Algebraic analysis of multiplex, signed, and two-mode networks

Antonio Rivero Ostoic

University of San Simón

Sunbelt XXXVIII ☆ Utrecht, Netherlands ☆ 26 June 2018

Agenda

visualization and algebraic analysis . . .

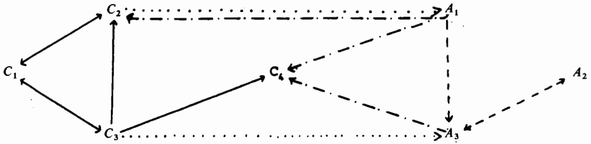
1. Introduction
   - (multilevel networks)
2. Algebraic structures
3. Multiplex networks
4. Signed networks
5. Affiliation networks

1. Introduction

(multilevel networks)

Multilevel Networks

- A **multilevel** network combines one-mode and two-mode structures
- Example: "Clients and Attorneys" (from Wasserman & Iacobucci, 1991)



$i \longrightarrow j$ Client i makes a recommendation to client j.
i ——> j Attorney i indicates respect for attorney j.
i ······> j Client i hires attorney j.
i —·—> j Attorney i sends a letter to client j.

Makes a recommendation

	C_1	C_2	C_3	C_4
C_1	–	1	1	0
C_2	1	–	0	0
C_3	1	1	–	1
C_4	0	0	0	–

Hires

	A_1	A_2	A_3
C_1	0	0	0
C_2	1	0	0
C_3	0	0	1
C_4	0	0	0

Mails letter

	C_1	C_2	C_3	C_4
A_1	0	1	0	1
A_2	0	0	0	0
A_3	0	0	0	1

Respects

	A_1	A_2	A_3
A_1	–	0	1
A_2	0	–	1
A_3	0	1	–

Figure 1. Social network interactions among four clients and three attorneys.

Package installation & Working directory

```r
# INSTALL 'multiplex' & 'multigraph' FROM CRAN
install.packages("multiplex")
install.packages("multigraph")

# OR INSTALL THE BETA VERSIONS FROM GITHUB
library("devtools")
devtools::install_github("mplex/multigraph", ref = "beta")
devtools::install_github("mplex/multiplex", ref = "beta")
```

```r
# SET WORKING DIRECTORY PATH (e.g.)
setwd("C:/sunbelt/")
```

One-mode network data creation

Define network data as matrices with the `transf` function:

```
library("multiplex")
# MAKES A RECOMENDATION
transf(c("C1, C2","C1, C3","C2, C1","C3, C1","C3, C2","C3, C4"))
```

```
   C1 C2 C3 C4
C1  0  1  1  0
C2  1  0  0  0
C3  1  1  0  1
C4  0  0  0  0
```

```
# RESPECTS
transf(c("A1, A3","A2, A3","A3, A2"))
```

```
   A1 A3 A2
A1  0  1  0
A3  0  0  1
A2  0  1  0
```

Multiplex network data creation

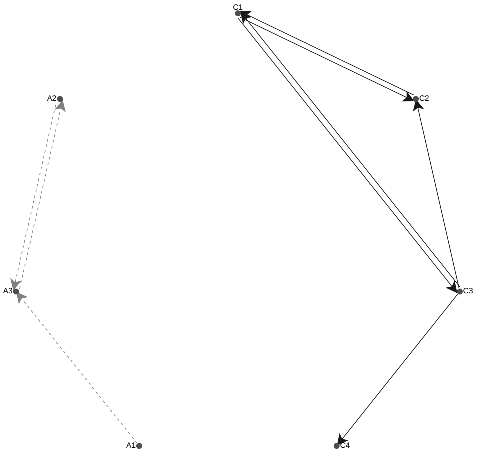
Multiplex networks via `transf` use a list of pairwise relations

```
# MULTIPLEX NETWORK WITH TWO TYPES OF RELATIONS 'A' AND 'B'
nt1 <- transf(list(A = c("C1, C2","C1, C3","C2, C1","C3, C1","C3, C2","C3, C4")
+    , B = c("A1, A3","A2, A3","A3, A2")))
```

```
, , A

   C1 C2 C3 C4 A1 A3 A2
C1  0  1  1  0  0  0  0
C2  1  0  0  0  0  0  0
C3  1  1  0  1  0  0  0
C4  0  0  0  0  0  0  0
A1  0  0  0  0  0  0  0
A3  0  0  0  0  0  0  0
A2  0  0  0  0  0  0  0

, , B

   C1 C2 C3 C4 A1 A3 A2
C1  0  0  0  0  0  0  0
C2  0  0  0  0  0  0  0
C3  0  0  0  0  0  0  0
C4  0  0  0  0  0  0  0
A1  0  0  0  0  0  1  0
A3  0  0  0  0  0  0  1
A2  0  0  0  0  0  1  0
```

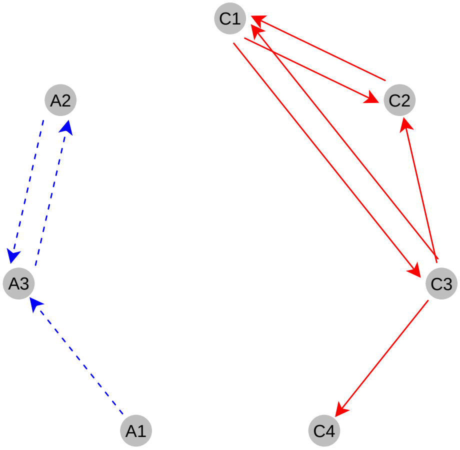
Network visualization

```
# LOOK AT THE MULTIGRAPH OF NETWORK 'nt1'
library("multigraph")
multigraph(nt1)
```



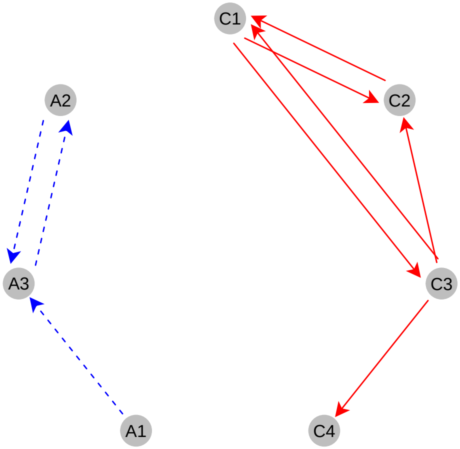
Network visualization

```
# ADD VERTEX / EDGE / GRAPH CHARACTERISTICS
multigraph(nt1, cex = 6, vcol =  8, ecol = c("red","blue"), lwd = 2, pos = 0)
```



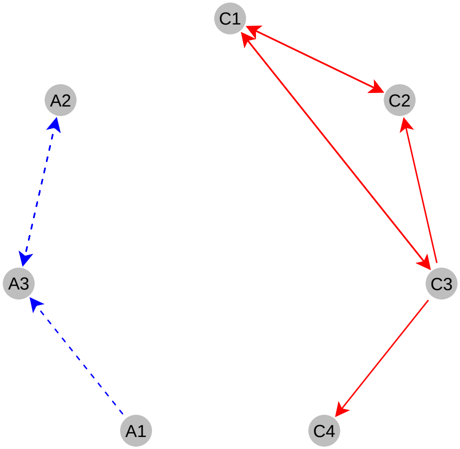
Network visualization

```
# DEFINE A 'list' OF VERTEX / EDGE / GRAPH CHARACTERISTICS
scp <- list(cex = 6, vcol =  8, ecol = c("red","blue"), lwd = 2, pos = 0)
multigraph(nt1, scope = scp)
```



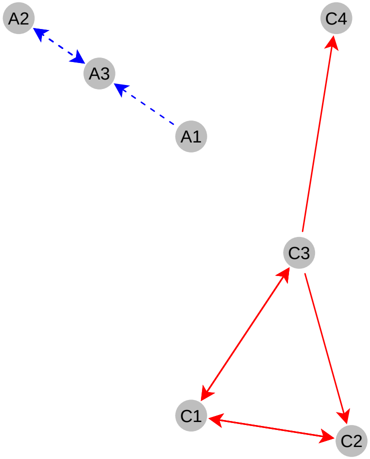
Network visualization

```
# COLLAPSE RECIPROCATED TIES
multigraph(nt1, scope = scp, collRecip = TRUE)
```



Network visualization

```
# APPLY A FORCE DIRECTED LAYOUT TO THE DIGRAPH
multigraph(nt1, scope = scp, collRecip = TRUE, layout = "force", seed = 123)
```



Two-mode network data creation

For two-mode data, use argument `type` `toarray2` in function `transf`

➟ arguments `add` (domain) and `adc` (codomain) are for adding isolates

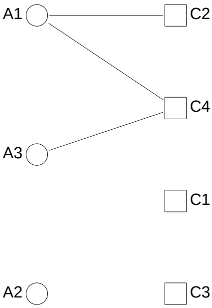
```
nt2 <- transf(list(C = c("A1, C2","A1, C4","A3, C4"), D = c("C2, A1","C3, A3")),
+    type = "toarray2", add = list("A2", "C1"), adc = list(c("C1","C3"), "A2"))
```

```
$C
   C2 C4 C1 C3
A1  1  1  0  0
A3  0  1  0  0
A2  0  0  0  0

$D
   A1 A3 A2
C2  1  0  0
C3  0  1  0
C1  0  0  0
```

Visualization two-mode networks

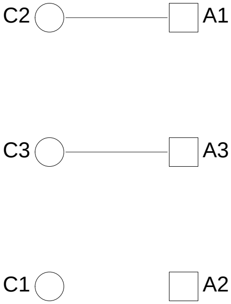
Function `bmgraph` depicts bipartite graphs

```
bmgraph(nt2[["C"]])
```



Mails letter, `nt2$C`

```
bmgraph(nt2[["D"]])
```



Hires, `nt2$D`

Multilevel network data creation

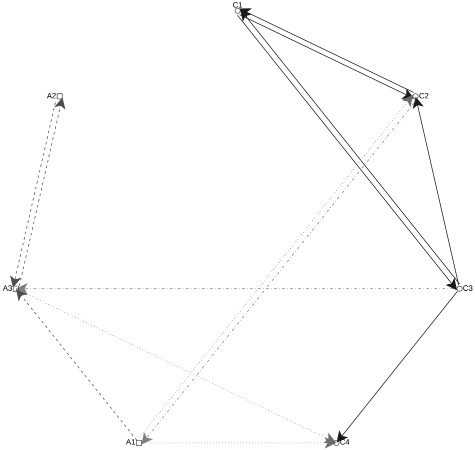
Function `mlvl` constructs multilevel structures

```
nt12 <- mlvl(nt1, nt2)
```

```
, , A

   C1 C2 C3 C4 A1 A3 A2
C1  0  1  1  0  0  0  0
C2  1  0  0  0  0  0  0
C3  1  1  0  1  0  0  0
C4  0  0  0  0  0  0  0
A1  0  0  0  0  0  0  0
A3  0  0  0  0  0  0  0
A2  0  0  0  0  0  0  0

...


, , D

   C1 C2 C3 C4 A1 A3 A2
C1  0  0  0  0  0  0  0
C2  0  0  0  0  1  0  0
C3  0  0  0  0  0  1  0
C4  0  0  0  0  0  0  0
A1  0  0  0  0  0  0  0
A3  0  0  0  0  0  0  0
A2  0  0  0  0  0  0  0
```

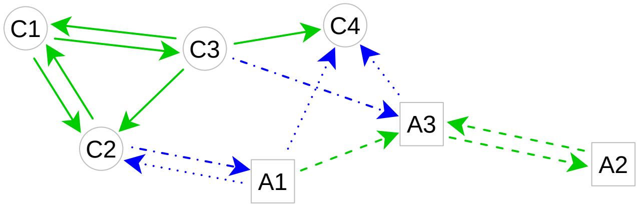
Visualization multilevel networks

```r
# DEFAULT LAYOUT OF multigraph()
multigraph(nt12)
```



Visualization multilevel networks

```
# COSTUMIZED LAYOUT ALTERS scp WITH VECTORS AS LISTS
scp2 <- c(scp, ecol = list(c(3,3,4,4)), bwd = .5, rot = 90)
multigraph(nt12, scope = scp2, layout = "force", seed = 1)
```



2. Algebraic structures

Representations for multiplex networks

Simple networks:

- *(Simple) graphs*, *matrices*

  ➟ for relations between actors

Multiplex networks:

- *Multigraphs*, *arrays*

  ➟ for (types of) relations between actors

- *Cayley graphs*, *tables*

  ➟ for relationships between relations

☛ *Different types of algebraic structures are represented by tables*

Multiplex networks

Algebraic systems representing multiplex networks:

Type of structure	Algebraic object
Elementary	*Group*
Complex	*Semigroup*, *Semiring*, *Lattice*, etc.

☛ Computations with `multiplex` and visualization with `multigraph`

Group

Algebraic elementary structure

A **group** is an algebraic structure with an *element set* and an endowed *operation*:

$$\langle\, G,\ \cdot \,\rangle$$

That for all $a, b, c, e \in G$ satisfies axioms:

$$\begin{aligned}
\text{Identity:} &\quad a \cdot e = e \cdot a = a \\
\textit{Inversion}\text{:} &\quad a \cdot a^{-1} = a^{-1} \cdot a = e \\
\text{Associativity:} &\quad (a \cdot b) \cdot c = a \cdot (b \cdot c) \\
\text{Closure:} &\quad a \cdot b \in G \qquad \text{(for all} \quad a, b\text{)}
\end{aligned}$$

➟ where e is the identity element in G.

