Linear transformation to matrix

Given an ordered basis $\mathcal{B}$ for a vector space V of dimension n and

an ordered basis $\mathcal{C}$ for a vector space W of dimension m and

a linear transformation $T : V \to W$, then there is a correspondence

between T and a matrix ${}_{\mathcal{C}}[T]_{\mathcal{B}}$ which is $m \times n$.

We begin with a helper function which returns the coefficient of a basis element b in v

```
def coeff(v, b):
    """
    ask for the coefficient of the basis element `b` in the vector `v`

    If b in the keys of the dictionary of v then return v[b], otherwise 0

    INPUT:
        - v - a vector represented as a dictionary
        - b - an element of the basis

    OUTPUT:
        an element of the basis field or 0

    EXAMPLES::
        sage: coeff({1:3,3:-5},3)
        -5
        sage: coeff({1:3,3:-5},2)
        0
    """
    if b in v.keys():
        return v[b]
    else:
        return 0
```

evaluate

We begin by defining the map $L_{\mathcal{B}} : V \to \mathbb{C}^n$ and the inverse map $L_{\mathcal{B}}^{-1} : \mathbb{C}^n \to V$

An element of V will be represented by a Sage dictionary where the keys of the dictionary are elements of $\mathcal{B}$.

```
def L_basis(v, B):
    """
    create the column vector as an element of F^n from a vector `v`

    INPUT:
        - v - a vector represented as a dictionary
        - B - a basis

    OUTPUT:
        - a column vector

    EXAMPLES::
        sage: L_basis({1:4, 2:2, 3:-1},[1,2,3])
        [ 4]
        [ 2]
        [-1]
        sage: Bbasis = [var('v%d' % i) for i in range(1,5)]
        sage: L_basis({Bbasis[0]:1, Bbasis[2]:-2},Bbasis)
        [ 1]
        [ 0]
        [-2]
        [ 0]
    """
    return matrix([[coeff(v,b)] for b in B])

def L_basis_inv(v, B):
    """
    create a vector with basis `B` from the column vector

    INPUT:
        - v - a column vector as an nx1 matrix
        - B - an index set for a basis of V

    OUTPUT:
        - a vector represented as a dictionary

    EXAMPLES::
        sage: Bbasis = [var('v%d' % i) for i in range(1,5)]
        sage: L_basis_inv(L_basis({Bbasis[0]:1, Bbasis[2]:-2},Bbasis), Bbasis)
        {v4: 0, v3: -2, v2: 0, v1: 1}
        sage: L_basis_inv(matrix([[4],[2],[-1]]),[1,2,3])
        {1: 4, 2: 2, 3: -1}
    """
    return {B[i] : v[i][0] for i in range(len(B))}
```

