SIAM J. COMPUT.
Vol. 40, No. 5, pp. 1292–1315

INTERVAL GRAPHS: CANONICAL REPRESENTATIONS IN LOGSPACE*

JOHANNES KÖBLER†, SEBASTIAN KUHNERT†, BASTIAN LAUBNER†, AND OLEG VERBITSKY‡

Abstract. We present a logspace algorithm for computing a canonical labeling, in fact, a canonical interval representation, for interval graphs. To achieve this, we compute canonical interval representations of interval hypergraphs. This approach also yields a canonical labeling of convex graphs. As a consequence, the isomorphism and automorphism problems for these graph classes are solvable in logspace. For proper interval graphs we also design logspace algorithms computing their canonical representations by proper and by unit interval systems.

Key words. graph isomorphism, graph canonization, logspace, interval graphs, interval hypergraphs, convex graphs, proper interval graphs, unit interval graphs

AMS subject classifications. 05C60, 05C85

DOI. 10.1137/10080395X

1. Introduction. There has been persistent interest in the algorithmic aspects of interval graphs in past decades, spurred much by their numerous applications; see, e.g., [Gol04]. In 1976, Booth and Lueker presented the first recognition algorithm for interval graphs [BL76] running in time linear in the number of vertices and edges, which they followed up by a linear-time algorithm for interval graph isomorphism [LB79]. These algorithms are based on a special data structure called *PQ-trees* that is used to enforce ordering constraints. By preprocessing the graph's modular decomposition tree, Hsu and Ma [HM99] later presented a simpler linear-time recognition algorithm that avoids the use of PQ-trees. Habib et al. [HMP+00] achieve the same time bound employing the lexicographic breadth-first search of Rose, Tarjan, and Lueker [RTL76] in combination with smart pivoting. Parallel AC^2 recognition and isomorphism algorithms were given by Klein in [Kle96].

All of the above algorithms have in common that they compute a *perfect elimination ordering* (peo) of the graph's vertices. This ordering has the property that for every vertex, its neighborhood among its successors forms a clique. Fulkerson and Gross [FG65] show that a graph has a peo if and only if it is chordal, and the above methods determine whether a graph is an interval graph in linear time once a peo is known.

Recognition of interval graphs in logspace follows from the results of Reif [Rei84] and Reingold [Rei08]. In this article, we describe a logspace algorithm that, given an interval graph G, constructs a canonical interval representation $\mathcal{I}_G$, i.e., G is isomorphic to the intersection graph of $\mathcal{I}_G$, and isomorphic graphs $G_1 \cong G_2$ are

*Received by the editors July 29, 2010; accepted for publication (in revised form) June 29, 2011; published electronically September 20, 2011. A preliminary version of this paper appeared in *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, 2010.
http://www.siam.org/journals/sicomp/40-5/80395.html

†Humboldt-Universität, Institut für Informatik, Berlin, Germany (koebler@informatik.hu-berlin.de, kuhnert@informatik.hu-berlin.de, laubner@informatik.hu-berlin.de). The second author's work was supported in part by DFG grant KO 1053/7–1.

‡Institute for Applied Problems of Mechanics and Mathematics, Ukrainian Academy of Sciences, Lviv, Ukraine (verbitsk@informatik.hu-berlin.de). This author's work was supported in part by the Alexander von Humboldt foundation.

mapped to equal interval representations $\mathcal{I}_{G_1} = \mathcal{I}_{G_2}$. This, in particular, gives another recognition algorithm and, moreover, implies that testing isomorphism of interval graphs is also possible in logspace. Our methods are optimized for space complexity. As such, our exposition neither relies on computing the graph's peo nor uses transitive orientation algorithms for comparability graphs as in [KVV85]. Instead, the basis of our work is the observation of Laubner [Lau10] that in an interval graph, the set of maximal cliques and a modular decomposition tree are definable in a certain logical formalism, which makes these objects tractable in logarithmic space. We remark that a logspace complexity bound implies an AC^1 bound, while logspace and linear time results are incomparable (neither implies the other).

More specifically, we reduce canonization of interval graphs to that of *interval hypergraphs*. We split interval hypergraphs into *overlap components* whose interval representations are essentially unique; we show how to compute them canonically using Reingold's algorithm [Rei08]. After placing these components in a tree and coloring them with their canonical interval representations, we apply Lindell's tree canonization algorithm [Lin92] and use its output to combine the canonical interval representations of the components to one for the whole hypergraph. The tree used in our algorithm is reminiscent of a PQ-tree in that it encodes all possible interval representations. Our tree allows us to choose one of the representations in a canonical way. This tree is constructible in logspace without the iterative refinement that is inherent to the linear-time algorithms.

A hypergraph is an interval hypergraph if its vertices can be ordered so that the vertices in each hyperedge are consecutive in this order. Switching to hypergraphs bears the advantage that these exhibit richer structure; this helps us to avoid technical complications and to focus on the essence of the algorithm. Recognition of interval hypergraphs is clearly equivalent to testing the so-called *consecutive-ones property*: A matrix (which we interpret as the incidence matrix of a hypergraph) has the consecutive-ones property for rows if its columns can be reordered such that the ones in each row are consecutive. Testing for this property has complexity similar to that of the recognition of interval graphs: Booth and Lueker gave a linear-time algorithm that uses PQ-trees [BL76], which was later simplified by Hsu and McConnell [HM03]. Parallel AC^2 algorithms were given by Chen and Yesha [CY91] and by Annexstein and Swaminathan [AS98]; they also follow from the parallel algorithms for PQ-trees by Klein and Reif [KR88]. Our result implies that testing the consecutive-ones property and finding an appropriate column permutation are in logspace.

As another consequence of our logspace canonization of interval hypergraphs, we show that *convex graphs* can be canonized in logspace. The isomorphism problem for this class was previously known to be decidable by a parallel algorithm in AC^2 [Che96] and by a sequential algorithm in linear time [Che99]. Convex graphs include bipartite permutation graphs. The isomorphism problem for the latter class was only known to be in AC^1 [CY93, YC96].

An interval graph is called *proper* if it admits an interval representation where no interval is contained in another. Such representations can be found in linear time by algorithms of Deng, Hell, and Huang [DHH96] and Hell, Shamir, and Sharan [HSS01]. An AC^2 algorithm is designed by Bang-Jensen, Huang, and Ibarra [BHI07]. We show how to compute canonical proper interval representations in logspace, implying also logspace recognition of proper interval graphs. *Unit interval graphs* are interval graphs representable by systems of intervals of unit length. Any such graph is obviously a proper interval graph, and the converse is also true by a classical result

of Roberts [Rob69]. Corneil et al. [CKN+95] show how to construct a unit interval representation in linear time. Based on their methods, we observe that logspace is sufficient to derive a unit interval representation from a proper one.

Finding logspace algorithms for the graph isomorphism problem of restricted graph classes is an active research area. It was started by Lindell with his canonization algorithm for trees [Lin92]. In a series of results, Datta et al. generalize this to planar graphs [DLN+09] (in fact, excluding one of K_5 or $K_{3,3}$ as minor is sufficient [DNT+09]), whereas Köbler and Kuhnert show the generalization to k-trees [KK09]. The graph classes considered in these results have in common that their clique size is bounded by a constant. To the best of our knowledge, our logspace algorithm for interval graph isomorphism is the first for a natural class of graphs containing cliques of arbitrary size. For all graph classes mentioned in this paragraph, the isomorphism problem has a matching lower bound; i.e., it turns out to be logspace complete.

Organization of the paper. Section 2 introduces some preliminaries, notably the decomposition of interval hypergraphs into overlap components, and includes a detailed overview of our algorithm in section 2.4. In section 3 we show how to compute a canonical interval representation for a single overlap component in logspace. Section 4 contains our main result: We give a logspace algorithm to obtain a canonical interval representation of an arbitrary interval hypergraph. In section 5, we state our results for interval graphs and convex graphs. Section 6 contains our algorithms for proper and unit interval representations. In section 7 we summarize our results and show that recognition and isomorphism testing of interval and convex graphs is hard for logspace, thereby proving logspace completeness for these problems.

2. Definitions and basic facts. As usual, L is the class of all languages decidable by Turing machines with a read-only input tape using only $\mathcal{O}(\log n)$ bounded space on the working tapes. FL is the class of all functions computable by such machines that additionally have a write-only output tape. For a set S, we denote its cardinality by $\|S\|$.

2.1. Graphs and set systems. We write $G \cong H$ to say that G and H are isomorphic graphs. The vertex set of a graph G is denoted by $V(G)$. The set of all vertices at distance at most 1 from a vertex $v \in V(G)$ is called the *(closed) neighborhood* of v and is denoted by $N[v]$. Note that $v \in N[v]$. We also use $N[u,v] = N[u] \cap N[v]$ for the common neighborhood of two vertices u and v. If $N[u] = N[v]$, we call these vertices *twins* (note that only adjacent vertices can be twins according to our terminology). We denote the *degree* of a vertex $v \in V(G)$ as $\deg(v) = \|N[v] \setminus \{v\}\|$.

Let $\mathcal{F}$ be a family of nonempty sets, which will also be called a *set system*. We allow $A = B$ for some $A, B \in \mathcal{F}$; i.e., $\mathcal{F}$ is a multiset whose elements are sets. The *support* of $\mathcal{F}$ is defined by $\operatorname{supp}(\mathcal{F}) = \bigcup_{X \in \mathcal{F}} X$. A *slot* is an inclusion-maximal subset S of $\operatorname{supp}(\mathcal{F})$ such that each set $A \in \mathcal{F}$ contains either all of S or none of it.

The *intersection graph* of $\mathcal{F}$ is the graph $\mathbb{I}(\mathcal{F})$ with vertex set $\mathcal{F}$ where A and B are adjacent if and only if they have a nonempty intersection. Note that, if $A = B$, these two vertices are twins in the intersection graph.

We consider intervals in the set of positive integers $\mathbb{N}$, using the standard notation $[a,b] = \{i \in \mathbb{N} \mid a \le i \le b\}$. We say $[a_1,b_1] < [a_2,b_2]$ if $a_1 < a_2$ or if $a_1 = a_2$ and $b_1 < b_2$. We extend this order to *interval systems*, i.e., multisets of intervals. For interval systems $\mathcal{I}$ and $\mathcal{J}$, we write $\mathcal{I} < \mathcal{J}$ if the smallest interval in the symmetric difference of $\mathcal{I}$ and $\mathcal{J}$ (with due regard to the multiplicities) belongs to $\mathcal{I}$.

A graph G is an *interval graph* if it is isomorphic to the intersection graph of a family of intervals $\mathcal{I}$. This is equivalent to the standard definition of interval graphs as intersection graphs of real intervals. Indeed, if we understand $[a, b]$ as a real interval, this does not change the intersection graph. On the other hand, given a finite system of real intervals $\mathcal{I}$, denote the set of all endpoints by P. Then the system of discrete intervals induced by $\mathcal{I}$ on P has the same intersection graph. It remains to embed P in $\mathbb{N}$ so that the order is preserved.

An isomorphism $\ell\colon V(G) \to \mathcal{I}$ from G to $\mathbb{I}(\mathcal{I})$ is called an *interval labeling* of G. The interval system $\mathcal{I}$ is called an *interval representation* of G and will also be denoted by G^ℓ.

A *labeling* of a graph G is a bijection $\ell\colon V(G) \to \{1, \dots, \|V(G)\|\}$. In this case G^ℓ will denote the isomorphic image of G on the vertex set $\{1, \dots, \|V(G)\|\}$. The *canonical (interval) labeling problem* for a class of graphs $\mathcal{G}$ consists of computing for a given graph $G \in \mathcal{G}$ an (interval) labeling ℓ_G such that $G^{\ell_G} = H^{\ell_H}$ whenever $G \cong H$. We call ℓ_G a *canonical (interval) labeling* and G^{ℓ_G} a *canonical (interval) representation* of G. Note that solving the canonical interval labeling problem implies solving the canonical labeling problem, as the intervals can be sorted and renamed.

2.2. Hypergraphs. We consider only hypergraphs $(V, \mathcal{H})$ without isolated vertices; i.e., the vertex set V is exactly $\operatorname{supp}(\mathcal{H})$. Hence, we often refrain from explicitly mentioning V. In order to represent multiple hyperedges, we assign to each hyperedge $H \in \mathcal{H}$ a positive integer $c(H) \geq 1$, called the *multiplicity* of H. It should be stressed that, speaking of a hypergraph $\mathcal{H}$, we will always suppose that $\mathcal{H}$ is a set rather than a multiset. In other words, if hyperedges $A \in \mathcal{H}$ and $B \in \mathcal{H}$ are equal as sets, they will be considered *the same* hyperedge (whose multiplicity can be more than 1). An isomorphism from a hypergraph $\mathcal{H}$ to a hypergraph $\mathcal{K}$ is a bijection $\phi\colon \operatorname{supp}(\mathcal{H}) \to \operatorname{supp}(\mathcal{K})$ such that

- $H \in \mathcal{H}$ if and only if $\phi(H) \in \mathcal{K}$ for every $H \subseteq \operatorname{supp}(\mathcal{H})$, and
- $c(H) = c(\phi(H))$ for every $H \in \mathcal{H}$.

