Exclude isomorphic reaction networks from sets of interest
December 2014

Load from the folder, then compare with all networks from the other folder (unique networks).

Initializations

```
> restart :
> interface(rtablesize = 400) :
> with(ListTools) :
> with(LinearAlgebra) :
> with(VectorCalculus) :
> with(GraphTheory) :
> with(combinat) :
> with(ArrayTools) :
> with(FileTools) :
> _Envsignum0 := 0 :
```

Functions for constructing graph from stoichiometric matrix

```
> findZ :=proc(A)
     local Z, n, m, i, j :
      n := Dimension(A)[1] :
      m := Dimension(A)[2] :
      Z := Matrix(n, m) :
    for i from 1 to n by 1 do
      for j from 1 to m by 1 do
         if  A[i, j] < 0 then Z[i, j] := z[i, j]; end if;        ### what is the z?
      end do;
   end do:
    return(Z) :
    end proc:
```

Find the DSR graph from labels, A and Z

```
> ##Create signed DSR graph: entries are two matrices and the labels of the nodes
   createDSRgraphsigned :=proc(mynodes, A, Z)
      local G, n, m, Adj, varsZ, Zsign, varsA, Asign, X :
```

```
    n := Dimension(A)[1] : m := Dimension(A)[2] :
    X := Transpose(Z) :
    varsZ := indets(X) :
    Zsign := subs(seq(varsZ[i] = 1, i = 1 ..numelems(varsZ)), X) :

   Adj := Matrix(n + m, n + m) :
   Adj[[n + 1 ..n + m], [1 ..n]] := Transpose(map(signum, A)) :
   Adj[[1 ..n], [n + 1 ..n + m]] := Transpose(Zsign) :

   G := GraphTheory[Graph](mynodes, Adj, weighted = true) :
   return(G) :
  end proc:

> ##Create DSR graph: entries are two matrices and the labels of the nodes
  createDSRgraph := proc(mynodes, A, Z)
    local G, n, m, Adj, varsZ, Zsign, varsA, Asign, X :
    n := Dimension(A)[1] : m := Dimension(A)[2] :
    X := Transpose(Z) :
    varsZ := indets(X) :
    Zsign := subs(seq(varsZ[i] = 1, i = 1 ..numelems(varsZ)), X) :
    varsA := indets(A) :
    Asign := subs(seq(varsA[i] = 1, i = 1 ..numelems(varsA)), A) :

   Adj := Matrix(n + m, n + m) :
   Adj[[n + 1 ..n + m], [1 ..n]] := Transpose(map(signum, Asign)) :
   Adj[[1 ..n], [n + 1 ..n + m]] := Transpose(Zsign) :

   G := GraphTheory[Graph](mynodes, Adj, weighted = true) :
   return(G) :
  end proc:
>
```

Functions for checking isomorphic reaction networks

```
> finduniquematrices := proc(originfolder, uniquefolder)
     local nodes, originfiles, uniquefiles, n, m, i, j, ori, uni, A1, A2, Z1, Z2, G1, G2, count :
     nodes := ["A", "B", "C", "D", "E", seq(R_i, i = 1 ..5)] :
     originfiles := ListDirectory(originfolder) :
     n := nops(originfiles) :
     uniquefiles := ListDirectory(uniquefolder) :

     for i from 1 to n by 1 do
        ori := op(i, originfiles) :
```

```
        A1 := ImportMatrix(cat(originfolder, "/", ori)) :
        Z1 := findZ(A1) :
        G1 := createDSRgraph(nodes, A1, Z1) :
        m := nops(uniquefiles) :
        if m > 0 then
          count := 0 :
          for j from 1 to m by 1 do

            uni := op(j, uniquefiles) :
            A2 := ImportMatrix(cat(uniquefolder, "/", uni)) :
            Z2 := findZ(A2) :
            G2 := createDSRgraph(nodes, A2, Z2) :
            if IsIsomorphic(G1, G2) then
              count := count + 1 :
              break:
            end if:
          end do:
          if count = 0 then
            ExportMatrix(cat(uniquefolder, "/", ori), A1, target = csv, format = rectangular,
       mode = ascii) :
            uniquefiles := ListDirectory(uniquefolder) :
          end if:
        else
          ExportMatrix(cat(uniquefolder, "/", ori), A1, target = csv, format = rectangular, mode
        = ascii) :
          uniquefiles := ListDirectory(uniquefolder) :
        end if:
      end do:
    end proc:
```

```
> originfolder := "5species/multistationary/competitionloop_intersectingloops" :
> uniquefolder := "5species/multistationary/unique_competitionloop_intersectingloops" :
>
> originfolder := "5species/multistationary/nocompetitionloop_intersectingloops" :
> uniquefolder := "5species/multistationary/unique_nocompetitionloop_intersectingloops" :
>
> originfolder := "5species/nonmultistationary/competitionloop_intersectingloops" :
> uniquefolder := "5species/nonmultistationary/unique_competitionloop_intersectingloops" :
>
> originfolder := "5species/nonmultistationary/nocompetitionloop_intersectingloops" :
> uniquefolder := "5species/nonmultistationary/unique_nocompetitionloop_intersectingloops" :

> finduniquematrices(originfolder, uniquefolder)
```

