Yes

Dillon Mayhew, Mike Newman, Geoff Whittle
University of Ottawa, Victoria University Wellington

Third Computational Matroid Workshop
La Vacquerie, June 2013

Yes

"The missing axiom of matroid theory is lost forever" (Vámos)

Dillon Mayhew, Mike Newman, Geoff Whittle
University of Ottawa, Victoria University Wellington

Third Computational Matroid Workshop
La Vacquerie, June 2013

Yes

"The missing axiom of matroid theory is lost forever" (Vámos)
"Is the missing axiom of matroid theory lost forever?" (MNW)

Dillon Mayhew, Mike Newman, Geoff Whittle
University of Ottawa, Victoria University Wellington

Third Computational Matroid Workshop
La Vacquerie, June 2013

motivation

Whitney . . .

motivation

Whitney . . .

Matroids are systems of subsets of a finite set.

$\mathbb{R}$ (a.k.a. infinite fields)

How to capture real representability?

$\mathbb{R}$ (a.k.a. infinite fields)

How to capture real representability?

- there exists a matrix . . .

$\mathbb{R}$ (a.k.a. infinite fields)

How to capture real representability?

- there exists a matrix . . .
- ??

$\mathbb{F}_2$ (a.k.a. finite fields)

How to capture binary representability?

$\mathbb{F}_2$ (a.k.a. finite fields)

How to capture binary representability?

- there exists a matrix . . .

$\mathbb{F}_2$ (a.k.a. finite fields)

How to capture binary representability?

- there exists a matrix . . .
- there exists no $U_{2,4}$ minor

$\mathbb{F}_2$ (a.k.a. finite fields)

How to capture binary representability?

- there exists a matrix . . .
- there exists no $U_{2,4}$ minor
- It is not the case that there exists an independent set Y and four points a, b, c, d not in Y such that $Y \cup T$ is independent if $|T| \leq 2$ and dependent if $|T| \geq 3$.

language

We define a language with the following ingredients

- variables X, Y, Z, . . . , all representing subsets of E
- a unary predicate ind with $\mathrm{ind}(X)$ true if X is independent
- a binary relation $\subseteq$ with $X \subseteq Y$ true if X is a subset of Y
- $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, $\rightarrow$

language

We define a language with the following ingredients

- variables X, Y, Z, . . . , all representing subsets of E
- a unary predicate ind with $\mathrm{ind}(X)$ true if X is independent
- a unary predicate sing with $\mathrm{sing}(X)$ true if $|X| = 1$
- a binary relation $\subseteq$ with $X \subseteq Y$ true if X is a subset of Y
- $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, $\rightarrow$

language

We define a language with the following ingredients

- variables X, Y, Z, . . . , all representing subsets of E
- a unary predicate ind with $\mathrm{ind}(X)$ true if X is independent
- a unary predicate sing with $\mathrm{sing}(X)$ true if $|X| = 1$
- a binary relation $\subseteq$ with $X \subseteq Y$ true if X is a subset of Y
- $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, $\rightarrow$

We combine atomic formulae using connectives, to obtain a statement about some variables, then we quantify all variables to obtain an "axiom".

language

We define a language with the following ingredients

- variables X, Y, Z, ..., all representing subsets of E
- a unary predicate ind with $\mathrm{ind}(X)$ true if X is independent
- a unary predicate sing with $\mathrm{sing}(X)$ true if $|X| = 1$
- a binary relation $\subseteq$ with $X \subseteq Y$ true if X is a subset of Y
- $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, $\rightarrow$

We combine atomic formulae using connectives, to obtain a statement about some variables, then we quantify all variables to obtain an "axiom".

