

# 1 Composition Direction of Seymour’s Theorem for 2 Regular Matroids—Formally Verified

3 **Martin Dvorak** ✉   
ISTA, Klosterneuburg, Austria

4 **Rida Hamadani** ✉   
LMAP, UPPA, Pau, France

5 **Evgenia Karunus** ✉   
University Of Bonn, Bonn, Germany

6 **Alexander Meiburg** ✉   
University of Waterloo & Perimeter Institute of  
Theoretical Physics, Waterloo, Canada

7 **Peter Nelson** ✉   
University of Waterloo, Waterloo, Canada

8 **Cameron Rampell** ✉  
Independent, Palo Alto, United States of America

**Tristan Figueroa-Reid** ✉   
Reed College, Portland, United States

**Byung-Hak Hwang** ✉   
Korea Institute for Advanced Study, Seoul, South  
Korea

**Vladimir Kolmogorov** ✉   
ISTA, Klosterneuburg, Austria

**Alexander Nelson** ✉   
Independent

**Mark Sandey** ✉   
UC Riverside, Riverside, United States of America

**Ivan Sergeev** ✉   
ISTA, Klosterneuburg, Austria

## 10 — Abstract —

11 Seymour’s decomposition theorem is a hallmark result in matroid theory presenting a structural  
12 characterization of the class of regular matroids. Formalization of matroid theory faces many  
13 challenges, most importantly that only a limited number of notions and results have been implemented  
14 so far. In this work, we formalize the proof of the forward (composition) direction of Seymour’s  
15 theorem for regular matroids. To this end, we develop a library in Lean 4 that implements definitions  
16 and results about totally unimodular matrices, vector matroids, their standard representations,  
17 regular matroids, and 1-, 2-, and 3-sums of matrices and binary matroids given by their standard  
18 representations. Using this framework, we formally state Seymour’s decomposition theorem and  
19 implement a formally verified proof of the composition direction in the setting where the matroids  
20 have finite rank and may have infinite ground sets.

21 **2012 ACM Subject Classification** Mathematics of computing → Matroids and greedoids; General  
22 and reference → General conference proceedings; General and reference → Verification

23 **Keywords and phrases** totally unimodular matrices, regular matroids, Seymour’s decomposition  
24 theorem, calculus of inductive constructions

25 **Digital Object Identifier** 10.4230/LIPIcs.ITP.2026.75

26 **Acknowledgements** We dedicate this paper to Klaus Truemper, whose monograph Matroid Decom-  
27 position [12] laid the foundation for our entire work.

## 28 **1** Introduction

29 Seymour’s regular matroid decomposition theorem is a hallmark structural result in matroid  
30 theory [9, 12, 4, 7]. It states that, on the one hand, any 1-, 2-, and 3-sum of two regular  
31 matroids is regular, and on the other hand, any regular matroid can be decomposed into  
32 matroids that are graphic, cographic, or isomorphic to  $R_{10}$  by repeated 1-, 2-, and 3-sum  
33 decompositions.

34 The interest in matroids comes from the fact that they capture and generalize many  
35 mathematical structures and properties, such as linear independence (captured by vector



© Martin Dvorak et al.;  
licensed under Creative Commons License CC-BY 4.0  
17th International Conference on Interactive Theorem Proving (ITP 2026).  
Editor: 68; Article No. 75; pp. 75:1–75:15



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

matroids), graphs (graphic matroids), and extensions of fields (algebraic matroids). Another advantage of matroids is that they admit a relatively short definition, making them amenable to formalization. As for Seymour’s theorem, it not only presents a structural characterization of the class of regular matroids, but also leads to several important applications, such as polynomial algorithms for testing if a matroid is binary and for testing if a matrix is totally unimodular. Additionally, Seymour’s theorem can offer a structural approach for solving certain combinatorial optimization problems, for example, it leads to the characterization and efficient algorithms for the cycle polytope.

Formalization of results about matroids faces several challenges. One of them is that the support for them is limited. In Mathlib, only selected basic definitions for matroids are implemented, such as maps, duals, and minors. However, many other fundamental notions are not yet implemented, including representability and regularity, the splitter theorem and the separation algorithm. Part of the difficulty stems from the fact that classically, matroids are defined only in the finite case (i.e., when the ground set and the rank are finite), while Mathlib implements matroids more generally, allowing them to be infinite and to have infinite rank. Additionally, the proofs presented in the existing literature require substantial additional work to make them easily amenable to formalization.

The goal of our work was to develop a general and reusable library proving a result that is at least as strong as the forward (composition) direction of classical Seymour’s theorem (i.e., stated for finite matroids). Moreover, our aim was to make our library modular and extensible by ensuring compatibility with matroids in Mathlib [8].

To achieve our goals, we made the following compromises. First, we focused on the implementation of the proof of the composition direction, while only stating the decomposition direction. Second, we assumed finiteness where it would simplify proofs, while making sure that the final results held for finite matroids (in fact, they hold for matroids with potentially infinite ground set and finite rank). Finally, we tailored our implementation specifically to Seymour’s theorem, avoiding introducing additional matroid notions if possible. Our project makes the following contributions:

- Formalized definition and selected properties of totally unimodular matrices, some of which were added to Mathlib.
- Implemented definitions and formally proved selected results about vector matroids, their standard representations, regular matroids, and 1-, 2-, and 3-sums of matrices and vector matroids given by their standard representations.
- Implemented a formally verified proof of the composition direction of Seymour’s theorem, i.e., that any 1-, 2-, and 3-sum of two regular matroids is regular, in the case where the matroids may have infinite ground sets and have finite rank.
- Implemented a formally verified proof that graphic and cographic matroids are regular.
- Stated the decomposition direction of Seymour’s theorem, i.e., that any regular matroid of finite rank can be decomposed into graphic matroids, cographic matroids, and matroids isomorphic to  $R_{10}$  by repeated 1-, 2-, and 3-sum decompositions.

Our formalization<sup>1</sup> is conceptually split into two parts: “implementation” and “presentation”. Implementation is contained in the `Seymour` folder and encompasses all definitions and lemmas used to obtain our results. Presentation is contained in the `Seymour.lean` file, which repeats selected definitions and theorems comprising the key final results of our contribution.

---

<sup>1</sup> link removed for blinded version

80 Every definition in the “presentation” file is checked to be definitionally equal to its coun-  
81 terpart from the “implementation” using the `recall` or the `example` command. Similarly,  
82 we `recall` every theorem presented here and then use the `#guard_msgs in #print axioms`  
83 command to check that the implementation of its proof (including the entire dependency tree)  
84 depends only on the three axioms `[propext, Classical.choice, Quot.sound]`, which are  
85 standard for Lean projects that use classical logic.

