Myhill–Nerode Methods for Hypergraphs

**René van Bevern · Rodney G. Downey · Michael R. Fellows ·
Serge Gaspers · Frances A. Rosamond**

Abstract We give an analog of the Myhill–Nerode theorem from formal language
theory for hypergraphs and use it to derive the following results for two NP-hard
hypergraph problems. (1) We provide an algorithm for testing whether a hypergraph
has cutwidth at most k that runs in linear time for constant k. In terms of parameterized
complexity theory, the problem is *fixed-parameter linear* parameterized by k. (2) We
show that it is not expressible in monadic second-order logic whether a hypergraph
has bounded (fractional, generalized) hypertree width. The proof leads us to conjec-
ture that, in terms of parameterized complexity theory, these problems are W[1]-hard
parameterized by the *incidence treewidth* (the treewidth of the incidence graph). Thus,

R. van Bevern (✉)
Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
e-mail: rene.vanbevern@tu-berlin.de

R. G. Downey
Victoria University of Wellington, Wellington, New Zealand
e-mail: rod.downey@vuw.ac.nz

M. R. Fellows · F. A. Rosamond
School of Engineering and IT, Charles Darwin University, Darwin, Australia
e-mail: michael.fellows@cdu.edu.au

F. A. Rosamond
e-mail: frances.rosamond@cdu.edu.au

S. Gaspers
The University of New South Wales and NICTA, Sydney, Australia
e-mail: sergeg@cse.unsw.edu.au

in the form of the Myhill–Nerode theorem for hypergraphs, we obtain a method to derive linear-time algorithms and to obtain indicators for intractability for hypergraph problems parameterized by incidence treewidth.

1 Introduction

There are two prevalent algorithmic techniques for solving NP-hard problems in linear time on graphs of bounded treewidth—a measure for the "tree-likeness" of a graph:

Technique 1. Compute a *tree decomposition*—a tree-like representation—of the input graph in linear time [6] and use dynamic programming from the leaves to the root of the tree decomposition.

Technique 2. Express the graph property to be decided in monadic second-order logic of graphs; the expression can be turned into a linear-time algorithm deciding the graph property [14, Theorem 6.4(1)].

For a primer on these algorithmic techniques, we refer to Niedermeier [41, Chapter 10].

In some cases, graph problems do not easily give in to these standard techniques. A third technique helps finding linear-time algorithms on graphs of bounded treewidth or to prove the inapplicability of the above standard techniques [2,8,26]: similarly to how regular languages can be recognized by finite automata, some graph problems on graphs of bounded treewidth can be solved in linear time by tree automata [17, Section 12.7]. In fact, many of the dynamic programming algorithms on tree decompositions used in Technique 1 are based on a standard approach that mimics tree automata [8]. Moreover, Technique 2 is based on the fact that an expression in monadic second-order logic can be turned into a tree automaton [14, Chapter 6]. Disproving the existence of a tree automaton for a problem therefore shows that it is presumably not straightforward to solve the problem on graphs of bounded treewidth using Technique 1 and even impossible using Technique 2.

A sufficient and necessary condition for the existence of a tree automaton deciding some graph problem can be given by an adaption of the Myhill–Nerode theorem from formal language theory to graphs [17, Section 12.7], which helped gain insight into the following graph problems:

CUTWIDTH. Testing a graph for bounded cutwidth can be done in linear time [2]. Thilikos et al. [45] later gave a dynamic programming algorithm that is significantly more technical, but has the advantage of constructing a solution instead of only answering whether a solution exists.

BANDWIDTH. The graph property of having bounded bandwidth is not recognizable by a tree automaton [2]. Note that this unconditional result significantly strengthens the previously known NP-hardness of the problem on trees [25] (which of course also excludes finite-state solvability of the problem for trees, but only under the assumption P $\neq$ NP).

TRIANGULATING COLORED GRAPHS. A tree automaton cannot decide whether a colored graph can be triangulated in such a way that adjacent vertices have distinct

colors [8]. The problem is known as PERFECT PHYLOGENY in the context of molecular biology and later turned out to be W[1]-hard [9,10] parameterized by the treewidth t, that is, not solvable in $O(f(t)n^c)$ time for any constant c independent of t under the widely accepted parameterized complexity assumption FPT $\neq$ W[1].

Our work extends the graph-theoretic analog of the Myhill–Nerode characterization of regular languages to hypergraphs. In this way, we provide a method to derive linear-time algorithms (or to obtain an indication for intractability) for hypergraph problems on hypergraphs with bounded *incidence treewidth* (treewidth of the incidence graph). Thus, our work is tightly connected to the existence of *fixed-parameter algorithms*— a rising technique that allows for solving NP-hard problems exactly *and* efficiently when certain parameters of the input data are small [17,22,41]. From this point of view, incidence treewidth is an interesting hypergraph parameter, since the incidence treewidth of a hypergraph is not greater than the treewidth of its primal or dual graph (two commonly used treewidth generalizations for hypergraphs) but can be arbitrarily smaller [31,44].

Applying Myhill–Nerode methods to hypergraphs, we obtain results for the problems HYPERGRAPH CUTWIDTH and (GENERALIZED, FRACTIONAL) HYPERTREE WIDTH, which will be formally defined in Sects. 4 and 5, respectively.

1.1 Related Work

Generalizations of the Myhill–Nerode theorem. The Myhill–Nerode theorem as sufficient and necessary condition for a formal language being regular is due to Myhill [38] and Nerode [40]. Since then, analogs of the Myhill–Nerode theorem were provided for graphs of bounded treewidth [2], matroids of bounded branchwidth [30], graphs of bounded rankwidth [24], and edge- and vertex-colored graphs of bounded treewidth and cliquewidth [14, Sections 4.2.2 and 4.4.2].

Finite-state approaches to solving graph problems for graphs of bounded pathwidth and treewidth. The earliest, and seminal work, applying ideas from finite automata theory to graph problems was the 1985 paper of Bern et al. [3] based on k-terminal recursively defined families of graphs. These ideas were quickly taken up and extended in many directions by many different groups of researchers with overlapping as well as independent results obtained in a flurry of activity. This includes the influential 1985 work of Wimer et al. [48] (see also Wimer's 1987 Ph.D. Thesis [47]), and Mahajan and Peters [35]. Finite-state and Myhill–Nerode-related methods were explored in regards of computing minor order obstruction sets by Fellows and Langston [18] and by Lagergren and Arnborg [33]. The regularity (in the sense of Bern, Lawler, and Wong [3]—finite-state dynamic programming multiplication tables) of bounded treewidth and pathwidth was shown by Bodlaender and Kloks [7] and, independently, by Lagergren and Arnborg [33].

In many cases, this early work circulated in some form (e.g., Technical Reports) years in advance of its eventual publication, making the historical record murky—but it was an exciting time for bounded treewidth and pathwidth algorithmics. The period

is ably surveyed by Borie, Parker, and Tovey [12], where many more references to early work in the area can be found.

Communication complexity and generalizations of the Myhill–Nerode theorem. Communication complexity was introduced by Yao [49] and measures the amount of information needed to be transferred between two processors for computing a function $f(x, y)$ when one processor receives x and the other processor receives y. A pioneering and somewhat overlooked 1986 paper by Lakshmipathy and Winklmann [34] investigated the communication complexity of graph problems by studying the following question: assume that G is a graph that can be obtained by "gluing" together two graphs G_1 and G_2 along a "boundary" of t vertices, what is the minimum amount $f(t)$ of information needed to be exchanged between two processors for deciding whether G has a certain property (for example, being Hamiltonian) when one processor receives G_1 and the other processor receives G_2 as input?

Although the work of Lakshmipathy and Winklmann [34] is completely unrelated to graphs of bounded treewidth, their notion is exactly what in our article is called the *large universe* of t-boundaried graphs that can be glued together using an operator $\oplus_c$ (for the precise definitions we refer to Definition 3.3 and Definition 3.6 in Sect. 3). Therefore, the Myhill–Nerode approach yields insights into the communication complexity of graph problems not only in the world of graphs of bounded treewidth. More specifically, it allows for proving or disproving that the minimum amount of information required to be transferred can be bounded by a function that only depends on the boundary size t. However, the Myhill–Nerode approach is not limited to the investigation of graph problems: in full generality, assume that one has

(1) a universe $\mathcal{U}$ of mathematical objects of whatever sort, on which there is a partially defined operation $\mu \colon \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ (sometimes called a *partial groupoid*), and
(2) a property $\mathcal{P} \subseteq \mathcal{U}$ of interest of these objects.

Then, one can define the *canonical Myhill–Nerode equivalence relation* $\sim_\mathcal{P}$ induced by $\mathcal{P}$ on $\mathcal{U}$ mimicking the formal language setting (there, $\mathcal{U} = \Sigma^*$ and μ is string concatenation): $x \sim_\mathcal{P} y$ if and only if, for all $z \in \mathcal{U}$, $\mu(x, z) \in \mathcal{P}$ if and only if $\mu(y, z) \in \mathcal{P}$ (assuming μ is defined in both cases). The analogy to the formal language setting naturally leads to following interesting question: for which properties (or *classes of properties*) $\mathcal{P}$ does the canonical equivalence relation $\sim_\mathcal{P}$ have a finite number of equivalence classes?

This abstract perspective often turns out to have powerful and elegant algorithmic connections, *as well as being of intrinsic interest in itself*. For example, it is intrinsically interesting that if $\mathcal{U}$ is the (large) universe of arbitrary t-boundaried graphs (of unbounded treewidth) and μ is the $\oplus_c$ gluing operation defined in Definition 3.3, then for any fixed t and any graph property $\mathcal{P}$ describable in monadic second-order logic, the canonical equivalence relation $\sim_\mathcal{P}$ has a finite number of equivalence classes. This statement is stronger than that of Courcelle's theorem [14, Theorem 6.3(2)], which proves the statement for the universe of graphs of treewidth at most t, and was first proved in the 1989 manuscript of Abrahamson and Fellows [1].

The proof is exposed in later monographs of Downey and Fellows [16,17] and exploits induction on the formula structure as well as the *method of test sets*, which

we will in the following apply also to hypergraph problems. Note that it is not obvious how to prove the above statement in full generality using other techniques that are frequently applied to solve graph problems on graphs of bounded treewidth, like dynamic programming or careful bookkeeping about partial solutions, since *there is nothing to dynamically program on* in the universe of *unbounded* treewidth graphs. Hence, the method of test sets seems to be essential if one is interested in results related to communication complexity (as we are, secondarily). Some further discussion of these different approaches and their virtues and weaknesses can be found in the concluding section of this article.

Hypergraph Cutwidth. HYPERGRAPH CUTWIDTH is a natural generalization of the NP-complete [27] CUTWIDTH problem and asks whether a hypergraph has cutwidth at most k. For a formal definition, we refer to Sect. 4. In the context of VLSI design, HYPERGRAPH CUTWIDTH is known as BOARD PERMUTATION [37]. Moreover, HYPERGRAPH CUTWIDTH naturally arises in solving CNF-SAT in the context of automatically testing digital hardware [43,46].

For the special case of CUTWIDTH on graphs, several fixed-parameter algorithms are known [2,11,19,20,45]. Cahoon and Sahni [13] showed algorithms for HYPERGRAPH CUTWIDTH with $k \leq 2$ running in $O(n)$ time for $k = 1$ and running in $O(n^3)$ time for $k = 2$ on n-vertex hypergraphs. For arbitrary k, Miller and Sudborough [37] designed an algorithm running in $O(n^{k^2+3k+3})$ time. Moreover, Nagamochi [39] presented a framework for solving cutwidth-related graph problems in $n^{O(k)}$ time.

Hypertree Width. HYPERTREE WIDTH, GENERALIZED HYPERTREE WIDTH, and FRACTIONAL HYPERTREE WIDTH are the problems of checking whether a hypergraph has (generalized, fractional) hypertree width k. All three measures are generalizations of treewidth to hypergraphs and formally defined in Sect. 5. It is known that HYPERTREE WIDTH is W[2]-hard parameterized by k [28] and that GENERALIZED HYPERTREE WIDTH remains NP-hard even for $k = 3$ [29]. Marx [36] expects FRACTIONAL HYPERTREE WIDTH also to be NP-hard for constant k. Hence, the computation of these width parameters is presumably not fixed-parameter tractable parameterized by k (that is, presumably not solvable in $O(f(k)n^c)$ time for any constant c independent of k). Hence, it makes sense to investigate whether the problems are fixed-parameter tractable with respect to larger parameters [21,32,42], like incidence treewidth.

1.2 Our Results and Organization of this Paper

In Sect. 2, we introduce the necessary graph and hypergraph notation, formally define treewidth, incidence treewidth, and tree automata.

In Sect. 3, we prove a Myhill–Nerode theorem for hypergraphs of bounded incidence treewidth. Moreover, the section discusses how the Myhill–Nerode theorem for hypergraphs yields linear-time algorithms and excludes the possibility for monadic second-order logic expressions for hypergraph problems.

In Sect. 4, we exploit the Myhill–Nerode theorem for hypergraphs to show that HY-PERGRAPH CUTWIDTH can be solved in $O(n + m)$ time for constant k, thus showing HYPERGRAPH CUTWIDTH to be *fixed-parameter linear* parameterized by k.

In Sect. 5, we exploit the Myhill–Nerode theorem to show that HYPERTREE WIDTH, GENERALIZED HYPERTREE WIDTH, and FRACTIONAL HYPERTREE WIDTH are not decidable by a finite tree automaton and, hence, not expressible in monadic second-order logic. Moreover, we obtain an indication that they are not fixed-parameter tractable parameterized by incidence treewidth.

2 Preliminaries

Graphs and hypergraphs. A *hypergraph* H is a pair (V, E), where $V(H) := V$ is a set of *vertices* and $E(H) := E$ is a set of *hyperedges* such that $e \subseteq V$ for each $e \in E$. In this work, we allow E to be a multiset and there may be singleton and empty hyperedges. If not stated otherwise, we use $n := |V|$ and $m := |E|$. Two hypergraphs G and H are isomorphic and we write $G \cong H$ if there is a bijection $f : V(G) \to V(H)$ such that e is an hyperedge with multiplicity i of G if and only if $\{f(v) \mid v \in e\}$ is an hyperedge with multiplicity i of H. The bijection f is called *(hypergraph) isomorphism*.

A *graph* is a hypergraph in which every hyperedge has cardinality two and is present at most once. Two vertices $v, w \in V$ are *adjacent* or *neighbors* if $\{v, w\} \in E$. The *(open) neighborhood* $N_G(v)$ of a vertex $v \in V$ in a graph G is the set of vertices that are adjacent to v. If the graph G is clear from the context, we drop the subscript G. A subset $S \subseteq V$ is an *independent set* if no two vertices in S are adjacent in G.

The *primal graph* of a hypergraph H, denoted $\mathcal{G}(H)$, is the graph with vertex set V that has an edge $\{u, v\}$ if u and v are together in some hyperedge in H. It is sometimes called the *Gaifman graph* of H. The *incidence graph* of a hypergraph H, denoted $\mathcal{I}(H)$, is the bipartite graph (V', E') with vertex set $V' = V \cup E$ and such that, for each $v \in V$ and $e \in E$, there is an edge $\{v, e\} \in E'$ if and only if $v \in e$.

Graph decompositions. A *tree decomposition* (T, β) for a graph $G = (V, E)$ consists of a rooted tree T and a mapping $\beta : T \to 2^V$ of each *node* x of the tree T to a subset $V_x := \beta(x) \subseteq V$, called *bag*, such that

(i) for each vertex $v \in V$, there is a node x of T with $v \in V_x$,
(ii) for each edge $\{u, w\} \in E$, there is a node x of T with $\{u, w\} \subseteq V_x$,
(iii) for each vertex $v \in V$, the nodes x of T for which $v \in V_x$ induce a subtree in T.

The *width* of a tree decomposition is the size of its largest bag minus one. The *treewidth* of G is the minimum width over all tree decompositions of G. The notions of *path decomposition* and *pathwidth* of G are defined in the same way, except that T is restricted to be a path. The *incidence treewidth* of a hypergraph is the treewidth of its incidence graph.