We say that two hyperedges A and B *overlap* and write $A \between B$ if A and B have a nonempty intersection but neither of them includes the other. The *overlap graph* $\mathbb{O}(\mathcal{H})$ is the subgraph of the intersection graph $\mathbb{I}(\mathcal{H})$ where the vertices corresponding to the hyperedges A and B are adjacent if and only if they overlap.

Of course, $\mathbb{O}(\mathcal{H})$ can be disconnected even if $\mathbb{I}(\mathcal{H})$ is connected. A subset $\mathcal{O}$ of the hyperedges of $\mathcal{H}$ corresponding to a connected component of $\mathbb{O}(\mathcal{H})$ will be referred to as an *overlap component* of $\mathcal{H}$. This is a subhypergraph of $\mathcal{H}$ and should not be confused with the corresponding induced subgraph of $\mathbb{O}(\mathcal{H})$. Note that a hyperedge of an overlap component inherits the multiplicity that it has in $\mathcal{H}$.

If $\mathcal{O}$ and $\mathcal{O}'$ are different overlap components, then either every two hyperedges $A \in \mathcal{O}$ and $A' \in \mathcal{O}'$ are disjoint or all hyperedges of one of the two components are contained in a single slot of the other component. (This follows from a simple observation that the conditions $B \subset A$, $B \between B'$, and $\neg(B' \between A)$ imply that $B' \subset A$.) This containment relation determines a tree-like decomposition of $\mathcal{H}$ into its overlap components. In the case that $\mathbb{O}(\mathcal{H})$ is connected, $\mathcal{H}$ will be called an *overlap-connected hypergraph*.

We call an interval system $\mathcal{I}$ an *interval representation* of a hypergraph $\mathcal{H}$ if $\mathcal{I}$ viewed as a hypergraph is isomorphic to $\mathcal{H}$. Hypergraphs having interval representations are known in the literature as *interval hypergraphs* [BLS99, section 8.7]. Note that interval graphs are not just interval hypergraphs with hyperedges of size 2 (those are exactly unions of paths). This difference stems from the fact that inter-

vals correspond to the vertices of interval graphs and to the hyperedges of interval hypergraphs.

Any isomorphism ϕ from $\mathcal{H}$ to $\mathcal{I}$ induces a labeling $\ell_\phi\colon \mathcal{H} \to \mathcal{I}$ of the hyperedges in $\mathcal{H}$ with intervals from $\mathcal{I}$ where $\ell_\phi(H) = \{\phi(x) \mid x \in H\}$. We call a function $\ell\colon \mathcal{H} \to \mathcal{I}$ an *interval labeling* of $\mathcal{H}$ if $\ell = \ell_\phi$ for some ϕ. For another hypergraph isomorphism ψ from $\mathcal{H}$ to $\mathcal{I}$, we have $\ell_\phi = \ell_\psi$ if and only if $\psi^{-1}\phi$ maps every slot of $\mathcal{H}$ onto itself. In other words, an interval labeling $\ell\colon \mathcal{H} \to \mathcal{I}$ specifies an isomorphism from $\mathcal{H}$ to $\mathcal{I}$ up to arbitrary rearrangements within slots.

Speaking of an interval representation $\mathcal{I}$, we will suppose that $\mathrm{supp}(\mathcal{I}) = [1, k]$, where $k = \|\mathrm{supp}(\mathcal{I})\|$. The map $r(x) = k{+}1{-}x$ will be called the *mirror reflection*, and the isomorphic interval system $\mathcal{I}^* = r(\mathcal{I})$ will be referred to as the *mirror image* of $\mathcal{I}$. An interval system $\mathcal{I}$ is *mirror-symmetric* if $\mathcal{I}^* = \mathcal{I}$. The mirror image of an interval labeling $\ell\colon \mathcal{H} \to \mathcal{I}$ is the interval labeling $\ell^*\colon \mathcal{H} \to \mathcal{I}^*$ defined as $\ell^*(A) = r(\ell(A))$.

LEMMA 2.1. *Let $\mathcal{H}$ be an overlap-connected hypergraph with at least two hyperedges. Then $\mathcal{H}$ has either none or exactly two interval labelings, being mirror images of each other.*

Lemma 2.1 can be deduced from an equivalent statement in [CY91, Theorem 2]. For the reader's convenience we here give a direct, independent proof.

Proof. Let $\ell\colon \mathcal{H} \to \mathcal{I}$ be an interval labeling of $\mathcal{H}$. Since $\mathcal{I}$ and $\mathcal{H}$ are isomorphic, $\mathbb{O}(\mathcal{I})$ is connected and $\mathcal{I}$ contains at least two intervals. The intervals $I \in \mathcal{I}$ containing 1 (resp., $\|\mathrm{supp}(\mathcal{I})\|$) will be called *leftmost* (resp., *rightmost*). Denote the longest leftmost (resp., rightmost) interval in $\mathcal{I}$ by L (resp., R). Note that $L \neq R$ or else $\mathcal{I}$ would contain only one interval or $\mathbb{O}(\mathcal{I})$ would not be connected. Call a hyperedge $X \in \mathcal{H}$ *marginal* if the overlaps $\{X \cap Y \mid Y \in \mathcal{H}, Y \between X\}$ form a single inclusion chain. It is not hard to see that $\ell(X) \in \{L, R\}$ if and only if X is inclusion-maximal and marginal. The latter conditions define an unordered pair of hyperedges, A and B, in $\mathcal{H}$, that does not depend on ℓ. Without loss of generality, suppose that $\ell(A) = L$ and $\ell(B) = R$. By definition, $\ell^*(B) = r(R)$.

Now consider any interval labeling $\ell'\colon \mathcal{H} \to \mathcal{I}'$ mapping $\mathcal{H}$ to an arbitrary interval system $\mathcal{I}'$ with $\mathrm{supp}(\mathcal{I}') = [1, \|\mathrm{supp}(\mathcal{H})\|]$. As we just observed, the leftmost interval in $\mathcal{I}'$ equals either $\ell'(A)$ or $\ell'(B)$. Consider first the former case. We have $\ell'(A) = [1, \|A\|] = L$; that is, ℓ' coincides with ℓ on A. Using induction on the distance d between A and X in $\mathbb{O}(\mathcal{H})$, we prove that ℓ' and ℓ coincide on all $X \in \mathcal{H}$. If $d = 1$, then $\ell'(X) = \ell(X)$ because both intervals must be equal to $[\|A \setminus X\| + 1, \|A \cup X\|]$. If $d \geq 2$, let $Z \between Y \between X$ be the terminal part of a shortest path from A to X in $\mathbb{O}(\mathcal{H})$. By the induction hypothesis, we have $\ell'(Y) = \ell(Y) = I$ and $\ell'(Z) = \ell(Z) = J$. It suffices to show that the intervals I and J uniquely determine $\ell(X)$ and $\ell'(X)$ and the determination rules for both are identical. Indeed, both $\ell(X)$ and $\ell'(X)$ contain exactly one endpoint of I, which is shared with J if and only if X and Z have nonempty intersection. This determines the side of I where $\ell(X)$ and $\ell'(X)$ have to be attached. The exact position of $\ell(X)$ and $\ell'(X)$ is determined by the length of the overlap with I, which is equal to $\|X \cap Y\|$. We have proved that $\ell' = \ell$.

In the case that $\ell'(B)$ is leftmost, we have $\ell'(B) = [1, \|B\|] = \ell^*(B)$, and the same argument shows that $\ell' = \ell^*$. Thus, there exists no interval labeling of $\mathcal{H}$ different from ℓ and ℓ^*. ∎

In section 3 we prove a constructive version of Lemma 2.1 (namely, Lemma 3.2), allowing us to show that the unique pair of mutually reversed interval labelings is efficiently computable. In fact, in section 3 we switch into another, equivalent language. Given an isomorphism ϕ from a hypergraph $\mathcal{H}$ to an interval system $\mathcal{I}$, note that ϕ

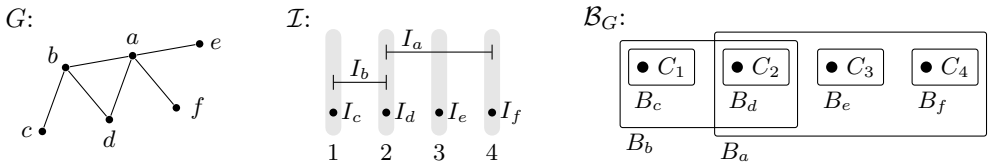

FIG. 1. *An interval graph G, a minimal interval representation $\mathcal{I}$ of G, and the bundle hypergraph $\mathcal{B}_G$ of G. The maxcliques of G are $C_1 = \{b, c\}$, $C_2 = \{a, b, d\}$, $C_3 = \{a, e\}$, and $C_4 = \{a, f\}$.*

takes a slot of $\mathcal{H}$ onto a slot of $\mathcal{I}$ and the slots of $\mathcal{I}$ form a partition of $\mathrm{supp}(\mathcal{I})$ into intervals. Thus, ϕ determines a natural geometric order $\prec_\phi$ between the slots of $\mathcal{H}$: for two slots S and S' we put $S \prec_\phi S'$ if $\phi(S)$ lies completely to the left of $\phi(S')$ in $\mathbb{N}$. It is easy to see that $\prec_\phi = \prec_\psi$ exactly when $\psi^{-1}\phi$ maps every slot of $\mathcal{H}$ onto itself. Therefore, we have equality $\ell_\phi = \ell_\psi$ for interval labelings exactly when we have equality $\prec_\phi = \prec_\psi$ for slot orderings. Moreover, given ℓ_ϕ, it is easy (in logspace) to construct $\prec_\phi$ and vice versa; hence, the two notions are equivalent for our purposes.

2.3. From interval graphs to interval hypergraphs. An inclusion-maximal clique in a graph G will be called a *maxclique*. The *(maxclique) bundle* B_v at a vertex v consists of all maxcliques containing v. The *bundle hypergraph* of G is defined by the set system $\mathcal{B}_G = \{B_v\}_{v \in V(G)}$; i.e., it has the maxcliques of G as vertices and the maxclique bundles as hyperedges. Note that for twins $u, v \in V(G)$, the bundles B_u and B_v are equal; in this case the corresponding hyperedge has multiplicity greater than one.

LEMMA 2.2. *Every maxclique C of an interval graph G contains vertices u and v such that $C = N[u, v]$.*

Proof. Given an interval representation of G, we will use the notation I_v for the interval corresponding to a vertex $v \in V(G)$. We have $C \subseteq N[u, v]$ for any $u, v \in C$, and, therefore, we need only find u, v such that $N[u, v] \subseteq C$. For this purpose, consider an interval representation of G and choose $u, v \in C$ so that $I_u \cap I_v$ is inclusion-minimal. For any $w \in C$, we have $I_w \supseteq I_u \cap I_v$ or else $I_u \cap I_w$ or $I_w \cap I_v$ would be strictly included in $I_u \cap I_v$. Suppose now that $z \in N[u, v]$. Since I_z intersects $I_u \cap I_v$, it has nonempty intersection with I_w for each $w \in C$. By maximality, $z \in C$. ∎

For adjacent vertices u and v in an arbitrary graph G holds: If $N[u, v]$ is a clique, it is maximal. Lemma 2.2 shows that, in an interval graph G, any maxclique is of this kind and, hence, can be represented by a pair of vertices u and v (that are adjacent and satisfy the condition that $N[u, v]$ is a clique). An explicit representation of the bundle hypergraph $\mathcal{B}_G$ of G can be computed in logspace by listing, for each bundle B_v, the maxcliques that contain v.

We call an interval representation $\mathcal{I}$ of an interval graph G *minimal* if the size of $\mathrm{supp}(\mathcal{I})$ is the smallest possible. The following lemma has several important consequences. First, it implies that G is an interval graph if and only if $\mathcal{B}_G$ is an interval hypergraph (this equivalence is well known; cf. [Gol04, Theorems 8.1 and 8.3]). Moreover, the bundle hypergraph $\mathcal{B}_G$ captures all information about a minimal interval representation of G, which is unique up to hypergraph isomorphism. In particular, $\mathcal{B}_G$ retains the isomorphism type of G (in fact, $G \cong \mathbb{I}(\mathcal{B}_G)$ via the isomorphism $v \mapsto B_v$ holds for any graph—not just for interval graphs). See Figure 1 for an example.