86 We refer to the statements of the final results and the definitions they (transitively)  
87 depend on as *trusted code*. The `Seymour.lean` file repeats all nontrivial trusted code, so that  
88 the reader can believe [10] our results without having to examine the entire implementation,  
89 assuming that the reader also uses the Lean compiler to check that all proofs are correct.  
90 Note that basic definitions from Lean and Mathlib are part of the trusted code but are not  
91 repeated in `Seymour.lean`, and we let the reader decide whether to blindly trust them or  
92 read them as well.

93 While working on our project, we leveraged the LeanBlueprint<sup>2</sup> tool to help guide our  
94 formalization efforts. In particular, we used it to create theoretical blueprints and dependency  
95 graphs, which allowed us to get a clearer overview of the results we were formalizing, as  
96 well as their dependencies. In our workflow, we first created a write-up encompassing the  
97 classical results from [12]. Based on this write-up, we developed a self-contained theoretical  
98 blueprint for our formalization by filling in gaps, fleshing out technical details, and sometimes  
99 re-working certain proofs. We followed this blueprint during the development of our library,  
100 keeping it up to date and turning it into documentation of our code.

101 We use Lean version 4.18.0 and we import Mathlib library revision aa936c3 (dated  
102 2025-04-01).

103 We made the code snippets in this paper as faithful to the content of the repository as  
104 possible, though we made some omissions. In particular, proofs inside definitions were replaced  
105 by the `sorry` keyword in the paper, while the repository contains full implementation.

## 106 2 Theory Underpinning the Formalization

107 There are two classical sources presenting the proof of Seymour’s decomposition theorem: [9]  
108 and [12], each with their own advantages and disadvantages.

109 Oxley2011 [9] develops a general theory of matroids and has a broader focus. It introduces  
110 many abstract notions and proves many statements about them, and Seymour’s theorem and  
111 its dependencies are also stated and proved in terms of these abstract notions alongside many  
112 other results. The advantages of following [9] would be the higher reusability, generality,  
113 and extensibility of the formalization. Indeed, since [9] introduces a lot of foundational  
114 notions and results, the resulting implementation could serve as the basis for formalization of  
115 many other results from classical matroid theory. Moreover, [9] is more general than [12] in  
116 certain aspects, for example, [12] defines 1- and 2-sums only for binary matroids, while their  
117 definitions in [9] do not have this restriction. Finally, it seems that the approach to theory of  
118 infinite matroids [3] is more closely aligned with the approach of [9] than [12], which might  
119 make it easier to generalize formalizations based on the former than the latter to the infinite  
120 matroid setting. However, proof formalization following [9] would face many challenges. First,  
121 the support for matroids in Mathlib [8] at the time we carried out our project was quite  
122 limited. Thus a lot of time would be dedicated to developing low-level definitions and results  
123 about them, especially in the infinite matroid setting to ensure compatibility with Mathlib.

---

<sup>2</sup> <https://github.com/PatrickMassot/leanblueprint>

124 Second, certain intermediate results could turn out difficult to formally prove. From our  
 125 experiments, proving the equivalence of multiple characterizations of regular matroids turned  
 126 out hard to formalize. Finally, [9] leaves many technical steps as exercises for the reader,  
 127 most crucially leaving out the proof of regularity of 3-sum, and contains many proofs that  
 128 crucially rely on graph theory which was not supported in Mathlib. This would make it  
 129 challenging to convert the proofs to their formalized versions.

130 In contrast, Truemper2016 [12] focuses on decomposition and composition of matroids,  
 131 with Seymour’s theorem being one of the most prominent theorems that it builds towards.  
 132 Truemper2016 [12] more frequently than Oxley2011 [9] utilizes explicit matrix representations  
 133 in definitions, theorems, and proofs, especially when it comes to 1-, 2-, and 3-sums of regular  
 134 matroids. Thus, following [12] would require implementing fewer intermediate definitions  
 135 and results to begin working with Seymour’s theorem itself. Moreover, Mathlib’s support for  
 136 matrices and linear independence was more extensive than for matroids, so this would allow  
 137 us to build upon more things that were already available. However, following the approach  
 138 of [12] had several important limitations. As mentioned earlier, it would be less general and  
 139 potentially less amenable to generalization to the infinite matroid setting than [9]. Moreover,  
 140 faithfully following [12] would mean implementing similar definitions and theorems on several  
 141 levels of abstraction. More specifically, 1-, 2-, and 3-sums would need to be implemented  
 142 separately for matrices, binary matroids defined by standard representation matrices, and  
 143 binary matroids in general, and the results about the sums of these objects would need to be  
 144 proved and propagated accordingly. Last but not least, similar to [9], one would need to fill  
 145 in the omitted technical details and re-work proofs that could be extremely challenging to  
 146 formalize directly, especially those involving graph-theoretic arguments.

147 Ultimately, we decided to follow the approach of [12] over [9] for formalizing Seymour’s  
 148 theorem, as it aligned more closely with our goals and values. We aimed to formalize the  
 149 statement of Seymour’s theorem and the proof of the composition direction, so having to  
 150 implement fewer intermediate definitions and lemmas and being able to use more tools from  
 151 Mathlib was a big plus. Though we did not mind limiting the generality of our contributions  
 152 to classical results, our final results go beyond that and hold for matroids of finite rank with  
 153 potentially infinite ground sets. The completeness of the presentation in [12] allowed us to  
 154 develop a theoretical blueprint, where we fleshed out the technical details, circumvented  
 155 problematic intermediate results, and streamlined the proofs, especially in the case of 3-sums.

### 156 **3 Proof Outline and Design Choices**

157 Before delving into technical details, we outline the structure of our formal proof and explain  
 158 key design decisions. Our development mirrors the theoretical decomposition: we implement  
 159 each matroid sum (1-, 2-, and 3-sum) at three levels (matrix, standard representation, and  
 160 abstract matroid) to manage complexity. For each sum, we first prove that the matrix-level  
 161 construction preserves total unimodularity, then lift this result to the matroid level via the  
 162 StandardRepr abstraction. The 1-sum and 2-sum proofs closely follow and streamline the  
 163 arguments in Truemper’s work, while the 3-sum case required a new approach. We re-designed  
 164 the 3-sum proof to avoid formalizing a difficult graph-theoretic re-signing argument: instead,  
 165 we re-sign each summand only once and introduce an intermediate structure `MatrixLikeSum3`  
 166 to capture the combined matrix blocks. This strategic design lets us systematically derive  
 167 total unimodularity for 3-sums (reusing parts of the 2-sum argument) and circumvents the  
 168 need for a complex graph argument in Lean. We also split our code into an “implementation”  
 169 (detailed definitions and lemmas) and a “presentation” (key results with Lean’s `#guard_msgs`

170 checks), ensuring the proof is both modular and trustworthy.