Testing

```
> originfiles := ListDirectory(originfolder)
originfiles := ["injectiveEx0_1517.csv", "injectiveEx0_1742.csv", "injectiveEx0_1775.csv",       (4.1)
    "injectiveEx0_2081.csv", "injectiveEx0_2084.csv", "injectiveEx0_2268.csv",
    "injectiveEx0_2931.csv", "injectiveEx0_3129.csv", "injectiveEx0_3132.csv",
    "injectiveEx0_3141.csv", "injectiveEx0_3169.csv", "injectiveEx0_3217.csv",
    "injectiveEx0_3223.csv", "injectiveEx0_3233.csv", "injectiveEx0_3235.csv",
    "injectiveEx0_3288.csv", "injectiveEx0_3356.csv", "injectiveEx0_3566.csv",
    "injectiveEx0_3604.csv", "injectiveEx0_3845.csv", "injectiveEx0_3957.csv",
    "injectiveEx0_4115.csv", "injectiveEx0_4135.csv", "injectiveEx0_4302.csv",
    "injectiveEx0_5463.csv", "injectiveEx0_5809.csv", "injectiveEx0_7185.csv",
    "injectiveEx0_7214.csv", "injectiveEx0_7218.csv", "injectiveEx0_7639.csv",
    "injectiveEx0_7758.csv", "injectiveEx0_7770.csv", "injectiveEx0_7856.csv",
    "injectiveEx0_7870.csv", "injectiveEx2_1741.csv", "injectiveEx2_1773.csv",
    "injectiveEx2_2083.csv", "injectiveEx2_2245.csv", "injectiveEx2_2926.csv",
    "injectiveEx2_3286.csv", "injectiveEx2_3659.csv", "injectiveEx2_4114.csv",
    "injectiveEx2_4741.csv", "injectiveEx2_4767.csv", "injectiveEx2_4800.csv",
    "injectiveEx2_4825.csv", "injectiveEx2_7852.csv", "injectiveEx2_7863.csv",
    "injectiveEx2_8375.csv", "injectiveEx2_8384.csv", "injectiveEx2_9147.csv",
    "injectiveEx2_9154.csv"]
> ori := op(1, originfiles)
                          ori := "injectiveEx0_1517.csv"                               (4.2)
> A1 := ImportMatrix(cat(originfolder, "/", ori))
```

$$A1 := \begin{bmatrix} 1 & 0 & -1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} \qquad \textbf{(4.3)}$$

```
>
>
>
>
> A1 := ImportMatrix("right_1478_injective1.csv")
> A2 := ImportMatrix("right_1479_injective1.csv")
                                                                                       (4.4)
```

$$A2 := \begin{bmatrix} 1 & 0 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} \tag{4.4}$$

