Enumerate reaction networks composed with heteromultimer and single transformations
November 2014

This particular code is to enumerate all possible reaction networks (with certain dimensions, $n \times m$ where n is speciese number m is reaction number) composed of three types of interactions:

$$A + B \rightarrow C$$
$$\mathrm{D} \rightarrow E + F$$
$$G \rightarrow H$$

Those are heterodimerization, disassociation and single transformation.

With those three types of elementary reactions, we could construct a set of reaction networks, then we could use DSR graphs and bipartie to characteristic those networks if they are multistationary and has closed competition loop.

But before to go through such checking, we need to preclude situations that clearly not a complex balanced reaction network, by which mean it obeys the following three constraints:

0. Only allow elementary reactions described above, which is the starting point to construct the matrix;
   (more complex version should be including $I \rightarrow 2J$ and $2K \rightarrow L$ in future)
   list all possibly combinations and select m of those into matrix (sequence does not matter), then seperate into pos and neg matrices

1. Mass conservation;
   (it's very difficult to check, currently implemented without this checking but manual checking afterwards)

2. Complex banlanced: each species has at least one in flow and one out flow;
   (easy to check)
   Check pos matrix and neg matrix, (there is no zeros in columSum)

Then we need to:

4. check competition: there are at least one species has two -1 and there is another -1 in each of the according reactions;
   (in neg matrix, check if there are any two intersections of colSum and rowSum $\geqslant -2$ in a row are -1)
   find col indices, then get the other two species (competitors)

5. check loop: take indices of competitors, do the network searching, find the loop from one to another and then from the other to this one.

Cluster matrix into four categories: bistable with closed competition loop, bistable without closed competition loop, monostable with closed competition loop, monostable without closed competition loop.

Some functions might need:
sprintf(fmt, x1, ..., xn)
StringTools[Join](stringList, sep)
StringTools[CaseJoin](stringList)

mkdir(dirName)
FileTools[RemoveDirectory](dirName, options)
FileTools[Remove](file, file2, ...)
ListDirectory(dir, opt1, opt2, ...)
ArrayTools[AddAlongDimension](A,dim)

Initializations

```
> restart :
> interface(rtablesize = 40) :
> with(ListTools) :
> with(LinearAlgebra) :
> with(GraphTheory) :
> with(combinat) :
> with(ArrayTools) :
> _Envsignum0 := 0 :
>
```

Functions for multistationality checking (execute before proceeding)

Functions for constructing stoichiomatric matrix and examine the existence of competition and closed loop.

1. constructing stoichiomatric vectors based on the certain reaction patterns

List all reaction types based on the number of species. $R_n = \binom{n}{2} \cdot 2 + \binom{n}{3} \cdot 6$

```
> listRs := proc(n)
      local R, r, se, i, j, k, sign, l :
      r := binomial(n, 2) · 2 + binomial(n, 3) · 6 :
      R := Matrix(r, n) :
      # Now construct the reaction pattern matrix
      i := 1 :

      # here we first consider single transformation
      for se from -1 to 1 by 2 do
         for j from 1 to n by 1 do
            for k from j + 1 to n by 1 do
```

```
            R[i,j] := se :
            R[i,k] := -se :
            i := i + 1 :
          end do:
        end do:
      end do:

      # now consider heterodimerization and disassociation
      for sign from -1 to 1 by 2 do
        for j from 1 to n by 1 do
          for k from j + 1 to n by 1 do
            for l from k + 1 to n by 1 do
              R[i,j] := sign :
              R[i,k] := -sign :
              R[i,l] := -sign :
              i := i + 1 :
              R[i,j] := sign :
              R[i,k] := sign :
              R[i,l] := -sign :
              i := i + 1 :
              R[i,j] := sign :
              R[i,k] := -sign :
              R[i,l] := sign :
              i := i + 1 :
            end do:
          end do:
        end do:
      end do:

      # in this case we don't consider homodimerization and disassociation

      # Here transpose the R
      # R:=Transpose(R) :
       # no need to ranspose, need to assign rows to untransposed A's rows
      return(R) :
    end proc:
```

Here is a long function to enumerate first two reaction pattern (will reduce large mount of symmetric reactions).

