Pasture Software Manual

Justin Chen and Tianyi Zhang

February 12, 2024

Outline

Basic operations

Examples

Basic operations

Getting started with M2

Quick links:

M2 website: `https://macaulay2.com`

Try M2 online: `http://www.unimelb-macaulay2.cloud.edu.au/`

Matroids package documentation

Foundations.m2 on GitHub

Matroids package in a nutshell

We can build a matroid in M2 by specifying its bases (default), or circuits, or non-bases, ...

```
i1 : load "Matroids/foundations.m2"

i2 : viewHelp "Matroids"

i3 : E = toList(0..3)

o3 = {0, 1, 2, 3}

o3 : List

i4 : M1 = matroid(E, {set{0,1},set{0,2},set{0,3},set{1,2},set{1,3},set{2,3}})

o4 = a "matroid" of rank 2 on 4 elements
```

It doesn't matter if we use sets or lists to describe the bases:

```
i5 : M2 = matroid_E subsets(E, 2)

o5 = a "matroid" of rank 2 on 4 elements

o5 : Matroid

i6 : M1 == M2

o6 = true
```

Constructing matroids

```
i7 : M3 = matroid(E, subsets(E, 3), EntryMode => "circuits")

o7 = a "matroid" of rank 2 on 4 elements

o7 : Matroid

i8 : M1 == M3

o8 = true

i9 : U24 = uniformMatroid_2 4

o9 = a "matroid" of rank 2 on 4 elements

o9 : Matroid

i10 : M1 == U24

o10 = true
```

Many well-known matroids are implemented in the package like uniform matroids, affine geometry, Pappus, Vamos, etc.

Working with matroids

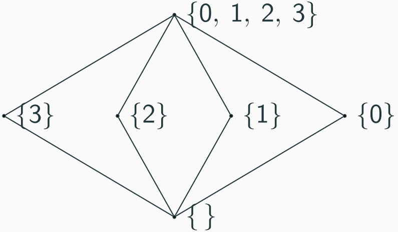
We can reveal a lot of information about the matroid, using various functions and commands.

```
i14 : ideal U24

o14 = monomialIdeal (x x x , x x x , x x x , x x x )
                      0 1 2   0 1 3   0 2 3   1 2 3

o14 : MonomialIdeal of QQ[x ..x ]
                           0   3

i15 : L = latticeOfFlats U24

o15 = L

o15 : Poset

i16 : displayPoset L

i17 : displayPoset(L, SuppressLabels => false)
```

Note: `displayPoset` opens an external viewer.

Example: lattice of flats

Here is the output of `displayPoset` for the lattice of flats of U_4^2.



(the TeX source can be obtained using the function `texPoset`).

Caching information

Much of the computed information is cached (which makes certain commands much quicker the second time they are run).

```
i18 : peek U24.cache

o18 = CacheTable{dual => a "matroid" of rank 2 on 4 elements
                 flats => {set {}, set {3}, set {2}, set {1}, set {0}, ...
                 flatsRelations => {{{}, {0}}, {{}, {0, 1, 2, 3}}, ...
                 flatsRelationsMatrix => | 1 1 1 1 1 1 |
                                         | 0 1 0 0 0 1 |
                                         | 0 0 1 0 0 1 |
                                         | 0 0 0 1 0 1 |
                                         | 0 0 0 0 1 1 |
                                         | 0 0 0 0 0 1 |
                 flatsRelationsTable => MutableHashTable{...6...}
                 groundSet => {0, 1, 2, 3}
                 hyperplanes => {set {3}, set {2}, set {1}, set {0}}
                 ideal => monomialIdeal (x x x , x x x , x x x , x x x )
                                          0 1 2   0 1 3   0 2 3   1 2 3
```

Pastures

We implement a pasture as an ordered triple of its multiplicative group, epsilon, and its fundamental pairs.

