Machine Checked Proofs and Programs in Algebraic Combinatorics

Florent Hivert

LISN / LMF / FRESCO project / Université Paris-saclay / CNRS / INRIA

CPP'25, Denver, January 2025

Algebraic Combinatorics

Going back and forth between

- algebraic identities
- algorithms and data-structure

Today:

- a Coq/Rocq+Mathematical Components based library about symmetric polynomials and characters formulas for the symmetric groups.
- flagship result: **Littlewood-Richardson rule**

Algebraic Combinatorics

Going back and forth between

- algebraic identities
- algorithms and data-structure

Today:

- a Coq/Rocq+Mathematical Components based library about symmetric polynomials and characters formulas for the symmetric groups.
- flagship result: **Littlewood-Richardson rule**

Algebraic Combinatorics

Going back and forth between

- algebraic identities
- algorithms and data-structure

The Littlewood-Richardson rule:

- proving a product rule of symmetric polynomials
- executing symbolically the Robinson-Schensted algorithm
- on the concatenation of two words

Algebraic Combinatorics

Going back and forth between

- algebraic identities
- algorithms and data-structure

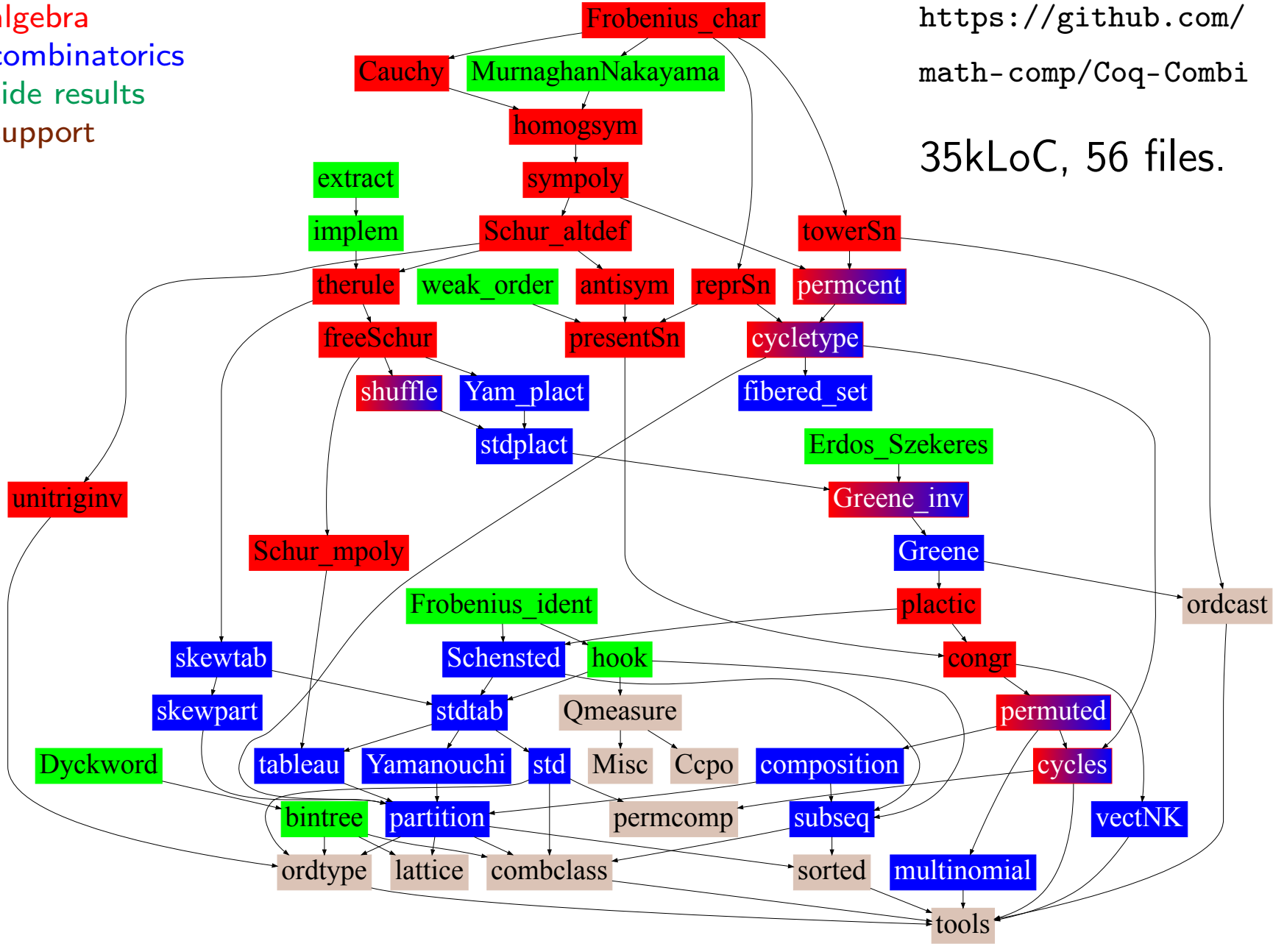
The Littlewood-Richardson rule:

