Oscar: First Steps to Functions

September 18, 2025

You have seen

- How to do simple things in Oscar
- How to solve mathematical problems in Oscar
- How to operate Git

So its time for the next step:

Create mathematics in Oscar
Solve your own problems!

Example

Linear algebra: you need to intersect subspaces U_i, $i = 1, 2$ of $V = \mathbb{Q}^n$.

Easy: all linear algebra, a subspace U_i is given via some (echelonized) basis, so a matrix M_i in `rref`.

Wait: is it the columns or the rows? Short panic, experimentally `rref` does row operations. So the rows are the basis.

Intersection: Some thinking later... Consider:

$$X = \begin{bmatrix} M_1 & I_r \\ M_2 & 0 \end{bmatrix}$$

with `rref`

$$Y = \begin{bmatrix} A & B \\ 0 & T \end{bmatrix}$$

where A is in echelon form and has no zero row.

Sp TM_1 is a basis for the intersection!

1st attempt

```
julia> U_1 = matrix(QQ, 2, 4, [1 2 3 4; 0 1 2 3]);
julia> U_2 = matrix(QQ, 2, 4, [1 2 3 4; 0 1 3 2]);
julia> X = zero_matrix(QQ, 4, 6);
julia> for i=1:2
  for j=1:4
    X[i,j] = U_1[i,j]
  end
  X[i, i+4] = 1
end

julia> for i=1:2 for j=1:4 X[i+2, j] = U_2[i,j]; end; end
```

1st attempt

```
julia> X
[1   2   3   4   1   0]
[0   1   2   3   0   1]
[1   2   3   4   0   0]
[0   1   3   2   0   0]

julia> Y = rref(X)
(4, [1 0 0 -3 0 -3; 0 1 0 5 0 3; 0 0 1 -1 0 -1; 0 0 0 0 1 0]
```

so $T = \begin{bmatrix}1 & 0\end{bmatrix}$ the the intersection is

$$\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}$$

Phew! But is this what you want to do? And how you want to do it?

1st attempt - check

- It is error prone - I'll never get the indices right the 1st time
- It is hard to read: matrix magic is just that: magic

Slightly better, very slighty:

```
julia> X = [M_1 identity_matrix(QQ, nrows(M_1)) ;
            M_2 zero_matrix(QQ, nrows(M_2), nrows(M_1))];
```

Do Math!

Your problem was not to do magic with matrices, it was about intersection of subspaces.

So where are the (sub)spaces?

```
julia> V = free_module(QQ, 4);
julia> U_1, _ = sub(V, [V([1, 2, 3, 4]), V([0, 1, 2, 3])])
(Subspace over QQ with 2 generators and no relations,
  Hom: U_1 -> V)

julia> U_2, _ = sub(V, [V([1, 2, 3, 4]), V([0, 1, 3, 2])]);

julia> U_12, _ = U_1 \cap U_2
Subspace over QQ with 1 generator and no relations,
  Hom: subspace over QQ with 1 generator and no relations ->
```

Do Math!

I can't see anything, so lets relate this back

```
julia> U_12[1]     # 1 * 1st basis element
(1)

julia> V(U_12[1]) # as an element of V
(1, 2, 3, 4)
```

I'd argue this is much easier to read and maintain.

Similar examples can be done everywhere...

In Oscar: try to do Math!
not algorithms

2nd example

Lets multiply permutations. A permutation is just a table

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 2 & 1 & 4 \end{pmatrix}$$

Lets skip the 1st row: a permutation is a (dense) list of integers:

```
julia> p = [5, 3, 2, 1, 4];

julia> q = [2, 4, 3, 5, 1];
```

What about *pq*? As permutations? As dot-product (`Vector` product)?

```
julia> function m(p::Vector{Int}, q::Vector{Int})
  return [p[q[i]] for i=1:length(p)]
end
```

Or was it `[q[p[i]] for i=1:length(p)]`?

2nd example

By giving permutations a separate type I can write (overload) functions to apply only to permutations, so provide the natural context.

```
julia> G = symmetric_group(5);

julia> p = G([5, 3, 2, 1, 4]);

julia> q = G([2, 4, 3, 5, 1]);

julia> p*q
(2,3,4)
```

Matrices are not Maps

Let

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

This might "be" a map from $\mathbb{Q}^2 \to \mathbb{Q}^4$ or from $\mathbb{Q}^4 \to \mathbb{Q}^2$.

A might encode a subspace (row/ column image or kernel) or a quotient (row/column coimage or cokernel)

Or just a collection of numbers to be used elsewhere (points on a curve, indices of interesting groups, your birthday).

Matrices are useful and powerful.

They are not maps.

Matrices are not Maps

```
julia> V = free_module(QQ, 4);

julia> U = free_module(QQ, 2);

julia> phi = hom(U, V, matrix(QQ, 2, 4, [1 2 3 4; 5 6 7 8]))
Module homomorphism
  from vector space of dimension 2 over QQ
  to vector space of dimension 4 over QQ
```

Now the ambiguity is gone. The kernel will be a subspace of U, the image of V, ...

Matrices are not Maps

```
julia> H, mH = hom(U, V)
(Vector space of dimension 8 over QQ,
  Map: H -> set of all homomorphisms from U to V)

julia> mH(H[2])
Module homomorphism
  from vector space of dimension 2 over QQ
  to vector space of dimension 4 over QQ

julia> matrix(ans)
[0   1   0   0]
[0   0   0   0]
```

Now we can do two things: use $H = \hom(U, V)$ as a vector space, or use elements as maps. But semantically clearly separated

Summary

- Use mathematical (ly inspired) types.
- Make use of existing types - there are much more than you think
- Improve existing types - ask
- Follow our conventions in
  - Naming
  - Behaviour
  - Argument order
  - Formatting/ layout/ documentation
- Consider interactions
- Harness the power of maps!