The language thus obtained is Monadic Second Order (MSO), with the given predicates/relations.

freebies

Various things we didn't include we get for free.
To get the set union $X \cup Y$, use Z where

$$X \subseteq Z \wedge Y \subseteq Z \wedge \forall T \{\mathsf{sing}(T) \wedge T \subseteq Z\} \rightarrow \{\vee T \subseteq X \vee T \subseteq Y\}$$

freebies

Various things we didn't include we get for free.
To get the set union $X \cup Y$, use Z where

$$X \subseteq Z \wedge Y \subseteq Z \wedge \forall T \{\mathsf{sing}(T) \wedge T \subseteq Z\} \rightarrow \{\vee T \subseteq X \vee T \subseteq Y\}$$

So $\forall X \exists Y \, \mathsf{ind}(X \cup Y)$ becomes

$$\forall X \exists Y \exists Z \, \mathsf{ind}(Z)$$
$$\wedge \left\{ X \subseteq Z \wedge Y \subseteq Z \wedge \forall T \{\mathsf{sing}(T) \wedge T \subseteq Z\} \rightarrow \{\vee T \subseteq X \vee T \subseteq Y\} \right\}$$

We can use union, intersection, etc, as if they were in our language (subroutines).

matroids

We can define matroids using this language.

matroids

We can define matroids using this language.

We check the basis exchange axiom.
First, a subroutine for "is a basis".

$$\mathsf{bas}(X) := I(X) \wedge \forall T \left\{ \mathsf{sing}(T) \wedge T \not\subseteq X \right\} \rightarrow \neg I(X \cup T)$$

Then, the axiom.

$$\begin{array}{l} \forall B_1 \forall B_2 \forall T_1 \\ \left\{ \mathsf{sing}(T_1) \wedge \mathsf{bas}(B_1) \wedge \mathsf{bas}(B_2) \wedge T_1 \subseteq B_1 \wedge T_1 \not\subseteq B_2 \right\} \\ \rightarrow \\ \exists T_2 \\ \left\{ \mathsf{sing}(T_2) \wedge T_2 \not\subseteq B_1 \wedge T_2 \subseteq B_2 \wedge \mathsf{bas}(B_1 - T_1 \cup T_2) \right\} \end{array}$$

minors

We can check whether a matroid has a fixed minor N. For instance, for $U_{2,4}$.

$$
\begin{aligned}
&\exists Y \exists A \exists B \exists C \exists D \\
&\operatorname{ind}(Y) \wedge \operatorname{sing}(A) \wedge \operatorname{sing}(B) \wedge \operatorname{sing}(C) \wedge \operatorname{sing}(D) \\
&\wedge A \not\subseteq Y \wedge B \not\subseteq Y \wedge C \not\subseteq Y \wedge D \not\subseteq Y \\
&\wedge \operatorname{ind}(Y \cup A \cup B) \wedge \operatorname{ind}(Y \cup A \cup C) \wedge \operatorname{ind}(Y \cup A \cup D) \\
&\wedge \operatorname{ind}(Y \cup B \cup C) \wedge \operatorname{ind}(Y \cup B \cup D) \wedge \operatorname{ind}(Y \cup C \cup D) \\
&\wedge \neg \operatorname{ind}(Y \cup A \cup B \cup C) \wedge \neg \operatorname{ind}(Y \cup A \cup B \cup D) \\
&\wedge \neg \operatorname{ind}(Y \cup A \cup C \cup D) \wedge \neg \operatorname{ind}(Y \cup B \cup C \cup D)
\end{aligned}
$$

goal

We would answer to show the following

Question (Whitney?)

Is there an MSO axiom that precisely captures real-representability?

goal

We would answer to show the following

Question (Whitney?)

Is there an MSO axiom that precisely captures real-representability?

Theorem (Mayhew, N, Whittle (201x))

There is no MSO axiom that precisely captures real-representability,

goal

We would answer to show the following

Question (Whitney?)

Is there an MSO axiom that precisely captures real-representability?

Theorem (Mayhew, N, Whittle (201x))

There is no MSO axiom that precisely captures real-representability, as long as you restrict to axioms that have at most one alternation of quantifiers such that all variables in the second block of quantification are singleton variables.