Graph and hypergraph representations. When speaking about linear-time solvability, it is crucial to agree on the graph and hypergraph representations we expect as input. We assume that graphs are represented as *adjacency lists*, that is, as a list of vertices, each being associated with a list of its neighbors.

We assume hypergraphs to be given as *hyperedge lists*, that is, as a list of hyperedges, each being a list of the vertices it contains. Note that a hypergraph given as hyperedge list is linear-time transformable into an adjacency list of its incidence graph and vice versa. Moreover, a hyperedge list is computable in linear time from a hypergraph given as incidence matrix.

Tree automata. A *(deterministic leaf-to-root finite-state) tree automaton* is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $q_0 \in Q$ is the *start state* and $F \subseteq Q$ is the set of *final states*, and, finally $\delta \colon (\Sigma \times Q) \cup (\Sigma \times Q \times Q) \to Q$ is the *transition function*. The set of all rooted binary trees with vertices labeled using letters from Σ is denoted by Σ^{**}. We assume that each child of a node of a rooted tree $T \in \Sigma^{**}$ is fixed to be either a "left" or a "right" child.

A tree automaton processes a tree $T \in \Sigma^{**}$ starting at its leaves to determine the state at the root node of T as follows: the state at a leaf node x of T with label $a \in \Sigma$ is determined by $\delta(a, q_0)$. The state at a node x of T with label a and a single child node y is determined by $\delta(a, q_y)$, where $q_y \in Q$ is the state at y. The state at a node x of T with label a, a left child y, and a right child z is determined by $\delta(a, q_y, q_z)$, where $q_y, q_z \in Q$ are the states of y and z, respectively.

A tree automaton *accepts* a tree $T \in \Sigma^{**}$ if its state at the root node of T is in F. A tree automaton A *recognizes* a tree language $L \subseteq \Sigma^{**}$ if, for every tree $T \in \Sigma^{**}$, the automaton A accepts T if and only if $T \in L$.

Note that an ordinary finite automaton for *words* $w \in \Sigma^*$ over the alphabet Σ can be understood as a tree automaton on rooted unary trees (paths).

3 Myhill–Nerode for Hypergraphs

The aim of this section is to generalize the Myhill–Nerode theorem from formal languages to hypergraphs. To this end, we first briefly recall the Myhill–Nerode theorem for formal languages in Sect. 3.1.

Section 3.3 will prove the Myhill–Nerode theorem for hypergraphs. Before, Sect. 3.2 will generalize the Myhill–Nerode theorem for graphs [17, Section 12.7] to vertex-colored graphs, since the Myhill–Nerode theorem for hypergraphs will exploit that every hypergraph can be represented as its incidence graph with two vertex types (or "colors"): one type representing hyperedges and one type representing the vertices of a hypergraph.

In Sect. 3.4, we finally describe how our Myhill–Nerode theorem yields linear-time algorithms for hypergraph problems and its relation to the expressibility of hypergraph properties in monadic second-order logic.

3.1 Formal Languages

The Myhill–Nerode theorem is a tool for proving or disproving that a formal language is *regular*, that is, decidable by a finite automaton. The theorem states that a language is regular if and only if its so-called *canonical right congruence* has a finite number of equivalence classes.
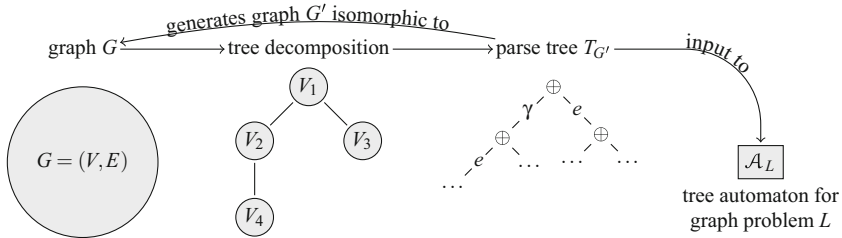
Fig. 1 Solving a graph problem L using a tree automaton: from a graph G with bounded treewidth, a minimum width tree decomposition can be computed in linear time [6]. The tree decomposition can be turned into a size-$O(n)$ expression over a set $\{\emptyset, e, u, \gamma, i, \oplus\}$ of operators in linear time such that the value of the expression is a graph G' isomorphic to G [17, Theorem 12.7.1]. The parse tree or expression tree $T_{G'}$ of the expression is fed to a tree automaton $\mathcal{A}_L$ that accepts $T_{G'}$ in $O(n)$ time if and only if $G' \in L$. The existence of $\mathcal{A}_L$ for the problem L can be proven or disproven by the Myhill–Nerode theorem for graphs

Definition 3.1 Let $L \subseteq \Sigma^*$ be a language. The *canonical right congruence* $\sim_L$ is defined as follows: for $v, w \in \Sigma^*$, $v \sim_L w : \iff \forall x \in \Sigma^* : vx \in L \iff wx \in L$, where vx is the concatenation of v and x.

Example 3.2 Consider the language $L := \{a^i b^j \mid i, j \in \mathbb{N}\} \subseteq \{a, b\}^*$ consisting of words starting with an arbitrary number of a's and ending in an arbitrary number of b's. Then, $a \sim_L aa$. However, $a \nsim_L ab$, since, for example, $aa \in L$ but $aba \notin L$.

Obviously, for a language $L \in \Sigma^*$, the canonical right congruence $\sim_L$ is an equivalence relation, that is, it is reflexive, symmetric, and transitive. The *index* of an equivalence relation is the number of its equivalence classes.

Myhill–Nerode Theorem *A language $L \subseteq \Sigma^*$ is recognizable by a finite automaton if and only if the canonical right congruence $\sim_L$ has finite index.*

Thus, the Myhill–Nerode theorem gives a necessary and sufficient condition for a language being recognizable by a finite automaton.

3.2 Colored Graphs

In order to show the Myhill–Nerode theorem for hypergraphs, we first present it for vertex-colored graphs. Downey and Fellows [17, Section 12.7] already proved the Myhill–Nerode theorem for graphs without colors; Fig. 1 gives a rough overview of the technique. We will see that lifting it to vertex-colored graphs is straightforward. Indeed, Courcelle and Engelfriet [14, Section 4.2.2] provide an even more general Myhill–Nerode theorem for graphs with vertex colors as well as edge colors. For our purposes, however, a vertex-colored variant is sufficient and we show it here as an introduction into the necessary concepts towards a Myhill–Nerode theorem for hypergraphs.

In order to apply tree automata to graphs, we first show how every graph of bounded treewidth can be represented by an expression over a constant-size set of operators, and, consequently, as the parse tree or expression tree of that expression (Fig. 2 gives an

Fig. 2 The expression tree or
parse tree of the arithmetic
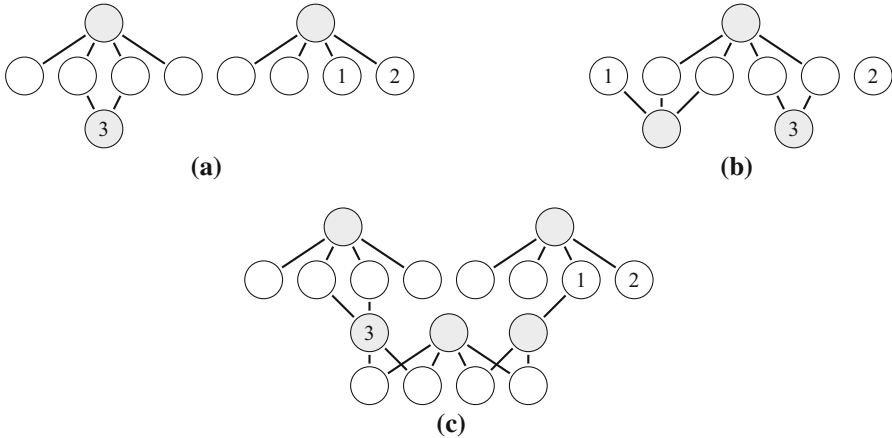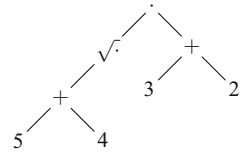expression $\sqrt{5+4} \cdot (3+2)$

Fig. 3 Two color-compatible 3-boundaried graphs G and H and their glued graph, where the boundary vertices are marked by their label. **a** A 2-colored 3-boundaried graph G. **b** A 2-colored 3-boundaried graph H. **c** The glued graph $G \oplus_c H$

example for a parse tree). Herein, the crucial operator corresponds to the concatenation of words in the language setting of the Myhill–Nerode theorem: like every word with more than one letter is the concatenation of shorter words, we will see that every graph of treewidth $t - 1$ with more than t vertices is isomorphic to the result of *gluing* smaller graphs together at a *boundary* consisting of t vertices.[1] This is formalized by the definition below and illustrated in Fig. 3.

Definition 3.3 A *t-boundaried graph* G is a graph with t distinguished vertices that are labeled from 1 to t. These labeled vertices are called *boundary vertices*. The *boundary* $\partial(G)$ is the set of boundary vertices of G.

Two colored t-boundaried graphs G_1 and G_2 are *isomorphic* and we write $G_1 \cong G_2$ if there is an isomorphism for the underlying (uncolored and unlabeled) graphs mapping each vertex to a vertex with the same color (but ignoring labels).

Let G_1 and G_2 be t-boundaried graphs whose vertices are colored with colors in $\{1, \ldots, c_{\max}\}$. We say that G_1 and G_2 are *color-compatible* if the vertices with the same labels in $\partial(G_1)$ and $\partial(G_2)$ have the same color.

For two color-compatible t-boundaried graphs, we denote by $G_1 \oplus_c G_2$ the colored graph obtained by *gluing* G_1 and G_2, that is, by taking the disjoint union of G_1 and G_2

---

[1] For keeping the presentation simpler, we leave graphs with less than t vertices out of consideration: since we only consider constant values of t throughout this work, any problem restricted to graphs with less than t vertices is a finite problem and can therefore be trivially solved by a finite automaton.

and identifying vertices of $\partial(G_1)$ and $\partial(G_2)$ having the same label; the vertex colors of $G_1 \oplus_c G_2$ are inherited from G_1 and G_2. For two color-incompatible graphs G_1 and G_2, we leave $\oplus_c$ undefined.

Together with $\oplus_c$, we use the following set of operators to create primitive graphs that can be glued together to larger graphs using $\oplus_c$, and to arbitrarily permute the labels on the boundary vertices.

Definition 3.4 The *size-t parsing operators* for $\{1, \ldots, c_{max}\}$-colored t-boundaried graphs are defined as follows:

(i) $\{\emptyset_{n_1, \ldots, n_{c_{max}}} \mid \sum_{i=1}^{c_{max}} n_i = t\}$ is a family of nullary operators that creates a graph consisting of isolated boundary vertices $1, \ldots, t$, of which the first n_1 vertices get color 1, the next n_2 vertices get color 2, and so on.
(ii) e is a unary operator that adds an edge between the boundary vertices labeled 1 and 2.
(iii) $\{u_\ell : 1 \leq \ell \leq c_{max}\}$ is a family of unary operators that add a new boundary vertex of color ℓ and labels it 1, unlabeling the vertex previously labeled 1.
(iv) γ is a unary operator that cyclically shifts the boundary. That is, γ moves label j to the vertex with label $j + 1 \pmod{t}$.
(v) i is a unary operator that assigns the label 1 to the vertex currently labeled 2 and label 2 to the vertex with label 1.
(vi) $\oplus_c$ is our gluing operator from Definition 3.3.

For a constant number of colors c_{max}, the set of size-t parsing operators is finite. Moreover, for $c_{max} = 1$, the given operators coincide with those given by Downey and Fellows [17, Section 12.7] for uncolored graphs, which allows us to show the following theorem:

Theorem 3.5 *Let G be a $\{1,\ldots,c_{max}\}$-colored graph with constant treewidth $t - 1$ and at least t vertices.*

Then, in linear time, G can be transformed into an expression over the size-t operators in Definition 3.4 whose value is a graph H that is isomorphic to G, that is, when ignoring the labels of H.

Proof For $c_{max} = 1$, Downey and Fellows [17, Theorem 12.7.1] provide a linear-time procedure for converting a tree decomposition of width $t - 1$ of a graph G into an expression over the size-t operators in Definition 3.4 such that the value of the expression is a graph isomorphic to G. This procedure is easily adapted for larger c_{max}: whenever Downey and Fellows [17] introduce vertices using $\emptyset_1$ or u_1 in the case $c_{max} = 1$, we introduce them using $\emptyset_{n_1, \ldots, c_{max}}$ and u_ℓ with the colors they have in G. □

We are now at a point where we can get each graph of bounded treewidth into a representation that we can feed into a tree automaton: we use the parse tree (or expression tree) of an expression over the operators in Definition 3.4. A central question remains: which graph problems can be decided by a tree automaton operating on such a parse tree? The Myhill–Nerode theorem for colored graphs will give a sufficient and necessary condition. To state the theorem, we first lift the concept of a canonical right congruence from the language setting (Definition 3.1) to graphs.

Definition 3.6 Let $\mathcal{U}_{t,c_{\max}}^{\text{large}}$ be the *large universe* of all $\{1, \ldots, c_{\max}\}$-colored t-boundaried graphs and $\mathcal{U}_{t,c_{\max}}^{\text{small}} \subseteq \mathcal{U}_{t,c_{\max}}^{\text{large}}$ be the *small universe* of $\{1, \ldots, c_{\max}\}$-colored t-boundaried graphs that can be generated by the size-t operators in Definition 3.4.

For $U \in \{\mathcal{U}_{t,c_{\max}}^{\text{small}}, \mathcal{U}_{t,c_{\max}}^{\text{large}}\}$, we say that $F \subseteq U$ is a *graph problem* if, for all $G \in F$ and $H \in U$ with $G \cong H$, we also have $H \in F$. That is, we assume graph problems to be closed under isomorphism and, in particular, that changing vertex labels does not influence membership in F.

Finally, for a graph problem $F \subseteq U$, where $U \in \{\mathcal{U}_{t,c_{\max}}^{\text{small}}, \mathcal{U}_{t,c_{\max}}^{\text{large}}\}$, we define the *canonical right congruence* $\sim_F$ *over* U for F as follows: for $G_1, G_2 \in U$, $G_1 \sim_F G_2$ if and only if G_1 and G_2 are color-compatible and if for all color-compatible $H \in U$, we have $G_1 \oplus_c H \in F \iff G_2 \oplus_c H \in F$.

Theorem 3.5 might mislead to the impression that $\mathcal{U}_{t,c_{\max}}^{\text{small}}$ contains all treewidth-$(t-1)$ graphs of $\mathcal{U}_{t,c_{\max}}^{\text{large}}$. However, this is not the case: for example, a path of length more than one whose first vertex has label 1 and whose last vertex has label 2 has treewidth 1 and is contained in $\mathcal{U}_{2,1}^{\text{large}}$. However, it cannot be generated by the size-2 operators in Definition 3.4 and, hence, is not in $\mathcal{U}_{2,1}^{\text{small}}$. We will see this detail to be important when showing that a graph problem is *not* recognizable by a finite tree automaton.

Theorem 3.7 *Let* $F \subseteq \mathcal{U}_{t,c_{\max}}^{small}$ *be a graph problem. The following statements are equivalent:*

(i) The collection of parse trees generating the graphs in F is recognizable by a finite tree automaton.

(ii) The canonical right congruence $\sim_F$ has finite index over $\mathcal{U}_{t,c_{\max}}^{small}$.

Downey and Fellows [17, Theorem 12.7.2] proved Theorem 3.7 for uncolored graphs, that is, for $c_{\max} = 1$.[2] The case $c_{\max} > 1$ can be proven analogously.

3.3 Hypergraphs

In this section, we show how tree automata can be used to recognize hypergraph properties and, in the form of a Myhill–Nerode theorem for hypergraphs, a necessary and sufficient characterization for the hypergraph properties that a tree automaton can decide. To this end, we first define the notion of gluing for hypergraphs.

Definition 3.8 A t-boundaried hypergraph H has t distinguished vertices and hyperedges labeled from 1 to t called *boundary objects*. The *boundary* $\partial(H)$ is the set of all boundary objects.

Two t-boundaried hypergraphs are *gluable* if no vertex of one hypergraph has the label of a hyperedge of the other hypergraph.