Group structure by permutations

Theorem (Cayley).

All of group theory can be found in permutations.

➟ we focus on permutation symmetry

A permutation operator is represented by a *permutation matrix*

➟ having one entry in each row and in each column, and 0 elsewhere

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

Definition (Permutation Group on X).

The set of all permutations S_X on X makes the *permutation group on* X

Definition (Symmetric Group of order n, S_n).

The set of all permutations $S_n = \{\sigma_1, \sigma_2, \ldots, \sigma_{n!}\}$ makes the *symmetric group* on a n-element set $\{1, 2, \ldots, n\}$.

- If $X = \{1, 2, \ldots, n\}$ then $S_X = S_n$
  ➠ the symmetric groups on n-elements are permutation groups

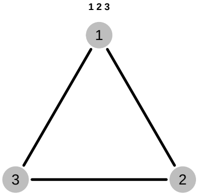
Definition (Dihedral Group of degree n, D_n, $n > 2$).

The set of all permutations which are symmetries on a regular n-sided polygon and the composition operation $\circ$ makes the *dihedral group* $(D_n, \circ)$.

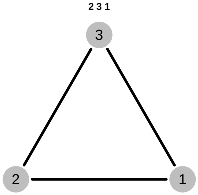
- the order of a dihedral group is twice its degree

Group of symmetries of the equilateral triangle

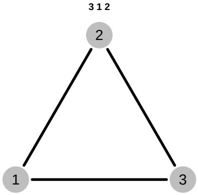
Dihedral group, D_3



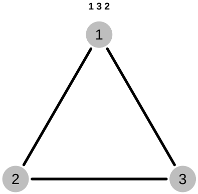
R_0



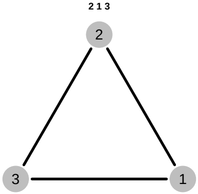
R_{120}



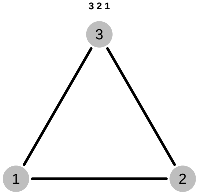
R_{240}



S_V



S_D



S_L

Dihedral group, D_3

Cayley table

$\circ$	R_0	R_{120}	R_{240}	S_V	S_D	S_L
R_0	R_0	R_{120}	R_{240}	S_V	S_D	S_L
R_{120}	R_{120}	R_{240}	R_0	S_D	S_L	S_V
R_{240}	R_{240}	R_0	R_{120}	S_L	S_V	S_D
S_V	S_V	S_L	S_D	R_0	R_{240}	R_{120}
S_D	S_D	S_V	S_L	R_{120}	R_0	R_{240}
S_L	S_L	S_D	S_V	R_{240}	R_{120}	R_0

Cayley colour graph

Definition (Cayley graph).
The *Cayley graph* Γ of a group G with respect to a generating set $C \subseteq G$:

$$\Gamma = \Gamma(G, C).$$

G is the node set in Γ

A generator $c \in C$ connects two nodes $a, b \in G$ whenever $b = ca$

➡ i.e. all pairs of the form $(a, c \cdot b)$ make the edge set in Γ

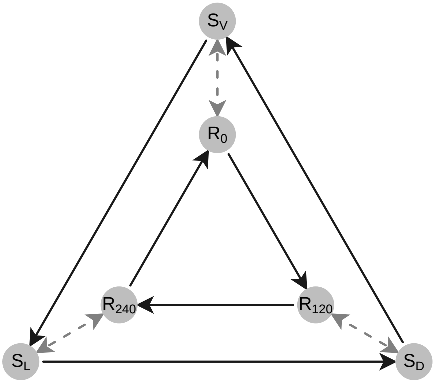
Cayley colour graph

Example (Cayley graph, $\mathbb{Z}_2$).

```
  e x
e e x
x x e

e=ee  => solid loop
e=xx  => solid loop

x=ex  => dashed arc
x=xe  => dashed arc
```

Dihedral group, D_3

Cayley graph



Group of symmetries of the equilateral triangle, D_3

```
# CONSTRUCT A TRIANGLE AS 3—CYCLE
tri <- transf(c("1, 2", "2, 3", "3, 1"))
```

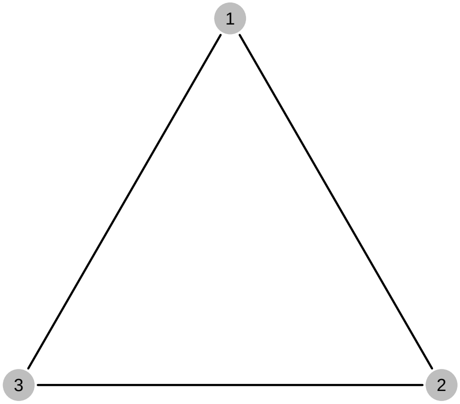
```
  1 2 3
1 0 1 0
2 0 0 1
3 1 0 0
```

```
# LOAD THE DISTINCT PERMUTATIONS ON THE EQUILATERAL TRIANGLE
load(file = "data/prm.rda")
```

```
     1 2 3
R0   1 2 3
R120 2 3 1
R240 3 1 2
SV   1 3 2
SD   2 1 3
SL   3 2 1
```

Equilateral triangle plot

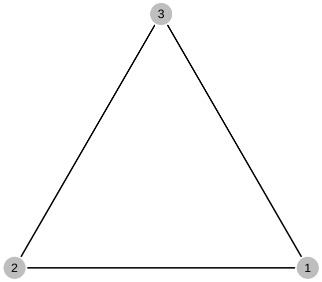
```
# PLOT 'tri' WITH COSTUMIZED OUTLINE
scp <- list(directed = FALSE, cex = 6, lwd = 3, ecol = 1, vcol = 8, pos = 0)
multigraph(tri, scope = scp)
```



Generators of the symmetric group, D_3

Rotation F

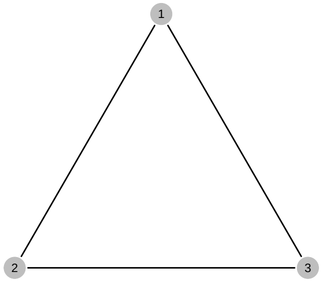
```r
trip <- perm(tri, clu = prm[2,])

multigraph(trip, scope = scp)
```



R_{120}

Reflection G

```r
trip <- perm(tri, clu = prm[4,])

multigraph(trip, scope = scp)
```



S_V

Generators of D_3

as permutation matrices

```
# DEFINE GENERATORS AS PERMUTATION MATRICES WITH A LEXICOGRAPHIC ORDER
CD3 <- transf(list(F = c("2, 1", "3, 2", "1, 3"),
+   G = c("1, 1", "3, 2", "2, 3")), type = "toarray", sort = TRUE)
```

```
, , F

  1 2 3
1 0 0 1
2 1 0 0
3 0 1 0

, , G

  1 2 3
1 1 0 0
2 0 0 1
3 0 1 0
```

Elements in the group structure, D_3

Function `strings` allows finding *word tables*

```
strings(CD3)
```

```
$wt
, , F        , , FF       , , GF

  1 2 3        1 2 3        1 2 3
1 0 0 1      1 0 1 0      1 0 0 1
2 1 0 0      2 0 0 1      2 0 1 0
3 0 1 0      3 1 0 0      3 1 0 0

, , G        , , FG       , , GG

  1 2 3        1 2 3        1 2 3
1 1 0 0      1 0 1 0      1 1 0 0
2 0 0 1      2 1 0 0      2 0 1 0
3 0 1 0      3 0 0 1      3 0 0 1
```

Equations in group structure, D_3 $(k = 3)$

Argument `equat` allows finding group equations with the identity

```
strings(CD3, equat = TRUE, k = 3)
```

```
$equat
$equat$F
[1] "F"   "GGF" "FGG"

$equat$G
[1] "G"   "GGG" "FGF"

$equat$FF
[1] "FF"  "GFG"

$equat$FG
[1] "FG"  "GFF"

$equat$GF
[1] "GF"  "FFG"

$equat$GG
[1] "GG"  "FFF"


$equate
$equate$e
[1] "e"   "GG"  "FFF"
```

Group structure, D_3

Function `semigroup` allows finding the group structure

➟ since "any group is a semigroup as well"

```
CD3S <- semigroup(CD3)
```

```
...

$st
[1] "F"  "G"  "FF" "FG" "GF" "GG"

$S
  1 2 3 4 5 6
1 3 4 6 5 2 1
2 5 6 4 3 1 2
3 6 5 1 2 4 3
4 2 1 5 6 3 4
5 4 3 2 1 6 5
6 1 2 3 4 5 6

attr(,"class")
[1] "Semigroup" "numerical"
```

Permutation of the group structure, D_3

`perm` for the rearrangement of elements' group structure in `CD3S`

```
CD3S <- perm(CD3S$S, clu = c(2,4,3,5,6,1))
```

```
  6 1 3 2 4 5
6 6 1 3 2 4 5
1 1 3 6 4 5 2
3 3 6 1 5 2 4
2 2 5 4 6 3 1
4 4 2 5 1 6 3
5 5 4 2 3 1 6
```

This comes from the string labels where `GG` in the identity element

```
..
$st
[1] "F"  "G"  "FF" "FG" "GF" "GG"
...
```

Depiction of the group structure, D_3

Cayley table

Relabeling of elements in group structure with `as.semigroup`

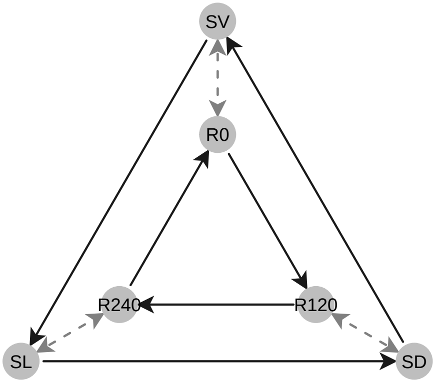
```
CD3S <- as.semigroup(CD3S, gens = c(2, 4),
+   lbs = c("R0", "R120", "R240", "SV", "SD", "SL"))
```

```
...
$st
[1] "R0"   "R120" "R240" "SV"   "SD"   "SL"

$gens
[1] "R120" "SV"

$S
       R0 R120 R240   SV   SD   SL
R0     R0 R120 R240   SV   SD   SL
R120 R120 R240   R0   SD   SL   SV
R240 R240   R0 R120   SL   SV   SD
SV     SV   SL   SD   R0 R240 R120
SD     SD   SV   SL R120   R0 R240
SL     SL   SD   SV R240 R120   R0

attr(,"class")
[1] "Semigroup" "symbolic"
```

Depiction of the group structure, D_3

Cayley graph

```
# PLOT CAYLEY COLOUR GRAPH WITH A 2—RADII CONCENTRIC LAYOUT
scpD3 <- list(cex = 7, lwd = 3, pos = 0, vcol = 8, tcex = 1.6)
ccgraph(CD3S, scope = scpD3, conc = TRUE, nr = 2)
```



Example: Group structure in social networks

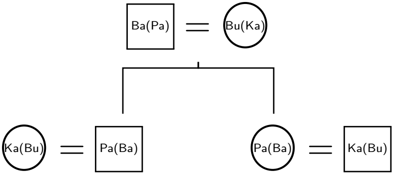
Kariera society kinship system

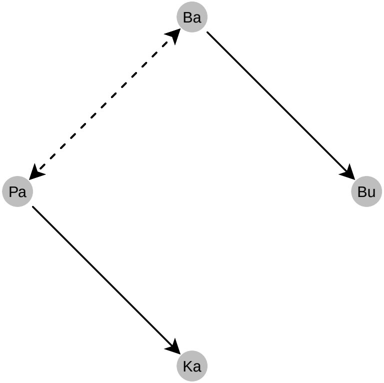
- Despite the symmetry, algebraic groups can model human societies
- Some primitive societies like the **Kariera** have kinship networks that follow the rules of a group structure
- Kariera has (had?) 4 clans with specific rules of marriage & descent: *Bakana*, *Burung*, *Karimera*, and *Palyeri*.