Ehrenfeucht-Fraïssé

Consider two players: Spoiler and Duplicator.

Ehrenfeucht-Fraïssé

Consider two players: Spoiler and Duplicator.

- Spoiler chooses an integer k; this will be the number of variables.

Ehrenfeucht-Fraïssé

Consider two players: Spoiler and Duplicator.

- Spoiler chooses an integer k; this will be the number of variables.
- Duplicator chooses a pair of objects (choice depends on k), where the first object is in the class.

Ehrenfeucht-Fraïssé

Consider two players: Spoiler and Duplicator.

- Spoiler chooses an integer k; this will be the number of variables.
- Duplicator chooses a pair of objects (choice depends on k), where the first object is in the class.
- `for i in range(1,k)`
  - Spoiler chooses one object, and chooses a particular value for the i-th variable.
  - Duplicator chooses a particular value for the i-th variable in the other object.

Ehrenfeucht-Fraïssé

Consider two players: Spoiler and Duplicator.

- Spoiler chooses an integer k; this will be the number of variables.
- Duplicator chooses a pair of objects (choice depends on k), where the first object is in the class.
- `for i in range(1,k)`
  - Spoiler chooses one object, and chooses a particular value for the i-th variable.
  - Duplicator chooses a particular value for the i-th variable in the other object.
- If the resulting structure is isomorphic, then Duplicator wins this round, otherwise Spoiler wins.

games and logic

Theorem (WKRMT, Ehrenfeucht?)

If Spolier can win for a particular k and any choice of objects, then there is an axiom that precisely captures this class.
If Duplicator can win for some particular choice of objects for every k, then there is no axiom that precisely captures this class.

whiteboard: playing games

using games

The technique used in our previous paper is essentially an EF game (we didn't realize this at the time).

using games

The technique used in our previous paper is essentially an EF game (we didn't realize this at the time).

EF games are rarely used to prove non-axiomatizability results (we didn't fully appreciate why until after we tried).

accepting strings

We imagine a way of encoding our matroids as strings. Given such an encoding, a class of matroids becomes a class of strings.

accepting strings

We imagine a way of encoding our matroids as strings. Given such an encoding, a class of matroids becomes a class of strings.

An (finite state) automaton is a machine with a fixed number of internal states. Some of these states are designated as “accepting” states.

accepting strings

We imagine a way of encoding our matroids as strings. Given such an encoding, a class of matroids becomes a class of strings.

An (finite state) automaton is a machine with a fixed number of internal states. Some of these states are designated as "accepting" states.

The automaton starts in some initial state, and takes as input a string of characters. It reads the characters one at a time, and at each character the current internal state is determined by the previous internal state and the character read.

accepting strings

We imagine a way of encoding our matroids as strings. Given such an encoding, a class of matroids becomes a class of strings.

An (finite state) automaton is a machine with a fixed number of internal states. Some of these states are designated as "accepting" states.

The automaton starts in some initial state, and takes as input a string of characters. It reads the characters one at a time, and at each character the current internal state is determined by the previous internal state and the character read.

At the end of the string, the machine is in some state. If it is an "accepting" state then the string is deemed to be "accepted", otherwise the string is "rejected".

Theorem (WKRMT)

There exists an automaton that accepts exactly the strings in a particular class if and only if there exists an MSO axiom that captures that class exactly.

Theorem (WKRMT)

There exists an automaton that accepts exactly the strings in a particular class if and only if there exists an MSO axiom that captures that class exactly.

We need to encode matroids as strings such that an automaton can implement the atomic formulae. There is an "automatic" way of then building an automaton that can then implement any compound formula and quantification thereof.

Theorem (WKRMT)

There exists an automaton that accepts exactly the strings in a particular class if and only if there exists an MSO axiom that captures that class exactly.

