

Formatted manual of Markov.lib

1 Singular libraries

1.1 Markov_lib

-----BEGIN OF PART WHICH IS INCLUDED IN MANUAL-----

Library: Markov.lib

Purpose: Markov Relations for Bayesian Networks

Procedures:

1.1.0.1 info

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov_lib\]](#), page 1).

Usage: `info(I);` I ideal

Return: list of integers `a[1]`, `a[2]` and `a[3]` with:
 - `a[1]` the codimension of I
 - `a[2]` the degree of I
 - `a[3]` the number of minimal generators of I

Example:

```
LIB "Markov.lib";
intvec d = 2,2,2,2;
int n = size(d);
def pdR = probring(d);
setring pdR;
intvec v = 1,1,0,0,1,1;
intmat m = bnet(n,v);
list l = localMarkov(m);
ideal I = MarkovIdeal(l,d);
info(I);
↳ // ** I is no standard basis
↳ // ** I is no standard basis
↳ // dimension (proj.) = 10
↳ // degree (proj.) = 24
↳ // ** right side is not a datum, assignment ignored
↳ [1]:
↳ 5
↳ [2]:
↳ 0
↳ [3]:
↳ 6
```

1.1.0.2 bnet

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov_lib\]](#), page 1).

Usage: `bnet(n,u);` n int, u intvec

Return: an $n \times n$ matrix whose lower triangle is given by u
`m[i,j]` implies the existence of an edge (i,j)

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
m;
↳ 0,0,0,0,
↳ 1,0,0,0,
↳ 1,0,0,0,
↳ 0,1,1,0

```

1.1.0.3 nondec

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `nondec(v,m);` n int, m intmat

Return: list: the nondescendents of the vertex v

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
nondec(1,m);
↳ [1]:
↳ 2
↳ [2]:
↳ 3
↳ [3]:
↳ 4

```

1.1.0.4 pairMarkov

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `pairMarkov(m);` m intmat

Return: l list: the pairwise Markov property

$l[i]$ corresponds to the i th conditional independence statement, $l[i][1]$ is independent of $l[i][2]$ given $l[i][3]$

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
pairMarkov(m);
↳ [1]:
↳ [1]:
↳ 1
↳ [2]:
↳ [1]:
↳ 4
↳ [3]:
↳ [1]:
↳ 2
↳ [2]:
↳ 3
↳ [2]:
↳ [1]:
↳ [1]:

```

```

↳      2
↳ [2]:
↳ [1]:
↳      3
↳ [3]:
↳ [1]:
↳      4

```

1.1.0.5 parent

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `parent(v,m)`; `n int`, `m intmat`

Return: list: the parents of the vertex `v`

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
parent(1,m);
↳ [1]:
↳ 2
↳ [2]:
↳ 3

```

1.1.0.6 nondecminusparents

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `nondec(v,m)`; `n int`, `m intmat`

Return: list: the nondescendants(excluding the parents) of the vertex `v`

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
nondecminusparents(1,m);
↳ [1]:
↳ 4

```

1.1.0.7 localMarkov

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `localMarkov(m)`; `m intmat`

Return: `l list`: the local Markov property
`l[i]` corresponds to the `i`th conditional independence statement, `l[i][1]` is independent of `l[i][2]` given `l[i][3]`

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
localMarkov(m);
↳ [1]:
↳ [1]:
↳ [1]:

```

```

↳      1
↳ [2]:
↳ [1]:
↳      4
↳ [3]:
↳ [1]:
↳      2
↳ [2]:
↳      3
↳ [2]:
↳ [1]:
↳ [1]:
↳      2
↳ [2]:
↳ [1]:
↳      3
↳ [3]:
↳ [1]:
↳      4

```

1.1.0.8 subset

Procedure from library `Markov.lib` (see Section 1.1 [[Markov.lib](#)], page 1).

Usage: `subset(k,X)`; `k` int, `X` list

Return: list: a subset of `X`

If $b_n \cdots b_1$ is the binary representation of the integer `k`, then `subset(k,X)` returns the set $\{X[i] \mid b_i = 1\}$

Example:

```

LIB "Markov.lib";
list l = 1,2,3;
for(int i=1;i<=7;i++)
{
subset(i,l);
}
↳ [1]:
↳ 1
↳ [1]:
↳ 2
↳ [1]:
↳ 1
↳ [2]:
↳ 2
↳ [1]:
↳ 3
↳ [1]:
↳ 1
↳ [2]:
↳ 3
↳ [1]:
↳ 2
↳ [2]:
↳ 3
↳ [1]:
↳ 1
↳ [2]:
↳ 2

```

```

↳ [3]:
↳ 3

```

1.1.0.9 children

Procedure from library `Markov.lib` (see Section 1.1 [Markov.lib], page 1).

