

Specification: Bounded RC Factor Algebra

1 Overview

The **Bounded RC Factor Algebra** of size n is a $\mathbb{Z}$ -algebra whose additive basis consists of tensor monomials in full Grassmannian RC graphs of prescribed row counts. It provides a combinatorial framework for multiplying Schubert polynomials via squash products and normalized tensor keys. The construction builds on the Molev–Sagan type formula for double Schubert polynomials [1].

2 Additive Basis

Definition 1 (Full Grassmannian RC graph). An RC graph G with k rows is *full Grassmannian* if either $\pi(G) = \text{id}$ or $\text{des}(\pi(G)) = \{k\}$ (i.e. the associated permutation has at most one descent, occurring at position k).

Definition 2 (Basis keys). Fix an integer $n \geq 1$. A **basis key** of size n is a tensor

$$E_1 \otimes E_2 \otimes \cdots \otimes E_{n-1} \otimes G_n$$

where:

- For $1 \leq i \leq n-1$, each E_i is an i -row full Grassmannian RC graph whose permutation lies in S_{i+1} (equivalently, E_i is an elementary symmetric RC graph: $\pi(E_i) \in S_{i+1}$ with $\text{des}(\pi(E_i)) \subseteq \{i\}$).
- G_n is an n -row full Grassmannian RC graph (so $\text{des}(\pi(G_n)) \subseteq \{n\}$).

Any factor with $\pi = \text{id}$ may be omitted from the tensor (the identity RC graph acts as a unit under the squash product).

The algebra is the free $\mathbb{Z}$ -module on all such keys (with fixed size n), modulo the normalization relations described below.

3 Product

3.1 Squash product on RC graphs

Given RC graphs A (with m rows) and B (with k rows, $k \geq m$), the **squash product** $A \cdot_{\text{sq}} B$ is computed by:

1. Form the *disjoint union*: place B next to A by shifting its column indices into a sufficiently high range that the corresponding Schubert polynomials commute (i.e. the permutations act on disjoint positions). The row counts are unchanged; the number of rows of the result increases only to accommodate the combined code length.

2. Repeatedly *zero out the last row* (applying the transition/exchange property to absorb the bottom row into the upper rows) until the result has k rows.

The result is a k -row RC graph.

3.2 Key-to-RC-graph evaluation

A key $K = (F_1, F_2, \dots, F_m)$ of size n evaluates to a single RC graph by left-to-right squash product:

$$\text{RC}(K) = ((\dots((F_1 \cdot_{\text{sq}} F_2) \cdot_{\text{sq}} F_3) \dots) \cdot_{\text{sq}} F_m),$$

resized to n rows. Each intermediate accumulator is resized upward as needed before squashing.

3.3 Multiplication of keys

Given keys K_L and K_R of the same size n , their product key is:

$$K_L \cdot K_R = \text{normalize}(K_L, K_R),$$

where the concatenation $(F_1^L, \dots, F_a^L, F_1^R, \dots, F_b^R)$ is passed through the normalization procedure.

3.4 Normalization

The normalization loop transforms an arbitrary tuple of full Grassmannian RC graphs into the canonical form $E_1 \otimes \dots \otimes E_{n-1} \otimes G_n$. It repeats until stable:

1. **Sort and merge.** Sort factors by row count (ascending). Merge adjacent factors of the same row count via squash product. If a merged result is the identity, remove it.
2. **Strip identities.** Remove any factor with $\pi = \text{id}$ and normalize each remaining RC graph by snapping the number of rows to the length of the Lehmer code.
3. **Peel oversized factors.** Scan factors from right to left. If factor F_i has $\ell(F_i) < n$ rows but $\pi(F_i) \notin S_{\ell(F_i)+1}$, then:
 - (a) Resize F_i to $\ell(F_i) + 1$ rows.
 - (b) Decompose it as (base, grass) via the squash decomposition, where the Grassmannian part has a single descent at the new bottom row.
 - (c) Repeat the squash decomposition on the base as long as it is not yet full Grassmannian.
 - (d) Replace F_i with the resulting sequence of smaller full Grassmannian factors.
 - (e) Restart the loop from step 1.