## 171 **4 Preliminaries**

172 This section reviews Mathlib declarations our code relies on.

173 Throughout this paper, we write  $\mathbb{Z}_n$  to denote `ZMod n` for any positive integer  $n$ , most  
174 often in the case  $\mathbb{Z}_2$  denoting `ZMod 2`, which is also written as `Z2` in the code.

### 175 **4.1 Matroids**

176 Matroids have many equivalent definitions [9, 12, 3]. In Mathlib, the structure `Matroid`  
177 captures the definition via the *base axioms* from [3]: a *matroid* is a pair  $M = (E, \mathcal{B})$  where  
178  $E$  is a (potentially infinite) ground set and  $\mathcal{B} \subseteq 2^E$  is a collection of sets such that:

179 (i)  $\mathcal{B} \neq \emptyset$ .

180 (ii) For all  $B_1, B_2 \in \mathcal{B}$  and all  $b_1 \in B_1 \setminus B_2$ , there exists  $b_2 \in B_2 \setminus B_1$  such that  $(B_1 \setminus \{b_1\}) \cup$   
181  $\{b_2\} \in \mathcal{B}$ .

182 (iii) For all  $X \subseteq E$  and  $I \subseteq X$  such that  $I \subseteq B_1$  for some  $B_1 \in \mathcal{B}$ , there exists a maximal  $J$   
183 such that  $I \subseteq J \subseteq X$  and  $J \subseteq B_2$  for some  $B_2 \in \mathcal{B}$ .

184 A set  $B \in \mathcal{B}$  is called a *base*, and 2 is known as the *base exchange property*. Additionally,  
185 if a set  $I \subseteq E$  is a subset of any base, then  $I$  is called *independent*. The definition above  
186 generalizes the classical notion of matroids [9, 12], which can only have finite ground sets.  
187 In our work, we construct matroids using the equivalent *independence axioms*, available  
188 in Mathlib as `IndepMatroid`. We use the assumption that the matroid has a finite rank  
189 (`RankFinite` in Mathlib). Note that the ground set is allowed to be infinite.

### 190 **4.2 Totally Unimodular Matrices**

191 In our work, regular matroids are defined in terms of totally unimodular matrices [9,  
192 12]. Before introducing their definition, let us review how matrices and submatrices are  
193 implemented in Mathlib. A matrix with rows indexed by  $m$ , columns indexed by  $n$ , and entries  
194 of type  $\alpha$  is represented by `Matrix m n α`, implemented as a (curried [11]) binary function  
195  $m \rightarrow n \rightarrow \alpha$ . Thus, the elements of matrix  $A$  can be accessed with `A i j`. Similarly,  
196 `Matrix.submatrix f g` is defined so that  $(A.\text{submatrix } f\ g) i\ j = A (f\ i) (g\ j)$  holds.  
197 Note that `Matrix.submatrix` may repeat and reorder rows and columns. For example, if

$$198 \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad f = ![0], \quad g = ![2, 2, 0, 0],$$

199 then `A.submatrix f g = [3 3 1 1]`, typed as a matrix, not a vector.

200 Now, a matrix  $A$  over a commutative ring  $R$  is called *totally unimodular* if every finite  
201 square submatrix of  $A$  (not necessarily contiguous, with no row or column taken twice) has  
202 determinant in  $\{-1, 0, 1\}$ . We implement this definition as follows:

```
def Matrix.IsTotallyUnimodular {m n R : Type*} [CommRing R] (A : Matrix m n R) : Prop :=
  ∀ k : ℕ, ∀ f : Fin k → m, ∀ g : Fin k → n, f.Injective → g.Injective →
    (A.submatrix f g).det ∈ Set.range SignType.cast
```

203 Here, `SignType` is an inductive type with three values: `zero`, `neg`, and `pos`; and `SignType.cast`  
204 maps them to  $(0 : R)$ ,  $(-1 : R)$ , and  $(1 : R)$ , respectively.

## 75:6 Composition Direction of Seymour's Theorem for Regular Matroids

205 Note that the indexing functions `f` and `g` are required to be injective in the definition,  
206 but this condition can be lifted. Indeed, lemma `Matrix.isTotallyUnimodular_iff` shows  
207 that one can equivalently check the determinants of all finite square submatrices, not just  
208 ones without repeated rows and columns.

209 Note that the definition of total unimodularity and lemma `Matrix.isTotallyUnimodular_iff`  
210 have been incorporated in `Mathlib`.

211 Keep in mind that the determinant is computed over  $R$ , so for certain commutative rings,  
212 all matrices are trivially totally unimodular, for example, for  $R = \mathbb{Z}_3$ .

### 213 4.3 Types and Subsets

214 In our project, we often have the following terms in the context:

```
( $\alpha$  : Type) (E : Set  $\alpha$ ) (I : Set  $\alpha$ ) (hIE : I  $\subseteq$  E)
```

215 Depending on the situation, there are three ways we may treat the set `I`. First, it may be  
216 viewed as a set of elements of type  $\alpha$ , its original type, so we simply write `I`. Second, we  
217 may need to re-type `I` as a set of elements of the type `E.Elem`. Then we write `E  $\downarrow$   $\cap$  I` using  
218 notation from `Mathlib`. Finally, `I` may be used as a set of elements of the type `I.Elem`. In  
219 this case, we write `Set.univ` of the correct type, which is usually inferred from the context.

### 220 4.4 Block Matrices

221 In this project, we often construct matrices by composing them from blocks using the  
222 following `Mathlib` definitions:

```
223 ■ Matrix.fromRows A1 A2 constructs 

|                |
|----------------|
| A <sub>1</sub> |
| A <sub>2</sub> |


```

```
224 ■ Matrix.fromCols A1 A2 constructs 

|                |                |
|----------------|----------------|
| A <sub>1</sub> | A <sub>2</sub> |
|----------------|----------------|


```

```
225 ■ Matrix.fromBlocks A11 A12 A21 A22 constructs 

|                 |                 |
|-----------------|-----------------|
| A <sub>11</sub> | A <sub>12</sub> |
| A <sub>21</sub> | A <sub>22</sub> |


```

## 226 5 Re-typing Matrix Dimensions

227 When constructing matroids, we often need to convert a block matrix whose blocks are  
228 indexed by disjoint sets into a matrix indexed by unions of those index sets. Although the  
229 contents of the matrix stay the same, both its dimensions change their type from a `Sum` of  
230 sets to a `Set` union of those sets. To this end, we implemented

```
def Subtype.toSum { $\alpha$  : Type*} {X Y : Set  $\alpha$ }  
  [ $\forall$  a, Decidable (a  $\in$  X)] [ $\forall$  a, Decidable (a  $\in$  Y)]  
  (i : (X  $\cup$  Y).Elem) : X.Elem  $\oplus$  Y.Elem :=  
  if hiX : i.val  $\in$  X then Sum.inl <i, hiX> else  
  if hiY : i.val  $\in$  Y then Sum.inr <i, hiY> else  
  (i.property.elim hiX hiY).elim
```

231 This allows us to re-type matrix dimensions and thus define the matrix transformation  
232 `Matrix.toMatrixUnionUnion` so that `A.toMatrixUnionUnion i j = A i.toSum j.toSum`.

