

# Brackets and Projective Geometry

Tim Duff

May 8, 2023

The goal of this project will be to develop the package-stub **Brackets** for Macaulay2. The source code for this package may be found at the following url (parallel to this document, on the appropriate branch):

<https://github.com/Macaulay2/Workshop-2023-Minneapolis/blob/brackets/Brackets/Brackets.m2>

Here I will provide a quick introduction to the mathematics involved, the current functionality of the package, and some tasks/extensions we can pursue during the course of the workshop.

## 1 Mathematical Background

The following notes highlight the main ideas in Chapter 3 of the following book:

Sturmfels, Bernd. *Algorithms in invariant theory*.  
Springer Science & Business Media, 2008.

Our notation mostly follows that in Sturmfels' book. Fix integers  $n \geq d \geq 1$ , and let  $X = (x_{ij})$  be an  $n \times d$  matrix of distinct variables in the polynomial ring  $k[x_{ij}]$  over a fixed field  $k$ . We think of each row of  $X$  as a point in the projective space  $\mathbb{P}^{d-1}$  of dimension  $(d-1)$  over  $k$ , so that  $X$  as represents a configuration of  $n$  points in this projective space. Many interesting geometric properties of this point configuration can be expressed in terms of the maximal minors of  $X$ . For notational convenience, it is common to write these minors in *bracket notation*. A bracket is just a formal expression  $[\lambda_1 \lambda_2 \dots \lambda_d]$  representing the minor of  $X$  whose rows are given by indices  $1 \leq \lambda_1 < \lambda_2 < \dots < \lambda_d \leq n$ .

**Example 1.** Let  $n = 4, d = 3$ . In the  $4 \times 3$  matrix

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \end{pmatrix},$$

each row represents a point in the projective plane  $\mathbb{P}^2$ . There are  $\binom{4}{3} = 4$  brackets, namely  $[123], [124], [134], [234]$ . The condition that three of the four points are collinear may be expressed by the bracket equation

$$[123][124][134][234] = 0.$$

**Example 2.** Let  $n = 6, d = 2$ , so that

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \\ x_{5,1} & x_{5,2} \\ x_{6,1} & x_{6,2} \end{pmatrix}.$$

There are  $6 = \binom{4}{2}$  brackets, and the matrix  $X$  represents a configuration of 6 points on the projective line  $\mathbb{P}^1$ . Unlike the previous example, the brackets are no longer algebraically independent, as they satisfy the quadratic Plücker relation,

$$[12][34] - [13][24] + [14][23] = 0. \quad (1)$$

**Exercise 1** (For beginners). Verify (1) using core Macaulay2 commands.

A monomial in brackets is called a *tableau*, and can be visualized as an array of integers,

$$T = \begin{pmatrix} \lambda_1^1 & \dots & \lambda_1^d \\ \vdots & & \vdots \\ \lambda_k^1 & \dots & \lambda_k^d \end{pmatrix}$$

An expression in brackets is said to be *straightened* if every tableau appearing in it has its columns sorted. The classical straightening algorithm rewrites an expression in brackets into an equivalent, straightened expression modulo the kernel of the ring map

$$\psi_{n,d} : k[[\lambda_{i_1} \cdots \lambda_{i_d}] \mid 1 \leq i_1 < \dots < i_d \leq n] \rightarrow k[X].$$

In modern times, this rewriting algorithm falls under the heading of Gröbner bases. The image of the map  $\psi_{n,d}$  is called the *Bracket ring*, denoted  $B_{n,d}$ . The fundamental theorem of invariant theory (Theorem 3.2.1) states, for  $k = \mathbb{C}$ , that  $B_{n,d}$  is the ring of polynomial invariants for the action of the group  $\mathrm{SL}(\mathbb{C}^d)$  by right-multiplication of  $X$ . This implies a connection between  $B_{n,d}$  and the geometry of point configurations. Further connections may be obtained via the *Grassmann-Cayley algebra*.

The Grassmann-Cayley algebra may be viewed as an algebra of linear subspaces of  $\mathbb{P}^{d-1}$ . In this algebra, there are two operations which correspond to the join and meet of subspaces. In Macaulay2, we denote these operators by  $\cdot$  and  $\wedge$ , respectively.<sup>1</sup> The first operator is simply multiplication in a skew-commutative polynomial ring  $\mathbb{C}\langle a_1, \dots, a_n \rangle$ . An algebraic formula for the meet operator is more complicated, but it can be defined using the *shuffle product*. As a  $k$ -vector space, the Grassmann-Cayley algebra has a direct-sum decomposition

$$\bigoplus_{k=0}^d \Lambda^k(a_1, \dots, a_n),$$

where  $\Lambda^k(a_1, \dots, a_n)$  is the space of *extensors* of the form  $a_{i_1} \cdots a_{i_k}$ . We may identify  $\Lambda^d(a_1, \dots, a_n) \cong B_{n,d}$ .

