Clifford algebra support in Maxima

Dimiter Prodanov[1]

[1] Department of Environment, Health and Safety, Neuroscience Research Flanders, Imec, Leuven, Belgium;

dimiterpp@gmail.com; dimiter.prodanov@imec.be

Applied Geometric Algebra in Computer Science and Engineering 2015, Barcelona, July 29 – 31, 2015

Abstract

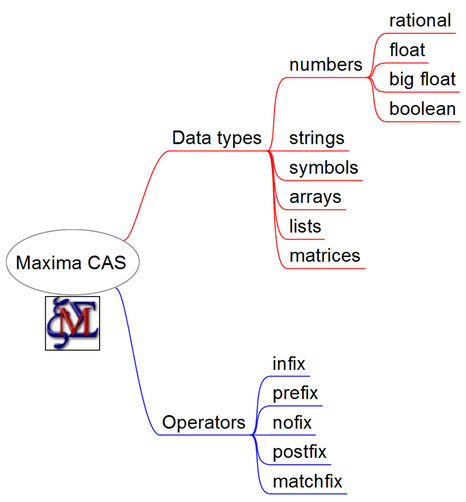
Maxima is the open source descendant of the first ever computer algebra system. It is written entirely in Lisp and is distributed under GNU General Public License. At present there are two packages which support Clifford algebras. *Maxima* has its own programming language, which is particularly well suited for handling formal mathematical expressions. The scripts can be compiled also to Lisp within the program itself. The package *atensor* partially implements generalized (tensor) algebras, including Clifford, Grassmann, and Lie-algebras. It is distributed as a shared package with the main program distribution. The package *clifford*, authored by the presenter, implements specifically Clifford algebras. *Clifford* relies heavily on the rule-based simplification system of Maxima for simplification of Clifford products, outer products, scalar products and inverses.

Introduction

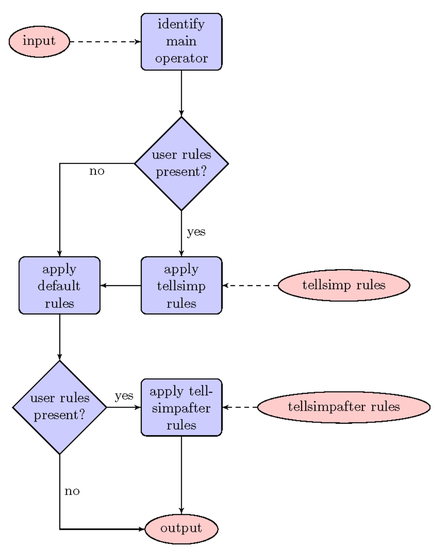
Maxima is the descendant of the first ever computer algebra system, MACSYMA. Maxima is derived from the *Macsyma* system, developed at MIT in the years 1968 – 1982. *Maxima* is written entirely in *Lisp* and is distributed under GNU General Public License. At present the system is supported by a team of volunteer developers. *Maxima* has its own programming language, which is particularly well suited for handling formal mathematical expressions. The scripts can be compiled also to Lisp binaries within the program itself. Lisp programs can be also loaded and addressed within the system at runtime. The system also offers the possibility of running batch tests and demonstrations.

Expression processing in Maxima

A



B



Maxima follows Lisp principle that program code (i.e. expressions) can be used as an input for computations. An *expression* contains a sequence of operators, numbers and symbols (Fig. 1A). The value of an expression is the value of the last assigned member. **In such way every expression in Maxima is a lambda construct**.

Different transformation rules can be associated with any given operators in Maxima. In addition it is possible to assign also transformation rules to general symbols. Maxima has an advanced pattern matching mechanism, which supports such operations. Rules can be add to the built-in simplifier in two ways: by using the commands *tellsimp* or *tellsimpafter*. *tellsimp* rules are applied when appropriate, based on the main operator, before the built-in simplification while *tellsimpafter*, are applied after the built in simplification (Fig. 1B). The augmented simplification is then treated as built in, so subsequent tellsimp rules are applied before the previous ones.

The simplifier subroutine operates by descending the tree of an expression until it gets to atoms, and then simplifies the smallest pieces and backs out.

Clifford Algebra Implementation

The package *atensor* authored by Viktor Toth partially implements generalized (tensor) algebras, including Clifford, Grassmann, and Lie-algebras. It is distributed as a shared package with the main program distribution. The package comes also with built -in support for complex, quaternion, Pauli and Dirac algebras.

The package *clifford*, designed by the present author, implements only on Clifford algebras. In contrast to *atensor*, *clifford* relies extensively on the Maxima simplification routines and is fully integrated into the command prompt evaluation. The package defines several geometric product simplification rules which are integrated in the built-in Maxima simplifier (see Listing 1). The code is distributed under GNU Lesser General Public License from GitHub `http://dprodanov.github.io/clifford/`. *Clifford* defines multiple rules for pre- and post-simplification of Clifford products, outer products, scalar products, inverses and powers of Clifford vectors.