The loop terminates when no further peeling is needed, producing a sorted tuple of full Grassmannian factors with non-decreasing row counts.

3.5 Multiplication of elements

Given elements $a = \sum_i \alpha_i K_i$ and $b = \sum_j \beta_j K_j$, the product is

$$a \cdot b = \sum_{i,j} \alpha_i \beta_j (K_i \cdot K_j),$$

where each $K_i \cdot K_j$ is computed via key concatenation and normalization as above.

4 Elementary Symmetric Elements

Definition 3. The elementary symmetric element $e_p^{(k)}$ (in size n) is defined as

$$e_p^{(k)} = \sum_G [G],$$

where the sum is over all k -row RC graphs G with $\pi(G) = s_{k-p+1}s_{k-p+2}\cdots s_k$ (i.e. $\text{code}(\pi(G)) = (\underbrace{0, \dots, 0}_{k-p}, \underbrace{1, \dots, 1}_p)$), and $[G]$ denotes the basis element for the key (G) of size n .

The SEM (standard elementary monomial) representation of any polynomial in the first n variables can be written as a linear combination of products $e_p^{(k)}$ for $1 \leq k \leq n$, $1 \leq p \leq k$ with distinct k .

5 Schubert Elements

The representation of a Schubert polynomial $\mathfrak{S}_w^{(n)}$ in the bounded RC factor algebra depends on whether $w \in S_n$ or w merely has Lehmer code of length $\leq n$.

5.1 Case 1: $w \in S_n$ (code length $\leq n - 1$)

When $w \in S_n$, the Schubert element has **no Grassmannian part** ($G_n = \text{id}$). It is defined purely via the SEM expansion:

$$\mathfrak{S}_w^{(n)} = \sum_{\mathbf{a}} c_{\mathbf{a}} e_{a_1}^{(1)} \cdot e_{a_2}^{(2)} \cdots e_{a_{n-1}}^{(n-1)},$$

where the sum is the SEM representation of $\mathfrak{S}_w$ and the $c_{\mathbf{a}}$ are integer coefficients.

5.2 Case 2: code length $= n$ (not in S_n , but code fits in n rows)

When $w \notin S_n$ but the Lehmer code of w has length $\leq n$, the Schubert element uses the Λ_n -module representation, computed as follows.

The n -variable polynomial ring $\mathbb{Z}[x_1, \dots, x_n]$ is a free module over Λ_n (the ring of symmetric polynomials in n variables), with basis $\{\mathfrak{S}_w : w \in S_n\}$. Concretely, for any w with code length $\leq n$:

$$\mathfrak{S}_w^{(n)} = \sum_{\substack{v \in S_n \\ G \text{ Grass.}}} c_{v,G} \mathfrak{S}_v \cdot [G],$$

where:

- v ranges over elements of S_n ,
- G ranges over n -row Grassmannian RC graphs,
- each $\mathfrak{S}_v$ is expanded in the S_n SEM basis (Case 1 above),
- $[G]$ is the basis element for the single-factor key (G) of size n ,
- the coefficients $c_{v,G}$ come from the full SEM decomposition.

This can be explicitly computed.

6 Key-to-RC-Graph Correspondence

Each basis key evaluates to a unique RC graph via iterated squash product (§3.2). Different keys may evaluate to the same RC graph; the normalization procedure ensures each key is canonical.

The map $K \mapsto \text{RC}(K)$ extends linearly:

$$\sum_i \alpha_i K_i \mapsto \sum_i \alpha_i \text{RC}(K_i).$$

Theorem 1. For each u, v with code length $\leq n$,

$$\text{RC}(\mathfrak{S}_u^{(n)} \cdot \mathfrak{S}_v^{(n)}) = \sum_w c_{u,v}^w \text{RC}(\mathfrak{S}_w^{(n)})$$

References

- [1] Matthew J. Samuel. A Molev-Sagan type formula for double Schubert polynomials. *J. Pure Appl. Algebra*, 228(7):107636, 2024.