- proving a product rule of symmetric polynomials
- executing symbolically the Robinson-Schensted algorithm
- on the concatenation of two words

algebra
combinatorics
side results
support

https://github.com/math-comp/Coq-Combi

35kLoC, 56 files.



Symmetric Polynomials

n-variables : $\mathbb{X} := \{x_0, x_1, \dots x_{n-1}\}$.
polynomials in $\mathbb{X}$: $\mathbb{C}[\mathbb{X}] = \mathbb{C}[x_0, x_1, \dots, x_{n-1}]$; ex: $3x_0^3x_2 + 5x_1x_2^4$.

Definition (Symmetric polynomial)

A polynomial is symmetric if it is invariant under any permutation of the variables: for all $\sigma \in \mathfrak{S}_n$,

$$P(x_0, x_1, \dots, x_{n-1}) = P(x_{\sigma(0)}, x_{\sigma(1)}, \dots, x_{\sigma(n-1)})$$

$$P(a, b, c) = a^2b + a^2c + b^2c + ab^2 + ac^2 + bc^2$$

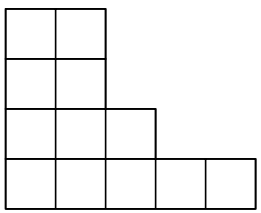
$$Q(a, b, c) = 5abc + 3a^2bc + 3ab^2c + 3abc^2$$

Integer Partitions

different ways of decomposing an integer $n \in \mathbb{N}$ as a sum:

$$12 = 12 = 11 + 1 = 10 + 2 = 10 + 1 + 1 + 1 = \cdots = 7 + 5$$
$$= 5 + 3 + 2 + 2 = 4 + 3 + 3 + 1 + 1 \text{ (77 partitions)}$$

Partition $\lambda := (\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_l > 0)$
$|\lambda| := \lambda_0 + \lambda_1 + \cdots + \lambda_l$ et $\ell(\lambda) := l$

Ferrer's diagram of a partitions : $(5, 3, 2, 2) \leftrightarrow$ 

```
Fixpoint is_part (sh : seq nat) := (* Boolean predicate *)
  if sh is sh0 :: sh'
  then (sh0 >= head 1 sh') && (is_part sh')
  else true.

Lemma is_partP sh : reflect  (* Boolean reflection lemma *)
  (last 1 sh != 0 /\ forall i, (nth 0 sh i) >= (nth 0 sh i.+1))
  (is_part sh).
```

Schur symmetric polynomials (Cauchy-Jacobi definition)

Definition (Schur symmetric polynomial)

Partition $\lambda := (\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{l-1})$ *with* $l \leq n$; *set* $\lambda_i := 0$ *for* $i \geq l$.

$$s_\lambda := \frac{\sum_{\sigma \in \mathfrak{S}_n} \operatorname{sign}(\sigma)\, \mathbb{X}_n^{\sigma(\lambda+\rho)}}{\prod_{0 \leq i < j < n} (x_j - x_i)} = \frac{\begin{vmatrix} x_1^{\lambda_{n-1}+0} & x_2^{\lambda_{n-1}+0} & \cdots & x_n^{\lambda_{n-1}+0} \\ x_1^{\lambda_{n-2}+1} & x_2^{\lambda_{n-2}+1} & \cdots & x_n^{\lambda_{n-2}+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\lambda_1+n-2} & x_2^{\lambda_1+n-2} & \cdots & x_n^{\lambda_1+n-2} \\ x_1^{\lambda_0+n-1} & x_2^{\lambda_0+n-1} & \cdots & x_n^{\lambda_0+n-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix}}$$

$$s_{(2,1)}(a,b,c) = a^2b + ab^2 + a^2c + 2abc + b^2c + ac^2 + bc^2$$

Littlewood-Richardson coefficients

Proposition

The family $(s_\lambda(\mathbb{X}_n))_{\ell(\lambda)\leq n}$ *is a (linear) basis of the ring of symmetric polynomials on* $\mathbb{X}_n$.