233 We also define a function `Matrix.toMatrixElemElem` for convenience, but it is not a part  
234 of the trusted code.

## 235 6 Vector Matroids

236 Vector matroids [9, 12] is the most fundamental matroid class formalized in our work, serving  
 237 as the basis for binary and regular matroids in later sections. A *vector matroid* is constructed  
 238 from a matrix  $A$  by taking the column index set as the ground set and declaring a set  $I$  to  
 239 be independent if the set of columns of  $A$  indexed by  $I$  is linearly independent. To this end,  
 240 we implemented the definition

```
def Matrix.toMatroid {α R : Type*} {X Y : Set α} [DivisionRing R] (A : Matrix X Y R) :
  Matroid α := sorry
```

241 Note that although linear independence is defined over semirings  $R$ , in the definition above  
 242 we need to assume that  $R$  is a `DivisionRing`, otherwise the resulting structure would not  
 243 satisfy the augmentation property of a matroid.

## 244 7 Standard Representations

245 The *standard representation* [9, 12] of a vector matroid is the following structure:

```
structure StandardRepr (α R : Type*)
  [DecidableEq α] where
  X : Set α
  Y : Set α
  hXY : Disjoint X Y
  B : Matrix X Y R
  decmemX : ∀ a, Decidable (a ∈ X)
  decmemY : ∀ a, Decidable (a ∈ Y)
```

246 In essence, this is a wrapper for the standard representation matrix  $B$  indexed by disjoint sets  
 247  $X$  and  $Y$ , bundled together with the membership decidability for  $X$  and  $Y$ . The standard  
 248 representation matrix  $B$  corresponds to the full representation matrix  $\begin{bmatrix} \mathbb{1} & B \end{bmatrix}$  with the  
 249 conversion implemented as

```
def StandardRepr.toFull {α R : Type*} [DecidableEq α] [Zero R] [One R]
  (S : StandardRepr α R) : Matrix S.X (S.X ∪ S.Y).Elem R :=
  ((Matrix.fromCols 1 S.B) · ◦ Subtype.toSum)
```

250 Thus, the vector matroid given by its standard representation is constructed as follows:

```
def StandardRepr.toMatroid {α R : Type*} [DecidableEq α] [DivisionRing R]
  (S : StandardRepr α R) : Matroid α :=
  S.toFull.toMatroid
```

251 In this matroid, the ground set is  $X \cup Y$ , and a set  $I \subseteq X \cup Y$  is independent if the columns  
 252 of  $\begin{bmatrix} \mathbb{1} & B \end{bmatrix}$  indexed by  $I$  are linearly independent over  $R$ .

253 Below are several results we prove about standard representations, which are either used  
 254 in the proof of regularity of 1-, 2-, and 3-sums, or could be useful for downstream projects.

255 First, we show that if the row index set  $X$  of a standard representation is finite, then  $X$   
 256 is a base in the resulting matroid:

```
lemma StandardRepr.toMatroid_isBase_X
  {α R : Type*} [DecidableEq α] [Field R]
  (S : StandardRepr α R) [Fintype S.X] : S.toMatroid.IsBase S.X
```

## 75:8 Composition Direction of Seymour's Theorem for Regular Matroids

257 This lemma characterizes what sets can serve as row index sets of standard representations  
 258 and motivates the corresponding hypotheses in the code snippets below.

259 Next, we prove that a full representation of a vector matroid can be transformed into a  
 260 standard representation of the same matroid, with a given base as the row index set:

```
lemma Matrix.exists_standardRepr_isBase
  {α R : Type*} [DecidableEq α] [DivisionRing R]
  {X Y G : Set α} (A : Matrix X Y R) (hAG : A.toMatroid.IsBase G) :
  ∃ S : StandardRepr α R, S.X = G ∧ S.toMatroid = A.toMatroid
```

261 In classical literature on matroid theory [9, 12], this follows by simply performing a sequence  
 262 of elementary row operations akin to Gaussian elimination. Our formal proof used a different  
 263 approach, utilizing Mathlib's results about bases and linear independence. First, we showed  
 264 that the columns indexed by  $G$  form a basis of the module generated by all columns of  $A$ .  
 265 Then we proved that performing a basis exchange yields the correct standard representation  
 266 matrix.

267 We also prove an analog of the above lemma that additionally preserves total unimodularity  
 268 of the representation matrix:

```
lemma Matrix.exists_standardRepr_isBase_isTU
  {α R : Type*} [DecidableEq α] [Field R]
  {X Y G : Set α} [Fintype G] (A : Matrix X Y R)
  (hAG : A.toMatroid.IsBase G) (hA : A.IsTotallyUnimodular) :
  ∃ S : StandardRepr α R, S.X = G ∧ S.toMatroid = A.toMatroid ∧ S.B.IsTotallyUnimodular
```

269 Classical literature [9, 12] observes that elementary row operations preserve total unimodu-  
 270 larity and then simply refers to the proof of the previous lemma. Unfortunately, we could not  
 271 take advantage of such a reduction, as it would be hard to verify that total unimodularity  
 272 is preserved in our prior approach. Thus, we implemented an inductive proof essentially  
 273 following the ideas of [9, 12]. Note that this lemma takes stronger assumptions than the  
 274 previous one, namely  $G$  has to be finite and multiplication in  $R$  has to commute.

275 Another result we prove is that two standard representations of the same vector matroid  
 276 over  $\mathbb{Z}_2$  with the same finite row index set must be identical:

```
lemma ext_standardRepr_of_same_matroid_same_X
  {α : Type*} [DecidableEq α]
  {S1 S2 : StandardRepr α Z2} [Fintype S1.X]
  (hSS : S1.toMatroid = S2.toMatroid) (hXX : S1.X = S2.X) : S1 = S2
```

277 Although this particular lemma is not employed later in our project, it captures an important  
 278 result that a binary matroid has an essentially unique standard representation [9, 12].  
 279 Nevertheless, we make use of a very similar result:

```
lemma support_eq_support_of_same_matroid_same_X
  {F1 : Type u1} {F2 : Type u2}
  {α : Type max u1 u2 v} [DecidableEq α]
  [DecidableEq F1] [DecidableEq F2]
  [Field F1] [Field F2]
  {S1 : StandardRepr α F1}
  {S2 : StandardRepr α F2}
  [Fintype S2.X]
  (hSS : S1.toMatroid = S2.toMatroid) (hXX : S1.X = S2.X) :
```

```

let hYY : S1.Y = S2.Y := sorry
hXX ▶ hYY ▶ S1.B.support = S2.B.support

```

280 This states that two standard representations of a vector matroid with identical (finite)  
 281 row index sets have the same support, i.e., the zeros in them appear on identical positions.  
 282 Crucially, this holds for any two standard representations over any two fields (where equality  
 283 is decidable), and we later use it for  $\mathbb{Q}$  and  $\mathbb{Z}_2$ .