LEMMA 2.3. *For every minimal representation $\mathcal{I}$ of an interval graph G, the interval system $\mathcal{I}$ viewed as a hypergraph is isomorphic to the bundle hypergraph $\mathcal{B}_G$ of G.*

Proof. Denote the set of all maxcliques of G by M. As in the proof of Lemma 2.2, the interval of $\mathcal{I}$ corresponding to a vertex v of G will be denoted by I_v. This proof actually shows that every $C \in M$ contains vertices u and v such that the three conditions $w \in C$, $I_u \cap I_v \subseteq I_w$, and $(I_u \cap I_v) \cap I_w \neq \emptyset$ are equivalent. It follows that for each C we can choose a point $x_C \in \operatorname{supp}(\mathcal{I})$ (in fact, an arbitrary point in $I_u \cap I_v$) so that

$$w \in C \quad \Leftrightarrow \quad I_w \ni x_C. \tag{2.1}$$

Note that $x_C \neq x_{C'}$ if $C \neq C'$. Let $X = \{x_C \mid C \in M\}$ and $I'_w = I_w \cap X$ for all $w \in V(G)$. The system $\mathcal{I}' = \{I'_w \mid w \in V(G)\}$ is still an interval representation of G (here we speak of intervals in the linearly ordered set X). Indeed, if u and v are adjacent, let C be a maxclique containing u and v and note that both I'_u and I'_v contain x_C; if u and v are nonadjacent, then $I'_u \cap I'_v = \emptyset$ because $I_u \cap I_v = \emptyset$.

By minimality of $\mathcal{I}$, we conclude that $\mathcal{I}' = \mathcal{I}$ and $\operatorname{supp}(\mathcal{I}) = X$. Therefore, the correspondence $C \mapsto x_C$ is a bijection from M to $\operatorname{supp}(\mathcal{I})$. By (2.1), this is actually a hypergraph isomorphism from $\mathcal{B}_G$ to $\mathcal{I}$ (since the condition $w \in C$ can be rewritten as $C \in B_w$). ∎

While it is often assumed that the endpoints of the intervals of an interval system are distinct, Lemma 2.3 gives our motivation to focus on computing minimal interval representations, where some of the endpoints coincide. This way we avoid arbitrary choices that are not inherent to the canonical representation problem. We remark that the two notions are equivalent: To obtain a minimal interval representation from a distinct-endpoint interval representation, contract each block of consecutive start points together with the following block of consecutive endpoints to a single point. For the reverse direction, replace each point by a set of consecutive points, one for each interval that starts or ends at this point, placing the start points first. If the start points are ordered by interval length and the endpoints by the corresponding start points, this transformation preserves canonicity.

Given an interval labeling $\ell\colon \mathcal{H} \to \mathcal{I}$ of a hypergraph $\mathcal{H}$, we will denote the interval system $\mathcal{I}$ also by $\mathcal{H}^\ell$. The *canonical interval labeling problem for hypergraphs* is to compute for a given interval hypergraph $\mathcal{H}$ an interval labeling $\ell_\mathcal{H}$ so that $\mathcal{H}^{\ell_\mathcal{H}} = \mathcal{K}^{\ell_\mathcal{K}}$ whenever $\mathcal{H} \cong \mathcal{K}$. We call $\ell_\mathcal{H}$ a *canonical interval labeling* and $\mathcal{H}^{\ell_\mathcal{H}}$ a *canonical interval representation* of $\mathcal{H}$.

LEMMA 2.4. *The canonical interval labeling problem for interval graphs is reducible in logspace to the canonical interval labeling problem for interval hypergraphs.*

Proof. Let G be an interval graph. By Lemma 2.3, the bundle hypergraph $\mathcal{H} = \mathcal{B}_G$ of G is interval. We define an interval labeling ℓ_G for G using the canonical interval labeling $\ell_\mathcal{H}$ of $\mathcal{H}$ by setting $\ell_G(v) = \ell_\mathcal{H}(B_v)$ for each vertex v of G. Since the correspondence $v \mapsto B_v$ is a graph isomorphism from G to $\mathbb{I}(\mathcal{H})$ and $\ell_\mathcal{H}$ is a hypergraph isomorphism from $\mathcal{H}$ to the interval system $\mathcal{H}^{\ell_\mathcal{H}}$, the map ℓ_G is an isomorphism from G to $\mathbb{I}(\mathcal{H}^{\ell_\mathcal{H}})$. This shows that ℓ_G is indeed an interval labeling of G. It is canonical because $\mathcal{B}_G \cong \mathcal{B}_{G'}$ whenever $G \cong G'$ and because $\ell_\mathcal{H}$ is canonical. It remains to note that the bundle hypergraph $\mathcal{B}_G$ and the map $v \mapsto B_v$ are constructible in logspace due to Lemma 2.2. ∎

2.4. An overview of our canonization algorithms. Lemma 2.4 reduces computing a canonical interval labeling for interval graphs to computing one for interval hypergraphs.

Given an interval hypergraph $\mathcal{H}$, we start computing a canonical interval labeling $\ell_\mathcal{H}$ piecewise, first finding interval representations for each overlap component $\mathcal{O}$ of

$\mathcal{H}$. Notice that the overlap components of $\mathcal{H}$ can be computed in logspace using undirected reachability in $\mathbb{O}(\mathcal{H})$ [Rei08]. By Lemma 2.1, $\mathcal{O}$ has exactly two interval labelings $\ell\colon \mathcal{O} \to \mathcal{I}$ and $\ell^*\colon \mathcal{O} \to \mathcal{I}^*$. As the canonical version $\mathcal{I}_\mathcal{O}$ we take the smaller of $\mathcal{I}$ and $\mathcal{I}^*$ with respect to the order on interval systems introduced in section 2.1 (or any of them in the mirror-symmetric case $\mathcal{I} = \mathcal{I}^*$). Section 3 explains how to perform this phase in logarithmic space.

To compose all $\mathcal{I}_\mathcal{O}$'s into an interval representation $\mathcal{I}_\mathcal{H}$ of the whole hypergraph $\mathcal{H}$, we use the tree-like decomposition of $\mathcal{H}$ into overlap components (see section 2.2) to construct a tree representation $\mathbb{T}(\mathcal{H})$ of $\mathcal{H}$ (see section 4). Lindell's tree canonization algorithm [Lin92] allows us to compute $\mathcal{I}_\mathcal{H}$ canonically.

3. Canonizing overlap components. In this section we show how to compute a canonical interval labeling for an overlap-connected interval hypergraph $\mathcal{O}$. We call two vertices $u, v \in \operatorname{supp}(\mathcal{O})$ *indistinguishable in* $\mathcal{O}$ and write $u \sim_\mathcal{O} v$, if there is no hyperedge $B \in \mathcal{O}$ that contains exactly one of them. Clearly, $\sim_\mathcal{O}$ is an equivalence relation on $\operatorname{supp}(\mathcal{O})$. The equivalence classes of $\sim_\mathcal{O}$ are the slots of $\mathcal{O}$. If $\mathcal{O}$ consists of a single hyperedge B, we use the interval labeling $\ell_\mathcal{O}$ that maps B to $[1, \|B\|]$. So from now on we assume that $\mathcal{O}$ consists of at least two hyperedges.

By Lemma 2.1 we know that $\mathcal{O}$ has a unique, up to reflection, interval labeling ℓ. Recall that an interval labeling of $\mathcal{O}$ uniquely determines the action of the underlying isomorphism on the slots of $\mathcal{O}$ (see section 2.2). The slots of $\mathcal{O}$ that are placed by ℓ at the left or right end will be called *side-slots*. We first show that the two side-slots of $\mathcal{O}$ can be identified in logarithmic space; then we show how to compute the order on the other slots once the left end is fixed. Note that in a logspace computation, any slot can be represented by a single vertex contained in it; the other vertices in the slot can be easily computed since the relation $\sim_\mathcal{O}$ is decidable in logspace. In the forthcoming argument, we will make use of the fact that sentences referring to the set-theoretic relations and involving quantification over vertices and hyperedges are verifiable in logspace.

LEMMA 3.1. *Let $\mathcal{O}$ be an overlap-connected interval hypergraph with at least two hyperedges. Then the two side-slots of $\mathcal{O}$ can be found in* FL.

Proof. As $\mathcal{O}$ is an interval hypergraph, there exists an interval labeling ℓ of $\mathcal{O}$. By Lemma 2.1, ℓ is unique up to reflection. Like in the proof of that lemma, we call a hyperedge B *marginal* if its intersections with the overlapping hyperedges $B' \between B$ form a single inclusion chain. Note that, unless an inclusion-maximal interval in $\mathcal{O}^\ell$ starts leftmost or ends rightmost, it is overlapped by other inclusion-maximal intervals from both sides; otherwise $\mathcal{O}$ would not be overlap-connected. It follows that we can identify (i.e., recognize in logspace) the two hyperedges $B_1, B_2 \in \mathcal{O}$ that are mapped by ℓ to the longest interval starting leftmost and the longest interval ending rightmost: (1) They are marginal, and (2) they are not included in any other hyperedge in $\mathcal{O}$. It is clear that each side-slot of $\mathcal{O}$ is contained in exactly one of the hyperedges B_1 and B_2. Denote the side-slots by S_1 and S_2, assuming that $S_i \subset B_i$. Our aim is to characterize S_1 and S_2 by logspace verifiable conditions. The following argument applies to each $i = 1, 2$.

Given a slot S of $\mathcal{O}$, define $\mathcal{B}(S) = \{B \in \mathcal{O} \mid S \subseteq B\}$. Note that $\mathcal{B}(S) \neq \mathcal{B}(S')$ if $S \neq S'$. Looking at the interval system $\mathcal{O}^\ell$, which is an isomorphic image of $\mathcal{O}$, we see that the slot $S = S_i$ satisfies the following condition: Every $B \in \mathcal{B}(S)$ is marginal and is a subset of B_i. Denote the set of all slots satisfying this condition by $\mathcal{S}_i$. Thus, we have to identify S_i amongst other slots in $\mathcal{S}_i$.

Let ϕ be an arbitrary isomorphism underlying the interval labeling ℓ. Without

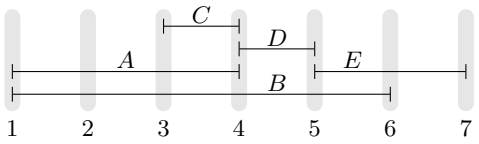

FIG. 2. *Identification of the side-slots S_1 and S_2: The two inclusion-maximal marginal hyperedges are $B_1 = B$ and $B_2 = E$; B_1 contains the marginal hyperedges A and C; $\mathcal{S}_1$ consists of the slots $S' = \{1, 2\}$ and $S'' = \{3\}$. Since $\mathcal{B}(S') = \{A, B\}$ is included in $\mathcal{B}(S'') = \{A, B, C\}$, we have $S_1 = S'$.*

loss of generality, suppose that $\ell(B_i) = \phi(B_i)$ is leftmost.

Claim A. Let S and S' be two slots in $\mathcal{S}_i$. If the interval $\phi(S)$ lies to the left of the interval $\phi(S')$, then $\mathcal{B}(S')$ properly contains $\mathcal{B}(S)$.

Claim A implies that S_i is exactly that slot in $\mathcal{S}_i$ which minimizes $\|\mathcal{B}(S)\|$ over $S \in \mathcal{S}_i$ (see an example in Figure 2). This gives us the lemma as, given a slot S, one can in logspace decide if $S \in \mathcal{S}_i$ and compute $\|\mathcal{B}(S)\|$.

It remains to prove Claim A. Let T be the slot in $\mathcal{S}_i$ with the interval $\phi(T)$ lying rightmost. It suffices to show that there is no hyperedge $B \in \mathcal{O}$ with $\phi(B)$ lying to the left of $\phi(T)$. Claim A follows from this because then, for any $S \in \mathcal{S}_i$ and $B \in \mathcal{B}(S)$, the interval $\phi(B)$ contains $\phi(T)$ and hence all $\phi(S')$ between $\phi(S)$ and $\phi(T)$.

To finish the proof, suppose to the contrary that some $\phi(B)$ lies to the left of $\phi(T)$. Let U be the slot of $\mathcal{O}$ whose image $\phi(U)$ is the right neighbor of $\phi(T)$ in $\mathcal{O}^\ell$. Consider a shortest overlap-chain of hyperedges $A_1 \between A_2 \between \dots \between A_k = B$ such that $U \subset A_1$ (such a chain exists because $\mathcal{O}$ is overlap-connected). Since $U \not\subseteq A_j$ for any $j \geq 2$, all $\phi(A_j)$ for $j \geq 2$ are to the left of $\phi(U)$. In particular, $\phi(A_2)$ overlaps $\phi(A_1)$ on its left side. Since $\phi(U) \not\subseteq \phi(A_2)$, the interval $\phi(A_1)$ must contain $\phi(T)$. Taking into account that $T \subset A_1$ and $T \in \mathcal{S}_i$, we conclude that $A_1 \subseteq B_i$ and A_1 is marginal. The latter condition implies that there is no hyperedge A_0 such that $\phi(A_0)$ overlaps $\phi(A_1)$ on its right side. Let W be the slot within A_1 whose image $\phi(W)$ is the rightmost slot within $\phi(A_1)$ (it is possible that $W = U$). Clearly, $\phi(W)$ lies to the right of $\phi(T)$. We show that $W \in \mathcal{S}_i$, contradicting the choice of T.