Kariera Rules for Marriage & Descent (I)

Bakana (Ba), Burung (Bu), Karimera (Ka), Palyeri (Pa)

Two types of descent rules among Bakana and Palyeri (ego male)

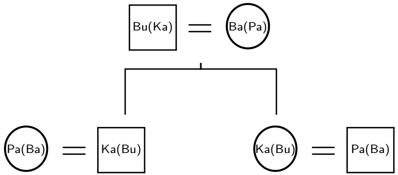


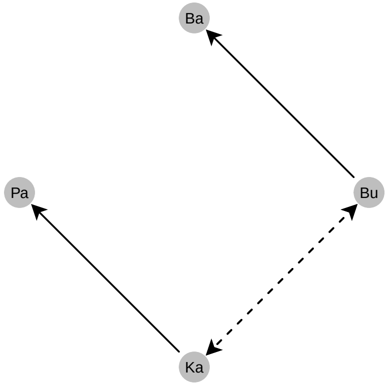


Kariera Rules for Marriage & Descent (II)

Bakana (Ba), Burung (Bu), Karimera (Ka), Palyeri (Pa)

Two types of descent rules among Burung and Karimera (ego male)

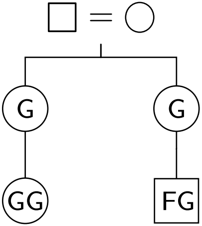




Parallel-cousins marriages in kinship networks

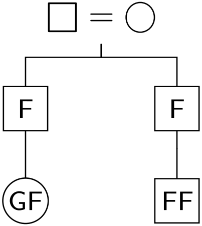
identifiers, F for male and G for female, are with right multiplication

$\mathsf{FG} = \mathsf{GG}$



(a) Matrilineal

$\mathsf{GF} = \mathsf{FF}$

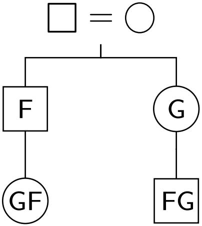


(b) Patrilineal

Cross-cousins marriages in kinship networks

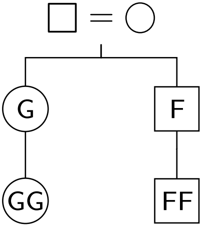
identifiers, F for male and G for female, are with right multiplication

$$\mathsf{FG} = \mathsf{GF}$$



(a) Matrilineal

$$\mathsf{FF} = \mathsf{GG}$$



(b) Patrilineal

Kariera kinship system

Group structure in societies

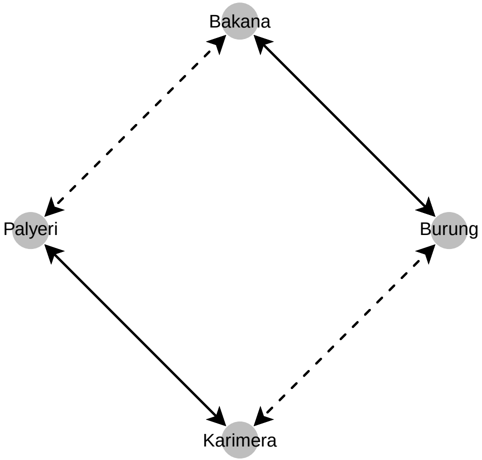
```
# CREATE PERMUTATION MATRICES FOR MARRIAGE & DESCENT RULES
kks <- transf(list(F = c("1, 2", "3, 4", "2, 1", "4, 3"),
+   G = c("1, 4", "2, 3", "3, 2", "4, 1")))
```

```
, , F

  1 2 3 4
1 0 1 0 0
2 1 0 0 0
3 0 0 0 1
4 0 0 1 0

, , G

  1 2 3 4
1 0 0 0 1
2 0 0 1 0
3 0 1 0 0
4 1 0 0 0
```

Kariera kinship system

Group structure in societies

```
# VISUALIZE MARRIAGE & DESCENT RULES AS MULTIGRAPH
scpKS <- c(scpD3, ecol = 1, collRecip = TRUE)
multigraph(kks, scope = scpKS, lbs = c("Bakana","Burung","Karimera","Palyeri"))
```



Set of Equations

The **set of equations** to detect allowed marriage types by commutation

```r
# THE EQUATIONS ALLOWS FINDING MARRIAGE TYPES IN 'kks'
strings(kks, equat = TRUE)
```

```
...
$st
[1] "F"  "G"  "FF" "FG"

$equat
$equat$FF
[1] "FF" "GG"

$equat$FG
[1] "FG" "GF"


$equate
$equate$e
[1] "e"  "FF" "GG"
```

☛ *Both cross-cousins marriages are permitted in the Kariera*

Multiplication table of the Group structure

The **multiplication table** reflects the group structure of the clan system

```
# SEMIGROUP WITH A SYMBOLIC FORMAT
semigroup(kks, type = "symbolic")
```

```
$dim
[1] 4

...

$ord
[1] 4

$st
[1] "F"  "G"  "FF" "FG"

$S
    F  G FF FG
F  FF FG  F  G
G  FG FF  G  F
FF  F  G FF FG
FG  G  F FG FF

attr(,"class")
[1] "Semigroup" "symbolic"
```

Algebraic Constraints in Group Structures

Two **algebraic constraints** for the analysis of the elementary structures:

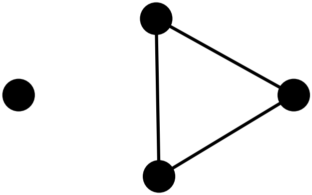
- *Set of equations* among different types of tie
- *Multiplication table* with relations between the different types of tie

☛ Complex structures have additional algebraic constraints

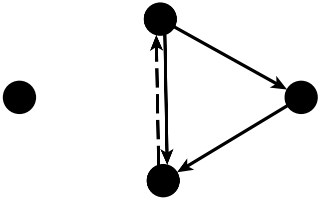
3. Multiplex networks

Multiplex networks

- Social networks are typically characterized by a single relationship
- But social life is more complex and people are embedded in different types of relations that are **interlocked** to each other



graph depicting a **simple** network



multigraph depicting a **multiple** network

☛ *find the right methods to analyse multiple types of tie simultaneously*

Tie interlock

- **Social structure** = Ties between actors
- **Relational structure** = Interrelations between relations
- **Role structure** = Relational system of aggregated relations

☛ we benefit from algebraic structures to represent relational systems

Semigroup

Algebraic structure

A **semigroup** is an algebraic structure with a set of elements with an associative operation attached to it:

$$\langle\, S,\ \circ \,\rangle$$

- S is the *underlying set*, closed under the operation
- $\circ$ is the *composition operation* on an ordered pair
  ➠ i.e. 'direct product' $\circ\colon S \times S \to S$
- for all $x, y, z \in S$, $\circ$ satisfies the associative law:

$$x \circ (y \circ z) \;=\; (x \circ y) \circ z$$

Semigroup of Relations

- The **semigroup of relations**, $S(R)$ serves to represent the relational structure in multiplex networks

- In $S(R)$, 'x' and 'y' are **generators** (or primitives) ties, whereas '$x \circ y$' constitutes a **compound** relation

  ➟ by *right multiplication* or "adding to the right"

- The elements in $S(R)$ are the *unique* representative strings made of generators (or generator) and (most likely) compounds relations as well

- The set of **equations** produces the unique strings in the system

  ➟ and also a closed system

Dyadic properties

multiplex networks

Tie Entrainment

Asymmetric character

Tie Exchange

Mutual character

Mixed pattern

Full pattern

Mutual character

Example of Semigroup of Relations, $S(R)$

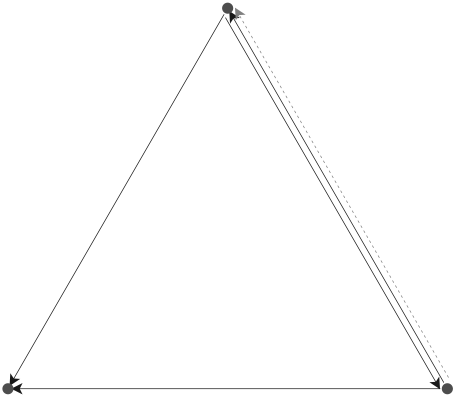
```
# CREATE PSEUDO RANDOM DATA
library("stats")
set.seed(123); arr1 <- array(runif(9), c(3, 3, 1))
set.seed(321); arr2 <- array(runif(9), c(3, 3, 1))
```

```
# zbind CREATES 3D ARRAY AND dichot DICHOTOMIZES WITH CUT-OFF VALUE
arr <- zbind(arr1, arr2)
arr <- dichot(arr, c = .5)
```

```
, , 1

     [,1] [,2] [,3]
[1,]    0    1    1
[2,]    1    1    1
[3,]    0    0    1

, , 2

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    1    0    0
[3,]    0    0    0
```

Example of Semigroup of Relations

```
# VISUALIZATION OF 'arr'
multigraph(arr)
```



Example $S(R)$: Bundle Patterns

```
# FUNCTION summaryBundles() REQUIRES A "Rel.Bundles" CLASS OBJECT
summaryBundles(bundles(arr))
```

```
                      Bundles
Asym1          ->{1} (1, 3)
Asym2          ->{1} (2, 3)
Mixd  <->{1} <-{2}  (1, 2)
```

```
bundle.census(arr)
```

```
      BUNDLES NULL ASYMM RECIP T.ENTR T.EXCH MIXED FULL
TOTAL       3    0     2     0      0      0     1    0
```

Example $S(R)$: Multiplication table

```
# SEMIGROUP OF RELATIONS IN SYMBOLIC FORMAT
semigroup(arr, type = "symbolic")$S
```

```
       1 2  11 21 211
 1    11 2  11 21 211
 2    21 2 211 21 211
 11   11 2  11 21 211
 21  211 2 211 21 211
 211 211 2 211 21 211

 ...
```

Example $S(R)$: Equations Set

```r
# EQUATIONS OF COMPOUNDS UNTIL LENGTH 3
strings(arr, equat = TRUE, k = 3)$equat
```

```
$`2`
[1] "2"   "22"  "12"  "122" "112" "222" "212"

$`11`
[1] "11"  "111"

$`21`
[1] "21"  "221" "121"
```

Partial Order

Complex structures with lack of symmetry

With semigroups typically there exists an ordering among its relations

- A **partial order** is defined by an *inclusion* relation $\leq$ among $x, y \in S(R)$ with the rule:

$$S(R)^{\leq}_{x,y} = \begin{cases} 1 & \text{iff relation } x \text{ is contained in relation } y \\ 0 & \text{otherwise} \end{cases}$$

where 'contained' implies that all ties in x are occurring in y as well

➟ A **partially ordered semigroup** is $S(R)$ with a partial order

Example $S(R)$: Partial order set (poset)

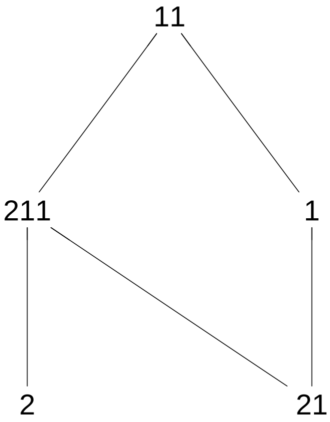
```r
# PARTIAL ORDER TABLE OF STRING RELATIONS
partial.order(strings(arr))
```

```
    1 2 11 21 211
1   1 0  1  0   0
2   0 1  1  0   1
11  0 0  1  0   0
21  1 0  1  1   1
211 0 0  1  0   1
attr(,"class")
[1] "Partial.Order" "strings"
```

Visualization of partial order structures

Hasse Diagram

```
# FUNCTION diagram PLOTS POSETS (REQUIRES "Rgraphviz" PACKAGE)
diagram(partial.order(strings(arr)))
```



Issues with the Semigroup Structure

- Modelling a multiple network by $S(R)$ typically results in a quite large structure, even if the system is small

- An important task is to reduce complexity of the network

  ➟ this is done by grouping different classes of actors

- *Blockmodeling* is an effective way to reduce the network and keeping the essential structure of the system

➥ **Positional Analysis**

☛ But it needs to preserve the network multiplicity of ties

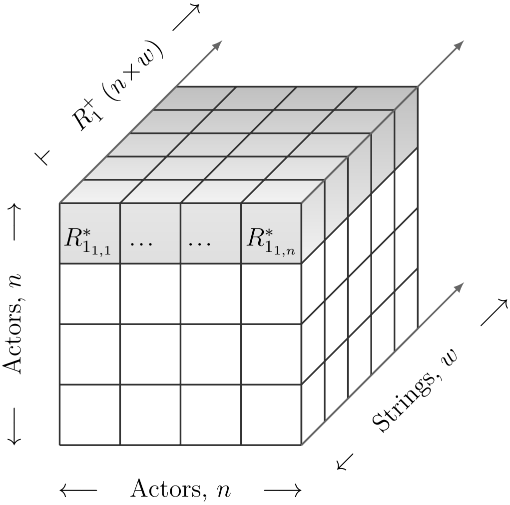
Equivalence Types in Positional Analysis

Each *type* of graph homomorphism (a structure-preserving mapping) induces to a particular kind of equivalence

➟ which represents a system of *positions* and *roles* of the network

- Equivalences from a **global perspective**:
  - Structural (Lorrain & White, 1971)
  - Automorphic (Winship & Mandel, 1983; Everett, 1985)
  - Regular (Sailer [Boyd], 1978; White & Reitz, 1983)
  - Generalized (Batagelj et al, 1992; Doreian et al, 1994)
- Equivalences from a **local perspective**:
  - Local Role (Winship & Mandel, 1983; Mandel, 1983)
  - **Compositional** (Breiger & Pattison, 1986; Mandel, 1978)

Compositional Equivalence: Relation-Box



Compositional Equivalence: Person Hierarchies

Person Hierarchies

- Builds on the ordering among the actors' Role Relations in a particular Relation Plane (shadow part in the Relation-Box)

- All perceived inclusions in R_l^+ represents the **Person Hierarchy** H_l defined for $l, i, j \in \mathscr{X}$ and relation x as:

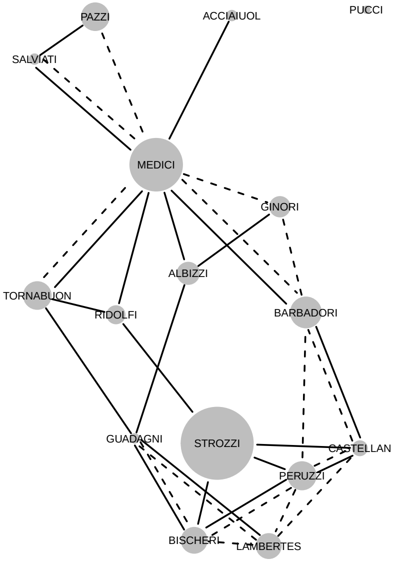
$$H_{l_{ij}} = \begin{cases} 1 & \text{iff } R^*_{l_{xi}} \leq R^*_{l_{xj}} \\ 1 & \text{iff } R^*_{l_{xi}} = R^*_{l_{xj}} \\ 0 & \text{iff } R^*_{l_{xi}} \nleq R^*_{l_{xj}} \\ 0 & \text{iff } \sum R^*_{l_{xi}} = 0 \end{cases}$$