Now we construct patterns of the first two reactions

```
> listR2 := proc(n)
      local R2, r2, R3, r3, R4, r4 :
      if n ≥ 4 then
        r2 := 29·2 :
        R2 := Matrix(r2, 4) :
```

 # in this case we don't consider homodimerization and disassociation

now we construct the reaction pattern with n = 4

$R2[1] := \langle 1,-1\rangle : R2[2] := \langle 0, 0, 1,-1\rangle :$
$R2[3] := \langle 1,-1\rangle : R2[4] := \langle 0, 1,-1\rangle :$
$R2[5] := \langle 1,-1\rangle : R2[6] := \langle 1, 0,-1\rangle :$
$R2[7] := \langle 1,-1\rangle : R2[8] := \langle 0,-1, 1\rangle :$
$R2[9] := \langle 1,-1\rangle : R2[10] := \langle -1, 0, 1\rangle :$
$R2[11] := \langle 1,-1\rangle : R2[12] := \langle -1, 1\rangle :$
$R2[13] := \langle 1,-1\rangle : R2[14] := \langle 0, 1,-1,-1\rangle :$
$R2[15] := \langle 1,-1\rangle : R2[16] := \langle 1, 0,-1,-1\rangle :$
$R2[17] := \langle 1,-1\rangle : R2[18] := \langle 0,-1, 1,-1\rangle :$
$R2[19] := \langle 1,-1\rangle : R2[20] := \langle -1, 0, 1,-1\rangle :$
$R2[21] := \langle 1,-1\rangle : R2[22] := \langle 0, 1, 1,-1\rangle :$
$R2[23] := \langle 1,-1\rangle : R2[24] := \langle 1, 0, 1,-1\rangle :$
$R2[25] := \langle 1,-1\rangle : R2[26] := \langle 0,-1, 1, 1\rangle :$
$R2[27] := \langle 1,-1\rangle : R2[28] := \langle -1, 0, 1, 1\rangle :$
$R2[29] := \langle 1,-1\rangle : R2[30] := \langle -1,-1, 1\rangle :$
$R2[31] := \langle 1,-1\rangle : R2[32] := \langle 1, 1,-1\rangle :$
$R2[33] := \langle 1,-1,-1\rangle : R2[34] := \langle 0, 1,-1,-1\rangle :$
$R2[35] := \langle 1,-1,-1\rangle : R2[36] := \langle 1, 0,-1,-1\rangle :$
$R2[37] := \langle 1,-1,-1\rangle : R2[38] := \langle 0,-1,-1, 1\rangle :$
$R2[39] := \langle 1,-1,-1\rangle : R2[40] := \langle -1,-1, 0, 1\rangle :$
$R2[41] := \langle 1,-1,-1\rangle : R2[42] := \langle 0, 1, 1,-1\rangle :$
$R2[43] := \langle 1,-1,-1\rangle : R2[44] := \langle 1, 1, 0,-1\rangle :$
$R2[45] := \langle 1,-1,-1\rangle : R2[46] := \langle 0, 1,-1, 1\rangle :$
$R2[47] := \langle 1,-1,-1\rangle : R2[48] := \langle -1, 1, 0, 1\rangle :$
$R2[49] := \langle 1,-1,-1\rangle : R2[50] := \langle -1, 1, 1\rangle ::$
$R2[51] := \langle 1, 1,-1\rangle : R2[52] := \langle 0, 1, 1,-1\rangle :$
$R2[53] := \langle 1, 1,-1\rangle : R2[54] := \langle 1, 1, 0,-1\rangle :$
$R2[55] := \langle 1, 1,-1\rangle : R2[56] := \langle 0, 1,-1, 1\rangle :$
$R2[57] := \langle 1, 1,-1\rangle : R2[58] := \langle 1,-1, 0, 1\rangle :$