Indeed, consider an arbitrary hyperedge $A \in \mathcal{B}(W)$. Since $\phi(A)$ can overlap $\phi(A_1)$ neither on the right nor on the left side, we have either $A \supset A_1$ or $A \subset A_1$. In the former case $T \subset A$, which implies that $A \subseteq B_i$ and A is marginal. In the latter case $A \subseteq A_1 \subseteq B_i$. Moreover, there is no hyperedge A_0 such that $\phi(A_0)$ overlaps $\phi(A)$ on its right side, for else $\phi(A_0)$ would overlap $\phi(A_1)$ on the same side. Therefore, A is marginal in this case as well. The proof is complete. ∎

Following [Lau10], we can now use a side-slot S to define a partial order $\prec_S$ on $\mathrm{supp}(\mathcal{O})$ as the smallest relation that satisfies the following properties:

1. $u \prec_S v$ for each $u \in S$ and $v \notin S$.
2. For each hyperedge B in $\mathcal{O}$ and vertices $u, v \in B$, $w \notin B$,

$$u \prec_S w \;\Leftrightarrow\; v \prec_S w \qquad \text{and} \qquad w \prec_S u \;\Leftrightarrow\; w \prec_S v. \tag{3.1}$$

$u \prec_S v$ can be read as "if slot S is leftmost, then vertex u is left of v."

LEMMA 3.2. *If S is a side-slot of $\mathcal{O}$, then $\prec_S$ induces a strict linear order on the slots of $\mathcal{O}$, which is equal to the order in which slots appear in an interval labeling of $\mathcal{O}$.*

Proof. Let $\mathcal{I}$ be an interval representation of $\mathcal{O}$ and ϕ be a hypergraph isomorphism from $\mathcal{O}$ to $\mathcal{I}$. Suppose that $\phi(S)$ is placed leftmost (reverse the interval

representation if necessary). Define a relation $\prec$ on $\mathrm{supp}(\mathcal{O})$ by setting $u \prec v$ if and only if the interval $\phi([u]_{\sim_\mathcal{O}})$ lies strictly to the left of the interval $\phi([v]_{\sim_\mathcal{O}})$. Since Conditions 1 and 2 in the definition of $\prec_S$ are true for $\prec$, $\prec_S$ is a subrelation of $\prec$. We will prove that $\prec_S$ is actually equal to $\prec$.

Given $u \not\sim_\mathcal{O} v$, it suffices to show that either $u \prec_S v$ or $v \prec_S u$. Let $B_0 \between \cdots \between B_k$ be a shortest overlap-path in $\mathcal{O}$ such that $S \subseteq B_0$ and B_k contains precisely one of u and v. The claim is proved by induction on the length k of the path. If $k = 0$, the claim holds by Conditions 1 and 2 because there is a hyperedge, namely, B_0, containing S and precisely one of u and v. If $k \geq 1$, suppose without loss of generality that $u \in B_k$. If $v \notin B_{k-1}$, then for any $w \in B_{k-1}$ we have either $v \prec_S w$ or $w \prec_S v$ by induction, and since $B_{k-1} \cap B_k \neq \emptyset$ the claim follows by (3.1). If $v \in B_{k-1}$, then also $u \in B_{k-1}$ since we assumed the path to be shortest. In this case, for any $w' \in B_k \setminus B_{k-1}$, we have $v \prec_S w'$ or $w' \prec_S v$ by induction, and again the claim follows by (3.1). ∎

By Lemma 3.2 we can use $\prec_S$ to define an interval labeling ℓ_S of $\mathcal{O}$ by

$$\ell_S(B) = \big[\mathrm{pos}(B), \mathrm{pos}(B) + \|B\| - 1\big], \tag{3.2}$$

where a vertex $v \in \mathrm{supp}(\mathcal{O})$ has position $\mathrm{pos}(v) = \|\{u \in \mathrm{supp}(\mathcal{O}) \mid u \preceq_S v\}\|$ and a hyperedge $B \in \mathcal{O}$ has position $\mathrm{pos}(B) = \min\{\mathrm{pos}(v) \mid v \in B\}$.

Let ℓ_{S_1} and ℓ_{S_2} be the interval labelings for $\mathcal{O}$ corresponding to the two side-slots S_1 and S_2 of $\mathcal{O}$. By Lemma 2.1, these are the only two interval labelings of $\mathcal{O}$. It easily follows that the pair $\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$ stays the same for any isomorphic copy of $\mathcal{O}$. Hence, we can choose a canonical interval labeling $\ell_\mathcal{O}$ of $\mathcal{O}$ among ℓ_{S_1} and ℓ_{S_2} such that the interval system $\mathcal{O}^{\ell_\mathcal{O}}$ is minimal in the pair $\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$ (if $\mathcal{O}^{\ell_{S_1}} = \mathcal{O}^{\ell_{S_2}}$ is mirror-symmetric, we choose it arbitrarily). We denote the corresponding partial order on the vertices of $\mathcal{O}$ by $\prec_\mathcal{O}$.

LEMMA 3.3. *Given an overlap-connected hypergraph $\mathcal{O}$, the following can be done in logspace:*

1. *computing the partial order $\prec_\mathcal{O}$ on* $\mathrm{supp}(\mathcal{O})$*,*
2. *computing a canonical interval labeling $\ell_\mathcal{O}$ of $\mathcal{O}$, and*
3. *deciding if $\mathcal{O}^{\ell_\mathcal{O}}$ is mirror-symmetric or not.*

Proof. To prove that $u \prec_S v$ can be decided in logspace, we construct an auxiliary undirected graph G:

$$\begin{aligned} V(G) &= \{s\} \cup \big\{(u,v) \mid u, v \in \mathrm{supp}(\mathcal{O}) \text{ with } u \neq v\big\}, \\ E(G) &= \big\{\{s, (u,v)\} \mid u \in S, v \notin S\big\} \\ &\quad \cup \big\{\{(u,w),(v,w)\} \mid \exists\, B \in \mathcal{O} : u, v \in B, w \notin B\big\} \\ &\quad \cup \big\{\{(w,u),(w,v)\} \mid \exists\, B \in \mathcal{O} : u, v \in B, w \notin B\big\}. \end{aligned}$$

A node (u,v) corresponds to the statement "$u \prec_S v$." Using this interpretation, the edges of G correspond closely to Conditions 1 and 2 in the definition of $\prec_S$; so we have $u \prec_S v$ if and only if there is a path from s to (u,v) in G. Reachability in undirected graphs is decidable in L using Reingold's algorithm [Rei08].

Once $\prec_S$ can be decided in logspace, it is easy to compute ℓ_S according to (3.2) and to choose the left side-slot $S \in \{S_1, S_2\}$ so that $\mathcal{O}^{\ell_S} = \min\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$. Finally, $\mathcal{O}^{\ell_\mathcal{O}}$ is mirror-symmetric if and only if both interval representations are equal. ∎

4. Canonizing interval hypergraphs. Let $\mathcal{H}$ be an interval hypergraph. We assume that $\mathcal{H}$ is connected: To ensure this, we add an additional hyperedge $B_0 = \mathrm{supp}(\mathcal{H})$ (and discard it once the canonical interval labeling is computed).

4.1. The tree representation. As noted before, the overlap components of $\mathcal{H}$ form a tree. Specifically, let S be a slot of an overlap component $\mathcal{O}$ of $\mathcal{H}$. We say that an overlap component $\mathcal{Q}$ of $\mathcal{H}$ is *located at* slot S of $\mathcal{O}$ if $\mathrm{supp}(\mathcal{Q}) \subseteq S$ and there is no "intermediate" overlap component $\mathcal{O}' \neq \mathcal{O}$ such that $\mathrm{supp}(\mathcal{O}') \subseteq S$ and $\mathcal{Q}$ is contained in some slot of $\mathcal{O}'$. Furthermore, a vertex v of $\mathcal{H}$ is *located at* slot S of $\mathcal{O}$ if $v \in S$ and there is no overlap component $\mathcal{O}'$ located at slot S of $\mathcal{O}$ such that $v \in \mathrm{supp}(\mathcal{O}')$. We now define the *overlap component tree* of $\mathcal{H}$ as follows: Its nodes are the overlap components of $\mathcal{H}$, their slots, and the vertices of $\mathcal{H}$. Since a slot S of $\mathcal{O}$ may belong also to another overlap component, we denote the corresponding slot node by $S_{\mathcal{O}}$. The children of an overlap component node $\mathcal{O}$ are the slots of $\mathcal{O}$. The children of a slot node $S_{\mathcal{O}}$ are the vertices and the overlap components located at the slot S of $\mathcal{O}$. As $\mathcal{H}$ is connected, there is an overlap component $\mathcal{O}_0$ with $\mathrm{supp}(\mathcal{H}) = \mathrm{supp}(\mathcal{O}_0)$. Thus, $\mathcal{O}_0$ is the root of the overlap component tree. Figure 3 shows an interval hypergraph $\mathcal{H}_1$ and its overlap component tree.

We remark that the overlap component tree of $\mathcal{H}$ is essentially a *PQ-tree* that encodes all possible interval representations of $\mathcal{H}$ (cf. [BL76]). Start with any interval representation of $\mathcal{H}$, and order sibling nodes in the tree by their position in this representation. Slot nodes are *P-nodes*; i.e., their children can be reordered arbitrarily. Overlap component nodes are *Q-nodes*; i.e., the order of their children can only be reversed. It is easy to verify that, after reordering in this way, reading the vertex nodes from left to right again corresponds to an interval representation of $\mathcal{H}$. Using Lemma 2.1 and a simple inductive argument on the depth of the overlap component tree, one can easily show that every interval representation of $\mathcal{H}$ is obtainable in this way (cf. [HM03]). Though we do not use the latter fact below, it may be of independent interest to note that part 1 of Lemma 3.3 implies logspace constructibility of a PQ-tree for $\mathcal{H}$ and, using this, logspace constructibility of a (noncanonical) interval representation for $\mathcal{H}$.

In order to obtain a canonical interval labeling of an interval hypergraph $\mathcal{H}$, we will construct a tree representation $\mathbb{T}(\mathcal{H})$ such that $\mathcal{H}_1 \cong \mathcal{H}_2$ if and only if $\mathbb{T}(\mathcal{H}_1) \cong \mathbb{T}(\mathcal{H}_2)$. To achieve this, we start with the overlap component tree of $\mathcal{H}$ and color the component nodes with their canonical interval representation. However, this is not enough to ensure that isomorphic tree representations imply isomorphic hypergraphs, as the tree isomorphisms are not restricted in the way they permute the children of an overlap component node. An example where this is a problem can be seen in Figure 3.

To make sure that isomorphic trees imply isomorphic hypergraphs, we constrain the order of the slot nodes of an overlap component $\mathcal{O}$. If $\mathcal{O}$ is not mirror-symmetric, we take a first step toward canonization and completely fix the order of its slots. We do this by assigning a distinct color to each slot S of $\mathcal{O}$, namely, its position $\mathrm{lpos}_{\mathcal{O}}(S)$ from the left in the canonical interval representation of $\mathcal{O}$:

$$\mathrm{lpos}_{\mathcal{O}}(S) = \|\{S' \mid S' \text{ is a slot of } \mathcal{O} \text{ with } S' \preceq_{\mathcal{O}} S\}\|.$$

For overlap components $\mathcal{O}$ with mirror-symmetric interval representation we need to make the order of the slot nodes $S_{\mathcal{O}}$ of $\mathcal{O}$ unique up to reflection. To this end, we color each $S_{\mathcal{O}}$ with the minimum of $\mathrm{lpos}_{\mathcal{O}}(S)$ and $\mathrm{rpos}_{\mathcal{O}}(S)$, where

$$\mathrm{rpos}_{\mathcal{O}}(S) = \|\{S' \mid S' \text{ is a slot of } \mathcal{O} \text{ with } S \preceq_{\mathcal{O}} S'\}\|$$

is the position of S from the right in the canonical interval representation of $\mathcal{O}$. Note that any two slot nodes with the same color are interchangeable, which still

