Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 43 (2010)

Interval Graphs: Canonical Representations in Logspace

Johannes Köbler[1], Sebastian Kuhnert[1], Bastian Laubner[1], and Oleg Verbitsky[2]

[1] Humboldt-Universität zu Berlin
[2] Institute for Applied Problems of Mechanics and Mathematics, Ukrainian Academy of Sciences, Lviv. Supported in part by the Alexander von Humboldt foundation.
{koebler,kuhnert,laubner,verbitsk}@informatik.hu-berlin.de

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 a logspace algorithm computing their canonical representations by proper and by unit interval systems.

1 Introduction

There has been persistent interest in the algorithmic aspects of interval graphs in the past decades, also spurred by their applicability to DNA sequencing (cf. [ZSF+94]) and scheduling problems (cf. [Möh84]). 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 interval graph isomorphism algorithm [LB79]. These algorithms are based on a special data structure called *PQ-trees*. By pre-processing 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. A parallel AC^2 algorithm was 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 in the ordering 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.

Our methods are optimized for space complexity. As such, our exposition neither relies on computing the graph's peo, nor do we use 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

ISSN 1433-8092

of maximal cliques and a modular decomposition tree are definable in a certain logical formalism. This makes these objects tractable in logarithmic space and leads us to the first logspace algorithm for computing a canonical interval representation for any interval graph (note that recognition of interval graphs in logspace follows from the results of Reif [Rei84]). 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 and show how to compute their interval representations canonically using Reingold's algorithm [Rei08]. We color these components with their canonical interval representations and place them in a tree that allows us to combine the canonical interval representations of the components to one for the whole hypergraph. To achieve this, we apply Lindell's algorithm [Lin92] to the colored decomposition tree.

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].

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, Limaye, Nimbhorkar, Thierauf and Wagner 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 completeness result for interval graph isomorphism is the first for a natural class of graphs containing cliques of arbitrary size. In each of these cases, the isomorphism problem has a matching lower bound, i. e. it turns out to be logspace complete.

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 unit intervals over rationals. Any such graph is obviously a proper interval graph, and the converse is also true by a classical result of Roberts, see [BLS99, Theorem 4.3.3]. 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.

Organisation of the paper. Section 2 introduces some preliminaries, notably the decomposition of interval hypergraphs into overlap components. In Section 3 we show how to compute a canonical interval representation for a sin-

gle 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 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 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 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 nonnegative integers $\mathbb{N}_0$, using the standard notation $[a, b] = \{i \in \mathbb{N}_0 \mid a \leq i \leq b\}$. We say $[a_1, b_1] < [a_2, b_2]$ if $a_1 < a_2$ or if $a_1 = a_2$ and $b_1 < b_2$. For interval systems $\mathcal{I}$ and $\mathcal{J}$, we write $\mathcal{I} < \mathcal{J}$ if the smallest uncommon interval (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 $\mathbb{N}_0$ so that the order is preserved.

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

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

2.2 Hypergraphs

We only consider hypergraphs $(V, \mathcal{H})$ without isolated vertices, i. e., the vertex set V is exactly $\mathrm{supp}(\mathcal{H})$. 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. An isomorphism from a hypergraph $\mathcal{H}$ to a hypergraph $\mathcal{K}$ is a bijection $\phi : \mathrm{supp}(\mathcal{H}) \to \mathrm{supp}(\mathcal{K})$ such that

- $H \in \mathcal{H}$ iff $\phi(H) \in \mathcal{K}$ for every $H \subseteq \mathrm{supp}(\mathcal{H})$, 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 $\mathbb{O}(\mathcal{H})$. 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 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 2-uniform interval hypergraphs, as the latter correspond to systems of intervals containing exactly 2 points. This difference stems from the fact that intervals 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 $\ell_\phi(H) = \big\{\phi(x) \mid x \in H\big\}$. We call a function $\ell : \mathcal{H} \to \mathcal{I}$ an *interval labeling* of $\mathcal{H}$ if $\ell = \ell_\phi$ for some ϕ. For another hypergraph isomorphism ψ, 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 $\mathrm{supp}(\mathcal{I}) = [0, k]$, where $k = \|\mathrm{supp}(\mathcal{I})\| - 1$. The map $r(x) = k - x$ 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 $\mathcal{I}$. 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.*

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 0 (resp. $\|\mathrm{supp}(\mathcal{I})\| - 1$) will be called *leftmost* (resp. *rightmost*). Denote the longest leftmost (resp. rightmost) interval $I \in \mathcal{I}$ by L (resp. R). Note that $L \neq R$ for 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, for all $Y \in \mathcal{H}$ such that $Y \between X$, the overlaps $X \cap Y$ form a single inclusion chain. It is not hard to see that $\ell(X) \in \{L, R\}$ iff 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 $\mathrm{supp}(\mathcal{I}') = [0, \|\mathrm{supp}(\mathcal{H})\| - 1]$. As we just observed, the leftmost interval in $\mathcal{I}'$ equals either $\ell'(A)$ or $\ell'(B)$. Consider first the former case. We have $\ell'(A) = [0, \|A\| - 1] = L$, 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 $[\|A \setminus X\| - 1, \|A \cup X\| - 1]$. 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 iff 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 $\ell'(B) = [0, \|B\| - 1] = \ell^*(B)$ 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 will prove a constructive version of Lemma 2.1, namely Lemma 3.2, showing 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 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 on the left from $\phi(S')$ in $\mathbb{N}_0$. 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 Bundles of maxcliques

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 bundles in $\mathcal{B}_G$ as hyperedges. Note that for twins $u, v \in V(G)$, the bundles B_u and B_v are equal; in this case $\mathcal{B}_G$ has multiple hyperedges.

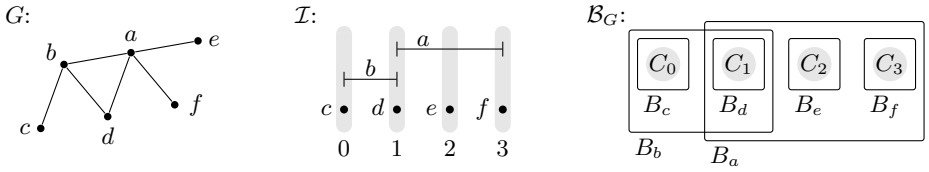

Fig. 1. An interval graph G, a minimal interval representation $\mathcal{I}$ of G, and the bundle hypergraph $\mathcal{B}_G$ of G. The maxcliques of G are $C_0 = \{b, c\}$, $C_1 = \{a, b, d\}$, $C_2 = \{a, e\}$, and $C_3 = \{a, f\}$.

