TREE AUTOMATA AND PIGEONHOLE CLASSES OF MATROIDS – I

DARYL FUNK, DILLON MAYHEW, AND MIKE NEWMAN

Abstract. Hliněný's Theorem shows that any sentence in the monadic second-order logic of matroids can be tested in polynomial time, when the input is limited to a class of $\mathbb{F}$-representable matroids with bounded branch-width (where $\mathbb{F}$ is a finite field). If each matroid in a class can be decomposed by a ternary tree in such a way that only a bounded amount of information flows across displayed separations, then the class has bounded decomposition-width. We introduce the pigeonhole property for classes of matroids: if every subclass with bounded branch-width also has bounded decomposition-width, then the class is pigeonhole. A computably pigeonhole class has a stronger property, involving an equivalence relation on subsets of the ground set that we can efficiently compute. We show that Hliněný's Theorem extends to any computably pigeonhole class. Using these ideas, we can extend Hliněný's Theorem to the classes of fundamental transversal matroids, lattice path matroids, bicircular matroids, and H-gain-graphic matroids, where H is any finite group. We also give a characterisation of the families of hypergraphs that can be described via tree automata. A family is defined by a tree automaton if and only if it has bounded decomposition-width. Furthermore, we show that if a class of matroids has the pigeonhole property, and can be axiomatised in monadic second-order logic, then any subclass with bounded branch-width has a decidable monadic second-order theory.

1. Introduction

The *model-checking* problem involves a class of structures and a logical language capable of expressing statements about those structures. We consider a sentence from the language. The goal is then to test whether or not structures from the class satisfies this sentence. Our starting point is the model-checking theorem for graphs due to Courcelle [4], and its matroid sequel by Hliněný [14]. Both theorems allow to us to test monadic-second order sentences in polynomial time, when the input class has bounded structural complexity.

The two theorems in fact supply something stronger than polynomial running time. Both theorems provide *fixed-parameter tractable* algorithms (see [7]). This means that the input contains a numerical parameter, λ. The notion of fixed-parameter tractability captures the distinction between

Date: October 14, 2019.

running times of order $n^{f(\lambda)}$ and those of order $f(\lambda)n^c$, where n is the size of the input, $f(\lambda)$ is a value depending only on λ, and c is a constant. When we restrict to a fixed value of λ, both running times become polynomial in n, but algorithms of the latter type will typically be feasible for a larger range of λ-values. An algorithm with a running time of $O(f(\lambda)n^c)$ is said to be fixed-parameter tractable.

Theorem 1.1 (Courcelle's Theorem)**.** *Let ψ be a sentence in MS_2. There is a fixed-parameter tractable algorithm for testing whether graphs satisfy ψ, where the parameter is tree-width.*

The monadic second-order logic MS_2 allows us to quantify over variables representing vertices, edges, sets of vertices, and sets of edges. NP-complete properties such as Hamiltonicity and 3-colourability can be expressed in MS_2. The extra structure imposed by bounding the tree-width of input graphs transforms these properties from being computationally intractable to tractable.

Theorem 1.2 (Hliněný's Theorem)**.** *Let ψ be a sentence in MS_0 and let $\mathbb{F}$ be a finite field. There is a fixed-parameter tractable algorithm for testing whether $\mathbb{F}$-representable matroids satisfy ψ, where the parameter is branch-width.*

The monadic second-order language MS_0 is described in Section 3. Our main theorem identifies the structural properties underlying the proof of Hliněný's Theorem.

Theorem 1.3. *Let $\mathcal{M}$ be a computably pigeonhole class of matroids. Let ψ be a sentence in MS_0. There is a fixed-parameter tractable algorithm for testing whether matroids in $\mathcal{M}$ satisfy ψ, where the parameter is branch-width.*

This theorem is proved by Proposition 6.1 and Theorem 6.5. In a sequel [10], we will prove that we can now extend Hliněný's Theorem to several natural classes of matroids. Fundamental transversal matroids, lattice path matroids, bicircular matroids, and H-gain-graphic matroids with H a finite group: all these classes have polynomial-time model-checking for the language MS_0 when the input is restricted to subclasses with bounded branch-width.

The pigeonhole property is motivated by matroids representable over finite fields. Let (U, V) be a separation of order at most λ in M, a simple matroid representable over a finite field, $\mathbb{F}$. We can consider M as being a subset of points in the projective space $P = \mathrm{PG}(r(M)-1, \mathbb{F})$. The subspaces of P spanned by U and V intersect in a subspace, P', with affine dimension at most $\lambda - 2$. If X and X' are subsets of U, and their spans intersect P' in the same subspace, then no subset of V can distinguish between them. By this we mean that both $X \cup Z$ and $X' \cup Z$ are independent or both are dependent, for any subset $Z \subseteq V$. This induces an equivalence relation

on subsets of U. The number of classes under this relation is at most the number of subspaces of $\mathrm{PG}(\lambda-2,\mathbb{F})$.

Now we generalise this idea. Let E be a finite set, and let $\mathcal{I}$ be a collection of subsets. If U is a subset of E, then $\sim_U$ is the equivalence relation on subsets of U such that $X \sim_U X'$ if no subset of $E-U$ can distinguish between X and X'; that is, for all $Z \subseteq E-U$, both $X \cup Z$ and $X' \cup Z$ are in $\mathcal{I}$, or neither of them is. A set-system, $(E,\mathcal{I})$, has *decomposition-width* at most q if there is a ternary tree with leaves in bijection with E, such that if U is any set displayed by the tree, then $\sim_U$ has at most q equivalence classes. Since every matroid is a set-system, the decomposition-width of a matroid is a natural specialisation. This notion of decomposition-width is equivalent to that used by Král [17] and by Strozecki [23, 24], although our definition is cosmetically quite different.

Theorem 1.3 relies on tree-automata to check whether monadic sentences are satisfied. (As do the theorems of Courcelle and Hliněný.) Tree automata also provide further evidence that the notion of decomposition-width is a natural one, as we see in the next theorem. In our conception, a tree automaton processes a tree from leaves to root, applying a state to each node. Each node of the tree is initially labelled with a character from a finite alphabet, and the state applied to a node depends on the character written on that node, as well as the states that have been applied to its children. The automaton accepts or rejects the tree according to the state it applies to the root. The characters applied to the leaves can encode a subset of the leaves, so we can think of the automaton as either accepting or rejecting each subset of the leaves. Thus each tree automaton gives rise to a family of set-systems. The ground set of such a set-system is the set of leaves of a tree, and a subset belongs to the system if it is accepted by the automaton. We say that a family of set-systems is *automatic* if there is an automaton which produces the family in this way (Definition 4.5). It is natural to ask which families of set-systems are automatic, and we answer this question in Section 5.

Theorem 1.4. *A class of set-systems is automatic if and only if it has bounded decomposition-width.*

This implies something that is not otherwise obvious: a class of matroids is pigeonhole if and only if the dual class is pigeonhole (Corollary 5.3).

A class of matroids with bounded decomposition-width must have bounded branch-width (Corollary 2.8). The converse does not hold ([10, Lemma 4.1]). If $\mathcal{M}$ is a class of matroids and every subclass with bounded branch-width also has bounded decomposition-width, then $\mathcal{M}$ is *pigeonhole* (Definition 2.9). The class of lattice path matroids has the pigeonhole property ([10, Theorem 7.2]). Other natural classes have an even stronger property. Let $\mathcal{M}$ be a class of matroids. Assume there is a value, $\pi(\lambda)$, for every positive integer λ, such that the following holds: We let M be a matroid in $\mathcal{M}$, and we let U be a subset of $E(M)$. If $\lambda_M(U)$, the connectivity

of U, is at most λ, then $\sim_U$ has at most $\pi(\lambda)$ equivalence classes. Under these circumstances, we say $\mathcal{M}$ is *strongly pigeonhole* (Definition 2.10). The matroids representable over a finite field ([10, Theorem 5.1]) and fundamental transversal matroids ([10, Theorem 6.3]) are strongly pigeonhole classes. A *computably pigeonhole* class is strongly pigeonhole, and has the additional property that we can efficiently compute a relation that refines $\sim_U$ (Definition 6.4).

Now we describe the structure of this article. Section 2 discusses decomposition-width, pigeonhole classes, and strongly pigeonhole classes. In Section 3 we describe the monadic form of logic, MS_0. Section 4 develops the necessary tree-automata ideas. Section 5 is dedicated to the proof of Theorem 1.4. In Section 6 we prove Theorem 1.3. We also show that Theorem 1.3 holds under the weaker condition that the 3-connected matroids in $\mathcal{M}$ form a computably pigeonhole class (Theorem 6.11). (However, we require that we can efficiently compute a description of any minor of the input matroid, so this is not a true strengthening of Theorem 1.3.) Theorem 6.11 is necessary because we do not know that bicircular matroids or H-gain-graphic matroids (with H finite) form a computably pigeonhole class. However, we do know that the subclasses consisting of 3-connected matroids are computably pigeonhole ([10, Theorem 8.4]). (And we conjecture that the entire classes are computably pigeonhole [10, Conjecture 9.3].)

In the final section (Section 7), we consider the question of *decidability*. A class of set-systems has a decidable monadic second-order theory if there is a Turing Machine (not time-constrained) which will take any sentence as input, and decide whether it is satisfied by all systems in the class. Our main result in this section says that if a class of matroids has the pigeonhole property and can be axiomatised by a sentence in monadic second-order logic, then any subclass with bounded branch-width has a decidable theory (Corollary 7.5). The special case of $\mathbb{F}$-representable matroids ($\mathbb{F}$ finite) has been noted by Hliněný and Seese [15, Corollary 5.3]. On the other hand, the class of $\mathbb{K}$-representable rank-3 matroids has an undecidable theory when $\mathbb{K}$ is an infinite field (Corollary 7.7).

A good introduction to automata can be found in [9]. For the basic concepts and notation of matroid theory, we rely on [20]. Recall that if M is a matroid, and (U, V) is a partition of $E(M)$, then $\lambda_M(U)$ is $r_M(U) + r_M(V) - r(M)$. Note that $\lambda_M(U) = \lambda_M(V)$ and $\lambda_M(U) \leq r(M)$. A set, U, is *k-separating* if $\lambda_M(U) < k$, and a *k-separation* is a partition, (U, V), of the ground set such that $|U|, |V| \geq k$, and both U and V are k-separating.

2. Pigeonhole classes

In this section we introduce one of our principal definitions. A class of set-systems has bounded decomposition-width if those set-systems can be decomposed by ternary trees in such a way that only a bounded amount of

information flows across any of the displayed separations. This section is dedicated to formalising these ideas.

Definition 2.1. A *set-system* is a pair $(E, \mathcal{I})$ where E is a finite set and $\mathcal{I}$ is a family of subsets of E. We may refer to members of $\mathcal{I}$ as *independent* sets.

Normally a set-system would be called a hypergraph and the independent sets would be called hyperedges, but we prefer to use more matroid-oriented language.

Definition 2.2. Let $(E, \mathcal{I})$ be a set-system, and let U be a subset of E. Let X and X' be subsets of U. We say X and X' are *equivalent* (relative to U), written $X \sim_U X'$, if for every subset $Z \subseteq E - U$, the set $X \cup Z$ is in $\mathcal{I}$ if and only if $X' \cup Z$ is in $\mathcal{I}$.

Informally, we think of $X \sim_U X'$ as meaning that no subset of $E - U$ can 'distinguish' between X and X'. It is clear that $\sim_U$ is an equivalence relation on subsets of U. Note that by taking Z to be the empty set, we can see that no member of $\mathcal{I}$ is equivalent to a subset not in $\mathcal{I}$. Assume that $\mathcal{I}$ is closed under subset containment (as would be the case if $\mathcal{I}$ were the family of independent sets in a matroid). In this special case, all subsets of U that are not in $\mathcal{I}$ are equivalent.

Proposition 2.3. *Let $(E, \mathcal{I})$ be a set-system and let (U, V, W) be a partition of E (into possibly empty sets). If $X \sim_U X'$ and $Y \sim_V Y'$, then $(X \cup Y) \sim_{(U \cup V)} (X' \cup Y')$. In particular, $X \cup Y$ belongs to $\mathcal{I}$ if and only if $X' \cup Y'$ does.*

Proof. Let Z be an arbitrary subset of W, and assume that $X \cup Y \cup Z$ is in $\mathcal{I}$. Because $Y \cup Z \subseteq E - U$ and $X \sim_U X'$, it follows that $X' \cup Y \cup Z$ is in $\mathcal{I}$. Now $X' \cup Z \subseteq E - V$ and $Y \sim_V Y'$, so $X' \cup Y' \cup Z$ is in $\mathcal{I}$. By an identical argument, we see that if $X' \cup Y' \cup Z$ is in $\mathcal{I}$, then so is $X \cup Y \cup Z$. □

Proposition 2.4. *Let $(E, \mathcal{I})$ be a set-system, and let (U, V) be a partition of E. If q is the number of equivalence classes under $\sim_U$, then the number of equivalence classes under $\sim_V$ is at most 2^q.*

Proof. Let the equivalence classes under $\sim_U$ be $\mathcal{E}_1, \ldots, \mathcal{E}_q$, and let X_i be a member of $\mathcal{E}_i$ for each i. Let Z be any subset of V. We define $b(Z)$ to be the binary string of length q, where the ith character is 1 if and only if $X_i \cup Z$ is in $\mathcal{I}$. It is clear that this string is well-defined, and does not depend on our choice of the representatives X_i. We complete the proof by showing that when $Z, Z' \subseteq Z$ satisfy $b(Z) = b(Z')$, they also satisfy $Z \sim_V Z'$. Assume this is not the case, and let $X \subseteq U$ be such that exactly one of $X \cup Z$ and $X \cup Z'$ is in $\mathcal{I}$. Without loss of generality, we assume $X \cup Z \in \mathcal{I}$. Assume that X is a member of $\mathcal{E}_i$. Since $b(Z) = b(Z')$, either both of $X_i \cup Z$ and $X_i \cup Z'$ are in $\mathcal{I}$, or neither is. In the first case, $X_i \cup Z' \in \mathcal{I}$ and $X \cup Z' \notin \mathcal{I}$, so we contradict $X_i \sim_U X$. In the second case, $X_i \cup Z \notin \mathcal{I}$ and $X \cup Z \in \mathcal{I}$, so we reach the same contradiction. □

A *ternary tree* is one in which every vertex has degree three or one. A degree-one vertex is a *leaf*. Let $M = (E, \mathcal{I})$ be a set-system. A *decomposition* of M is a pair (T, φ), where T is a ternary tree, and φ is a bijection from E into the set of leaves of T. Let e be an edge joining vertices u and v in T. Then e partitions E into sets (U_e, V_e) in the following way: an element $x \in E$ belongs to U_e if and only if the path in T from $\varphi(x)$ to u does not contain v. We say that the partition (U_e, V_e) and the sets U_e and V_e are *displayed* by the edge e. Define $\mathrm{dw}(M; T, \varphi)$ to be the maximum number of equivalence classes in $\sim_U$, where the maximum is taken over all subsets, U, displayed by an edge in T. Define $\mathrm{dw}(M)$ to be the minimum value of $\mathrm{dw}(M; T, \varphi)$, where the minimum is taken over all decompositions (T, φ) of M. This minimum is the *decomposition-width* of M. The notion of decomposition-width specialises to matroids in the obvious way.

Definition 2.5. Let M be a matroid. Then $\mathrm{dw}(M)$ is equal to $\mathrm{dw}(E(M), \mathcal{I}(M))$.

It is an exercise to show that this notion of decomposition-width is equivalent to those used by Král [17] or Strozecki [23, 24]. Král states the next result without proof.

Proposition 2.6. *Let x be an element of the matroid M. Then* $\mathrm{dw}(M\backslash x) \leq \mathrm{dw}(M)$ *and* $\mathrm{dw}(M/x) \leq \mathrm{dw}(M)$.

Proof. Let (T, φ) be a decomposition of M and assume that whenever U is a displayed set, then $\sim_U$ has no more than $\mathrm{dw}(M)$ equivalence classes. Let T' be the tree obtained from T by deleting $\varphi(x)$ and then contracting an edge so that every vertex in T' has degree one or three. Let U be any subset of $E(M) - x$ displayed by T'. Then either U or $U \cup x$ is displayed by T. Let M' be either $M\backslash x$ or M/x. We will show that in M', the number of equivalence classes under $\sim_U$ is no greater than the number of classes under $\sim_U$ or $\sim_{U \cup x}$ in M. Let X and X' be representatives of distinct classes under $\sim_U$ in M'. We will be done if we can show that these representatives correspond to distinct classes in M. Without loss of generality, we can assume that Z is a subset of $E(M) - (U \cup x)$ such that $X \cup Z$ is independent in M', but $X' \cup Z$ is dependent. If $M' = M\backslash x$, then $X \cup Z$ is independent in M and $X' \cup Z$ is dependent, and thus we are done. So we assume that $M' = M/x$. If U is displayed by T, then we observe that $X \cup (Z \cup x)$ is independent in M, while $X' \cup (Z \cup x)$ is dependent. On the other hand, if $U \cup x$ is displayed, then $(X \cup x) \cup Z$ is independent in M and $(X' \cup x) \cup Z$ is dependent. □

Proposition 2.6 shows that the class of matroids with decomposition-width at most k is minor-closed.

Let M be a matroid. The *branch-width* of M (written $\mathrm{bw}(M)$) is defined as follows. If (T, φ) is a decomposition of $M = (E(M), \mathcal{I}(M))$, then $\mathrm{bw}(M; T, \varphi)$ is the maximum value of

$$\lambda_M(U_e) + 1 = r_M(U_e) + r_M(V_e) - r(M) + 1,$$

where the maximum is taken over all partitions (U_e, V_e) displayed by edges of T. Now $\mathrm{bw}(M)$ is the minimum value of $\mathrm{bw}(M; T, \varphi)$, where the minimum is taken over all decompositions of M. We next show that for classes of matroids, bounded decomposition-width implies bounded branch-width.