Definition (Littlewood-Richardson coefficients)

Coefficients $c^\nu_{\lambda,\mu}$ *of the expansion of the product:*

$$s_\lambda s_\mu = \sum_\nu c^\nu_{\lambda,\mu} s_\nu \,.$$

Fact: $s_\lambda(\mathbb{X}_{n-1}, x_n := 0) = s_\lambda(\mathbb{X}_{n-1})$ if $\ell(\lambda) < n$ else 0.
Consequence: $c^\nu_{\lambda,\mu}$ are independant of the number of variables.

The Littlewood-Richardson rule

Statement + 1st wrong proof: 1934, 1st correct proof: 1977

The Littlewood–Richardson rule is **notorious for the number of errors** that appeared prior to its complete, published proof. Several published attempts to prove it are incomplete, and it is particularly difficult to avoid errors when doing hand calculations with it: **even the original example** in D. E. Littlewood and A. R. Richardson (1934) **contains an error** – Wikipedia

Unfortunately the Littlewood-Richardson rule is **much harder to prove than was at first suspected**. I was once told that the Littlewood-Richardson rule helped to get men on the moon but was not proved until after they got there. The first part of this story might be an exaggeration. – Gordon James

Sample of applications

- computing a LR coeff is $\#P$-complete (counting version of NP)
- Mulmuley's geometric complexity theory: attack $P \neq NP$
- multiplicity of induction or restriction of irreducible representations of the symmetric groups
- multiplicity of the tensor product of the irreducible representations of linear groups
- geometry: intersection numbers of grassmanian varieties, cup product of the cohomology
- Horn problem: eigenvalues of the sum of two hermitian matrix
- extension of abelian groups (Hall algebra)
- application in quantum physics and chemistry (spectrum rays of the hydrogen atoms)

Combinatorial ingredients: Young Tableau

Definition

- *Filling of a partition shape*
- *non decreasing along the rows*
- *strictly increasing along the columns.*

- *row reading = natural reading*

d	*d*	*e*				
b	*c*	*c*	*c*	*d*		
a	*a*	*a*	*b*	*b*	*d*	*e*

$= ddebcccdaaabbde$

5				
2	6	9		
1	3	4	7	8

$= 526913478$

Young Tableau: formal definition

```
Variable (T : ordType) (Z : T). (* Type with a total order *)

Definition dominate (u v : list T) : bool :=
  (size u <= size v) &&
   (all (fun i => nth Z u i > nth Z v i) (iota 0 (size u))).

Lemma dominateP u v : (* Boolean reflexion lemma *)
  reflect (size u <= size v /\
           forall i, i < size u -> nth Z u i > nth Z v i)
          (dominate u v).

Fixpoint is_tableau (t : list (list T)) : bool :=
  if t is t0 :: t' then
    [&& (t0 != [::]), sorted t0,
      dominate (head [::] t') t0 & is_tableau t']
  else true.

Definition to_word t := flatten (rev t).  (* Row reading *)
```

Combinatorial definition of Schur functions

Definition

$$s_\lambda(\mathbb{X}) = \sum_{T \text{ tableaux of shape } \lambda} \mathbb{X}^T$$

where $\mathbb{X}^T$ *is the product of the elements of* T*.*

$$s_{(2,1)}(a,b,c) = a^2b + ab^2 + a^2c + 2abc + b^2c + ac^2 + bc^2$$

$$s_{(2,1)}(a,b,c) = \begin{array}{|c|c|}\hline b \\ \hline a & a \\ \hline \end{array} + \begin{array}{|c|c|}\hline b \\ \hline a & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & a \\ \hline \end{array} + \begin{array}{|c|c|}\hline b \\ \hline a & c \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline b & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & c \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline b & c \\ \hline \end{array}$$

Note: equivalence with Cauchy-Jacobi's definitions as a consequence of a particular case of the LR-rule (Piery rule), by recursively unfolding determinants.