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 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 only need to 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 $I_w \supseteq I_u \cap I_v$ for 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, $z \in C$. □

In any graph G, 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 (satisfying the first-order definable condition that $N[u, v]$ is a clique). A bundle B_v can be represented by the corresponding vertex v. The binary relations $B_u \subseteq B_v$ and $B_u \between B_v$ between bundles become first-order definable (in terms of the adjacency and equality relations on $V(G)$) and, therefore, decidable in logspace.

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 the bundle hypergraph $\mathcal{B}_G$ of an interval graph G is an interval hypergraph. Furthermore, $\mathcal{B}_G$ 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, $G \cong \mathbb{I}(\mathcal{B}_G)$ holds for any graph, not just for interval graphs).

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 $x_C \in \mathrm{supp}(\mathcal{I})$ (in fact, an arbitrary point in $I_u \cap I_v$) so that

$$w \in C \Leftrightarrow I_w \ni x_C. \tag{1}$$

Note that $x_C \neq x_{C'}$ if $C \neq C'$. Let $X = \{x_C \mid C \in M\}$ and $I'_w = I_w \cap X$. 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 $\mathrm{supp}(\mathcal{I}) = X$. Therefore, the correspondence $C \mapsto x_C$ is a bijection from M to $\mathrm{supp}(\mathcal{I})$. By (1), this is actually a hypergraph isomorphism from $\mathcal{B}_G$ to $\mathcal{I}$ (since the condition $w \in C$ can be rewritten as $C \in B_w$). □

2.4 An overview of our canonization algorithms

Given an interval labeling $\ell : \mathcal{H} \to \mathcal{I}$ of a hypergraph $\mathcal{H}$, we will denote the interval system $\mathcal{I}$ also by $\mathcal{H}^\ell$. By a *canonical interval labeling for hypergraphs* we mean an algorithm that, given an interval hypergraph $\mathcal{H}$, produces its interval labeling $\ell_\mathcal{H}$ so that $\mathcal{H}^{\ell_\mathcal{H}} = \mathcal{K}^{\ell_\mathcal{K}}$ whenever $\mathcal{H} \cong \mathcal{K}$. Lemma 2.3 reduces canonization of interval graphs to canonization of their bundle hypergraphs.

Lemma 2.4. *Computing a canonical interval labeling for an interval graph G is reducible in logspace to computing a canonical interval labeling of an interval hypergraph by setting $\ell_G(v) = \ell_{\mathcal{B}_G}(B_v)$.* □

Given an interval hypergraph $\mathcal{H}$, we start computing a canonical interval labeling $\ell_\mathcal{H}$ piecewise, finding first 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 the class of overlap-connected interval hypergraphs. Let $\mathcal{O}$ be such a hypergraph. We call two vertices $u, v \in \mathrm{supp}(\mathcal{O})$ *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 $\mathrm{supp}(\mathcal{O})$. 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 $[0, \|B\| - 1]$. So from now on we additionally assume that $\mathcal{O}$ consists of at least two hyperedges.

By Lemma 2.1 we know that $\mathcal{O}$ has a unique, up to reversing, interval labeling ℓ. Recall that an interval labeling of $\mathcal{O}$ uniquely determines the action of the underlying isomorphism on slots of $\mathcal{O}$ (see Section 2.2). The slots of $\mathcal{O}$ 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.

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 reversing. 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. We can identify 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 $\mathcal{O}$. (Other hyperedges are overlapped from both sides, yielding two inclusion chains, or covered by a larger hyperedge; otherwise $\mathcal{O}$ would not be overlap-connected.)

We observe that the side slot S_i at the side of B_i can be characterized by the following conditions:

1. $S_i \subseteq B_i$,
2. $\forall B \in \mathcal{O} : S_i \subseteq B \Rightarrow B$ is marginal $\wedge\, B \subseteq B_i$, and
3. $\|\{B \in \mathcal{O} \mid S_i \subseteq B\}\|$ is minimal.

Clearly, the construction is possible in logspace. To represent a slot S in logspace, we store a vertex $s \in S$; the other vertices in S can be easily computed since the relation $\sim_\mathcal{O}$ is decidable in logspace.

It remains to prove the correctness of this construction. It is clear that a side-slot S_i can only be contained in marginal hyperedges that are subsets of B_i. Denote the set of all hyperedges with these properties by $\mathcal{B}_i$. Let $U_i = \mathrm{supp}(\mathcal{O} \setminus \mathcal{B}_i)$. Note that $S_i \subset B_i \setminus U_i$. Let $\mathcal{B}'_i = \{B \in \mathcal{B}_i \mid B \not\subset U_i\}$. Thus, the inclusion $S_i \subset B$ is possible only for $B \in \mathcal{B}'_i$.

We claim that $\phi(U_i)$ is an interval, where ϕ is any isomorphism underlying the interval labeling ℓ. It is clear that $\phi(U_i)$ contains the interval $\mathrm{supp}(\mathcal{O}^\ell) \setminus \ell(B_i)$ and we have to show that $\phi(U_i) \cap \ell(B_i)$ is an interval too. Without loss of generality, suppose that $\ell(B_i)$ is leftmost. If every nonmarginal $A \subset B_i$ is included in some $B \between B_i$, then $\phi(U_i) \cap \ell(B_i)$ is equal to the intersection of the intervals $\ell(B_i)$ and $\bigcup_{B \between B_i} \ell(B)$. Otherwise, among nonmarginal subhyperedges of B_i that are not covered by any $B \between B_i$ we choose the hyperedge $A \subset B_i$ with $\ell(A)$ leftmost. If there are two or more candidates (with the same left end point), we choose the one with the longest $\ell(A)$. We are done if we show that there exists an overlap-chain of hyperedges $B_i \between A_1 \between A_2 \between \cdots \between A_k = A$ with all A_j's nonmarginal, possibly except A_1. For this role we consider a shortest overlap-path from B_i to A. Since $A_1 \between A_2 \between \cdots \between A_k$ has the smallest possible length k, we have $A_j \subset B_i$ for all $j > 1$. In particular, A_2 overlaps with A_1 on the left. It suffices to verify that this is so along all the path: A_{j+1} overlaps with A_j on the left for all $j < k$. Assume the contrary and set j $(1 < j < k)$ to be the smallest index such that A_{j+1} overlaps with A_j on the right. We must have $A_{j+1} \subset A_{j-1}$ for else the overlap-path $A_1 \between A_2 \between \cdots \between A_k$ could be shortened. Note that either $A_{j-1} \subset B_i$ is nonmarginal or $A_{j-1} = A_1 \between B_i$. By assumption, in any case $\ell(A)$ must be strictly on the left of $\ell(A_{j-1})$. Therefore, the path $A_{j+1} \between \cdots \between A_k$ should eventually go outside of A_{j-1} and can be at that point shortened. This contradiction proves the claim.