The function below converts a linear transformation into a matrix and

the second function converts a matrix to a linear transformation

```
def lin_trans_to_matrix(T,B,C):
    """
    Given a linear transformation T from a vector
    space with basis B to a vector space with basis C
    return the size(C)xsize(B) matrix corresponding to T

    INPUT:
        - T - a linear transformation V to W
        - B - a basis for V
        - C - a basis for W

    OUTPUT:
        - a matrix of size len(C)xlen(B)

    EXAMPLES::
        sage: B = [var('v%d' % i) for i in range(1,5)]
        sage: T = lambda v: {1 : coeff(v,B[0])-coeff(v,B[1]), 2:
coeff(v,B[1])+coeff(v,B[1])+coeff(v,B[2])+coeff(v,B[3]), 3: coeff(v,B[1])-coeff(v,B[2])}
        sage: lin_trans_to_matrix(T, B, [1,2,3])
        [ 1 -1  0  0]
        [ 1  1  1  1]
        [ 0  1 -1  0]
        sage: T2 = lambda w: {B[0]: coeff(w,1)+coeff(w,3), B[1]: coeff(w,2)-3*coeff(w,3), B[3]:2*coeff(w,1)}
        sage: lin_trans_to_matrix(T2, [1,2,3], B)
        [ 1  0  1]
        [ 0  1 -3]
        [ 0  0  0]
        [ 2  0  0]
    """
    return matrix(sum((list(L_basis(T({b:1}),C).transpose()) for b in B),[])).transpose()

def matrix_to_lin_trans(M,B,C):
    """
    Given a matrix `M` and a basis `B` of V with # of columns(M) elements
    and a basis `C` of W with # of rows(M) elements, return a function
    which is a linear transformation from V to W.

    INPUT::
        - M - an nxm matrix
        - B - a basis with m elements
        - C - a basis with n elements

    OUTPUT:
        a function which accepts a vector with basis B and returns
        a vector with basis C

    EXAMPLES::
        sage: type(matrix_to_lin_trans(matrix([[1,2,3,4],[5,6,7,8]]),[1,2,3,4],[0,1]))
        <type 'function'>
        sage: matrix_to_lin_trans(matrix([[1,2,3,4],[5,6,7,8]]),[1,2,3,4],[0,1])({1:1,2:1,3:1,4:0})
        {0: 6, 1: 18}
    """
    return lambda v: L_basis_inv(M*L_basis(v,B),C)
```

BELOW THIS POINT ARE EXAMPLES::

```
# create 4 variables that will be our basis B
(v1,v2,v3,v4) = (var('v%d' % i) for i in range(1,5)) # this creates 4 basis elements
B = [v1,v2,v3,v4]
```

```
v = {v1:3, v2:-1, v3:0, v4:1} # a vector will be represented as a dictionary
```

```
L_basis(v, B) # our map L_B takes as input a vector and a basis and returns a column vector
```

```
[ 3]
[-1]
[ 0]
[ 1]
```

```
# and L_B^{-1} converts it back to a dictionary
L_basis_inv(matrix([[3],[-1],[0],[1]]), [v1,v2,v3,v4])
```

```
{v4: 1, v3: 0, v2: -1, v1: 3}
```

```
# let C be a basis for a vector space spanned by w1,w2,w3
C = [var('w%d' % i) for i in range(1,4)]
(w1,w2,w3) = tuple(C)
```

```
# T is an example linear transformation which converts a vector v to on in span(C)
T = lambda v: {w1: coeff(v, v1)-3*coeff(v,v2)-coeff(v,v4), w2: coeff(v,v1)+coeff(v,v2)+coeff(v,v3),
w3:7*coeff(v,v4)}
```

```
# my main function converts the linear transformation into a matrix
lin_trans_to_matrix(T,B,C)
```

```
[ 1 -3  0 -1]
[ 1  1  1  0]
[ 0  0  0  7]
```

```
# but since this is a correspondence and I have already written the functions,
# I can also convert a matrix to a linear transformation
# lets take a 4x3 matrix and use it to create a linear transformation from span(C) to span(B)
M = matrix([[1,0,1],[0,1,1],[-2,-2,1],[0,0,3]])
print M
T = matrix_to_lin_trans(matrix([[1,0,1],[0,1,1],[-2,-2,1],[0,0,3]]),C,B)
```

```
[ 1  0  1]
[ 0  1  1]
[-2 -2  1]
[ 0  0  3]
```

```
T
```

```
<function <lambda> at 0x1994057d0>
```

```
#lets see how T acts on each of the basis vectors
```

```
T({w1:1})
```

```
{v4: 0, v3: -2, v2: 0, v1: 1}
```

```
T({w2:1})
```

```
{v4: 0, v3: -2, v2: 1, v1: 0}
```

```
T({w3:1})
```

```
{v4: 3, v3: 1, v2: 1, v1: 1}
```

```
# and one example where T acts on a linear combination of basis vectors
```

```
T({w1:1,w2:1,w3:1})
```

```
{v4: 3, v3: -3, v2: 2, v1: 2}
```

```
```