- A **Cumulated Person Hierarchy** $\mathscr{H}$ matrix is based on the union of all person hierarchies with transitive closure

  ➟ the establishment of roles and positions are from the perspectives of individual actors, but it also considers common relational features

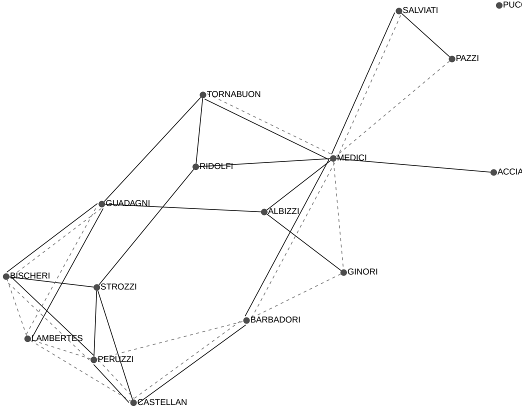
Compositional Equivalence: Undirected Networks

Florentine Families (Padgett). Solid: Marriage. Dashed: Business



Florentine Families

```
# FLORENTINE FAMILIES DATA SET AS A UCINET DL FILE
flf <- read.dl(file = "http://moreno.ss.uci.edu/padgett.dat")
multigraph(flf, directed = FALSE, layout = "force", seed = 1)
```



Or locally `read.dl(file ="data/padgett.dl")`

Florentine Families

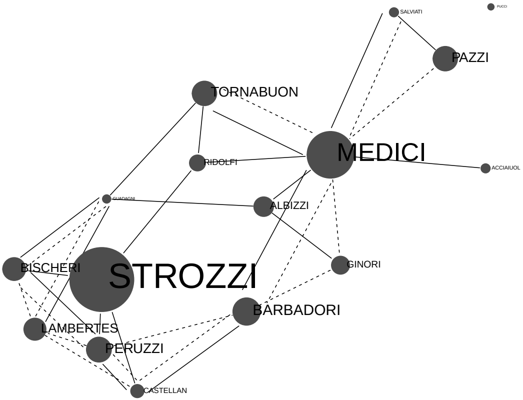
```
# ACTOR ATTRIBUTES
flfa <- read.dl(file = "http://moreno.ss.uci.edu/padgw.dat")
flfa <- flfa[order(rownames(flfa)), ]
```

```
           WEALTH #PRIORS #TIES
ACCIAIUOL      10      53     2
ALBIZZI        36      65     3
BARBADORI      55       0    14
BISCHERI       44      12     9
CASTELLAN      20      22    18
GINORI         32       0     9
GUADAGNI        8      21    14
LAMBERTES      42       0    14
MEDICI        103      53    54
PAZZI          48       0     7
PERUZZI        49      42    32
PUCCI           3       0     1
RIDOLFI        27      38     4
SALVIATI       10      35     5
STROZZI       146      74    29
TORNABUON      48       0     7
```

Or locally `read.dl(file ="data/padgw.dl")`

Florentine Families

```
# PLOTTING WITH ACTOR ATTRIBUTES
multigraph(flf, directed = FALSE, "force", seed = 1, ecol = 1, cex = flfa[,1])
```



Florentine Families

```
# INSPECT THE NETWORK RELATIONAL SYSTEM
rel.sys(flf, bonds = "full")$incl
```

```
 [1] "BARBADORI" "BISCHERI"  "CASTELLAN" "GUADAGNI"  "LAMBERTES" "MEDICI"    "PERUZZI"
 [8] "SALVIATI"  "TORNABUON"
```

```
# WHO IS NOT LINKED AT BOTH LEVELS
rel.sys(flf, bonds = "full")$excl
```

```
 [1] "ACCIAIUOL" "ALBIZZI"   "GINORI"    "PAZZI"     "PUCCI"     "RIDOLFI"   "STROZZI"
```

Compositional Equivalence: Relation-Box

Florentine Families

```
# FUNCTION TO CONSTRUCT THE RELATION—BOX
formals("rbox")
```

```
$w


$transp
[1] FALSE

$smpl
[1] FALSE

$k
[1] 3

$tlbs
```

Compositional Equivalence: Cumulated Person Hierarchy

Florentine Families

```
# FUNCTION cph() TO CONSTRUCT THE CUMULATED PERSON HIERARCHY
# INPUT MUST BE A "Rel.Box" CLASS. OUTPUT IS A "Partial.Order" "CPH" CLASS
cph(rbox(flf))
```

```
          ACCIAIUOL ALBIZZI BARBADORI BISCHERI CASTELLAN GINORI GUADAGNI LAMBERTES MEDICI PAZZI
ACCIAIUOL         1       1         1        1         1      1        1         1      1     1
ALBIZZI           1       1         1        1         1      1        1         1      1     1
BARBADORI         1       1         1        1         1      1        1         1      1     1
BISCHERI          1       1         1        1         1      1        1         1      1     1
CASTELLAN         1       1         1        1         1      1        1         1      1     1
GINORI            1       1         1        1         1      1        1         1      1     1
GUADAGNI          1       1         1        1         1      1        1         1      1     1
LAMBERTES         1       1         1        1         1      1        1         1      1     1
MEDICI            1       1         1        1         1      1        1         1      1     1
PAZZI             1       1         1        1         1      1        1         1      1     1
PERUZZI           1       1         1        1         1      1        1         1      1     1
PUCCI             0       0         0        0         0      0        0         0      0     0
RIDOLFI           1       1         1        1         1      1        1         1      1     1
SALVIATI          1       1         1        1         1      1        1         1      1     1
STROZZI           1       1         1        1         1      1        1         1      1     1
TORNABUON         1       1         1        1         1      1        1         1      1     1
attr(,"class")
[1] "Partial.Order" "CPH"
```

(Extract)

Compositional Equivalence: Cumulated Person Hierarchy

Florentine Families

```r
cph(rbox(flf, k = 4))
```

```
          ACCIAIUOL ALBIZZI BARBADORI BISCHERI CASTELLAN GINORI GUADAGNI LAMBERTES MEDICI PAZZI
ACCIAIUOL         1       1         1        1         1      1        1         1      1     1
ALBIZZI           1       1         1        1         1      1        1         1      1     1
BARBADORI         1       1         1        1         1      1        1         1      1     1
BISCHERI          1       1         1        1         1      1        1         1      1     1
CASTELLAN         1       1         1        1         1      1        1         1      1     1
GINORI            1       1         1        1         1      1        1         1      1     1
GUADAGNI          1       1         1        1         1      1        1         1      1     1
LAMBERTES         1       1         1        1         1      1        1         1      1     1
MEDICI            1       1         1        1         1      1        1         1      1     1
PAZZI             1       1         1        1         1      1        1         1      1     1
PERUZZI           1       1         1        1         1      1        1         1      1     1
PUCCI             0       0         0        0         0      0        0         0      0     0
RIDOLFI           1       1         1        1         1      1        1         1      1     1
SALVIATI          1       1         1        1         1      1        1         1      1     1
STROZZI           1       1         1        1         1      1        1         1      1     1
TORNABUON         1       1         1        1         1      1        1         1      1     1
attr(,"class")
[1] "Partial.Order" "CPH"
```

(Extract)

Compositional Equivalence: Cumulated Person Hierarchy

Florentine Families

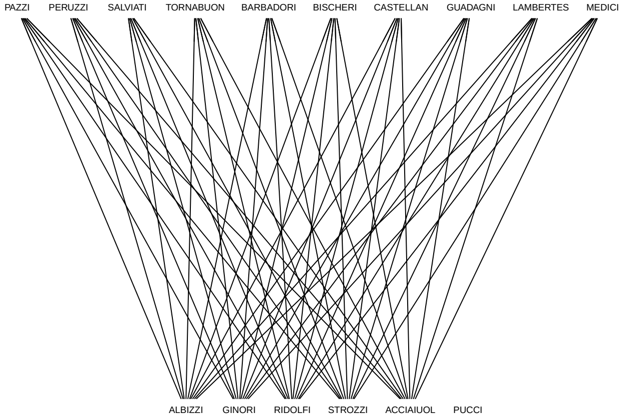
```
# TEST OBJECTS FOR EXACT EQUALITY
identical(cph(rbox(flf, k = 3)), cph(rbox(flf, k = 4)))

  [1] TRUE
```

```
identical(cph(rbox(flf, k = 4)), cph(rbox(flf, k = 5)))

  [1] FALSE
```

Visualization of the Poset

Florentine Families

```
# CPH IS A POSET
diagram(cph(rbox(flf, k = 5)))
```



Compositional Equivalence: Positional Analysis

Florentine Families

```r
# LEVELS IN THE PLOTED HASSE DIAGRAM
diagram.levels(cph(rbox(flf, k = 5)))
```

```
          2       2         1        1         1      2        1         1      1     1
1 ACCIAIUOL ALBIZZI BARBADORI BISCHERI CASTELLAN GINORI GUADAGNI LAMBERTES MEDICI PAZZI
        1     3       2        1       2         1
1 PERUZZI PUCCI RIDOLFI SALVIATI STROZZI TORNABUON
```

```r
# OBTAIN THE CLUSTERING WITH perm ARGUMENT
diagram.levels(cph(rbox(flf, k = 5)), perm = TRUE)$clu
```

```
[1] 2 2 1 1 1 2 1 1 1 1 1 3 2 1 2 1
```

☛ However, levels in the plotted Hasse diagram are not always the best criteria for classifying the actors

Compositional Equivalence: Positional Analysis

Florentine Families

```r
# FIRST RECORD THE CLUSTERING VECTOR
flfclu <- diagram.levels(cph(rbox(flf, k = 5)), perm = TRUE)$clu

# APPLY CLUSTERING TO PRODUCE A POSITIONAL SYSTEM WITH FUNCTION reduc()
flfps <- reduc(flf, clu = flfclu)
```

```
, , PADGM

  2 1 3
2 1 1 0
1 1 1 0
3 0 0 0

, , PADGB

  2 1 3
2 1 1 0
1 1 0 0
3 0 0 0
```

Compositional Equivalence: Role Structure

Florentine Families

```
# THE SEMIGROUP OF THE POSITIONAL SYSTEM IN DEFAULT FORMAT
semigroup(flfps)
```

```
$dim
[1] 3

$gens

...

$ord
[1] 2

$st
[1] "PADGM" "PADGB"

$S
  1 2
1 1 1
2 1 1

attr(,"class")
[1] "Semigroup" "numerical"
```

Compositional Equivalence with Actor Attributes

For a given attribute defined in α, and for $i = x_1, x_2, \ldots, x_n$, attribute information is analyzed in relational terms where pair of vectors are element of an indexed matrix $\mathbf{A}^\alpha$ as:

$$a_{ij}^{\alpha} =; \delta_{ij} ,$$

Here

$$c_i = \begin{cases} 1 & \text{if the corresponding attribute is tied to actor } i \\ 0 & \text{otherwise.} \end{cases}$$

And δ_{ij} is defined for nodes $i, j = x_1, x_2, \ldots, x_n$ in $\mathscr{X}$ by the Kronecker delta function as:

$$\delta_{ij} = \begin{cases} 1 & \text{for } \ i = j \\ 0 & \text{for } \ i \neq j. \end{cases}$$

☛ That is, $\mathbf{A}^\alpha$ is a **diagonal matrix**.