Denote the left endpoint of $\ell(A)$ by x_i. Thus, we have proved that $\phi(U_i)$ is the interval with one endpoint x_i and the other endpoint equal to the right endpoint of $\mathrm{supp}(\mathcal{O}^\ell)$. It is now not hard to see that $x_i \in \ell(B)$ for every $B \in \mathcal{B}'_i$. It follows that the intervals $\ell(B) \setminus U_i$ for $B \in \mathcal{B}'_i$ form an inclusion-chain. Therefore, the third condition in our construction uniquely determines the side-slot among the candidate slots $S \subseteq B_i \setminus U_i$ that remain after the first two conditions. An example can be found in Fig. 2. □

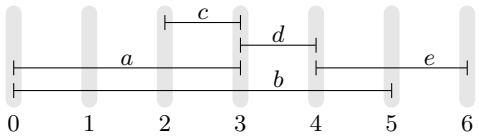

Fig. 2. Identification of the side-slots S_1 and S_2: The two inclusion-maximal marginal hyperedges are $B_1 = b$ and $B_2 = e$. We have $\mathcal{B}_1 = \{a, b, c\}$ and $\mathcal{B}_2 = \{e\}$, yielding $S_1 = \{0, 1\}$ and $S_2 = \{6\}$. Note that $x_1 = 3$.

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 $u, v \in B$, $w \notin B$:

$$u \prec_S w \Leftrightarrow v \prec_S w \qquad \text{and} \qquad w \prec_S u \Leftrightarrow w \prec_S v \tag{2}$$

$u \prec_S v$ 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 $u \prec v$ iff the interval $\phi([u]_{\sim_\mathcal{O}})$ lies strictly on 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 clearly holds. If $k \geq 1$, suppose w. l. o. g. 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 (2). If $v \in B_{k-1}$, then also $u \in B_{k-1}$ since we assumed the path to be shortest. Then 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 (2). □

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

$$\ell_S(B) = \big[\,\mathrm{pos}(B), \mathrm{pos}(B) + \|B\| - 1\big], \tag{3}$$

where a vertex $v \in \mathrm{supp}(\mathcal{O})$ has position $\mathrm{pos}(v) = \|\{u \in \mathrm{supp}(\mathcal{O}) \mid u \prec_S v\}\|$ 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 $\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$ is a complete invariant 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} by requiring that the set $\mathcal{O}^{\ell_\mathcal{O}} = \big\{\ell_\mathcal{O}(B) \mid B \in \mathcal{O}\big\}$ of intervals becomes minimal (if $\mathcal{O}^{\ell_{S_1}}$ is mirror-symmetric, we choose it arbitrarily). We denote the corresponding partial order on the vertices of $\mathcal{O}$ by $\prec_\mathcal{O}$. By $\vec{\mathcal{O}}$ we denote the list of hyperedges $B \in \mathcal{O}$, ordered by their intervals $\ell_\mathcal{O}(B)$ (we will need it in the next section).

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 $\mathcal{O}$,*
3. *computing the corresponding ordered list of hyperedges $\vec{\mathcal{O}}$, and*
4. *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:

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

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 $u \prec_S v$ iff 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) and to choose the left side-slot $S \in \{S_1, S_2\}$ so that $\mathcal{O}^{\ell_S} = \min\{\mathcal{O}^{\ell_{S_1}}, \mathcal{O}^{\ell_{S_2}}\}$. Furthermore, $\mathcal{O}^{\ell_{\mathcal{O}}}$ is mirror-symmetric iff both interval representations are equal.

Given $\ell_{\mathcal{O}}$, it is easy to sort the hyperedges in $\mathcal{O}$, and thereby compute $\vec{\mathcal{O}}$. □

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 $B_0 = \operatorname{supp}(\mathcal{H})$; it can be discarded once the canonical interval labeling is computed.

4.1 The tree representation

As noted before, the overlap components of $\mathcal{H}$ form a tree. We say that an overlap component $\mathcal{O}'$ is *located at* a slot S (of an overlap component $\mathcal{O}$), if $\operatorname{supp}(\mathcal{O}') \subseteq S$ and there is no intermediate overlap component $\mathcal{O}''$, i. e. $\operatorname{supp}(\mathcal{O}'') \subseteq S$ and $\mathcal{O}'$ is contained in some slot of $\mathcal{O}''$.

Our goal is to construct a tree representation $\mathbb{T}(\mathcal{H})$ of an interval hypergraph $\mathcal{H}$ such that $\mathcal{H}_1 \cong \mathcal{H}_2 \Leftrightarrow \mathbb{T}(\mathcal{H}_1) \cong \mathbb{T}(\mathcal{H}_2)$. To achieve this, we color the component nodes with their canonical interval representation and introduce slot nodes to make sure that overlap components that are located at the same slot are kept together in any isomorphic copy of the tree representation. However, this is not enough to ensure that isomorphic tree representations imply isomorphic hypergraphs, as can be seen in Fig. 3.

To make sure that isomorphic trees imply isomorphic hypergraphs, we constrain the order of the slots of an overlap component. For overlap components with mirror-symmetric interval representation we fix the slot order up to reversing; for asymmetric ones we fix it completely. The latter can be achieved by just using the position from the left as colors for the slots; in the former case we use

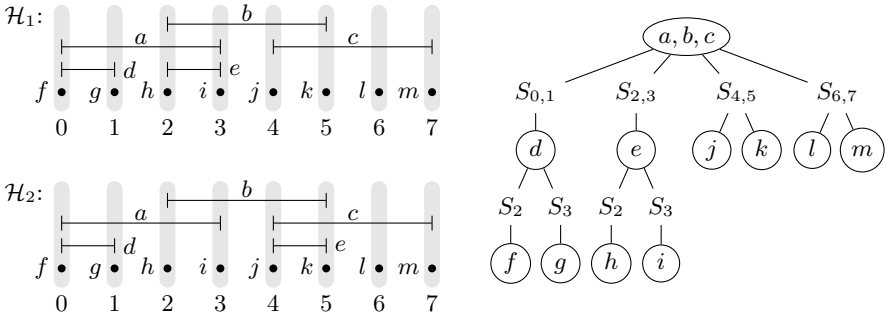

Fig. 3. The interval hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ are non-isomorphic. Yet they end up with isomorphic tree representations, if only component and slot nodes are used.

the position from the margin which is closer. Here, the position of a slot S in $\mathcal{O}$ from the left and from the right is determined by counting the number of slots that are placed by $\prec_{\mathcal{O}}$ to the left and to the right of S, respectively:

$$\begin{aligned}\mathrm{lpos}(S) &= \|\{S' \mid S' \text{ is a slot of } \mathcal{O} \text{ with } S' \prec_{\mathcal{O}} S\}\| \\ \mathrm{rpos}(S) &= \|\{S' \mid S' \text{ is a slot of } \mathcal{O} \text{ with } S \prec_{\mathcal{O}} S'\}\|\end{aligned}$$

Now we need to make sure that either all or none of the slots with the same color change their positions. To do so, we introduce an additional type of nodes in the tree to group the slots. If $\mathcal{O}$ has a mirror-symmetric canon $\mathcal{O}^{\ell o}$, we call a slot S of $\mathcal{O}$ *low* if $\mathrm{lpos}(S) < \mathrm{rpos}(S)$, *middle* if $\mathrm{lpos}(S) = \mathrm{rpos}(S)$, and *high* if $\mathrm{lpos}(S) > \mathrm{rpos}(S)$. If $\mathcal{O}^{\ell o}$ is not mirror-symmetric, we call all its slots *low*.