Table 1: Main functions in *Clifford*

method name	functionality
dotsimpc (ab)	simplification of dot-products
dotinvsimp (ab)	simplification of inverses
powsimp (ab)	simplification of exponents
cliffsimpall (expr)	full simplification of expressions
cinv (ab)	Clifford inverse of element
grade (expr)	grade decomposition of expressions
ctranspose (ab)	Clifford transpose of product
creverse (ab)	Clifford reverse of product
cinvolve (expr)	Clifford involution of expressions
cconjugate (expr)	Clifford conjugate of expressions
cnorm(expr)	squared norm of expression

The inner (dot) and outer (wedge) products of two vectors a and b are defined in Table 2:

operator	notation	definition
$\sim$	$a \circ b$	$\frac{a.b + b.a}{2}$
&	$a \wedge b$	$\frac{a.b - b.a}{2}$

Listing 1: Simplification rules in *Clifford*

```
/*
simplification rules
*/
matchdeclare(dd, lambda([u],  freeof(asymbol,u)));
defrule (clifsimp1, dd*aa, dd*dotsimpc(aa));
defrule (clifsimp10, aa, dotsimpc(aa));
defrule (clifsimp2, dd/aa, dd*dotinvsimp(1/aa));

/*
full simplification of expressions
*/
declare (cliffsimpall, evfun);
cliffsimpall(expr):=block([res],
res:expand(expr),
res:apply1(res, clifsimp2, clifsimp1, clifsimp10),
ratsimp(res)
);
```

Demo: Quaternion algebra with *Clifford*

Anti-commutativity

$e[2].e[1]$

$-e_1.e_2$

Simplification of inverses

$1/(1+e[1]), cliffsimpall;$

$$-\frac{-1+e_1}{2}$$

Computation of Quaternion multiplication table

$clifford(e, 0, 2, 0);$

$[-1, -1]$

$mtable2();$

$$\begin{pmatrix} 1 & e_1 & e_2 & e_1.e_2 \\ e_1 & -1 & e_1.e_2 & -e_2 \\ e_2 & -e_1.e_2 & -1 & e_1 \\ e_1.e_2 & e_2 & -e_1 & -1 \end{pmatrix}$$

Inner product calculation

$(1+e[1]) \sim (1+e[1]), expand;$

$2\,e_1$

Outer product calculation

$e[1]\ \&\ e[2];$

$e_1.e_2$

$(1+e[1])\ \&\ (1+e[1]), expand;$

0

Computation of Quaternion inverses

```
block(declare([a,b,c,d],scalar),
cc:a+b*e[1]+c*e[2]+d*e[1] . e[2],
dd:cinv(cc)
);
```

$$\frac{a - e_1\,b - e_2\,c - (e_1.e_2)\,d}{a^2 + b^2 + c^2 + d^2}$$

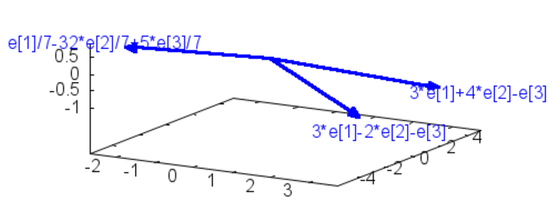
Checking the obtained result:

$ev(dd.cc, expand, ratsimp)$

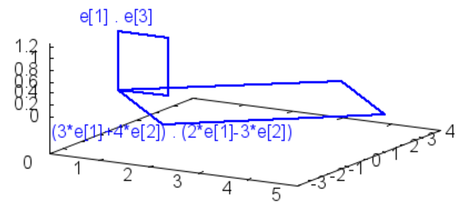
1

Demo: Vector and bi-vector display in *Clifford*

Reflection of the vector $a = 3\,e_1 + 4\,e_2 - e_3$ by $b = 3\,e_1 - 2\,e_2 - e_3$



Rotation of Bi-vectors spanned by the vectors $Span\,\{e_1, e_2\}$ and $Span\,\{3e_1 + 4e_2, 2e_1 - 3e_2\}$



References

[1] L. Dorst. *Geomteric Computing with Clifford Algebras*, chapter Honing Geometric Algebra for Its Use in the Computer Sciences. Springer, 2001.

[2] Maxima Project, http://maxima.sourceforge.net/docs/manual/maxima.html. *Maxima 5.35.1 Manual*, Dec 2014.

[3] Viktor Toth. Tensor manipulation in GPL Maxima. 2007.

Acknowledgements

The work has been supported in part by a grant from Research Fund - Flanders (FWO), contract number 0880.212.840.

Fonds Wetenschappelijk Onderzoek Vlaanderen
Opening new horizons

imec