Proposition 2.7. *Let M be a matroid, and let U be a subset of $E(M)$. There are at least $\lambda_M(U) + 1$ equivalence classes under the relation $\sim_U$.*

Proof. Let λ stand for $\lambda_M(U)$. We will prove that $\sim_U$ has at least $\lambda + 1$ equivalence classes. Define V to be $E(M) - U$. Let B_V be a basis of $M|V$, and let B be a basis of M that contains B_V. Then $B \cap U$ is independent in $M|U$, and

$$\begin{aligned} r(U) - |B \cap U| = r(U) - (|B| - |B_V|) &= r(U) - (r(M) - r(V)) \\ &= r(U) - (r(U) - \lambda) = \lambda. \end{aligned}$$

Therefore we let $(B \cap U) \cup \{x_1, \ldots, x_\lambda\}$ be a basis of $M|U$, where $x_1, \ldots, x_\lambda$ are distinct elements of $U - B$. Next we construct a sequence of distinct elements, $y_1, \ldots, y_\lambda$ from B_V such that $(B - \{y_1, \ldots, y_i\}) \cup \{x_1, \ldots, x_i\}$ is a basis of M for each $i \in \{0, \ldots, \lambda\}$. We do this recursively. Let C be the unique circuit contained in

$$(B - \{y_1, \ldots, y_i\}) \cup \{x_1, \ldots, x_i\} \cup x_{i+1}$$

and note that x_{i+1} is in C. If C contains no elements of B_V, then it is contained in $(B \cap U) \cup \{x_1, \ldots, x_\lambda\}$, which is impossible. So we simply let y_{i+1} be an arbitrary element in $C \cap B_V$.

We complete the proof by showing that

$$(B \cap U) \cup \{x_1, \ldots, x_i\} \quad \text{and} \quad (B \cap U) \cup \{x_1, \ldots, x_j\}$$

are inequivalent under $\sim_U$ whenever $0 \leq i < j \leq \lambda$. Indeed, if $Z = B_V - \{y_1, \ldots, y_i\}$, then $(B \cap U) \cup \{x_1, \ldots, x_i\} \cup Z$ is a basis of M, and is properly contained in $(B \cap U) \cup \{x_1, \ldots, x_j\} \cup Z$, so the last set is dependent, and we are done. □

Corollary 2.8. *Let M be a matroid. Then* $\mathrm{dw}(M) \geq \mathrm{bw}(M)$.

Proof. Assume that $\mathrm{bw}(M) > \mathrm{dw}(M)$. Let (T, φ) be a decomposition of M such that if U is any set displayed by an edge of T, then $\sim_U$ has at most $\mathrm{dw}(M)$ equivalence classes. There is some edge e of T displaying a set U_e such that $\lambda_M(U_e) + 1 > \mathrm{dw}(M)$, for otherwise this decomposition of M certifies that $\mathrm{bw}(M) \leq \mathrm{dw}(M)$. But $\sim_{U_e}$ has at least $\lambda_M(U_e)+1$ equivalence classes by Proposition 2.7. As $\lambda_M(U_e) + 1 > \mathrm{dw}(M)$, this contradicts our choice of (T, φ). □

It is easy to see that the class of rank-3 sparse paving matroids has unbounded decomposition width (see [10, Lemma 4.1]), so the converse of Corollary 2.8 does not hold. Král proved the special case of Corollary 2.8 when M is representable over a finite field [17, Theorem 2].

Since we would like to consider natural classes of matroids that have unbounded branch-width, we are motivated to make the next definition.

Definition 2.9. Let $\mathcal{M}$ be a class of matroids. Then $\mathcal{M}$ is *pigeonhole* if, for every positive integer, λ, there is an integer $\rho(\lambda)$ such that $\mathrm{bw}(M) \leq \lambda$ implies $\mathrm{dw}(M) \leq \rho(\lambda)$, for every $M \in \mathcal{M}$.

Thus a class of matroids is pigeonhole if every subclass with bounded branch-width also has bounded decomposition-width. The class of $\mathbb{F}$-representable matroids is pigeonhole when $\mathbb{F}$ is a finite field [10, Theorem 5.1]. Note that the class of $\mathbb{F}$-representable matroids certainly has unbounded decomposition-width, since it has unbounded branch-width. Some natural classes possess a stronger property than the pigeonhole property:

Definition 2.10. Let $\mathcal{M}$ be a class of matroids. Assume that for every positive integer λ, there is a positive integer $\pi(\lambda)$, such that whenever $M \in \mathcal{M}$ and $U \subseteq E(M)$ satisfies $\lambda_M(U) \leq \lambda$, there are at most $\pi(\lambda)$ equivalence classes under $\sim_U$. In this case we say that $\mathcal{M}$ is *strongly pigeonhole*.

Proposition 2.11. *If a class of matroids is strongly pigeonhole, then it is pigeonhole.*

Proof. Let $\mathcal{M}$ be a strongly pigeonhole class, and let π be the function from Definition 2.10. We may as well assume that π is non-decreasing. Let λ be any positive integer, and let M be a matroid in $\mathcal{M}$ with branch-width at most λ. Let (T, φ) be a decomposition of M such that $\lambda_M(U)+1 \leq \lambda$ for any set U displayed by an edge of T. Then there are at most $\pi(\lambda-1)$-equivalence classes under $\sim_U$. Thus (T, φ) demonstrates that $\mathrm{dw}(M) \leq \pi(\lambda - 1)$. So $\mathrm{bw}(M) \leq \lambda$ implies $\mathrm{dw}(M) \leq \pi(\lambda - 1)$ for each $M \in \mathcal{M}$, and the result follows. □

Remark 2.12. To see that the strong pigeonhole property is strictly stronger than the pigeonhole property, let $\mathcal{M}$ be the class of rank-two matroids. Let M be a member of $\mathcal{M}$ with t parallel pairs (where $t \geq 2$). Let U be a set that contains exactly one element from each of these pairs. Then $\lambda_M(U) = 2$. However, it is easy to demonstrate that there are at least t equivalence classes under $\sim_U$, so this number is unbounded. This demonstrates that $\mathcal{M}$ is not strongly pigeonhole. However, if M is in $\mathcal{M}$, then there is a decomposition of M such that whenever (U, V) is a displayed partition, at most one parallel class contains elements of both U and V. Now we easily check that $\sim_U$ has at most five equivalence classes, so $\mathrm{dw}(M) \leq 5$ for all $M \in \mathcal{M}$, implying that $\mathcal{M}$ is pigeonhole.

3. Monadic logic

In this section we construct the formal language MS_0. We give ourselves a countably infinite supply of variables: $X_1, X_2, X_3, \ldots$. There is a single unary predicate: Ind, and one binary predicate: $\subseteq$. We use the standard connectives $\wedge$ and $\neg$, and the quantifier $\exists$. The *atomic formulas* have the

form $\mathrm{Ind}(X_i)$, and $X_i \subseteq X_j$. The atomic formula $\mathrm{Ind}(X_i)$ has X_i as its *free variable*, whereas the free variables of $X_i \subseteq X_j$ are X_i and X_j. A *formula* is constructed by a finite application of the following rules:

(i) an atomic formula is a formula,
(ii) if ψ is a formula, then $\neg\psi$ is a formula with the same free variables as ψ,
(iii) if ψ is a formula, and X_i is a free variable in ψ, then $\exists X_i\psi$ is a formula; its free variables are the free variables of ψ except for X_i, which is a *bound variable* of $\exists X_i\psi$,
(iv) if ψ and ϕ are formulas, and no variable is free in one of ψ and ϕ and bound in the other, then $\psi \wedge \phi$ is a formula, and its free variables are exactly those that are free in either ψ or ϕ. (We can rename bound variables, so this restriction does not significantly constrain us.)

A formula is a *sentence* if it has no free variables, and is *quantifier-free* if it has no bound variables.

Let $(E, \mathcal{I})$ be a set-system. Let ψ be a formula in MS_0 and let F be the set of free variables in ψ. An *interpretation* of ψ in $(E, \mathcal{I})$ is a function θ from F into the power set of E. So we think of θ as a set of ordered pairs with the first element being a variable in F and the second being a subset of E. We define what it means for the pair $(E, \mathcal{I})$ to *satisfy* ψ under the interpretation θ. If ψ is $\mathrm{Ind}(X_i)$, then $(E, \mathcal{I})$ satisfies ψ if $\theta(X_i)$ is in $\mathcal{I}$. Similarly, $X_i \subseteq X_j$ is satisfied if $\theta(X_i) \subseteq \theta(X_j)$. Now we extend this definition to formulas that are not atomic. If $\psi = \neg\phi$, then $(E, \mathcal{I})$ satisfies ψ if and only if it does not satisfy ϕ under θ. If $\psi = \phi_1 \wedge \phi_2$, then ψ is satisfied if $(E, \mathcal{I})$ satisfies both ϕ_1 and ϕ_2 under the interpretations consisting of θ restricted to the free variables of ϕ_1 and ϕ_2. Finally, if $\psi = \exists X_i\phi$, then $(E, \mathcal{I})$ satisfies ψ if and only if there is a subset $Y_i \subseteq E$ such that $(E, \mathcal{I})$ satisfies ϕ under the interpretation $\theta \cup \{(X_i, Y_i)\}$.

We use $\psi \vee \phi$ as shorthand for $\neg((\neg\psi) \wedge (\neg\phi))$, and $\psi \to \phi$ as shorthand for $(\neg\psi) \vee \phi$. The predicate $\psi \leftrightarrow \phi$ is shorthand for $(\psi \to \phi) \wedge (\phi \to \psi)$. If X_i is a free variable in ψ, then $\forall X_i\psi$ stands for $\neg\exists X_i\neg\psi$. The predicate $\mathrm{Empty}(X_i)$ stands for

$$\forall X(X \subseteq X_i \to X_i \subseteq X)$$

and is satisfied exactly when X_i is interpreted as the empty set. (Here X is a variable not equal to X_i.) Similarly, $\mathrm{Sing}(X_i)$ stands for

$$\neg\,\mathrm{Empty}(X_i) \wedge \forall X(X \subseteq X_i \to (\mathrm{Empty}(X) \vee X_i \subseteq X))$$

and is satisfied exactly when X_i is interpreted as a singleton set.

As is demonstrated in [18], there are sentences that are satisfied by $(E, \mathcal{I})$ if and only if $\mathcal{I}$ is the family of independent sets of a matroid. Furthermore, there are MS_0 sentences that characterise any minor-closed class of matroids having only finitely many excluded minors (see [18] or [13, Lemma 5.1]). On

the other hand, the main theorem of [18] shows that no MS_0 sentence characterises the class of representable matroids, or the class of $\mathbb{K}$-representable matroids when $\mathbb{K}$ is an infinite field.

4. Automatic classes

Our second principal definition involves families of set-systems that can be encoded by a tree, where that tree can be processed by a machine simulating an independence oracle. We start by introducing tree automata. We use [9] as a general reference.

Definition 4.1. Let T be a tree with a distinguished *root* vertex, t. Assume that every vertex of T other than t has degree one or three, and that if T has more than one vertex, then t has degree two. The *leaves* of T are the degree-one vertices. In the case that t is the only vertex, we also consider t to be a leaf. Let $L(T)$ be the set of leaves of T. If T has more than one vertex, and v is a non-leaf, then v is adjacent with two vertices that are not contained in the path from v to t. These two vertices are the *children* of v. We distinguish the *left* child and the *right* child of v. Now let Σ be a finite *alphabet* of *characters*. Let σ be a function from $V(T)$ to Σ. Under these circumstances we say that (T, σ) is a Σ-*tree*.

Definition 4.2. A *tree automaton* is a tuple $(\Sigma, Q, F, \delta_0, \delta_2)$, where Σ is a finite alphabet, and Q is a finite set of *states*. The set of *accepting states* is a subset $F \subseteq Q$. The functions, $\delta_0 \colon \Sigma \to 2^Q$ and $\delta_2 \colon \Sigma \times Q \times Q \to 2^Q$, are known as *transition rules*.

We think of the automaton as processing the vertices in a Σ-tree, from leaves to root, applying a set of states to each vertex. The set of states applied to a leaf, v, is given by the image of δ_0, applied to the Σ-label of v. For a non-leaf vertex, v, we apply δ_2 to the tuple consisting of the Σ-label of v, a state applied to the left child, and a state applied to right child. We take the union of all such outputs, as we range over all states applied to the children of v, and this union is the set we apply to v.

More formally, let $A = (\Sigma, Q, F, \delta_0, \delta_2)$ be an automaton. Let (T, σ) be a Σ-tree with root t. We let $r \colon V(T) \to 2^Q$ be the function recursively defined as follows:

(i) if v is a leaf of T, then $r(v) = \delta_0(\sigma(v))$,
(ii) if v has left child v_L and right child v_R, then

$$r(v) = \bigcup_{(q_L, q_R) \in r(v_L) \times r(v_R)} \delta_2(\sigma(v), q_L, q_R).$$

We say that r is the *run* of the automaton A on (T, σ). Note that we define a union taken over an empty collection to be the empty set. Thus if a child of v has been assigned an empty set of states, then v too will be assigned an empty set of states. We say that A *accepts* (T, σ) if $r(t)$ contains an accepting state.

The automaton, $A = (\Sigma, Q, F, \delta_0, \delta_2)$, is *deterministic* if all the images of δ_0 and δ_2 are singleton sets. Thus a deterministic automaton applies a (set containing a) single state to each vertex. The next result shows that non-determinism in fact gives us no extra computing power. The idea here dates to Rabin and Scott [21] (see [8, Theorem 12.3.1]).

Lemma 4.3. *Let $A' = (\Sigma, Q, F', \delta_0', \delta_2')$ be a tree automaton. There exists a deterministic tree automaton,*

$$A = (\Sigma, 2^Q, F, \delta_0, \delta_2),$$

such that A' and A accept exactly the same Σ-trees.

Proof. Note that the states in A are sets of states in A'. Let F be $\{X \in 2^Q \colon X \cap F' \neq \emptyset\}$. Thus a state is accepting in A if and only if it contains an accepting state of A'. For each $\sigma \in \Sigma$, we define $\delta_0(\sigma)$ to be $\{\delta_0'(\sigma)\}$. For any $\sigma \in \Sigma$, and any $X, Y \in 2^Q$, we set

$$\delta_2(\sigma, X, Y) = \left\{ \bigcup_{(q_L, q_R) \in X \times Y} \delta_2'(\sigma, q_L, q_R) \right\}.$$

Thus every image of δ_0 or δ_2 is a singleton set, so A is deterministic, as desired.

Let (T, σ) be a Σ-tree with root t. Let r' and r be the runs of A' and A on (T, σ). We easily establish that $r(v) = \{r'(v)\}$, for each vertex v. If A' accepts (T, σ), then $r'(t)$ contains a state in F'. Therefore $r'(t)$ is a member of F, so $r(t) = \{r'(t)\}$ contains a member of F. Hence A also accepts (T, σ). For the converse, assume that A accepts (T, σ). Then $r(t) = \{r'(t)\}$ contains an accepting state. This means that $r'(t)$ is not disjoint from F', so A' also accepts (T, σ), and we are done. □

We would like to use tree automata to decide if a formula in MS_0 is satisfied by a set-system, $(E, \mathcal{I})$. This formula may have free variables, and in this case, deciding whether the formula is satisfied only makes sense if we assign subsets of E to the free variables. So our next job is to formalise a way to encode this assignment into the leaf labels of a tree.

Let I be a set of positive integers. We use $\{0,1\}^I$ to denote the set of functions from I into $\{0,1\}$. If I is empty, then $\{0,1\}^I$ is the empty set. Let Σ be a finite alphabet, and let (T, σ) be a Σ-tree. Let φ be a bijection from the finite set E into $L(T)$. We let $I \subseteq \mathbb{Z}^+$ be a set of indices, and let $\mathcal{S} = \{Y_i\}_{i \in I}$ be a family of subsets of E. Now we define $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$ to be a $(\Sigma \cup \Sigma \times \{0,1\}^I)$-tree with T as its underlying tree. If I is empty, then we simply set $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$ to be (T, σ). Now we assume I is non-empty. If v is a non-leaf vertex of T, then it receives the label $\sigma(v)$ in $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$. However, if v is a leaf, then it receives a label $(\sigma(v), s)$, where s is the function from I to $\{0,1\}$ taking i to 1 if and only if $\varphi^{-1}(v)$ is in Y_i. Thus the label on the leaf v encodes the membership of $\varphi^{-1}(v)$ in the sets belonging to $\mathcal{S}$.

We say that a tree automaton is *n-ary* if its alphabet is $\Sigma \cup \Sigma \times \{0,1\}^I$, where Σ is a finite set, and I is a set of n positive integers.

Definition 4.4. Let Σ be a finite set, and let A be a 1-ary tree automaton with alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{i\}}$. Let (T,σ) be a Σ-tree, and let φ be a bijection from the finite set E into $L(T)$. We define the set-system $M(A,T,\sigma,\varphi)$ as follows:

$$M(A,T,\sigma,\varphi) = (E, \{Y_i \subseteq E\colon A \text{ accepts } \operatorname{enc}(T,\sigma,\varphi,\{Y_i\})\}).$$

Now we are ready to give our second main definition.

Definition 4.5. Let $\mathcal{M}$ be a class of set-systems. Assume that A is a 1-ary tree automaton with alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{i\}}$. Assume also that for any $M = (E,\mathcal{I})$ in $\mathcal{M}$, there is a Σ-tree (T_M,σ_M), and a bijection $\varphi_M\colon E \to L(T_M)$ having the property that $M = M(A,T_M,\sigma_M,\varphi_M)$. In this case we say that $\mathcal{M}$ is *automatic*.

We say that (T_M,σ_M) from Definition 4.5 is a *parse tree* for M (relative to the automaton A).

Definition 4.6. Let $\mathcal{M}$ be a class of matroids. We say that $\mathcal{M}$ is *automatic* if the class of set-systems $\{(E(M),\mathcal{I}(M))\colon M \in \mathcal{M}\}$ is automatic.

Thus a class of matroids is automatic if there is an automaton that acts as follows: for each matroid M in the class, there is a parse tree (T_M,σ_M), and a bijection φ_M from the ground set of M to the leaves, such that when the leaf labels encode the set $Y_i \subseteq E(M)$, the automaton accepts if and only if Y_i is independent. In other words, there is an automaton that will simulate an independence oracle on an appropriately chosen parse tree for any matroid in the class.

The next lemma says that if there is an automaton that simulates an independence oracle, then there is an automaton that will test any MS_0 formula. The ideas in the proof appear to have originated with Kleene [16].