Combinatorial definition of Schur functions

Definition

$$s_\lambda(\mathbb{X}) = \sum_{T \text{ tableaux of shape } \lambda} \mathbb{X}^T$$

where $\mathbb{X}^T$ *is the product of the elements of* T*.*

$$s_{(2,1)}(a,b,c) = a^2b + ab^2 + a^2c + 2abc + b^2c + ac^2 + bc^2$$

$$s_{(2,1)}(a,b,c) = \begin{array}{|c|c|}\hline b \\ \hline a & a \\ \hline \end{array} + \begin{array}{|c|c|}\hline b \\ \hline a & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & a \\ \hline \end{array} + \begin{array}{|c|c|}\hline b \\ \hline a & c \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline b & b \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline a & c \\ \hline \end{array} + \begin{array}{|c|c|}\hline c \\ \hline b & c \\ \hline \end{array}$$

Note: equivalence with Cauchy-Jacobi's definitions as a consequence of a particular case of the LR-rule (Piery rule), by recursively unfolding determinants.

Formal combinatorial definition of Schur functions

```
Variable n : nat.
Variable R : comRingType.

(* {mpoly R[n]} : the ring of polynomial over the commutative ring R *)
(*                in n variables ('X_0, 'X_1 ...)                    *)

Definition is_tab_of_shape (sh : list (list 'I_n)) :=
  [ pred t | (is_tableau t) && (shape t == sh) ].

(* Sigma type for tableaux of shape sh *)
Structure tabsh n (sh : 'P_d) :=
    TabSh { tabshval; _ : is_tab_of_shape sh tabshval }.
[...]
Canonical tabsh_finType n sh := [...] (* finite type = enumeration *)

Definition Schur d (sh : 'P_d) : {mpoly R[n]} :=
  \sum_(t : tabsh n sh) \prod_(i <- to_word t) 'X_i.
```

Yamanouchi Words

$|w|_x$: number of occurrence of x in w.

Definition

Sequence $w_0, \dots, w_{l-1}$ of integers such that for all k, i,

$$|w_i, \dots, w_{l-1}|_k \geq |w_i, \dots, w_{l-1}|_{k+1}$$

Equivalently $(|w|_i)_{i \leq \max(w)}$ is a partition and $w_1, \dots, w_{l-1}$ is also Yamanouchi.

$$(), 0, 00, 10, 000, 100, 010, 210,$$

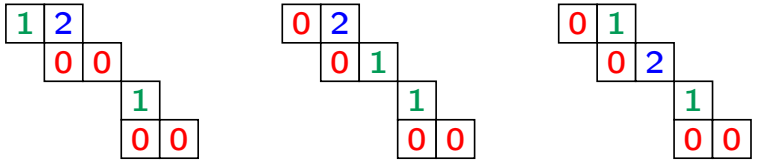
$$0000, 1010, 1100, 0010, 0100, 1000, 0210, 2010, 2100, 3210$$

The LR Rule at last !

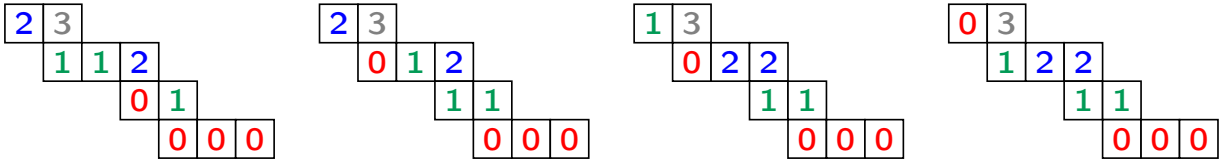
Theorem (Littlewood-Richardson rule)

$c^{\nu}_{\lambda,\mu}$ is the number of (skew) tableaux of shape the difference ν/λ, whose row reading is a Yamanouchi word of evaluation μ (ie. that is a permutations of $0^{\mu_0}1^{\mu_1}2^{\mu_2}\dots$).

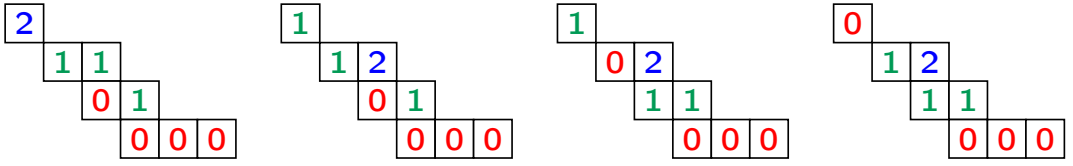
$C^{5432}_{331,421} = 3$



$C^{7542}_{431,4321} = 4$



$C^{7542}_{4321,431} = 4$



The formal statement

```
Definition is_skew_reshape_tableau (P P1 : list nat) (w : list T) : bool :=
  is_skew_tableau P1 (skew_reshape P1 P w).
Lemma is_skew_reshape_tableauP (P P1 : list nat) (w : list T) :
  size w = sumn (P / P1) ->
  reflect
    (exists tab, [/\ is_skew_tableau P1 tab,
                     shape tab = P / P1 & to_word tab = w])
    (is_skew_reshape_tableau P P1 w).

Definition LRyam_set P1 P2 P : {set : yameval P2} :=
  [set y : yameval P2 | is_skew_reshape_tableau P P1 y].
Definition LRyam_coeff P1 P2 P := #|LRyam_set P1 P2 P|.
```