Let H_1 and H_2 be gluable t-boundaried hypergraphs. We denote by $H_1 \oplus_h H_2$ the t-boundaried hypergraph obtained by taking the disjoint union of H_1 and H_2, identifying each labeled vertex of H_1 with the vertex of H_2 with the same label, and replacing the hyperedges with the same label ℓ by their union.

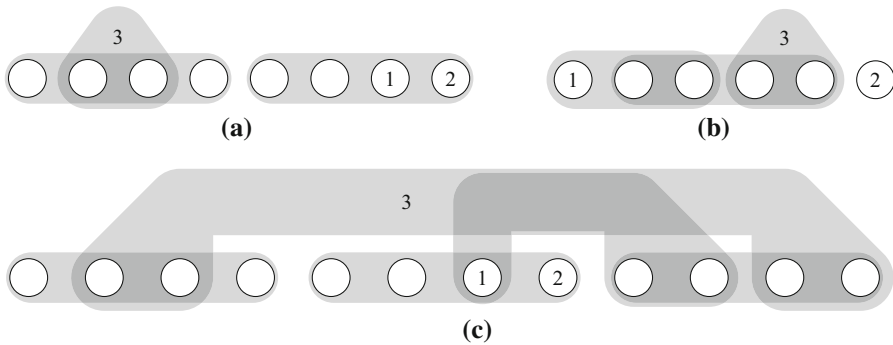[2] The proof of (ii) → (i) given by Downey and Fellows [17] is flawed but repairable [4].

Fig. 4 The two hypergraphs represented by the t-boundaried hypergraph generators G and H in Fig. 3 and the glued hypergraph $\mathcal{H}(G) \oplus_h \mathcal{H}(H) = \mathcal{H}(G \oplus_c H)$. **a** The 3-boundaried hypergraph $\mathcal{H}(G)$. **b** The 3-boundaried hypergraph $\mathcal{H}(H)$. **c** The glued hypergraph $\mathcal{H}(G) \oplus_h \mathcal{H}(H) = \mathcal{H}(G \oplus_c H)$

In order to apply tree automata to hypergraphs, in contrast to Sect. 3.2 for colored graphs, we will not define a set of additional operators for generating hypergraphs. Instead, we will generate hypergraphs from two-colored incidence graphs: vertices of one color will represent the vertices of the hypergraph, vertices of the other color will represent the hyperedges. That is, instead of solving a hypergraph problem, we will in fact solve a graph problem on colored incidence graphs. The goal of the next definition is to give a representation of a hypergraph problem as a graph problem. It is illustrated in Fig. 4.

Definition 3.9 A t-*boundaried hypergraph generator* is a $\{1, 2\}$-colored t-boundaried graph $G = (U \uplus W, E)$ such that all vertices in U have color 1 and all vertices in W have color 2, and each of U and W form an independent set.

For a t-boundaried hypergraph generator $G = (U \uplus W, E)$, we denote by $\mathcal{H}(G)$ the t-boundaried hypergraph with the vertex set U and the hyperedge set $\{N(w) \mid w \in W\}$. Moreover, each vertex of $\mathcal{H}(G)$ inherits its label from G and each hyperedge e in $\mathcal{H}(G)$ inherits its label from the vertex $w \in W$ of G that induced e.

For a set $F \subseteq \mathcal{U}_{t,2}^{\text{small}}$ of t-boundaried hypergraph generators, we denote $\mathcal{H}(F) := \bigcup_{G \in F} \mathcal{H}(G)$ and we call F *generator-total* if, for all t-boundaried hypergraph generators $G \in \mathcal{U}_{t,2}^{\text{small}}$, $\mathcal{H}(G) \in \mathcal{H}(F) \implies G \in F$.

We use $\mathcal{H}_t^{\text{large}}$ to denote the *large universe* of all t-boundaried hypergraphs and by $\mathcal{H}_t^{\text{small}}$ we denote the *small universe* $\mathcal{H}(\mathcal{U}_{t,2}^{\text{small}})$, that is, the t-boundaried hypergraphs that can be generated from t-boundaried hypergraph generators created by the operators in Definition 3.4.

We say that $\mathcal{F} \subseteq U$ for $U \in \{\mathcal{H}_t^{\text{large}}, \mathcal{H}_t^{\text{small}}\}$ is a *hypergraph problem* if, for all $G \in \mathcal{F}$ and $H \in U$ with $G \cong H$, we also have $H \in \mathcal{F}$. That is, we assume hypergraph problems to be closed under isomorphism and, in particular, that changing boundary labels does not influence membership in $\mathcal{F}$.

The following observation allows us, where helpful, to denote hypergraphs H using $\mathcal{H}(G)$ for some graph G with $\mathcal{H}(G) = H$, and to denote hypergraph problems $\mathcal{F}$ using $\mathcal{H}(F)$ for some generator-total graph problem F.

Observation 3.10 (i) A graph $G \in \mathcal{U}_{t,2}^{\text{large}}$ is isomorphic to the incidence graph
   of $\mathcal{H}(G) \in \mathcal{H}_t^{\text{large}}$. Therefore, the treewidth of G equals the incidence treewidth
   of $\mathcal{H}(G)$.

(ii) For two graphs $G, H \in \mathcal{U}_{t,2}^{\text{large}}$, we have $\mathcal{H}(G) \oplus_h \mathcal{H}(H) = \mathcal{H}(G \oplus_c H)$.

(iii) For a generator-total $F \subseteq \mathcal{U}_{t,2}^{\text{small}}$ and each t-boundaried hypergraph genera-
   tor $G \in \mathcal{U}_{t,2}^{\text{small}}$, we have $G \in F$ if and only if $\mathcal{H}(G) \in \mathcal{H}(F)$.[3]

(iv) For every t-boundaried hypergraph $H \in \mathcal{H}_t^{\text{small}}$, by definition of $\mathcal{H}_t^{\text{small}}$, there
   is a t-boundaried graph $G \in \mathcal{U}_{t,2}^{\text{small}}$ such that $\mathcal{H}(G) = H$. Consequently, for
   every hypergraph problem $\mathcal{F} \subseteq \mathcal{H}_t^{\text{small}}$, there is a generator-total $F \subseteq \mathcal{U}_{t,2}^{\text{small}}$ with
   $\mathcal{H}(F) = \mathcal{F}$. Moreover, in terms of Definition 3.6, F is a graph problem.

In order to state the Myhill–Nerode theorem for hypergraphs, we define the canonical
right congruence for hypergraphs.

Definition 3.11 Let $\mathcal{F} \subseteq U$ for $U \in \{\mathcal{H}_t^{\text{large}}, \mathcal{H}_t^{\text{small}}\}$ be a hypergraph problem. We
define the *canonical right congruence* $\sim_{\mathcal{F}}$ *over* U for $\mathcal{F}$ as follows: for $G_1, G_2 \in U$,
$G_1 \sim_{\mathcal{F}} G_2$ if and only if G_1 and G_2 are gluable and for all $H \in U$ that are gluable
to G_1 and G_2, $G_1 \oplus_h H \in \mathcal{F} \iff G_2 \oplus_h H \in \mathcal{F}$.

We now state our Myhill–Nerode theorem for hypergraphs. As Theorem 3.7, the
following theorem only makes a statement about when a tree automaton can decide
hypergraph problems $\mathcal{F} \subseteq \mathcal{H}_t^{\text{small}}$. However, in Sect. 3.4, we will see that this restriction
is not important in most cases.

Theorem 3.12 *Let* $\mathcal{F} \subseteq \mathcal{H}_t^{small}$ *be a hypergraph problem, that is,* $\mathcal{F} = \mathcal{H}(F)$ *for some
generator-total* $F \subseteq \mathcal{U}_{t,2}^{small}$. *The following statements are equivalent:*

*(i) The collection of parse trees generating the graphs in F is recognizable by a tree
     automaton.*
(ii) The canonical right congruence $\sim_{\mathcal{F}}$ has finite index over $\mathcal{H}_t^{small}$.
(iii) The canonical right congruence $\sim_F$ has finite index over $\mathcal{U}_{t,2}^{small}$.

*Moreover, if the index p of $\sim_{\mathcal{F}}$ and the index q of $\sim_F$ are finite, they bound each
other as* $2^t q \geq p \geq q/2^t - 1$.

Proof Since $F \subseteq \mathcal{U}_{t,2}^{\text{small}}$, we can apply Theorem 3.7, which states that (i) and (iii) are
equivalent. It remains to show that (iii) and (ii) are equivalent. That is, we show that
$\sim_{\mathcal{F}}$ has finite index over $\mathcal{H}_t^{\text{small}}$ if and only if $\sim_F$ has finite index over $\mathcal{U}_{t,2}^{\text{small}}$.

First, assume that $\sim_F$ has infinite index over $\mathcal{U}_{t,2}^{\text{small}}$. We show that $\sim_{\mathcal{F}}$ has
infinite index over $\mathcal{H}_t^{\text{small}}$. Since $\sim_F$ has infinite index over $\mathcal{U}_{t,2}^{\text{small}}$, there is
an infinite set $\{G_1, G_2, G_3, \ldots\} \subseteq \mathcal{U}_{t,2}^{\text{small}}$ of graphs that are pairwise non-
equivalent under $\sim_F$. Since there are only 2^t possibilities to assign two colors
to t boundary vertices, there is an infinite number of color-compatible graphs

---

[3] If F is not generator-total, it might be that $G \notin F$ but $\mathcal{H}(G) \in \mathcal{H}(F)$ because $H \in F$ for some t-
boundaried hypergraph generator $H \neq G$ with $\mathcal{H}(G) = \mathcal{H}(H)$: the graphs G and H might represent the
hyperedges of $\mathcal{H}(G) = \mathcal{H}(H)$ using different mathematical objects.

among $\{G_1, G_2, \ldots\}$. Moreover, notice that all graphs $G_i \in \mathfrak{U}_{t,2}^{\mathrm{small}}$ that are not t-bounderied hypergraph generators are equivalent under $\sim_F$: since F contains only t-bounderied hypergraph generators, G_i cannot be completed into graphs in F by gluing any graph onto G_i. Therefore, without loss of generality, we assume that $\{G_1, G_2, \ldots\}$ are pairwise color-compatible t-bounderied hypergraph generators. Now, for each pair G_i, G_j, there is a graph $H_{ij} \in \mathfrak{U}_{t,2}^{\mathrm{small}}$ such that, without loss of generality, $G_i \oplus_{\mathrm{c}} H_{ij} \in F$ but $G_j \oplus_{\mathrm{c}} H_{ij} \notin F$. From $G_i \oplus_{\mathrm{c}} H_{ij} \in F$, it follows that H_{ij} is a t-bounderied hypergraph generator that is color-compatible with G_i. Hence, $\mathcal{H}(H_{ij}) \in \mathcal{H}_t^{\mathrm{small}}$. Now, from $G_i \oplus_{\mathrm{c}} H_{ij} \in F$, we get $\mathcal{H}(G_i) \oplus_{\mathrm{h}} \mathcal{H}(H_{ij}) = \mathcal{H}(G_i \oplus_{\mathrm{c}} H_{ij}) \in \mathcal{H}(F) = \mathcal{F}$. Moreover, since F is generator-total, from $G_j \oplus_{\mathrm{c}} H_{ij} \notin F$ it follows that $\mathcal{H}(G_j) \oplus_{\mathrm{h}} \mathcal{H}(H_{ij}) = \mathcal{H}(G_j \oplus_{\mathrm{c}} H_{ij}) \notin \mathcal{H}(F) = \mathcal{F}$. That is, $\mathcal{H}(G_i) \not\sim_{\mathcal{F}} \mathcal{H}(G_j)$ over $\mathcal{H}_t^{\mathrm{small}}$ and, therefore, $\sim_{\mathcal{F}}$ has infinite index.

Now, assume that $\sim_{\mathcal{F}}$ has infinite index over $\mathcal{H}_t^{\mathrm{small}}$. We show that $\sim_F$ has infinite index over $\mathfrak{U}_{t,2}^{\mathrm{small}}$. Since $\sim_{\mathcal{F}}$ has infinite index over $\mathcal{H}_t^{\mathrm{small}}$, there is a set $\{\mathcal{H}(G_1), \mathcal{H}(G_2), \mathcal{H}(G_3), \ldots\} \subseteq \mathcal{H}_t^{\mathrm{small}}$ of hypergraphs that are pairwise non-equivalent under $\sim_{\mathcal{F}}$. Since there are only 2^t partitions of t labels into hyperedge labels and vertex labels, there is an infinite number of pairwise gluable hypergraphs among $\{\mathcal{H}(G_1), \mathcal{H}(G_2), \ldots\}$. Therefore, without loss of generality, assume that all these hypergraphs are pairwise gluable. Now, for each pair $\mathcal{H}(G_i), \mathcal{H}(G_j)$, there is a hypergraph $\mathcal{H}(H_{ij}) \in \mathcal{H}_t^{\mathrm{small}}$ such that, without loss of generality, we have $\mathcal{H}(G_i) \oplus_{\mathrm{h}} \mathcal{H}(H_{ij}) \in \mathcal{H}(F) = \mathcal{F}$ but $\mathcal{H}(G_j) \oplus_{\mathrm{h}} \mathcal{H}(H_{ij}) \notin \mathcal{H}(F) = \mathcal{F}$. Since $\mathcal{H}(G_i) \oplus_{\mathrm{h}} \mathcal{H}(H_{ij}) = \mathcal{H}(G_i \oplus_{\mathrm{c}} H_{ij})$ and F is generator-total, we have $G_i \oplus_{\mathrm{c}} H_{ij} \in F$. Moreover, $G_j \oplus_{\mathrm{c}} H_{ij} \notin F$. Since $H_{ij} \in \mathfrak{U}_{t,2}^{\mathrm{small}}$, it follows that $\sim_F$ has infinite index over $\mathfrak{U}_{t,2}^{\mathrm{small}}$.

Finally, observe that our proof yields even more information in the case that $\sim_F$, and equivalently $\sim_{\mathcal{F}}$, have finite index: we have first shown that, for any set of q graphs that are pairwise nonequivalent under $\sim_F$, there are at least $p \geq \lceil q/2^t \rceil - 1$ hypergraphs nonequivalent under $\sim_{\mathcal{F}}$. We have then shown that, for any set of p hypergraphs that are pairwise nonequivalent under $\sim_{\mathcal{F}}$, there are at least $q \geq \lceil p/2^t \rceil$ graphs nonequivalent under $\sim_F$. Hence, for the index p of $\sim_{\mathcal{F}}$ and the index q of $\sim_F$, we have $2^t q \geq p \geq q/2^t - 1$. $\qquad\square$

3.4 Fixed-Parameter Algorithms and Monadic Second-Order Logic

In Sect. 3.3, we have seen a tool allowing us to show when a hypergraph problem $\mathcal{F} \in \mathcal{H}_t^{\mathrm{small}}$ can be recognized by a finite tree automaton. The provided Theorem 3.12, however, is strongly tied to the representation of hypergraphs as incidence graphs in $\mathfrak{U}_{t,2}^{\mathrm{small}}$. This section shows three corollaries to ease the application of Theorem 3.12 for classifying hypergraph problems.

Showing tractability and constructing tree automata. The following corollary will make it easier to show that a hypergraph problem is fixed-parameter linear parameterized by incidence treewidth. Essentially, we do not have to care about whether the hypergraphs we consider are contained in $\mathcal{H}_t^{\mathrm{small}}$.

Corollary 3.13 *Let $\mathcal{F} \subseteq \mathcal{H}_t^{large}$ be a decidable hypergraph problem and that is restricted to hypergraphs of constant incidence treewidth $t - 1$.*

Given a hypergraph $H \in \mathcal{H}_t^{large}$ and a constant upper bound on the index of $\sim_{\mathcal{F}}$ over $\mathcal{H}_t^{large}$, we can compute in constant time a tree automaton $\mathcal{A}$ and in linear time a tree T such that $\mathcal{A}$ processes T in linear time and accepts T if and only if $H \in \mathcal{F}$.

Proof Let $F \subseteq \mathcal{U}_{t,2}^{small}$ be generator-total such that $\mathcal{F} \cap \mathcal{H}_t^{small} = \mathcal{H}(F)$ and let p be the constant given upper bound on $\sim_{\mathcal{F}}$. The proof relies on two claims:

(i) $H \in \mathcal{F}$ holds if and only if all graphs $G \in \mathcal{U}_{t,2}^{small}$ with $\mathcal{H}(G) \cong H$ are in F.