The most convenient way to input a pasture is to specify its generators, multiplicative and additive relations.

```
i19 : D = pasture([x,y], "x+y, x*y^(-1)")

o19 = a "pasture" on 2 generators with 1 hexagon

o19 : Pasture
```

Even easier to construct pasture from a finite field.

```
i20 : F2 = pasture GF 2

o20 = a "pasture" on 0 generators with 0 hexagons

o20 : Pasture
```

Pasture constructions

As with specific matroids, there is a database of specific pastures.

```
i21 : S = specificPasture sign

o21 = a "pasture" on 1 generator with 1 hexagon

o21 : Pasture

i22 : peek S

o22 = Pasture{cache => CacheTable{}                         }
              epsilon => | 1 |
              hexagons => {{{0, 0}, {0, | 1 |}, {0, | 1 |}}}
              multiplicativeGroup => cokernel | 2 |

i23 : code methods specificPasture

o23 = ...
```

Products and coproducts

We can calculate products and co-products of pastures. (Later on we discuss the relative versions.)

```
i24 : peek (D ** F2) -- coproduct (over F1pm)

o24 = Pasture{cache => CacheTable{}                                  }
              epsilon => 0
              hexagons => {{{| 1 |, | 1 |}, {| -1 |, 0}, {| -1 |, 0}}}
                                          1
              multiplicativeGroup => ZZ

i25 : peek (D * F2) -- product (over K)

o25 = Pasture{cache => CacheTable{}               }
              epsilon => | 1 |
                         | 0 |
              hexagons => {}
              multiplicativeGroup => cokernel | 2 |
                                              | 0 |
```

Foundations

We can compute the foundation of a matroid.

```
i26 : NF = specificMatroid nonfano

o26 = a "matroid" of rank 3 on 7 elements

o26 : Matroid

i27 : foundation NF

o27 = a "foundation" on 2 generators with 1 hexagon

i28 : peek NF.cache

o28 = ...

i29 : peek (foundation NF).cache

o29 = ...
```

Pasture morphisms

We can compute pasture morphisms (assuming that the source is generated by fundamental elements.) We represent morphisms as matrices giving the homomorphisms of multiplicative abelian groups. Thus we can do usual matrix operations, like multiply and take determinant.

```
i30 : U = foundation U24

o30 = a "foundation" on 3 generators with 1 hexagon

o30 : Foundation

i31 : k = GF 4

o31 = k

o31 : GaloisField

i32 : morphisms(U, pasture k, FindOne => true)

o32 = {| 0 -1 1 |}

o32 : List

i33 : morphisms(U, pasture k)

o33 = {| 0 -1 1 |, | 0 1 -1 |}

o33 : List
```

Isomorphisms

The FindIso option specifies that only invertible morphisms are found.

```
i34 : F = U ** U

o34 = a "pasture" on 5 generators with 2 hexagons

o34 : Pasture

i35 : #morphisms(F, F, FindIso => true)

o35 = 72

i36 : tally (morphisms(F, F) /det)

o36 = Tally{-1 => 36}
            0 => 72
            1 => 36

o36 : Tally

i37 : areIsomorphic(foundation(U24 ++ U24), F)

o37 = true

i38 : areIsomorphic(foundation NF, D)

o38 = true

i39 : areIsomorphic(foundation specificMatroid fano, F2)

o39 = true
```

Representations

Given a matroid M and a finite field F, we can compute the list of all representations of M over F up to rescaling equivalence.

```
i40 : representations(U24, k)

o40 = {| 1 0 1 1   |, | 1 0 1 1 |}
       | 0 1 1 a+1 |  | 0 1 1 a |

o40 : List

i41 : P = specificMatroid pappus

o41 = a "matroid" of rank 3 on 9 elements

o41 : Matroid

i42 : representations(P, GF 5)

o42 = {}

o42 : List

i43 : representations(P, GF 7)
```