## 284 8 Regular Matroids

285 Regular matroids [9, 12] are the core subject of Seymour's theorem. A matroid is *regular* if  
 286 it can be constructed (as a vector matroid) from a rational totally unimodular matrix:

```

def Matroid.IsRegular {α : Type*} (M : Matroid α) : Prop :=
  ∃ X Y : Set α, ∃ A : Matrix X Y ℚ, A.IsTotallyUnimodular ∧ A.toMatroid = M

```

287 One key result we prove is that every regular matroid is in fact *binary*, i.e., can be constructed  
 288 from a binary matrix:

```

lemma Matroid.IsRegular.isBinary
  {α : Type*} [DecidableEq α]
  {M : Matroid α} (hM : M.IsRegular) :
  ∃ X : Set α, ∃ Y : Set α, ∃ A : Matrix X Y ℤ2, A.toMatroid = M

```

289 Another important lemma we prove about regular matroids is their equivalent characterization  
 290 in terms of totally unimodular signings. First, let us introduce the necessary definitions. We  
 291 say that a matrix  $A$  is a *signing* of matrix  $U$  if their values are identical up to signs:

```

def Matrix.IsSigningOf {X Y R : Type*} [LinearOrderedRing R] {n : ℕ}
  (A : Matrix X Y R) (U : Matrix X Y (ℤMod n)) : Prop :=
  ∀ i : X, ∀ j : Y, |A i j| = (U i j).val

```

292 We then say that a binary matrix  $U$  has a *totally unimodular signing* if it has a signing  
 293 matrix  $A$  that is rational and totally unimodular:

```

def Matrix.IsTuSigningOf {X Y : Type*} (A : Matrix X Y ℚ) (U : Matrix X Y ℤ2) : Prop :=
  A.IsTotallyUnimodular ∧ A.IsSigningOf U

```

```

def Matrix.HasTuSigning {X Y : Type*} (U : Matrix X Y ℤ2) : Prop :=
  ∃ A : Matrix X Y ℚ, A.IsTuSigningOf U

```

294 Now, we can state the characterization: given a standard representation over  $\mathbb{Z}_2$ , its matrix  
 295 has a totally unimodular signing if and only if the matroid obtained from the representation  
 296 is regular.

```

lemma StandardRepr.toMatroid_isRegular_iff_hasTuSigning {α : Type*} [DecidableEq α]
  (S : StandardRepr α ℤ2) [Finite S.X] : S.toMatroid.IsRegular ↔ S.B.HasTuSigning

```

297 Out of all definitions in this section, only `Matroid.IsRegular` is a part of the trusted code.

## 298 9 The 1-Sum

299 Let  $B_\ell \in \mathbb{Z}_2^{X_\ell \times Y_\ell}$  and  $B_r \in \mathbb{Z}_2^{X_r \times Y_r}$  be standard representation matrices where  $X_\ell, Y_\ell, X_r, Y_r$   
 300 are pairwise disjoint sets. The 1-sum  $B = B_\ell \oplus_1 B_r$  of  $B_\ell$  and  $B_r$  is

$$301 \quad B = \begin{bmatrix} B_\ell & 0 \\ 0 & B_r \end{bmatrix} \in \mathbb{Z}_2^{(X_\ell \cup X_r) \times (Y_\ell \cup Y_r)}.$$

302 A matroid  $M$  is a 1-sum of matroids  $M_\ell$  and  $M_r$  if there exist standard  $\mathbb{Z}_2$  representation  
 303 matrices  $B_\ell, B_r$ , and  $B$  (for  $M_\ell, M_r$ , and  $M$ , respectively) of the form above.

304 All matroid sums are implemented on three levels: the **Matrix** level, the **StandardRepr**  
 305 level, and the **Matroid** level. The **Matrix** level defines the standard representation matrix of  
 306 the output matroid:

```
def matrixSum1 {R : Type*} [Zero R] {Xℓ Yℓ Xr Yr : Type*}
  (Aℓ : Matrix Xℓ Yℓ R) (Ar : Matrix Xr Yr R) : Matrix (Xℓ ⊕ Xr) (Yℓ ⊕ Yr) R :=
  Matrix.fromBlocks Aℓ 0 0 Ar
```

307 The **StandardRepr** level converts the output matrix indices from **Sum** types to set unions,  
 308 provides a proof that the resulting row and column index sets are disjoint, and checks whether  
 309 the operation is valid—returning **none** if preconditions are not met. The **Matroid** level  
 310 defines a predicate—when  $M$  is a 1-sum of  $M_\ell$  and  $M_r$ :

```
def Matroid.IsSum1of {α : Type*} [DecidableEq α]
  (M : Matroid α) (Mℓ Mr : Matroid α) : Prop :=
  ∃ S Sℓ Sr : StandardRepr α Z2,
  ∃ hXY : Disjoint Sℓ.X Sr.Y,
  ∃ hYX : Disjoint Sℓ.Y Sr.X,
  standardReprSum1 hXY hYX = some S
  ∧ S.toMatroid = M
  ∧ Sℓ.toMatroid = Mℓ
  ∧ Sr.toMatroid = Mr
```

311 In addition to basic API about the 1-sum, we also provide a theorem `Matroid.IsSum1of.eq_disjointSum`  
 312 that establishes the equality between the disjoint sum (defined in `Mathlib`) and the 1-sum  
 313 (defined in our project) of binary matroids.

## 314 10 The 2-Sum

315 Let  $B_\ell \in \mathbb{Z}_2^{X_\ell \times Y_\ell}$  and  $B_r \in \mathbb{Z}_2^{X_r \times Y_r}$  be standard representation matrices where  $X_\ell \cap X_r = \{x\}$ ,  
 316  $Y_\ell \cap Y_r = \{y\}$ , and  $X_\ell$  is disjoint with  $Y_r$  and  $X_r$  is disjoint with  $Y_\ell$ . Let  $A_\ell = B_\ell(X_\ell \setminus \{x\}, Y_\ell)$ ,  
 317  $A_r = B_r(X_r, Y_r \setminus \{y\})$ ,  $r = B_\ell(x, Y_\ell) \neq 0$ , and  $c = B_r(X_r, y) \neq 0$ . The 2-sum  $B = B_\ell \oplus_2 B_r$   
 318 is defined as

$$319 \quad B = \begin{bmatrix} A_\ell & 0 \\ c \otimes r & A_r \end{bmatrix}.$$

320 A matroid  $M$  is a 2-sum of matroids  $M_\ell$  and  $M_r$  if there exist standard  $\mathbb{Z}_2$  representation  
 321 matrices  $B_\ell, B_r$ , and  $B$  (for  $M_\ell, M_r$ , and  $M$ , respectively) of the form above.