Using these notions, we now define a tree representation for interval hypergraphs.

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

$$\begin{aligned}V(\mathbb{T}(\mathcal{H})) =& \{\vec{\mathcal{O}}, lo_{\mathcal{O}}, mi_{\mathcal{O}}, hi_{\mathcal{O}} \mid \mathcal{O} \text{ is an overlap component of } \mathcal{H}\} \\ & \cup \{S \mid S \text{ is a slot of some overlap component } \mathcal{O} \text{ of } \mathcal{H}\} \\ E(\mathbb{T}(\mathcal{H})) =& \{(\vec{\mathcal{O}}, lo_{\mathcal{O}}), (\vec{\mathcal{O}}, mi_{\mathcal{O}}), (\vec{\mathcal{O}}, hi_{\mathcal{O}}) \mid \mathcal{O} \text{ is an overlap component of } \mathcal{H}\} \\ & \cup \{(lo_{\mathcal{O}}, S), (mi_{\mathcal{O}}, S), (hi_{\mathcal{O}}, S) \mid S \text{ is a low/middle/high slot in } \mathcal{O}\} \\ & \cup \{(S, \vec{\mathcal{O}}) \mid \text{the overlap component } \mathcal{O} \text{ is located at slot } S\}\end{aligned}$$

Further we define a coloring c of the component-nodes $\vec{\mathcal{O}}$ and slot-nodes S by

$$\begin{aligned}c(\vec{\mathcal{O}}) &= \mathcal{O}^{\ell o} \\ c(S) &= \begin{cases}\mathrm{lpos}(S) & \text{if } S \text{ is low or middle,} \\ \mathrm{rpos}(S) & \text{if } S \text{ is high.}\end{cases}\end{aligned}$$

As $\mathcal{H}$ is connected, there is an overlap component $\mathcal{O}_0$ with $\text{supp}(\mathcal{H}) = \text{supp}(\mathcal{O}_0)$. $\vec{\mathcal{O}}_0$ is the root of the directed tree $\mathbb{T}(\mathcal{H})$. See Fig. 4 for an example of a tree representation.

Note that, for an overlap component $\mathcal{O}$ with symmetric interval representation, the definition of $\prec_{\mathcal{O}}$ is only unique up to reversing, so lpos and rpos can exchange their values depending on the arbitrary choice of $\prec_{\mathcal{O}}$. These choices influence the construction of the tree. However, $\mathbb{T}(\mathcal{H})$ is unique up to isomorphism, as only the $lo_{\mathcal{O}}$ and $hi_{\mathcal{O}}$ nodes can be exchanged and the colors of the slots stay the same.

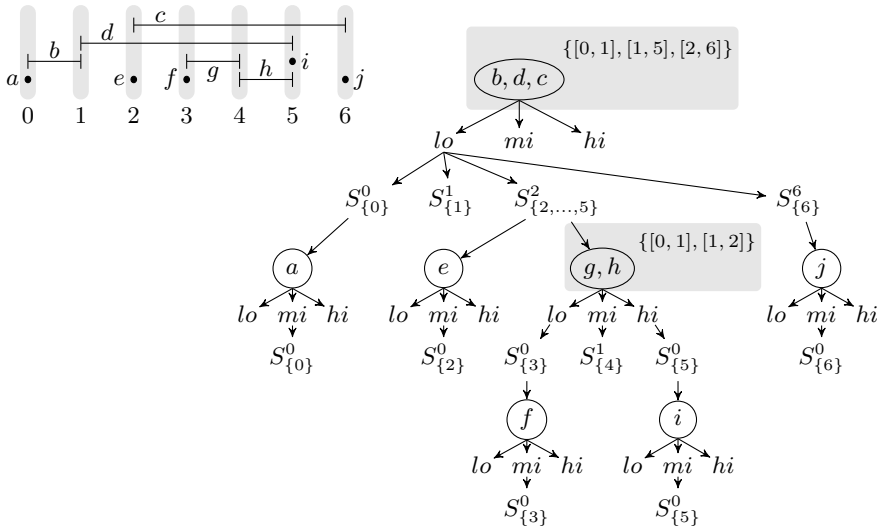

Fig. 4. An interval hypergraph $\mathcal{H}$ and the corresponding tree representation $\mathbb{T}(\mathcal{H})$. Gray rectangles in $\mathbb{T}(\mathcal{H})$ indicate the color of overlap components. Overlap components have color $\{[0, 0]\}$ where not indicated. Slot name $S^k_{\{i,\dots,j\}}$ denotes slot $\{i, \dots, j\}$ and indicates that its color is k. We omit the indices of the *lo*, *mi* and *hi* nodes as they are clear from the structure of the tree.

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, we must first compute $\mathbb{T}(\mathcal{H})$ in logspace.

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

Proof. Notice that a slot S of an overlap component $\mathcal{O}$ can be represented by a tuple (u, B), where u is any vertex contained in S and B is any hyperedge in $\mathcal{O}$. Lemma 3.3 allows us to compute $\vec{\mathcal{O}}$, $\prec_{\mathcal{O}}$ and $\ell_{\mathcal{O}}$ for an overlap component $\mathcal{O}$ in logspace. Using $\prec_{\mathcal{O}}$, we can enumerate all slots and compute $\text{lpos}(S)$ and $\text{rpos}(S)$. Using this information, $\mathbb{T}(\mathcal{H})$ can easily be constructed in logspace. □

We proceed to show a basic structural property of $\mathbb{T}(\mathcal{H})$ that we exploit in order to compute the canonical interval labeling. We call a slot S of an overlap component $\mathcal{O}$ *leaf-slot* if it occurs as a leaf-node in $\mathbb{T}(\mathcal{H})$. Note that S is a leaf-slot iff it contains no other overlap component.