Representations, II

```
o43 = {| 1 0 1 0 1 1 0  1 1  |, | 1 0 1 0 1  1  0  1 1  |, | 1 0 1 0 1  1  0
       | 0 1 1 0 3 3 1  1 -2 |  | 0 1 1 0 -1 -1 1  1 -2 |  | 0 1 1 0 -2 -2 1
       | 0 0 0 1 1 2 -2 3 2  |  | 0 0 0 1 1  -2 -1 2 -2 |  | 0 0 0 1 1  -3 3
      ------------------------------------------------------------------------
      1  1  |, | 1 0 1 0 1  1  0 1  1 |, | 1 0 1 0 1  1  0 1 1  |, | 1 0 1 0
      1  3  |  | 0 1 1 0 -3 -3 1 1  3 |  | 0 1 1 0 -2 -2 1 1 -1 |  | 0 1 1 0
      -2 -3 |  | 0 0 0 1 1  3  2 -1 3 |  | 0 0 0 1 1  3  3 2 3  |  | 0 0 0 1
      ------------------------------------------------------------------------
      1 1  0  1 1  |, | 1 0 1 0 1 1  0  1  1  |, | 1 0 1 0 1  1  0 1  1 |, |
      3 3  1  1 -1 |  | 0 1 1 0 2 2  1  1  -1 |  | 0 1 1 0 -2 -2 1 1  2 |  |
      1 -1 -2 2 -1 |  | 0 0 0 1 1 -2 -3 -1 -2 |  | 0 0 0 1 1  2  3 -1 2 |  |
      ------------------------------------------------------------------------
      1 0 1 0 1 1  0  1  1  |, | 1 0 1 0 1  1  0  1  1 |, | 1 0 1 0 1  1  0 1
      0 1 1 0 3 3  1  1  2  |  | 0 1 1 0 -1 -1 1  1  2 |  | 0 1 1 0 -3 -3 1 1
      0 0 0 1 1 -3 -2 -1 -3 |  | 0 0 0 1 1  3  -1 -3 3 |  | 0 0 0 1 1  -2 2 3
      ------------------------------------------------------------------------
      1  |, | 1 0 1 0 1  1  0 1  1  |, | 1 0 1 0 1 1  0  1  1  |, | 1 0 1 0 1
      2  |  | 0 1 1 0 -2 -2 1 1  -3 |  | 0 1 1 0 3 3  1  1  -3 |  | 0 1 1 0 2
      -2 |  | 0 0 0 1 1  -1 3 -3 -1 |  | 0 0 0 1 1 -2 -2 -3 -2 |  | 0 0 0 1 1
      ------------------------------------------------------------------------
      1 0  1  1  |}
      2 1  1  -3 |
      3 -3 -2 3  |
```

These are valid matrices over $\mathbb{F}_7$ in M2. Moreover, in some cases we can use field representations to determine partial field representations (as we will see later).

Single element extension

We can do one element extension for matroids that are represented over a finite field. The procedure is simple, just adding one column.

```
i44 : elapsedTime extensions(P, k, {})
 -- 4.49239 seconds elapsed

o44 = {a "matroid" of rank 3 on 10 elements, a "matroid" of rank 3 on 10
      ------------------------------------------------------------------------
      elements}

o44 : List

i45 : elapsedTime extensions(P, k, {S})
 -- 7.27329 seconds elapsed

o45 = {a "matroid" of rank 3 on 10 elements}

o45 : List
```

The 3rd entry is a list of pastures which we require the matroids to be representable over.

Question
Can we do the same thing for partial fields without too much effort?

Fundamental diagrams

We can construct fundamental diagrams of a matroid.

```
i46 : U25 = uniformMatroid_2 5

o46 = a "matroid" of rank 2 on 5 elements

o46 : Matroid

i47 : U36 = uniformMatroid_3 6

o47 = a "matroid" of rank 3 on 6 elements

o47 : Matroid

i48 : fD = fundamentalDiagram U36

o48 = ({30, 6, 0, 6, 0, 0, 0, 0}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
      ------------------------------------------------------------------------
      13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
      ------------------------------------------------------------------------
      31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41}, {{0, 30}, {0, 37}, {1,
      ------------------------------------------------------------------------
      30}, {1, 38}, {2, 30}, {2, 36}, {3, 30}, {3, 39}, {4, 30}, {4, 40}, {5,
      ------------------------------------------------------------------------
      31}, {5, 37}, {6, 31}, {6, 38}, {7, 31}, {7, 36}, {8, 31}, {8, 39}, {9,
      ------------------------------------------------------------------------
      31}, {9, 41}, {10, 32}, {10, 37}, {11, 32}, {11, 38}, {12, 32}, {12,
      ------------------------------------------------------------------------
      36}, {13, 32}, {13, 40}, {14, 32}, {14, 41}, {15, 33}, {15, 37}, {16,
```