Then

```
Theorem LRyam_coeffP :
  Schur P1 * Schur P2 =
  \sum_(P : 'P_(d1 + d2) | included P1 P) LRyam_coeff P1 P2 P * Schur P.
```

Idea of the proof

- the longest increasing subsequence problem
- Robinson-Schensted (RS) bijection:

$$abababcabbad \quad \longleftrightarrow \quad \begin{array}{|c|c|c|c|c|c|}\hline c & & & & & \\ \hline b & b & b & & & \\ \hline a & a & a & a & b & d \\ \hline \end{array} \;,\; \begin{array}{|c|c|c|c|c|c|}\hline 8 & & & & & \\ \hline 2 & 5 & 6 & & & \\ \hline 0 & 1 & 3 & 4 & 7 & 9 \\ \hline \end{array}$$

- reimplement RS using some local rewriting rules
- realize that RS is actually computing a normal form in a quotient of the free monoid
- lift the LR rule at a non-commutative level
- the proof is done by symbolically executing the RS algorithms on the concatenation of two words.

Outline of a proof

Lascoux, Leclerc and Thibon *The Plactic monoid*, *in* M. Lothaire, Algebraic combinatorics on words, Cambridge Univ. Press.

1. increasing subsequences and Schensted's algorithms;
2. the Robinson-Schensted bijection;
3. Green's invariants: computing the maximum sum of the length of k disjoint non-decreassing subsequences;
4. Knuth relations, the plactic monoïd;
5. Green's invariants are plactic invariants: Equivalence between Robinson-Schensted and plactic;
6. standardization; symmetry of RS;
7. lifting to non commutative polynomials : Free quasi-symmetric function and shuffle product;
8. non-commutative lifting the LR-rule : The free/tableau LR-rule;
9. back to Yamanouchi words: a final bijection.

The longest increasing subsequence problem

Some increasing subsequences:

$$ababcabbadbab$$
$$ababcabbadbab$$
$$ababcabbadbab$$

Problem (Schensted)

Given a finite sequence w, compute the ***maximum length of a increasing subsequence****.*

Schensted's algorithm

Algorithm

Start with an empty row r, insert the letters l of the word one by one from left to right by the following rule:

- *replace the first letter strictly larger that l by l;*
- *append l to r if there is no such letter.*

Insertion of $ababcabbadbab$

$$\emptyset \xrightarrow{a} \boxed{a} \xrightarrow{b} \boxed{a}\boxed{b} \xrightarrow{a} \boxed{a}\boxed{a} \xrightarrow{b} \boxed{a}\boxed{a}\boxed{b} \xrightarrow{c} \boxed{a}\boxed{a}\boxed{b}\boxed{c}$$

$$\xrightarrow{a} \boxed{a}\boxed{a}\boxed{a}\boxed{c} \xrightarrow{b} \boxed{a}\boxed{a}\boxed{a}\boxed{b} \xrightarrow{b} \boxed{a}\boxed{a}\boxed{a}\boxed{b}\boxed{b}$$

$$\xrightarrow{a} \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b} \xrightarrow{d} \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b}\boxed{d} \xrightarrow{b} \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b}\boxed{b}$$

$$\xrightarrow{a} \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b} \xrightarrow{b} \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b}\boxed{b}$$

Schensted's specification

Warning: list index start from 0.

Theorem (Schensted 1961)

The i-th entry $r[i+1]$ of the row r is the smallest letter which ends a increasing subsequence of length i.

$$\text{Schensted}(\ ababcabbadbab\ ) = \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{b}\boxed{b}$$

```
Theorem Sch_max_size w :
  size (Sch w) = \max_(s : subseqs w | is_sorted s) size s.
```

Robinson-Schensted's bijection

Bumped (replaced) letters are **insert it in a next row**.
Remembering which cell was added allows to inverse to process.