322 The implementation of the 2-sum follows the same three-level structure as the 1-sum.  
 323 The **Matrix** level places the two given matrices along the main diagonal of the resulting block  
 324 matrix, with the bottom-left block containing the outer product of the two given vectors:

```

def matrixSum2 {R : Type*} [Semiring R] {Xℓ Yℓ Xr Yr : Type*}
  (Aℓ : Matrix Xℓ Yℓ R) (r : Yℓ → R) (Ar : Matrix Xr Yr R) (c : Xr → R) :
  Matrix (Xℓ ⊕ Xr) (Yℓ ⊕ Yr) R :=
  Matrix.fromBlocks
    Aℓ 0 (fun i j => c i * r j) Ar

```

325 The `StandardRepr` level first slices the last row of  $S_\ell \cdot B$  and the first column of  $S_r \cdot B$  as the  
 326 two separate vectors ( $r$  and  $c$ ), naming the two remaining matrices  $A_\ell$  and  $A_r$ . To identify  
 327 the special row and column, we need a specific element  $x$  in  $S_\ell \cdot X \cap S_r \cdot X$  and a specific  
 328 element  $y$  in  $S_\ell \cdot Y \cap S_r \cdot Y$  with no other element in any pairwise intersection among the four  
 329 indexing sets. The following picture shows how  $S_\ell \cdot B$  and  $S_r \cdot B$  are taken apart:

330 
$$S_\ell \cdot B = \begin{array}{|c|} \hline A_\ell \\ \hline r \\ \hline \end{array}, \quad S_r \cdot B = \begin{array}{|c|c|} \hline c & A_r \\ \hline \end{array}$$

331 The `Matroid` level is again a predicate—when  $M$  is a 2-sum of  $M_\ell$  and  $M_r$ :

```

def Matroid.IsSum2of {α : Type*} [DecidableEq α]
  (M : Matroid α) (Mℓ Mr : Matroid α) : Prop :=
  ∃ S Sℓ Sr : StandardRepr α Z2,
  ∃ x y : α,
  ∃ hXX : Sℓ.X ∩ Sr.X = {x},
  ∃ hYY : Sℓ.Y ∩ Sr.Y = {y},
  ∃ hXY : Disjoint Sℓ.X Sr.Y,
  ∃ hYX : Disjoint Sℓ.Y Sr.X,
  standardReprSum2 hXX hYY hXY hYX = some S
  ∧ S.toMatroid = M
  ∧ Sℓ.toMatroid = Mℓ
  ∧ Sr.toMatroid = Mr

```

## 332 11 The 3-Sum

333 The 3-sum of binary matroids is defined as follows. Let  $X_\ell, Y_\ell, X_r,$  and  $Y_r$  be sets with the  
 334 following properties:

- 335 ■  $X_\ell \cap X_r = \{x_2, x_1, x_0\}$  for some distinct  $x_0, x_1,$  and  $x_2$
- 336 ■  $Y_\ell \cap Y_r = \{y_0, y_1, y_2\}$  for some distinct  $y_0, y_1,$  and  $y_2$
- 337 ■  $X_\ell \cap Y_\ell = X_\ell \cap Y_r = X_r \cap Y_\ell = X_r \cap Y_r = \emptyset$
- 338 Let  $B_\ell \in \mathbb{Z}_2^{X_\ell \times Y_\ell}$  and  $B_r \in \mathbb{Z}_2^{X_r \times Y_r}$  be matrices of the form

339 
$$B_\ell = \begin{array}{|c|c|c|c|} \hline & & & 0 \\ \hline & A_\ell & & \\ \hline & & 1 & 1 & 0 \\ \hline D_\ell & D_0 & & 1 \\ \hline & & & 1 \\ \hline \end{array}, \quad B_r = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 0 \\ \hline D_0 & 1 & & \\ \hline & 1 & & A_r \\ \hline D_r & & & \\ \hline \end{array}$$

## 75:12 Composition Direction of Seymour's Theorem for Regular Matroids

340 where  $D_0$  is invertible. Then the 3-sum  $B = B_\ell \oplus_3 B_r$  is

$$341 \quad B = \begin{array}{|c|c|c|} \hline & A_\ell & 0 \\ \hline & \begin{array}{|c|c|c|} \hline 1 & 1 & 0 \\ \hline \end{array} & \\ \hline D_\ell & D_0 & \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array} \\ \hline D_{\ell r} & D_r & A_r \\ \hline \end{array} \quad \text{where } D_{\ell r} = D_r \cdot D_0^{-1} \cdot D_\ell$$

342 Here  $D_0 \in \mathbb{Z}_2^{\{x_0, x_1\} \times \{y_0, y_1\}}$ ,  $\begin{array}{|c|c|c|} \hline 1 & 1 & 0 \\ \hline D_0 & 1 & \\ \hline & 1 & \\ \hline \end{array} \in \mathbb{Z}_2^{\{x_2, x_0, x_1\} \times \{y_0, y_1, y_2\}}$ , and the indexing is kept

343 consistent between  $B_\ell$ ,  $B_r$ , and  $B$ . A matroid  $M$  is a 3-sum of matroids  $M_\ell$  and  $M_r$  if they  
 344 admit standard representations over  $\mathbb{Z}_2$  with matrices  $B$ ,  $B_\ell$ , and  $B_r$  of the form above. The  
 345 Matroid-level predicate `Matroid.IsSum3of` is defined similarly to those for 1- and 2-sums.

### 346 **12** Sums Preserve Regularity

347 In our library, the final theorems that regularity is preserved under 1-, 2-, and 3-sums are  
 348 stated as follows.

**theorem** `Matroid.IsSum1of.isRegular`  $\{\alpha : \text{Type}^*\}$  `[DecidableEq α] {M Mℓ Mr : Matroid α} :`  
`M.IsSum1of Mℓ Mr → M.RankFinite → Mℓ.IsRegular → Mr.IsRegular → M.IsRegular`

**theorem** `Matroid.IsSum2of.isRegular`  $\{\alpha : \text{Type}^*\}$  `[DecidableEq α] {M Mℓ Mr : Matroid α} :`  
`M.IsSum2of Mℓ Mr → M.RankFinite → Mℓ.IsRegular → Mr.IsRegular → M.IsRegular`

**theorem** `Matroid.IsSum3of.isRegular`  $\{\alpha : \text{Type}^*\}$  `[DecidableEq α] {M Mℓ Mr : Matroid α} :`  
`M.IsSum3of Mℓ Mr → M.RankFinite → Mℓ.IsRegular → Mr.IsRegular → M.IsRegular`

349 Note that these three theorems are stated for matroids and have the same interface. Moreover,  
 350 when applying one of these results, a user is able to provide different representations for  
 351 witnessing that  $M$  is a 1-, 2-, or 3-sum of  $M_\ell$  and  $M_r$ , for witnessing that  $M$  has finite rank,  
 352 and for witnessing that  $M_\ell$  and  $M_r$  are regular.