Lemma 4.3. *Each slot of $\mathcal{H}$ occurs exactly once as leaf-slot in $\mathbb{T}(\mathcal{H})$.*

Proof. It is easy to see that every leaf-slot S is a slot of the hypergraph $\mathcal{H}$ and, vice versa, every slot of $\mathcal{H}$ is a leaf-slot of some overlap component. Note also that the same slot S of $\mathcal{H}$ cannot be a leaf-slot of different overlap components. Therefore, we have a one-to-one correspondence between the leaves of $\mathbb{T}(\mathcal{H})$ and the slots of $\mathcal{H}$. □

We will compute a canonical labeling of $\mathbb{T}(\mathcal{H})$ and call it $\ell_{\mathbb{T}(\mathcal{H})}$. For this, we observe 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 additional condition (where s and t are the roots of the trees to compare). 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, increment i and print "$v \mapsto i$". □

4.2 Computing a canonical interval labeling

Our aim is a traversal of $\mathbb{T}(\mathcal{H})$ that is left-to-right in the resulting canon. That is, we visit the leaf-slots in ascending order of the positions of their corresponding vertex sets in the computed canonical interval representation. To achieve this, we use the canonical labeling $\ell_{\mathbb{T}(\mathcal{H})}$ of $\mathbb{T}(\mathcal{H})$.

We first recall the *logspace tree traversal* that is used in Lindell's canonization algorithm to traverse a tree where the children of each node are linearly ordered. Only the current node must be remembered, because when given a 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. "First" and "next" can be respective to any order on the children of a node that can be evaluated in logspace. In our left-to-right traversal we use the following order:

- The children of an overlap component node $\vec{\mathcal{O}}$ are either ordered $lo_{\mathcal{O}} < mi_{\mathcal{O}} < hi_{\mathcal{O}}$ (if $\mathcal{O}^{\ell o}$ is not mirror-symmetric or if $\ell_{\mathbb{T}(\mathcal{H})}(lo_{\mathcal{O}}) < \ell_{\mathbb{T}(\mathcal{H})}(hi_{\mathcal{O}})$) or $hi_{\mathcal{O}} < mi_{\mathcal{O}} < lo_{\mathcal{O}}$ (otherwise).
- The children of the first child of an overlap component node $\vec{\mathcal{O}}$ (this can be either $lo_{\mathcal{O}}$ or $hi_{\mathcal{O}}$) are visited in ascending order of their colors.
- The children of the last child of an overlap component node $\vec{\mathcal{O}}$ (this can be either $hi_{\mathcal{O}}$ or $lo_{\mathcal{O}}$) are visited in descending order of their colors.

– The children of a slot node are ordered by the label assigned to them by $\ell_{\mathbb{T}(\mathcal{H})}$.

Note that the children of $lo_{\mathcal{O}}$ and $hi_{\mathcal{O}}$ all have different colors. Also, $mi_{\mathcal{O}}$ can have at most one child. All these conditions can be evaluated in logspace without using non-local information. Traversing $\mathbb{T}(\mathcal{H})$ in this order makes sure that the slots of an overlap component $\mathcal{O}$ are visited either in ascending or descending order of their positions. The latter case can only occur if $\mathcal{O}^{\ell_{\mathcal{O}}}$ is mirror-symmetric.

We complete the description of our algorithm by showing how, while processing $\mathbb{T}(\mathcal{H})$, a canonical interval labeling can be computed in logspace. Additionally to the current node we store a *current offset* o that equals to the number of vertices in the leaf-slots we have passed already. We initialize $o = 0$ and increment it whenever the logspace tree traversal passes a leaf-slot node by the size of that slot. Whenever we enter an overlap component node $\vec{\mathcal{O}} = (B_1, \dots, B_k)$ for the first time, we output the mappings $B_i \mapsto [l_i + o, r_i + o]$ where $[l_i, r_i]$ is the ith-smallest interval in $c(\vec{\mathcal{O}}) = \mathcal{O}^{\ell_{\mathcal{O}}}$ if $lo_{\mathcal{O}} < mi_{\mathcal{O}} < hi_{\mathcal{O}}$, and the ith-largest interval otherwise. In the first case this results in $\ell_{\mathcal{H}}(B) = \ell_{\mathcal{O}}(B) + o$. In the second case this association is mirrored: If $r \colon \mathcal{O} \to \mathcal{O}$ is the hypergraph isomorphism that reverses $\mathcal{O}$, then $\ell_{\mathcal{H}}(B) = \ell_{\mathcal{O}}(r(B)) + o$. After traversing all of $\mathbb{T}(\mathcal{H})$, we have output a mapping for each $B \in \mathcal{H}$. We call this mapping $\ell_{\mathcal{H}}$.

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

Proof. Take any two hyperedges $B, B' \in \mathcal{H}$. Let $\mathcal{O}$ and $\mathcal{O}'$ be the overlap components containing B and B', and let o and o' be the current offsets when $\mathcal{O}$ and $\mathcal{O}'$ are first entered, respectively. If $\mathcal{O} = \mathcal{O}'$, we are done because $\ell_{\mathcal{O}}$ is an interval labeling and the offset o preserves intersection.

If $\operatorname{supp}(\mathcal{O})$ and $\operatorname{supp}(\mathcal{O}')$ do not intersect, then B and B' are not adjacent. W. l. o. g. assume $o < o'$. Indeed we have $o + \|\operatorname{supp}(\mathcal{O})\| \leq o'$, as each vertex in $\operatorname{supp}(\mathcal{O})$ is contained in a leaf-slot below $\vec{\mathcal{O}}$ and the offset is advanced by one for each of these. As $\ell_{\mathcal{O}}(B) \subseteq [0, \|\operatorname{supp}(\mathcal{O})\| - 1]$ (see the definition of $\ell_{\mathcal{O}}$), $\ell_{\mathcal{H}}(B)$ and $\ell_{\mathcal{H}}(B')$ do not intersect.

If $\operatorname{supp}(\mathcal{O})$ and $\operatorname{supp}(\mathcal{O}')$ do intersect, one must be contained in the other. W. l. o. g. assume $\operatorname{supp}(\mathcal{O}') \subset \operatorname{supp}(\mathcal{O})$ and let S be the slot of $\mathcal{O}$ in which $\operatorname{supp}(\mathcal{O}')$ is contained. If $B \supseteq S$, then B and B' are adjacent. By Lemma 4.3 and the order of our tree traversal we have $\ell_{\mathcal{O}}(B) + o \supseteq [o', o' + \|\operatorname{supp}(\mathcal{O}')\| - 1]$. Finally, if $B \cap S = \emptyset$, then u and v are not adjacent. Also the leaf-slots corresponding to the maxcliques in B will be processed all before or all after $\vec{\mathcal{O}}'$, so $\ell_{\mathcal{H}}(B)$ and $\ell_{\mathcal{H}}(B')$ do not intersect. □

In the following we prove that the interval labeling $\ell_{\mathcal{H}}$ is canonical.