Lemma 4.7. *Let A' be a 1-ary tree automaton with alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{i\}}$. Let ψ be a formula in MS_0 with free variables $\{X_i\}_{i\in I}$. There is a $|I|$-ary tree automaton, A with alphabet $\Sigma \cup \Sigma \times \{0,1\}^I$, such that for every Σ-tree (T,σ), every bijection, φ, from a finite set E into $L(T)$, and every family $\mathcal{S} = \{Y_i\}_{i\in I}$ of subsets of E, the automaton A accepts $\operatorname{enc}(T,\sigma,\varphi,\mathcal{S})$ if and only if the set-system $M(A',T,\sigma,\varphi)$ satisfies ψ under the interpretation taking X_i to Y_i for each $i \in I$.*

When we say that A *decides* ψ, we mean that A accepts $\operatorname{enc}(T,\sigma,\varphi,\mathcal{S})$ if and only if $M(A',T,\sigma,\varphi)$ satisfies ψ, for any T, σ, and φ.

Remark 4.8. If $\mathcal{M}$ is an automatic class, then by definition, for each $M \in \mathcal{M}$, we can choose T, σ, and φ so that $M(A',T,\sigma,\varphi)$ is M. Therefore Lemma 4.7 will provide us with a way to test whether M satisfies ψ: we simply run A on the appropriately labelled tree.

Proof of Lemma 4.7. We prove the lemma by induction on the number of steps used to construct the formula ψ. Start by assuming that ψ is atomic. Assume that ψ is $\mathrm{Ind}(X_j)$. Then the result follows from the definitions by setting A to be A'.

Next we assume that ψ is the atomic formula $X_j \subseteq X_k$. We let the state space of A be $\{\checkmark, \times\}$, and let $\checkmark$ be the only accepting state. Define δ_0 so that for any $\alpha \in \Sigma$ and any function $s \in \{0,1\}^{\{j,k\}}$, the image $\delta_0(\alpha, s)$ is $\{\times\}$ if $(s(i), s(j)) = (1,0)$, and otherwise $\delta_0(\alpha, s)$ is $\{\checkmark\}$. (We also define $\delta_0(\alpha) = \{\times\}$ for each $\alpha \in \Sigma$, so that the domain of δ_2 is the entire alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{j,k\}}$.) We define δ_2 so that for any $\alpha \in \Sigma$,

$$\delta_2(\alpha, \times, \times) = \delta_2(\alpha, \times, \checkmark) = \delta_2(\alpha, \checkmark, \times) = \{\times\}$$

and $\delta_2(\alpha, \checkmark, \checkmark) = \{\checkmark\}$. (We additionally say that if the input to δ_2 includes a character in $\Sigma \times \{0,1\}^{\{j,k\}}$, then the output is $\{\times\}$.) Note that as A processes the tree, it assigns $\times$ to a leaf if and only if the corresponding element of E is in Y_j but not Y_k. If any leaf is assigned $\times$, then this state is propagated towards the root. Thus A decides the formula $X_j \subseteq X_k$, as desired.

We may now assume that ψ is not atomic. First assume that ψ is a negation, $\neg\phi$. Note that the free variables of ϕ are $\{X_i\}_{i \in I}$. By induction, there is an automaton, A_ϕ, that accepts $\mathrm{enc}(T, \sigma, \varphi, \{Y_i\}_{i \in I})$ if and only if $M(A', T, \sigma, \varphi)$ satisfies ϕ. By Lemma 4.3, we can assume that A_ϕ is deterministic. Now we produce A by modifying A_ϕ so that a state is accepting in A exactly when it is not accepting in A_ϕ. Then A decides $\neg\phi$.

Next we assume that ψ is a conjunction, $\phi_1 \wedge \phi_2$. For $i = 1, 2$, let I_i be the set of free variables in ϕ_i. Thus $I = I_1 \cup I_2$. Inductively, there are automata A_1 and A_2 that decide ϕ_1 and ϕ_2. For $i = 1, 2$, assume that A_i is the automaton

$$(\Sigma \cup \Sigma \times \{0,1\}^{I_i}, Q_i, F_i, \delta_{0,i}, \delta_{2,i}).$$

The idea of this proof is quite simple: we let A run A_1 and A_2 in parallel, and accept if and only if both A_1 and A_2 accept. To that end, we set Q to be $Q_1 \times Q_2$, and set F to be $F_1 \times F_2$. If s is a function in $\{0,1\}^I$, then $s \restriction_{I_i}$ is the restriction of s to I_i. Now we define δ_0 so that it takes (α, s) to

$$\delta_{0,1}(\alpha, s \restriction_{I_1}) \times \delta_{0,2}(\alpha, s \restriction_{I_2})$$

for any $\alpha \in \Sigma$ and any $s \in \{0,1\}^I$. (And any other output of δ_0 is the empty set.) We similarly define δ_2 so that $\delta_2(\alpha, (q_{L,1}, q_{L,2}), (q_{R,1}, q_{R,2}))$ is

$$\delta_{2,1}(\alpha, q_{L,1}, q_{R,1}) \times \delta_{2,2}(\alpha, q_{L,2}, q_{R,2}).$$

(And any other output is the empty set.) It is easy to see that A acts as we desire, and therefore decides ψ.

Finally, we must assume that ψ is $\exists X_i \phi$, where the free variables of ϕ are $\{X_j\}_{j \in I \cup \{i\}}$ and i is not in I. By induction, we can assume that the automaton

$$A_\phi = (\Sigma \cup \Sigma \times \{0,1\}^{I \cup i}, Q_\phi, F_\phi, \delta_{0,\phi}, \delta_{2,\phi})$$

decides ϕ. For each $s \in \{0,1\}^I$, we set s^0 to be the function in $\{0,1\}^{I \cup i}$ such that $s^0 \restriction_I = s$, and $s^0(i) = 0$. We similarly define $s^1 \in \{0,1\}^{I \cup i}$ so that $s^1 \restriction_I = s$ and $s^1(i) = 1$. Now for each $\alpha \in \Sigma$ we set

$$\delta_0(\alpha, s) = \delta_{0,\phi}(\alpha, s^0) \cup \delta_{0,\phi}(\alpha, s^1).$$

We define any other output of δ_0 to be the empty set. Thus δ_0 sends (α, s) to the set of states that could be applied by A_ϕ to a leaf labelled by (α, s'), where s' extends the domain of s to include i. We define $\delta_2(\alpha, q_L, q_R)$ to be $\delta_{2,\phi}(\alpha, q_L, q_R)$ when α is in Σ. Any output of δ_2 that is not yet defined, we now declare to be the empty set. We define the state space and the accepting states of A to be exactly those of A_ϕ. We must now show that A decides $\exists X_i \phi$. We let (T, σ) be an arbitrary Σ-tree, and we let φ be a bijection from the finite set E into $L(T)$.

Assume that $M(A', T, \sigma, \varphi)$ satisfies $\exists X_i \phi$ under the interpretation that takes X_j to $Y_j \subseteq E$ for each $j \in I$. Then there is a subset $Y_i \subseteq E$ such that $M(A', T, \sigma, \varphi)$ satisfies ϕ under the interpretation that takes X_j to Y_j for all $j \in I \cup i$. Let $\mathcal{S}_\phi$ be $\{Y_j\}_{j \in I \cup i}$. By induction, A_ϕ accepts $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S}_\phi)$. Let r_ϕ be the run of A_ϕ on $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S}_\phi)$. Then $r_\phi(t)$ contains a state in F_ϕ, where t is the root of T. Let $\mathcal{S} = \{Y_j\}_{j \in I}$, and let r be the run of A on $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$. It is easy to inductively prove that $r(v) \supseteq r_\phi(v)$, for every vertex v. Therefore $r(t)$ contains an accepting state, so A accepts $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$.

For the converse, assume that A accepts $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$, where $\mathcal{S} = \{Y_j\}_{j \in I}$ is a family of subsets of E. Let r be the run of A on $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S})$. We recursively nominate a state $q(v)$ chosen from $r(v)$, for each vertex v. Since A accepts, there is an accepting state in $r(t)$. We define $q(t)$ to be this accepting state. Now assume that $q(v)$ is defined, and that the children of v are v_L and v_R. Then there are states $q_L \in r(v_L)$ and $q_R \in r(v_R)$ such that $\delta_2(\sigma(v), q_L, q_R)$ contains $q(v)$. We choose $q(v_L)$ to be q_L, and $q(v_R)$ to be q_R. Thus we have defined $q(v)$ for each vertex v.

We will now define a set $Y_i \subseteq E$. Let v be an arbitrary leaf. We describe a method for deciding if $\varphi^{-1}(v)$ is in Y_i. Let $s \in \{0,1\}^I$ be the function that records whether $\varphi^{-1}(v)$ is in Y_j, for $j \in I$. Thus $\delta_0(\sigma(v), s)$ includes $q(v)$. Now

$$\delta_0(\sigma(v), s) = \delta_{0,\phi}(\sigma, s^0) \cup \delta_{0,\phi}(\sigma, s^1)$$

If $q(v)$ is in $\delta_{0,\phi}(\sigma, s^0)$, we declare $\varphi^{-1}(v)$ not to be in Y_i. Otherwise we declare $\varphi^{-1}(v)$ to be in Y_i.

Let $\mathcal{S}_\phi$ be the family $\{Y_j\}_{j \in I \cup i}$. Let r_ϕ be the run of A_ϕ on $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S}_\phi)$. It is easy to prove by induction that $r_\phi(v)$ contains $q(v)$, for every vertex v. Therefore $r_\phi(t)$ contains an accepting state, so A_ϕ accepts $\mathrm{enc}(T, \sigma, \varphi, \mathcal{S}_\phi)$. By induction, this means that $M(A', T, \sigma, \varphi)$ satisfies ϕ under the interpretation taking each X_j to Y_j for $j \in I \cup i$. Hence $\exists X_i \phi$ is satisfied by the interpretation taking X_j to Y_j for $i \in I$. This completes the proof that A decides $\psi = \exists X_i \phi$, and hence the proof of the lemma. $\square$

5. Characterising automatic classes

Now we can prove Theorem 1.4. We split the proof into two lemmas.

Lemma 5.1. *Let $\mathcal{M}$ be a class of set-systems. If $\mathcal{M}$ is automatic, then it has bounded decomposition-width.*

Proof. Let A be a 1-ary tree-automaton with alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{i\}}$ and state space Q such that for every $M = (E, \mathcal{I})$ in $\mathcal{M}$, there is a Σ-tree (T_M, σ_M) and a bijection $\varphi_M \colon E \to L(T_M)$ having the property that A accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ if and only if Y_i is in $\mathcal{I}$, for any $Y_i \subseteq E$. By applying Lemma 4.3, we assume that A is deterministic.

Let $M = (E, \mathcal{I})$ be an arbitrary set-system in $\mathcal{M}$. Let e be an arbitrary edge in T_M, and assume e is incident with the vertices u and v. The subgraph of T_M obtained by deleting e contains two components, T_u and T_v, containing u and v respectively. By relabelling as necessary, we will assume that T_v contains the root t. We let U_e be the set containing elements $z \in E(M)$ such that the path from $\varphi_M(z)$ to t contains the edge e. Let V_e be $E - U_e$. We will show that the equivalence relation $\sim_{U_e}$ induces at most $|Q|$ classes. Proposition 2.4 will then imply that $\mathcal{M}$ has decomposition-width at most $2^{|Q|}$. (Although (T_M, φ_M) is not a decomposition of M, it can easily be turned into one by contracting an edge incident with the root, and then forgetting the distinction between left and right children.)

Let Y and Y' be arbitrary subsets of U_e. Let r_1 and r'_1 be the runs of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y\})$ and $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y'\})$ respectively. We declare Y and Y' to be equivalent if and only if these runs apply the same singleton set to u; that is, if $r_1(u) = r'_1(u)$. It is clear that this is an equivalence relation on subsets of U_e with at most $|Q|$ equivalence classes, so it remains to show that this equivalence relation refines $\sim_{U_e}$. Assume that Y and Y' are equivalent subsets, and let Z be an arbitrary subset of V_e. Let r_2 and r'_2 be the runs of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y \cup Z\})$ and $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y' \cup Z\})$. Any leaf in T_u receives the same label in both $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y\})$ and $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y \cup Z\})$. Now it is easy to prove by induction that $r_1(w) = r_2(w)$ for all vertices w in T_u. Similarly, $r'_1(w) = r'_2(w)$ for all such w. In particular, $r_2(u) = r_1(u) = r'_1(u) = r'_2(u)$, where the middle equality is because of the equivalence of Y and Y'. Using the fact that $r_2(u) = r'_2(u)$, we can prove by induction that $r_2(w) = r'_2(w)$ for all vertices w in T_v. In particular, $r_2(t) = r'_2(t)$, so A accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y \cup Z\})$ if and only if it accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y' \cup Z\})$. This implies that $Y \cup Z$ is in $\mathcal{I}$ if and only if $Y' \cup Z$ is. Thus $\sim_{U_e}$ has at most $|Q|$ classes, as desired. □

The other direction is known to Král [17] and to Strozecki [23, 24].

Lemma 5.2. *Let $\mathcal{M}$ be a class of set-systems. If $\mathcal{M}$ has bounded decomposition-width, then it is automatic.*

Proof. Let K be an integer such that $\mathrm{dw}(M) \leq K$ for all members $M \in \mathcal{M}$. Thus, any member $\mathcal{M}$ has a decomposition such that each displayed set

contains at most K equivalence classes. We construct a tree automaton, A, that decides the formula $\mathrm{Ind}(X_i)$. The set of states of A is $Q = \{\mathsf{indep}, \mathsf{dep}, q_1, \ldots, q_K\}$.

Let $M = (E, \mathcal{I})$ be an arbitrary set-system in $\mathcal{M}$, and let (T, φ) be a decomposition of M, where $\sim_U$ has at most K equivalence classes for any set U displayed by an edge of T. We start by showing how to construct the parse tree (T_M, σ_M) by modifying T. First, we arbitrarily choose an edge of T, and subdivide it with the new vertex t, where t will be the root of T_M. For each non-leaf vertex of T, we make an arbitrary decision as to which of its children is the left child, and which is the right. This describes the tree T_M. The bijection φ_M is set to be identical to φ.

For each edge e, let U_e be the set of elements $z \in E$ such that the path from $\varphi_M(z)$ to t contains the edge e. Then $\sim_{U_e}$ induces at most K equivalence classes. Let ℓ_e be some function from the subsets of U_e into $\{q_1, \ldots, q_K\}$ such that $\ell_e(X) = \ell_e(X')$ implies $X \sim_{U_e} X'$. We think of ℓ_e as applying labels to the equivalence classes of $\sim_{U_e}$. (Although we allow the possibility that equivalent subsets under $\sim_{U_e}$ receive different labels under ℓ_e. In other words, the equivalence relation induced by ℓ_e refines $\sim_{U_e}$.) For each q_j in the image $\mathrm{Im}(\ell_e)$, we arbitrarily choose a representative subset $\mathrm{Rep}_e(q_j) \subseteq U_e$ such that $\ell_e(\mathrm{Rep}_e(q_j)) = q_j$.

Next we describe the function σ_M, which labels each vertex of T_M with a function. Let u be a leaf of T_M. We apply to u a function, f, whose domain is $\{0, 1\}$. In the case that u is also the root of T_M, we set $f(0)$ to be the symbol indep if $\emptyset$ is in $\mathcal{I}$, and otherwise we set $f(0)$ to be the symbol dep. Similarly, $f(1) = \mathsf{indep}$ if $\{\varphi_M^{-1}(u)\}$ is in $\mathcal{I}$, and otherwise $f(1) = \mathsf{dep}$. Now assume that u is a non-root leaf, and let e be the edge incident with u. Then $f(0)$ is the label $\ell_e(\emptyset)$, and $f(1)$ is $\ell_e(\{\varphi_M^{-1}(u)\})$.

Now let u be a non-leaf vertex. Let e_L and e_R be the edges joining u to its children. We apply a function, f, to u whose domain is $\mathrm{Im}(\ell_{e_L}) \times \mathrm{Im}(\ell_{e_R})$. Let (q_j, q_k) be in $\mathrm{Im}(\ell_{e_L}) \times \mathrm{Im}(\ell_{e_R})$, and assume $X_j \subseteq U_{e_L}$ is the representative $\mathrm{Rep}_{e_L}(q_j)$, while X_k is $\mathrm{Rep}_{e_R}(q_k)$. Assume that u is not the root, and let e be the first edge in the path from u to t. Then $f(q_j, q_k)$ is $\ell_e(X_j \cup X_k)$, for each such (q_j, q_k). Next assume that u is the root. Then $f(q_j, q_k)$ is indep if $X_j \cup X_k \in \mathcal{I}$, and otherwise $f(q_j, q_k) = \mathsf{dep}$.

Now we have completed our description of σ_M, which labels the vertices of T_M with functions. Therefore (T_M, σ_M) is a Σ-tree, where Σ is the alphabet of functions whose domain is either $\{0, 1\}$, or a member of $2^{\{q_1, \ldots, q_K\}} \times 2^{\{q_1, \ldots, q_K\}}$, and whose codomain is $\{\mathsf{indep}, \mathsf{dep}, q_1, \ldots, q_K\}$.

Our next task is to describe the automaton, A. As we have said, the state space is $Q = \{\mathsf{indep}, \mathsf{dep}, q_1, \ldots, q_K\}$. The alphabet is $\Sigma \cup \Sigma \times \{0, 1\}^{\{i\}}$, where Σ is the set of functions that we described in the previous paragraph. The only accepting state is indep. To define the transition rule δ_0, we consider the input (f, s), where f is a function from $\{0, 1\}$ into Q, and s is a function in $\{0, 1\}^{\{i\}}$. Then we define $\delta_0(f, s)$ to be $\{f(s(i))\}$. We set the output of δ_0 on

any input not yet described to be $\{\mathsf{dep}\}$. Now we consider the transition rule δ_2. Let f be a function whose domain is a member of $2^{\{q_1,\ldots,q_K\}} \times 2^{\{q_1,\ldots,q_K\}}$, Assume that (q_i, q_j) is in the domain of f. Then $\delta_2(f, q_i, q_j)$ is defined to be $\{f(q_i, q_j)\}$. If (q_i, q_j) is not in the domain of f, then we define $\delta_2(f, q_i, q_j)$ to be $\{\mathsf{dep}\}$. Indeed, we let the output of δ_2 be $\{\mathsf{dep}\}$ on any input not yet described. This completes our description of the automaton A. Note that it is deterministic.

Claim 5.2.1. Let Y_i be a subset of E. Let u be a non-root vertex of T_M, and let e be the first edge on the path from u to t. Let q be the state applied to u by the run of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. Then $(Y_i \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q)$.