Usage: `children(v,m)`; `n` int, `m` intmat

Return: list: the children of the vertex `v`

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
children(4,m);
↳ [1]:
↳ 2
↳ [2]:
↳ 3

```

1.1.0.10 Bayes_ball

Procedure from library `Markov.lib` (see Section 1.1 [Markov.lib], page 1).

Usage: `Bayes_ball(A,C,m)`; `A` list, `C` list, `m` intmat

Return: list: a maximal set of vertices `B` such that `A` and `B` are d-separated by `C`

Example:

```

LIB "Markov.lib";
intvec v = 1,0,1,0,1,0;
intmat m = bnet(4,v);
list A = 1;
list C = 2;
Bayes_ball(A,C,m);
↳ [1]:
↳ 3
↳ [2]:
↳ 4

```

1.1.0.11 globalMarkov

Procedure from library `Markov.lib` (see Section 1.1 [Markov.lib], page 1).

Usage: `globalMarkov(m)`; `m` intmat

Return: list: the global Markov property

`l[i]` corresponds to the *i*th conditional independence statement, `l[i][1]` is independent of `l[i][2]` given `l[i][3]`

Example:

```

LIB "Markov.lib";
intvec v = 1,1,0,0,1,1;
intmat m = bnet(4,v);
globalMarkov(m);
↳ [1]:
↳ [1]:
↳ [1]:
↳ 1

```

```

↳ [2]:
↳ [1]:
↳ 4
↳ [3]:
↳ [1]:
↳ 2
↳ [2]:
↳ 3
↳ [2]:
↳ [1]:
↳ [1]:
↳ 2
↳ [2]:
↳ [1]:
↳ 3
↳ [3]:
↳ [1]:
↳ 4

```

1.1.0.12 equivStatements

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `equivStatements(s,t)`; s list, t list

Return: 1 if $s[1]=t[2]$, $s[2]=t[1]$ and $s[3]=t[3]$
0 otherwise

Example:

```

LIB "Markov.lib";
list s = 1,list(2,3),4;
list t = list(2,3),1,4;
equivStatements(s,t);
↳ 1

```

1.1.0.13 next

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `next(u,j,d)`; u intvec, j int, d intvec

Return: intvec: the index of the next variable in the ring

Example:

```

LIB "Markov.lib";
intvec d = 2,2,2;
def pdR = probring(d);
setring pdR;
intvec idx;
for (int i=1; i<=size(d); i++)
{
  idx[i] = 1;
}
idx = next(idx,size(d),d);
idx;
↳ 1,1,2
idx = next(idx,size(d),d);
idx;
↳ 1,2,1

```

1.1.0.14 sdec

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `sdec(id)`; `id` intvec

Return: string: $id[1]*10^{(n-1)}+id[2]*10^{(n-2)}+\dots+id[n]$

Example:

```
LIB "Markov.lib";
intvec id = 1,4,10;
sdec(id);
↳ 150
```

1.1.0.15 probring

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `probring(d[,f,v,o])`; `d` intvec, `f` string, `v` string, `o` string

Return: ring: ring R with coefficient field f , ring variables $v_1\dots 1,\dots,v_d[1]\dots d[n]$ and term ordering o
The default values for f , v and o are "0", "p" and "dp" respectively

Example:

```
LIB "Markov.lib";
intvec d = 2,2,3;
probring(d);
↳ // characteristic : 0
↳ // number of vars : 12
↳ // block 1 : ordering dp
↳ // : names p111 p112 p113 p121 p122 p123 p211 p212 p2\
13 p221 p222 p223
↳ // block 2 : ordering C
```

1.1.0.16 index

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `index(linput,d)`; `linput` list, `d` intvec

Return: int: the index of the corresponding indeterminate

Example:

```
LIB "Markov.lib";
intvec d = 2,2,2;
def pdR = probring(d);
setring pdR;
list l = 1,2,1;
index(l,d);
↳ 3
var(index(l,d));
↳ p121
```

1.1.0.17 cartesian

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `cartesian(linput)`; `linput` list

Return: list: the cartesian product of a list of lists

Example:

```

LIB "Markov.lib";
list l = list(1,2),list(1,2),list(1);
cartesian(l);
↳ [1]:
↳ [1]:
↳ 1
↳ [2]:
↳ 1
↳ [3]:
↳ 1
↳ [2]:
↳ [1]:
↳ 1
↳ [2]:
↳ 2
↳ [3]:
↳ 1
↳ [3]:
↳ [1]:
↳ 2
↳ [2]:
↳ 1
↳ [3]:
↳ 1
↳ [4]:
↳ [1]:
↳ 2
↳ [2]:
↳ 2
↳ [3]:
↳ 1

```

1.1.0.18 Pairs

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\], page 1](#)).

Usage: Pairs(L); L list

Return: list: the set of all pairs of L

Example:

```

LIB "Markov.lib";
Pairs(list(1,2,3));
↳ [1]:
↳ [1]:
↳ 1
↳ [2]:
↳ 2
↳ [2]:
↳ [1]:
↳ 1
↳ [2]:
↳ 3
↳ [3]:
↳ [1]:
↳ 2
↳ [2]:
↳ 3

```

1.1.0.19 levels

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `levels(di)`; `di` int

Return: list: the levels of the random variable X_i

Example:

```
LIB "Markov.lib";
levels(3);
↳ [1]:
↳ 1
↳ [2]:
↳ 2
↳ [3]:
↳ 3
```

1.1.0.20 Prob

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `Prob(linut,d)`; `linut` list, `d` intvec

Return: `poly`: the marginalization over the subset of the random variables specified "IND"

Example:

```
LIB "Markov.lib";
intvec d = 2,2,2;
def pdR = probring(d);
setring pdR;
list l = 1,"IND","IND";
Prob(l,d);
↳ p111+p112+p121+p122
```

1.1.0.21 Quad

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `Quad (A,a,B,b,C,c,d)`; `A` list, `a` list, `B` list, `b` list, `C` list, `c` list, `d` intvec

Return: `poly`: the quadric associated to the probability $P(A=a[1],B=b[1],C=c) * P(A=a[2],B=b[2],C=c) - P(A=a[2],B=b[1],C=c) * P(A=a[1],B=b[2],C=c)$

Example:

```
LIB "Markov.lib";
/* Computes the probability P(X1=1,X2=1,X3=1)*P(X1=2,X2=2,X3=1)
-P(X1=2,X2=1,X3=1)*P(X1=1,X2=2,X3=1) */
intvec d = 2,2,2;
def pdR = probring(d);
setring pdR;
list A,B,C;
list a,b,c;
A = list(1);
B = list(2);
C = list(3);
a[1] = levels(d[1]);
b[1] = levels(d[2]);
c[1] = levels(d[3]);
```

```

a = Pairs(cartesian(a));
b = Pairs(cartesian(b));
c = cartesian(c);
Quad(A,a[1],B,b[1],C,c[1],d);
↳ -p121*p211+p111*p221

```

1.1.0.22 StatementQuadrics

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `StatementQuadrics(A,B,C,d)`; A list, B list, C list, d intvec

Return: poly: the list of all quadrics associated to the conditional independence statement, A is independent of B given C

Example:

```

LIB "Markov.lib";
/* Lists all quadrics associated to the conditional independence statement, X1 is inde-
pendent of X2 given X3 */
intvec d = 2,2,2;
def pdR = probring(d);
setring pdR;
StatementQuadrics(list(1),list(2),list(3),d);
↳ [1]:
↳ -p121*p211+p111*p221
↳ [2]:
↳ -p122*p212+p112*p222

```

1.1.0.23 MarkovIdeal

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: `MarkovIdeal(L,d)`; L list, d intvec

Return: ideal: the ideal of the independence model given by L

Example:

```

LIB "Markov.lib";
intvec d = 2,2,2,2; int n = size(d);
def pdR = probring(d);
setring pdR;
intvec v15 = 1,1,0,0,1,1;
intmat m15 = bnet(n,v15);
list l15 = localMarkov(m15);
list pw15 = pairMarkov(m15);
list g15 = globalMarkov(m15);
ideal I15 = MarkovIdeal(l15,d);
info(I15);
↳ // ** I is no standard basis
↳ // ** I is no standard basis
↳ // dimension (proj.) = 10
↳ // degree (proj.) = 24
↳ // ** right side is not a datum, assignment ignored
↳ [1]:
↳ 5
↳ [2]:
↳ 0
↳ [3]:
↳ 6
ideal G15 = MarkovIdeal(g15,d);

```