We need to encode matroids as strings such that an automaton can implement the atomic formulae. There is an "automatic" way of then building an automaton that can then implement any compound formula and quantification thereof.

We want a pair of matroids, one real-representable and one not, that cannot be distinguished by such an automaton.

Theorem (WKRMT)

There exists an automaton that accepts exactly the strings in a particular class if and only if there exists an MSO axiom that captures that class exactly.

We need to encode matroids as strings such that an automaton can implement the atomic formulae. There is an "automatic" way of then building an automaton that can then implement any compound formula and quantification thereof.

We want a pair of matroids, one real-representable and one not, that cannot be distinguished by such an automaton.

idea

There are a finite number of states in the machine, so there must be two different matroids that leave the machine in the same state.

We want to attach some other matroid to both of them such that one of the two resulting compound matroids is representable and the other is not.

The machine will get half-way through and be in the same state for both matroids, after which all remaining input is identical, so the final state will be the same.

Thus, the machine will not be able to tell them apart.

some dowlings

We define two families of Dowling matrids, M_j and N_k.

some dowlings

We define two families of Dowling matrids, M_j and N_k.
For M_j we have the following matrix, where there are j repetitions like the first two columns.

$$M_j = \begin{pmatrix} 1 & 1 & & & & & 1 & \alpha^{-(j-1)} & \alpha^{-j} \\ 1 & \alpha & 1 & 1 & & & & & \\ & & 1 & \alpha & & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & 1 & & \\ & & & & & 1 & \alpha & 1 \quad 1 & \quad 1 \end{pmatrix}$$

some dowlings

We define two families of Dowling matrids, M_j and N_k.
For M_j we have the following matrix, where there are j repetitions like the first two columns.

$$M_j = \begin{pmatrix} 1 & 1 & & & & & 1 & \alpha^{-(j-1)} & \alpha^{-j} \\ 1 & \alpha & 1 & 1 & & & & & \\ & & 1 & \alpha & & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & 1 & & \\ & & & & & 1 & \alpha & 1 \quad 1 & \quad 1 \end{pmatrix}$$

The matrix for N_k doesn't fit nicely on a slide. :(

It has k repetitions like the first two columns except with α^{k-1}, $k-1$ repetitions with α^k, and an extra column at the end with $\alpha^{-k(k-1)}$.

whiteboard: pictures, amalgams, representability

amalgamation and representability

We note that M_j and N_k are defined as matroids, not as represented matroids or matrices.

amalgamation and representability

We note that M_j and N_k are defined as matroids, not as represented matroids or matrices.

We define the matroid $M_j \star N_k$ to be the proper amalgam of M_j and N_k formed by identifying the last three elements of M_j with the second, third and fourth last elements of N_k.

amalgamation and representability

We note that M_j and N_k are defined as matroids, not as represented matroids or matrices.

We define the matroid $M_j \star N_k$ to be the proper amalgam of M_j and N_k formed by identifying the last three elements of M_j with the second, third and fourth last elements of N_k.

Lemma

The proper amalgam of $M_j \star N_k$ exists. (Oxley p.###)
The circuits of $M_j \star N_k$ are either circuits of M_j, circuits of N_k, or symmetric differences of a circuit of M_j with a circuit of N_k.

amalgamation and representability

We note that M_j and N_k are defined as matroids, not as represented matroids or matrices.

We define the matroid $M_j \star N_k$ to be the proper amalgam of M_j and N_k formed by identifying the last three elements of M_j with the second, third and fourth last elements of N_k.

Lemma

The proper amalgam of $M_j \star N_k$ exists. (Oxley p.###)
The circuits of $M_j \star N_k$ are either circuits of M_j, circuits of N_k, or symmetric differences of a circuit of M_j with a circuit of N_k.

Lemma

$M_j \star N_k$ is representable if and only if $j = k$.

matroids to strings [draft version!!]