Proof. Assume that u has been chosen so that it is as far away from t as possible, subject to the constraint that the claim fails for u. Let f be the function applied to u by the labelling σ_M.

First assume that u is a leaf, so that $U_e = \{\varphi_M^{-1}(u)\}$. Then u receives the label (f, s) in $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$, where $s(i)$ is 1 if $\varphi_M^{-1}(u) \in Y_i$, and is 0 otherwise. The construction of A means that $q = f(s(i))$. If $Y_i \cap U_e = \emptyset$, then $q = f(0) = \ell_e(\emptyset)$. Now $\ell_e(\mathrm{Rep}_e(q)) = q$, by definition, so $\mathrm{Rep}_e(q) \sim_{U_e} \emptyset$, by the nature of the function ℓ_e. Therefore $(Y_i \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q)$, as desired. The other possibility is that $Y_i \cap U_e = U_e = \{\varphi_M^{-1}(u)\}$. In this case $q = f(1) = \ell_e(U_e)$. Again $\mathrm{Rep}_e(q) \sim_{U_e} U_e$, and hence $(Y_i \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q)$.

Now we must assume that u is not a leaf, so that u is joined to its children, u_L and u_R, by the edges e_L and e_R. Assume that u receives the label f in $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. Let q_L and q_R be the states applied to u_L and u_R by the run of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. Our inductive assumption on u means that $(Y_i \cap U_{e_L}) \sim_{U_{e_L}} \mathrm{Rep}_{e_L}(q_L)$ and $(Y_i \cap U_{e_R}) \sim_{U_{e_R}} \mathrm{Rep}_{e_R}(q_R)$. Let X_j be $\mathrm{Rep}_{e_L}(q_L)$ and use X_k to denote $\mathrm{Rep}_{e_R}(q_R)$. Now Proposition 2.3 implies that $(Y_i \cap U_e) = (Y_i \cap U_{e_L}) \cup (Y_i \cap U_{e_R})$ is equivalent to $X_j \cup X_k$ under $\sim_{U_e}$. The construction of f and A means that $q = \ell_e(X_j \cup X_k)$. Obviously $\ell_e(\mathrm{Rep}_e(q)) = q$, so the nature of the function ℓ_e implies $(X_j \cup X_k) \sim_{U_e} \mathrm{Rep}_e(q)$. Now we see that $(Y_i \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q)$, so u fails to provide a counterexample after all. □

If the root, t, is a leaf, then A applies indep to t if and only if $Y_i \cap \{\varphi_M^{-1}(t)\} = Y_i$ is in $\mathcal{I}$. Assume that t is not a leaf, and that the edges e_L and e_R join t to its children, u_L and u_R. Let q_L and q_R be the states applied to u_L and u_R. Let X_j be $\mathrm{Rep}_{e_L}(q_L)$, and let X_k be $\mathrm{Rep}_{e_R}(q_R)$. Then $(Y_i \cap U_{e_L}) \sim_{U_{e_L}} X_j$ and $(Y_i \cap U_{e_R}) \sim_{U_{e_R}} X_k$, by Claim 5.2.1. If we apply Proposition 2.3 with $U = U_{e_L}$, $V = U_{e_R}$, and $W = \emptyset$, we see that both of $Y_i = (Y_i \cap U_{e_L}) \cup (Y_i \cap U_{e_R})$ and $X_j \cup X_k$ belong to $\mathcal{I}$, or neither does. In the former case, A applies indep to t during its run on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$, and hence accepts. In the latter case, A applies dep, and does not accept. Therefore A decides $\mathrm{Ind}(Y_i)$, exactly as we want. □

Recall that a class of matroids is pigeonhole if every subclass with bounded branch-width also has bounded decomposition-width. Now we can deduce the following (perhaps not obvious) fact.

Corollary 5.3. *Let $\mathcal{M}$ be a pigeonhole class of matroids. Then $\{M^* \colon M \in \mathcal{M}\}$ is pigeonhole.*

Proof. Assume that $\mathcal{M}$ is pigeonhole. For every positive integer, λ, there is an integer $\rho(\lambda)$ such that any matroid in $\mathcal{M}$ with branch-width at most λ has decomposition-width at most $\rho(\lambda)$.

Let λ be an arbitrary positive integer. Let $\mathcal{M}_\lambda$ be the class of matroid in $\mathcal{M}$ with branch-width at most λ. As $\mathcal{M}_\lambda$ has bounded decomposition-width, Lemma 5.2 implies that it is an automatic class. Let A' be a 1-ary automaton such that for every matroid $M \in \mathcal{M}_\lambda$, there is a parse tree (T_M, σ_M) and a bijection $\varphi_M \colon E(M) \to L(T_M)$ such that $M = M(A', T_M, \sigma_M, \varphi_M)$.

The predicate

$$\mathrm{Basis}(X_2) = \mathrm{Ind}(X_2) \wedge \forall X_3((\mathrm{Ind}(X_3) \wedge X_2 \subseteq X_3) \to X_3 \subseteq X_2)$$

is satisfied exactly by interpretations that take X_2 to a basis of a matroid. Similarly,

$$\mathrm{Coind}(X_1) = \exists X_2(\mathrm{Basis}(X_2) \wedge \neg\exists X_4(\mathrm{Sing}(X_4) \wedge X_4 \subseteq X_1 \wedge X_4 \subseteq X_2)$$

is satisfied exactly by the interpretations that take X_1 to coindependent sets. Now Lemma 4.7 implies that there is an automaton, A, that accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ if and only if Y_i is coindependent in M, for each $M \in \mathcal{M}_\lambda$. Therefore $M(A, T_M, \sigma_M, \varphi_M) = M^*$, so this establishes that $\{M^* \colon M \in \mathcal{M}_\lambda\}$ is an automatic class of matroids. Lemma 5.1 implies there is an integer $\rho^*(\lambda)$ such that $\mathrm{dw}(M^*) \leq \rho^*(\lambda)$ whenever M is in $\mathcal{M}_\lambda$.

The branch-width of a matroid is equal to the branch-width of its dual [20, Proposition 14.2.3]. Hence

$$\{M^* \colon M \in \mathcal{M},\ \mathrm{bw}(M^*) \leq \lambda\} = \{M^* \colon M \in \mathcal{M}_\lambda\}.$$

We have just shown that any matroid in this class has decomposition-width at most $\rho^*(\lambda)$, and this establishes the result. □

We do not know if we can prove Corollary 5.3 without relying on Theorem 1.4. Nor do we know if the dual of a strongly pigeonhole class must be strongly pigeonhole, but we conjecture that this is the case.

Conjecture 5.4. *Let $\mathcal{M}$ be a strongly pigeonhole class of matroids. Then $\{M^* \colon M \in \mathcal{M}\}$ is strongly pigeonhole.*

6. Complexity theory

In this section, we discuss complexity theoretical applications of tree automata. We start with a simple observation.

Proposition 6.1. *Let ψ be any sentence in MS_0. Let $\mathcal{M}$ be an automatic class of set-systems. There exists a Turing Machine which will take as input a parse tree for any set system $M = (E, \mathcal{I}) \in \mathcal{M}$ (relative to an automaton) and then test whether or not M satisfies ψ. The running time is $O(n)$, where $n = |E|$.*

Proof. Since $\mathcal{M}$ is automatic, we can assume that A' is a 1-ary tree automaton with alphabet $\Sigma \cup \Sigma \times \{0,1\}^{\{i\}}$, and for any $M = (E, \mathcal{I}) \in \mathcal{M}$ we let (T_M, σ_M) be a parse tree of M relative to A'. So there is a bijection $\varphi_M \colon E \to L(T_M)$ such that A' accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ if and only if $Y_i \in \mathcal{I}$. The proof of Lemma 4.7 is constructive, and shows us how to build an automaton, A, which will accept $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \emptyset)$ if and only if M satisfies ψ. This construction is done during pre-processing, so it has no impact on the running time. While A processes $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \emptyset)$, the computation that occurs at each node takes a constant amount of time. So the running time of A is proportional to the number of nodes. This number is $2n - 1$, so the result follows. □

Various models of matroid computation have been studied. Here, we will concentrate on classes of matroids that have compact descriptions.

Definition 6.2. Let $\mathcal{M}$ be a class of matroids. A *succinct representation* of $\mathcal{M}$ is a relation, Δ, from $\mathcal{M}$ into the set of finite binary strings. We write $\Delta(M)$ to indicate any string in the image of $M \in \mathcal{M}$. We insist that there is a polynomial p and a Turing Machine which, when given any input $(\Delta(M), X)$, where $M \in \mathcal{M}$ and X is a subset of $E(M)$, will return an answer to the question "Is X independent in M?" in time bounded by $p(|E(M)|)$.

Thus we insist that an independence oracle can be efficiently simulated using the output of a succinct representation. Note that the length $|\Delta(M)|$ can be no longer than $p(|E(M)|)$. Descriptions of graphic or finite-field representable matroids as graphs or matrices provide succinct representations.

Proposition 6.3. *Let $\mathcal{M}$ be a class of matroids with succinct representation Δ. There is a Turing Machine which, for any integer $\lambda > 0$, will take as input any $\Delta(M)$ for $M \in \mathcal{M}$ satisfying $\mathrm{bw}(M) \leq \lambda$, and return a branch-decomposition of M with width at most $3\lambda + 1$. The running time is $O(8^\lambda n^{3.5} p(n))$, where $n = |E(M)|$ and p is as in* Definition 6.2.

Proof. The proof of this proposition requires nothing more than an analysis of the proof of [19, Corollary 7.2], so we provide a sketch only. Let $M = (E, \mathcal{I})$ be a matroid in $\mathcal{M}$ with $\mathrm{bw}(M) \leq \lambda$. A *partial decomposition* of M consists of a ternary tree, along with a partition of E and a bijection from the blocks of this partition into the leaf-set of T. Each edge, e, of T partitions E into two sets, U_e and V_e, and the *width* of e is $r_M(U_e) + r_M(V_e) - r(M) + 1$. We start with a partial decomposition containing a single block, and successively partition blocks into two parts, until every block is a singleton set. This process therefore takes $n - 1$ steps. At each step, we ensure that each edge

has width at most $3\lambda+1$, so at the end of the process, we will have the desired decomposition. Assume that U is a block in the partition with $|U| > 1$. Let l be the leaf corresponding to U, and let e be the edge incident with l (if T is not a single vertex). Let V be $E - U$. We inductively assume that the weight of e is at most $3\lambda + 1$. If it is less than $3\lambda + 1$, then we arbitrarily choose an element $u \in U$, subdivide e and join a new leaf to this new vertex. We label the new leaf with $\{u\}$, and relabel the leaf corresponding to U with $U - \{u\}$. Therefore we can assume that the width of e is exactly $3\lambda+1$, and hence $\lambda_M(U) = 3\lambda$ (assuming T has more than one vertex).

We use the greedy algorithm to find an arbitrary basis, B, of M in $O(np(n))$ steps. For any subset $X \subseteq U$, define $\lambda_B(X)$ to be

$$r_M(X \cup (B - V)) + r_M(V \cup (B - X)) - |B - X| - |B - V| + 1.$$

Then $\lambda_B(X)$ is the rank function of a matroid on the ground set U [19, Propositions 4.1 and 7.1]. Let this matroid be M_B. The rank of M_B is $3\lambda + 1$. Finding the rank of $X \cup (B - V)$ takes $O(np(n))$ steps, using the greedy algorithm, and similarly for $V \cup (B - X)$ in M. By again using the greedy algorithm, we can find a basis, D, of M_B, in $O(np(n)+n^2p(n))$ steps.

Now we loop over all partitions of D into an ordered pair of two sets, (D_1, D_2). This takes $2^{3\lambda+1}$ steps. We let M_1 and M_2 be $M/D_1\backslash D_2$ and $M\backslash D_1/D_2$ respectively. The ranks of M_1 and M_2 can be found in $O(np(n))$ time, and it then takes $p(n)$ steps to test whether a subset is a basis of M_1 or M_2. Now it follows from [5, Theorem 4.1] that we can use an equivalent form of the matroid intersection algorithm to find a set, Z, satisfying $D_1 \subseteq Z \subseteq E - D_2$ that minimises $\lambda_M(Z)$. Furthermore, this can be done in $O(np(n) + n^{2.5}p(n))$ steps. If $\lambda_M(Z) + 1 \geq \min\{|D_1|, |D_2|\}$, then $\mathrm{bw}(M) \geq |D|/3 = \lambda + 1/3$ and we have a contradiction [19, Theorem 5.1]. Therefore $\lambda_M(Z) + 1 < \min\{|D_1|, |D_2|\}$. We subdivide e and attach a leaf to the new vertex. This leaf corresponds to the set $U \cap Z$, and we relabel l with the set $U - Z$. (If T has only one vertex, we simply create a tree with two vertices, and label these with $U \cap Z$ and $U - Z$.)

The proof of [19, Theorem 5.2] shows that the width of every edge in the new decomposition is at most $3\lambda + 1$, so we can reiterate this process until we have a branch decomposition. □

We wish to develop efficient model-checking algorithms for strongly pigeonhole matroid classes. We have to strengthen this condition somewhat, by insisting not only that there is a bound on the number of equivalence classes, but that we can efficiently compute a finer equivalence relation.

Definition 6.4. Let $\mathcal{M}$ be a class of matroids with a succinct representation Δ. Assume there is a constant, c, and that for every integer, $\lambda > 0$, there is an integer, $\pi(\lambda)$, and a Turing Machine, M_λ, with the following properties: M_λ takes as input any tuple of the form $(\Delta(M), U, X, X')$, where M is in $\mathcal{M}$, $U \subseteq E(M)$ satisfies $\lambda_M(U) \leq \lambda$, and X and X' are subsets of U. The

machine M_λ computes an equivalence relation, $\approx_U$, on the subsets of U, so that M_λ accepts $(\Delta(M), U, X, X')$ if and only if $X \approx_U X'$. Furthermore,

(i) $X \approx_U X'$ implies $X \sim_U X'$,
(ii) the number of equivalence classes under $\approx_U$ is at most $\pi(\lambda)$, and
(iii) M_λ runs in time bounded by $O(\pi(\lambda)|E(M)|^c)$.

Under these circumstances, we say that $\mathcal{M}$ is *computably pigeonhole* (relative to Δ).

It follows immediately that if a class of matroids is computably pigeonhole, then it is strongly pigeonhole. We will later see that many natural classes are computably pigeonhole.

Theorem 6.5. *Let $\mathcal{M}$ be a class of matroids with a succinct representation Δ. Assume that $\mathcal{M}$ is computably pigeonhole. Let λ be a positive integer. There is a Turing Machine which accepts as input any $\Delta(M)$ when $M \in \mathcal{M}$ satisfies* $\mathrm{bw}(M) \leq \lambda$*, and returns a parse tree for M (relative to an automaton). The running time is $O((8^\lambda n^{3.5} + \pi(3\lambda)^2)p(n) + \pi(3\lambda)^4 n^{c+1})$, where $n = |E(M)|$, p is as in* Definition 6.2*, and π and c are as in* Definition 6.4*.*

Proof. We start by applying Proposition 6.3 to obtain a branch-decomposition with width at most $3\lambda + 1$. Let T be the tree underlying this branch-decomposition, and let φ be the bijection from $E(M)$ to the leaves of T. We construct T_M by subdividing an edge of T with a root vertex, t, and distinguishing between left and right children. We let φ_M be φ. If U is a set displayed by an edge of T_M, then $\lambda_M(U) \leq 3\lambda$. We let K be $\pi(3\lambda)$, where π is the function provided by Definition 6.4.

From this point we closely follow the proof of Lemma 5.2. For each edge, e, in T_M, we perform the following procedure. Let u be the end-vertex of e that is further from t in T_M, and define U_e as in the proof of Lemma 5.2. We construct representative subsets, $\mathrm{Rep}_e(q)$, of U_e, where q is a label in $\{q_1, \ldots, q_K\}$, in such a way that distinct representative states are inequivalent under $\approx_{U_e}$. At the same time, we will construct a function, f, which will be applied to u by the labelling function σ_M.

First assume that u is a leaf, so that $U_e = \{\varphi_M^{-1}(u)\}$. The domain of f will be $\{0, 1\}$. As in Lemma 5.2, we must consider the case that u is also the root of T_M. In this case, we set $f(0)$ to be indep, and set $f(1)$ to be indep or dep depending on whether $\{\varphi_M^{-1}(u)\}$ is independent. Now assume that u is a non-root leaf. Let $\emptyset$ be $\mathrm{Rep}_e(q_1)$, and set $f(0)$ to be q_1. In $O(Kn^c)$ steps, we test whether $\{\varphi_M^{-1}(u)\} \approx_{U_e} \emptyset$. If so, then we set $f(1)$ to be q_1. Assuming that $\{\varphi_M^{-1}(u)\} \not\approx_{U_e} \emptyset$, we define $\mathrm{Rep}_e(q_2)$ to be $U_e = \{\varphi_M^{-1}(u)\}$, and we set $f(1)$ to be q_2.

Now assume that u is not a leaf. Let e_L and e_R be the edges joining u to its children. Recursively, we assume that $\mathrm{Rep}_{e_L}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_L}\}$, and $\mathrm{Rep}_{e_R}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_R}\}$. The function f will have domain $\{q_1, \ldots, q_{s_L}\} \times \{q_1, \ldots, q_{s_R}\}$. For each of the $O(K^2)$ pairs, (q_j, q_k), with $q_j \in \{q_1, \ldots, q_{s_L}\}$ and $q_k \in \{q_1, \ldots, q_{s_R}\}$,

we perform the following steps. Let X_j stand for $\mathrm{Rep}_{e_L}(q_j)$ and X_k stand for $\mathrm{Rep}_{e_R}(q_k)$. In time bounded by $O(K^2n^c)$, we check whether $X_j \cup X_k$ is equivalent under $\approx_{U_e}$ to any of the representative subsets of U_e that we have already constructed. If not, then we define $\mathrm{Rep}_e(q_l)$ to be $X_j \cup X_k$, where q_l is the first label in $\{q_1, \ldots, q_K\}$ that has not already been assigned to a representative subset of U_e. In this case, we set $f(q_j, q_k)$ to be q_l. However, if $X_j \cup X_k$ is equivalent under $\approx_{U_e}$ to a previously chosen representative, say $\mathrm{Rep}_e(q_m)$, then we set $f(q_j, q_k)$ to be q_m. Note that the number of edges in T_M is $2n-2$, so this entire procedure takes $O(n(K^4n^c))$ steps.