Lemma 4.6. *If $\mathcal{H}$ and $\mathcal{K}$ are isomorphic connected interval hypergraphs, then $\mathbb{T}(\mathcal{H}) \cong \mathbb{T}(\mathcal{K})$.*

Proof. Since any isomorphism ϕ between $\mathcal{H}$ and $\mathcal{K}$ induces a unique mapping of the overlap components $\mathcal{O}$ of $\mathcal{H}$ to isomorphic overlap components $\mathcal{O}'$ of $\mathcal{K}$, it is clear how to define an isomorphism ϕ' between $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$ on the

component-nodes $\vec{\mathcal{O}}$ of $\mathbb{T}(\mathcal{H})$. Further, since the canonical interval representations $\mathcal{O}^{\ell_\mathcal{O}}$ and $\mathcal{O}'^{\ell_{\mathcal{O}'}}$ coincide, $\vec{\mathcal{O}}$ and $\phi'(\vec{\mathcal{O}})$ indeed have the same colors.

In order to define ϕ' on the *lo*, *mi* and *hi* nodes of $\mathbb{T}(\mathcal{H})$, consider a component-node $\vec{\mathcal{O}} = (B_1, \ldots, B_k)$ of $\mathbb{T}(\mathcal{H})$. If $\mathcal{O}^{\ell_\mathcal{O}}$ is not mirror-symmetric, then it follows that $\vec{\mathcal{O}}' = (\phi(B_1), \ldots, \phi(B_k))$. Otherwise, it is also possible that $\vec{\mathcal{O}}' = (\phi(B_k), \ldots, \phi(B_1))$. In the first case we let $\phi'(lo_\mathcal{O}) = lo_{\mathcal{O}'}$, $\phi'(mi_\mathcal{O}) = mi_{\mathcal{O}'}$, $\phi'(hi_\mathcal{O}) = hi_{\mathcal{O}'}$; in the second we let $\phi'(lo_\mathcal{O}) = hi_{\mathcal{O}'}$, $\phi'(mi_\mathcal{O}) = mi_{\mathcal{O}'}$ and $\phi'(hi_\mathcal{O}) = lo_{\mathcal{O}'}$.

Finally, since all children of a *lo*, *mi* or *hi* node have different colors, there is a unique way to define ϕ' on the slot-nodes of $\mathbb{T}(\mathcal{H})$.

Now it can be easily checked that ϕ' indeed is an isomorphism between $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$. □

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

Theorem 4.7. *Given an interval hypergraph $\mathcal{H}$, a canonical interval labeling $\ell_\mathcal{H}$ for $\mathcal{H}$ can be computed in* FL.

Proof. We have to show that the labelings $\ell_\mathcal{H}$ and $\ell_\mathcal{K}$ of any two isomorphic interval hypergraphs $\mathcal{H}$ and $\mathcal{K}$ map these graphs to the same interval representation $\mathcal{H}^{\ell_\mathcal{H}} = \mathcal{K}^{\ell_\mathcal{K}}$:

By Lemma 4.6, the colored trees $\mathbb{T}(\mathcal{H})$ and $\mathbb{T}(\mathcal{K})$ are isomorphic. Hence it follows that the canonical labelings $\ell_{\mathbb{T}(\mathcal{H})}$ and $\ell_{\mathbb{T}(\mathcal{K})}$ map these trees to the same colored tree $\mathbb{T}(\mathcal{H})^{\ell_{\mathbb{T}(\mathcal{H})}} = \mathbb{T}(\mathcal{K})^{\ell_{\mathbb{T}(\mathcal{K})}}$. Further, it is easy to see that the interval representation $\mathcal{H}^{\ell_\mathcal{H}}$ only depends on the tree $\mathbb{T}(\mathcal{H})^{\ell_{\mathbb{T}(\mathcal{H})}}$, implying that $\mathcal{H}^{\ell_\mathcal{H}} = \mathcal{K}^{\ell_\mathcal{K}}$. □

5 Canonizing interval graphs and convex graphs

The reduction of Lemma 2.4 transforms Theorem 4.7 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 iff $v \in H$. Note that this transformation yields exactly the convex graphs when applied to interval hypergraphs.

Corollary 5.2. *Canonical labelings for convex graphs 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 convex graph with vertex classes U and V, which can be found in logspace using Reingold's algorithm [Rei08]. Reversing the aforementioned transformation of hypergraphs into bigraphs, 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 $\mathcal{H}_U$ given by Lemma 3.2. 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. □

6 Computing proper and unit interval representations

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 $\ell : V(G) \to \mathcal{I}$ of a graph G is *proper* if the interval representation $\mathcal{I}$ is proper. Graphs admitting such labelings are called *proper interval graphs*. Let $\mathcal{I} = \big\{[a_i, b_i] \,\big|\, 1 \leq i \leq n\big\}$. In the absence of inclusions, the endpoints a_i's are pairwise distinct and the same is true about the b_i's. 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.[3] If combined with the aforementioned result of Harary and McKee and our Theorem 4.7, this immediately gives us a logspace computable complete invariant for proper interval graphs. This invariant is generally not an interval representation of G. Note

[3] It is also not hard to prove the converse: If $\mathcal{N}_G$ is an interval hypergraph, then G is a proper interval graph (see [Duc84, Corollary 8.9]).

also that the minimal interval representation constructed in Corollary 5.1 is not proper if the graph contains at least one edge (see Fig. 5 for an example). Now our aim is to come up with a canonical proper interval representation of a given proper interval graph.

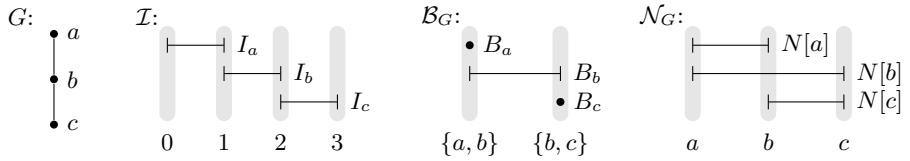

Fig. 5. A proper interval graph G and a proper interval representation $\mathcal{I}$ of G. However, the bundle hypergraph $\mathcal{B}_G$ is not proper. The neighborhood hypergraph $\mathcal{N}_G$ is neither proper nor an interval representation of G.

It is easy to see that a proper interval graph with n vertices always has a 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 $\ell : V(G) \to \mathcal{I}$, the graph G has also the reversed proper interval labeling $\ell^* : V(G) \to \mathcal{I}^*$ 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 by a different method, which allows us to obtain also a logspace computability result.

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

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

Let $\ell : V(G) \to \mathcal{I}$ 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 it was mentioned, $\prec_\ell$ defines an interval representation of the neighborhood hypergraph $\mathcal{N}_G$.

