(7), 902–911 (2006)

[26] Li, H.: Automated theorem proving in the homogeneous model with Clifford bracket algebra. In: Dorst, L. (eds.) Applications of Geometric Algebra in Computer Science and Engineering, pp. 69–78. Boston (2002)

[27] Li, M., Guan, J.: Possibilistic C-Spherical Shell clustering algorithm based on conformai geometric algebra. Signal Processing (ICSP), IEEE 10th International Conference on, pp. 1347–1350 (2010)

[28] Li, H., Hestenes, D., Rockwood, A.: A universal model for conformal geometries of euclidean, spherical and double-hyperbolic spaces. Geom. Comput. Clifford Algebras, 77–104 (2001)

[29] Lopez-Franco, C., Arana-Daniel, N., Bayro-Corrochano, E.: Vision-based robot control with omnidirectional cameras and conformal geometric algebra. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2543–2548 (2010)

[30] Lounesto, P.: Clifford algebras and Hestenes spinors. Found. Phys. **23**(9), 1203–1237 (1993)

[31] Oviedo-Barriga, J., Carbajal-Espinosa, O., Gonzalez-Jimenez, L. et al.: Robust tracking of bio-inspired references for a biped robot using geometric algebra and sliding modes. In: Proc. of IEEE ICRA, pp. 5317–5322 (2013)

[32] Pham, M.T., Tachibana, K., Yoshikawa, T. et al.: A clustering method for geometric data based on approximation using conformal geometric algebra. In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 2540–2545 (2011)

[33] Resten, Y., Maler, O., Marcus, M. et al.: Symbolic model checking with rich assertional languages. Comput. Aided Verification **43**(2), 424–435 (2006)

[34] Shirokov, D.: Calculation of elements of spin groups using generalized Pauli's theorem. Adv. Appl. Clifford Algebras **25**(1), 227–244 (2014)

[35] Siddique, U., Hasan, O.: On the formalization of gamma function in HOL. J. Autom. Reason. 407–429 (2014)

[36] Suter, J.: Geometric Algebra Primer. Available: http://www.jaapsuter.com (2003)

[37] Wang, C., Wu, H., Miao, Q.: Inverse kinematics computation in robotics using Conformal Geometric Algebra. Technology and Innovation Conference (ITIC), International pp. 1–5. IET (2009)

[38] Wu, W.J.: Basic principles of mechanical theorem proving in elementary geometries. J. Autom. Reason. **2**(3), 221–252 (1986)

Sha Ma, Zhiping Shi, Zhenzhou Shao, Yong Guan, Liming Li, Yongdong Li
College of Information Engineering
Capital Normal University
Beijing
China
e-mail: shizp@cnu.edu.cn

Sha Ma
e-mail: masha@cnu.edu.cn

Zhenzhou Shao
e-mail: zshao@cnu.edu.cn

Yong Guan
e-mail: guanyong@cnu.edu.cn

Liming Li
e-mail: liliminga@126.com

Yongdong Li
e-mail: lydbeijing@163.com

Zhiping Shi, Yong Guan, Liming Li
Beijing Advanced Innovation Center for Imaging Technology
Beijing
China

Zhenzhou Shao, Yongdong Li
Beijing Key Laboratory of Light Industrial Robot and Safety Verification
Beijing
China

Received: June 22, 2015.
Accepted: February 25, 2016.