> $Z1 := findZ(A1) :$

> $Z2 := findZ(A2) :$

> $species := [\text{"A"}, \text{"B"}, \text{"C"}, \text{"D"}, \text{"E"}, seq(R_i, i = 1 ..5)]$

$$species := [\text{"A"}, \text{"B"}, \text{"C"}, \text{"D"}, \text{"E"}, R_1, R_2, R_3, R_4, R_5] \tag{4.5}$$

> $A3 := A1$

$$A3 := \begin{bmatrix} 1 & 0 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{4.6}$$

> $a34 := A3[4]$

$$a34 := \begin{bmatrix} 0 & -1 & 1 & 0 & 0 \end{bmatrix} \tag{4.7}$$

> $A3[4] := A3[3]$

$$A3_4 := \begin{bmatrix} 0 & 1 & 0 & 0 & -1 \end{bmatrix} \tag{4.8}$$

> $A3[3] := a34$

$$A3_3 := \begin{bmatrix} 0 & -1 & 1 & 0 & 0 \end{bmatrix} \tag{4.9}$$

> $A3$

$$\begin{bmatrix} 1 & 0 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 & 1 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{4.10}$$

> $Z3 := findZ(A3) :$

> $G1 := createDSRgraph(species, A1, Z1) :$

> $G2 := createDSRgraph(species, A2, Z2) :$

> $G3 := createDSRgraph(species, A3, Z3) :$

> $Gs1 := createDSRgraph(species, A1, Z1) :$

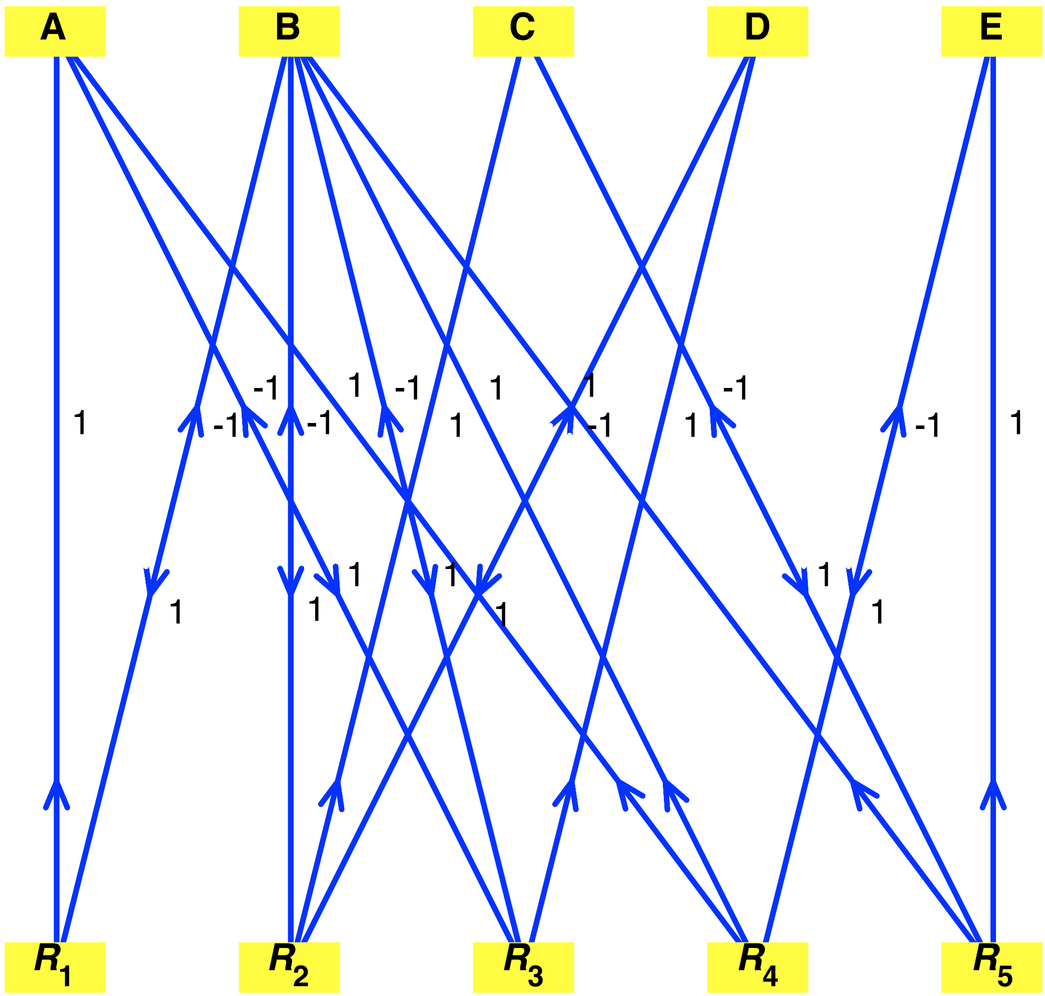
> $Gs2 := createDSRgraph(species, A2, Z2) :$