(ii) $\sim_F$ has index $q \leq 2^t(p + 1)$ over $\mathcal{U}_{t,2}^{small}$.

From (i) then immediately follows that a tree automaton $\mathcal{A}$ deciding F decides $H \in \mathcal{F}$ correctly when fed the parse tree T_G of any $G \in \mathcal{U}_{t,2}^{small}$ with $\mathcal{H}(G) \cong H$. By Theorem 3.5, this G exists and we obtain the parse tree T_G in linear time from the incidence graph of H.

From (ii) and Theorem 3.12, it follows that the tree automaton $\mathcal{A}$ indeed exists. It can be constructed in constant time given that we know a constant upper bound s on the number of states of $\mathcal{A}$: the crucial observation is that $\mathcal{A}$ reaches at least one state twice when processing a parse tree of height greater than s. Thus, for any parse tree T of height greater than s, there is a parse tree T' of height at most s such that $\mathcal{A}$ accepts T if and only if it accepts T'. It follows that we only have to construct $\mathcal{A}$ so that it works correctly on all parse trees of height at most s. Since our operators in Definition 3.4 are all nullary, unary, or binary, the parse trees of expressions over them are binary trees. Thus, there are only a constant number of parse trees of height at most s. Moreover, for any parse tree T of height at most s, we can decide in constant time whether the hypergraph it generates is in $\mathcal{F}$, since $\mathcal{F}$ is decidable. Hence, we can construct in constant time by brute force a tree automaton $\mathcal{A}$ that correctly answers for all parse trees of height at most s and, consequently, recognizes F. Moreover, since $\mathcal{A}$ has a constant number of states, it takes only linear time for $\mathcal{A}$ to process any parse tree.

The constant upper bound on the number of states of $\mathcal{A}$ we obtain as follows: the index of $\sim_F$ is bounded by (ii), which, in turn bounds the number of states of $\mathcal{A}$ [17, Theorem 12.7.2]. Thus, it only remains to prove (i) and (ii).

(i) First, assume that there is some graph $G \in \mathcal{U}_{t,2}^{small}$ with $\mathcal{H}(G) \cong H$ in F. Then, since G is a t-boundaried hypergraph generator in $\mathcal{U}_{t,2}^{small}$, $\mathcal{H}(G) \in \mathcal{H}(F)$. Since $\mathcal{H}(F) \subseteq \mathcal{F}$ and $\mathcal{F}$ is closed under isomorphism, we conclude $H \in \mathcal{F}$. If, for the opposite direction, $H \in \mathcal{F}$, then let $G \in \mathcal{U}_{t,2}^{small}$ be any graph such that $\mathcal{H}(G) \cong H$. Since $G \in \mathcal{U}_{t,2}^{small}$, we have $\mathcal{H}(G) \in \mathcal{H}_t^{small}$. Moreover, since $\mathcal{F}$ is closed under isomorphism, $\mathcal{H}(G) \in \mathcal{F}$ and, hence, $\mathcal{H}(G) \in \mathcal{H}(F)$. Finally, since F is generator-total, we have $G \in F$.

(ii) We show that the index p' of $\sim_{\mathcal{H}(F)}$ over $\mathcal{H}_t^{small}$ is at most p. Then, from Theorem 3.12, it follows that $p \geq p' \geq q/2^t p - 1$ and, hence, $q \leq 2^t(p + 1)$. Thus, we only have to show, for any $\mathcal{H}(G_1), \mathcal{H}(G_2) \in \mathcal{H}_t^{small}$ equivalent under $\sim_{\mathcal{F}}$, that they are also equivalent under $\sim_{\mathcal{H}(F)}$. This is trivial, since, for $i \in \{1, 2\}$ and any $\mathcal{H}(H) \in \mathcal{H}_t^{small}$, we have $\mathcal{H}(G_i) \oplus_h \mathcal{H}(H) \in \mathcal{H}(F) \iff$

$\mathcal{H}(G_i) \oplus_h \mathcal{H}(H) \in \mathcal{F}$, since $\mathcal{H}(G_i) \oplus_h \mathcal{H}(H) = \mathcal{H}(G_i \oplus_c H) \in \mathcal{H}_t^{small}$ and $\mathcal{H}(F) = \mathcal{F} \cap \mathcal{H}_t^{small}$. □

From Corollary 3.13, it follows that, to obtain a fixed-parameter linear algorithm for some hypergraph problem $\mathcal{F}$ parameterized by incidence treewidth, we just have to show that $\sim_\mathcal{F}$ has finite index over $\mathcal{H}_t^{large}$.

Showing intractability. We have seen that it was enough to show that $\sim_\mathcal{F}$ has finite index over $\mathcal{H}_t^{large}$ to show that a hypergraph problem $\mathcal{F} \subseteq \mathcal{H}_t^{large}$ of hypergraphs with incidence treewidth $t-1$ is decidable by a tree automaton. To show the opposite, it is not sufficient to show that $\sim_\mathcal{F}$ has infinite index over the hypergraphs with treewidth $t-1$ in $\mathcal{H}_t^{large}$: assume, for example, that there are two hypergraphs $H_i, H_j \in \mathcal{H}_t^{small}$ that are non-equivalent under $\sim_\mathcal{F}$. Then, there is some $H_{ij} \in \mathcal{H}_t^{large}$ satisfying $\mathcal{H}(G_i) \oplus_h H_{ij} \in \mathcal{F}$ but $\mathcal{H}(G_j) \oplus_h H_{ij} \notin \mathcal{F}$. If $H_{ij} \notin \mathcal{H}_t^{small}$, then this does not necessarily mean that H_i and H_j are nonequivalent under $\sim_\mathcal{F}$ over $\mathcal{H}_t^{small}$. Thus, we cannot conclude that $\sim_\mathcal{F}$ has infinite index over $\mathcal{H}_t^{small}$ and Theorem 3.12 is inapplicable. However, the following corollary gives a simple criterion for intractability.

Corollary 3.14 *Let $\mathcal{F} \subseteq \mathcal{H}_t^{large}$ be a hypergraph problem and $\mathcal{F}_t \subseteq \mathcal{F}$ be an arbitrary subset of hypergraphs H whose incidence graphs have tree decompositions of width $t-1$ such that $\partial(H)$ is a bag.*

If $\sim_{\mathcal{F}_t}$ has infinite index over $\mathcal{H}_t^{large}$, then $\sim_F$ has infinite index over $\mathcal{U}_{t,2}^{small}$ for $F \subseteq \mathcal{U}_{t,2}^{small}$ being the generator-total graph problem such that $\mathcal{H}(F) = \mathcal{F} \cap \mathcal{H}_t^{small}$.

Consequently, there is no tree automaton that decides $H \in \mathcal{F}$ correctly when fed the parse tree of the incidence graph of H.

Proof Any hypergraph $H \in \mathcal{F}_t$ allows for a tree decomposition T of width $t-1$ of its incidence graph that has a bag $\partial(H)$. The procedure by Downey and Fellows [17, Theorem 12.7.1] produces a parse tree for a graph $G \in \mathcal{U}_{t,2}^{small}$ with $\mathcal{H}(G) \cong H$. The crucial observation is that, when choosing the bag $\partial(H)$ as the root of the tree decomposition T, the procedure generates a parse tree for a graph $G \in \mathcal{U}_{t,2}^{small}$ with $\mathcal{H}(G) \cong_t H$, where we use $\cong_t$ to denote that there is an isomorphism between $\mathcal{H}(G)$ and H that maps the t boundary objects of $\mathcal{H}(G)$ to boundary objects in H with the same label.

Now, let $\{H_1, H_2, \ldots\} \subseteq \mathcal{F}_t$ be a set of hypergraphs that are pairwise non-equivalent with respect to $\sim_{\mathcal{F}_t}$. As before, we may assume that they are pairwise gluable. Hence, for each pair H_i, H_j, there is a hypergraph $H_{ij} \in \mathcal{H}_t^{large}$ such that $H_i \oplus_h H_{ij} \in \mathcal{F}_t$ but $H_j \oplus_h H_{ij} \notin \mathcal{F}_t$. Since $H_i \oplus_h H_{ij} \in \mathcal{F}_t$, the hypergraph $H_i \oplus_h H_{ij}$ has a tree decomposition of width $t-1$ with a bag $\partial(H_i \oplus_h H_{ij})$ and, hence, H_{ij} has such a tree decomposition as well. It follows that there are graphs $G_i, G_j, G_{ij} \in \mathcal{U}_{t,2}^{small}$ such that $\mathcal{H}(G_i) \cong_t H_i$, $\mathcal{H}(G_j) \cong_t H_j$, and $\mathcal{H}(G_{ij}) \cong_t H_{ij}$. Then, $\mathcal{H}(G_i \oplus_c G_{ij}) = \mathcal{H}(G_i) \oplus_h \mathcal{H}(G_{ij}) \cong_t H_i \oplus_h H_{ij} \in \mathcal{F}$. Since $\mathcal{F}$ is closed under isomorphism, it follows that $\mathcal{H}(G_i \oplus_c G_{ij}) \in \mathcal{F} \cap \mathcal{H}_t^{small}$. Since F is generator-total, $G_i \oplus_c G_{ij} \in F$. With the same argumentation, it follows that $G_j \oplus_c G_{ij} \notin F$.

It follows that $\sim_F$ has infinite index over $\mathcal{U}_{t,2}^{small}$ and by Theorem 3.12, there is no tree automaton recognizing the parse trees in F. □

Excluding expressibility in monadic second-order logic. A standard way of showing linear-time solvability of a graph problem F on graphs of bounded treewidth is expressing the property of being a yes-instance of F in monadic second-order logic of graphs [41, Section 10.6].

Previously, we have seen how to show that some hypergraph problem $\mathcal{F}$ cannot be solved by a finite tree automaton. An immediate consequence is that the property of being a yes-instance for $\mathcal{F}$ is not expressible in monadic second-order logic for hypergraphs; we now give a little detail about this connection.

Definition 3.15 (Monadic second-order logic for graphs and hypergraphs) A formula in the MS_1-*logic for* $\{1, \ldots, c_{\max}\}$-*colored graphs* may consist of the logic operators $\vee, \wedge, \neg$, vertex variables, set variables, quantifiers $\exists$ and $\forall$ over vertices and vertex sets, and the predicates

(i) $x \in X$ for a vertex variable x and a set X,
(ii) $\mathrm{adj}(v, w)$, being true if v and w are adjacent vertices,
(iii) $\mathrm{col}_i(v)$ for $1 \leq i \leq c_{\max}$, being true if v is a vertex with color i,
(iv) equality of vertex variables and set variables.

A formula of the MS_2-*logic for hypergraphs* may consist of the logic operators $\vee, \wedge, \neg$, vertex variables, hyperedge variables, set variables, quantifiers $\exists$ and $\forall$ over vertices, hyperedges, and sets, and the predicates

(i) $x \in X$ for a vertex or hyperedge variable x and a set X,
(ii) $\mathrm{inc}(e, v)$, being true if e is a hyperedge containing v,
(iii) $\mathrm{adj}(v, w)$, being true if v and w occur in a common hyperedge, and
(iv) equality of vertex variables, edge variables, and set variables.

We use upper-case letters for set variables and lower-case letters for vertex and hyperedge variables.

Corollary 3.16 *Let* $\mathcal{F} \subseteq \mathcal{H}_t^{small}$ *be a hypergraph problem such that* $\sim_{\mathcal{F}}$ *has infinite index over* $\mathcal{H}_t^{small}$. *Then, there is no* MS_2-*formula that a hypergraph* $H \in \mathcal{H}_t^{small}$ *satisfies if and only if* $H \in \mathcal{F}$.

Proof Let $F \subseteq \mathcal{U}_{t,2}^{\mathrm{small}}$ be generator-total such that $\mathcal{H}(F) = \mathcal{F}$ and assume, towards a contradiction, that there is an MS_2-formula φ for hypergraphs such that $\mathcal{F} = \{H \in \mathcal{H}_t^{\mathrm{small}} \mid H \text{ satisfies } \varphi\}$. We will turn φ into an MS_1-formula φ^* for colored graphs such that a hypergraph $H \in \mathcal{H}_t^{\mathrm{small}}$ satisfies φ if and only if all graphs $G \in \mathcal{U}_{t,2}^{\mathrm{small}}$ with $\mathcal{H}(G) = H$ satisfy φ^*. That is, $F = \{G \in \mathcal{U}_{t,2}^{\mathrm{small}} \mid G \text{ satisfies } \varphi^*\}$. Courcelle's theorem [14, Theorem 6.3(2)] shows that φ^* can be turned into a tree automaton $\mathcal{A}_{\varphi^*}$ such that the parse tree of a graph $G \in \mathcal{U}_{t,2}^{\mathrm{small}}$ is accepted by $\mathcal{A}_{\varphi^*}$ if and only if G satisfies φ^*. Consequently, A_{φ^*} recognizes F. This, by Theorem 3.12, contradicts $\sim_{\mathcal{F}}$ having infinite index.

It remains to describe the transformation from φ to φ^*. To this end, recall that, by Definition 3.9 of t-boundaried hypergraph generators, the color-1 vertices in a graph $G \in \mathcal{U}_{t,2}^{\mathrm{small}}$ with $\mathcal{H}(G) = H$ represent vertices of H while the color-2 vertices in G represent hyperedges of H. Hence, the vertex and hyperedge variables in φ both become vertex variables in φ^*. Moreover, the formula φ^* makes sure that the

graph $G \in \mathcal{U}_{t,2}^{\text{small}}$ satisfying φ^* is a t-boundaried hypergraph generator, that is, vertices of the same color are nonadjacent in G and each vertex in G has a color. Thus, we let

$$\varphi^* := \text{all-labeled} \wedge \text{bipartite} \wedge \varphi',$$

$$\text{all-labeled} := \forall v[\text{col}_1(v) \vee \text{col}_2(v)],$$

$$\text{bipartite} := \forall v \forall w[(\text{col}_1(v) \wedge \text{col}_2(w)) \vee (\text{col}_2(v) \wedge \text{col}_1(w)) \vee \neg \text{adj}(v, w)],$$

where we obtain φ' by replacing terms in φ referring to hypergraphs by equivalent terms referring to incidence graphs. The term replacement translates incidence and adjacency of hypergraph objects into adjacency of the corresponding incidence graph vertices. Specifically, we replace the following hypergraph MS_2-expressions on the left-hand side by the equivalent graph MS_1-expressions on the right-hand side:

$$\text{inc}(e, v) \equiv \text{col}_2(e) \wedge \text{col}_1(v) \wedge \text{adj}(e, v), \text{ and}$$

$$\text{adj}(v, w) \equiv \text{col}_1(v) \wedge \text{col}_1(w) \wedge \exists e[\text{col}_2(e) \wedge \text{adj}(e, v) \wedge \text{adj}(e, w)].$$

Quantification over the vertices, hyperedges of a hypergraph $H = (V, E)$ are realized by the term replacements

$$\exists v \in V[\psi] \equiv \exists v[\text{col}_1(v) \wedge \psi], \qquad \exists S \subseteq V[\psi] \equiv \exists S[\forall x[x \notin S \vee \text{col}_1(x)] \wedge \psi],$$
$$\exists e \in E[\psi] \equiv \exists e[\text{col}_2(e) \wedge \psi], \qquad \exists S \subseteq E[\psi] \equiv \exists S[\forall x[x \notin S \vee \text{col}_2(x)] \wedge \psi],$$
$$\forall v \in V[\psi] \equiv \forall v[\neg \text{col}_1(v) \vee \psi], \quad \forall S \subseteq V[\psi] \equiv \forall S[\exists x[x \in S \wedge \neg \text{col}_1(x)] \vee \psi],$$
$$\forall e \in E[\psi] \equiv \forall e[\neg \text{col}_2(v) \vee \psi], \quad \forall S \subseteq E[\psi] \equiv \forall S[\exists x[x \in S \wedge \neg \text{col}_2(x)] \vee \psi].$$

$\square$

4 Hypergraph Cutwidth is Fixed-Parameter Linear

In this section, we use the Myhill–Nerode theorem for hypergraphs to show that HYPERGRAPH CUTWIDTH is fixed-parameter linear. We first formally define the problem. Let $H = (V, E)$ be a hypergraph. A *linear layout* of H is an injective map $l \colon V \to \mathbb{R}$ of vertices onto the real line. The *cut at position* $i \in \mathbb{R}$ in H with respect to l, denoted $\text{Cut}_H^l(i)$, is the set of hyperedges that contain at least two vertices v, w such that $l(v) < i < l(w)$. We will also say that v is to the *left* of i and that w is to the *right* of i. The *cutwidth* of the layout l is

$$\max_{i \in \mathbb{R}} |\text{Cut}_H^l(i)|.$$

The *cutwidth* of the hypergraph H is the minimum cutwidth over all the linear layouts of H. The hypergraph shown in Fig. 5 has cutwidth at most three. The HYPERGRAPH CUTWIDTH problem is defined as follows.