Fundamental diagrams, II

```
    --------------------------------------------------------------------------
    33}, {16, 39}, {17, 33}, {17, 36}, {18, 33}, {18, 40}, {19, 33}, {19,
    --------------------------------------------------------------------------
    41}, {20, 34}, {20, 37}, {21, 34}, {21, 39}, {22, 34}, {22, 38}, {23,
    --------------------------------------------------------------------------
    34}, {23, 40}, {24, 34}, {24, 41}, {25, 35}, {25, 36}, {26, 35}, {26,
    --------------------------------------------------------------------------
    39}, {27, 35}, {27, 38}, {28, 35}, {28, 40}, {29, 35}, {29, 41}})

o48 : Sequence
```

The output is a triple. The first entry is the number of embedded minors of types $\{U_4^2, U_5^2, {C_5}^1, U_5^3, C_5^*, U_4^2 \oplus U_1^0, U_4^2 \oplus U_1^1, U_4^2 \oplus U_2^1\}$ respectively. The second and third entries are the vertices and edges of the graph.

```
i49 : G = graph(fD#1, fD#2);

i50 : displayGraph G
-- running: dot -Tjpg /tmp/M2-122444-0/2.dot -o /tmp/M2-122444-0/3.jpg
```

Note: `displayGraph` opens an external viewer.

[1] The one with rank 3

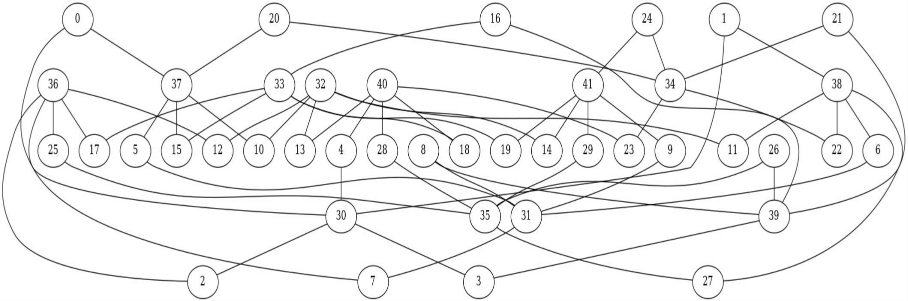
Fundamental diagrams, III

We can create fundamental diagrams with our choice of embedded minors. The first list consists of the minors that play the role of U_4^2 minors, while the second lists consists of the minors that play the role of the other minors in a default fundamental diagram.

```
i51 : fundamentalDiagram3Connected (U36, {U24}, {U25, dual U25})

o51 = ({30, 6, 6}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
      ------------------------------------------------------------------------
      17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
      ------------------------------------------------------------------------
      35, 36, 37, 38, 39, 40, 41}, {{0, 30}, {1, 30}, {2, 30}, {3, 30}, {4,
      ------------------------------------------------------------------------
      30}, {5, 31}, {6, 31}, {7, 31}, {8, 31}, {9, 31}, {10, 32}, {11, 32},
      ------------------------------------------------------------------------
      {12, 32}, {13, 32}, {14, 32}, {15, 33}, {16, 33}, {17, 33}, {18, 33},
      ------------------------------------------------------------------------
      {19, 33}, {20, 34}, {21, 34}, {22, 34}, {23, 34}, {24, 34}, {25, 35},
      ------------------------------------------------------------------------
      {26, 35}, {27, 35}, {28, 35}, {29, 35}, {2, 36}, {7, 36}, {12, 36},
      ------------------------------------------------------------------------
      {17, 36}, {25, 36}, {0, 37}, {5, 37}, {10, 37}, {15, 37}, {20, 37}, {1,
      ------------------------------------------------------------------------
      38}, {6, 38}, {11, 38}, {22, 38}, {27, 38}, {3, 39}, {8, 39}, {16, 39},
      ------------------------------------------------------------------------
      {21, 39}, {26, 39}, {4, 40}, {13, 40}, {18, 40}, {23, 40}, {28, 40},
      ------------------------------------------------------------------------
      {9, 41}, {14, 41}, {19, 41}, {24, 41}, {29, 41}})
```