Finally, let the children of the root, t, be u_L and u_R, and assume that t is joined to these children by e_L and e_R. Assume $\mathrm{Rep}_{e_L}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_L}\}$, and $\mathrm{Rep}_{e_R}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_R}\}$. Again, f has domain $\{q_1, \ldots, q_{s_L}\} \times \{q_1, \ldots, q_{s_R}\}$. We define $f(q_j, q_k)$ to be indep if $\mathrm{Rep}_{e_L}(q_j) \cup \mathrm{Rep}_{e_R}(q_k)$ is independent, and we let $f(q_j, q_k)$ be dep otherwise. Constructing this function takes $O(K^2p(n))$ steps. Now we have completed the construction of the parse tree (T_M, σ_M), and we have done so in $O((8^\lambda n^{3.5} + K^2)p(n) + K^4n^{c+1})$ steps.

To complete the proof, we must check that (T_M, σ_M) genuinely behaves as a parse tree should. The automaton A is exactly as in Lemma 5.2. But the statement of Claim 5.2.1 still holds in this case, and can be proved by the same argument. There is one point which deserves some attention: with the notation as in the proof of Claim 5.2.1, the fact that the state q is applied to u means that $(X_j \cup X_k) \approx_{U_e} \mathrm{Rep}_e(q)$. But the definition of $\approx_{U_e}$ then implies $(X_j \cup X_k) \sim_{U_e} \mathrm{Rep}_e(q)$, and hence $(Y_i \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q)$, exactly as in Claim 5.2.1. The rest of the proof follows exactly as in Lemma 5.2. □

Now Theorem 1.3 follows immediately from Proposition 6.1 and Theorem 6.5.

6.1. **Automata and 2-sums.** In [10], we extend Hliněný's Theorem to the classes of bicircular matroids and H-gain-graphic matroids (where H is a finite group). In this section, we show that it suffices to show that the 3-connected H-gain-graphic (or bicircular) matroids form a computably pigeonhole class. Because our arguments here do not depend on the nature of bicircular or H-gain-graphic matroids, we operate at a greater level of generality.

Let M_1 and M_2 be matroids on the ground sets E_1 and E_2. Assume that $E_1 \cap E_2 = \{e\}$, where e is neither a loop nor a coloop in M_1 or in M_2. The *parallel connection*, $P(M_1, M_2)$, along the *basepoint* e, has $E_1 \cup E_2$ as its ground set. Let $\mathcal{C}_i$ be the family of circuits of M_i for $i = 1, 2$. The family of circuits of $P(M_1, M_2)$ is

$$\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{(C_1 - e) \cup (C_2 - e) \colon C_1 \in \mathcal{C}_1,\ C_2 \in \mathcal{C}_2,\ e \in C_1 \cap C_2\}.$$

Note that $P(M_1, M_2)|E_i = M_i$, for $i = 1, 2$. The 2-*sum* of M_1 and M_2, written $M_1 \oplus_2 M_2$, is defined to be $P(M_1, M_2)\backslash e$.

Let T be a tree, where each node, x, is labelled with a matroid, M_x. Let the edges of T be labelled with distinct elements, $e_1, \ldots, e_m$. Let x and y be distinct nodes. We insist that if x and y are not adjacent, then $E(M_x)$ and $E(M_y)$ are disjoint. If x and y are joined by the edge e_i, then $E(M_x) \cap E(M_y) = \{e_i\}$, where e_i is neither a loop nor a coloop in M_x or M_y. Such a tree describes a matroid, as we now show. If e_i is an edge joining x to y, then contract e_i from T, and label the resulting identified node with $P(M_x, M_y)$. Repeat this procedure until there is only one node remaining. We use $P(T)$ to denote the matroid labelling this one node. It is an easy exercise to see that $P(T)$ is well-defined, so that it does not depend on the order in which we contract the edges of T. We define $\oplus_2(T)$ to be $P(T)\backslash\{e_1, \ldots, e_m\}$. If M is a connected matroid, there exists a (not necessarily unique) tree T satisfying $M = \oplus_2(T)$ where every node of the tree is labelled with a 3-connected matroid.

Definition 6.6. Let Δ be a succinct representation of $\mathcal{M}$, a class of matroids. We say that Δ is *minor-compatible* if there is a polynomial-time algorithm which will accept any tuple $(\Delta(M), X, Y)$ when $M \in \mathcal{M}$ and X and Y are disjoint subsets of $E(M)$, and return a string of the form $\Delta(M/X\backslash Y)$.

Theorem 6.7. *Let $\mathcal{M}$ be a minor-closed class of matroids with a minor-compatible representation, Δ. Assume that $\{M \in \mathcal{M} \colon M$ is 3-connected$\}$ is computably pigeonhole. There is a fixed-parameter tractable algorithm (with respect to the parameter of branch-width) which accepts as input any $\Delta(M)$ when $M \in \mathcal{M}$ and returns a parse tree for M.*

Remark 6.8. Theorems 6.5 and 6.7 are independent of each other, as we now discuss. Since any subclass of a computably pigeonhole class is computably pigeonhole, we can easily construct a computably pigeonhole class that is not minor-closed, and this class will therefore not be covered by Theorem 6.7. On the other hand, we can construct a minor-closed class $\mathcal{M}$ such that $\{M \in \mathcal{M} \colon M$ is 3-connected$\}$ is computably pigeonhole, and yet $\mathcal{M}$ is not even strongly pigeonhole. Such a class will be covered by Theorem 6.7, but not Theorem 6.5. For an example, let $n \geq 3$ be an integer, and let $U^+_{2,n}$ be the rank-2 matroid obtained from $U_{2,n}$ by replacing each element with a parallel pair. If U contains exactly one element from each parallel pair, then it is 3-separating, and yet $\sim_U$ has at least n equivalence classes. So if $\mathcal{M}$ is the smallest minor-closed class containing $\{U^+_{2,n} \colon n \geq 3\}$, then $\mathcal{M}$ is not strongly pigeonhole. However, every 3-connected member of $\mathcal{M}$ is uniform. It is therefore not difficult to show that $\{M \in \mathcal{M} \colon M$ is 3-connected$\}$ is computably pigeonhole with respect to any minor-compatible representation. (See the proof of [10, Proposition 3.5].)

Proof of Theorem 6.7. The ideas here are very similar to those in the proof of Theorem 6.5, but there are several technical complications introduced by the fact that we have to deal with non-3-connected matroids as a separate

case. Let $M \in \mathcal{M}$ be a matroid with ground set E and branch-width λ. We assume that we are given the description $\Delta(M)$. The algorithm we describe in this proof runs in polynomial-time, and to ensure that it is fixed-parameter tractable with respect to λ, we will be careful to observe that whenever we call upon a polynomial-time subroutine, λ does not appear in the exponent of the running time.

To start with we consider the case that M is connected, and at the end of the proof we will show that this is sufficient to establish the entire theorem.

Discussion in [1] shows that we can use a 'shifting' algorithm to find a 2-separation of M, if it exists. This takes $O(|E|^3)$ oracle calls. Therefore we can test whether M is 3-connected in polynomial time. If M is 3-connected, then we use Theorem 6.5 to construct a parse tree for M. So henceforth we assume that M is connected but not 3-connected.

Constructing T_M. We have noted that it takes $O(|E|^3)$ oracle calls to find a 2-separation of M. Assume that (U_1, U_2) is such a separation. Then M can be expressed as the 2-sum of matroids M_1 and M_2, where the ground set of M_i is $U_i \cup e$, and e is an element not in E. Both M_1 and M_2 are isomorphic to minors of M, and hence are in $\mathcal{M}$. If B_1 is a basis of $M|U_1$, and B is a basis of M containing B_1, then $B \cap U_2$ does not span U_2, so we can let x be an element in $U_2 - \mathrm{cl}_M(B \cap U_2)$. Now M_1 can be produced from M by contracting $B \cap U_2$ and deleting all elements in $U_2 - (B \cup x)$. We then relabel x as e. From this discussion, and the fact that Δ is minor-compatible, it follows that we can construct $\Delta(M_1)$ and $\Delta(M_2)$ in polynomial time. By reiterating this procedure, we can construct a labelled tree, T', such that $M = \oplus_2(T')$. Each node, x, of T' is labelled by a 3-connected matroid, M_x, with at least three elements, and for each such node we have an associated string $\Delta(M_x)$. Let the edge labels of T' be $e_1, \ldots, e_m$. We arbitrarily select e_m. Say that it joins x_L to x_R in T'. We delete e_m, add a new node, t, and edges $e_{m,L}$ and $e_{m,R}$ joining t to x_L and x_R. At the same time, we relabel e_m as $e_{m,L}$ in M_{x_L} and as $e_{m,R}$ in M_{x_R}. Let T be the tree that we obtain in this way. We think of t as being the *root* of T. We associate t with the matroid M_t, which is a copy of $U_{1,2}$ with ground set $\{e_{m,L}, e_{m,R}\}$. Note that $\oplus_2(T) = \oplus_2(T') = M$.

For each non-root node, x, of T, the labelling matroid M_x is isomorphic to a minor of M. Therefore $\mathrm{bw}(M_x) \leq \lambda$ [20, Proposition 14.2.3]. We use Proposition 6.3 to construct a branch-decomposition of M_x with width at most $3\lambda + 1$. Let T_x be the tree underlying the branch-decomposition of M_x, and let φ_x be the bijection from $E(M_x)$ to the leaves of T_x. We define the tree T_t to be a path of two edges, and we define φ_t so that it applies the labels $e_{m,L}$ and $e_{m,R}$ to the leaves and t to the middle vertex. We say that $\varphi_t(e_{m,L})$ is the *left child* of t and $\varphi_t(e_{m,R})$ is the *right child*.

Let x be a non-root node in T and consider the path in T from x to t. Let e_α be the first edge in this path. Then we say that e_α is the *parent basepoint* of T_x. For each internal vertex, u, of T_x, note that u is adjacent to two vertices that are not in the path from u to $\varphi_x(e_\alpha)$, where e_α is the

parent basepoint of T_x. We say that these two vertices are the *children* of u, and we make an arbitrary distinction between the *left child* and the *right child*.

The collection $\cup\{T_x\}$, where x ranges over all nodes of T, forms a forest that we now assemble into a single tree, T_M. For each edge, e_α, in $\{e_1, \ldots, e_{m-1}, e_{m,L}, e_{m,R}\}$, we perform the following operation. Let the node x of T be chosen so that e_α is the parent basepoint of T_x, and let y be the other end-vertex of e_α in T. Let u be the vertex of T_x that is adjacent to the leaf $\varphi_x(e_\alpha)$. We delete $\varphi_x(e_\alpha)$ from T_x and then identify u with the leaf $\varphi_y(e_\alpha)$ in T_y. We say that the edge in T_y that is now incident with u is a *basepoint* edge in T_M. If u is a non-leaf vertex of T_x, we allow u to carry its children over from T_x to T_M. In the case that a child of u in T_x represents a basepoint element, e_α, then that child of u in T_M will be an internal vertex of another tree, T_y. Now T_M is a rooted tree where every non-leaf vertex has a left child and a right child. Figure 1 illustrates this construction by showing the tree T', along with the collection of decompositions $\cup\{T_x\}$. In these diagrams, the basepoints of the parallel connections are coded via colour. In Figure 2, we have assembled these trees together into the tree T_M.

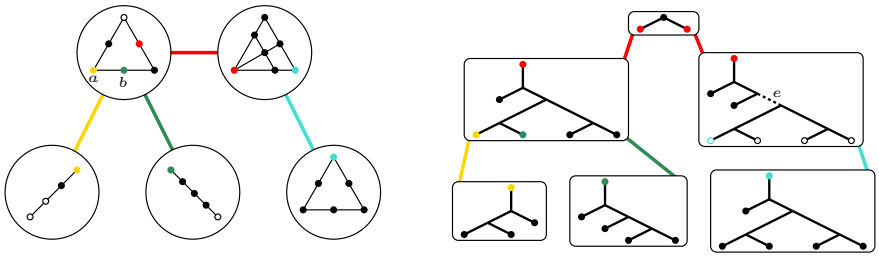

FIGURE 1. The decomposition tree, T', and the decompositions $\cup\{T_x\}$.

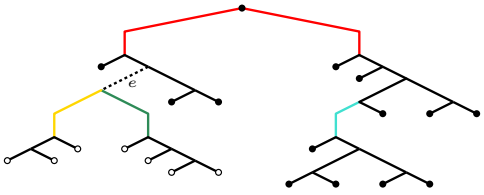

FIGURE 2. The tree T_M.

Every edge of T_M is an edge of exactly one tree T_x, where x is a node of T. Our method of construction means that if u is a non-leaf vertex of T_M, then both the edges joining u to its children are edges of the same tree T_x. Moreover, if x is a non-root node of T, then the only edge of T_x not contained in T_M is the one incident with $\varphi_x(e_\alpha)$, where e_α is the parent basepoint of T_x. Let e be any edge of T_M, and let u be the end-vertex of e

that is further away from t in T_M. Then we say that u is the *bottom* vertex of e.

Note that there is a bijection, φ_M, from E to the leaves of T_M. In particular, φ_M restricted to $E(M_x) \cap E$ is equal to φ_x restricted to the same set, for any node x. It is easy to check that if the set U is displayed by an edge of T_M, then $\lambda_M(U) \leq 3\lambda$. This is obvious when U is a subset of $E(M_x)$, where x is a non-root node of T, for then U is also displayed by the tree T_x. It is only a little more difficult to verify when U is not contained in $E(M_x)$ for any x.

Defining three pieces of notation. Next we describe three related notations for subsets of E and $E(M_x)$. Let e be any edge of T_M, and let $\mathrm{desc}(e)$ be the set of elements $z \in E$ such that the path in T_M from $\varphi_M(z)$ to t passes through e. In Figure 2, when e is the dashed edge, the set $\mathrm{desc}(e)$ is indicated by the hollow vertices. Note that $\mathrm{desc}(e)$ is not necessarily contained in $E(M_x)$ for any node x of T, but that it is contained in E.

Next, we let e be any edge of T_M and let x be the node of T such that e is an edge of the tree T_x. If x is a non-root node of T, then we can assume that e_α is the parent basepoint of T_x. We let U_e be the set of elements $z \in E(M_x)$ such that the path in T_x from $\varphi_x(z)$ to $\varphi_x(e_\alpha)$ contains e. If x is the root t, and e joins t to $\varphi_t(e_{m,L})$ then we define U_e to be $\{e_{m,L}\}$, and if e joins t to $\varphi_t(e_{m,R})$, then we define U_e to be $\{e_{m,R}\}$. In Figure 1, the righthand diagram contains a dashed edge e, and the set U_e is indicated by hollow vertices. Unlike $\mathrm{desc}(e)$, the set U_e may not be contained in E, as it may contain basepoint elements. As U_e is displayed by the edge e in T_x, we have that $\lambda_{M_x}(U_e) \leq 3\lambda$.

Finally, we let Y_i be any subset of E, and we let x be a node of T. We recursively describe a subset, $\langle Y_i \rangle_x \subseteq E(M_x)$. First, assume that x is a leaf of T. Then $\langle Y_i \rangle_x$ is simply $Y_i \cap E(M_x)$. Now we assume that x is not a leaf. Let $e_{\alpha_1}, \ldots, e_{\alpha_s}$ be the labels of edges in T that are incident with x but not on the path from x to t. Now define $\langle Y_i \rangle_x$ so that it contains $Y_i \cap E(M_x)$, along with any basepoint e_{α_j} such that if y labels the other node incident with e_{α_j}, then $e_{\alpha_j} \in \mathrm{cl}_{M_y}(\langle Y_i \rangle_y)$. Note that this means that any element of $\langle Y_i \rangle_x$ is contained in either $Y_i \cap E(M_x)$ or is a basepoint element. For example, in Figure 1, we let x be the top-left node in the lefthand diagram, and we let Y_i be the set indicated by the hollow vertices. Then $\langle Y_i \rangle_x$ contains the single hollow vertex in $E(M_x)$, as well as the element a, but *not* the element b. Note that by construction, $(Y_i \cap E(M_x)) \subseteq \langle Y_i \rangle_x \subseteq E(M_x)$.

With these definitions established, we can proceed.

Constructing representative subsets. Let e be any edge of T_M, and let $U_e \subseteq E(M_x)$ be as defined above. Recall that $\lambda_{M_x}(U_e) \leq 3\lambda$. Noting that $\{M \in \mathcal{M} \colon M \text{ is 3-connected}\}$ is computably pigeonhole and M_x is 3-connected, we refer to Definition 6.4, and we let π be the function from that definition. Let K be $\pi(3\lambda)$, and note that K is constant with respect to the size of E. Let $\approx_{U_e}$ be the equivalence relation from Definition 6.4. Then we can decide whether two subsets of U_e are equivalent under $\approx_{U_e}$ in time

bounded by $O(K|E(M)|^c)$. Furthermore, $\approx_{U_e}$ has at most K equivalence classes.

Note that T_M has exactly $2|E| - 2$ edges. For each such edge, e, we will construct, in polynomial time, a set of representatives such that each representative is a subset of U_e. Each representative will be an independent subset of U_e, and distinct representatives will represent different $(\approx_{U_e})$-classes. We will apply the labels $q_1, \ldots, q_K$ to representative subsets, and use $\mathrm{Rep}_e(q)$ to denote the representative with label q, assuming that it exists. We do not claim that our set of representatives is complete, so there may be $(\approx_{U_e})$-classes that do not have a representative.

Let x be the node of T such that e is an edge of T_x. If $x \neq t$, then T_x has a parent basepoint, e_α, and we let u be the end-vertex of e that is further from $\varphi_x(e_\alpha)$ in T_x. If $x = t$, then let u be the end-vertex of e that is not the root.

First assume that u is a leaf in T_x. Then $U_e = \{\varphi_x^{-1}(u)\}$. In this case, we choose $\emptyset$ as a representative, and apply the label q_1 to it, so that $\emptyset = \mathrm{Rep}_e(q_1)$. Because $\mathcal{M}$ has a succinct representation, we can check in polynomial time whether U_e is dependent. If so, we take no further action, so assume that U_e is independent in M_x. In polynomial time we can check whether $U_e \approx_{U_e} \emptyset$ holds. If so, then we are done. If $U_e \not\approx_{U_e} \emptyset$, then we choose U_e as the representative with label q_2, so that $U_e = \mathrm{Rep}_e(q_2)$.