HYPERGRAPH CUTWIDTH
Input: A hypergraph $H = (V, E)$ and a natural number k.
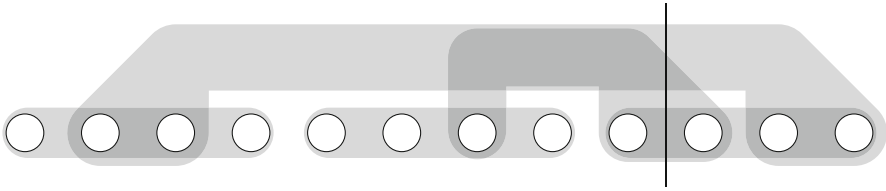*Question:* Does H have cutwidth at most k?

Fig. 5 The shown hypergraph has cutwidth at most three since the *black line* cuts a maximum number of hyperedges in the presented linear layout. Actually, it is possible to change the linear layout to see that the hypergraph has cutwidth two

To solve HYPERGRAPH CUTWIDTH using the Myhill–Nerode theorem for hypergraphs, in the remainder of this section we consider a constant k and the class k-HCW of all hypergraphs with cutwidth at most k. We will solve k-HCW in linear time using Corollary 3.13. This will immediately yield the main result of this section:

Theorem 4.1 HYPERGRAPH CUTWIDTH *is fixed-parameter linear. Specifically, there is an algorithm that, when given a hypergraph H as hyperedge list and a constant k, decides in linear time whether H has cutwidth at most k.*

In order to use Corollary 3.13 to prove Theorem 4.1, we first show that the hypergraphs in k-HCW have a constant upper bound on their incidence treewidth. Then, we show that the canonical right congruence $\sim_{k\text{-HCW}}$ has finite index. By Corollary 3.13, it then follows that k-HCW is solvable in $f(k) \cdot n$ time, completing the proof of Theorem 4.1.

Lemma 4.2 *Let H be a hypergraph. If H has cutwidth at most k, then*

(i) H has incidence treewidth at most $\max\{k, 1\}$, and
(ii) the incidence graph of H has pathwidth at most $k + 1$.

Proof Suppose that $H = (V, E)$ has cutwidth at most k. Let $H' = (V, E')$ denote the hypergraph obtained from H by removing all hyperedges of size at most one. Consider a linear layout l of cutwidth at most k of the vertices of H'. Without loss of generality, assume that l maps to the natural numbers $[n]$ and let $V = \{v_1, \ldots, v_n\}$ be such that $l(v_i) = i$. We construct a path decomposition for the incidence graph $\mathcal{I}(H')$ of H' with the bags $L_1, R_1, L_2, R_2, \ldots, L_n, R_n$ that are connected by a path in this order. For every $i \in [n]$, let $L_i := \text{Cut}^l_{H'}(i - 1/2) \cup \{v_i\}$ and $R_i := \text{Cut}^l_{H'}(i + 1/2) \cup \{v_i\}$, that is, L_i contains v_i and all hyperedges cut at $i - 1/2$, while R_i contains v_i and all hyperedges cut at $i + 1/2$. Herein, recall that the hyperedges of H' are vertices in $\mathcal{I}(H')$. We now prove that this is a path decomposition for $\mathcal{I}(H')$.

First, we show that each edge of $\mathcal{I}(H')$ is contained in at least one bag. Let $\{v_i, e\}$ be any edge in $\mathcal{I}(H')$ for some vertex $v_i \in V$ and a hyperedge $e \in E'$. We show that v_i and e occur together in at least one bag. Since $v_i \in e$ and $|e| \geq 2$, the hyperedge e contains at least one vertex to the left or to the right of v_i. Hence, we have $e \in \text{Cut}^l_{H'}(i - 1/2)$ or $e \in \text{Cut}^l_{H'}(i + 1/2)$. Therefore, it holds that $e \in R_i$ or $e \in L_i$. Since $v_i \in R_i \cap L_i$, the vertices v_i and e occur together in at least one bag.

Now, we show that the bags containing a vertex of $\mathcal{I}(H')$ induce a subpath in this path decomposition. Obviously, each vertex $v_i \in V$ is contained in two

bags of the path decomposition: in L_i and R_i. These bags are consecutive and thus induce a path. Finally, consider a hyperedge $e \in E'$. It occurs in all bags $R_i, L_{i+1}, R_{i+1}, \ldots, L_{j-1}, R_{j-1}, L_j$, where v_i is the leftmost vertex in the layout l occurring in e and v_j is the rightmost vertex in l occurring in e. These bags are all consecutive on the path and, thus, induce a path. The width of this path decomposition is $\max_{0 \leq i \leq n} |\mathrm{Cut}^l_{H'}(i + 1/2)| \leq k$.

(i) To obtain a tree decomposition for $\mathfrak{I}(H)$ from the path decomposition of $\mathfrak{I}(H')$, we only need to take care of hyperedges of size at most one. For every hyperedge $e \in E$ of size one, add a new bag $\{e, v\}$, where v is the unique vertex contained in e, and make it adjacent to an arbitrary bag containing v. For every empty hyperedge $e \in E$, add a new bag $\{e\}$, and make it adjacent to an arbitrary bag. In this way, we obtain a tree decomposition for the incidence graph $\mathfrak{I}(H)$ of H of width at most $\max\{k, 1\}$. Thus, H has incidence treewidth at most $\max\{k, 1\}$.

(ii) To obtain a path decomposition for $\mathfrak{I}(H)$ from the path decomposition of $\mathfrak{I}(H')$, one can proceed similarly: For every hyperedge $e \in E$ of size at most one, choose some bag B of size at most $k + 1$ with $e \subseteq B$ and add the bag $B \cup \{e\}$ as its neighbor to the path decomposition. Such a bag B exists since the width of the path decomposition of H' is k. The resulting path decomposition will contain bags of size $k + 2$ and, thus, has width $k + 1$.   □

To obtain a linear-time algorithm for k-HCW using Corollary 3.13 and thus proving Theorem 4.1, it remains to prove that the canonical right congruence $\sim_{k\text{-HCW}}$ of k-HCW has finite index over $\mathcal{H}_t^{\text{large}}$ for all $t \leq k + 1$.

To show that $\sim_{k\text{-HCW}}$ has finite index over $\mathcal{H}_t^{\text{large}}$, we show that, given a t-boundaried hypergraph G, only a finite number of bits of information about a t-boundaried hypergraph H is needed in order to decide whether $G \oplus_{\text{h}} H \in k$-HCW. To this end, we employ the method of test sets [17, Section 12.7]: let $\mathcal{T}$ be a set of objects called *tests* (we will formally define a test later). A t-boundaried graph can *pass* a test. For t-boundaried hypergraphs G_1 and G_2, let $G_1 \sim_{\mathcal{T}} G_2$ if and only if G_1 and G_2 pass the same subset of tests in $\mathcal{T}$. Obviously, $\sim_{\mathcal{T}}$ is an equivalence relation. Our aim is to find a set $\mathcal{T}$ of tests such that $\sim_{\mathcal{T}}$ refines $\sim_{k\text{-HCW}}$ (that is, $G_1 \sim_{\mathcal{T}} G_2$ implies $G_1 \sim_{k\text{-HCW}} G_2$). Then, if $\sim_{\mathcal{T}}$ has finite index, so does $\sim_{k\text{-HCW}}$. To show that $\sim_{\mathcal{T}}$ has finite index, we show that we can find a *finite* set $\mathcal{T}$ such that $\sim_{\mathcal{T}}$ refines $\sim_{k\text{-HCW}}$.

Intuitively, we will define, for a hypergraph H, an H-test that a hypergraph G satisfies if $G \oplus_{\text{h}} H \in k$-HCW. We define the test so that it contains only the necessary information of H and so that we can later shrink all tests to equivalent tests of constant size. We now formally define a test for k-HCW. The definition is illustrated in Fig. 6 and, after the definition, we give an intuitive description.

Definition 4.3 A *size-n test* T for k-HCW over $\mathcal{H}_t^{\text{large}}$ is a triple (π, S, k), where

– $\pi : \{1, \ldots, t\} \rightarrow \{1, \ldots, n\}$ is a map of boundary labels to integer positions, and
– $S = (S_0, S_1, \ldots, S_n)$ is a sequence of pairs $S_i = (w_i, E_i) \in \{0, \ldots, k\} \times 2^{\{1, \ldots, t\}}$ such that if $\ell \in E_p$ and $\ell \in E_q$, then $\ell \in E_i$ for all $i \in \{p, \ldots, q\}$.

Now, let G and H be t-boundaried hypergraphs such that $G \oplus_{\text{h}} H \in k$-HCW and $l : V \rightarrow \mathbb{R}$ be a linear layout for $G \oplus_{\text{h}} H$ with minimum cutwidth, which, without
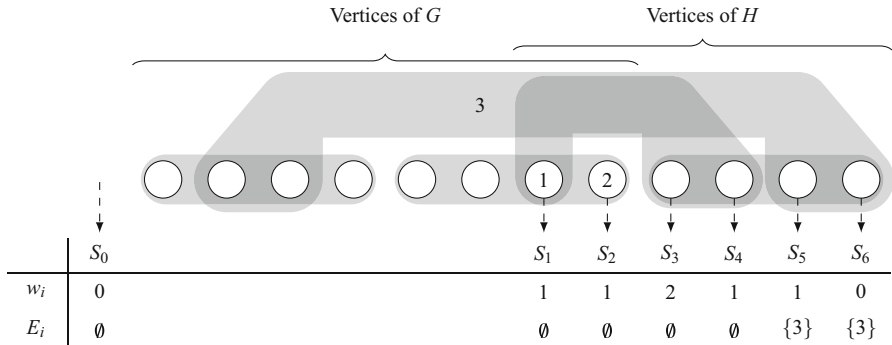
Fig. 6 Construction of the H-test illustrated using the glued hypergraph $G \oplus_h H$, where G and H are the hypergraphs shown in Fig. 4a, b, respectively. That is, G and H have only the vertices labeled 1 and 2 in common and both have a hyperedge with label 3. The vertices of H are to be understood as lying at the positions $\{1, 2, \ldots, 6\}$ and the non-boundary vertices of G lie in the open interval $(0, 1)$

loss of generality, maps vertices of the n-vertex hypergraph H to the integer positions $\{1, \ldots, n\}$ and the non-boundary vertices of G to non-integer positions.

We define an H-*test* $T = (\pi, S, k)$ for k-HCW of size n as follows: for a vertex $v \in \partial(H)$ with label ℓ, set $\pi(\ell) := l(v)$. Finally, for $i \in \{0, \ldots, n\}$, we define $S_i := (w_i, E_i)$ with

- w_i being the number of unlabeled hyperedges in H containing vertices v, w of H with $l(v) \leq i < l(w)$, and
- E_i being the set of labels of hyperedges in H containing vertices v, w of H with $l(v) \leq i \leq l(w)$.

The goal of Definition 4.3 is that if a hypergraph G passes an H-test for k-HCW, then $G \oplus_h H \in k$-HCW. More precisely, we want that if a hypergraph G passes an H-test, then $G \oplus_h H$ has a linear layout l of cutwidth at most k that lays out the vertices of H in the same way as the layout used to create the H-test. Of course, the H-test does not record the precise structure of H but only the most important information:

Assume that we want to verify that the cutwidth of the layout l of $G \oplus_h H$ is at most k without knowing H but only knowing G and the H-test. Then, for any non-integer position i, the value $w_{\lfloor i \rfloor}$ counts the unlabeled hyperedges of H cut at i. Thus, to the size of any cut for G at position $i \in \mathbb{R} \setminus \mathbb{N}$, we have to add the value $w_{\lfloor i \rfloor}$. For labeled hyperedges of H, things are more difficult: they contain vertices of $G \oplus_h H$ that originate from G as well as from H. Since an H-test corresponds to a fixed layout for H, to count a hyperedge with label ℓ of $G \oplus_h H$ that is cut at some position, it is sufficient to know the vertices of the hyperedge with label ℓ in G and the positions of the leftmost and the rightmost vertex of H contained in the hyperedge with label ℓ in H. However, in order to easier shrink all tests to constant size later, we chose a more convenient way to keep this information in the H-test: for any position i between the leftmost and the rightmost vertex of a hyperedge e in H with label ℓ, we have $\ell \in E_i$. We now precisely define what it means to pass a test.

Definition 4.4 Let $G = (V, E)$ be a t-boundaried hypergraph and $T = (\pi, S, k)$ be a test of size n, where $S = (S_0, \ldots, S_n)$ and $S_i = (w_i, E_i)$.

A T-*compatible layout* for G is an injective function $f : V \to \mathbb{R}$ such that each vertex $v \in \partial(G)$ with label ℓ is mapped to $\pi(\ell)$ and such that each vertex $v \in V \setminus \partial(G)$ is mapped into some open interval $(i, i + 1)$ for $0 \leq i \leq n$.

For a hyperedge e in G, we define the *positions of e* as

$$\mathrm{Pos}(e) := \begin{cases} \{f(v) \mid v \in e\} & \text{if } e \text{ is unlabeled,} \\ \{f(v) \mid v \in e\} \cup \{i \mid \ell \in E_i\} & \text{if } e \text{ has label } \ell. \end{cases}$$

The *joint cut at i* in G with respect to f is the set $\mathrm{Jcut}_G^f(i)$ of hyperedges e of G for which there are positions $j, k \in \mathrm{Pos}(e)$ with $j < i < k$. The *joint cutwidth of f* is

$$\max_{i \in \mathbb{R} \setminus \mathbb{N}} (|\mathrm{Jcut}_G^f(i)| + w_{\lfloor i \rfloor}).$$

Finally, G *passes the test T* if there is a T-compatible layout f for G whose joint cutwidth is at most k.

We can now show that, indeed, if two graphs satisfy the same tests, then they are equivalent under $\sim_{k\text{-HCW}}$. We will then show that, actually, there is only a finite set of pairwise nonequivalent tests, thus showing that $\sim_{k\text{-HCW}}$ has finite index.

Lemma 4.5 *For $\mathcal{T}$ being the set of all tests for k-HCW, the equivalence relation $\sim_{\mathcal{T}}$ refines $\sim_{k\text{-HCW}}$.*

To prove Lemma 4.5, we show that if two t-boundaried hypergraphs G_1, G_2 pass the same subset of tests of $\mathcal{T}$, then, for all t-boundaried hypergraphs H, $G_1 \oplus_h H \in k$-HCW if and only if $G_2 \oplus_h H \in k$-HCW. The proof is based on the following two claims.

Claim 4.6 *If $G_1 \oplus_h H \in k$-HCW, then G_1 passes some H-test.*

Claim 4.7 *If G_2 passes any H-test, then $G_2 \oplus_h H \in k$-HCW.*

From these two claims, Lemma 4.5 then easily follows: let H be a t-boundaried hypergraph such that $G_1 \oplus_h H \in k$-HCW. By Claim 4.6, G_1 passes some H-test T. Since G_1 and G_2 pass the same tests, also G_2 passes T. By Claim 4.7, it follows that $G_2 \oplus_h H \in k$-HCW. The reverse direction is proved symmetrically. It only remains to prove Claim 4.6 and Claim 4.7.