Fundamental diagrams, IV

Here is the output of `displayGraph` for the fundamental diagram of U_6^3.



It takes time to create such a graph when the fundamental diagram is huge. For instance, the fundamental diagram of the Vamos matroid has 552 vertices (and takes ≈ 1 min to create the graph).

Examples

Minimal 3-connected extensions of C_5

Here is a process to search for minimal extensions of C_5. We use the methods `hasMinor` and `is3Connected` in the Matroids package.

```
i54 : C5 = specificMatroid C5

o54 = a "matroid" of rank 3 on 5 elements

o54 : Matroid

i55 : L6 = select(allMatroids 6, is3Connected)

o55 = {a "matroid" of rank 2 on 6 elements, a "matroid" of rank 3 on 6
      ----------------------------------------------------------------------
      elements, a "matroid" of rank 3 on 6 elements, a "matroid" of rank 3 on
      ----------------------------------------------------------------------
      6 elements, a "matroid" of rank 3 on 6 elements, a "matroid" of rank 3
      ----------------------------------------------------------------------
      on 6 elements, a "matroid" of rank 4 on 6 elements}

o55 : List

i56 : min6 = select(L6, M -> hasMinor(M, C5))

o56 = {a "matroid" of rank 3 on 6 elements, a "matroid" of rank 3 on 6
      ----------------------------------------------------------------------
      elements, a "matroid" of rank 3 on 6 elements}

o56 : List
```

Minimal extensions, II

```
i57 : L7 = select(allMatroids 7, is3Connected)

o57 = {a "matroid" of rank 2 on 7 elements, a "matroid" of rank 3 on 7
      ------------------------------------------------------------------------
      elements, a "matroid" of rank 3 on 7 elements, a "matroid" of rank 3 on
      ------------------------------------------------------------------------
      7 elements, a "matroid" of rank 3 on 7 elements, a "matroid" of rank 3
      ------------------------------------------------------------------------
      on 7 elements, a "matroid" of rank 3 on 7 elements, a "matroid" of rank
      ------------------------------------------------------------------------
      3 on 7 elements, a "matroid" of rank 3 on 7 elements, a "matroid" of
      ------------------------------------------------------------------------
      rank 3 on 7 elements, a "matroid" of rank 3 on 7 elements, a "matroid"
      ------------------------------------------------------------------------
      of rank 3 on 7 elements, a "matroid" of rank 3 on 7 elements, a
      ------------------------------------------------------------------------
      "matroid" of rank 3 on 7 elements, a "matroid" of rank 3 on 7 elements,
      ------------------------------------------------------------------------
      a "matroid" of rank 3 on 7 elements, a "matroid" of rank 3 on 7
      ------------------------------------------------------------------------
      elements, a "matroid" of rank 3 on 7 elements, a "matroid" of rank 3 on
      ------------------------------------------------------------------------
      7 elements, a "matroid" of rank 4 on 7 elements, ...}

o57 : List

i58 : select(L7, M -> hasMinor(M, C5) and all(min6, N -> not hasMinor(M, N)))

o58 = {}

o58 : List
```

Saving and reading foundations to/from a file

In the remaining examples, we search within all rank 4 matroids on 8 elements for certain properties. To prepare this, we can save the list of foundations to an external file, and read it back later into M2.

```
i59 : r4n8 = allMatroids(8,4); -- 940 matroids
```

We precompute foundation information as follows:

```
apply(#r4n8, i -> (
     fullRankSublattice foundation r4n8#i;
     saveFoundation(r4n8#i, toString i | ".txt")
))
```