Actor Attributes in Relational Structures

Florentine Families

```
           WEALTH #PRIORS #TIES
 ACCIAIUOL     10      53     2
 ALBIZZI       36      65     3
 BARBADORI     55       0    14
 BISCHERI      44      12     9
 CASTELLAN     20      22    18
 GINORI        32       0     9
 GUADAGNI       8      21    14
 LAMBERTES     42       0    14
 MEDICI       103      53    54
 PAZZI         48       0     7
 PERUZZI       49      42    32
 PUCCI          3       0     1
 RIDOLFI       27      38     4
 SALVIATI      10      35     5
 STROZZI      146      74    29
 TORNABUON     48       0     7
```

```
# FUNCTION read.srt() TRANSFORMS DATA FRAMES INTO ARRAYS
read.srt(flfa, attr = TRUE, rownames = TRUE)

# SPLIT RICH ACTORS FROM THE VERY RICH ONES AND BIND IT TO THE NETWORK
fw <- dichot(read.srt(flfa, attr = TRUE, rownames = TRUE)[, , 1], c = 40)
flfw <- zbind(flf, fw)
```

Compositional Equivalence: CPH with Actor Attributes

Florentine Families

```
# TEST OBJECTS FOR EXACT EQUALITY
identical(cph(rbox(flfw, k = 2)), cph(rbox(flfw, k = 3)))
```

```
[1] TRUE
```

```
identical(cph(rbox(flfw, k = 3)), cph(rbox(flfw, k = 4)))
```

```
[1] TRUE
```

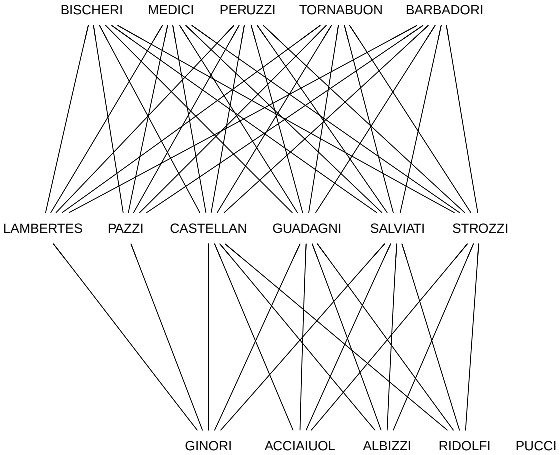
```
identical(cph(rbox(flfw, k = 4)), cph(rbox(flfw, k = 5)))
```

```
[1] FALSE
```

Hasse Diagram of $\mathscr{H}$ with Actor Attributes

Florentine Families

```r
diagram(cph(rbox(flfw, k = 5)))
```



Positional Analysis with Actor Attributes

Florentine Families

```
# POSITIONAL SYSTEM WITH THE CLUSTERING INFO OF THE HASSE DIAGRAM
flfwclu <- diagram.levels(cph(rbox(flfw, k = 5)), perm = TRUE)$clu
flfwps <- reduc(flfw, clu = flfwclu)
```

```
, , PADGM

  3 1 2 4
3 1 1 1 0
1 1 1 1 0
2 1 1 1 0
4 0 0 0 0

, , PADGB

  3 1 2 4
3 1 1 1 0
1 1 1 0 0
2 1 0 0 0
4 0 0 0 0

, , 3

  3 1 2 4
3 1 0 0 0
1 0 1 0 0
2 0 0 0 0
4 0 0 0 0
```

Algebraic Constraint: Role Table

Florentine Families Role Structure with Actor Attributes

```
# SEMIGROUP OF ROLE RELATIONS WITH COSTUMIZED LABELS
semigroup(flfwps, type = "symbolic", lbs = c("M", "B", "W"))$S

# OR EVEN BETTER...
dimnames(flfwps)[3][[1]] <- c("M", "B", "W")
semigroup(flfwps, type = "symbolic")$S
```

```
      M  B   W  MW  BW WM WB WMW
M     M  M  MW  MW  MW  M  M  MW
B     M  M  BW  MW  MW  M  M  MW
W    WM WB   W WMW WMW WM WB WMW
MW    M  M  MW  MW  MW  M  M  MW
BW    M  M  BW  MW  MW  M  M  MW
WM   WM WM WMW WMW WMW WM WM WMW
WB   WM WM WMW WMW WMW WM WM WMW
WMW  WM WM WMW WMW WMW WM WM WMW
```

Algebraic Constraint: Set of Equations

Florentine Families Role Structure with Actor Attributes

```
# FUNCTION strings() SERVES TO FIND EQUATIONS AMONG RELATIONS
flfwst <- strings(flfwps, equat = TRUE, k = 3)$equat
```

```
$M
 [1] "M"   "MM"  "BB"  "MB"  "BM"  "MMM" "BBM" "MBB" "MMB" "BBB" "BMM" "BMB" "MBM" "MWM"
[15] "BWB" "MWB" "BWM"

$W
[1] "W"   "WW"  "WWW"

$MW
[1] "MW"  "MWW" "MMW" "BBW" "MBW" "BMW"

$BW
[1] "BW"  "BWW"

$WM
[1] "WM"  "WWM" "WMM" "WBB" "WMB" "WBM"

$WB
[1] "WB"  "WWB"

$WMW
[1] "WMW" "WBW"
```

Algebraic Constraint: Partial Ordering

Florentine Families Role Structure with Actor Attributes

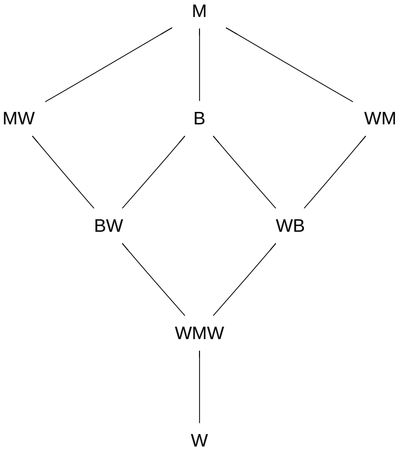
```
# PARTIAL ORDERING OF STRING RELATIONS
partial.order(flfwst, type = "strings")
```

```
    M B W MW BW WM WB WMW
M   1 0 0  0  0  0  0   0
B   1 1 0  0  0  0  0   0
W   1 1 1  1  1  1  1   1
MW  1 0 0  1  0  0  0   0
BW  1 1 0  1  1  0  0   0
WM  1 0 0  0  0  1  0   0
WB  1 1 0  0  0  1  1   0
WMW 1 1 0  1  1  1  1   1
attr(,"class")
[1] "Partial.Order" "strings"
```

Hasse Diagram for the Partial Order of String Relations

Florentine Families Role Structure with Actor Attributes

```
diagram(partial.order(flfwst, type = "strings"))
```



Decomposition of Relational Structures

Subdirect representation

- An **aggregated** role structure is obtained by means of synthesis rules of the relational system

- A *subdirect* representation implies finding **congruence** relations, which are correspondences that preserve the operation
  ➟ certain overlapping exists with this synthesis rule

  Function `cngr` computes the congruence relations in *abstract semigroups*

  Function `fact` computes induced inclusions in partial order of $S(R)$
  ➟ where π-relations are partitions on the *partially ordered semigroup*

- The decomposition for both cases is through function `decomp`

Decomposition of Relational Structures

Florentine Families Role Structure with Actor Attributes

First record the partially ordered structure

```
# SEMIGROUP OF ROLE RELATIONS
flfrt <- semigroup(flfwps, type = "symbolic")
```

```
# PARTIAL ORDERING OF STRING ROLE RELATIONS
flfpo <- partial.order(flfwst, type = "strings")
```

- Function `fact` performs the factorisation by *induced inclusions* to the partial order
  - ➟ as the full factorization from the PACNET module of StOCNET
- Function `pi.rels` finds partition relations on $S(R)$ and the partial order
  - ➟ π-relations are based on the output from the induced inclusions

Factorization of the partially ordered semigroup

Decomposition of Relational Structures

```r
# FACTORISATION OF ROLE TABLE WITH THE PARTIAL ORDER STRUCTURE
flfii <- fact(flfrt, flfpo)
```

```
$iin
$iin$`1, 2`
[1] "4, 2" "4, 5" "6, 2" "6, 7"

$iin$`4, 2`
[1] "4, 2" "4, 5"

$iin$`6, 2`
[1] "6, 2" "6, 7"
...

$atm
$atm$`4, 2`
[1] "4, 2" "4, 5"

$atm$`6, 2`
[1] "6, 2" "6, 7"


$mc
$mc[[1]]
```

Factorization of the partially ordered semigroup

Decomposition of Relational Structures

```
# OBTAIN π−RELATIONS FROM INDUCED INCLUSIONS
flfpr <- pi.rels(flfii, flfpo, po.incl = TRUE)
```

```
flfpr$pi
```

```
$pi
, , 1, 2

  1 2 3 4 5 6 7 8
1 1 0 0 0 0 0 0 0
2 1 1 0 0 0 0 0 0
3 1 1 1 1 1 1 1 1
4 1 1 0 1 1 0 0 0
5 1 1 0 1 1 0 0 0
6 1 1 0 0 0 1 1 0
7 1 1 0 0 0 1 1 0
8 1 1 0 1 1 1 1 1
...
```

Factorization of the partially ordered semigroup

π-relations of the atoms

```
flfpr$at
```

```
, , 1

  1 2 3 4 5 6 7 8
1 1 0 0 0 0 0 0 0
2 1 1 0 0 0 0 0 0
3 1 1 1 1 1 1 1 1
4 1 1 0 1 1 0 0 0
5 1 1 0 1 1 0 0 0
6 1 0 0 0 0 1 0 0
7 1 1 0 0 0 1 1 0
8 1 1 0 1 1 1 1 1

, , 2
...
```

Factorization of the partially ordered semigroup

π-relations of the meet-complements of the atoms

```
flfpr$mc
```

```
[[1]]
  1 2 3 4 5 6 7 8
1 1 0 0 1 0 0 0 0
2 1 1 0 1 0 0 0 0
3 1 1 1 1 1 1 1 1
4 1 0 0 1 0 0 0 0
5 1 1 0 1 1 0 0 0
6 1 1 0 1 1 1 1 1
7 1 1 0 1 1 1 1 1
8 1 1 0 1 1 1 1 1

[[2]]
...
```

Aggregated role structures from meet-complements

Decomposition of the Florentine Families Role Structure with Actor Attributes

```
# LOGIC 1
decomp(flfrt, flfpr, type = "mc", reduc = TRUE)
```

```
$clu
$clu[[1]]
  M   B   W  MW  BW  WM  WB WMW
  1   2   3   1   4   5   5   5

...

$IM[[1]]
    M  B  W BW WM
M   M  M  M  M  M
B   M  M BW  M  M
W  WM WM  W WM WM
BW  M  M BW  M  M
WM WM WM WM WM WM
```

Aggregated role structures from meet-complements

Decomposition of the Florentine Families Role Structure with Actor Attributes

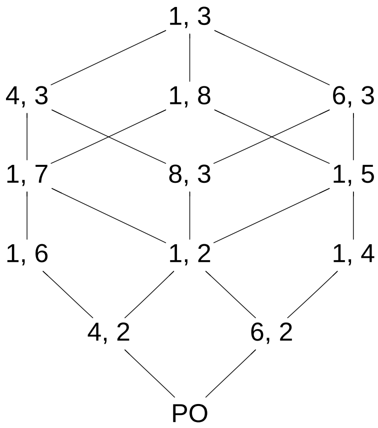
```r
# LOGIC 2
decomp(flfrt, flfpr, type = "mc", reduc = TRUE)
```

```
$clu[[2]]
  M   B   W  MW  BW  WM  WB WMW
  1   2   3   4   4   1   5   4

...

$IM[[2]]
   M  B  W MW WB
M  M  M MW MW  M
B  M  M MW MW  M
W  M WB  W MW WB
MW M  M MW MW  M
WB M  M MW MW  M
```

Decomposition of Relational Structures

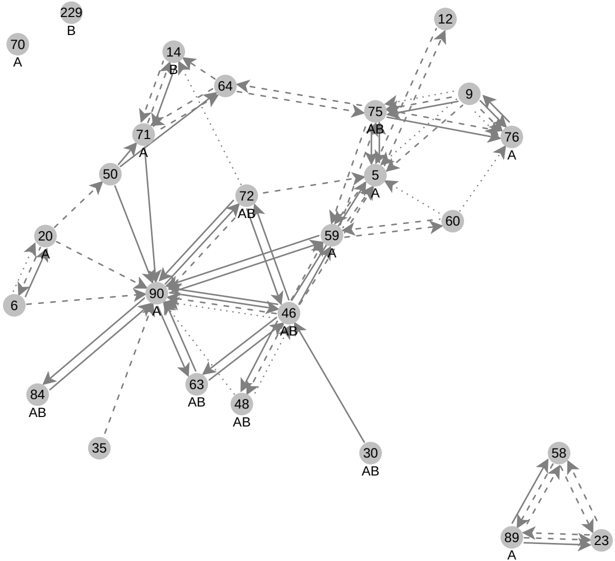
Florentine Families Role Structure with Actor Attributes

```
# hierarchy of aggregated role relations with partial order
diagram(partial.order(flfpr, type = "pi.rels", po.incl = TRUE))
```



Compositional Equivalence: Directed Networks

Incubator network. Solid: Collaboration. Dotted: Friendship. Dashed: Competition



Incubator network

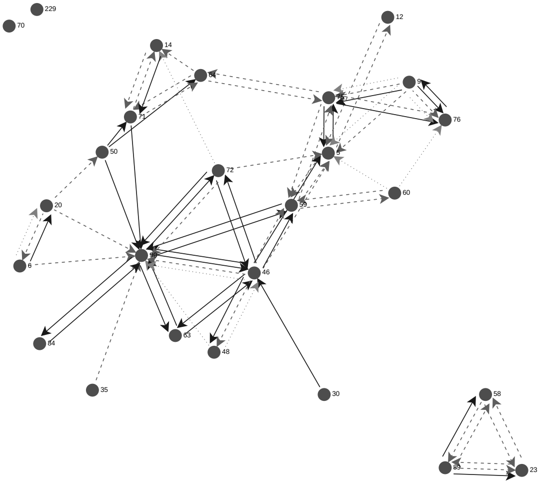
```
# INCUBATOR NETWORK ('A') DATA SET
data("incubA")
str(incubA)
```

```
List of 2
 $ net: num [1:26, 1:26, 1:5] 0 0 0 0 0 0 0 0 0 1 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:26] "5" "6" "9" "12" ...
  .. ..$ : chr [1:26] "5" "6" "9" "12" ...
  .. ..$ : chr [1:5] "C" "F" "K" "A" ...
 $ IM : num [1:4, 1:4, 1:7] 1 1 1 0 0 1 0 0 1 0 ...
...
```

```
# RECORD NETWORK AND ACTOR ATTRIBUTES
netA <- incubA$net[,,1:3]
attA <- incubA$net[,,4:5]
```