Proof (of Claim 4.6) Let T be the H-test obtained from an optimal layout l of $G_1 \oplus_h H$, which, without loss of generality, maps the vertices of the n-vertex graph H to the integer positions $\{1, \ldots, n\}$ and the vertices of $V(G_1) \setminus \partial(G_1)$ to non-integer positions in the interval $(0, n + 1)$. Then, l obviously is a T-compatible layout for G_1. We show that the joint cutwidth $\max_{i \in \mathbb{R} \setminus \mathbb{N}} (|\mathrm{Jcut}_{G_1}^l(i)| + w_{\lfloor i \rfloor})$ of l from Definition 4.4 is at most k.

To this end, for an $i \in \mathbb{R} \setminus \mathbb{N}$, consider the set $A := \mathrm{Cut}^l_{G_1 \oplus_h H}(i)$ of hyperedges of $G_1 \oplus_h H$ containing two vertices v, w with $l(v) < i < l(w)$. Since $G_1 \oplus_h H \in k$-HCW, we have $|A| \leq k$. Thus, it is sufficient to show that $|\mathrm{Jcut}^l_{G_1}(i)| + w_{\lfloor i \rfloor} \leq |A|$. We partition A into two sets B and C, where B are the unlabeled hyperedges in H.

Since $i \notin \mathbb{N}$, by Definition 4.3, $w_{\lfloor i \rfloor}$ counts exactly the hyperedges in B. It remains to show that $|\mathrm{Jcut}^l_{G_1}(i)| \leq |C|$. Recall from Definition 4.4 that $\mathrm{Jcut}^l_{G_1}(i)$ is the set of hyperedges e of G_1 for which $\mathrm{Pos}(e)$ contains two positions j, k with $j < i < k$. If e is unlabeled, then, by Definition 4.4 of $\mathrm{Pos}(e)$, the hypergraph G_1 contains vertices v, w with $l(v) = j$ and $l(w) = k$. Since e, v, and w are also in $G_1 \oplus H$, we have $e \in C$.

If e is labeled, then $G_1 \oplus_h H$ instead of e contains a hyperedge $e' \supseteq e$. Now, since $j \in \mathrm{Pos}(e)$, the hypergraph G_1 contains a vertex v with $l(v) = j$ or $\ell \in E_j$, which, by Definition 4.3, implies that there is a hyperedge with label ℓ containing a vertex v in H with $l(v) \leq i$. Likewise, G_1 contains a vertex w with $l(w) = k$ or $\ell \in E_k$, which implies that there is a hyperedge with label ℓ containing a vertex w in H with $k \leq l(w)$. In all cases, we have that $l(v) < i < l(w)$. Since the hyperedge e' of $G_1 \oplus_h H$ contains v and w, we get $e' \in C$. $\square$

Proof (of Claim 4.7) Let T be an H-test T obtained from a linear layout l of cutwidth k for some $G^* \oplus_h H$ and assume that G_2 passes T. Then, there is a T-compatible layout f for G_2 with joint cutwidth at most k. First note that l and f agree on the layout of vertices in $\partial(G_2)$ and $\partial(H)$ and that, apart from these, f lays out vertices at non-integral positions, whereas l lays out vertices of H at integral positions. Because of this, in a layout g for $G_2 \oplus_h H$ that lays out vertices v of H at position $l(v)$ and vertices v of G_2 at position $f(v)$, every two vertices in $G_2 \oplus_h H$ are laid out at distinct positions by g. Hence, g is injective and, therefore, a layout.

We show that g is a layout of cutwidth at most k for $G_2 \oplus_h H$. That is, we show $\max_{i \in \mathbb{R}} |\mathrm{Cut}^g_{G_2 \oplus_h H}(i)| \leq k$. To this end, note that

$$\max_{i \in \mathbb{R}} |\mathrm{Cut}^g_{G_2 \oplus_h H}(i)| \leq \max_{i \in \mathbb{R} \setminus \mathbb{N}} |\mathrm{Cut}^g_{G_2 \oplus_h H}(i)|,$$

since for every $i \in \mathbb{N}$, we have $\mathrm{Cut}^g_{G_2 \oplus_h H}(i) \subseteq \mathrm{Cut}^g_{G_2 \oplus_h H}(i + \varepsilon)$ for $0 < \varepsilon < 1$ chosen so that no vertex is mapped by g to the interval $(i, i + \varepsilon]$. That is, we only have to show that, for each $i \in \mathbb{R} \setminus \mathbb{N}$, we have $|\mathrm{Cut}^g_{G_2 \oplus H}(i)| \leq |\mathrm{Jcut}^f_{G_2}(i)| + w_{\lfloor i \rfloor}$, since f is a layout for G_2 with joint cutwidth $\max_{i \in \mathbb{R} \setminus \mathbb{N}}(|\mathrm{Jcut}^f_{G_2}(i)| + w_{\lfloor i \rfloor}) \leq k$.

For some position $i \in \mathbb{R} \setminus \mathbb{N}$, consider the set $A := \mathrm{Cut}^g_{G_2 \oplus_h H}(i)$ of hyperedges of $G_2 \oplus_h H$ containing vertices v, w with $g(v) < i < g(w)$ and let it be partitioned into two sets B and C, where B contains the unlabeled hyperedges of H. We show that $|A| \leq w_{\lfloor i \rfloor} + |\mathrm{Jcut}^f_{G_2}(i)|$. By Definition 4.3, we clearly have $|B| \leq w_{\lfloor i \rfloor}$. Hence, it remains to show that $|C| \leq |\mathrm{Jcut}^f_{G_2}(i)|$.

To this end, let $e \in C$ be a hyperedge. If e contains only vertices of G_2, then for any vertex $v \in e$, we have $g(v) = f(v) \in \mathrm{Pos}(e)$. Furthermore, if e contains a vertex v of H, then e has a label ℓ and H has a hyperedge with label ℓ containing v. Hence, in this case, we have $\ell \in E_{l(v)}$. It follows that $g(v) = l(v) \in \mathrm{Pos}(e')$ for the hyperedge $e' \subseteq e$ of G_2 with label ℓ. Hence, for any vertex $v \in e$, we have $g(v) \in \mathrm{Pos}(e')$. Since

e contains vertices v, w with $g(v) < i < g(w)$, it follows that $\mathrm{Pos}(e')$ contains positions $j = g(v)$ and $k = g(w)$ with $j < i < k$ and, therefore, $e' \in \mathrm{Jcut}^f_{G_2}(i)$. $\square$

Towards our goal of showing that $\sim_{k\text{-HCW}}$ has finite index, Lemma 4.5 shows a set of tests $\mathcal{T}$ such that $\sim_{\mathcal{T}}$ refines $\sim_{k\text{-HCW}}$, where two hypergraphs are equivalent with respect to $\sim_{\mathcal{T}}$ if and only if they pass the same subset of tests of $\mathcal{T}$. However, since the set $\mathcal{T}$ is infinite, we cannot yet conclude that $\sim_{\mathcal{T}}$ and, therefore, $\sim_{k\text{-HCW}}$ has finite index. The following lemma will, for every test $T \in \mathcal{T}$, find a test $T' \in \mathcal{T}$ such that a hypergraph G passes T' if and only if it passes T and such that T' has size at most $(2t + 1)(t + 1)(2k + 2)$. Thus, the equivalence relation $\sim_{\mathcal{T}'}$ for $\mathcal{T}'$ being the set of all tests of size $(2t + 1)(t + 1)(2k + 2)$ is the same as $\sim_{\mathcal{T}}$ and, consequently, refines $\sim_{k\text{-HCW}}$. Since there is only a constant number of tests of size $(2t + 1)(t + 1)(2k + 2)$ for constant k and $t \leq k + 1$, the size of $\mathcal{T}'$ is constant. Since $\sim_{\mathcal{T}'}$ and, therefore, $\sim_{k\text{-HCW}}$ has at most $2^{|\mathcal{T}'|}$ equivalence classes, it follows that $\sim_{k\text{-HCW}}$ has finite index. Thus, the following lemma finishes our proof of Theorem 4.1.

Lemma 4.8 *Let G be a t-boundaried hypergraph. For every test T_1, there is a test T_2 of size $(2t + 1)(t + 1)(2k + 2)$ such that G passes T_1 if and only if G passes T_2.*

Proof Let the size of the test $T_1 = (\pi, S, k)$ be n. For $E \subseteq \{1, \ldots, t\}$, we call a maximal subsequence $S_j = (E_j, w_j), \ldots, S_k = (E_k, w_k)$ of S with $E = E_j = \cdots = E_k$ a *strait*. We first show that there are at most $2t + 1$ straits, and then show that we can shorten each strait to length at most $(t + 1)(2k + 2)$ by removing some elements from S without changing the satisfiability of the test.

For a label $\ell \subseteq \{1, \ldots, t\}$, let $I_\ell := \{i \leq n \mid \ell \in E_i\}$. By Definition 4.3, each I_ℓ for some label $\ell \in \{1, \ldots, t\}$ is an interval of the natural numbers with a minimum element and a maximum element, which we both call *events*. Hence, the I_ℓ for all $\ell \in \{1, \ldots, t\}$ in total have at most $2t$ events. Since straits can only start at an event or at S_0, and since only one strait can start at a fixed event, it follows that S is partitioned into at most $2t + 1$ straits.

It remains to shorten the straits. To this end, we apply data reduction rules already used by Downey and Fellows [17, Theorem 12.7.5] for the cutwidth problem on graphs. Let $S_j = (E, w_j), \ldots, S_k = (E, w_k)$ be a strait in T_1. We call a maximal subsequence of the w_i of the strait such that π maps no boundary label to i a *load pattern*. Hence, each strait decomposes into at most $t + 1$ load patterns, each of which we will shorten to length at most $2k + 2$.

To this end, first observe that if the test T_1 passed by G contains a pair $S_i = (E, w_i)$, then G also passes the test obtained from T_1 by replacing S_i by $S'_i = (E, w'_i)$ with $w'_i \leq w_i$. Moreover, assume that, as illustrated in Fig. 7, T_1 contains two pairs $S_i = (E, w_i)$, $S_{i+1} = (E, w_{i+1})$ with $w_i = w_{i+1}$ such that π maps no boundary label to $i + 1$. Then G passes the test obtained from T_1 by removing S_{i+1}. Moreover, G then also passes the test obtained from T_1 by adding a copy of S_i behind S_i.

Based on these observations, Downey and Fellows [17, Theorem 12.7.5] give a proof that the following three data reduction rules applied to a load pattern s of the strait $S_j, \ldots, S_k$ turn T_1 into a test T_2 that G passes if and only if it passes T_1:
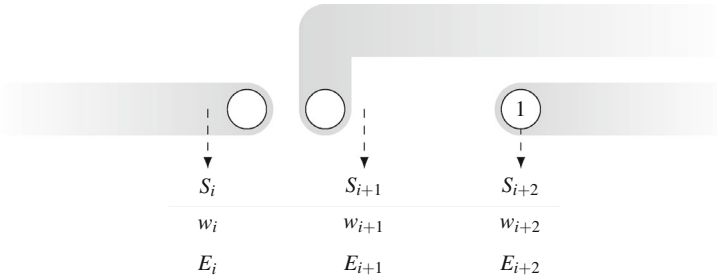
Fig. 7 Shown are two unlabeled vertices and one labeled vertex of a graph G laid out according to a T-compatible layout for some test $T = (\pi, S, k)$. That is, the label 1 is mapped to the integer position $i + 2$ by π, while the others vertices are laid out at non-integer positions. Assume that $S_i = S_{i+1}$ and that no label is mapped to position $i + 1$ by π. Then, we can assume that no vertex of G lies in $[i + 1, i + 2]$: moving it to $(i, i + 1)$ would yield a T_1-compatible layout with equal joint cutwidth. The joint cutwidth will also not be altered by deleting S_{i+1} or adding copies of S_i behind S_i

(R1) If $s = (\ldots, w_i, w_{i+1}, w_{i+2}, \ldots)$ such that $w_i \leq w_{i+1} \leq w_{i+2}$ or $w_i \geq w_{i+1} \geq w_{i+2}$, then delete S_{i+1}.

(R2) If $s = (\ldots, a, s_i, \ldots, s_{i'}, b, \ldots)$ such that each of $s_i, \ldots, s_{i'}$ is at least $\max(a, b)$, then replace $S_i, \ldots, S_{i'}$ by $S^* := (E, w)$, where w is the maximum of $s_i, \ldots, s_{i'}$.

(R3) If $s = (\ldots, a, s_i, \ldots, s_{i'}, b, \ldots)$ such that each of $s_i, \ldots, s_{i'}$ is at most $\min(a, b)$, then replace $S_i, \ldots, S_{i'}$ by $S^* := (E, w)$, where w is the minimum of $s_i, \ldots, s_{i'}$.

Downey and Fellows [17, Theorem 12.7.5] show that a load pattern, to which none of the rules apply, has length at most $2k + 2$. □

Historical Remarks. The above results about reduced load patterns in the construction of test sets were first proved by Abrahamson and Fellows [1,2] in the context of proving that, for simple graphs, the property $\mathcal{P}_k$ of having cutwidth bounded by k has finite index over $\mathcal{U}_t^{\text{large}}$ for all fixed k and t. An essentially equivalent notion, termed *typical sequences*, was introduced independently by Bodlaender and Kloks [7] in the context of linear-time dynamic programming algorithms for PATHWIDTH and TREEWIDTH. Such sequences are also implicit in early work of Lagergren and Arnborg [33].

5 Hypertree Width and Variants

In in this section, we show a negative application of our hypergraph Myhill–Nerode analog to GENERALIZED HYPERTREE WIDTH [29]. First, we precisely define the problem.

Let H be a hypergraph. Generalized hypertree width is defined with respect to tree decompositions of the primal graph $\mathcal{G}(H)$, however, the width of the tree decompositions is measured differently. Suppose H has no isolated vertices (otherwise, remove them). A *cover* of a bag is a set of hyperedges such that each vertex in the bag is contained in at least one of these hyperedges. The *cover width* of a bag is the minimum possible number of hyperedges covering it. The *cover width* of a tree decomposition is

the maximum cover width of any bag in the decomposition. The *generalized hypertree width* of H is the minimum cover width over all tree decompositions of $\mathcal{G}(H)$.

GENERALIZED HYPERTREE WIDTH
Input: A hypergraph $G = (V, E)$ and a natural number k.
Question: Does G have generalized hypertree width at most k?

Since GENERALIZED HYPERTREE WIDTH is NP-hard for $k = 3$ [29], it is natural to search for non-standard parameters with respect to which the problem is fixed-parameter tractable [21,32,42]. While it is known that the generalized hypertree width of a hypergraph is at most the incidence treewidth plus one [23], the incidence treewidth may be arbitrarily large even for hypergraphs with hypertree width one, since adding a universal hyperedge to any hypergraph reduces its hypertree width to one. Therefore, one could hope for positive results with respect to incidence treewidth. However, we will show that GENERALIZED HYPERTREE WIDTH cannot be solved by finite tree automata on tree decompositions of incidence graphs:

Theorem 5.1 *Let k-GHTW be the set of hypergraphs with generalized hypertree width at most k. The canonical right congruence $\sim_{k\text{-GHTW}}$ does not have finite index over $\mathcal{H}_t^{small}$ for $k = 4$ and $t \geq 41$.*

By Corollary 3.16, it follows that k-GHTW is not expressible in monadic second-order logic. Moreover, the construction we use in the proof leads us to conjecture that, actually, the problem might turn out to be W[1]-hard, as did BANDWIDTH and TRIANGULATING COLORED GRAPHS after it was shown that they do not have finite index [9].

We will discuss this after proving Theorem 5.1. Moreover, after proving Theorem 5.1, we will discuss that the theorem also holds for the problem variants HYPERTREE WIDTH and FRACTIONAL HYPERTREE WIDTH.

To prove Theorem 5.1, we apply Corollary 3.14: for every $n \geq 1$, we give a construction of a t-boundaried hypergraph H_n whose incidence graph allows for a tree decomposition of width $t - 1$, of which one bag contains $\partial(H_n)$. Then we show that $H_n \oplus_h H_m$ has generalized hypertree width 4 if and only if $n = m$. This implies that the canonical right congruence $\sim_{4\text{-HTW}}$ has infinite index over $\mathcal{H}_t^{large}$ and, by Corollary 3.14, it follows that it has infinite index also over $\mathcal{H}_t^{small}$.