On resuming a new M2 session, we then run

```
i60 : elapsedTime L48 = apply(940, i -> readFoundation(r4n8#i, toString i | "
.txt"));
 -- 52.7198 seconds elapsed
```

Foundations with 1 hexagon

We will search for foundations of rank 4 matroids on 8 elements (without U^3_6 minors) that has exactly one hexagon as an example. We start by finding the indices of such matroids with exactly one hexagon.

```
i61 : singleHex = positions(L48, F -> #F.hexagons == 1)

o61 = {183, 228, 259, 268, 428, 432, 442, 465, 489, 493, 501, 504, 507, 510,
      ------------------------------------------------------------------------
      512, 513, 517, 519, 530, 545, 548, 549, 561, 562, 563, 565, 576, 578,
      ------------------------------------------------------------------------
      585, 597, 599, 603, 604, 609, 637, 659, 661, 663, 664, 665, 666, 669,
      ------------------------------------------------------------------------
      671, 685, 687, 689, 690, 691, 692, 707, 709, 711, 712, 714, 715, 717,
      ------------------------------------------------------------------------
      718, 722, 724, 725, 727, 729, 730, 731, 734, 738, 740, 744, 745, 748,
      ------------------------------------------------------------------------
      750, 752, 764, 765, 767, 769, 771, 775, 776, 778, 779, 783, 784, 786,
      ------------------------------------------------------------------------
      787, 791, 797, 798, 799, 800, 805, 806, 807, 815, 822, 841, 844, 854,
      ------------------------------------------------------------------------
      860, 861, 863, 866, 867, 873, 876, 877, 879, 885, 886, 887, 888, 893,
      ------------------------------------------------------------------------
      894, 895, 902, 908, 919, 922, 925, 927}

o61 : List

i62 : #singleHex

o62 = 120
```

Isotypes

As it turns out, none of these have U_6^3 as a minor.

```
i63 : elapsedTime all(singleHex, i -> not hasMinor(r4n8#i, U36))
 -- 15.8002 seconds elapsed

o63 = true
```

Finally, we identify the distinct isomorphism types among these foundations.

```
i64 : elapsedTime isoTypes L48_singleHex
 -- 4.05891 seconds elapsed

o64 = {a "foundation" on 2 generators with 1 hexagon, a "foundation" on 2
      --------------------------------------------------------------------
      generators with 1 hexagon, a "foundation" on 1 generator with 1
      --------------------------------------------------------------------
      hexagon, a "foundation" on 1 generator with 1 hexagon, a "foundation"
      --------------------------------------------------------------------
      on 1 generator with 1 hexagon, a "foundation" on 3 generators with 1
      --------------------------------------------------------------------
      hexagon}

o64 : List

i65 : #oo

o65 = 6
```

These 6 pastures are precisely $\mathbb{D}, \mathbb{G}, \mathbb{F}_3, \mathbb{D} \otimes \mathbb{F}_2, \mathbb{H}$, and $\mathbb{U}$.

Zach's counterexample

This example comes from Zach's counterexample that the foundation of a modular sum is not equal to the relative tensor product of the foundations over the foundation of the modular flat.

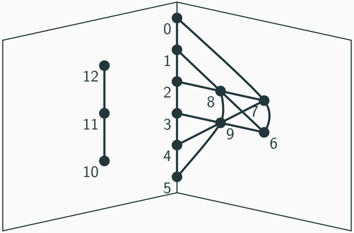


Figure 1: A generalized parallel connection for which $F(P_T(M_1, M_2))$ is not isomorphic to $F(M_1) \otimes_{F(T)} F(M_2)$.