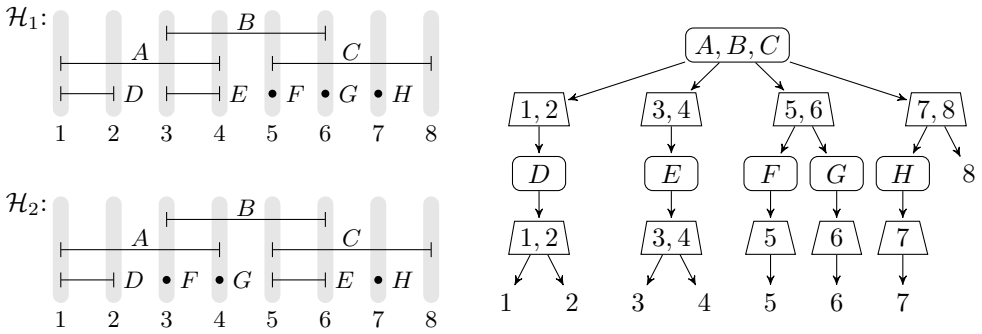

FIG. 3. *An interval hypergraph $\mathcal{H}_1$ and its overlap component tree. In the tree, the node of an* overlap component *$\mathcal{O}$ is given by listing the hyperedges in $\mathcal{O}$; a* slot *node $S_\mathcal{O}$ is given by listing the vertices contained in S (we omit the reference to $\mathcal{O}$ as it is understood from the tree structure). The hypergraph $\mathcal{H}_2$ is not isomorphic to $\mathcal{H}_1$. Yet both have isomorphic overlap component trees.*

causes the problem shown in Figure 3. To overcome this problem, we group the slots of the overlap components by introducing an additional type of node in the tree: For each overlap component $\mathcal{O}$, we add three nodes $lo_\mathcal{O}$, $mi_\mathcal{O}$, and $hi_\mathcal{O}$ in a layer between $\mathcal{O}$ and its slots. If $\mathcal{O}$ has a mirror-symmetric canon $\mathcal{O}^{\ell o}$, we call a slot S of $\mathcal{O}$ *low* if $\mathrm{lpos}_\mathcal{O}(S) < \mathrm{rpos}_\mathcal{O}(S)$, *middle* if $\mathrm{lpos}_\mathcal{O}(S) = \mathrm{rpos}_\mathcal{O}(S)$, and *high* if $\mathrm{lpos}_\mathcal{O}(S) > \mathrm{rpos}_\mathcal{O}(S)$. If $\mathcal{O}^{\ell o}$ is not mirror-symmetric, we call all its slots *low*. Low, middle, and high slots of $\mathcal{O}$ become children of the nodes $lo_\mathcal{O}$, $mi_\mathcal{O}$, and $hi_\mathcal{O}$, respectively. Note that, if $\mathcal{O}^{\ell o}$ is mirror-symmetric, the node names $lo_\mathcal{O}$ and $hi_\mathcal{O}$ are interchangeable.

Using these notions, we now formally define the tree representation of an interval hypergraph.

DEFINITION 4.1. *For a connected interval hypergraph $\mathcal{H}$, its* tree representation *$\mathbb{T}(\mathcal{H})$ is defined by*

$$\begin{aligned}
V(\mathbb{T}(\mathcal{H})) = {} & \{\mathcal{O}, lo_\mathcal{O}, mi_\mathcal{O}, hi_\mathcal{O} \mid \mathcal{O} \text{ is an overlap component of } \mathcal{H}\} \\
& \cup \{S_\mathcal{O} \mid S \text{ is a slot of overlap component } \mathcal{O}\} \cup \mathrm{supp}(\mathcal{H}), \\
E(\mathbb{T}(\mathcal{H})) = {} & \{(\mathcal{O}, lo_\mathcal{O}), (\mathcal{O}, mi_\mathcal{O}), (\mathcal{O}, hi_\mathcal{O}) \mid \mathcal{O} \text{ is an overlap component of } \mathcal{H}\} \\
& \cup \{(lo_\mathcal{O}, S_\mathcal{O}), (mi_\mathcal{O}, S_\mathcal{O}), (hi_\mathcal{O}, S_\mathcal{O}) \mid S \text{ is a low/middle/high slot of } \mathcal{O}\} \\
& \cup \{(S_\mathcal{O}, \mathcal{O}') \mid \text{the overlap component } \mathcal{O}' \text{ is located at slot } S \text{ of } \mathcal{O}\} \\
& \cup \{(S_\mathcal{O}, v) \mid \text{the vertex } v \text{ is located at slot } S \text{ of overlap component } \mathcal{O}\}.
\end{aligned}$$

Furthermore, we define a coloring c of the component and the slot nodes by

$$c(\mathcal{O}) = \mathcal{O}^{\ell o},$$

$$c(S_\mathcal{O}) = \begin{cases} \mathrm{lpos}_\mathcal{O}(S) & \text{if } S \text{ is a low or a middle slot of } \mathcal{O}, \\ \mathrm{rpos}_\mathcal{O}(S) & \text{if } S \text{ is a high slot of } \mathcal{O}. \end{cases}$$

See Figure 4 for an example of a tree representation.

Note that, for an overlap component $\mathcal{O}$ with symmetric interval representation, the order $\prec_\mathcal{O}$ is defined only up to reflection, so $\mathrm{lpos}_\mathcal{O}$ and $\mathrm{rpos}_\mathcal{O}$ can exchange their values depending on the particular choice of $\prec_\mathcal{O}$. These choices influence the construction of the tree. However, all possible $\mathbb{T}(\mathcal{H})$ are isomorphic, as the *lo* and *hi* nodes can be exchanged and the colors of the slots stay the same.

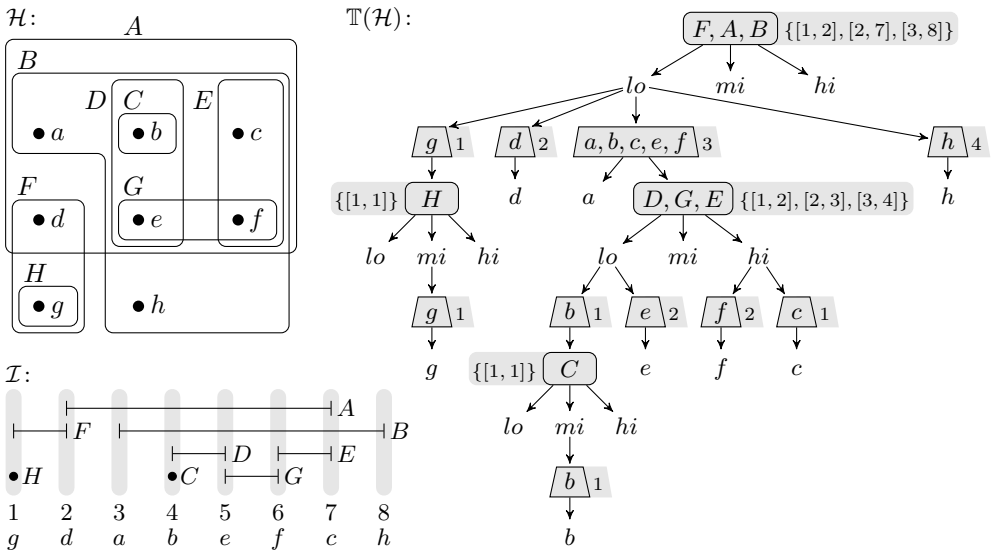

FIG. 4. *An interval hypergraph $\mathcal{H}$ and the corresponding tree representation $\mathbb{T}(\mathcal{H})$. Gray areas in $\mathbb{T}(\mathcal{H})$ indicate the color of* overlap components *and their* slots*. An overlap component $\mathcal{O}$ is represented by listing the hyperedges in $\mathcal{O}$ (sorted, for the reader's convenience, by their corresponding intervals in the canon of $\mathcal{O}$). We omit the references from slot and connector nodes to their overlap components as they are understood from the tree structure. The interval system $\mathcal{I} \cong \mathcal{H}$ can be derived from $\mathbb{T}(\mathcal{H})$ by reading the vertex nodes from left to right (see section* 4.2 *for details).*

Our goal is to compute a canonical interval labeling of $\mathcal{H}$ using a modified version of Lindell's canonization algorithm for trees [Lin92] on $\mathbb{T}(\mathcal{H})$. For this approach, it is necessary that $\mathbb{T}(\mathcal{H})$ is computable in logspace and that isomorphic interval hypergraphs have isomorphic tree representations.

LEMMA 4.2. *For a given interval hypergraph $\mathcal{H}$, its tree representation $\mathbb{T}(\mathcal{H})$ can be computed in* FL.

Proof. Each overlap component $\mathcal{O}$ of $\mathcal{H}$ can be identified in FL by running Reingold's algorithm [Rei08] on $\mathbb{O}(\mathcal{H})$ and represented by any hyperedge $B \in \mathcal{O}$. Moreover, $\mathrm{supp}(\mathcal{O})$ and the slots of $\mathcal{O}$ are easy to compute. Notice that a slot S of $\mathcal{O}$ can be represented by a tuple (u, B), where u is any vertex contained in S. Computability of the relation "located at" follows from its definition. Lemma 3.3 allows us to compute $\prec_{\mathcal{O}}$ and $\ell_{\mathcal{O}}$ for an overlap component $\mathcal{O}$ in logspace. Using $\prec_{\mathcal{O}}$, we can enumerate all slots and compute $\mathrm{lpos}_{\mathcal{O}}(S)$ and $\mathrm{rpos}_{\mathcal{O}}(S)$. With this information, the adjacency relation of $\mathbb{T}(\mathcal{H})$ and the vertex coloring can easily be constructed in logspace. ∎

LEMMA 4.3. *If $\mathcal{H}$ and $\mathcal{K}$ are isomorphic connected interval hypergraphs, then $\mathbb{T}(\mathcal{H}) \cong \mathbb{T}(\mathcal{K})$.*

Proof. Given an isomorphism ϕ from $\mathcal{H}$ onto $\mathcal{K}$, we will define an isomorphism ϕ' from $\mathbb{T}(\mathcal{H})$ onto $\mathbb{T}(\mathcal{K})$. Given an overlap component $\mathcal{O} = \{B_1, \ldots, B_k\}$ of $\mathcal{H}$, let $\phi'(\mathcal{O}) = \{\phi(B_1), \ldots, \phi(B_k)\}$. This is an overlap component of $\mathcal{K}$ isomorphic to $\mathcal{O}$; hence the nodes $\mathcal{O}$ and $\phi'(\mathcal{O})$ get the same color in the trees $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$. For any slot S of $\mathcal{O}$, we define ϕ' on the slot node $S_{\mathcal{O}}$ by $\phi'(S_{\mathcal{O}}) = \phi(S)_{\phi'(\mathcal{O})}$. The set $\phi(S)$ is a slot of $\phi'(\mathcal{O})$ because ϕ induces an isomorphism from $\mathcal{O}$ onto $\phi'(\mathcal{O})$. By the same reasoning, $S_{\mathcal{O}}$ and $\phi'(S_{\mathcal{O}})$ are equally colored in $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$. If $\mathcal{O}$ has an asymmetric canonical representation, let ϕ' take the *lo*, *mi*, and *hi* children of

$\mathcal{O}$ to the lo, mi, and hi children of $\phi'(\mathcal{O})$, respectively; otherwise ϕ' may swap the lo and the hi nodes in order to obey adjacency. Finally, we let $\phi'(v) = \phi(v)$ for all $v \in \mathrm{supp}(\mathcal{H})$, certainly respecting adjacency in the trees $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$. □

4.2. Computing a canonical interval labeling. Our construction of a canonical interval labeling for $\mathcal{H}$ is based on a traversal of $\mathbb{T}(\mathcal{H})$ such that the vertex nodes of $\mathbb{T}(\mathcal{H})$ are visited exactly in the order in which the corresponding vertices will appear in the resulting canonical interval representation of $\mathcal{H}$. The tree representation (viewed as a PQ-tree) already restricts the possible traversals to those that visit the slots of each overlap component $\mathcal{O}$ in either ascending or descending order of their position in $\mathcal{O}^{\ell\circ}$, where the latter is possible only if $\mathcal{O}^{\ell\circ}$ is symmetric. Thus, there remains a freedom to choose one of two possible orientations for each symmetric $\mathcal{O}^{\ell\circ}$. Also, for each slot S of an overlap component of $\mathcal{H}$, there is a freedom to choose the order of appearance of the vertices and the overlap components located at S. To come up with a canonical choice between the remaining possibilities, we employ a generalization of Lindell's tree canonization algorithm [Lin92].

LEMMA 4.4. *Lindell's algorithm* [Lin92] *can be extended to colored trees and to output not only a canonical form but also a canonical labeling. This modification preserves the logarithmic space bound.*

Proof sketch. Colors can be handled by extending the *tree isomorphism order* defined in [Lin92] by using $color(s) < color(t)$ as an additional condition (where s and t are the roots of the trees to compare). For concreteness, we give this additional condition the highest priority. The canonical labeling can be computed by using a counter i initialized to 0: Instead of printing (the opening parenthesis of) the canon of a node v, the modified algorithm increments i and prints "$v \mapsto i$." □

We recall the logspace traversal algorithm for trees with a linear order among siblings (as used in, e.g., Lindell's canonization). For a given node, it is possible in logspace to (1) go to its first child, (2) go to its next sibling, and (3) go to its parent. These operations allow a depth-first traversal of the tree where only the current node and whether we arrived there from its last child must be remembered. The only requirement is that the order among siblings can be evaluated in logspace. In our traversal of $\mathbb{T}(\mathcal{H})$ we use the following order, where $\lambda = \lambda_{\mathbb{T}(\mathcal{H})}$ is the canonical labeling of $\mathbb{T}(\mathcal{H})$ as obtained by Lemma 4.4.