> $Gs3 := createDSRgraph(species, A3, Z3) :$

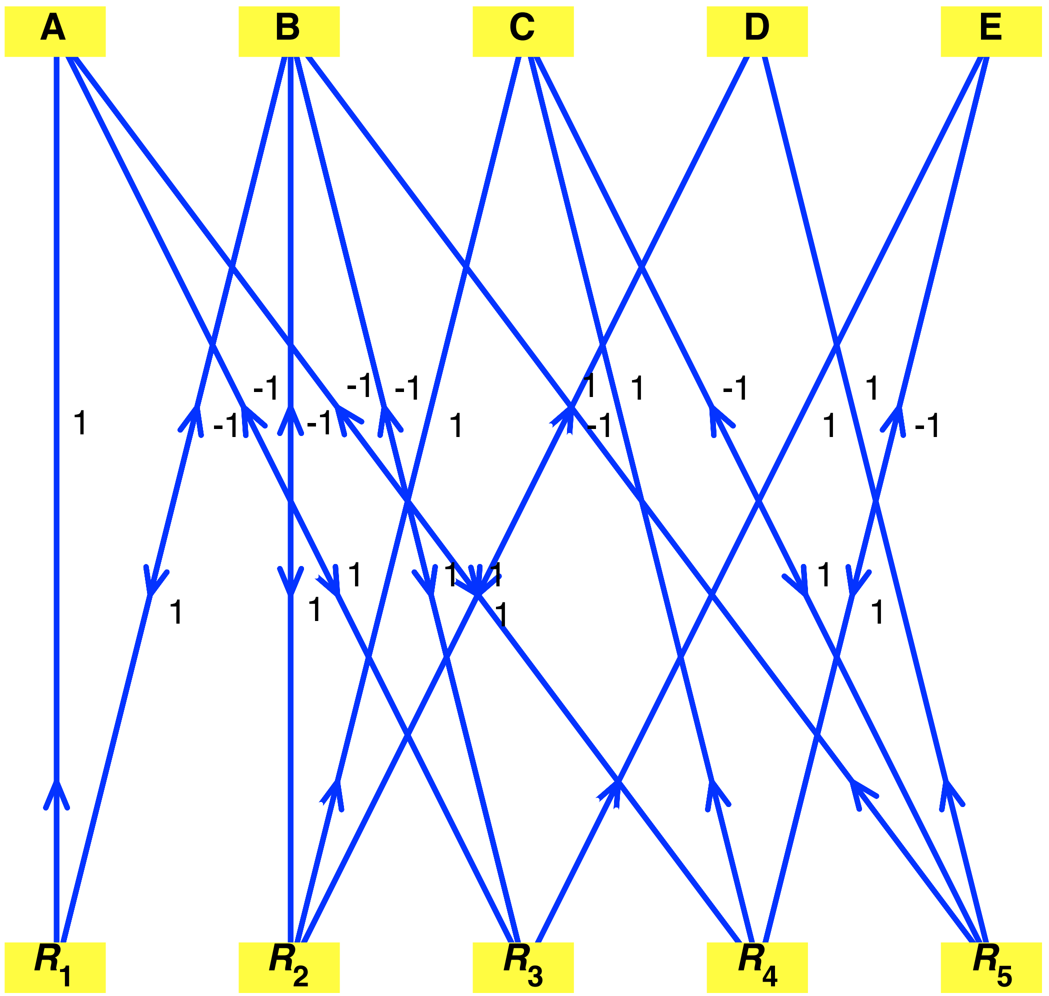
> $IsIsomorphic(G1, G2)$

$$false \tag{4.11}$$

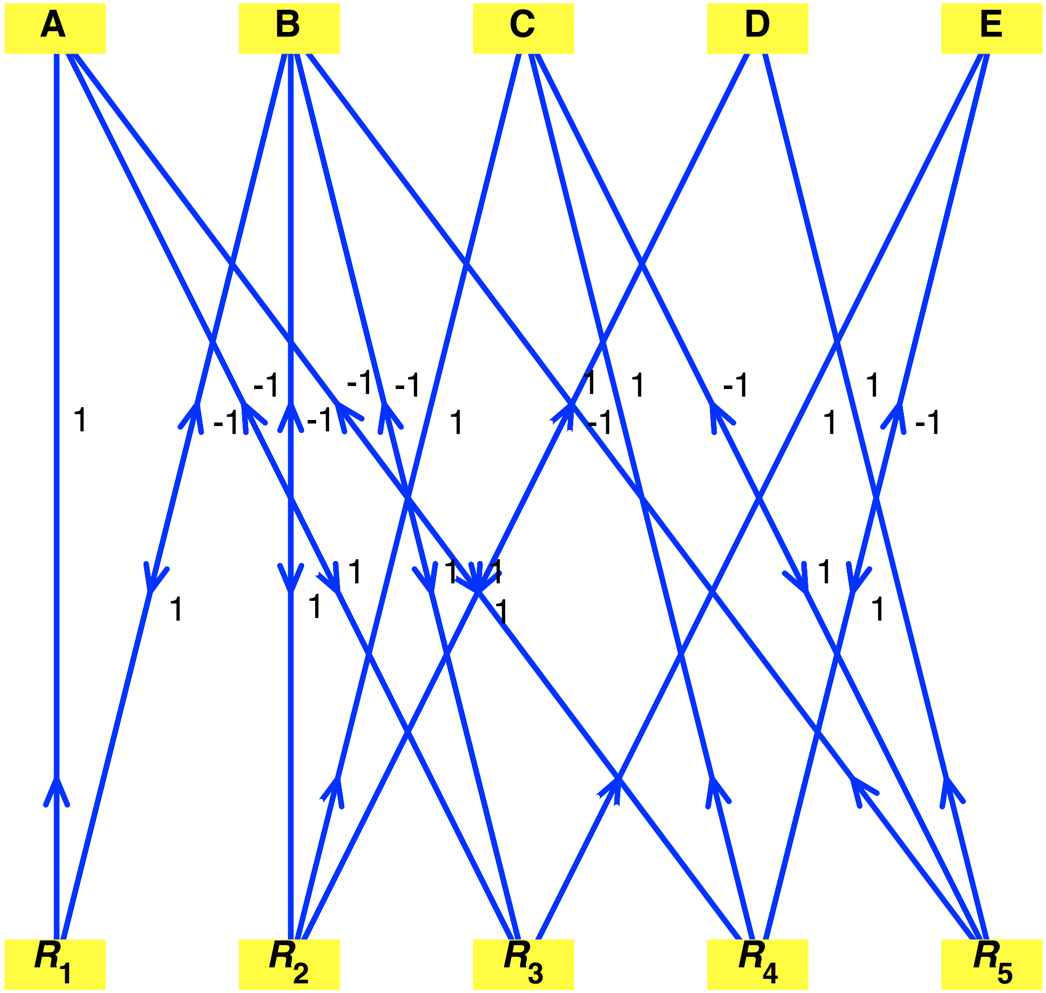
```
> IsIsomorphic(G1, G3)
                                  true                                  (4.12)
> IsIsomorphic(Gs1, Gs2)
                                  false                                 (4.13)
> IsIsomorphic(Gs1, Gs3)
                                  true                                  (4.14)
> DrawGraph(G1)
```