Given a non-central vertex v_i, let $s = s(i)$ be the largest index such that $s < i$ and $v_s \notin N[v_i]$ and, similarly, $t = t(i)$ 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 $N[v_i] \not\subset N[v_s]$. 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 $N[v_i] \not\subset N[v_t]$. Note that neither v_s nor v_t is central. It follows that, for any non-central v_i, there is a subsequence of indices $i_1, \ldots, i_k$ containing i such that

$$N[v_{i_1}] \not\subset N[v_{i_2}] \not\subset \cdots \not\subset N[v_{i_k}] \quad \text{and} \quad N[v_{i_1}] \cup N[v_{i_2}] \cup \cdots \cup N[v_{i_k}] = V(G).$$

If there is a central 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

$$\begin{aligned} a_i &= i + \|\{j < i \mid v_j \text{ is non-adjacent to } v_i\}\| \\ \text{and } \ b_i &= a_i + 1 + \deg(v_i). \end{aligned} \tag{4}$$

The computability result readily follows by (4) 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 $\ell : V(G) \to \mathcal{I}$ and $\ell^* : V(G) \to \mathcal{I}^*$. 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 reversing). 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 (4). 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 algorithm. For each of them, we compute the canonical labeling $\ell_{G_j} : V(G_j) \to \mathcal{I}_j$ and sort out the interval representations $\mathcal{I}_1, \ldots, \mathcal{I}_k$. Then we merge the ℓ_{G_j}'s into an integrated labeling $\ell_G : V(G) \to \mathcal{I}$ so that the supports of the $\mathcal{I}_j$'s 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).

Lastly, 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 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 strict 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 tree T on V that is rooted at v_1.

For every 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 a postorder traversal of T, where the children of each node in T are ordered as in the preceding paragraph. Since T is easy to define in logspace, both values can be computed in FL for any $v \in V$. Assign to any vertex v the value $v^L = k(v) + p(v)/n$. Corneil et al. show in [CKN$^+$95, Theorem 3.2] that $\{(v_i^L, v_i^L + 1) \mid i \in [n]\}$ is a unit interval representation of G, and that assigning $(v_i^L, v_i^L + 1)$ to v_i for every $i \in [n]$ 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

Having a canonical (interval) labeling 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 4.7, Lemma 2.4 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 interval graph is in* FL. *Further, the automorphism problem (i. e., deciding if a given graph has a non-trivial automorphism) for interval graphs is in* L.

The same holds for interval hypergraphs and convex graphs.

In this section, we additionally prove 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 the L-algorithms for these classes are already shown in the previous sections. To prove the 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 $\{c, c'\}$. 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. □

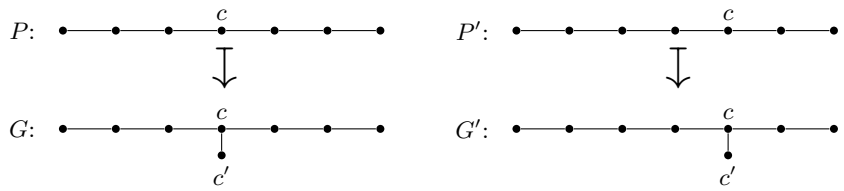

Fig. 6. The reduction from PathCenter to the automorphism problem of caterpillars.

The following lemma can be proved using a similar construction, that also marks either of the two end vertices of P (using two additional vertices) 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.* □

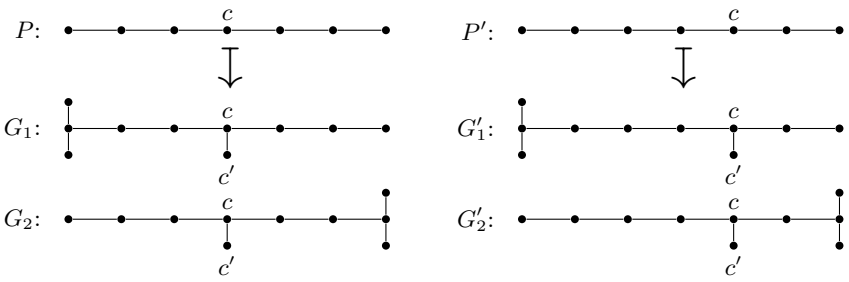

Fig. 7. The reduction from PathCenter to the isomorphism problem of caterpillars.

We observe that these constructions can be modified to yield unit 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). To obtain a unit interval labeling, the marker vertices can be mapped to the interval of the corresponding marked vertex, modified by a small offset that does not change intersections.

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. The only difference is that no new edges are added, but the edges incident to the vertex that is to be marked is extended to also include the marker vertices.

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. *It is* L*-complete to decide if a given graph is*

- *an interval graph,*
- *a proper interval graph,*
- *a convex graph,*
- *a biconvex graph,*
- *a bipartite permutation graph,*
- *a caterpillar, and*
- *a path, respectively.*

Recognition of interval graphs in logspace follows from the results of Reif [Rei84]. Each of the classes of bipartite permutation, biconvex, and convex graphs admits a characterization in terms of so-called asteroidal triples; see, e.g., [BLS99, Proposition 6.2.1] and [Spi03, Section 9.7.2]. 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 just by noticing the logspace equivalence between recognition of interval hypergraphs and convex graphs. 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 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 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 s or t are among the first 2 vertices in the path. If so, output a trivial *yes* or *no*-instance depending on whether s has been encountered first.

If neither s nor t are among the first 2 vertices of P, then we construct an undirected graph G from P as follows: For each vertex $v \in V(P)$ insert a new vertex v' between v and its successor. 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. Fig. 8 shows an illustration of this construction. It is easy to verify that positive instances of Ord are mapped to paths, while the image of negative instances contains a chordless circle of odd length consisting of at least 5 vertices.

This construction can clearly be done in DLOGTIME-uniform AC^0. □

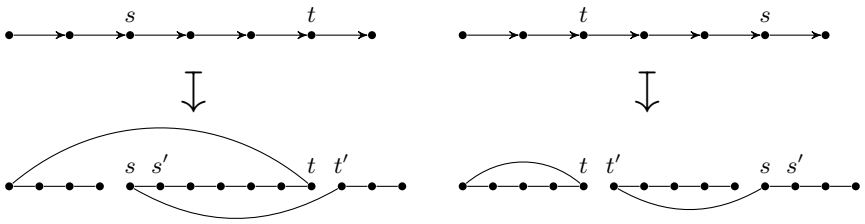

Fig. 8. The reduction that maps positive instances of Ord to paths and negative instances to non-chordal non-bipartite graphs.

Remark 7.8. Using Reingold's undirected graph reachability algorithm [Rei08], it is easy to see 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 of these two graphs classes.

We also remark that deciding if an interval graph is a proper one is in AC^0, as an interval graph is proper iff it has no induced copy of $K_{1,3}$ [Rob69].