if $n \geq 5$ **then**
now we add the reaction pattern with n = 5

$r3 := 43 \cdot 2 :$
$R3 := Matrix(r3, 5) :$
$R3[1..r2] := R2 :$

$R3[59] := \langle 1,-1\rangle : R3[60] := \langle 0, 0, 1,-1,-1\rangle :$
$R3[61] := \langle 1,-1\rangle : R3[62] := \langle 0, 0, 1, 1,-1\rangle :$
$R3[63] := \langle 1,-1,-1\rangle : R3[64] := \langle 0, 0, 1,-1,-1\rangle :$
$R3[65] := \langle 1,-1,-1\rangle : R3[66] := \langle 1, 0, 0,-1,-1\rangle :$
$R3[67] := \langle 1,-1,-1\rangle : R3[68] := \langle 0, 0,-1, 1,-1\rangle :$
$R3[69] := \langle 1,-1,-1\rangle : R3[70] := \langle -1, 0, 0, 1,-1\rangle :$
$R3[71] := \langle 1,-1,-1\rangle : R3[72] := \langle 0, 0, 1, 1,-1\rangle :$
$R3[73] := \langle 1,-1,-1\rangle : R3[74] := \langle 1, 0, 0, 1,-1\rangle :$
$R3[75] := \langle 1,-1,-1\rangle : R3[76] := \langle 0, 0,-1, 1, 1\rangle :$

```
R3[77] := ⟨1,-1,-1⟩ : R3[78] := ⟨-1, 0, 0, 1, 1⟩ :
R3[79] := ⟨1, 1,-1⟩ : R3[80] := ⟨0, 0, 1, 1,-1⟩ :
R3[81] := ⟨1, 1,-1⟩ : R3[82] := ⟨1, 0, 0, 1,-1⟩ :
R3[83] := ⟨1, 1,-1⟩ : R3[84] := ⟨0, 0,-1, 1, 1⟩ :
R3[85] := ⟨1, 1,-1⟩ : R3[86] := ⟨-1, 0, 0, 1, 1⟩ :


if n ≥ 6 then
  ## now we add the reaction pattern with n = 6

  r4 := 46·2 :
  R4 := Matrix(r4, n) :
  R4[1..r3] := R3 :

  R4[87] := ⟨1,-1,-1⟩ : R4[88] := ⟨0, 0, 0, 1,-1,-1⟩ :
  R4[89] := ⟨1,-1,-1⟩ : R4[90] := ⟨0, 0, 0, 1, 1,-1⟩ :
  R4[91] := ⟨1, 1,-1⟩ : R4[92] := ⟨0, 0, 0, 1, 1,-1⟩ :

  return(R4) :
else
  return(R3) :
end if:
else
  return(R2) :
end if:

end proc:
```

2. Now we can construct stoichiomatric matrix based on the reaction patterns.

Here, we construct the stoichiomatric matrices.

The total number of stoichiomatric matrices is $\binom{R_n}{m}$, which is still a huge number. But currently there seems no other better options.

We first consider when $m = 6$.

```
> constrM := proc(n)
    local R, A, r, g, h, i, j, k, l, m, total, right, x, y, z, absAdd, Add, V :
    r := binomial(n, 2)·2 + binomial(n, 3)·6 :

    R := listRs(n) :

    ## need to change here
    A := Matrix(3, n) :
```

```
    total := binomial(r, 3) :

    # here we use some variable to count how many reactions are correct.
    right := 0 :

    for g from 1 to r by 1 do
      for h from g + 1 to r by 1 do
        for i from h + 1 to r by 1 do
          #for j from i+1 to r by 1 do
            #for k from j+1 to r by 1 do
              #for l from k+1 to r by 1 do
                #for m from l+1 to r by 1 do
                  A[1] := R[g] :
                  A[2] := R[h] :
                  A[3] := R[i] :
                  #A[4] := R[j] :
                  #A[5] := R[k] :
                  #A[6] := R[l] :
                  # A[7] := R[m] :
                  # B:=Transpose(A) :

                  # now we have matrix A, we need to exam A with constraints.
                  absAdd := AddAlongDimension(|A|, 1) :
                  z := Search(0, absAdd) :
                  if z ≠ 0 then
                    Add := AddAlongDimension(A, 1) :
                    x := Search(0, VectorAdd(absAdd, Add, 1, -1)) :
                    if x ≠ 0 then
                      y := Search(0, VectorAdd(absAdd, Add, 1, 1)) :
                      if y ≠ 0 then
                        right := right + 1 :
                      end if:
                    end if:
                  end if:
                #end do:
              #end do:
            #end do:
          #end do:
        end do:
      end do:
    end do:

    V := [right, total, r] :
    return(V) :
  end proc:
>
```

Here we test all functions:

```
> V := constrM(5)    ## m=5, fix 2
```

$$V := [33500, 82160, 80] \quad \textbf{(1)}$$

```
> V := constrM(6)    ## m=5, fix 2
```

$$V := [390260, 551300, 150] \quad \textbf{(2)}$$

```
> V := constrM(5)    ## m=6, fix 2
```

$$V := [289575, 1581580, 80] \quad \textbf{(3)}$$

```
> V := constrM(6)    ## m=6, fix 2
```

$$V := [8615805, 20260275, 150] \quad \textbf{(4)}$$

```
>
```