Zach's counterexample, II

Here is how we build the matroid M.

```
i70 : I1 = {{0,6,7},{1,6,8},{2,7,8},{3,6,9},{4,7,9},{5,8,9}}

o70 = {{0, 6, 7}, {1, 6, 8}, {2, 7, 8}, {3, 6, 9}, {4, 7, 9}, {5, 8, 9}}

o70 : List

i71 : N1 = matroid(10, I1 | (bases U36)/toList, EntryMode => "nonbases")

o71 = a "matroid" of rank 3 on 10 elements

o71 : Matroid

i72 : type1 = flatten apply({10,11,12}, e -> apply(bases N1, s -> s + set{e}));

i73 : I2 = subsets(10,2) - set subsets(6,2);

i74 : type2 = flatten apply({10,11,12}, e -> apply(I2, s -> set s + set{10,11
,12} - set{e}));

i75 : M = matroid(13, type1 | type2)

o75 = a "matroid" of rank 4 on 13 elements

o75 : Matroid

i76 : isWellDefined M

o76 = true
```

Zach's counterexample, III

```
i78 : hexTypes foundation M

o78 = HashTable{"D" => 0 }
                "F3" => 0
                "H" => 0
                "U" => 31

o78 : HashTable
```

Next, we construct the two factors separately.

```
i79 : N1 == M \ {10,11,12}

o79 = true

i80 : f9 = inducedMapFromMinor(N1, 9, "delete");

o80 : PastureMorphism

i81 : f8 = inducedMapFromMinor(N1 \ {9}, 8, "delete");

o81 : PastureMorphism

i82 : f7 = inducedMapFromMinor(N1 \ {9,8}, 7, "delete");

o82 : PastureMorphism

i83 : f6 = inducedMapFromMinor(N1 \ {9,8,7}, 6, "delete");
```

Zach's counterexample, IV

```
i84 : phi2 = f9*f8*f7*f6

o84 = | 1 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 1 0 0  0  |
      | 0 1  0 1 0 0 0 0 -1 -1 |
      | 0 -1 0 0 1 0 0 0 0  1  |
      | 0 0  0 0 0 1 0 0 1  1  |
      | 0 1  0 0 0 0 0 1 0  0  |

o84 : PastureMorphism
```

For the other factor, we need to be a bit careful with the indices.

```
i85 : N2 = M \ {6,7,8,9}

o85 = a "matroid" of rank 3 on 9 elements

o85 : Matroid

i86 : (N2_*, groundSet N2)

o86 = ({0, 1, 2, 3, 4, 5, 10, 11, 12}, set {0, 1, 2, 3, 4, 5, 6, 7, 8})

o86 : Sequence

i87 : f12 = inducedMapFromMinor(N2, 8, "delete");

o87 : PastureMorphism
```

Zach's counterexample, V

```
i88 : f11 = inducedMapFromMinor(N2 \ {12}, 7, "delete");

o88 : PastureMorphism

i89 : f11 === inducedMapFromMinor(N2 \ set{8}, 7, "delete")

o89 = true

i90 : f10 = inducedMapFromMinor(N2 \ {12,11}, 6, "delete");

o90 : PastureMorphism

i91 : phi1 = f12*f11*f10

o91 = | 1 0  0  0  0  0  0  0  0  0  |
      | 0 1  0  0  0  0  0  0  0  0  |
      | 0 0  1  0  0  0  0  0  0  0  |
      | 0 0  0  1  0  0  0  0  0  0  |
      | 0 0  0  0  1  0  0  0  0  0  |
      | 0 0  0  0  0  1  0  0  0  0  |
      | 0 0  0  0  0  0  1  0  0  0  |
      | 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 |
      | 0 0  0  0  0  0  0  1  0  0  |
      | 0 0  0  0  0  0  0  0  1  0  |
      | 0 0  0  0  0  0  0  0  0  1  |
      | 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  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  |
      | 0 0  0  0  0  0  0  0  0  0  |
```

Zach's counterexample, VI

Finally, we compute the fiber coproduct of `phi1` and `phi2`.

```
i92 : fiberCoproduct(phi1, phi2)

o92 = a "pasture" on 14 generators with 30 hexagons

o92 : Pasture

i93 : hexTypes oo

o93 = HashTable{"D" => 0 }
                "F3" => 0
                "H" => 0
                "U" => 30

o93 : HashTable
```