- The children of an overlap component node $\mathcal{O}$ are ordered $lo_{\mathcal{O}} < mi_{\mathcal{O}} < hi_{\mathcal{O}}$ if $\mathcal{O}^{\ell\circ}$ is not mirror-symmetric or if $\lambda(lo_{\mathcal{O}}) < \lambda(hi_{\mathcal{O}})$; otherwise they are ordered $hi_{\mathcal{O}} < mi_{\mathcal{O}} < lo_{\mathcal{O}}$.
- The children of the first child of an overlap component node $\mathcal{O}$ (this can be either $lo_{\mathcal{O}}$ or $hi_{\mathcal{O}}$) are visited in ascending order of their colors; note that these colors are all distinct.
- The children of the last child of an overlap component node $\mathcal{O}$ (this can be either $hi_{\mathcal{O}}$ or $lo_{\mathcal{O}}$) are visited in descending order of their colors; again, these are all distinct.
- A $mi_{\mathcal{O}}$ node can have at most one child, which does not need ordering.
- The children of a slot node are ordered by the label assigned to them by λ.

Note that ordering $\mathbb{T}(\mathcal{H})$ in this way yields a permissible realization of the PQ-tree; i.e., the slots of each overlap component $\mathcal{O}$ are placed in ascending or descending order of their position in $\mathcal{O}^{\ell\circ}$. In the following, we exploit this to compute an interval labeling of the entire hypergraph $\mathcal{H}$.

Let $\phi\colon \mathrm{supp}(\mathcal{H}) \to [1, \|\mathrm{supp}(\mathcal{H})\|]$ number the vertices of $\mathcal{H}$ in the same order as their nodes are visited in the described traversal of $\mathbb{T}(\mathcal{H})$, and let $\ell_{\mathcal{H}}(B) = \{\phi(v) \mid v \in$

$B\}$ be the induced mapping on the hyperedges of $\mathcal{H}$. It is clear that both ϕ and $\ell_{\mathcal{H}}$ can be computed in logspace.

LEMMA 4.5. *$\ell_{\mathcal{H}}$ is an interval labeling of $\mathcal{H}$.*

Proof. We have to show for every hyperedge $B \in \mathcal{H}$ that $\ell_{\mathcal{H}}(B)$ is an interval. Let $B \in \mathcal{H}$ be any hyperedge and let $\mathcal{O}$ be the overlap component that contains B. We have already ensured that the slot nodes of $\mathcal{O}$ are visited in either ascending or descending order of their appearance in the interval representation $\mathcal{O}^{\ell_{\mathcal{O}}}$ of $\mathcal{O}$. This implies that the slots of $\mathcal{O}$ that are contained in B are visited consecutively, and with them also the vertices contained in them, as these are exactly the vertices contained in the subtrees rooted at these slot nodes. ∎

Now we are ready to prove our main result on interval hypergraphs.

THEOREM 4.6. *Given an interval hypergraph $\mathcal{H}$, a canonical interval labeling $\ell_{\mathcal{H}}$ for $\mathcal{H}$ can be computed in* FL.

Proof. Let $\ell_{\mathcal{H}}$ be the interval labeling of $\mathcal{H}$ defined above. By Lemma 4.4 the canonical labeling $\lambda_{\mathbb{T}(\mathcal{H})}$ of $\mathbb{T}(\mathcal{H})$ can be computed in logspace; the rest of the computation of $\ell_{\mathcal{H}}$ is possible in logspace as well.

It remains to show that the labelings $\ell_{\mathcal{H}}$ and $\ell_{\mathcal{K}}$ of isomorphic interval hypergraphs $\mathcal{H} \cong \mathcal{K}$ map these graphs to the same interval representation $\mathcal{H}^{\ell_{\mathcal{H}}} = \mathcal{K}^{\ell_{\mathcal{K}}}$. By Lemma 4.3, the colored trees $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$ are isomorphic. Hence, the canonical labelings $\lambda_{\mathbb{T}(\mathcal{H})}$ and $\lambda_{\mathbb{T}(\mathcal{K})}$ map these trees to the same colored tree $\mathbb{T}(\mathcal{H})^{\lambda_{\mathbb{T}(\mathcal{H})}} = \mathbb{T}(\mathcal{K})^{\lambda_{\mathbb{T}(\mathcal{K})}}$. Note that the interval representation $\mathcal{H}^{\ell_{\mathcal{H}}}$ depends only on the tree $\mathbb{T}(\mathcal{H})^{\lambda_{\mathbb{T}(\mathcal{H})}}$. It follows that $\mathcal{H}^{\ell_{\mathcal{H}}} = \mathcal{K}^{\ell_{\mathcal{K}}}$. ∎

5. Canonizing interval graphs and convex graphs. The reduction given by Lemma 2.4 transforms Theorem 4.6 into a result on interval graphs.

COROLLARY 5.1. *Given an interval graph G, a canonical interval labeling ℓ_G for G can be computed in* FL.

A bipartite graph G is called *convex* if one of its vertex classes can be linearly ordered so that the neighborhoods of the vertices in the other class are intervals with respect to this order. If both vertex classes have such orderings, G is called *biconvex*. The standard representation of hypergraphs as bipartite graphs transforms a hypergraph with vertex set V and hyperedge set $\mathcal{H}$ into the bipartite graph with vertex classes V and $\mathcal{H}$ where vertices $v \in V$ and $H \in \mathcal{H}$ are adjacent if and only if $v \in H$. Note that this transformation yields exactly the convex graphs when applied to interval hypergraphs.

COROLLARY 5.2. *Given a convex graph G, a canonical labeling ℓ_G for G can be computed in* FL.

Proof. The *open neighborhood* of a vertex v is defined by $N(v) = N[v] \setminus \{v\}$. Let G be a connected convex graph with vertex classes U and V. Reversing the aforementioned transformation of hypergraphs into bipartite graphs, we obtain two hypergraphs $\mathcal{H}_U = (U, \{N(v) \mid v \in V\})$ and $\mathcal{H}_V = (V, \{N(u) \mid u \in U\})$. Since G is convex, at least one of them is an interval hypergraph. Suppose that this is true for $\mathcal{H}_U$ and let ℓ be its canonical interval labeling. Sorting the labels $\ell(N(v))$, $v \in V$, we obtain a labeling of V. A labeling of U is obtained from the canonical ordering of slots of $\mathcal{H}_U$ determined by ℓ (as noted in the end of section 2.2). If $\mathcal{H}_V$ is an interval hypergraph too (i.e., G is biconvex), we compare the two labelings and, if they differ, choose the lexicographically smaller.

If G is disconnected, we split it into connected components using Reingold's algorithm [Rei08], canonize each of the components, and compose the total labeling of G componentwise in lexicographic order. ∎

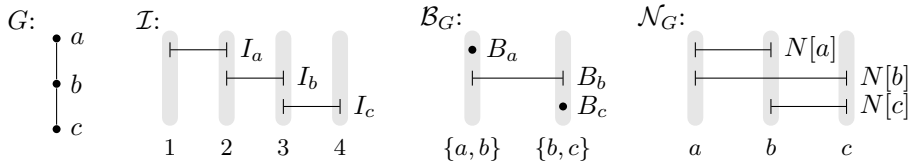

FIG. 5. *A proper interval graph G and a proper interval representation $\mathcal{I}$ of G. However, the bundle hypergraph $\mathcal{B}_G$ is not proper. The neighborhood hypergraph $\mathcal{N}_G$ is neither proper nor an interval representation of G.*

6. Computing proper and unit interval representations. We first prove a useful fact on overlap-connected hypergraphs.

LEMMA 6.1. *Let $\mathcal{F}$ and $\mathcal{E}$ be overlap-connected hypergraphs with* $\operatorname{supp}(\mathcal{F}) = \operatorname{supp}(\mathcal{E})$*, each containing at least two hyperedges. Then their union $\mathcal{F} \cup \mathcal{E}$ is overlap-connected, too.*

Proof. Choose $F \in \mathcal{F}$ and $E \in \mathcal{E}$ with nonempty intersection. If $F \between E$ or $F = E$, the claim is obviously true. Otherwise, suppose that $E \subset F$. Note that $F \neq \operatorname{supp}(\mathcal{F})$ because F is not the only hyperedge in $\mathcal{F}$ and $\mathbb{O}(\mathcal{F})$ is connected. Let $x \in \operatorname{supp}(\mathcal{F}) \setminus F$. Since $\operatorname{supp}(\mathcal{F}) = \operatorname{supp}(\mathcal{E})$, there is a hyperedge $E' \in \mathcal{E}$ containing x. By the connectedness of $\mathbb{O}(\mathcal{E})$, there is an $\between$-path $E_1 \between E_2 \between \ldots \between E_l$ connecting $E = E_1$ and $E' = E_l$. Let $m < l$ be the largest index such that $E_m \subseteq F$. Then $E_{m+1} \between F$. ∎

The set system $\mathcal{N}_G = \{N[v]\}_{v \in V(G)}$ is called the *(closed) neighborhood hypergraph* of the graph G. It is clear that $\mathcal{N}_G \cong \mathcal{N}_H$ whenever $G \cong H$. Harary and McKee [HM94] show that the converse is true if G is chordal.

An interval system $\mathcal{I}$ is *proper* if there is no inclusion $I \subseteq J$ between two intervals I and J in $\mathcal{I}$. An interval labeling $\ell\colon V(G) \to \mathcal{I}$ of a graph G is *proper* if the interval representation $\mathcal{I}$ is proper. Graphs admitting such labelings are called *proper interval graphs*. Let $\mathcal{I} = \{[a_i, b_i] \mid 1 \leq i \leq n\}$. In the absence of inclusions, the left endpoints a_i are pairwise distinct, and the same is true about the right endpoints b_i. Suppose that $a_i < a_{i+1}$ for all $i < n$; then we also have $b_i < b_{i+1}$. This yields a natural geometric order on $\mathcal{I}$.

Let $v_1, \ldots, v_n$ be the corresponding order on $V(G)$, that is, $\ell(v_i) = [a_i, b_i]$. Observe that, if v_i is adjacent to v_j with $j > i$, then v_i is adjacent to v_k for all $i < k \leq j$. This implies that each $N[v_i]$ is an interval with respect to the introduced order; therefore, $\mathcal{N}_G$ is an interval hypergraph.[1] If combined with the aforementioned result of Harary and McKee and our Theorem 4.6, this immediately gives us a logspace computable complete invariant for proper interval graphs: G is isomorphic to another graph G' exactly when the canonical interval representations of $\mathcal{N}_G$ and $\mathcal{N}_{G'}$ are equal. Note that an interval representation of the hypergraph $\mathcal{N}_G$ is generally not an interval representation of the graph G. Note also that the minimal interval representation constructed from the maxclique bundle hypergraph $\mathcal{B}_G$ (as in Corollary 5.1) is not proper if the graph contains at least one edge; see Figure 5 for an example. Now our aim is to come up with a canonical proper interval representation of a given proper interval graph.

It is easy to see that a proper interval graph with n vertices always has a

[1] The converse is also true: If $\mathcal{N}_G$ is an interval hypergraph, then G is a proper interval graph (see [Rob69, Duc84]).

proper interval representation $\mathcal{I} = \{[a_i, b_i] \mid 1 \le i \le n\}$ where $\{a_i, b_i \mid 1 \le i \le n\} = \{1, 2, \ldots, 2n\}$. From now on we will consider only such representations. Together with a proper interval labeling $\ell\colon V(G) \to \mathcal{I}$, the graph G has also the reversed proper interval labeling $\ell^*\colon V(G) \to \mathcal{I}^*$ with $\ell^*(v_i) = r([a_i, b_i])$, where $r(x) = 2n + 1 - x$. Under this interval labeling, the vertices of G appear as intervals in the reversed order $v_n, \ldots, v_1$. The first, graph-theoretic part of the following lemma is a version of a result by Deng, Hell, and Huang [DHH96, Corollary 2.5]. We state it in another form and prove it by a different method, which allows us to obtain a logspace computability result also.

LEMMA 6.2. *Let G be a connected proper interval graph with no twins. Then, up to reflection, G has a unique proper interval labeling. The latter is computable in logspace.*

Proof. Call a vertex u of G *universal* if $N[u] = V(G)$. Since universal vertices are twins, G can have at most one such vertex.

Let $\ell\colon V(G) \to \mathcal{I}$ be a proper interval labeling of G. Let $v_1, \ldots, v_n$ be the associated geometric ordering of the vertices of G. Denote the corresponding strict order on $V(G)$ by $\prec_\ell$. As was mentioned, $\prec_\ell$ defines an interval representation of the neighborhood hypergraph $\mathcal{N}_G$.