```

info(G15);
↳ // ** I is no standard basis
↳ // ** I is no standard basis
↳ // dimension (proj.) = 10
↳ // degree (proj.) = 24
↳ // ** right side is not a datum, assignment ignored
↳ [1]:
↳ 5
↳ [2]:
↳ 0
↳ [3]:
↳ 6
quotient(I15,G15);
↳ _[1]=1
ideal T15 = torideal(I15,d);
quotient(I15,T15);
↳ _[1]=p1222*p2221-p1221*p2222
↳ _[2]=p1212*p2211-p1211*p2212
↳ _[3]=p1122*p2121-p1121*p2122
↳ _[4]=p1112*p2111-p1111*p2112
↳ _[5]=p1112*p1222+p1222*p2112+p1112*p2222+p2112*p2222
↳ _[6]=p1111*p1222+p1222*p2111+p1111*p2222+p2111*p2222
↳ _[7]=p1112*p1221+p1221*p2112+p1112*p2221+p2112*p2221
↳ _[8]=p1111*p1221+p1221*p2111+p1111*p2221+p2111*p2221
↳ _[9]=p1122*p1212+p1212*p2122+p1122*p2212+p2122*p2212
↳ _[10]=p1121*p1212+p1212*p2121+p1121*p2212+p2121*p2212
↳ _[11]=p1122*p1211+p1211*p2122+p1122*p2211+p2122*p2211
↳ _[12]=p1121*p1211+p1211*p2121+p1121*p2211+p2121*p2211
ideal Q15 = sat(I15,T15)[1];
list pd15 = primdecGTZ(Q15);
info(T15)[1];
↳ // dimension (proj.) = 9
↳ // degree (proj.) = 48
↳ // ** right side is not a datum, assignment ignored
↳ 6
for (int i=1; i<=size(pd15); i++)
{
info(std(pd15[i][1]))[1];
}
↳ // dimension (proj.) = 9
↳ // degree (proj.) = 4
↳ // ** right side is not a datum, assignment ignored
↳ 6
↳ // dimension (proj.) = 9
↳ // degree (proj.) = 4
↳ // ** right side is not a datum, assignment ignored
↳ 6
↳ // dimension (proj.) = 9
↳ // degree (proj.) = 4
↳ // ** right side is not a datum, assignment ignored
↳ 6
↳ // dimension (proj.) = 9
↳ // degree (proj.) = 4
↳ // ** right side is not a datum, assignment ignored
↳ 6

```

1.1.0.24 torideal

Procedure from library `Markov.lib` (see [Section 1.1 \[Markov.lib\]](#), page 1).

Usage: torideal(I,d); I ideal, d intvec

Return: ideal: $I:p^\infty$ where p is the product of all the linear forms
For example, if $d=2,2,2$, then $p=p_{111} \dots p_{222} p_{11} \dots p_{22} p_{111} \dots p_{222}$

Example:

```
LIB "Markov.lib";
intvec d = 2,2,2,2; int n = size(d);
def pdR = probring(d);
setring pdR;
intvec v = 1,1,0,0,1,1;
intmat m = bnet(n,v);
list l = localMarkov(m);
ideal I = MarkovIdeal(l,d);
ideal T = torideal(I,d);
```

1.1.0.25 map_observable

Procedure from library Markov.lib (see Section 1.1 [Markov.lib], page 1).

Usage: map_observable(H,d,r); H ring, d intvec, r int

Return: map: the ring map from H to basering induced by the inclusion of H in basering
It is assumed that H is the ring generated by the indeterminates that represent the observable probability distribution $P(X_1, \dots, X_r)$ while basering is generated by the indeterminates for the probability distribution $P(X_1, \dots, X_n)$ where $r < n$. Each variable in H is mapped to the marginalization over the hidden variables in basering.

Example:

```
LIB "Markov.lib";
/* Computes the polynomial constraints for the Bayesian network X1 <- X3 -> X2
where X1 and X2 are observable and X3 is hidden. */
intvec d = 3,3,2;
int n = size(d);
int r = 2;
def pdR = probring(d);
intvec d2 = d[1..r];
def H = probring(d2);
setring pdR;
def Phi = map_observable(H, d, r);
Phi;
↳ Phi[1]=p111+p112
↳ Phi[2]=p121+p122
↳ Phi[3]=p131+p132
↳ Phi[4]=p211+p212
↳ Phi[5]=p221+p222
↳ Phi[6]=p231+p232
↳ Phi[7]=p311+p312
↳ Phi[8]=p321+p322
↳ Phi[9]=p331+p332
intvec v = 0,1,1;
intmat m = bnet(n,v);
list g = globalMarkov(m);
ideal G = MarkovIdeal(g,d);
ideal T = torideal(G,d);
setring H;
ideal Q = preimage(pdR,Phi,T);
```

```
Q;  
↪ Q[1]=p13*p22*p31-p12*p23*p31-p13*p21*p32+p11*p23*p32+p12*p21*p33-p11*p22*\  
p33
```

2 Index

B

Bayes_ball	5
bnet	1

C

cartesian	7
children	5

E

equivStatements	6
-----------------------	---

G

globalMarkov	5
--------------------	---

I

index	7
info	1

L

levels	9
localMarkov	3

M

map_observable	12
----------------------	----

Markov.lib	1
Markov_lib	1
MarkovIdeal	10

N

next	6
nondec	2
nondecminusparents	3

P

pairMarkov	2
Pairs	8
parent	3
Prob	9
probring	7

Q

Quad	9
------------	---

S

sdec	7
StatementQuadrics	10
subset	4

T

torideal	11
----------------	----