Incubator network

```r
multigraph(netA, layout = "force", seed = 1)
```



Positional Analysis: Directed Networks

Incubator network

- Compositional equivalence with directed networks performs better by including **relational contrast** in the modeling
  - ➟ This is operationalized through the *transpose* of the primitive ties

```
netAat <- zbind(netA, attA)
dimnames(netAat)[3][[1]]

  [1] "C" "F" "K" "A" "B"
```

- Function `rbox` can generate tie transposes

- However, since actor attributes are represented by diagonal matrices, it does not make any sense to include the transposes in the modeling
  - ➟ we control the labeling of transposes through argument `tlbs`

```
rbox(netAat, transp = TRUE, tlbs = c("D","G","L",NA,NA))
```

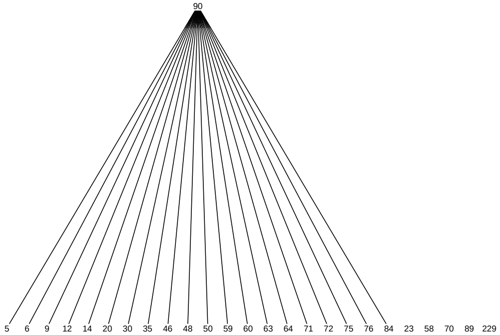
Cumulated Person Hierarchy *Incubator network*

```
cph(rbox(netAat, transp = TRUE, tlbs = c("D","G","L",NA,NA), k = 2))


     5 6 9 12 14 20 23 30 35 46 48 50 58 59 60 63 64 70 71 72 75 76 84 89 90 229
  5  1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  6  1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  9  1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  12 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  14 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  20 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  23 0 0 0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0   0
  30 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  35 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  46 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  48 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  50 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  58 0 0 0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0   0
  59 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  60 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  63 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  64 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  70 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0   0
  71 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  72 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  75 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  76 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  84 1 1 1  1  1  1  0  1  1  1  1  1  0  1  1  1  1  0  1  1  1  1  1  0  1   0
  89 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0   0
  90 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1   0
  229 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0   1
  ...
```

Cumulated Person Hierarchy

Compositional equivalence with directed networks

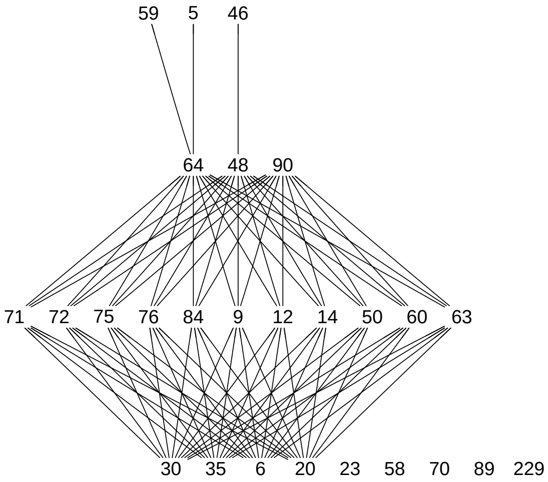
```
diagram(cph(rbox(netAat, transp = TRUE, tlbs = c("D","G","L",NA,NA), k = 2)))
```



☛ not an optimal structure

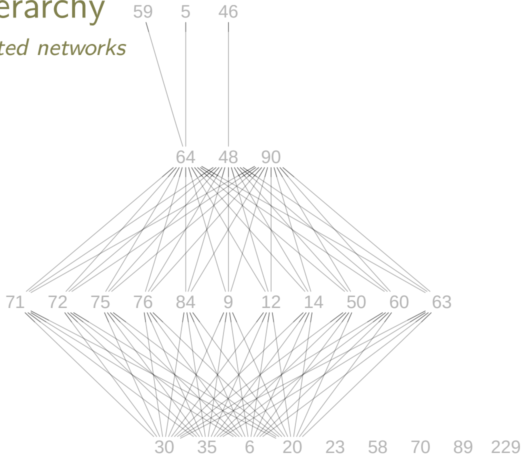
Cumulated Person Hierarchy

```
diagram(cph(rbox(netAat, transp = TRUE, tlbs = c("D","G","L",NA,NA), k = 3)))
```



Cumulated Person Hierarchy

Compositional equivalence, directed networks




```
as.table(rbind(dimnames(netAat)[1][[1]],
+     c(3,2,1,1,1,2,4,2,2,3,3,1,4,3,1,1,3,4,1,1,1,1,1,4,3,4)))

  A B C D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
A 5 6 9 12 14 20 23 30 35 46 48 50 58 59 60 63 64 70 71 72 75 76 84 89 90 229
B 3 2 1 1  1  2  4  2  2  3  3  1  4  3  1  1  3  4  1  1  1  1  1  4  3  4
```

Positional System for the Incubator network

```
netArb <- rbox(netAat, transp = TRUE, tlbs = c("D","G","L",NA,NA), k = 3)

netAatps <- reduc(netArb$w,
+     clu = c(3,2,1,1,1,2,4,2,2,3,3,1,4,3,1,1,3,4,1,1,1,1,1,4,3,4))
```

C

1	0	1	0
0	1	1	0
1	0	1	0
0	0	0	1

F

1	0	1	0
1	1	1	0
1	0	1	0
0	0	0	1

K

1	0	1	0
0	1	0	0
0	0	1	0
0	0	0	0

A

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

D

1	0	1	0
0	1	0	0
1	1	1	0
0	0	0	1

G

1	1	1	0
0	1	0	0
1	1	1	0
0	0	0	1

L

1	0	0	0
0	1	0	0
1	0	1	0
0	0	0	0

B

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Role Structure of Incubator network

Three **algebraic constraints** of the Role Structure:

```
# ROLE TABLE
semigroup(netAatps, type = "symbolic")
```

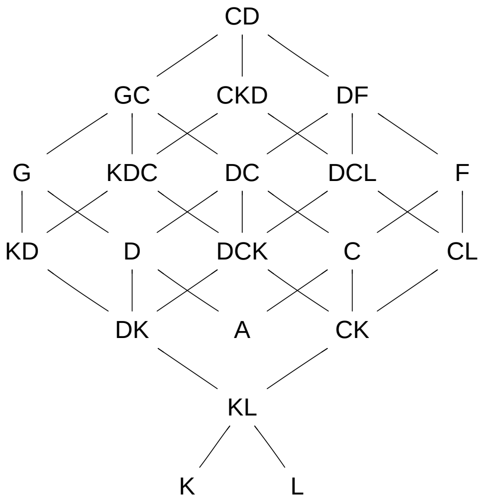
```
# SET OF EQUATIONS
netAatst <- strings(netAatps, equat = TRUE, k = 3)
netAatst$equat
```

```
# SET OF INCLUSIONS
partial.order(netAatst, type = "strings")
```

Hasse Diagram with the Set of Inclusions

Role structure of Incubator network

```r
diagram(partial.order(netAatst, type = "strings"))
```



Decomposition of Relational Structures: Incubator network

```r
# FIRST RECORD THE ROLE AND PARTIAL ORDER TABLES
netAatrt <- semigroup(netAatps, type = "symbolic")
netAatpo <- partial.order(netAatst, type = "strings")
```

```r
## CONGRUENCES IN THE ABSTRACT SEMIGROUP
cngr(netAatrt)
```

```r
# UNIQUE CONGRUENCES IN THE PARTIALLY ORDERED SEMIGROUP
netAatcg <- cngr(S = netArt, PO = netApo, unique = TRUE)
```

```r
# DECOMPOSITION OF ROLE TABLES BASED ON CONGRUENCE CLASSES
decomp(netAatrt, netAatcg, type = "cc")
```

```r
# DECOMPOSITION WITH THE REDUCTION OPTION
decomp(netAatrt, netAatcg, type = "cc", reduc = TRUE)
```

4. Signed networks

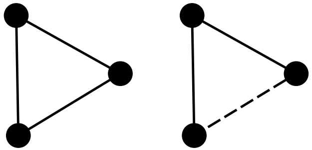
Structural Balance

- Simmel (1950) studied "conflict as a mechanism for integration" in triadic relations
- Heider (1958) developed the **Structural Balance** theory as a special cases of transitivity
- Structural Balance theory applies to networks to see whether the system has an inherent equilibrium or not

"all positive ties within groups; all negative ties between groups"

Structural Balance

- A balanced structure is represented by a **signed network**
  - ➟ a special case of a multiple network



- Paths in signed graphs are positive when they have an even number of negative edges; otherwise negative

☛ **extension**: a path/semipath is ambivalent iff contains at least one ambivalent edge

Structures in Balance Theory

balanced	$\rightarrow$	**clusterable**	$\rightarrow$	'*weak*' clusterable
(Cartwright & Harary, 1956)		(Davis, 1967)		

$\circ$	p	n
p	p	n
n	n	p

Classical

$\circ$	p	n	a
p	p	n	a
n	n	a	a
a	a	a	a

Extended

p $\rightarrow$ positive n $\rightarrow$ negative a $\rightarrow$ ambivalent

Semiring

Algebraic structure

A **semiring** is an object set endowed with a pair operations, multiplication and addition, together with two neutral elements:

$$\langle\ Q,\ +,\ \cdot\ ,\ 0,\ 1\ \rangle$$

properties:

- closed, associative, and commutative under addition
- multiplication distributes over addition, i.e. for all $\mathsf{p}, \mathsf{n}, \mathsf{a} \in Q$:

$$\mathsf{p}\ \cdot\ (\mathsf{n} + \mathsf{a}) = (\mathsf{p}\ \cdot\ \mathsf{n}) + (\mathsf{p}\ \cdot\ \mathsf{a}) \quad \text{and} \quad (\mathsf{p} + \mathsf{n})\ \cdot\ \mathsf{a} = (\mathsf{p}\ \cdot\ \mathsf{a}) + (\mathsf{n}\ \cdot\ \mathsf{a})$$

☛ *Semirings help us to evaluate the relational system in terms of balance theory by looking at paths and semipaths*

Semiring Operations

$\cdot$	o	n	p	a
o	o	o	o	o
n	o	p	n	a
p	o	n	p	a
a	o	a	a	a

$+$	o	n	p	a
o	o	n	p	a
n	n	n	a	a
p	p	a	p	a
a	a	a	a	a

Balance

$\cdot$	o	n	p	a	q
o	o	o	o	o	o
n	o	q	n	n	q
p	o	n	p	a	q
a	o	n	a	a	q
q	o	q	q	q	q

$+$	o	n	p	a	q
o	o	n	p	a	q
n	n	n	a	a	n
p	p	a	p	a	p
a	a	a	a	a	a
q	q	n	p	a	q

Clustering

Semiring function

```
# ARGUMENTS IN FUNCTION semiring()
formals("semiring")
```

```
$x


$type
c("balance", "cluster")

$symclos
[1] TRUE

$transclos
[1] TRUE

$k
[1] 2

$lbs
```

Balanced Structures

Example as in Doreian, et al (2005)

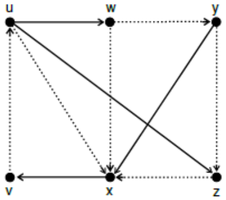


Figure 10.3. An example from Roberts.

Table 10.4. The Value Matrix and Its Closure for Roberts's Example

	u	*v*	*w*	*x*	*y*	*z*		*u*	*v*	*w*	*x*	*y*	*z*
u	0	0	*p*	*n*	0	*p*	*u*	**p**	*n*	**p**	*n*	*n*	**p**
v	*n*	0	0	0	0	0	*v*	*n*	**p**	*n*	**p**	**p**	*n*
w	0	0	0	*n*	*n*	0	*w*	**p**	*n*	**p**	*n*	*n*	**p**
x	0	*p*	0	0	0	0	*x*	*n*	**p**	*n*	**p**	**p**	*n*
y	0	0	0	*p*	0	*n*	*y*	*n*	**p**	*n*	**p**	**p**	*n*
z	0	0	0	*n*	0	0	*z*	**p**	*n*	**p**	*n*	*n*	**p**

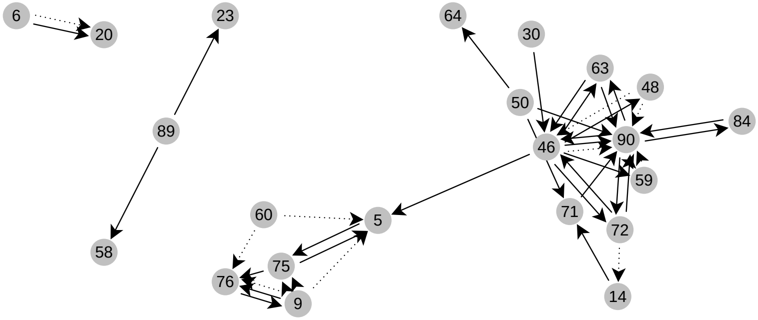
Balance Semiring

```
# CREATE MATRIX DATA TYPE
mat <- matrix(nrow=6, ncol=6)
rownames(mat) <- letters[21:26]
colnames(mat) <- rownames(mat)

# ASSING VALUES
mat[1,] <- c(0,0,1,-1,0,1)
mat[2,] <- c(-1,0,0,0,0,0)
mat[3,] <- c(0,0,0,-1,-1,0)
mat[4,] <- c(0,1,0,0,0,0)
mat[5,] <- c(0,0,0,1,0,-1)
mat[6,] <- c(0,0,0,-1,0,0)


   u  v  w  x  y  z
u  0  0  1 -1  0  1
v -1  0  0  0  0  0
w  0  0  0 -1 -1  0
x  0  1  0  0  0  0
y  0  0  0  1  0 -1
z  0  0  0 -1  0  0
```

```
# BALANCE SEMIRING STRUCTURE
semiring(as.signed(mat), type="balance")


$val
[1] 1  0  -1

$s
   1 2 3  4  5  6
1  0 0 1 -1  0  1
2 -1 0 0  0  0  0
3  0 0 0 -1 -1  0
4  0 1 0  0  0  0
5  0 0 0  1  0 -1
6  0 0 0 -1  0  0

$Q
  1 2 3 4 5 6
1 p n p n n p
2 n p n p p n
3 p n p n n p
4 n p n p p n
5 n p n p p n
6 p n p n n p

$k
[1] 2

attr(,"class")
[1] "Rel.Q"   "balance"
```