Changxin's examples

The goal is to find all representations of 3-connected 6th-root of unity matroids. We can lift representations over $\mathbb{F}_7$ to get the 6th-root of unity representations.

```
i94 : H = specificPasture H

o94 = a "pasture" on 1 generator with 1 hexagon

o94 : Pasture

i95 : F1 = specificPasture F1pm

o95 = a "pasture" on 1 generator with 0 hexagons

o95 : Pasture

i96 : elapsedTime positions(L48, F -> not areIsomorphic(F1, F) and #morphisms
(F, H, FindOne => true) > 0)
 -- 40.3762 seconds elapsed

o96 = {385, 428, 442, 467, 483, 493, 501, 507, 510, 512, 513, 517, 530, 545,
      ------------------------------------------------------------------------
      554, 556, 561, 562, 563, 565, 576, 585, 588, 589, 597, 598, 599, 603,
      ------------------------------------------------------------------------
      604, 609, 610, 637, 639, 640, 641, 659, 663, 664, 665, 666, 671, 684,
      ------------------------------------------------------------------------
```

Changxin's examples, II

```
      685, 687, 689, 690, 691, 692, 707, 708, 709, 711, 712, 714, 715, 717,
      ------------------------------------------------------------------------
      718, 720, 722, 724, 725, 727, 729, 730, 731, 734, 738, 740, 744, 745,
      ------------------------------------------------------------------------
      748, 750, 752, 762, 764, 765, 767, 769, 771, 775, 776, 778, 779, 783,
      ------------------------------------------------------------------------
      784, 786, 787, 791, 797, 798, 799, 800, 805, 806, 807, 815, 822, 841,
      ------------------------------------------------------------------------
      854, 855, 858, 860, 861, 863, 866, 867, 871, 873, 876, 877, 879, 885,
      ------------------------------------------------------------------------
      886, 887, 888, 893, 894, 895, 902, 908, 919, 922, 925, 927}

i97 : select(oo, i -> is3Connected r4n8#i)

o97 = {428, 442, 501, 507, 510, 512, 513, 517, 585}

o97 : List

i98 : netList apply(oo, i -> representations(r4n8#i, GF 7))

      +----------------------+----------------------+
o98 = || 1 0 1 0 1  0 1 0  ||| 1 0 1 0 1 0 1 0   | |
      || 0 1 1 0 -2 0 0 1  ||| 0 1 1 0 3 0 0 1   | |
      || 0 0 0 1 1  0 1 2  ||| 0 0 0 1 1 0 1 -3 | |
      || 0 0 0 0 0  1 1 -1 ||| 0 0 0 0 0 1 1 -1 | |
      +----------------------+----------------------+
      ...
```

Searching quaternary orientable matroids

```
i99 : elapsedTime quaternary8 = positions(L48, F -> #morphisms(F, pasture k,
FindOne => true) > 0)
 -- 36.1285 seconds elapsed

o99 = {72, 111, 159, 178, 193, 210, 214, 228, 230, 263, 269, 293, 307, ...

i100 : elapsedTime quaternaryOrientable8 = select(quaternary8, i -> #morphism
s(L48#i, S, FindOne => true) > 0)
 -- 12.8828 seconds elapsed

o100 = {72, 111, 178, 214, 228, 293, 307, 316, 320, 324, 358, 360, 369, ...

i101 : elapsedTime quatOri3Conn = select(quaternaryOrientable8, i -> is3Conne
cted r4n8#i)
 -- 7.61115 seconds elapsed

o101 = {72, 111, 178, 214, 228, 293, 307, 316, 358, 360, 369, 374, 376, 394,
       ------------------------------------------------------------------------
       408, 416, 420, 423, 426, 433, 435, 439, 441, 442, 450, 453, 456, 461,
       ------------------------------------------------------------------------
       463, 478, 481, 485, 488, 492, 497, 498, 501, 503, 506, 507, 509, 510,
       ------------------------------------------------------------------------
       511, 512, 513, 514, 515, 517, 518, 584, 585}
```

We can keep doing this to investigate all quaternary orientable matroids of rank 4.
The method "extensions" is exponential in terms of the rank.