353 We split the proof of each of these theorems into three stages corresponding to the three  
 354 abstraction layers used for the definitions: `Matroid`, `StandardRepr`, and `Matrix`.

355 The final Matroid-level theorems are reduced to the respective lemmas for standard  
 356 representations by applying `StandardRepr.toMatroid_isRegular_iff_hasTuSigning` and  
 357 `StandardRepr.finite_X_of_toMatroid_rankFinite` in all three proofs (for the 1-, 2-, and  
 358 3-sums). The reductions from the `StandardRepr` level to the `Matrix` level for 1- and 2-sums  
 359 is straightforward—plug the standard representation matrices and their (rational) signings  
 360 into `matrixSum1` and `matrixSum2`, respectively. For 3-sums, this reduction is more involved,  
 361 as we additionally apply the following lemma to simplify the assumption on  $D_0$ :

**lemma** `Matrix.isUnit_2x2`  $(A : \text{Matrix} (\text{Fin } 2) (\text{Fin } 2) \mathbb{Z}_2) (hA : \text{IsUnit } A) :  
`∃ f : Fin 2 ≃ Fin 2, ∃ g : Fin 2 ≃ Fin 2,`  
`A.submatrix f g = 1 ∨ A.submatrix f g = !![1, 1; 0, 1]`$

362 Therefore, up to reindexing,  $D_0$  is either  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  or  $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ . Performing the reduction at this  
 363 stage allows us to invoke `Matrix.isUnit_2x2` only once and then simply consider the two  
 364 special forms of  $D_0$ .

365 On the `Matrix` level, our formal proof that 1-sums preserve total unimodularity of matrices  
 366 is nearly identical to [12]. For 2-sums, we streamlined the proof by reformulating it as a  
 367 forward argument by induction. For 3-sums, the entire argument was significantly reworked  
 368 to simplify and streamline the approach of [12]. On a high level, we make two major changes,  
 369 which we discuss in detail below.

370 The first key difference is that we re-sign the summands only once, rather than multiple  
 371 times. Like in [12], we start with totally unimodular signings exhibiting regularity of the two  
 372 summands. Then we multiply their rows and columns by  $\pm 1$  factors (which preserves total

373 unimodularity) so that the submatrix 

1	1	0
$D_0$		1
		1

 is signed in both summands simultaneously

374 as either  $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix}$  or  $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ , depending on whether  $D_0$  is  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  or  $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ . Thus,

375 we get totally unimodular signings of the summands that coincide on the intersection, which  
 376 allows us to define the *canonical* signing of the entire 3-sum: use the same signs as in  
 377 the re-signed summands everywhere except for the bottom-left block, which is signed via  
 378  $D'_{\ell_r} = D'_r \cdot (D'_0)^{-1} \cdot D'_\ell$ , and the 0 block, which remains as is.

379 The main advantage of our approach is that we avoid chained constructions and proofs of  
 380 properties of such constructions, and we do not need to define  $\Delta$ -sums. Moreover, unlike [12],  
 381 our proof does not rely on the general lemma about re-signing totally unimodular matrices.  
 382 This detail is crucial, as the proof of this lemma in [12] involves a graph-theoretic argument,  
 383 which would be very challenging to formalize in Lean with the current tools available in  
 384 Mathlib.

385 The second major difference from the approach of [12] is that our main argument does  
 386 not deal with signings of 3-sums directly. Instead, we work with a matrix family called  
 387 `MatrixLikeSum3` in our code. This allows us to split the proof of regularity of 3-sums into  
 388 three clear steps. First, we show that pivoting on a non-zero entry in the top-left block of any  
 389 matrix from this family produces a matrix that also belongs to this family. Next, we utilize  
 390 the result from the first step to prove that every matrix in this family is totally unimodular.  
 391 We do this via a similar argument to the proof that 2-sums of totally unimodular matrices  
 392 are totally unimodular. Finally, we show that every canonical signing of the 3-sum matrix  
 393 defined above is included in this matrix family and is thus totally unimodular. Overall, this  
 394 proof takes a more systematic approach to deriving properties of signings of 3-sums and  
 395 using them to prove their total unimodularity. Additionally, it conveniently reuses a large  
 396 portion of the argument for 2-sums.

397 In some proofs, we worked with large case splits with up to 896 cases. To handle such  
 398 situations, we used `all_goals try` followed by one or more tactics, discharging multiple  
 399 goals at once without selecting them by hand or repeating the proof. We repeatedly applied  
 400 this method to discharge the remaining goals in waves until the proof was complete.

401 **13** Related Work

402 In Lean 4, the largest library formalizing matroid theory is due to Peter Nelson<sup>3</sup>. It  
 403 implements infinite matroids following [3] together with many key notions and results  
 404 about them. The definition that is fully formalized and is the most related to our work  
 405 is `Matroid.disjointSum`. For binary matroids, this definition is equivalent to the 1-sum  
 406 implemented in this paper. Moreover, it can be used for any matroids with disjoint ground  
 407 sets, while our implementation is restricted to vector matroids constructed from  $\mathbb{Z}_2$  matrices.  
 408 Peter Nelson’s repository also makes progress towards formalizing other related notions, such  
 409 as representable matroids, though this work is still ongoing. It is also worth noting that the  
 410 results in Mathlib<sup>4</sup> have been copied over from this repository and comprise a strict subset  
 411 of it.

412 Building upon Peter Nelson’s work, Gusakov2024’s thesis [5] formalizes the proof of  
 413 Tutte’s excluded minor theorem and to this end implements definitions and results about  
 414 representable matroids. The thesis formalizes representations and standard representations  
 415 of matroids, which we also do in our work, but it takes a different approach. In particular,  
 416 instead of working with matrix representations, the thesis implements a representation of  
 417 `Matroid`  $\alpha$  as a mapping from the entire type  $\alpha$  to a vector space, which maps non-elements  
 418 of the matroid to the zero vector and independent sets to linearly independent vectors. The  
 419 advantage of this approach is that certain proofs become easier to formalize, but this comes  
 420 at a cost of making it harder to match the implementation with the theory and believe the  
 421 correctness of the code.