We encode a matroid as a string of 1's of length $|E|$, followed by a ":", followed by a $x\checkmark$-string of length $2^{|E|}$ denoting the dependent/independent sets.

matroids to strings [draft version!!]

We encode a matroid as a string of 1's of length $|E|$, followed by a ":", followed by a $x\checkmark$-string of length $2^{|E|}$ denoting the dependent/independent sets.

For instance, $U_{2,3}$ becomes "11 : $\checkmark\checkmark\checkmark\checkmark\checkmark\checkmark\checkmark x$"

1	2	3		$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
1	1	1	:	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	x

strings of amalgamations

We encode a matroid of the form $M_j \star N_k$ as the concatenation of the two strings, with the convention that the elements are ordered so that the amalgamation is performed across the first three points of each.

Within each string, the entries in the second half are:

- x if the set is dependent
- $\checkmark$ if the set is independent and spans none of the points of amalgamation
- 1 [2, 3] if the set is independent and spans exactly the first [second, third] point of amalgamation
- ∞ if the set is independent and spans all the points of amalgamation

strings of amalgamations

We encode a matroid of the form $M_j \star N_k$ as the concatenation of the two strings, with the convention that the elements are ordered so that the amalgamation is performed across the first three points of each.

Within each string, the entries in the second half are:

- x if the set is dependent
- $\checkmark$ if the set is independent and spans none of the points of amalgamation
- 1 [2, 3] if the set is independent and spans exactly the first [second, third] point of amalgamation
- ∞ if the set is independent and spans all the points of amalgamation

$$111\ldots:\checkmark\checkmark x1\infty22\checkmark x\ldots|111\ldots:\checkmark x2\infty31xx\infty\ldots$$

encodings and MSO

We need to check that there exists an automaton that can accept $X \subseteq Y$, $\mathrm{ind}(X)$ and $\mathrm{sing}(X)$.

Once this works, there is an "automated" process that itermatively constructs automata to do conjunctions, quantifications, etc

We write two strings, one over the other, and consider a new automaton obtained by taking as characters of the alphabet the columnmns.

This is a rather technical, but essentially straightforward exercise.

whiteboard: encodings? or just skip it.

Theorem (Mayhew, N, Whittle)

There is no axiom in MSO that precisely captures real representability.

Theorem (Mayhew, N, Whittle)

There is no axiom in MSO that precisely captures real representability.

Consider any finite automaton. It has a finite number of states, therefore there are matroids $M_j \star N_j$ and $M_k \star N_j$ such that, upon reaching the end of the first half, the automaton is in the same state.
From this point on, the automaton is given the same remaining input for either matroid and so must end in the same final state.
Thus the automaton must either accept both matroids or reject both matroids. But the first is representable and the second is not.

axioms and classes

Consider any axiom; it defines a class of matroids (those matroids for which it is true). Call a class axiomatizable if there is an axiom that precisely captures it.

So: binary matroids are axiomatizable, real-representable matroids are not.

axioms and classes

Consider any axiom; it defines a class of matroids (those matroids for which it is true). Call a class axiomatizable if there is an axiom that precisely captures it.
So: binary matroids are axiomatizable, real-representable matroids are not.

Question: Is there an axiomatizable class $\mathcal{M}$ such that all real representable matroids are in $\mathcal{M}$ and almost all matroids in $\mathcal{M}$ are real-representable? (so $\mathcal{M}$ is a little bit too big)

Question: Is there an axiomatizable class $\mathcal{M}$ such that almost all real representable matroids are in $\mathcal{M}$ and all matroids in $\mathcal{M}$ are real-representable? (so $\mathcal{M}$ is a little bit too small)

Question: Is there an axiomatizable class $\mathcal{M}$ such that almost all real representable matroids are in $\mathcal{M}$ and almost all matroids in $\mathcal{M}$ are real-representable? (so $\mathcal{M}$ is a little bit too wiggly)