Incubator network

```
# COOPERATION AND COMPETITION TIES IN 'netA' WITHOUT ISOLATES
netAck <- rm.isol(netA[ , , c(1,3)])

# PLOT THE MULTIGRAPH BY REUSING THE OUTLINE
multigraph(netAck, scope = scpA, signed = TRUE, layout = "force", seed = 9)
```



Signed Network *C and K in Incubator A*

```
# FUNCTION signed() CREATES A "Signed" CLASS OBJECT FROM 2 MATRICES
netAsg <- signed(netAck)
```

```
$val
[1] p o n a

$s
   5 6 9 14 20 23 30 46 48 50 58 59 60 63 64 71 72 75 76 84 89 90
5  o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  p  o  o  o  o
6  o o o o  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
9  n o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  a  a  o  o  o
14 o o o o  o  o  o  o  o  o  o  o  o  o  o  p  o  o  o  o  o  o
20 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
23 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
30 o o o o  o  o  o  p  o  o  o  o  o  o  o  o  o  o  o  o  o  o
46 p o o o  o  o  o  o  p  o  o  p  o  p  o  o  p  o  o  o  o  a
48 o o o o  o  o  o  n  o  o  o  o  o  o  o  o  o  o  o  o  o  n
50 o o o o  o  o  o  o  o  o  o  o  o  o  p  p  o  o  o  o  o  p
58 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
59 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  p
60 n o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  n  o  o  o
63 o o o o  o  o  o  p  o  o  o  o  o  o  o  o  o  o  o  o  o  p
64 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
71 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  p
72 o o o n  o  o  o  p  o  o  o  o  o  o  o  o  o  o  o  o  o  p
75 p o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  p  o  o  o
76 o o p o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
84 o o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  p
89 o o o o  o  p  o  o  o  o  p  o  o  o  o  o  o  o  o  o  o  o
90 o o o o  o  o  o  p  o  o  o  p  o  p  o  o  p  o  o  p  o  o
```

Semiring structures

```r
# BALANCE SEMIRING 2—PATHS (DEAFULT)
semiring(netAsg, type = "balance")

# 3—PATHS
semiring(netAsg, type = "balance", k = 3)

# 2—SEMIPATHS
semiring(netAsg, type = "balance", symclos = FALSE)
# ...
```

```r
# CLUSTER SEMIRING 2—PATHS (DEAFULT)
semiring(netAsg, type = "cluster")

# 3—PATHS
semiring(netAsg, type = "cluster", k = 3)

# 2—SEMIPATHS
semiring(netAsg, type = "cluster", symclos = FALSE)
# ...
```

Checking for Balance

```
identical(
+    semiring(netAsg, type = "balance", k = 3)$Q,
+    semiring(netAsg, type = "balance", k = 2)$Q )
```

```
[1] FALSE
```

```
identical(
+    semiring(netAsg, type = "balance", k = 3)$Q,
+    semiring(netAsg, type = "balance", k = 4)$Q )
```

```
[1] FALSE
```

```
identical(
+    semiring(netAsg, type = "balance", k = 4)$Q,
+    semiring(netAsg, type = "balance", k = 5)$Q )
```

```
[1] TRUE
```

Checking for Balance (Cluster)

```
identical(
+   semiring(netAsg, type = "cluster", k = 3)$Q,
+   semiring(netAsg, type = "cluster", k = 2)$Q )

  [1] FALSE
```

```
identical(
+   semiring(netAsg, type = "cluster", k = 3)$Q,
+   semiring(netAsg, type = "cluster", k = 4)$Q )

  [1] FALSE
```

```
identical(
+   semiring(netAsg, type = "cluster", k = 4)$Q,
+   semiring(netAsg, type = "cluster", k = 5)$Q )

  [1] TRUE
```

Weak Balance Structure

```
# BALANCE WITH SEMIPATHS
netAQb <- semiring(netAsg, type = "balance", k = 4)

perm(netAQb$Q, clu = c(1,6,1,2,6,6,3,2,2,3,6,2,4,2,5,2,2,1,1,2,6,2))

    5 9 75 76 14 46 48 59 63 71 72 84 90 30 50 60 64 6 20 23 58 89
5   a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
9   a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
75  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
76  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
14  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
46  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
48  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
59  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
63  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
71  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
72  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
84  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
90  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
30  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
50  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
60  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
64  a a a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  o o  o  o  o
6   o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  a o  o  o  o
20  o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o a  o  o  o
23  o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  p  p  o
58  o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  p  p  o
89  o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  p
```

Weak Balance Structure

```
# BALANCE WITH PATHS
netAQbp <- semiring(netAsg, type = "balance", symclos = FALSE, k = 4)

perm(netAQbp$Q, clu = c(1,6,1,2,6,6,3,2,2,3,6,2,4,2,5,2,2,1,1,2,6,2))
```

```
   5 9 75 76 14 46 48 59 63 71 72 84 90 30 50 60 64 6 20 23 58 89
5  a a a  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
9  a a a  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
75 a a a  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
76 a a a  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
14 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
46 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
48 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
59 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
63 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
71 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
72 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
84 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
90 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
30 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
50 a a a  a  a  a  a  a  a  a  a  a  a  o  o  o  o  o o  o  o  o
60 a a a  a  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
64 o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
6  o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
20 o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
23 o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
58 o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
89 o o o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o o  o  o  o
```

Weak Balance Structure

Main component of Incubator A

```
# FUNCTION comps() FINDS COMPONENTS AND ISOLATES
comps(netAck)
```

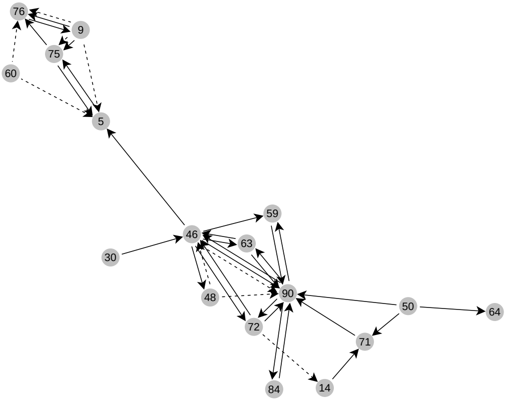
```
$com
$com[[1]]
 [1] "5"  "50" "59" "60" "63" "64" "71" "72" "75" "76" "84" "90" "9"  "14" "30" "46" "48"

$com[[2]]
[1] "58" "89" "23"

$com[[3]]
[1] "6"  "20"


$isol
character(0)
```

```
# RECORD TIES FROM MAIN COMPONENT OF netAck
netAc1 <- rel.sys(incubA$net, "toarray", sel = comps(netAck)$com[[1]])
```

Weak Balance Structure

Main component of Incubator A

```
# PLOT NETWORK RELATIONS 'C' AND 'K' IN THE COMPONENT
multigraph(netAc1[,,c(1,3)], scope = scpA, layout = "force", seed = 6)
```



Weak Balance Structure

Outline

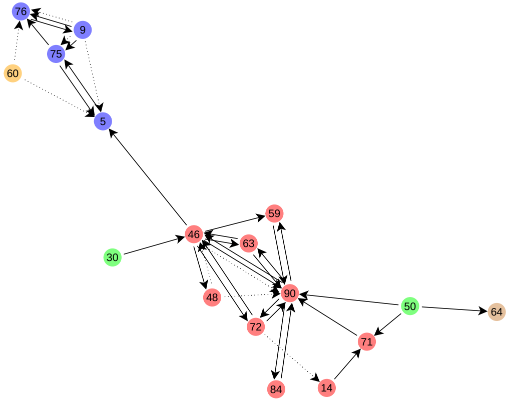
```
# MAKE OUTLINE WITH INFO FROM WEAK BALANCE STRUCTURE OF PATHS
scpAck <- list(lty = c(1,3), clu = c(1,1,2,3,2,2,3,2,4,2,5,2,2,1,1,2,2),
+   vcol = c("blue","red","green","orange","peru"), alpha = .5)
```

```
c(scpAck,scpA)

  $lty
  [1] 1 3

  $clu
   [1] 1 1 2 3 2 2 3 2 4 2 5 2 2 1 1 2 2

  $vcol
  [1] "blue"   "red"    "green"  "orange" "peru"

  $alpha
  [1] 0.5

  $ecol
  [1] 1

  $vcol
  [1] "#C0C0C0"

  ...
```

Weak Balance Structure

Main component of Incubator A

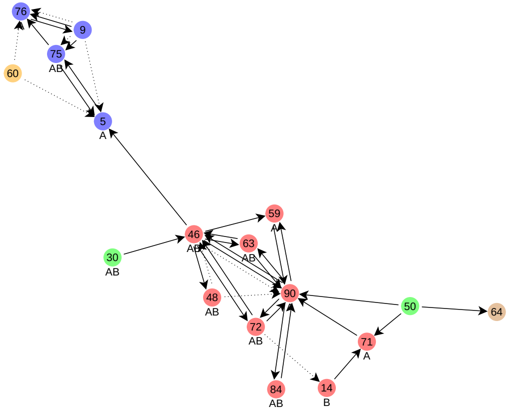
```
# ARGUMENT NAMES CAN BE OMITTED OUTSIDE scope
multigraph(netAc1[,,c(1,3)], scope = c(scpA, scpAck), "force", seed = 6)
```



Weak Balance Structure

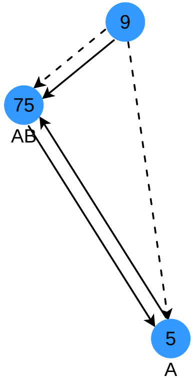
for social influence through comparison

```
multigraph(netAc1[,,c(1,3)], att = netAc1[,,4:5], layout = "force", seed = 6,
+    scope = c(scpA, scpAck))
```



Balance semiring *(Signed triad)*

```
# 2—Paths  (9, 75)
"n, p" "o, a" "a, o"

# multiplication
"n" "o" "o"

# addition
n
```



	5	9	75
5	o	o	p
9	n	o	a
75	p	o	o

t^{α}

	5	9	75
5	p	o	o
9	a	o	n
75	o	o	p

t^{α} paths, $k > 1$

	5	9	75
5	p	a	a
9	a	a	n
75	a	n	a

t^{α} semipaths, $k = 2$

	5	9	75
5	a	a	a
9	a	a	a
75	a	a	a

t^{α} semipaths, $k > 2$

$5.$ Affiliation networks

(two-mode data)

Affiliation networks

- Ties between two sets of entities represent two-mode, bipartite, or **affiliations networks**

  ➟ like the duality between *"people and groups"*, *"person and events"*, *"actors and their attributes"*

- In a 2-mode matrix data the domain and the codomain are not equal

  ➟ serves to represent affiliations networks

- An algebraic approach to affiliation networks is found in **Formal Concept Analysis**

Formal Concept Analysis

(Ganter & Wille, 1996)

- Formal Concept Analysis is an analytical framework for the study of affiliation networks

- Elements in the domain and codomain are called *Objects* and *Attributes* resp.