Construction 5.2 For every $n \geq 1$, we construct a t-boundaried hypergraph H_n with $t = 28$, generalized hypertree width 4, and incidence treewidth at most 12. The vertex set of H_n is $V := A \cup B \cup C \cup D \cup S \cup T \cup X$, where $A := \{a, y\}$, $B := \{b, z\}$, $C := \{c, y\}$, $D := \{d, z\}$, $S := \{s_1, \ldots, s_8\}$, $T := \{t_1, \ldots, t_8\}$ and $X := \{x_1, \ldots, x_{6n}\}$. The hyperedge set of H_n is $E := \{A, B, C, D\} \cup B_S \cup \{S_c, S_d, S_y, S_z\} \cup B_T \cup \{T_a, T_b, T_y, T_z\} \cup \{E_{3i}, E_{3i+1} : 1 \leq i < 2n\} \cup \{E_{i,i+1} : 1 \leq i < 6n\}$, where

B_S is the set of all possible binary hyperedges on S,
$$S_c := \{c, s_1, s_2\}, \qquad\qquad S_d := \{d, s_3, s_4\},$$
$$S_y := \{y, s_5, s_6\}, \qquad\qquad S_z := \{z, s_7, s_8\},$$

B_T is the set of all possible binary hyperedges on T,

$$T_a := \{a, t_1, t_2\}, \qquad\qquad T_b := \{b, t_3, t_4\},$$
$$T_y := \{y, t_5, t_6\}, \qquad\qquad T_z := \{z, t_7, t_8\},$$
$$E_1 := \{s_8, x_1\},$$
$$E_{3i} := \{a, c, y, x_{3i}\} \qquad\qquad \text{for } 1 \leq i < 2n,$$
$$E_{3i+1} := \{b, d, z, x_{3i+1}\} \qquad\qquad \text{for } 1 \leq i < 2n,$$
$$E_{6n} := \{x_{6n}, t_1\},$$
$$E_{6i+1,6i+2} := \{a, b, x_{6i+1}, x_{6i+2}\} \qquad\qquad \text{for } 0 \leq i < n,$$
$$E_{6i+4,6i+5} := \{c, d, x_{6i+4}, x_{6i+5}\} \qquad\qquad \text{for } 0 \leq i < n, \text{ and}$$
$$E_{3i,3i+1} := \{x_{3i}, x_{3i+1}\} \qquad\qquad \text{for } 1 \leq i < 2n.$$

The set of boundary hyperedges is $\{A, B, C, D, S_c, S_d, S_y, S_z, T_a, T_b, T_y, T_z\}$. The set of boundary vertices is $S \cup T$. They are labeled from 1 to 28 in this order and by increasing indices. See Fig. 8 for an illustration of H_2 induced on $V \setminus (S \cup T)$.

We first give an outline of the remaining proof. Consider a tree decomposition for $H_n \oplus_h H_m$ with generalized hypertree width 4. The aim is to prove $n = m$. The vertex sets S and T and the hyperedges containing them make sure that some bag $\mathcal{B}_S$ of the decomposition contains all of S and that some bag $\mathcal{B}_T$ contains all of T. Now, both in $\mathcal{G}(H_n)$ and in $\mathcal{G}(H_m)$, there is a path from a vertex in S to a vertex in T passing through all vertices x_i by increasing indices. The edges of this path are covered

Fig. 8 The incidence graph of H_2 induced on $V \setminus (S \cup T)$. *Boxes* represent hyperedges
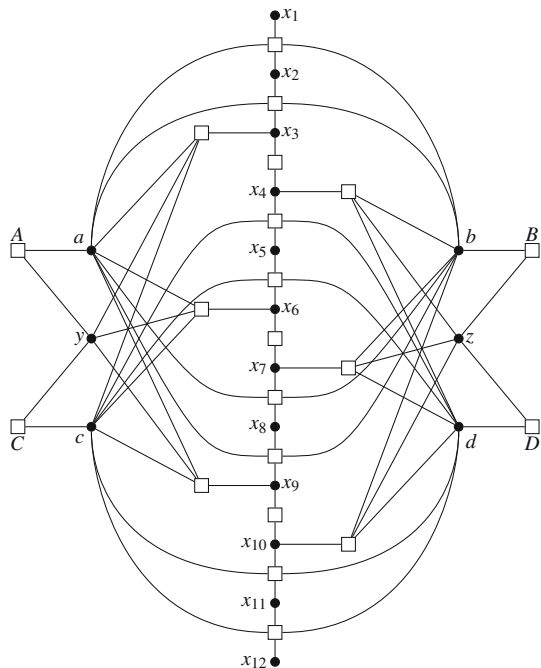
by intermediate bags lying on the path from $\mathcal{B}_S$ to $\mathcal{B}_T$ on the tree decomposition. Observe that no vertex x_i is contained in a boundary hyperedge. Therefore, when we restrict the tree decomposition to the vertices in H_n, we recover a tree decomposition for H_n where all intermediate bags are covered by at most 3 hyperedges. Moreover, our construction makes sure that when a bag is covered by 3 hyperedges, at least 2 of them are boundary hyperedges. In every such tree decomposition for H_n, when considering the intermediate bags starting from $\mathcal{B}_S$ that contain either A, B or C, D in their cover, we first encounter bags covered by C, D, then bags covered by A, B, then bags covered by C, D, and so on, and there are exactly n alternations from C, D to A, B in this sequence. Therefore, in order to be able to merge such decompositions for H_n and H_m, we must have $n = m$.

We now give a more detailed proof of Theorem 5.1. In the construction of H_n, the vertices in S and T and the hyperedges containing them are only used to make sure that every tree decomposition of H_n with hypertree width 4 contains a bag $\mathcal{B}_{-1}$ with the vertices $S \cup \{c, d, y\}$ and a bag $\mathcal{B}_{6n+1}$ with the vertices $T \cup \{b, y, z\}$. Since the sets $S \cup \{c, d, y, z\}$ and $T \cup \{a, b, y, z\}$ can also be covered by 4 hyperedges, all of which are boundary hyperedges, let $\mathcal{D} = (\{V_i : i \in I\}, T)$ be a tree decomposition for H_n with the bags $\mathcal{B}_{-1} = S \cup \{c, d, y, z\}$ and $\mathcal{B}_{6n+1} = T \cup \{a, b, y, z\}$. We observe that all other vertices of H_n occur in bags that are in the same connected component of the forest obtained from $\mathcal{D}$ by removing these two bags.

Claim 5.3 The tree decomposition $\mathcal{D}$ contains a bag $\mathcal{B}_i$, $0 \le i \le 6n$, with $\{s_8, x_1, c, d, y, z\} \subseteq \mathcal{B}_0$, $\{t_1, x_{6n}, a, b, y, z\} \subseteq \mathcal{B}_{6n}$, and $\{a, b, c, d, y, z, x_i, x_{i+1}\} \subseteq \mathcal{B}_i$, for every i, $1 \le i < 6n$.

Proof The primal graph $\mathcal{G}(H_n)$ contains the cliques $\{s_8, x_1\}$, $\{a, b, x_1, x_2\}$, $\{a, b, x_2, x_3\}$, $\{a, c, y, x_3\}$, $\{x_3, x_4\}$, $\{x_4, b, d, z\}$, $\{c, d, x_4, x_5\}$, $\{c, d, x_5, x_6\}$, $\{a, c, y, x_6\}$, $\{x_6, x_7\}$, $\{b, d, z, x_7\}$, $\{a, b, x_7, x_8\}$, ..., $\{x_{6n}, t_1\}$, and every two consecutive cliques in this list intersect in at least one vertex. In particular, we observe the path $(s_8, x_1, x_2, ..., x_{6n}, t_1)$ in $\mathcal{G}(H_n)$. Thus, $\mathcal{D}$ contains bags $\mathcal{B}_0 \supseteq \{s_8, x_1\}$, $\mathcal{B}_{6n} \supseteq \{t_1, x_{6n}\}$, and $\mathcal{B}_i \supseteq \{x_i, x_{i+1}\}$, $1 \le i < 6n$. Moreover, each $\mathcal{B}_i$, $0 \le i < 6n$, contains c, d, y, z since $\mathcal{B}_{-1}$ contains c, d, y, z, $\mathcal{B}_{6n-1}$ contains c, d, $\mathcal{B}_{6n+1}$ contains y, z, and without loss of generality, we can assume the $\mathcal{B}_i$, $0 \le i \le 6n$, were chosen such that they are on the path from $\mathcal{B}_{-1}$ to $\mathcal{B}_{6n+1}$ in T. Similarly, each $\mathcal{B}_i$, $1 \le i \le 6n$, contains a, b. □

A tree decomposition for H_n is a *good* tree decomposition if it contains the bags $\mathcal{B}_{-1} = S \cup \{c, d, y, z\}$ and $\mathcal{B}_{6n+1} = T \cup \{a, b, y, z\}$ and every bag except $\mathcal{B}_{-1}$ and $\mathcal{B}_{6n+1}$ can be covered with at most 3 hyperedges, and in case such a bag is covered with exactly 3 hyperedges, two of these hyperedges are in the boundary. A *good* cover for a good tree decomposition is a cover for each bag according to the specifications of a good tree decomposition.

Claim 5.4 If $\mathcal{D}$ is a good tree decomposition for H_n, then, for every i, $-1 \le i \le 6n$, there is a path from the bag $\mathcal{B}_i$ to the bag $\mathcal{B}_{i+1}$ that avoids all the bags $\mathcal{B}_j$, $j \in \{-1, ..., 6n + 1\} \setminus \{i, i + 1\}$.

Proof Suppose the path from $\mathcal{B}_i$ to $\mathcal{B}_{i+1}$ passes through $\mathcal{B}_j$ with $j \in \{-1, ..., 6n + 1\} \setminus \{i, i + 1\}$. Since every bag on the path from $\mathcal{B}_i$ to $\mathcal{B}_{i+1}$ contains $\mathcal{B}_i \cap$

$\mathcal{B}_{i+1}$, we have that $x_{i+1} \in \mathcal{B}_j$. But then $\{s_8, x_1, c, d, y, z, x_{i+1}\} \subseteq \mathcal{B}_j$ (if $j = 0$) or $\{t_1, x_{6n}, a, b, y, z, x_{i+1}\} \subseteq \mathcal{B}_j$ (if $j = 6n$) or $\{a, b, c, d, y, z, x_j, x_{j+1}, x_{i+1}\} \subseteq \mathcal{B}_j$ (otherwise), implying that $\mathcal{B}_j$ cannot be covered by two hyperedges and it cannot be covered by three hyperedges of which two are in the boundary. □

Claim 5.5 In every good cover, $\mathcal{B}_0$ is covered by $\{E_1, C, D\}$, $\mathcal{B}_{6n}$ is covered by $\{E_{6n}, A, B\}$, and for every i, $1 \leq i < 6n$,

$$\mathcal{B}_i \text{ is covered by } \begin{cases} \{E_{i,i+1}, C, D\} & \text{if } i \equiv 1 \pmod 6, \\ \{E_{i,i+1}, C, D\} & \text{if } i \equiv 2 \pmod 6, \\ \{E_i, E_{i+1}\} & \text{if } i \equiv 3 \pmod 6, \\ \{E_{i,i+1}, A, B\} & \text{if } i \equiv 4 \pmod 6, \\ \{E_{i,i+1}, A, B\} & \text{if } i \equiv 5 \pmod 6, \text{ and} \\ \{E_i, E_{i+1}\} & \text{if } i \equiv 0 \pmod 6. \end{cases}$$

Proof The claim easily follows from Claim 5.3. □

Suppose $\mathcal{D}$ is a good tree decomposition for H_n. The *backbone* of $\mathcal{D}$ is the path P in T starting at the bag $\mathcal{B}_{-1}$ and ending at the bag $\mathcal{B}_{6n+1}$. By Claim 5.4, P visits $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_{6n}$ in this order. Let $P_{i,j}$ denote the subpath of P starting at $\mathcal{B}_i$ and ending at $\mathcal{B}_j$.

Claim 5.6 For every $i \in \{0, 6, 12, \ldots, 6n - 6\}$, no bag on $P_{i,i+3}$ is covered by a set of hyperedges $\mathcal{Q}$ with $A, B \in \mathcal{Q}$ in a good cover.

Proof Consider a bag $\mathcal{B}$ on $P_{i,i+3}$ and let $\mathcal{Q} \supseteq \{A, B\}$ be a cover for $\mathcal{B}$. The bag $\mathcal{B}$ contains the intersection of two bags that are consecutive in the list $\mathcal{B}_i, \mathcal{B}_{i+1}, \mathcal{B}_{i+2}, \mathcal{B}_{i+3}$. Therefore, at least one of $x_{i+1}, x_{i+2}, x_{i+3}$ is in $\mathcal{B}$. We also have that $c, d \in \mathcal{B}$ since $c, d \in \mathcal{B}_i \cap \mathcal{B}_{i+3}$. However, no hyperedge contains x_{i+1}, c, d or x_{i+2}, c, d or x_{i+3}, c, d. Thus, $|\mathcal{Q}| \geq 4$, and therefore $\mathcal{Q}$ is not part of a good cover. □

Claim 5.7 For every $i \in \{3, 9, 15, \ldots, 6n - 3\}$, no bag on $P_{i,i+3}$ is covered by a set of hyperedges $\mathcal{Q}$ with $C, D \in \mathcal{Q}$ in a good cover.

Proof The proof is symmetric to the proof of Claim 5.6. □

Consider a good cover of $\mathcal{D}$. A *switch* is an inclusion-wise minimal subpath $(Y_i, \ldots, Y_j)$ of the backbone of $\mathcal{D}$ where Y_i is covered by $\mathcal{Q}_i$ with $C, D \in \mathcal{Q}_i$ and Y_j is covered by $\mathcal{Q}_j$ with $A, B \in \mathcal{Q}_j$. The *signature* of a good cover of $\mathcal{D}$ is its number of switches.

Claim 5.8 Each good cover of each good tree decomposition of H_n has signature n.

Proof The claim follows from Claims 5.5, 5.6, and 5.7. □

Due to Claim 5.8, we can speak of the signature of H_n and the signature of a good tree decomposition of H_n as the signature of some good cover of such a tree decomposition.

Let $H = H_n$ and $H' = H_m$. Consider a tree decomposition $\mathcal{D} = (\{V_i : i \in I\}, T)$ of $H \oplus_h H'$ with generalized hypertree width 4. Without loss of generality, suppose the bags $\mathcal{B}_{-1} = S \cup \{c, d, y, z\}$ and $\mathcal{B}_{6n+1} = T \cup \{a, b, y, z\}$ are leafs of this decomposition and their neighboring bags contain both copies of x_1 and x_{6n}, respectively. Let $\mathcal{D}_{|H}$ denote the restriction of $\mathcal{D}$ to H, i.e., it has the same tree, but each bag is restricted to the vertices of H.

Claim 5.9 $\mathcal{D}_{|H}$ is a good tree decomposition for H.

Proof Consider a bag $\mathcal{B}$ of $\mathcal{D}$ besides $\mathcal{B}_{-1}$ and $\mathcal{B}_{6n+1}$. The bag $\mathcal{B}$ contains a copy of some x_i from H'. This vertex is covered by some hyperedge from H' that does not belong to the boundary. Therefore, $\mathcal{B}_{|H}$ is covered by at most 3 hyperedges. Suppose $\mathcal{B}_{|H}$ is covered by exactly 3 hyperedges. Then, the cover of $\mathcal{B}_{|H'}$ contains at most one hyperedge that does not belong to the boundary. But, since each such hyperedge covers at most 2 vertices among $\{a, b, c, d\}$, the cover of $\mathcal{B}$ contains at least 2 boundary hyperedges. This proves the claim. □

Symmetrically, $\mathcal{D}_{|H'}$ is a good tree decomposition for H'. Since $\mathcal{D}_{|H}$ and $\mathcal{D}_{|H'}$ have the same signature, we conclude that $n = m$ due to Claim 5.8. This proves that the canonical right congruence $\sim_{4\text{-GHTW}}$ does not have finite index over $\mathcal{H}_t^{\text{large}}$. To prove Theorem 5.1, it remains to prove that it also has infinite index over $\mathcal{H}_t^{\text{small}}$.