Given a nonuniversal vertex v_i, let $s = s(i)$ be the largest index such that $s < i$ and $v_s \notin N[v_i]$, and, similarly, let $t = t(i)$ be the smallest index such that $t > i$ and $v_t \notin N[v_i]$. At least one of these indices is well defined. Note that $N[v_i] \between N[v_s]$. Indeed, $v_s \notin N[v_i]$, $v_i \notin N[v_s]$, and v_{s+1} belongs to both sets (v_s and v_{s+1} are adjacent because G is connected). Similarly, we have $N[v_i] \between N[v_t]$. Note that neither v_s nor v_t is universal. It follows that, for any nonuniversal v_i, there is a subsequence of indices $i_1, \ldots, i_k$ containing i such that

$$N[v_{i_1}] \between N[v_{i_2}] \between \cdots \between N[v_{i_k}] \quad \text{and} \quad N[v_{i_1}] \cup N[v_{i_2}] \cup \cdots \cup N[v_{i_k}] = V(G).$$

If there is a universal vertex u, remove $N[u]$ from $\mathcal{N}_G$ and denote the modified hypergraph by $\mathcal{N}'_G$. By Lemma 6.1 we conclude that $\mathcal{N}'_G$ is overlap-connected. By Lemmas 2.1 and 3.2 there is a canonical pair of mutually reversed strict orders $\prec$, $\prec^*$ on the slots of $\mathcal{N}'_G$ such that the slots appear according to one of these orders in any interval labeling of the hypergraph.

Since G has no twins, the slots of $\mathcal{N}_G$ are singletons $\{v_1\}, \ldots, \{v_n\}$. Note that $\mathcal{N}'_G$ has all the same slots. Thus, $\prec$ and $\prec^*$ can be considered orders on $V(G)$, and one of them must coincide with $\prec_\ell$. To prove the uniqueness result, it now suffices to notice that $\prec_\ell$, i.e., the sequence $v_1, \ldots, v_n$, uniquely determines ℓ. Indeed, we must have

$$\begin{aligned} a_i &= i + \|\{j < i \mid v_j \text{ is nonadjacent to } v_i\}\| \\ \text{and } \ b_i &= a_i + 1 + \deg(v_i). \end{aligned} \tag{6.1}$$

The computability result readily follows by (6.1) from the logspace computability of the sequence $v_1, \ldots, v_n$ and/or its reversal; see Lemma 3.3. ∎

THEOREM 6.3. *Given a proper interval graph G, a canonical proper interval labeling ℓ_G for G can be computed in* FL.

Proof. Assume that G is connected. If G has no twins, Lemma 6.2 allows us to compute two mutually reversed proper interval labelings $\ell\colon V(G) \to \mathcal{I}$ and $\ell^*\colon V(G) \to \mathcal{I}^*$. We choose ℓ as canonical if $\mathcal{I} < \mathcal{I}^*$ (the order on interval systems is defined in section 2.1); otherwise ℓ^* is chosen (if $\mathcal{I} = \mathcal{I}^*$, either choice is good).

If G has twins, we still have a canonical pair ℓ, ℓ^* which is unique up to interchanging labels within a twin class. In order to compute this pair, we replace each twin class by a single representative, obtaining a twins-free quotient graph G'. As in the proof of Lemma 6.2, we compute the proper ordering $v'_1, \ldots, v'_{n'}$ on $V(G')$ (unique up to reflection). Further, we expand this sequence by substituting each v'_i with all its twins, obtaining an ordering $v_1, \ldots, v_n$ of $V(G)$. Finally, the intervals $\ell(v_i) = [a_i, b_i]$ are computed accordingly to (6.1). Another candidate is $\ell^* = r \circ \ell$; we choose one of the two which gives a $<$-least interval representation.

If G is disconnected, we split it into connected components $G_1, \ldots, G_k$ using Reingold's reachability algorithm. For each of them, we compute the canonical labeling $\ell_{G_j} \colon V(G_j) \to \mathcal{I}_j$ and sort out the interval representations $\mathcal{I}_1, \ldots, \mathcal{I}_k$. Then we merge the labelings ℓ_{G_j} into an integrated labeling $\ell_G \colon V(G) \to \mathcal{I}$ so that the supports of the interval representations $\mathcal{I}_j$ appear in $\mathrm{supp}(\mathcal{I})$ according to the established order. □

Note that both the linear-time [DHH96, HSS01] and the AC^2 [BHI07] representation algorithms for proper interval graphs are based on computing the canonical order of vertices of the input graph as in the proof of Lemma 6.2 (as we already mentioned, this lemma is proved in [DHH96] in a different language and by a different argument).

Finally, let us turn to the task of finding a canonical unit interval labeling for a given proper interval graph. A graph is a unit interval graph if it has an interval model over rationals in which every interval has unit length. It is well known that the class of proper interval graphs is equal to the class of unit interval graphs [Rob69].

Given a unit interval graph $G = (V, E)$, let ℓ_G be the canonical proper interval labeling for G as in Theorem 6.3. We assume that G is connected; if it is not, then we deal with the connected components individually and piece them back together in the end. Let $v_1, \ldots, v_n$ be the vertices of G in the ordering induced by ℓ_G. For every vertex $v \neq v_1$, let l_v be the least neighbor of v in this ordering. The edge (l_v, v) is called *principal edge at* v. It is easy to see that the set of principal edges forms a directed tree T on V that is rooted at v_1. We order the children of each vertex in T according to the above ordering.

For a vertex v, let $k(v)$ be the level at which v is located in T, and let $p(v)$ be the number assigned to v in the postorder traversal of T. Since T is constructible in logspace, both values can be computed in FL for any $v \in V$. Assign to each vertex v the value $v^L = k(v) + p(v)/n$. Corneil et al. show in [CKN+95, Theorem 3.2] that assigning $[v_i^L, v_i^L + 1]$ to v_i for every $i \in [1, n]$ yields a unit interval labeling for G. Since we started with a canonical proper interval representation of G and the procedure does not involve any arbitrary choices, we obtain a canonical unit interval labeling for G in FL.

7. Completeness results. Being able to compute canonical (interval) labelings for a (hyper)graph class in FL immediately implies that the isomorphism problem of that class is in L. Thus the isomorphism problem of interval hypergraphs, interval graphs, and convex graphs is in L by Theorem 4.6, Corollary 5.1, and Corollary 5.2. Moreover, there is a standard Turing reduction of the automorphism group problem (i.e., computing a generating set of the automorphism group of a given graph) to the search version of graph isomorphism for colored graphs (cf. [Hof82, KST93]). It is not hard to see that this reduction can be performed in logspace. We obtain the following result for interval graphs.

COROLLARY 7.1. *Computing a generating set of the automorphism group of a given interval graph, and hence computing a canonical labeling coset for a given*

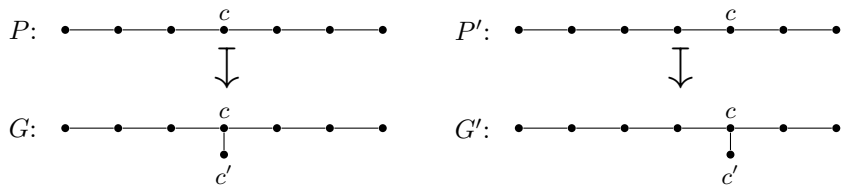

FIG. 6. *The reduction from* PATHCENTER *to the automorphism problem of caterpillars.*

interval graph, is in FL. *Further, the automorphism problem (i.e., deciding if a given graph has a nontrivial automorphism) for interval graphs is in* L. *The same holds for interval hypergraphs and convex graphs.*

In this section, we additionally prove the hardness of these problems for L. The hardness results are under DLOGTIME-uniform AC^0 reductions.

THEOREM 7.2. *The isomorphism and automorphism problems of interval graphs, bipartite permutation graphs, biconvex graphs, and convex graphs are* L*-complete.*

Bipartite permutation graphs are a subclass of biconvex graphs, which are in turn a subclass of convex graphs; so logspace algorithms for these classes are already presented in the previous sections. To prove hardness, we show that the isomorphism and automorphism problems are L-hard even for caterpillars, a subclass of the mentioned graph classes. Caterpillars are trees that become paths when all leaves are removed.

LEMMA 7.3. *Given a caterpillar G, it is* L*-hard to decide if G has a nontrivial automorphism.*

Proof. We reduce from the L-complete problem PATHCENTER (cf. [KK09]): Given an undirected path P of odd length and a vertex $c \in V(P)$, decide if c is the center node of P. We can assume that c has distance at least two to both ends; otherwise the instance can be trivially decided. We use the reduction $(P, c) \mapsto G$, where G is the graph that is obtained from P by adding a new vertex c' and an edge $\{c, c'\}$; see Figure 6 for an example. It is clear that G is a caterpillar. If c is the center of P, then G has the nontrivial automorphism that reflects P and maps c' to itself. If c is not the center of P, consider any automorphism φ of G. It must map c to itself as it is the only vertex of degree 3. Also, it must map c' and the ends of p to themselves, as they are the only vertices of degree 1 and all of them have a different distance to c. This implies that the other vertices are fixed as well, so φ must be the identity. ∎

The following lemma can be proved using a similar construction, which also marks either of the two end vertices of P (using two additional vertices—see Figure 7) in G_1 and G_2, respectively.

LEMMA 7.4. *Given two caterpillars G_1 and G_2, it is* L*-hard to decide if G_1 and G_2 are isomorphic.*

We observe that these constructions can be modified to yield proper interval graphs: For this, we add edges from the newly introduced marker vertices to all neighbors of the corresponding marked vertex (including other marker vertices).

THEOREM 7.5. *The automorphism and isomorphism problems of proper interval graphs are* L*-complete.*

Another modification of these constructions can be used to show that the automorphism and isomorphism problems of interval hypergraphs are hard for L: Paths can also be viewed as 2-uniform hypergraphs (i.e., hypergraphs with all hyperedges of size 2). The only difference is that no new edges are added, but the edges incident to the vertex that is to be marked are extended to also include the marker vertices.

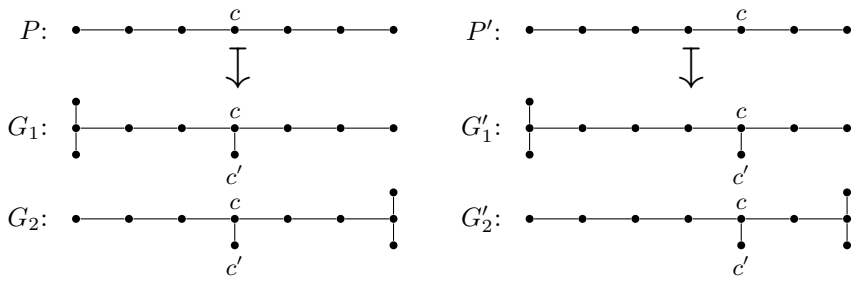

FIG. 7. *The reduction from* PATHCENTER *to the isomorphism problem of caterpillars.*

THEOREM 7.6. *The automorphism and isomorphism problems of interval hypergraphs are* L*-complete.*

We now turn to the recognition problems of the mentioned graph classes.

THEOREM 7.7. *The recognition problem is* L*-complete for*

- *interval graphs,*
- *proper interval graphs,*
- *convex graphs,*
- *biconvex graphs,*
- *bipartite permutation graphs,*
- *caterpillars,*
- *paths, and*
- *interval hypergraphs, respectively.*

Recognition of interval graphs in logspace follows from the results of Reif [Rei84] and Reingold [Rei08] (or as well from our Corollary 5.1). Each of the classes of convex, biconvex, and bipartite permutation graphs admits a characterization in terms of so-called asteroidal triples; see [BLS99, Proposition 6.2.1] and [Tuc72, Theorem 6]. This characterization enables recognition of each of the three classes in logspace by a simple reduction to the connectivity problem. Interval hypergraphs are also recognizable in logspace either by using a characterization by Duchet [Duc78] (or our Theorem 4.6). Proper interval graphs can be recognized in logspace using our results from section 6.

Proof. To prove the hardness, we give a reduction from the L-complete problem ORD such that positive instances are mapped to paths (which are included in all graph classes listed above and also are 2-uniform interval hypergraphs) and negative instances are mapped to graphs which are neither chordal nor bipartite (and thus are not in any of the listed classes).

ORD was proved to be L-complete by Etessami [Ete97] and can be described as follows: Given a directed path P and two vertices $s, t \in V(P)$, decide if there is a path from s to t, that is, if s is smaller than t in the order induced by the edge relation.

If P is such a directed path, it can be checked in AC^0 if t is among the first two vertices in the path. If so, output a trivial *yes-* or *no*-instance depending on whether s has been encountered first.