<sup>1</sup>Sturmfels' book uses  $\vee$  and  $\wedge$ . Note that  $\vee$  may be viewed as the usual exterior product, which is denoted by  $\vee$  in many sources.

The following automatic proof of Desargues' theorem illustrates the usefulness of Grassmann-Cayley algebras and the straightening algorithm.

```
G = gc(a..f,3) -- Grassmann-Cayley algebra for 6 points in P^2
abLine = (a * b)_G -- line spanned by a and b
deLine = (d * e)_G -- line spanned by d and e
bcLine = (b * c)_G -- line spanned by b and c
efLine = (e * f)_G -- line spanned by e and f
acLine = (a * c)_G -- line spanned by a and c
dfLine = (d * f)_G -- line spanned by d and f
pt1 = abLine ^ deLine -- intersection of ab and de
pt2 = bcLine ^ efLine -- intersection of bc and ef
pt3 = acLine ^ dfLine -- intersection of ac and df
linePerspective = pt1 * pt2 * pt3 -- Condition that the pts p1, p2, p3 are collinear
adLine = (a * d)_G -- line spanned by a and d
beLine = (b * e)_G -- line spanned by b and e
cfLine = (c * f)_G -- line spanned by c and f
pointPerspective = adLine ^ beLine ^ cfLine -- Condition that the 3 lines above meet.
--
"pointPerspective" and "linePerspective" are two degree-0 elements of the Grassmann Cayley
algebra, which we identify with elements of the bracket ring B_(2,6).
The representatives of these elements do not share any common factors.
But, applying the straightening algorithm below produces two normal forms such that

[abc] * [def] * nf(linePerspective) = 2 * nf(pointPerspective)

So, if a,b,c are not collinear and d,e,f are not collinear,
line and point perspective are the same.
*-
netList factors linePerspective
netList factors pointPerspective
(n1, n2) = (normalForm pointPerspective, normalForm linePerspective);
netList factor n1
netList factor n2
```

## 2 Things to Work On

### 2.1 Beginner

1. Learn how to use the package by working out the following examples
  - (a) Pascal's theorem (Example 3.4.3.)
  - (b) Transversals to lines in  $\mathbb{P}^3$  (Example 3.4.5)
  - (c) Turnbull-Young invariant (Exercise 3.5.3. Note: you will probably need to implement one of the more efficient strategies suggested below.)

- (d) Synthetic resultant (Exercise 3.5.4.)
  - (e) Examples from Section 3.2.
  - (f) For each of the above, add them to the package's list of tests.
2. Tableaux
- (a) Write a class **Tableau** for bracket ring elements with one term.
  - (b) Write a function computing the weight of a tableau  $T \in k[\Lambda(n, d)]$ , which outputs the vector  $(w_1, \dots, w_n)$  where  $w_i$  counts the number of occurrences of the  $i$ -th vector symbol in  $B_{n,d}$ .
  - (c) Write a function which illustrates the individual steps of the straightening algorithm.
3. Write the following methods for Grassman-Cayley expressions (see Sec 3.5 for details): **isSimple**, **isHomogeneous**, **isMultilinear**.
4. Find code in the package that needs to be exported/tested/documented, and export/test/document it.

## 2.2 Intermediate

1. Implement new strategies for the constructor method **bracketRing**. The current implementation is the “classical” straightening algorithm that makes use of the “classical” straightening algorithm in the so-called tableaux order.
  - (a) Implement the “SAGBI algorithm” (Algorithm 3.2.8) using the package **SubalgebraBases**.
  - (b) Emulate the existing strategy, but use the function **Grassmannian** instead. You may need to re-order the  $x$ -variables!
  - (c) Using the command **forceGB**, code the Plücker ideal by its Gröbner basis of *van der Waerden syzygies*  $S_{n,d}^*$  on p83). Does it help to prune this Gröbner basis down further to a *reduced* Gröbner basis?
2. Understand how the data types **AbstractGCRing**, **BracketRing**, **GCAgebra** work. Do we need to implement any other methods for them?
3. Write a method **normalForm(RingElement, BracketRing)** that accepts a polynomial  $p(X)$  and computes expressions  $q, r$  such that

$$p(X) = q([\lambda] \mid [\lambda] \in B_{n,d}) + r(X)$$

By Theorem 3.2.1,  $p(X)$  is a  $\mathrm{SL}(\mathbb{C}^d)$ -invariant iff  $r(X) = 0$ .

4. Implement *multilinear Cayley factorization* (Algorithm 3.5.6). Some concrete first steps

- (a) Write a method that returns the tree representation of an object of class `GCEXpression`
- (b) Write a method that extracts the *atomic extensors* (p113) of a Grassman-Cayley expression.