Now we assume that u is not a leaf of T_x. Let e_L and e_R be the edges joining u to its children in T_x. Recursively, we assume that we have chosen a representative subset $\mathrm{Rep}_{e_L}(q) \subseteq U_{e_L}$ whenever q is in $\{q_1, \ldots, q_{s_L}\}$, and that $\mathrm{Rep}_{e_R}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_R}\}$. For each pair (q_j, q_k) in $\{q_1, \ldots, q_{s_L}\} \times \{q_1, \ldots, q_{s_R}\}$, we let X_j stand for $\mathrm{Rep}_{e_L}(q_j)$ and X_k stand for $\mathrm{Rep}_{e_R}(q_k)$. If $X_j \cup X_k$ is dependent in M_x, then we move to the next pair. Assuming that $X_j \cup X_k$ is independent, we check in polynomial time whether $X_j \cup X_k$ is equivalent under $\approx_{U_e}$ to any of the representative subsets of U_e that we have already constructed. If so, we are done. If not, then we let q_l be the first label in $\{q_1, \ldots, q_K\}$ not already assigned to a subset of U_e, and we define $\mathrm{Rep}_e(q_l)$ to be $X_j \cup X_k$.

Labelling the vertices. The alphabet of our automaton is going to contain a set of functions. Our next job is show how we apply, in polynomial time, functions in the alphabet to the vertices of T_M. Let u be a vertex of T_M, and assume that u is the bottom vertex of the edge e. Let x be the node of T such that e is an edge in the tree T_x.

First, we assume that u is a leaf of T_M. Then $U_e = \varphi_M^{-1}(u)$. We label u with a function, f, with the domain $\{0, 1\}$. Set $f(0)$ to be q_1, recalling that $\emptyset$ is the representative $\mathrm{Rep}_e(q_1)$. If U_e is dependent in M_x, then we set $f(1)$ to be the symbol dep. Assume that U_e is independent. If $U_e \approx_{U_e} \emptyset$, then we set $f(1)$ to be q_1. Otherwise we set $f(1)$ to be q_2, recalling that in this case $U_e = \mathrm{Rep}_e(q_2)$. Henceforth we assume that u is not a leaf of T_M. Let e_L and e_R be the edges joining u to its children in T_M.

Assume that e is not a basepoint edge. This implies that U_e is the disjoint union of U_{e_L} and U_{e_R}. (If e were a basepoint edge, then U_e would be a singleton subset of $E(M_x)$, whereas U_{e_L} and U_{e_R} would be subsets of $E(M_y)$ for some other node y of T.) Assume that $\mathrm{Rep}_{e_L}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_L}\}$, and that $\mathrm{Rep}_{e_R}(q)$ is defined when q is in $\{q_1, \ldots, q_{s_R}\}$. In this case, we label u with a function, f, having $\{\mathsf{dep}, q_1, \ldots, q_{s_L}\} \times \{\mathsf{dep}, q_1, \ldots, q_{s_R}\}$ as its domain. We define the output of f to be dep on any input that includes the symbol dep. Now assume that X_j is $\mathrm{Rep}_{e_L}(q_j)$ and X_k is $\mathrm{Rep}_{e_R}(q_k)$, for some $1 \leq j \leq s_L$ and some $1 \leq k \leq s_R$. If $X_j \cup X_k$ is dependent in M_x, then define $f(q_j, q_k)$ to be dep. Otherwise, $X_j \cup X_k$ is equivalent under $\approx_e$ to some representative subset of U_e. We find this representative, say $\mathrm{Rep}_e(q_l)$, in polynomial time, and we set $f(q_j, q_k)$ to be q_l.

Now we assume that e is a basepoint edge. Assume that in T_x, e is incident with the leaf $\varphi_x(e_\alpha)$, where e_α is in $\{e_1, \ldots, e_{m-1}, e_{m,L}, e_{m,R}\}$. Therefore $U_e = \{e_\alpha\}$. Note that e_L and e_R are edges of T_y, where y is the node of T joined to x by e_α. Assume that $\mathrm{Rep}_{e_L}(q)$ has been chosen when $q \in \{q_1, \ldots, q_{s_L}\}$, and $\mathrm{Rep}_{e_R}(q)$ is defined when $q \in \{q_1, \ldots, q_{s_R}\}$. We will apply to u a function, f, whose domain is again $\{\mathsf{dep}, q_1, \ldots, q_{s_L}\} \times \{\mathsf{dep}, q_1, \ldots, q_{s_R}\}$, and whose codomain is $\{\mathsf{dep}, q_1, q_2\}$. The output of f is dep on any input including dep. Consider the input (q_j, q_k). Let X_j and X_k be $\mathrm{Rep}_{e_L}(q_j)$ and $\mathrm{Rep}_{e_R}(q_k)$ respectively. If $X_j \cup X_k$ is dependent, then we set $f(q_j, q_k)$ to be dep. Now we assume that $X_j \cup X_k$ is independent. If $X_j \cup X_k \cup \{e_\alpha\}$ is independent in M_y then we set $f(q_j, q_k)$ to be q_1. Otherwise, $X_j \cup X_k \cup \{e_\alpha\}$ is dependent in M_y, and we set $f(q_j, q_k)$ to be q_1 if $U_e \approx_{U_e} \emptyset$ holds, and q_2 if $U_e \not\approx_{U_e} \emptyset$.

Finally, the root t is labelled with a function, f, that takes $\{\mathsf{dep}, q_1, q_2\}^2$ as input. Any ordered pair that contains dep produces dep as output. Similarly, $f(q_2, q_2) = \mathsf{dep}$. Any other ordered pair produces the symbol indep as output.

Now we have described the function that we apply to each vertex of T_M. Let σ_M be the labelling that applies these functions. Thus (T_M, σ_M) is a Σ-tree, where Σ contains functions whose domain is either $\{0, 1\}$ or sets of the form $\{\mathsf{dep}, q_1, \ldots, q_{s_L}\} \times \{\mathsf{dep}, q_1, \ldots, q_{s_R}\}$, and whose codomain is $\{\mathsf{dep}, \mathsf{indep}, q_1, \ldots, q_K\}$.

Constructing the automaton. Now that we have shown how to efficiently construct the Σ-tree (T_M, σ_M), it is time to consider the workings of the automaton, A. The state space, Q, of A is the set $\{\mathsf{dep}, \mathsf{indep}, q_1, \ldots, q_K\}$. The alphabet is $\Sigma \cup \Sigma \times \{0, 1\}^{\{i\}}$, where Σ is the set of functions into Q that we have previously described. The only accepting state is indep. The transition rule, δ_0, acts as follows. If f is a function from $\{0, 1\}$ into Q, and s is a function in $\{0, 1\}^{\{i\}}$, then $\delta_0(f, s) = \{f(s(i))\}$. Similarly, δ_2 is defined so that if f is a function in Σ, and (α, β) is in the domain of f, then $\delta_2(f, \alpha, \beta) = \{f(\alpha, \beta)\}$. If (α, β) is not in the domain of f, then we set $\delta_2(f, \alpha, \beta)$ to be $\{\mathsf{dep}\}$. Indeed, any output of δ_0 or δ_2 that has

not now been defined, we set to be {dep}. This completes the description of A. Note that it is a deterministic automaton.

Proof of correctness. We must now prove that (T_M, σ_M) truly is a parse tree relative to the automaton A. That is, we must prove that A accepts a subset of the leaves of T_M if and only if the corresponding set is independent in M.

Lemma 6.9. *Let Y_i be a subset of E, and let u be a non-leaf vertex of T_M. Let e_L and e_R be the edges of T_M joining u to its children. Let y be the node of T such that e_L and e_R are edges of T_y.*

(i) *If $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$ is dependent in M_y, then $Y_i \cap (\mathrm{desc}(e_L) \cup \mathrm{desc}(e_R))$ is dependent in M.*

(ii) *If $Y_i \cap \mathrm{desc}(e_L)$ and $Y_i \cap \mathrm{desc}(e_R)$ are independent in M, but $Y_i \cap (\mathrm{desc}(e_L) \cup \mathrm{desc}(e_R))$ is dependent, then $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$ is dependent in M_y.*

Proof. We start by defining D, a set of nodes in T. Let y' be a node in T. If there exists $d \in \mathrm{desc}(e_L) \cup \mathrm{desc}(e_R)$ such that the path in T_M from $\varphi_M(d)$ to u uses an edge in the tree $T_{y'}$, then y' is in D, and otherwise $y' \notin D$. If $y' \neq y$ and d is in $\mathrm{desc}(e_L)$, we say y' is a *left* vertex, if d is in $\mathrm{desc}(e_R)$, then y' is a *right* vertex. We say that y is both a left and a right vertex of D, and note that any vertex in $D - y$ is either left or right, but not both. Let $y_0, \ldots, y_s$ be an ordering of the vertices in D such that $y_0 = y$, and whenever y_k is on the path from y_j to y in T, $k \leq j$.

To prove (i), we let C be a circuit of M_y contained in $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$. We will construct a sequence of circuits, $C_0, \ldots, C_s$, of $P(T)$ such that:

(a) C_j is contained in

$$(\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})) \cup \bigcup_{k=1}^{j} \langle Y_i\rangle_{y_k}$$

for each j, and

(b) if $0 \leq k \leq j$, and e' is an basepoint edge on the path from y_k to y in T, then $e' \notin C_j$.

Assume we succeed in constructing this sequence. Then C_s does not contain any element in $\{e_1, \ldots, e_{m-1}, e_{m,L}, e_{m,R}\}$, so it is a circuit of $P(T)\backslash\{e_1, \ldots, e_{m-1}, e_{m,L}, e_{m,R}\} = M$, and is contained in $Y_i \cap (\mathrm{desc}(e_L) \cup \mathrm{desc}(e_R))$.

For C_0, we can just use C. Assume we have constructed C_{j-1}. Let e_α be the parent basepoint of T_{y_j}, so that e_α is the first edge on the path in T from y_j to y. Assume that e_α joins y_j to y_k, where $k < j$. If $e_\alpha \notin C_{j-1}$, then we set C_j to be C_{j-1} and we are done. Therefore we assume that e_α is in C_{j-1}, and hence e_α is in $\langle Y_i\rangle_{y_k}$, by the inductive assumption. By the definition of $\langle Y_i\rangle_{y_k}$, this means that e_α is in $\mathrm{cl}_{M_{y_j}}(\langle Y_i\rangle_{y_j})$. Let C' be a circuit of M_{y_j} such that $e_\alpha \in C' \subseteq (\langle Y_i\rangle_{y_j} \cup e_\alpha)$. The definition of the parallel connection

means that $(C_{j-1} - e_\alpha) \cup (C' - e_\alpha)$ is a circuit of $P(T)$, so we set C_j to be this circuit, and we are done.

Now we prove (ii). Assume that $Y_i \cap \mathrm{desc}(e_L)$ and $Y_i \cap \mathrm{desc}(e_R)$ are independent in M, but that C is a circuit contained in $Y_i \cap (\mathrm{desc}(e_L) \cup \mathrm{desc}(e_R))$. We construct a sequence $C_s, C_{s-1}, \ldots, C_0$ of circuits of $P(T)$ such that:

(a) C_j is contained in

$$(\langle Y_i \rangle_y \cap (U_{e_L} \cup U_{e_R})) \cup \bigcup_{k=1}^{j} \langle Y_i \rangle_{y_k}$$

for each j, and

(b) for each j, there is a left vertex y_L and a right vertex y_R such that C_j contains elements of both $\langle Y_i \rangle_{y_L}$ and $\langle Y_i \rangle_{y_R}$.

Assuming we succeed in constructing this sequence, C_0 will certify that $\langle Y_i \rangle_y \cap (U_{e_L} \cup U_{e_R})$ is dependent.

Note that C is contained in neither $Y_i \cap \mathrm{desc}(e_L)$ nor $Y_i \cap \mathrm{desc}(e_R)$. From this it follows that we can take C_s to be C. Now assume that we have constructed C_j. If C_j contains no elements of $\langle Y_i \rangle_{y_j}$, then we set C_{j-1} to be C_j. So assume that $C_j \cap \langle Y_i \rangle_{y_j} \neq \emptyset$. Let e_α be the parent basepoint of T_{y_j}, and assume that e_α joins y_j to y_k in T, where $j > k$. It cannot be the case that C_j is a circuit of $\langle Y_i \rangle_{y_j}$, or else condition (b) would be violated. Therefore C_j can be expressed as $(C' - e_\alpha) \cup (C_{j-1} - e_\alpha)$, where C' and C_{j-1} are circuits of $P(T)$ containing e_α, and C' is a circuit of M_{y_j}, while C_{j-1} intersects $E(M_{y_j})$ only in e_α. Note that the circuit C' implies that e_α is in $\langle Y_i \rangle_{y_k}$. If y_j is a left vertex, then so is y_k, so C_{j-1} must also contain an element from $\langle Y_i \rangle_{y_R}$, where y_R is some right vertex. Therefore C_{j-1} is the desired next circuit in the sequence. The symmetric argument applies when y_j and y_k are both right vertices. □

Lemma 6.10. *Let Y_i be a subset of $E(M)$. Assume that u is the bottom vertex of the edge e in T_M. Let x be the node of T such that e is an edge of T_x. Let q be the state applied to u by the run of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. Then $q = \mathsf{dep}$ if and only if $Y_i \cap \mathrm{desc}(e)$ is dependent in M. If $Y_i \cap \mathrm{desc}(e)$ is independent, then $q = q_l$ for some value l, and $(\langle Y_i \rangle_x \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q_l)$.*

Proof. We assume that the Lemma 6.10 fails for the vertex u, and that subject to this constraint, u has been chosen so that it is as far away from t as is possible in T_M. Let f be the function applied to u by the labelling σ_M.

Claim 6.10.1. u is not a leaf of T_M.

Proof. Let us assume that u is a leaf. Note that $\mathrm{desc}(e) = U_e = \{\varphi_x^{-1}(u)\}$. The label applied to u in the Σ-tree $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ is (f, s), where $s \in \{0,1\}^{\{i\}}$ is the function such that $s(i) = 1$ if $\varphi_x^{-1}(u)$ is in Y_i, and otherwise $s(i) = 0$. We have defined A in such a way that $q = f(s(i))$.

Assume that $Y_i \cap \operatorname{desc}(e)$ is dependent. The only way this can occur is if $\varphi_x^{-1}(u)$ is a loop contained in Y_i. In this case $q = f(s(i)) = f(1)$, and $f(1) = \mathsf{dep}$, by the construction of f. Therefore u does not provide a counterexample to Lemma 6.10. Next we assume that $q = \mathsf{dep}$. But q is $f(s(i))$, and this takes the value dep only if $s(i) = 1$ and $Y_i \cap \operatorname{desc}(e) = U_e$, and furthermore, this set is dependent. Again, u does not provide a counterexample.

Now we assume that $Y_i \cap \operatorname{desc}(e)$ is independent. Observe that e is not a basepoint edge, as this would imply $|\operatorname{desc}(e)| \geq 2$, and this is not the case when u is a leaf. From this we deduce that $\langle Y_i \rangle_x \cap U_e = Y_i \cap U_e$. Assume that $Y_i \cap U_e = \emptyset$. Then $q = f(s(i)) = f(0) = q_1$, where $\operatorname{Rep}_e(q_1)$ is the empty set. Thus $\langle Y_i \rangle_e \cap U_e$ and $\operatorname{Rep}_e(q)$ are actually equal, and thus certainly equivalent under $\sim_{U_e}$, as desired. Now we assume that $U_e \subseteq Y_i$, so $\langle Y_i \rangle_x \cap U_e = U_e$. Then $q = f(s(i)) = f(1)$, and this value is either q_1 or q_2. In the former case, $U_e \approx_{U_e} \emptyset$, so $(\langle Y_i \rangle_x \cap U_e) \approx_{U_e} \emptyset$. As $\emptyset$ is $\operatorname{Rep}_e(q_1)$, we are done. Therefore we consider the case that $q = q_2$. In this case $\operatorname{Rep}_e(q_2) = U_e = \langle Y_i \rangle_x \cap U_e$, so Lemma 6.10 holds. This contradiction means that Claim 6.10.1 is proved. □

Now we let u_L and u_R be the children of u in T_M, and we assume that these are the bottom vertices of the edges e_L and e_R. Note that $\operatorname{desc}(e)$ is the disjoint union of $\operatorname{desc}(e_L)$ and $\operatorname{desc}(e_R)$. Observe also that e_L and e_R are edges of the same tree, T_y, where y is a node of T that may or may not be equal to x. If $y \neq x$, then e is a basepoint edge. Let q_L and q_R be the states applied to u_L and u_R by the run of A on $\operatorname{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. Our inductive assumption on u means that Lemma 6.10 holds for u_L and u_R.

Claim 6.10.2. *$Y_i \cap \operatorname{desc}(e_L)$ and $Y_i \cap \operatorname{desc}(e_R)$ are independent in M, and $\langle Y_i \rangle_y \cap U_{e_L}$ and $\langle Y_i \rangle_y \cap U_{e_R}$ are independent in M_y.*

Proof. If $Y_i \cap \operatorname{desc}(e_L)$ is dependent, then so is $Y_i \cap \operatorname{desc}(e)$. In this case the inductive assumption tells us that $q_L = \mathsf{dep}$. Now the construction of f and A means that $q = \mathsf{dep}$. But this means that u does not provide us with a counterexample. Hence $Y_i \cap \operatorname{desc}(e_L)$, and symmetrically $Y_i \cap \operatorname{desc}(e_R)$, is independent in M.