- A set of Objects G and a set of Attributes M are associated with an incident relation $I \subseteq G \times M$ in a **formal context**

- The **formal concept** of a formal context is a pair of sets of maximally contained objects A and attributes B

  ➟ (i.e. maximal rectangles in the formal context)

  A and B are said to be the *extent* and *intent* of the formal concept

Example: G20 Countries Affiliation network

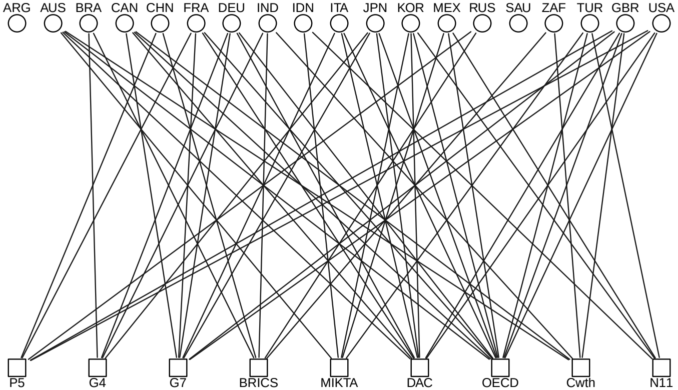
```r
# LOAD AFFILIATION DATA G20 COUNTRIES
load("data/G20.rda")
```

```r
# OBJECT G20 IS A DATA FRAME THAT REPRESENTS A FORMAL CONTEXT
G20
```

```
    P5 G4 G7 BRICS MIKTA DAC OECD Cwth N11
ARG  0  0  0     0     0   0    0    0   0
AUS  0  0  0     0     1   1    1    1   0
BRA  0  1  0     1     0   0    0    0   0
CAN  0  0  1     0     0   1    1    1   0
CHN  1  0  0     1     0   0    0    0   0
FRA  1  0  1     0     0   1    1    0   0
DEU  0  1  1     0     0   1    1    0   0
IND  0  1  0     1     0   0    0    1   0
IDN  0  0  0     0     1   0    0    0   1
ITA  0  0  1     0     0   1    1    0   0
JPN  0  1  1     0     0   1    1    0   0
KOR  0  0  0     0     1   1    1    0   1
MEX  0  0  0     0     1   0    1    0   1
RUS  1  0  0     1     0   0    0    0   0
SAU  0  0  0     0     0   0    0    0   0
ZAF  0  0  0     1     0   0    0    1   0
TUR  0  0  0     0     1   0    1    0   1
GBR  1  0  1     0     0   1    1    1   0
USA  1  0  1     0     0   1    1    0   0
```

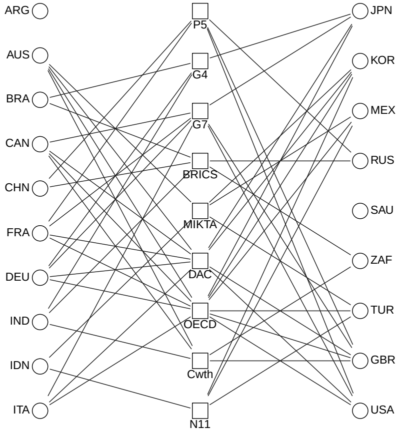
G20 Countries (affiliation network)

```
# BIPARTITE GRAPH OF 'G20'
bmgraph(G20, rot = 90, mirrorX = TRUE)
```



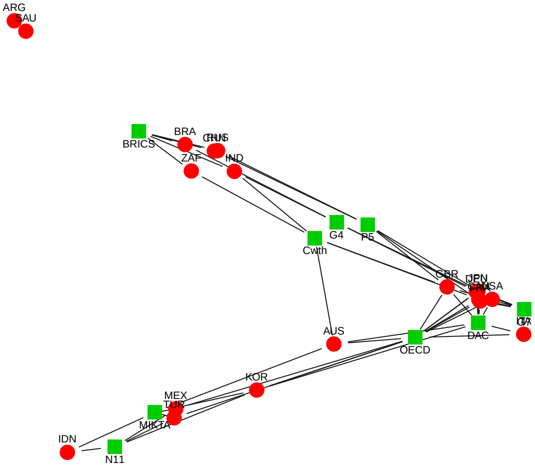
G20 Countries (affiliation network)

```
# BIPARTITE GRAPH WITH THREE COLUMNS
bmgraph(G20, layout = "bip3", cex = 3, tcex = 1)
```



G20 Countries (affiliation network)

```r
# APPLY CORRESPONDENCE ANALYSIS TO THE PLOT
bmgraph(G20, layout = "CA", rot = 99, vcol = 2:3, pch = c(19, 15), jitter = .1)
```



Galois Derivations

- A **Galois derivation** between G and M is defined for any subsets $A \subseteq G$ and $B \subseteq M$ by

$$\begin{aligned} A' &= m \in M \mid (g, m) \in I \quad \text{(for all } g \in A) \\ B' &= g \in G \mid (g, m) \in I \quad \text{(for all } m \in B) \end{aligned}$$

– A' is the set of attributes common to all the objects in the intent

– B' the set of objects possessing the attributes in the extent

```
formals("galois")

  $x


  $labeling
  c("full", "reduced")
```

Galois derivations in G20

```
galois(G20)

  $P5
  [1] "CHN, FRA, GBR, RUS, USA"

  $G4
  [1] "BRA, DEU, IND, JPN"

  $`DAC, G7, OECD`
  [1] "CAN, DEU, FRA, GBR, ITA, JPN, USA"

  $BRICS
  [1] "BRA, CHN, IND, RUS, ZAF"

  $MIKTA
  [1] "AUS, IDN, KOR, MEX, TUR"

  $`DAC, OECD`
  [1] "AUS, CAN, DEU, FRA, GBR, ITA, JPN, KOR, USA"

  $OECD
  [1] "AUS, CAN, DEU, FRA, GBR, ITA, JPN, KOR, MEX, TUR, USA"

  $Cwth
  [1] "AUS, CAN, GBR, IND, ZAF"

  $`MIKTA, N11`
  [1] "IDN, KOR, MEX, TUR"

  $`BRICS, Cwth, DAC, G4, G7, MIKTA, N11, OECD, P5`
  character(0)

  ...
```

Galois derivations in G20 – Reduced labeling

```r
g20gc <- galois(G20, labeling = "reduced")
```

```
$reduc
$reduc$P5
character(0)

$reduc$G4
character(0)

$reduc$G7
[1] "ITA"

$reduc$BRICS
character(0)

$reduc$MIKTA
character(0)

$reduc$DAC
character(0)

$reduc$OECD
character(0)

$reduc$Cwth
character(0)
```

```
$reduc$N11
[1] "IDN"

$reduc[[10]]
character(0)

$reduc[[11]]
[1] "FRA, USA"

$reduc[[12]]
[1] "CHN, RUS"

$reduc[[13]]
[1] "GBR"

$reduc[[14]]
[1] "DEU, JPN"

$reduc[[15]]
[1] "BRA"

$reduc[[16]]
[1] "IND"

$reduc[[17]]
[1] "CAN"
```

```
$reduc[[18]]
[1] "ZAF"

$reduc[[19]]
[1] ""

$reduc[[20]]
character(0)

$reduc[[21]]
[1] "AUS"

$reduc[[22]]
character(0)

$reduc[[23]]
[1] "KOR"

$reduc[[24]]
[1] "MEX, TUR"

$reduc[[25]]
[1] "ARG, SAU"
```

Partial ordering of the Concepts

A *hierarchy* of concepts is given by the sub–superconcept relation

$$(A, B) \leq (A_2, B_2) \quad \Leftrightarrow \quad A_1 \subseteq A_2 \qquad (\Leftrightarrow \quad B_1 \subseteq B_2)$$

Concept Lattice of the Context

- built from the hierarchy structure of concepts
- The greatest lower bound of the meet and the least upper bound of the join are defined for an index set T as

$$\bigwedge_{t \in T} \; (A_t, B_t) \;=\; \Big(\bigcap_{t \in T} A_t, \;\; (\bigcup_{t \in T} B_t)'' \Big)$$
$$\bigvee_{t \in T} \; (A_t, B_t) \;=\; \Big((\bigcup_{t \in T} A_t)'', \;\; \bigcap_{t \in T} B_t \Big)$$

Partial order of concepts

```
# FUNCTION partial.order() CONSTRUCTS HIERARCHY OF CONCEPTS
g20gcpo <- partial.order(g20gc, type = "galois")

                  {P5} {} {G4} {} {G7} {ITA} {BRICS} {} {MIKTA} {} {DAC} {} {OECD} {} {Cwth} {}
  {P5} {}               1       0           0          0           0        0         0         0
  {G4} {}               0       1           0          0           0        0         0         0
  {G7} {ITA}            0       0           1          0           0        1         1         0
  {BRICS} {}            0       0           0          1           0        0         0         0
  {MIKTA} {}            0       0           0          0           1        0         0         0
  {DAC} {}              0       0           0          0           0        1         1         0
  {OECD} {}             0       0           0          0           0        0         1         0
  {Cwth} {}             0       0           0          0           0        0         0         1
  {N11} {IDN}           0       0           0          0           1        0         0         0
  10                    1       1           1          1           1        1         1         1
  {} {FRA, USA}         1       0           1          0           0        1         1         0
  {} {CHN, RUS}         1       0           0          1           0        0         0         0
  {} {GBR}              1       0           1          0           0        1         1         1
  {} {DEU, JPN}         0       1           1          0           0        1         1         0
  {} {BRA}              0       1           0          1           0        0         0         0
  {} {IND}              0       1           0          1           0        0         0         1
  {} {CAN}              0       0           1          0           0        1         1         1
  {} {ZAF}              0       0           0          1           0        0         0         1
  19                    0       0           0          0           1        1         1         0
  20                    0       0           0          0           1        0         1         0
  {} {AUS}              0       0           0          0           1        1         1         1
  22                    0       0           0          0           0        1         1         1
  {} {KOR}              0       0           0          0           1        1         1         0
  {} {MEX, TUR}         0       0           0          0           1        0         1         0
  {} {ARG, SAU}         0       0           0          0           0        0         0         0
```

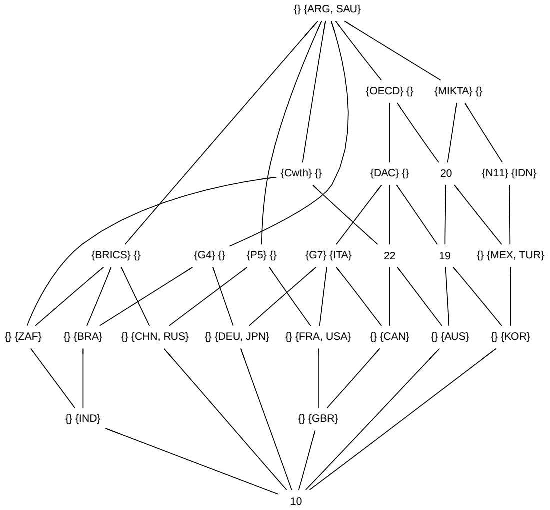
(Extract)

Galois derivations and partial ordering

```
# STRUCTURE OF g20gc OBJECT CREATED WITH A REDUCED LABELING
str(g20gc)

 List of 2
  $ full :List of 25
   ..$ P5                                         : chr "CHN, FRA, GBR, RUS, USA"
   ..$ G4                                         : chr "BRA, DEU, IND, JPN"
   ..$ DAC, G7, OECD                              : chr "CAN, DEU, FRA, GBR, ITA, JPN, USA"
   ..$ BRICS                                      : chr "BRA, CHN, IND, RUS, ZAF"
   ..$ MIKTA                                      : chr "AUS, IDN, KOR, MEX, TUR"
   ..$ DAC, OECD                                  : chr "AUS, CAN, DEU, FRA, GBR, ITA, JPN,
   ..$ OECD                                       : chr "AUS, CAN, DEU, FRA, GBR, ITA, JPN,
   ..$ Cwth                                       : chr "AUS, CAN, GBR, IND, ZAF"
   ..$ MIKTA, N11                                 : chr "IDN, KOR, MEX, TUR"
   ..$ BRICS, Cwth, DAC, G4, G7, MIKTA, N11, OECD, P5: chr(0)
 ...

 ..- attr(*, "class")= chr [1:2] "Galois" "full"
  $ reduc:List of 25
   ..$ P5   : chr(0)
   ..$ G4   : chr(0)
   ..$ G7   : chr "ITA"
   ..$ BRICS: chr(0)
   ..$ MIKTA: chr(0)
   ..$ DAC  : chr(0)
   ..$ OECD : chr(0)
   ..$ Cwth : chr(0)
   ..$ N11  : chr "IDN"
   ..$      : chr(0)
 ...
```

Concept lattice of the context

```
# PLOT HIERARCHY OF CONCEPTS AS LATTICE DIAGRAM
diagram(g20gcpo)
```



Filters and Ideals

formal definition

- Let $(P, \leq)$ be an ordered set, and a, b are elements in P
- A non-empty subset U [resp. D] of P is an upset [resp. downset] called a **filter** [resp. **ideal**] if, for all $a \in P$ and $b \in U$ [resp. D]

$$b \leq a \quad \text{implies} \quad a \in U \qquad\qquad [\text{ resp. } a \leq b \quad \text{implies} \quad a \in D\ ]$$

- The upset $\uparrow x$ formed for all the upper bounds of $x \in P$ is called a **principal filter** generated by x
- Dually, $\downarrow x$ is a **principal ideal** with all the lower bounds of $x \in P$

☛ filters and ideals not coinciding with P are called *proper*

Filters and Ideals

```
# fltr() FINDS PRINCIPAL FILTERS IN THE PARTIAL ORDER OF THE CONTEXT
formals("fltr")

  $x


  $PO


  $rclos
  [1] TRUE

  $ideal
  [1] FALSE
```

Principal Filters

```
# PRINCIPAL FILTER OF THE FIRST CONCEPT IN g20gcpo
fltr(1, g20gcpo)

  $`1`
  [1] "{P5} {}"

  $`25`
  [1] "{} {ARG, SAU}"
```

```
# ANOTHER OPTION IS TO USE INTENT LABELS OF DIFFERENT CONCEPTS
fltr(c("P5", "BRICS"), g20gcpo)

  $`1`
  [1] "{P5} {}"

  $`4`
  [1] "{BRICS} {}"

  $`25`
  [1] "{} {ARG, SAU}"
```

Principal Ideals

```
# PRINCIPAL IDEAL OF THE FIRST CONCEPT IN g20gcpo
fltr("P5", g20gcpo, ideal = TRUE)

  $`1`
  [1] "{P5} {}"

  $`10`
  [1] "10"

  $`11`
  [1] "{} {FRA, USA}"

  $`12`
  [1] "{} {CHN, RUS}"

  $`13`
  [1] "{} {GBR}"
```

☛ Beware that *ideals* in groups, semigroups, and semirings have a different meaning

Thank you!

http://CRAN.R-Project.org/package=multiplex

CRAN in views: Psychometrics, SocialSciences

http://CRAN.R-Project.org/package=multigraph