ICERM Tutorial

This notebook was part of a tutorial on "Crystals" presented by Anne Schilling at ICERM February 4 and 6, 2013.

Building a crystal

This shows how to build a classical crystal, access its weight lattice, and the main operations on crystal elements.

```
B = CrystalOfTableaux(['C',2], shape=[1])
B

The crystal of tableaux of type ['C', 2] and shape(s) [[1]]

P = B.weight_lattice_realization()
P

Ambient space of the Root system of type ['C', 2]
P.basis()

Finite family {0: (1, 0), 1: (0, 1)}

B.list()

[[[1]], [[2]], [[-2]], [[-1]]]

b = B(rows=[])

[[2]]

b.weight()

(0, 1)

b.f(2)

[[-2]]
```
Tensor products of crystals

Now we learn how to take tensor products of crystals and decompose tensor products into irreducible components.

```python
B = CrystalOfTableaux(['A',2],shape=[1])
T = TensorProductOfCrystals(B,B,B)
for b in T:
    if b.is_highest_weight():
        print b
[[[1]], [[1]], [[1]]]
[[[1]], [[2]], [[1]]]
[[[2]], [[1]], [[1]]]
[[[3]], [[2]], [[1]]]
```
view(T)
Characters from crystals

We can compute the character by taking the sum over all crystal elements $b$ of $x^{\text{weight}(b)}$.

```python
B = CrystalOfTableaux(['A',2], shape=[2,1])
B
```

The crystal of tableaux of type ['A', 2] and shape(s) [[2, 1]]

```python
view(B)
```
[b.weight() for b in B]
R.<x1,x2,x3> = PolynomialRing(ZZ,3)
R
Multivariate Polynomial Ring in x1, x2, x3 over Integer Ring
x = R.gens()
x
(x1, x2, x3)

weight_sum = sum(prod(x[i]**b.weight()[i] for i in [0,1,2]) for b in B)
weight_sum
x1^2*x2 + x1*x2^2 + x1^2*x3 + 2*x1*x2*x3 + x2^2*x3 + x1*x3^2 + x2*x3^2

Symmetric functions

The character is nothing else, but the Schur function, which we can easily check on the computer!

Sym = SymmetricFunctions(ZZ)
s = Sym.schur()

Sym.from_polynomial(weight_sum)
2*m[1, 1, 1] + m[2, 1]
s(Sym.from_polynomial(weight_sum))
s[2, 1]

Demazure crystals

We can also investigate Demazure crystals on the computer. Note that for this functionality you will need the following patch applied http://trac.sagemath.org/sage_trac/ticket/14052.

B = CrystalOfTableaux(['A',2], shape=[2,1])
B
The crystal of tableaux of type ['A', 2] and shape(s) [[2, 1]]
C = CombinatorialFreeModule(QQ,B); C
Free module generated by The crystal of tableaux of type ['A', and shape(s) [[2, 1]] over Rational Field
t = B.highest_weight_vector()
t
b = C(t)

D = B.demazure_operator(b,[2,1]); D

\[ B[[[1, 1], [3]]] + B[[[1, 2], [2]]] + B[[[1, 3], [2]]] + B[[[1, 3], [3]]] + B[[[1, 1], [2]]] \]

D.support()

\[ [[[1, 1], [2]], [[[1, 2], [2]], [[[1, 3], [2]], [[[1, 1], [3]], [[[3], [3]]]]] \]

G = B.digraph(subset=D.support())
view(G)