Assume that $\langle Y_i \rangle_y \cap U_{e_L}$ is dependent in M_y. If u_L is not a leaf of T_M and e_L is not a basepoint edge, then we can apply Lemma 6.9 (i) to the two edges connecting u_L to its children. This then implies that $\langle Y_i \rangle_y \cap \operatorname{desc}(e_L)$ is dependent in M, contradicting the conclusion of the previous paragraph. Therefore u_L is a leaf or e_L is a basepoint edge. In either case, U_{e_L} is a singleton set, and this set must contain a loop, as $\langle Y_i \rangle_y \cap U_{e_L}$ is dependent. A basepoint cannot be a loop, so u_L is a leaf of T_M. Thus $U_{e_L} = \operatorname{desc}(e_L)$. Now the dependence of $\langle Y_i \rangle_y \cap U_{e_L}$ implies the dependence of $Y_i \cap \operatorname{desc}(e_L)$, a contradiction. The claim follows by a symmetrical argument for $\langle Y_i \rangle_y \cap U_{e_R}$. □

Claim 6.10.2 and the inductive assumption now mean that $q_L = q_j$ and $q_R = q_k$, for some values of j and k. Let X_j and X_k stand for $\mathrm{Rep}_{e_L}(q_j)$ and $\mathrm{Rep}_{e_R}(q_k)$. Then $(\langle Y_i\rangle_y \cap U_{e_L}) \sim_{U_{e_L}} X_j$ and $(\langle Y_i\rangle_y \cap U_{e_R}) \sim_{U_{e_R}} X_k$.

Claim 6.10.3. $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$ is independent in M_y, $Y_i \cap \mathrm{desc}(e)$ is independent in M, and $\langle Y_i\rangle_x \cap U_e$ is independent in M_x.

Proof. Assume that

$$\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R}) = (\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R})$$

is dependent in M_y. Then Proposition 2.3 implies that $X_j \cup X_k$ is dependent in M_y. The construction of f and A now means that $q = \mathsf{dep}$. Lemma 6.9 (i) implies that $Y_i \cap \mathrm{desc}(e)$ is dependent, so u fails to provide a counterexample. Therefore $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$ is independent in M_y.

Assume that $Y_i \cap \mathrm{desc}(e)$ is dependent in M. As $Y_i \cap \mathrm{desc}(e_L)$ and $Y_i \cap \mathrm{desc}(e_R)$ are independent by Claim 6.10.2, Lemma 6.9 (ii) implies that $\langle Y_i\rangle_y \cap (U_{e_L} \cup U_{e_R})$ is dependent in M_y, contradicting the previous paragraph.

Finally, assume that $\langle Y_i\rangle_x \cap U_e$ is dependent in M_x. Then $x \neq y$, or else U_e is the disjoint union of U_{e_L} and U_{e_R}, and we have a contradiction to the first paragraph. Hence e is a basepoint edge, meaning that U_e is a single element, and this element must be a loop. A basepoint cannot be a loop, so we have a contradiction. □

Assume that $x = y$, so that e, e_L, and e_R are all edges of T_x. In this case U_e is the disjoint union of U_{e_L} and U_{e_R}. Claim 6.10.3 says that $\langle Y_i\rangle_x \cap U_e = (\langle Y_i\rangle_x \cap U_{e_L}) \cup (\langle Y_i\rangle_x \cap U_{e_R})$ is independent in M_x, so Proposition 2.3 implies that $X_j \cup X_k$ is independent. Therefore $q = q_l$ for some q_l such that $(X_j \cup X_k) \approx_{U_e} \mathrm{Rep}_e(q_l)$. Hence $(X_j \cup X_k) \sim_{U_e} \mathrm{Rep}_e(q_l)$. We also know from Proposition 2.3 that $(\langle Y_i\rangle_x \cap U_e) \sim_{U_e} (X_j \cup X_k)$. Therefore $(\langle Y_i\rangle_x \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q_l)$ and Lemma 6.10 holds for u, a contradiction.

Now we must assume that $y \neq x$, so that e is a basepoint edge. This means that e is incident with a leaf, $\varphi_x(e_\alpha)$, in T_x, and e_α is the parent basepoint of T_y. Therefore $U_e = \{e_\alpha\}$, and e_α is in $\langle Y_i\rangle_x$ if and only if e_α is in $\mathrm{cl}_{M_y}(\langle Y_i\rangle_y) = \mathrm{cl}_{M_y}((\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R}))$. Note that $(\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R})$ is independent in M_y, by Claim 6.10.3.

Assume that e_α is in $\langle Y_i\rangle_x$, so that $\langle Y_i\rangle_x \cap U_e = U_e = \{e_\alpha\}$. In this case

$$(\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R}) \cup \{e_\alpha\}$$

is dependent in M_y. From Proposition 2.3, we have that $X_j \cup X_k$ is independent in M_y, and equivalent under $\sim_{(U_{e_L} \cup U_{e_R})}$ to $(\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R})$. Therefore $X_j \cup X_k \cup \{e_\alpha\}$ is also dependent in M_y. The construction of the function f now means that q is either q_1 or q_2. In the first case, $U_e \approx_{U_e} \emptyset$. Hence $(\langle Y_i\rangle_x \cap U_e) \sim_{U_e} \emptyset$, and as $\emptyset = \mathrm{Rep}_e(q_1)$, we see that u satisfies the lemma. Therefore $q = q_2$, and $\mathrm{Rep}_e(q_2) = U_e$. In this case $\langle Y_i\rangle_x \cap U_e$ and $\mathrm{Rep}_e(q_2)$ are equal, so $(\langle Y_i\rangle_x \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q_2)$ certainly holds, and we have a contradiction.

Now we must assume that e_α is not in $\langle Y_i\rangle_x$. Hence $\langle Y_i\rangle_x \cap U_e = \emptyset$. But in this case

$$(\langle Y_i\rangle_y \cap U_{e_L}) \cup (\langle Y_i\rangle_y \cap U_{e_R}) \cup \{e_\alpha\}$$

is independent in M_y. Using the arguments from the previous paragraph, we show that $X_j \cup X_k \cup \{e_\alpha\}$ is also independent in M_y, so $q = q_1$, where $\mathrm{Rep}_e(q_1) = \emptyset$. Now $\emptyset = \langle Y_i\rangle_x \cap U_e$, so $(\langle Y_i\rangle_x \cap U_e) \sim_{U_e} \mathrm{Rep}_e(q_1)$ holds, and this contradiction completes the proof of Lemma 6.10. □

Now we can prove that (T_M, σ_M) is a parse tree for A. Let the left child of the root t be u_L, and let the right child be u_R. We let e_L and e_R be the edges joining t to these children. Recall $U_{e_L} = \{e_{m,L}\}$ and $U_{e_R} = \{e_{m,R}\}$, and M_t is a copy of $U_{1,2}$ on the ground set $\{e_{m,L}, e_{m,R}\}$. Let Y_i be a subset of $E(M)$, and let q be the state applied to t by the run of A on $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$. We let q_L and q_R be the states applied to u_L and u_R.

In the first case, we assume that $q = \mathsf{dep}$, and we aim to show that Y_i is dependent. Our construction of the function labelling t means (q_L, q_R) either contains the symbol dep, or is (q_2, q_2). If $q_L = \mathsf{dep}$, then $Y_i \cap \mathrm{desc}(e_{m,L})$ is dependent in M by Lemma 6.10, and hence Y_i is dependent in M. By symmetry, we assume that neither q_L nor q_R is dep, so $(q_L, q_R) = (q_2, q_2)$. From this we see that $\mathrm{Rep}_{e_L}(q_2) = U_{e_L} = \{e_{m,L}\}$, and moreover $(\langle Y_i\rangle_t \cap U_{e_L}) \sim_{U_{e_L}} U_{e_L}$. Because $\mathrm{Rep}_{e_L}(q_2)$ is defined, $U_{e_L} \not\sim_{U_{e_L}} \emptyset$, so $\langle Y_i\rangle_t \cap U_{e_L}$ is not empty. Therefore $\langle Y_i\rangle_t$ contains $e_{m,L}$. By symmetry, $\langle Y_i\rangle_t$ contains $e_{m,R}$. Now $\langle Y_i\rangle_t \cap (U_{e_L} \cup U_{e_R}) = \{e_{m,L}, e_{m,R}\}$ is dependent in M_t. Lemma 6.9 (i) implies that $Y_i \cap (\mathrm{desc}(e_L) \cup \mathrm{desc}(e_R)) = Y_i$ is dependent in M, exactly as we wanted.

In the second case, we assume that Y_i is dependent in M. If $Y_i \cap \mathrm{desc}(e_L)$ is dependent, then $q_L = \mathsf{dep}$ by Lemma 6.10. In this case, $q = \mathsf{dep}$, which is what we want. Therefore we assume by symmetry that $Y_i \cap \mathrm{desc}(e_L)$ and $Y_i \cap \mathrm{desc}(e_R)$ are independent in M. As Y_i is dependent, Lemma 6.9 (ii) implies that $\langle Y_i\rangle_t \cap (U_{e_L} \cup U_{e_R}) = \langle Y_i\rangle_t \cap \{e_{m,L}, e_{m,R}\}$ is dependent in M_t. Therefore $\langle Y_i\rangle_t = \{e_{m,L}, e_{m,R}\}$. Let x_L be the node of T joined to t by $e_{m,L}$, and define x_R similarly. Then $e_{m,L} \in \langle Y_i\rangle_t$ implies that $e_{m,L}$ is in $\mathrm{cl}_{M_{x_L}}(\langle Y_i\rangle_{x_L})$.

Since every node of T other than t corresponds to a matroid with at least three elements, it follows that neither u_L nor u_R is a leaf in T_M. Let e_{LL} and e_{LR} be the edges that join u_L to its children: u_{LL} and u_{LR}. Then $Y_i \cap \mathrm{desc}(e_{LL})$ and $Y_i \cap \mathrm{desc}(e_{LR})$ are independent in M, as they are subsets of $Y_i \cap \mathrm{desc}(e_L)$. Therefore A applies states q_j and q_k to u_{LL} and u_{LR}. Let X_j and X_k be $\mathrm{Rep}_{e_{LL}}(q_j)$ and $\mathrm{Rep}_{e_{LR}}(q_j)$ respectively. Then $(\langle Y_i\rangle_{x_L} \cap U_{e_{LL}}) \sim_{U_{e_{LL}}} X_j$ and $(\langle Y_i\rangle_{x_L} \cap U_{e_{LR}}) \sim_{U_{e_{LR}}} X_k$ by Lemma 6.10. Proposition 2.3 implies that $\langle Y_i\rangle_{x_L} \cap (U_{e_{LL}} \cup U_{e_{LR}}) = \langle Y_i\rangle_{x_L}$ is equivalent to $X_j \cup X_k$ under $\sim_{(U_{e_{LL}} \cup U_{e_{LR}})}$. From $e_{m,L} \in \mathrm{cl}_{M_{x_L}}(\langle Y_i\rangle_{x_L})$, we see that $\langle Y_i\rangle_{x_L} \cup \{e_{m,L}\}$ is dependent in M_{x_L}. Therefore $X_j \cup X_k \cup \{e_{m,L}\}$ is also dependent. Because $U_{e_L} = \{e_{m,L}\}$ is certainly not equivalent to $\emptyset$ under

$\sim_{U_{e_L}}$, we see that A applies the state q_2 to u_L. By symmetry it applies q_2 to u_R. Thus $q = f(q_2, q_2) = \mathsf{dep}$, where f is the function applied to t by σ_M.

We have shown that A accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ if and only if Y_i is independent in M, exactly as we wanted.

Reducing to the connected case. Our final task is to show that we can construct a parse tree for M when M is not connected. In the first part of the proof, we have established that there is a fixed-parameter tractable algorithm for constructing a parse tree relative to the automaton A, when M is connected.

We augment A to obtain the automaton A'. We add a new character, κ, to the alphabet of A, and we add new states, dep' and indep', to its state space. We augment the transition rules so that $\delta_2(\kappa, \alpha, \beta)$ is $\{\mathsf{indep}'\}$ when both α and β are indep' or accepting states of A, and set $\delta_2(\kappa, \alpha, \beta)$ to be $\{\mathsf{dep}'\}$ otherwise. Any output of δ_0 or δ_2 that has now not been defined is set to be $\{\mathsf{dep}'\}$. The accepting states of A' are the accepting states of A, along with indep'. We can identify the connected components, $M_1, \ldots, M_n$, of M in polynomial time [1]. We assume that $n > 1$. Each M_j is in $\mathcal{M}$, as $\mathcal{M}$ is minor-closed, and we can construct a description $\Delta(M_j)$ in polynomial time, as Δ is minor-compatible. Moreover, $\mathrm{bw}(M_j) \leq \lambda$ [20, Proposition 14.2.3], Therefore we have a fixed-parameter tractable algorithm for constructing the parse trees, (T_{M_j}, σ_{M_j}). Now we construct a rooted tree with n leaves, where each non-leaf has a left child and a right child, and we apply the label κ to each non-leaf vertex. We then identify the n leaves with the n roots in $T_{M_1}, \ldots, T_{M_n}$. Now it is straightforward to verify that A' will use A to check independence in each connected component of M, and accept if and only if A accepts in each of those components. Therefore A' decides $\mathrm{Ind}(X_i)$ for any matroid in $\mathcal{M}$. Thus we have constructed a parse tree for M. This completes the proof of Theorem 6.7. □

Proposition 6.1 and Theorem 6.7 immediately lead to the following result.

Theorem 6.11. *Let $\mathcal{M}$ be a minor-closed class of matroids with a minor-compatible representation, Δ. Assume that $\{M \in \mathcal{M} \colon M$ is 3-connected$\}$ is computably pigeonhole. Let ψ be any sentence in MS_0. There is a fixed-parameter tractable algorithm which will test whether ψ holds for matroids in $\mathcal{M}$, where the parameter is branch-width.*

7. Decidability and axiomatisability

The theorems of Courcelle and Hliněný have as their goal efficient model-checking: given a sentence and a graph/matroid, we test whether the sentence is satisfied by that object. Decidability is orthogonal to this problem: given a class of objects and a sentence, we want to decide (in finite time, but not necessarily efficiently) if that sentence is a theorem for the class.

Definition 7.1. Let $\mathcal{M}$ be a class of set-systems. The *MS_0 theory* of $\mathcal{M}$ is the collection of MS_0 sentences that are satisfied by all set-systems in $\mathcal{M}$.

We say that the MS_0 theory of $\mathcal{M}$ is *decidable* if there is a Turing Machine which takes as input any sentence in MS_0, and after a finite amount of time decides whether or not the sentence is in the theory of $\mathcal{M}$.

The key idea in the forthcoming decidability proofs is that, given a tree automaton, there is a finite procedure which will decide if there is a tree that the automaton will accept. (See, for example, [9, Theorem 3.74].)

Lemma 7.2. *Let $A = (\Sigma, Q, F, \delta_0, \delta_2)$ be a tree automaton. Let Z be a subset of Q, and let q be a state in $Q - Z$. There is a finite procedure for deciding the following question: does there exist a Σ-tree, (T, σ), with root t such that if r is the run of A on (T, σ), then $q \in r(t)$, and $r(v) \cap Z = \emptyset$ for every vertex v of T.*

Proof. Note that if $r(v)$ contains q, where v is a non-root vertex, then we may as well consider the subtree of (T, σ) that has v as its root. This means that we lose no generality in searching only for Σ-trees such that q is contained in $r(t)$, but not in $r(v)$ when v is a non-root vertex. Our search will construct the desired tree, T, or establish that it does not exist.

Now we proceed by induction on $|Q - Z|$. Assume $Q - Z$ contains only q. If $\delta_0(\alpha) = \{q\}$ for some $\alpha \in \Sigma$, then we return YES: we simply consider the Σ-tree consisting of a single leaf labelled with α. If no such α exists, then we return NO. This completes the proof of the base case, so now we make the obvious inductive assumption.

If $\delta_0(\alpha) \cap Z = \emptyset$ and $q \in \delta_0(\alpha)$ for some $\alpha \in \Sigma$, then we can construct the Σ-tree with a single leaf labelled with α, and the answer is YES. Therefore we assume that no such α exists.

We search for tuples $(\alpha, q_L, q_R) \in \Sigma \times Q \times Q$ such that $q \in \delta_2(\alpha, q_L, q_R)$ and $\delta_2(\alpha, q_L, q_R) \cap Z = \emptyset$. If no such tuple exists, then we halt and return NO. Otherwise, for each such tuple, we search for Σ-trees, (T_L, σ_L) and (T_R, σ_R), with the following properties: If r_L and r_R are the runs on these trees, then $r_L(v) \cap (Z \cup \{q\}) = \emptyset$ for each vertex v of T_L, and similarly $r_R(v) \cap (Z \cup \{q\}) = \emptyset$. Furthermore, q_L is in $r_L(t_L)$, and q_R is in $r_R(t_R)$, where t_L and t_R are the roots of T_L and T_R. By induction, we can construct such trees, if they exist. If they do exist, then we construct T from the disjoint union of T_L and T_R by adding a root t, and making its children t_L and t_R. We then apply the label α to t. This justifies returning the answer YES. If we find that no such trees exist for each tuple (α, q_L, q_R), then we return NO. □

Corollary 7.3. *Let $A = (\Sigma, Q, F, \delta_0, \delta_2)$ be a tree automaton. There is a finite procedure to decide whether there exists a Σ-tree that A accepts.*

Proof. We repeatedly apply Lemma 7.2 with Z set to be the empty set, and q set to be a state in F. □

When we say that a class of set-systems is *axiomatisable* we mean there is an MS_0 sentence, τ, such that a set-system satisfies τ if and only if it is

in the class. The matroid independence axioms can be stated in MS_0. If N is a fixed matroid, there is an MS_0 sentence that characterises the matroids having a minor isomorphic to N [13, Lemma 5.1]. From this it follows that a minor-closed of matroids is axiomatisable if it has a finite number of excluded minors. There are also axiomatisable minor-closed classes that have infinitely many excluded minors (Remark 7.8). The class of $\mathbb{K}$-representable matroids is not axiomatisable when $\mathbb{K}$ is an infinite field [18].

Theorem 7.4. *Let $\mathcal{M}$ be a axiomatisable class of set-systems with bounded decomposition-width. The MS_0 theory of $\mathcal{M}$ is decidable.*

Proof. Let ψ be an arbitrary sentence in MS_0. We wish to decide whether all set-systems in $\mathcal{M}$ satisfy ψ. This is equivalent to deciding whether there exists a set-system in $\mathcal{M}$ satisfying $\neg\psi$. Let τ be an MS_0 sentence such that a set-system belongs to $\mathcal{M}$ if and only if it satisfies τ.

Lemma 5.2 implies that there is a 1-ary automaton A' such that for every $M = (E, \mathcal{I})$ in $\mathcal{M}$, there is a Σ-tree (T_M, σ_M) and a bijection $\varphi_M \colon E \to L(T_M)$ where A' accepts $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \{Y_i\})$ if and only if Y_i is in $\mathcal{I}$, for any $Y_i \subseteq E(M)$. We use (the proof of) Lemma 4.7 to construct an automaton, A, such that A accepts $\mathrm{enc}(T, \sigma, \varphi, \emptyset)$ if and only if $M(A', T, \sigma, \varphi)$ satisfies $\tau \wedge \neg\psi$, for each Σ-tree (T, σ) and each bijection φ from a finite set to the leaves of T.