If t is not among the first 2 vertices of P, then we construct an undirected graph G from P as follows: For each vertex $v \in V(P)$ we insert a new vertex v' after v. Replace the incoming edge of s with (t', s) and replace the edge (t, t') with an edge that connects the first vertex in P and t. Finally, forget about the directions of the edges. Figure 8 shows an illustration of this construction. It is easy to verify that positive instances of ORD are mapped to paths, while the images of negative instances

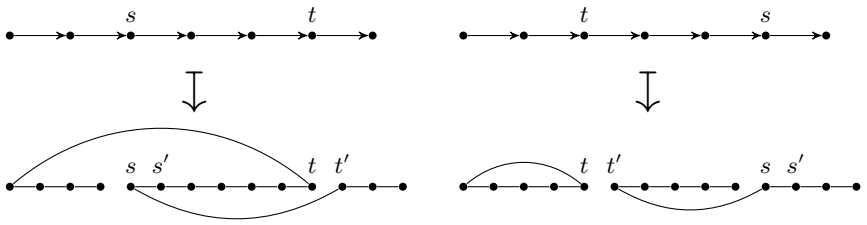

FIG. 8. *The reduction that maps positive instances of* ORD *to paths and negative instances to nonchordal nonbipartite graphs.*

contain a chordless circle of odd length at least 5. □

Remark 7.8. Using Reingold's undirected graph reachability algorithm [Rei08], it can be shown that recognition of bipartite graphs is in L. Together with a result by Reif [Rei84], Reingold's algorithm also implies that recognition of chordal graphs can be done in logspace. Thus, the proof of Theorem 7.7 also implies L-completeness of the recognition problems for these two graph classes.

We also remark that an interval graph is proper if and only if it has no induced copy of $K_{1,3}$ [Rob69] and that this property can be tested in AC^0.

COROLLARY 7.9. *The problems of computing a perfect elimination order (peo) and an interval labeling for a given interval graph G are in* FL *and are logspace hard.*

Proof. An interval labeling can be constructed in logspace by Corollary 5.1. Any interval labeling induces an ordering of $V(G)$; it is not hard to see that such an ordering is a peo. On the other hand, computing a peo is logspace hard even for paths by a reduction from ORD [KK09]. □

8. Conclusion. We have proved that canonical interval labelings of interval graphs and interval hypergraphs can be computed in logarithmic space. In particular, this puts into FL the problems of deciding graph isomorphism and finding a generating set of the automorphism group of interval graphs, interval hypergraphs, and convex graphs. We also gave logspace algorithms to compute interval representations of interval graphs and interval hypergraphs, and proper and unit interval representations of proper interval graphs, placing the recognition of proper interval graphs in L as well. Finally, we showed L-hardness of the recognition, isomorphism, and automorphism problems of these graph classes and concluded that all these problems are in fact L-complete.

Our canonization techniques can be used to show that each interval graph can be succinctly defined in first-order logic with counting quantifiers, where the vocabulary consists of the adjacency and the equality relations on the vertex set. By the results of Laubner [Lau10], this is possible in a logic with a bounded number of first-order variables. Cai, Fürer, and Immerman [CFI92] established a general connection between definability of graphs and solvability of the isomorphism problem by the multidimensional Weisfeiler–Lehman algorithm. As a consequence, there is a constant k such that the k-dimensional Weisfeiler–Lehman algorithm correctly decides isomorphism of two interval graphs (in time $O(n^k)$). We are now able to show that interval graphs are definable in a finite-variable counting logic with logarithmic quantifier depth. By a result of Grohe and Verbitsky [GV06], this implies that a parallel version of the Weisfeiler–Lehman algorithm solves interval graph isomorphism in TC^1.

Going beyond interval graphs, there are several natural graph classes that sug-

gest an investigation into whether they can similarly be handled in L. For example, circular-arc graphs generalize interval graphs as intersection graphs of arcs on a circle. While circular-arc graphs can be recognized in linear time [McC03], the best known isomorphism test for this class [Hsu95] runs in time $O(mn)$, where m and n denote the number of edges and vertices in the input graph. Whereas intuition suggests a reduction of circular-arc graphs to interval graphs by "cutting open" the circle that carries the graph's circular-arc representation, all known algorithms require additional techniques that are fairly specific to circular-arc graphs. One of the obstacles is that maxcliques cannot be handled as easily as in Lemma 2.2, since there are possibly exponentially many of them.

Another generalization of interval graphs is the class of rooted directed path graphs, i.e., intersection graphs of paths in a rooted directed tree. For this class, only polynomial-time isomorphism algorithms are known [BPT96, EPT00]. While maxcliques in a rooted directed path graph can still be recognized in a way similar to that of this paper, the procedure for linearly ordering maxcliques as given in section 3 cannot be employed in the presence of tree nodes of degree more than 2.

In the above paragraph, it is important that trees are rooted and directed accordingly, as intersection graphs of paths in unrooted directed trees are isomorphism-complete [BPT96]. Also, two well-known extensions of interval graphs, chordal graphs and co-comparability graphs, are known to be isomorphism-complete [LB79]. The same is true for boxicity-2 graphs, the intersection graphs of axis-parallel boxes in the plane (cf. [Ueh08]). Another obstacle for efficiently computing a canonical (or even arbitrary) intersection model for boxicity-2 graphs is that recognition of this class is NP-hard [Kra94]. This applies also to disk graphs, the intersection graphs of plane disks [Kra96].

Finally, we would like to refer the reader to [Spi03] for further graph classes for which the precise complexity of recognition and isomorphism remains open.

Acknowledgment. We thank the anonymous referees for helpful comments and detailed suggestions on how to improve this paper.

REFERENCES

[AS98] F. Annexstein and R. Swaminathan, *On testing consecutive-ones property in parallel*, Discrete Appl. Math., 88 (1998), pp. 7–28.

[BPT96] L. Babel, I. N. Ponomarenko, and G. Tinhofer, *The isomorphism problem for directed path graphs and for rooted directed path graphs*, J. Algorithms, 21 (1996), pp. 542–564.

[BHI07] J. Bang-Jensen, J. Huang, and L. Ibarra, *Recognizing and representing proper interval graphs in parallel using merging and sorting*, Discrete Appl. Math., 155 (2007), pp. 442–456.

[BL76] K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[BLS99] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*, SIAM Monogr. Discrete Math. Appl. 3, SIAM, Philadelphia, 1999.

[CFI92] J. Cai, M. Fürer, and N. Immerman *An optimal lower bound on the number of variables for graph identification*, Combinatorica, 12 (1992), pp. 389–410.

[Che96] L. Chen, *Graph isomorphism and identification matrices: Parallel algorithms*, IEEE Trans. Parallel Distrib. Syst., 7 (1996), pp. 308–319.

[Che99] L. Chen, *Graph isomorphism and identification matrices: Sequential algorithms*, J. Comput. System Sci., 59 (1999), pp. 450–475.

[CKN+95] D. G. Corneil, H. Kim, S. Natarajan, S. Olariu, and A. P. Sprague, *Simple linear time recognition of unit interval graphs*, Inform. Process. Lett., 55 (1995), pp. 99–104.

[CY91] L. Chen and Y. Yesha, *Parallel recognition of the consecutive-ones property with applications*, J. Algorithms, 12 (1991), pp. 375–392.

[CY93] L. Chen and Y. Yesha, *Efficient parallel algorithms for bipartite permutation graphs*, Networks, 23 (1993), pp. 29–39.

[DHH96] X. Deng, P. Hell, and J. Huang, *Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM J. Comput., 25 (1996), pp. 390–403.

[DNT+09] S. Datta, P. Nimbhorkar, T. Thierauf, and F. Wagner, *Graph isomorphism for $K_{3,3}$-free and K_5-free graphs is in log-space*, in FSTTCS, Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2009, pp. 145–156.

[DLN+09] S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, and F. Wagner, *Planar graph isomorphism is in log-space*, in CCC, IEEE Computer Society, Washington, DC, 2009, pp. 203–214.

[Duc78] P. Duchet, *Propriété de helly et problèmes de représentation*, in Problèmes Combinatoires et Théorie des Graphes (Orsay, 1976), Colloq. Int. CNRS 260, CNRS, Paris, France, 1978, pp. 117–118.

[Duc84] P. Duchet, *Classical perfect graphs*, Ann. Discrete Math., 21 (1984), pp. 67–96.

[Ete97] K. Etessami, *Counting quantifiers, successor relations, and logarithmic space*, J. Comput. System Sci., 54 (1997), pp. 400–411.

[EPT00] S. Evdokimov, I. N. Ponomarenko, and G. Tinhofer *Forestal algebras and algebraic forests (on a new class of weakly compact graphs)*, Discrete Math., 225 (2000), pp. 149–172.

[FG65] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835–855.

[Gol04] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 2nd ed., Ann. Discrete Math. 57, Elsevier, New York, 2004.

[GV06] M. Grohe and O. Verbitsky, *Testing graph isomorphism in parallel by playing a game*, in ICALP, Springer, Berlin, 2006, pp. 3–14.

[HMP+00] M. Habib, R. M. McConnell, C. Paul, and L. Viennot, *Lex-BFS and partition refinement*, Theoret. Comput. Sci., 234 (2000), pp. 59–84.

[HM94] F. Harary and T. A. McKee, *The square of a chordal graph*, Discrete Math., 128 (1994), pp. 165–172.

[HM03] W.-L. Hsu and R. M. McConnell, *PC trees and circular-ones arrangements*, Theoret. Comput. Sci., 296 (2003), pp. 99–116

[HSS01] P. Hell, R. Shamir, and R. Sharan, *A fully dynamic algorithm for recognizing and representing proper interval graphs*, SIAM J. Comput., 31 (2001), pp. 289–305.

[Hof82] C. M. Hoffmann, *Group-Theoretic Algorithms and Graph Isomorphism*, Lecture Notes in Comput. Sci. 136, Springer, Berlin, 1982.

[Hsu95] W.-L. Hsu, *$O(M \cdot N)$ algorithms for the recognition and isomorphism problems on circular-arc graphs*, SIAM J. Comput., 24 (1995), pp. 411–439.

[HM99] W.-L. Hsu and T.-H. Ma, *Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs*, SIAM J. Comput., 28 (1999), pp. 1004–1020.

[Kle96] P. N. Klein, *Efficient parallel algorithms for chordal graphs*, SIAM J. Comput., 25 (1996), pp. 797–827.

[KK09] J. Köbler and S. Kuhnert, *The isomorphism problem for k-trees is complete for logspace*, in MFCS, Springer, Berlin, 2009, pp. 537–548.

[KR88] P. N. Klein and J. H. Reif, *An efficient parallel algorithm for planarity*, J. Comput. System Sci., 37 (1988), pp. 190–246.

[KST93] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem: Its Structural Complexity*, Progr. Theoret. Comput. Sci., Birkhäuser Boston, Boston, 1993.

[KVV85] D. Kozen, U. V. Vazirani, and V. V. Vazirani, *NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching*, in FSTTCS, Lecture Notes in Comput. Sci. 206, Springer, Berlin, 1985, pp. 496–503.

[Kra94] J. Kratochvíl, *A special planar satisfiability problem and a consequence of its NP–completeness*, Discrete Appl. Math., 52 (1994), pp. 233–252.

[Kra96] J. Kratochvíl, *Intersection graphs of noncrossing arc–connected sets in the plane*, in GD, Springer, Berlin, 1996, pp. 257–270.

[Lau10] B. Laubner, *Capturing polynomial time on interval graphs*, in LICS, IEEE Computer Society, Washington, DC, 2010, pp. 199–208.

[Lin92] S. Lindell, *A logspace algorithm for tree canonization*, in STOC, ACM, New York, 1992, pp. 400–404.

[LB79] G. S. Lueker and K. S. Booth, *A linear time algorithm for deciding interval graph isomorphism*, J. ACM, 26 (1979), pp. 183–195.

[McC03] R. M. McConnell, *Linear-time recognition of circular-arc graphs*, Algorithmica, 37 (2003), pp. 93–147.

[Rei84] J. H. Reif, *Symmetric complementation*, J. ACM, 31 (1984), pp. 401–421.

[Rei08] O. Reingold, *Undirected connectivity in log-space*, J. ACM, 55 (2008), pp. 17.1–17.24.

[Rob69] F. S. Roberts, *Indifference graphs*, in Proof Techniques in Graph Theory: Proceedings of the 2nd Annual Arbor Graph Theory Conference, Academic Press, New York, 1969, pp. 139–146.

[RTL76] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[Spi03] J. P. Spinrad, *Efficient Graph Representations*, Fields Inst. Monogr. 19, AMS, Providence, RI, 2003.

[Tuc72] A. Tucker, *A structure theorem for the consecutive* 1*'s property*, J. Combin. Theory Ser. B, 12 (1972), pp. 153–162.

[Ueh08] R. Uehara, *Simple geometrical intersection graphs*, in WALCOM, Lecture Notes in Comput. Sci. 4921, Springer, Berlin, 2008, pp. 25–33.

[YC96] C.-W. Yu and G.-H. Chen, *An efficient parallel recognition algorithm for bipartite-permutation graphs*, IEEE Trans. Parallel Distrib. Syst., 7 (1996), pp. 3–10.