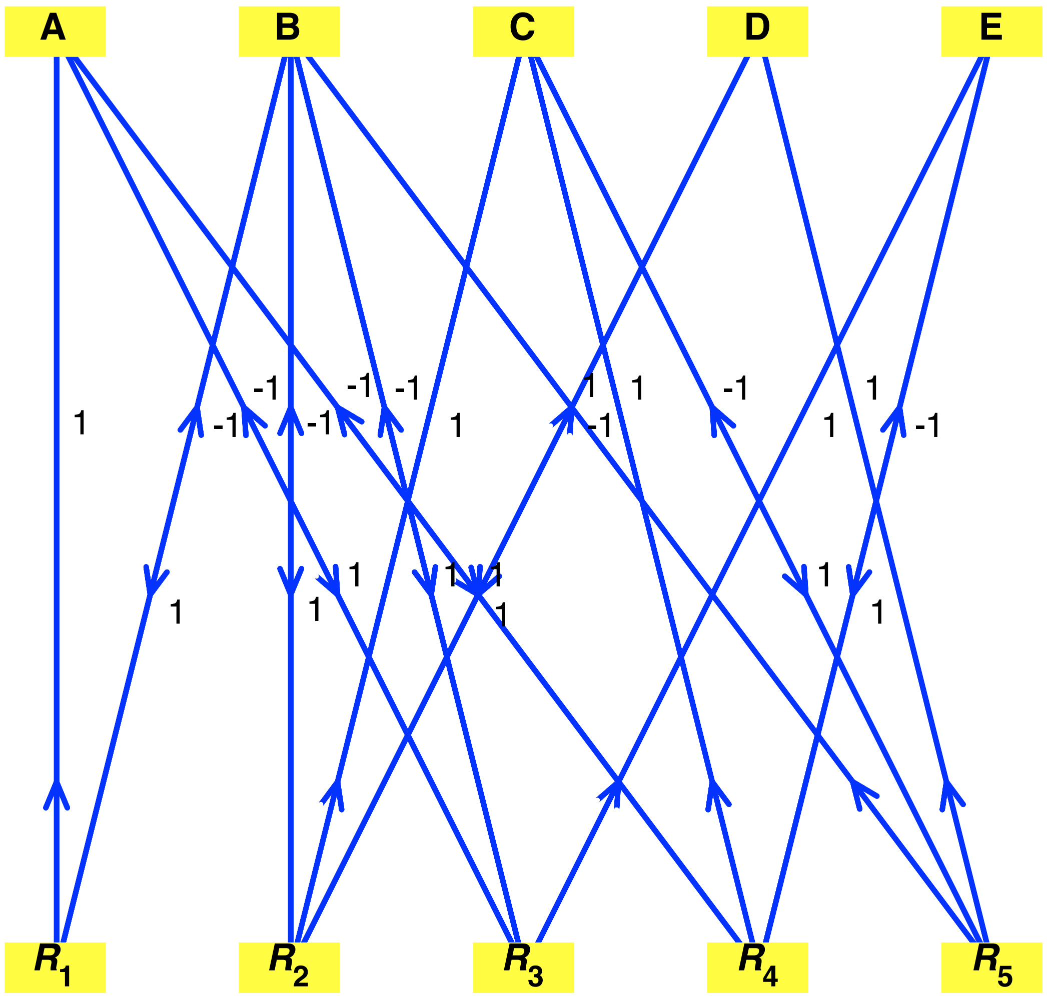
```
> DrawGraph(G2)
```



```
> DrawGraph(G2)
```



```
> DrawGraph(Gs2)
```


A
B
C
D
E
-1
-1
-1
1
1
-1
1
1
-1
-1
1
-1
1
-1
1
1
1
1
1
1
1
1
R_1
R_2
R_3
R_4
R_5