According to Corollary 7.3, we can decide in finite time whether or not there is a Σ-tree that is accepted by A. If (T, σ) is such a tree, then let φ be the identity function on $L(T)$. The set-system $M(A', T, \sigma, \varphi)$ satisfies $\tau \wedge \neg\psi$, so it is a set-system in $\mathcal{M}$ (as it satisfies τ) that does not satisfy ψ. Therefore ψ is not in the theory of $\mathcal{M}$. On the other hand, if $M \in \mathcal{M}$ does not satisfy ψ, then $M = M(A', T_M, \sigma_M, \varphi_M)$ satisfies $\tau \wedge \neg\psi$, so A will accept at least one tree, namely $\mathrm{enc}(T_M, \sigma_M, \varphi_M, \emptyset)$. □

Corollary 7.5. *Let $\mathcal{M}$ be an axiomatisable pigeonhole class of matroids. Let λ be a positive integer. The MS_0 theory of $\{M \in \mathcal{M} \colon \mathrm{bw}(M) \leq \lambda\}$ is decidable.*

Proof. The family of matroids with branch-width at most λ is minor-closed, and it has finitely many excluded minors [12]. Therefore we can let τ be an MS_0 sentence encoding the independence axioms for matroids, membership of $\mathcal{M}$, and branch-width of at most λ. Thus a set-system satisfies τ if and only if it is in $\{M \in \mathcal{M} \colon \mathrm{bw}(M) \leq \lambda\}$. This class has bounded decomposition-width, so we apply Theorem 7.4. □

The resolution of Rota's conjecture [11] means that the class of $\mathbb{F}$-representable matroids is axiomatisable when $\mathbb{F}$ is a finite field. Corollary 7.5 now implies that the MS_0 theory of $\mathbb{F}$-representable matroids with branch-width at most λ is decidable. This was previously proved by Hliněný and Seese [15, Corollary 5.3], who did not rely on Rota's conjecture. Let $\mathcal{M}$ be any minor-closed class of $\mathbb{F}$-representable matroids, where $\mathbb{F}$ is a finite field. Geelen, Gerards, and Whittle have also announced that $\mathcal{M}$ has finitely many

excluded minors [11, Theorem 6]. Therefore $\mathcal{M}$ is axiomatisable, and hence the MS_0 theory of $\{M \in \mathcal{M}\colon \mathrm{bw}(M) \leq \lambda\}$ is decidable, for any positive integer λ.

A basis, B, of the matroid M is *fundamental* if $B \cap Z$ spans Z whenever Z is a cyclic flat. A matroid with a fundamental basis is a *fundamental transversal matroid* (see [3]). It is an exercise to prove that B is fundamental if and only if x is freely placed in the flat spanned by the fundamental circuit, $C(x, B)$, for every $x \notin B$. This is equivalent to saying that if Z is a cyclic flat containing x, then Z contains $C(x, B)$. This property can clearly be expressed in MS_0. In [10, Theorem 6.3], we prove that the class of fundamental transversal matroids is pigeonhole. Therefore the MS_0 theory of fundamental transversal matroids with branch-width at most λ is decidable by Corollary 7.5.

The class of bicircular matroids has finitely many excluded minors [deVos and Goddyn, personal communication], which implies that it can be characterised by an MS_0 sentence. The class is also pigeonhole, as we prove in [10, Theorem 8.4], so the MS_0 theory of bicircular matroids with branch-width at most λ is decidable.

7.1. **Undecidable theories.** We also have some results that allow us to prove results in the negative direction, by showing that certain classes have undecidable theories.

Assume that F is a flat of the matroid M, and let M' be a single-element extension of M. Let e be the element in $E(M') - E(M)$. We say that M' is a *principal extension* of M by F if $F \cup e$ is a flat of M' and whenever $X \subseteq E(M)$ spans e in M', it spans $F \cup e$.

Let G be a simple graph with vertex set $\{v_1, \ldots, v_n\}$ and edge set $\{e_1, \ldots, e_m\}$. Let $m(G)$ be the rank-3 sparse paving matroid with ground set $\{v_1, \ldots, v_n\} \cup \{e_1, \ldots, e_m\}$. The only non-spanning circuits of $m(G)$ are the sets $\{v_i, e_k, v_j\}$, where e_k is an edge of G joining the vertices v_i and v_j.

Theorem 7.6. *Let $\mathcal{M}$ be a class of matroids that contains all rank-3 uniform matroids, and is closed under principal extensions. The MS_0 theory of $\mathcal{M}$ is undecidable.*

Proof. We let G be a simple graph, and we let $m^+(G)$ be the matroid obtained from $m(G)$ by placing a new element parallel to each 'vertex' element v_i. It is easy to check that $m^+(G)$ is contained in $\mathcal{M}$ for every simple graph G. Moreover, we can characterise the matroids that are equal to $m^+(G)$ for some graph G in the following way. Let M be a matroid. Then M is equal to $m^+(G)$ for some graph G (with at least three vertices) if and only if the following properties hold:

(i) $r(M) = 3$,
(ii) M is loopless, and any rank-1 flat has cardinality one or two,

(iii) any rank-2 flat that contains at least three rank-1 flats contains exactly three such flats, one of cardinality one, and two of cardinality two,
(iv) if x is an element that is not in a parallel pair, then x is in exactly one rank-2 flat that contains three rank-1 flats.

From this it is clear that there is an MS_0 sentence, τ, such that a matroid satisfies τ if and only if it is isomorphic to $m^+(G)$ for some simple graph G.

Now we consider the logical language, MS_1, for graphs. In this language, we can quantify over variables that represent vertices, and variables that represent subsets of vertices. We have a binary predicate that expresses when a vertex is in a set of vertices, and another that expresses when two vertices are adjacent. Let ψ be a sentence in MS_1. There is a corresponding sentence, ψ', in MS_0 such that a simple graph, G, satisfies ψ if and only if $m^+(G)$ satisfies ψ'. Let $\mathrm{Vert}(X)$ stand for a MS_0 formula expressing the fact that X is a 2-element circuit. To construct ψ', we make the following interpretations:

(i) we replace Qv with $QX_v\,\mathrm{Vert}(X_v)$, where Q is a quantifier, and v is a vertex variable,
(ii) we replace QV with

$$QX\forall X_1(\mathrm{Sing}(X_1)\wedge X_1\subseteq X)\rightarrow \exists X_2(X_1\subseteq X_2\wedge X_2\subseteq X\wedge \mathrm{Vert}(X_2)),$$

where Q is a quantifier, and V is a set variable,
(iii) we replace $v\sim v'$ (the adjacency predicate) with a MS_0 formula saying that $X_v\cup X_{v'}$ is not a flat.

Seese has shown that the MS_1 theory of simple graphs is undecidable ([22, Theorem 4], see also [15, Theorem 4.8]). Imagine that the MS_0 theory of $\mathcal{M}$ is decidable. Then for any MS_1 sentence, ψ, we could decide if $\tau\rightarrow\psi'$ is a theorem for $\mathcal{M}$. This is equivalent to deciding whether ψ is a theorem for all simple graphs, so we have contradicted Seese's result. □

The argument in [10, Proposition 6.1] shows that the class of (strict) gammoids is closed under principal extensions. The next result follows easily.

Corollary 7.7. *Let $\mathbb{K}$ be an infinite field. The MS_0 theory of rank-3 $\mathbb{K}$-representable matroids is undecidable. The MS_0 theories of rank-3 cotransversal matroids and gammoids are undecidable. Therefore the MS_0 theory of corank-3 transversal matroids is undecidable.*

Corollary 7.7 is complementary to a result by Hliněný and Seese [15, Theorem 7.2], who have shown that the MS_0 theory of spikes is undecidable. Although every spike has branch-width three, spikes have unbounded rank, and are not representable over a common field.

7.2. **Axiomatisability and pigeonhole classes.** We will now show that axiomatisability and the pigeonhole property are independent of each other by exhibiting classes that have exactly one of these properties. One case is easy: the class of sparse paving matroids can be axiomatised by insisting that every non-spanning circuit is a hyperplane. But [10, Lemma 4.1] implies this class is not pigeonhole.

For the other case, we consider *polygon* matroids. Let C_n be the rank-3 sparse paving matroid with ground set $\{e_1, \ldots, e_{2n}\}$ and non-spanning circuits

$$\{e_1, e_2, e_3\}, \{e_3, e_4, e_5\}, \ldots, \{e_{2n-3}, e_{2n-2}, e_{2n-1}\}, \{e_{2n-1}, e_{2n}, e_1\}.$$

The polygon matroids form a well-known infinite antichain [20, Example 14.1.2]. Next we consider *path* matroids. They too are rank-3 sparse paving matroids. The ground set of a path matroid can be ordered $e_1, \ldots, e_n$ in such a way that any non-spanning circuit is a set of three consecutive elements in the ordering. Note that the sparse paving property implies that if $\{e_i, e_{i+1}, e_{i+2}\}$ is a non-spanning circuit, then $\{e_{i+1}, e_{i+2}, e_{i+3}\}$ is not. Every rank-3 proper minor of a polygon matroid is a path matroid, and every path matroid is a minor of a polygon matroid.

Remark 7.8. Path matroids can be characterised as follows: they are rank-three matroids where every non-spanning circuit contains exactly three elements, and no element is in more than two non-spanning circuits. Moreover, any non-spanning circuit contains an element that is in exactly one non-spanning circuit. Finally, if $\{u, v, w\}$ is a non-spanning circuit, and w is in exactly one non-spanning circuit, then there is a partition, (U, V), of $E(M) - w$ with $u \in U$ and $v \in V$, where $\{u, v, w\}$ is the only non-spanning circuit containing elements of both U and V. This characterisation shows that the class of path matroids is axiomatisable. Furthermore, matroids with rank at most two can be characterised by saying that any subset containing three pairwise disjoint singleton sets is dependent. Therefore the minor-closed class consisting of path matroids and all matroids with rank at most two is axiomatisable. However, it has an infinite number of excluded minors, since every polygon matroid is an excluded minor.

Proposition 7.9. *Let $\mathcal{M}$ be the class containing all polygon and path matroids, and all matroids of rank at most two. Then $\mathcal{M}$ has bounded decomposition-width.*

Proof. It is easy to see that if M is a matroid with rank zero or one, then $\mathrm{dw}(M) \leq 3$. Now assume that M is a rank-2 matroid. Let (T, φ) be a decomposition of M such that if (U, V) is a displayed partition, then no more than one parallel class of M intersects both U and V. It is clear that such a decomposition exists. Now we can easily verify that $\mathrm{dw}(M) \leq 5$.

Let M be a rank-3 matroid in $\mathcal{M}$. It is easy to see that there is an ordering $e_1, \ldots, e_n$ of $E(M)$ such that for any partition $(U, V) =$

$(\{e_1, \ldots, e_t\}, \{e_{t+1}, \ldots, e_n\})$, at most two non-spanning circuits contain elements from both U and V. We let (T, φ) be a decomposition that displays only partitions of this type. It is straightforward to verify that this type of decomposition leads to an upper bound on the decomposition-width of all rank-3 matroids in $\mathcal{M}$. □

Now we can prove the existence of a minor-closed class that is pigeonhole without being axiomatisable.

Lemma 7.10. *There is a minor-closed class, $\mathcal{M}$, of matroids with the following properties. Each matroid in $\mathcal{M}$ has rank (and therefore branch-width) at most three. Furthermore $\mathcal{M}$ has bounded decomposition-width, and is therefore pigeonhole. However, $\mathcal{M}$ has an undecidable MS_0 theory, so $\mathcal{M}$ is not axiomatisable.*

Proof. We consider the classes consisting of all matroids with rank at most two, all path matroids, and some subset of the polygon matroids. The fact that these classes have bounded decomposition-width follows from Proposition 7.9. The number of such classes is the cardinality of the power set of the natural numbers, so there are uncountably many such classes. Assume that every such class has a decidable theory. There are countably many Turing Machines, so we let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two distinct such classes, such that exactly the same Turing Machine decides the theories of $\mathcal{M}_1$ and $\mathcal{M}_2$. Therefore $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfy exactly the same MS_0 sentences. Without loss of generality, we can let P be a polygon matroid in $\mathcal{M}_1$ but not in $\mathcal{M}_2$. It is easy to express the statement 'this matroid is not isomorphic to P' in MS_0. This sentence is a theorem for $\mathcal{M}_2$, but not for $\mathcal{M}_1$, so we have a contradiction. □

7.3. **Open problems.** The class of lattice path matroids has infinitely many excluded minors [2]. Despite this, we make the following conjecture.

Conjecture 7.11. *The class of lattice path matroids can be characterised by a sentence in MS_0.*

We have proved that the class of lattice path matroids is pigeonhole [10, Theorem 7.2], so Conjecture 7.11, along with Corollary 7.5, would imply the decidability of the MS_0 theory of lattice path matroids with bounded branch-width.

The following conjecture would imply that any minor-closed class of H-gain-graphic matroids can be characterised with a sentence in MS_0, when H is a finite group.

Conjecture 7.12. *Let H be a finite group, and let $\mathcal{M}$ be a minor-closed class of H-gain-graphic matroids. Then $\mathcal{M}$ has only finitely many excluded minors.*

We also conjecture that the class of H-gain-graphic matroids is (computably) pigeonhole ([10, Conjecture 9.3]). This conjecture, combined with

Conjecture 7.12, would imply decidability for any minor-closed of H-gain-graphic matroids with bounded branch-width.

DeVos, Funk, and Pivotto have proved that if H is an infinite group, then the class of H-gain-graphic matroids has infinitely many excluded minors [6, Corollary 1.3]. We conjecture a stronger property.

Conjecture 7.13. *Let H be an infinite group. The class of H-gain-graphic matroids cannot be characterised with a sentence in MS_0.*

It is not too difficult to see that the techniques of [18] settle this conjecture when H contains elements of arbitrarily high order. Thus it is open only in the case that H is an infinite group with finite exponent. An easy example of such a group is the infinite direct product $(\mathbb{Z}/2\mathbb{Z})^{\mathbb{Z}^+}$, but there exist more sophisticated examples, such as Tarski monster groups.

We also believe the following.

Conjecture 7.14. *Let H be an infinite group. The class of rank-3 H-gain-graphic matroids has an undecidable MS_0 theory.*

8. Acknowledgements

We thank Geoff Whittle for several important conversations. Funk and Mayhew were supported by a Rutherford Discovery Fellowship, managed by Royal Society Te Apārangi.

References

[1] R. E. Bixby and W. H. Cunningham. Matroid optimization and algorithms. In *Handbook of combinatorics, Vol. 1, 2*, pp. 551–609. Elsevier, Amsterdam (1995).

[2] J. E. Bonin. Lattice path matroids: the excluded minors. *J. Combin. Theory Ser. B* **100** (2010), no. 6, 585–599.

[3] J. E. Bonin, J. P. S. Kung, and A. de Mier. Characterizations of transversal and fundamental transversal matroids. *Electron. J. Combin.* **18** (2011), no. 1, Paper 106, 16.

[4] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. and Comput.* **85** (1990), no. 1, 12–75.

[5] W. H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM J. Comput.* **15** (1986), no. 4, 948–957.

[6] M. DeVos, D. Funk, and I. Pivotto. When does a biased graph come from a group labelling? *Adv. in Appl. Math.* **61** (2014), 1–18.

[7] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer-Verlag, New York (1999).

[8] R. G. Downey and M. R. Fellows. *Fundamentals of parameterized complexity.* Texts in Computer Science. Springer, London (2013).

[9] J. Engelfriet. Tree automata and tree grammars. *arXiv e-prints* (2015), arXiv:1510.02036.

[10] D. Funk, D. Mayhew, and M. Newman. Tree automata and pigeonhole classes of matroids – II. *arXiv e-prints* (2019), arXiv:1910.04361.

[11] J. Geelen, B. Gerards, and G. Whittle. Solving Rota's conjecture. *Notices Amer. Math. Soc.* **61** (2014), no. 7, 736–743.

[12] J. F. Geelen, A. M. H. Gerards, N. Robertson, and G. P. Whittle. On the excluded minors for the matroids of branch-width k. *J. Combin. Theory Ser. B* **88** (2003), no. 2, 261–265.
[13] P. Hliněný. On matroid properties definable in the MSO logic. In *Mathematical foundations of computer science 2003*, volume 2747 of *Lecture Notes in Comput. Sci.*, pp. 470–479. Springer, Berlin (2003).
[14] P. Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Combin. Theory Ser. B* **96** (2006), no. 3, 325–351.
[15] P. Hliněný and D. Seese. Trees, grids, and MSO decidability: from graphs to matroids. *Theoret. Comput. Sci.* **351** (2006), no. 3, 372–393.
[16] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, Annals of mathematics studies, no. 34, pp. 3–41. Princeton University Press, Princeton, N. J. (1956).
[17] D. Král. Decomposition width of matroids. *Discrete Appl. Math.* **160** (2012), no. 6, 913–923.
[18] D. Mayhew, M. Newman, and G. Whittle. Yes, the "missing axiom" of matroid theory is lost forever. *Trans. Amer. Math. Soc.* **370** (2018), no. 8, 5907–5929.
[19] S.-i. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B* **96** (2006), no. 4, 514–528.
[20] J. Oxley. *Matroid theory*. Oxford University Press, New York, second edition (2011).
[21] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.* **3** (1959), 114–125.
[22] D. Seese. The structure of the models of decidable monadic theories of graphs. *Ann. Pure Appl. Logic* **53** (1991), no. 2, 169–195.
[23] Y. Strozecki. *Enumeration complexity and matroid decomposition*. Ph.D. thesis, Université Paris Diderot (2010).
[24] Y. Strozecki. Monadic second-order model-checking on decomposable matroids. *Discrete Appl. Math.* **159** (2011), no. 10, 1022–1039.

Faculty of Science and Technology, Douglas College, Vancouver, Canada
Email address: funkd@douglascollege.ca

School of Mathematics and Statistics, Victoria University of Wellington, New Zealand
Email address: dillon.mayhew@vuw.ac.nz

Department of Mathematics and Statistics, University of Ottawa, Ottawa, Canada
Email address: mnewman@uottawa.ca