Corollary 7.9. *It is logspace complete to compute a perfect elimination order (peo) or an interval labeling of an interval graph G.*

Proof. An interval labeling can be constructed in logspace by Corollary 5.1. Computing a peo is logspace hard even for paths [KK09]. Finally, it is not hard to see that the ordering the vertices of G induced by any interval labeling of G is a peo. □

8 Conclusion

We have proved that a canonical interval labeling of a 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 problems of the recognition, isomorphism and automorphism problems of these graph classes and conclude 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 the Weisfeiler-Lehman algorithm solves the interval graph isomorphism with parallel complexity in TC^1. The details will appear in a follow-up paper.

Going beyond interval graphs, there are several natural graph classes that suggest an investigation whether they can similarly be handled in L. For example, circular-arc graphs generalize interval graphs as intersection graphs of arcs on a circle. Just like interval graphs, circular-arc graphs can be recognized efficiently in linear time (cf. [KN06]). However, while 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 and directed tree. While in this class, maxcliques can still be recognized in a similar way as in this paper, the recursive procedure for linearly ordering maxcliques as given in Section 3 cannot be employed in the presence of tree nodes of degree ≥ 3 (cf. [Lau10]).

In the above paragraph, it is important that trees are rooted and directed accordingly, as intersection graphs of paths in undirected trees are isomorphism-complete (cf. [BPT96]). The same is true for boxicity-d graphs ($d \geq 2$), the intersection graphs of axis-parallel boxes in $\mathbb{R}^d$ (cf. [Ueh08]). Also, the two arguably most manifest extensions of interval graphs, chordal graphs and co-comparability graphs, are known to be isomorphism-complete. Finally, we would like to point to [Spi03] for further graph classes for which recognition and isomorphism is not known to be in L.

Acknowledgement. We thank the anonymous referees of the conference version for helpful comments and detailed suggestions on how to improve this paper.

References

[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(3):542–564, 1996.

[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(4):442–456, 2007.

[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. Syst. Sci.*, 13(3):335–379, 1976.

[BLS99] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discr. Math. and Appl., SIAM, 1999.

[CFI92] Cai, J. and Fürer, M. and Immerman, N. An Optimal Lower Bound on the Number of Variables for Graph Identification. *Combinatorica*, 12(4):389–410, 1992.

[Che96] L. Chen. Graph isomorphism and identification matrices: Parallel algorithms. *IEEE Trans. Paral. Distrib. Syst.* 7(3):308-319, 1996.

[Che99] L. Chen. Graph isomorphism and identification matrices: Sequential algorithms. *J. Comput. Syst. Sci.*, 29(3):450–475, 1999.

[CKN+95] D.G. Corneil, H. Kim, S. Natarajan, S. Olariu, and A.P. Sprague. Simple linear time recognition of unit interval graphs. *Inf. Process. Lett.*, 55(2):99–104, 1995.

[CY93] L. Chen and Y. Yesha. Efficient parallel algorithms for bipartite permutation graphs. *Networks*, 23(1):29–39, 1993.

[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(2):390–403, 1996.

[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*, pages 145–156, 2009.

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

[Duc78] P. Duchet. Propriété de helly et problèmes de représentation. Problèmes combinatoires et théorie des graphes, Orsay 1976, Colloq. int. CNRS No. 260, 117-118, 1978.

[Duc84] P. Duchet. Classical perfect graphs. Ann. Discrete Math., 21:67–96, 1984.

[Ete97] K. Etessami. Counting quantifiers, successor relations, and logarithmic space. *J. Comput. Syst. Sci.*, 54(3):400 – 411, 1997.

[FG65] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[GV06] M. Grohe and O. Verbitsky. Testing Graph Isomorphism in Parallel by Playing a Game. *ICALP*, 3–14, 2006.

[HMP+00] M. Habib, R.M. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement. *Theor. Comput. Sci.*, 234(1-2):59–84, 2000.

[HM94] F. Harary and T.A. McKee. The square of a chordal graph. *Discrete Math.*, 128(1-3):165–172, 1994.

[HSS01] P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Comput.*, 31(1):289–305, 2001.

[Hof82] C.M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism*, volume 136 of *LNCS*. Springer, 1982.

[HM99] Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM J. Comput.*, 28(3):1004–1020, 1999.

[KN06] H. Kaplan and Y. Nussbaum. A simpler linear-time recognition of circular-arc graphs. In *SWAT*, volume 4059 of *LNCS*, pages 41–52. Springer, 2006.

[Kle96] P.N. Klein. Efficient parallel algorithms for chordal graphs. *SIAM J. Comput.*, 25(4):797–827, 1996.

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

[KST93] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Birkhäuser, 1993.

[KVV85] D. Kozen, U.V. Vazirani, and V.V. Vazirani. NC algorithms for comparability graphs, interval gaphs, and testing for unique perfect matching. In *FSTTCS*, volume 206 of *LNCS*, pages 496–503. Springer, 1985.

[Lau10] B. Laubner. Capturing polynomial time on interval graphs. *LICS* 2010, to appear.

[Lin92] S. Lindell. A logspace algorithm for tree canonization. *STOC* 1992, pages 400–404.

[LB79] G.S. Lueker and K.S. Booth. A linear time algorithm for deciding interval graph isomorphism. *J. ACM*, 26(2):183–195, 1979.

[Möh84] R.H. Möhring. *Graphs and Order*, volume 147 of *NATO ASI Series C, Mathematical and Physical Sciences*, pages 41–102. D. Reidel, 1984.

[Rei84] J.H. Reif. Symmetric complementation. *J. ACM*, 31(2):401–421, 1984.

[Rei08] O. Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17.1–17.24, 2008.

[Rob69] F.S. Roberts. Indifference graphs. In *Proof techniques in graph theory: Proc. 2nd Ann Arbor Graph Theory Conference*, pages 139–146. Academic Press, 1969.

[RTL76] D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[Spi03] J.P. Spinrad. *Efficient graph representations*. Volume 19 of Field Institute Monographs, 2003.

[Tuk72] A. Tucker. A structure theorem for the consecutive 1's property. *J. Comb. Theory*, Series B, 12(2):153–162, 1972.

[Ueh08] R. Uehara. Simple geometrical intersection graphs. In *WALCOM*, volume 4921 of *LNCS*, pages 25–33. Springer, 2008.

[YC96] C.-W. Yu and G.-H. Chen. An efficient parallel recognition algorithm for bipartite-permutation graphs. *IEEE Trans. Parallel Distrib. Syst.*, 7(1):3–10, 1996.

[ZSF+94] P. Zhang, E.A. Schon, S.G. Fischer, E. Cayanis, J. Weiss, S. Kistler, and P.E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *Bioinformatics*, 10(3):309–317, 1994.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de