(Proof of Theorem 5.1) We aim to apply Corollary 3.14. First, we show that the constructed graphs H_n have incidence treewidth at most 12. The graph $H_n \setminus (S \cup T \cup \{a, b, c, d, y, z\})$ is a disjoint union of trees and therefore, has tree decomposition of width 1. From this tree decomposition, we obtain a tree decomposition of width 7 for $H_n \setminus (S \cup T)$ by adding $\{a, b, c, d, y, z\}$ to each bag of the decomposition. Finally, we obtain a tree decomposition for H_n of width at most 12 by adding the two bags $S \cup \{a, b, c, d, y, z\}$ and $T \cup \{a, b, c, d, y, z\}$ and making them adjacent to arbitrary bags of the tree decomposition for $H_n \setminus (S \cup T)$. To obtain a tree decomposition where one bag contains $\partial(H_n)$, we modify the tree decomposition of width 12 for the incidence graph H_n by adding the 28 boundary objects to each bag. The result is a tree decomposition of width 40 where $\partial(H_n)$ is contained in one bag. Then, we obtain a hypergraph H_n' from H_n by adding to H_n 13 additional, isolated boundary vertices. Clearly, H_n and H_n' have the same generalized hypertree width. We use the tree decomposition of the incidence graph of H_n also for H_n', but we attach a bag consisting of $\partial(H_n)$ and the 13 additional boundary vertices of H_n'. This bag has width $28 + 12 = 40$. Hence, the family H_n' we constructed from H_n has tree decompositions of width 40 of their incidence graphs such that one bag contains all 41 boundary objects. Thus, Corollary 3.14 applies to our family. □

Other width measures for hypergraphs. Theorem 5.1 easily applies also to the problems HYPERTREE WIDTH and FRACTIONAL HYPERTREE WIDTH, which asks whether a hypergraph has (fractional) hypertree width at most k. HYPERTREE WIDTH is W[2]-hard [28] with respect to k and FRACTIONAL HYPERTREE WIDTH is expected to be NP-hard for constant k [36]. Before discussing how Theorem 5.1 applies to these problems, we formally define these width measures.

The *hypertree width* of H is defined in a similar way as the generalized hypertree width, except that, additionally, the tree of the decomposition is rooted and a hyperedge e can only be used in the cover of a bag V_i if V_i contains all vertices of e that occur in bags of the subtree rooted at the node i.

The *fractional hypertree width* of H is also defined similarly, except that it uses fractional covers: in a *fractional cover* of a bag, each hyperedge is assigned a non-negative weight, and for each vertex in the bag, the sum of the weights of the hyperedges incident to it is at least 1. The *fractional cover width* of the bag is the minimum total sum of all hyperedges of a fractional cover.

Let k-HTW be the family of hypergraphs of hypertree width at most k and k-FHTW be the family of hypergraphs of fractional hypertree width at most k. To see that the proof of Theorem 5.1 applies to $\sim_{4\text{-HTW}}$, observe that, in our construction, every hyperedge covering a bag is a subset of that bag. To see that it extends to $\sim_{4\text{-FHTW}}$, observe that for every bag $\mathcal{B}_i$, $0 \leq i \leq 6n$, an optimal fractional cover is integral, and 5.6 can be extended to $A, B \in \mathcal{Q}$ with weight 1—similarly for Claim 5.7.

Corollary 5.10 FRACTIONAL HYPERTREE WIDTH *and* HYPERTREE WIDTH *do not have finite index.*

Indication for intractability. Formally, Theorem 5.1 only shows that a tree automaton cannot decide the property of having constant HYPERTREE WIDTH and, by Corollary 3.16, that this property is not expressible in monadic second-order logic for hypergraphs. However, following the argumentation of Bodlaender, Fellows, and Warnow [8] for the TRIANGULATING COLORED GRAPHS problem introduced in Sect. 1 leads us to the following conjecture.

Conjecture 5.11 GENERALIZED HYPERTREE WIDTH is W[1]-hard with respect to the parameter incidence treewidth.

The reason for this conjecture lies in the number of equivalence classes observed in the proof of Theorem 5.1, which entails a lower bound on the amount of information that needs to be maintained by an algorithm when it decides whether a given hypergraph has (generalized, fractional) hypertree width k using a tree decomposition of the incidence graph. Typical such algorithms associate with each bag of the tree decomposition a table that is computed from the tables associated with the tables of the child bags. Observe that such an algorithm is essentially a tree automaton; its states are the tables. Since the number of equivalence classes of the canonical right congruence gives a lower bound on the number of states a tree automaton needs to have in order to decide GENERALIZED HYPERTREE WIDTH, it also gives a lower bound on the number different tables that have to be handled by such a (simple) dynamic programming algorithm in order not to make wrong decisions.

However, restricting the construction in the proof of Theorem 5.1 to graphs of at most n vertices, the proof of Theorem 5.1 exhibits a class $\mathcal{C}$ of t-boundaried hypergraphs on at most n vertices with constant incidence treewidth $t-1$, for which the canonical right congruence has $\Omega(n)$ equivalence classes. Now, consider a class $\mathcal{C}'$ of $O(k)$-boundaried hypergraphs where each hypergraph contains k copies of hypergraphs from $\mathcal{C}$ and has at most n' vertices. Then, the number of equivalence classes

of the canonical right congruence is $\Omega((n'/k)^k)$ for $\mathcal{C}'$. Hence, we conjecture that an algorithm with running time $f(k) \cdot n^c$ for a constant c and a computable function f does not exist.

6 Summary, Discussion and Open Problems

We extended the Myhill–Nerode theorem to hypergraphs, making the methodology more widely applicable. We did this in the general framework of the hypergraph analogs of the *large* and *small* universes of t-boundaried graphs. We used the Myhill–Nerode approach to obtain fixed-parameter linear-time algorithms for HYPERGRAPH CUTWIDTH by

(1) using the *method of test sets* to prove "finiteness" results in the *large universe*, which is not only more convenient but also of independent interest in the context of communication complexity, and
(2) then translating this result into an algorithm by means of a general machinery that links large universe results to the small universe of bounded treewidth representations.

This approach has the advantage of being relatively simple and powerful in the sense of (1) yet relatively general in the sense of (2). One of our principle objectives in this article was to establish powerful and general methodologies to solve hypergraph problems. If the principal objective were to have the most efficient algorithm for HYPERGRAPH CUTWIDTH, then this is not the way to go. That would probably be to hunker down as tightly as possible into the small universe, and do some serious dynamic programming [7,33,45]. Our machinery only shows when such seriousness is worthwhile: for example, it is for HYPERGRAPH CUTWIDTH, but probably not for HYPERTREE WIDTH and its variants.

There are many interesting open questions in this relatively under-explored area. The general theme is (referring to the very abstract program about Myhill–Nerode equivalence classes): how do these finiteness results relate to computational complexity? How do these finiteness issues translate between different settings? Courcelle and Lagergren [15] proved some very interesting results about translating finiteness results between the small and large universes when the property $\mathcal{P}$ is restricted to simple graphs of bounded treewidth. They showed that

(1) if $\mathcal{P}$ is restricted to simple graphs of treewidth at most t, then finite index of $\sim_{\mathcal{P}}$ over $\mathcal{U}_t^{\text{small}}$ implies finite index over to $\mathcal{U}_t^{\text{large}}$, and also that
(2) (Rephrasing their Fact 9.4:) there is a property $\mathcal{P}$ (of unbounded treewidth), such that for all t, the canonical Myhill–Nerode equivalence relation $\sim_{\mathcal{P}}$ has finite index over $\mathcal{U}_t^{\text{small}}$, but infinite index on $\mathcal{U}_t^{\text{large}}$.

Do those results extend to hypergraphs in the representation framework we have explored here?

Another question one might ask is whether there might be general methods and meta-theorems for obtaining XP-complexity results, based on communication complexity results with respect to $\mathcal{U}_t^{\text{large}}$.

References

1. Abrahamson, K.R., Fellows, M.R.: Cutset Regularity Beats Well-Quasi-Ordering for Bounded Treewidth. Tech. rep., Dept. Computer Science, University Victoria, Canada (1989)
2. Abrahamson, K.R., Fellows, M.R.: Finite automata, bounded treewidth, and well-quasiordering. In: Graph Structure Theory, American Mathematical Society, Contemporary Mathematics, vol. 147, pp. 539–564 (1991)
3. Bern, M.W., Lawler, E.L., Wong, A.L.: Why certain subgraph computations require only linear time. In: Proceedings of the 26th FOCS, IEEE Computer Society, pp. 117–125 (1985)
4. van Bevern, R., Downey, R.G., Fellows, M.R., Gaspers, S., Rosamond, F.A.: Myhill–Nerode Methods for Hypergraphs. arXiv:1211.1299v5 [cs.DM] (2015)
5. van Bevern, R., Fellows, M.R., Gaspers, S., Rosamond, F.A.: Myhill–Nerode methods for hypergraphs. In: Proceedings of the 24th ISAAC, LNCS, vol. 8283, pp. 372–382. Springer, Berlin (2013)
6. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)
7. Bodlaender, H.L., Kloks, T.: Efficient and constructive algorithms for the pathwidth and treewidth of graphs. J. Algorithms **21**(2), 358–402 (1996)
8. Bodlaender, H.L., Fellows, M.R., Warnow, T.J.: Two strikes against perfect phylogeny. In: Proceedings of the 19th ICALP, LNCS, vol. 623, pp. 273–283. Springer, Berlin (1992)
9. Bodlaender, H.L., Fellows, M.R., Hallett, M.T.: Beyond NP-completeness for problems of bounded width (extended abstract): hardness for the W hierarchy. In: Proceedings of the 26th STOC, pp. 449–458. ACM (1994)
10. Bodlaender, H.L., Fellows, M.R., Hallett, M.T., Wareham, H.T., Warnow, T.J.: The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. Theor. Comput. Sci. **244**(1–2), 167–188 (2000)
11. Bodlaender, H.L., Fellows, M.R., Thilikos, D.M.: Derivation of algorithms for cutwidth and related graph layout parameters. J. Comput. Syst. Sci. **75**(4), 231–244 (2009)
12. Borie, R.B., Parker, R.G., Tovey, C.A.: Solving problems on recursively constructed graphs. ACM Comput. Surv. **41**(1) (2009). doi:10.1145/1456650.1456654
13. Cahoon, J., Sahni, S.: Exact algorithms for special cases of the board permutation problem. In: Proceedings of the 21st Annual Allerton Conference on Communication, Control, and Computing, pp. 246–255 (1983)
14. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic—A Language-Theoretic Approach, Encyclopedia of mathematics and Its Applications, vol. 138. Cambridge University Press, Cambridge (2012)
15. Courcelle, B., Lagergren, J.: Equivalent definitions of recognizability for sets of graphs of bounded tree-width. Math. Struct. Comput. Sci. **6**(2), 141–165 (1996)
16. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin (1999)
17. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Springer, Berlin (2013)
18. Fellows, M., Langston, M.: An analogue of the Myhill–Nerode theorem and its use in computing finite-basis characterizations. In: Proceedings of the 30th FOCS, pp. 520–525. IEEE Computer Society (1989)
19. Fellows, M.R., Langston, M.A.: On well-partial-order theory and its application to combinatorial problems of VLSI design. SIAM J. Discrete Math. **5**(1), 117–126 (1992)
20. Fellows, M.R., Langston, M.A.: On search, decision, and the efficiency of polynomial-time algorithms. J. Comput. Syst. Sci. **49**(3), 769–779 (1994)
21. Fellows, M.R., Jansen, B.M.P., Rosamond, F.: Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity. Eur. J. Combin. **34**(3), 541–566 (2013)

22. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
23. Fomin, F.V., Golovach, P.A., Thilikos, D.M.: Approximating acyclicity parameters of sparse hypergraphs. In: Proceedings of the 26th STACS, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, LIPIcs, vol. 3, pp. 445–456 (2009)
24. Ganian, R., Hliněný, P.: On parse trees and Myhill–Nerode-type tools for handling graphs of bounded rank-width. Discrete Appl. Math. **158**(7), 851–867 (2010)
25. Garey, M.R., Graham, R.L., Johnson, D.S., Knuth, D.E.: Complexity results for bandwidth minimization. SIAM J. Appl. Math. **34**(3), 477–495 (1978)
26. Gaspers, S., Naroditskiy, V., Narodytska, N., Walsh, T.: Possible and necessary winner problem in social polls. In: Proceedings of the AAMAS'13, IFAAMAS, pp. 1131–1132 (2013)
27. Gavril, F.: Some NP-complete problems on graphs. In: Proceedings of the 1977 Conference on Information Science and Systems, Johns Hopkins University, pp. 91–95 (1977)
28. Gottlob, G., Grohe, M., Musliu, N., Samer, M., Scarcello, F.: Hypertree decompositions: structure, algorithms, and applications. In: Proceedings of the 31st WG, LNCS, vol. 3787, pp. 1–15. Springer, Berlin (2005)
29. Gottlob, G., Miklós, Z., Schwentick, T.: Generalized hypertree decompositions: NP-hardness and tractable variants. J. ACM **56**(6) (2009). doi:10.1145/1568318.1568320
30. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. J. Comb. Theory B **96**(3), 325–351 (2006)
31. Kolaitis, P.G., Vardi, M.Y.: Conjunctive-query containment and constraint satisfaction. J. Comput. Sci. **61**(2), 302–332 (2000)
32. Komusiewicz, C., Niedermeier, R.: New races in parameterized algorithmics. In: Proceedings of the 37th MFCS, LNCS, vol. 7464, pp. 19–30. Springer, Berlin (2012)
33. Lagergren, J., Arnborg, S.: Finding minimal forbidden minors using a finite congruence. In: Proceedings of the 18th ICALP, LCNS, vol. 510, pp. 532–543. Springer, Berlin (1991)
34. Lakshmipathy, N., Winklmann, K.: "Global" graph problems tend to be intractable. J. Comput. Syst. Sci. **32**(3), 407–428 (1986)
35. Mahajan, S., Peters, J.G.: Regularity and locality in k-terminal graphs. Discrete Appl. Math. **54**(2–3), 229–250 (1994)
36. Marx, D.: Approximating fractional hypertree width. ACM Trans Algorithms **6**(2), 29 (2010)
37. Miller, Z., Sudborough, I.H.: A polynomial algorithm for recognizing bounded cutwidth in hypergraphs. Math. Syst. Theory **24**(1), 11–40 (1991)
38. Myhill, J.: Finite Automata and Representation of Events. Tech. Rep. WADD TR-57-624, Wright-Patterson Air Force Base, Ohio, USA (1957)
39. Nagamochi, H.: Linear layouts in submodular systems. In: Proceedings of the 23rd ISAAC, LNCS, vol. 7676, pp. 475–484. Springer, Berlin (2012)
40. Nerode, A.: Linear automaton transformations. Proc. Am. Math. Soc. **9**(4), 541–544 (1958)
41. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
42. Niedermeier, R.: Reflections on multivariate algorithmics and problem parameterization. In: Proceedings of the 27th STACS, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, LIPIcs, vol. 5, pp. 17–32 (2010)
43. Prasad, M.R., Chong, P., Keutzer, K.: Why is ATPG easy? In: Proceedings of the 36th DAC, pp. 22–28. ACM (1999)
44. Samer, M., Szeider, S.: Constraint satisfaction with bounded treewidth revisited. J. Comput. Sci. **76**(2), 103–114 (2010)
45. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Cutwidth I: a linear time fixed parameter algorithm. J. Algorithms **56**(1), 1–24 (2005)
46. Wang, D., Clarke, E., Zhu, Y., Kukula, J.: Using cutwidth to improve symbolic simulation and Boolean satisfiability. In: Proceedings of the 6th HLDVT, pp. 165–170. IEEE (2001)
47. Wimer, T.V.: Linear Algorithms on k-Terminal Graphs. PhD thesis, Clemson University (1987)
48. Wimer, T.V., Hedetniemi, S.T., Laskar, R.: A methodology for constructing linear graph algorithms. Congr. Numer. **50**, 43–60 (1985)
49. Yao, A.C.: Some complexity questions related to distributed computing. In: Proceedings of the 11th STOC, pp. 209–213. ACM (1979)