$$\emptyset,\emptyset \xrightarrow{a} \boxed{a},\boxed{0} \xrightarrow{b} \boxed{a}\boxed{b},\boxed{0}\boxed{1} \xrightarrow{a} \begin{array}{|c|c|}\hline b & \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|}\hline 2 & \\ \hline 0 & 1 \\ \hline \end{array} \xrightarrow{b}$$

$$\begin{array}{|c|c|c|}\hline b & & \\ \hline a & a & b \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline 2 & & \\ \hline 0 & 1 & 3 \\ \hline \end{array} \xrightarrow{c} \begin{array}{|c|c|c|c|}\hline b & & & \\ \hline a & a & b & c \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline 2 & & & \\ \hline 0 & 1 & 3 & 4 \\ \hline \end{array} \xrightarrow{a} \begin{array}{|c|c|c|c|}\hline b & b & & \\ \hline a & a & a & c \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline 2 & 5 & & \\ \hline 0 & 1 & 3 & 4 \\ \hline \end{array} \xrightarrow{b}$$

$$\begin{array}{|c|c|c|c|}\hline b & b & c & \\ \hline a & a & a & b \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline 2 & 5 & 6 & \\ \hline 0 & 1 & 3 & 4 \\ \hline \end{array} \xrightarrow{b} \begin{array}{|c|c|c|c|c|}\hline b & b & c & & \\ \hline a & a & a & b & b \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|}\hline 2 & 5 & 6 & & \\ \hline 0 & 1 & 3 & 4 & 7 \\ \hline \end{array} \xrightarrow{a}$$

$$\begin{array}{|c|c|c|c|c|}\hline c & & & & \\ \hline b & b & b & & \\ \hline a & a & a & a & b \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|}\hline 8 & & & & \\ \hline 2 & 5 & 6 & & \\ \hline 0 & 1 & 3 & 4 & 7 \\ \hline \end{array} \xrightarrow{d} \begin{array}{|c|c|c|c|c|c|}\hline c & & & & & \\ \hline b & b & b & & & \\ \hline a & a & a & a & b & d \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|c|}\hline 8 & & & & & \\ \hline 2 & 5 & 6 & & & \\ \hline 0 & 1 & 3 & 4 & 7 & 9 \\ \hline \end{array}$$

Idea of the proof: the non commutative lifting

Use RS to define a set of word

$$L_Q := \{w \mid RS(w)_2 = Q\}$$

whose commutative image is a Schur function:

$$S_{\text{shape}(Q)} = \sum_{w \in L_Q} \text{comm}(w)$$

Theorem (Noncommutative LR rule)

There exists an explicit set $\Omega(Q, R)$ such that

$$L_Q L_R = \bigcup_{T \in \Omega(Q,R)} L_T.$$

Character theory of the symmetric groups

Frobenius characteristic: **isometry** from symmetric function to the character ring of the symmetric groups.
We can translate the statements in a group theoretic language:

```
'SG_n       : the symmetric groups on the set [0, .., n-1]
permCT mu   : an element of the conjugacy class indexed by mu
'irrSG[la]  : the irreducible character for ['SG_n] associated to the
              partition [la] of n.

Theorem Fchar_isometry (f g : 'CF('SG_n)) : '[Fchar f | Fchar g] = '[f, g].

Theorem Murnaghan_NakayamaCT n (la mu : 'P_n) :
  'irrSG[la] (permCT mu) = MN_coeff la mu.

Theorem LR_rule_irrSG c d (la : 'P_c) (mu : 'P_d) :
  'Ind['SG_(c + d)] ('irrSG[la] \o^ 'irrSG[mu]) =
  \sum_(nu : 'P_(c + d) | included la nu) LRyam_coeff la mu nu * 'irrSG[nu].
```

Conclusion for combinatorialists

It's feasible !!!!

even by someone without any prior knowledge of type theory or lambda calculus; first version of the proof 14kLoC, 6 month.

Schützenberger's proof was correct !

A certified implementation ($\#P$-complete)

Conclusion for formal proof community

- **boolean reflexion** SSReflect/MathComp: very good at automatically **dealing with trivial cases**: extremely important for combinatorics
- extending the **algebraic hierarchy** was very hard, **hierarchy builder is a game changer**
- formalizing **algebra relatively easy** with MathComp, **combinatorics harder**, because very poorly reusable
- I feel that Mathcomp is too much **oriented toward finite**
- in many case, the **definition which is given is not the one which is used** in papers
- estimation: in 35kLoC of Coq/Rocq formalized 5% of what is in Sagemath about these topics (50kLoC python/C, 2300 functions) where combinat = 500kLoC, 18000 functions