422 There are also two Lean 3 repositories due to Artem Vasilyev<sup>5</sup> and Bryan Gin-ge Chen<sup>6</sup>  
 423 dedicated to formalization of matroid theory. Both of them work with finite matroids following  
 424 [9] and implement basic definitions and properties of matroids concerning circuits, bases, and  
 425 rank functions. These results are completely subsumed by the current implementation of  
 426 matroids in Mathlib.

427 Jonas Keinholtz [6] formalizes the classical definition of (finite) matroids [9, 12] in Isa-  
 428 belle/HOL along with other basic ideas such as minors, bases, circuits, rank, and closure.  
 429 More recently, Wan2025 use Keinholtz’s formalization to design a verification framework using  
 430 a Locale that checks if a given collection of subsets of a given set is a matroid. The authors  
 431 then showcase the verification algorithm by checking that the 0-1 knapsack problem does not  
 432 conform to the matroid structure, while the fractional knapsack problem does. In comparison,  
 433 Lean 4’s Mathlib implements a more general definition of matroids and formalizes more  
 434 results about them than either Matroids-AFP or Wan2025, but Lean lacks a procedure for  
 435 formally verifying if a collection of sets has matroid structure.

436 In the HOL Light GitHub repository<sup>7</sup>, John Harrison formalizes finitary matroids. The  
 437 formalization closely follows the field theory notes of Pete L. Clark<sup>8</sup>. In particular, finitary  
 438 matroids are defined in terms of a closure operator with similar properties as those proposed  
 439 in [3]. This repository also includes a formal proof that this notion of (finitary) matroids is  
 440 equivalent to the definition of a matroid using independent sets. Unlike Lean 4’s Mathlib

---

<sup>3</sup> <https://github.com/apnelson1/lean-matroids>

<sup>4</sup> <https://github.com/leanprover-community/mathlib4/tree/master/Mathlib/Combinatorics/Matroid>

<sup>5</sup> <https://github.com/VArtem/lean-matroids>

<sup>6</sup> <https://github.com/bryangingechen/lean-matroids>

<sup>7</sup> <https://github.com/jrh13/hol-light/blob/master/Library/matroids.ml>

<sup>8</sup> <https://plclark.github.io/PeteLClark/Expositions/FieldTheory.pdf>

441 formalization (which includes formalizations of the closure operator and the notions of  
 442 spanning sets), however, this notion of infinite matroids does not respect the notion of duality  
 443 that is defined for matroids in [9, 12] as noted by Bruhn2013.

444 Grzegorz Bancerek and Yasunari Shidama [1] formalize matroids in Mizar. Their formal-  
 445 ization includes basic notions like rank, basis, and cycle as well as examples like the matroid  
 446 of linearly independent subsets for a given vector space. Overall, the scope of the Mizar  
 447 formalization is comparable to the Isabelle/HOL formalization, except that the Mizar form-  
 448 alization allows for infinite matroids. In this sense, it is comparable to the Lean definition in  
 449 Mathlib, which also allows for infinite matroids. However, whereas Mizar uses independence  
 450 axioms to define matroids, Lean uses base axioms for the main definition and provides an  
 451 API for constructing matroids via independence axioms.

## 452 14 Conclusion

453 In this work, we formally stated Seymour’s decomposition theorem for regular matroids and  
 454 implemented a formally verified proof of the forward (composition) direction of this theorem  
 455 in the setting where the matroids have finite rank and may have infinite ground sets. To  
 456 this end, we developed a modular and extensible library in Lean 4 formalizing definitions  
 457 and lemmas about totally unimodular matrices, vector matroids, regular matroids, and 1-  
 458 2-, and 3-sums of matrices, standard representations of vector matroids, and matroids. Our  
 459 work demonstrates that one can effectively use Lean and Mathlib to formally verify advanced  
 460 results from matroid theory and extend classical results to a more general setting.

461 Formalizing Seymour’s theorem presented several challenges. First, the limited matroid  
 462 theory in Mathlib meant we had to develop many fundamentals from scratch (e.g. represent-  
 463 ability and regularity definitions). We addressed this by introducing a *StandardRepr* structure  
 464 to bridge matrices and matroids, enabling us to work around the absence of a general matroid  
 465 representation theory. Moreover, some proofs required managing enormous case splits (our  
 466 3-sum proof involved up to 896 subcases). We tackled this with structured automation - for  
 467 instance, using `all_goals try` tactics to discharge many cases at once - thereby keeping  
 468 the proof tractable in Lean. We also avoided certain combinatorial arguments (such as the  
 469 graph-theoretic re-signing lemma from [12]) that would be cumbersome to formalize, opting  
 470 for alternative approaches better suited to Lean.

471 The most natural continuation of our project is proving the decomposition direction of  
 472 Seymour’s theorem, stated as `Matroid.IsRegular.isGood` in our library. Our work can also  
 473 serve as a starting point for formalizing Seymour’s theorem for matroids of infinite rank [2].

## 474 ——— References ———

- 475 1 Grzegorz Bancerek and Yasunari Shidama. Introduction to matroids. *Formalized Mathematics*,  
 476 16(4):325–332, 2008.
- 477 2 Nathan Bowler and Johannes Carmesin. The ubiquity of psi-matroids, 2013.
- 478 3 Henning Bruhn, Reinhard Diestel, Matthias Kriesell, Rudi Pendavingh, and Paul Wollan.  
 479 Axioms for infinite matroids, 2013.
- 480 4 Jim Geelen and Bert Gerards. Regular matroid decomposition via signed graphs. *Journal of*  
 481 *Graph Theory*, 48(1):74–84, 2005.
- 482 5 Alena Gusakov. Formalizing the excluded minor characterization of binary matroids in the  
 483 lean theorem prover. Master’s thesis, University of Waterloo, 2024.
- 484 6 Jonas Keinholtz. Matroids. *Archive of Formal Proofs*, November 2018. [https://isa-afp.org/](https://isa-afp.org/entries/Matroids.html)  
 485 [entries/Matroids.html](https://isa-afp.org/entries/Matroids.html), Formal proof development.

## 75:16 Composition Direction of Seymour's Theorem for Regular Matroids

- 486 7 Sandra R. Kingan. On seymour's decomposition theorem. *Annals of Combinatorics*, 19(1):171–  
487 185, Mar 2015.
- 488 8 The mathlib community. The Lean Mathematical Library. In Jasmin Blanchette and Cătălin  
489 Hritcu, editors, *CPP 2020*, pages 367–381. ACM, 2020.
- 490 9 James Oxley. *Matroid Theory*. Oxford University Press, Oxford, 02 2011.
- 491 10 Robert Pollack. How to believe a machine-checked proof. *BRICS Report Series*, 4(18), January  
492 1997.
- 493 11 Moses Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*,  
494 92:305–316, 1924.
- 495 12 Klaus Truemper. *Matroid Decomposition*. Leibniz Company, 2016.