Fixed-Point Definability and Polynomial Time on Chordal Graphs and Line Graphs

Martin Grohe
Humboldt University Berlin

March 22, 2010

Abstract

The question of whether there is a logic that captures polynomial time was formulated by Yuri Gurevich in 1988. It is still wide open and regarded as one of the main open problems in finite model theory and database theory. Partial results have been obtained for specific classes of structures. In particular, it is known that fixed-point logic with counting captures polynomial time on all classes of graphs with excluded minors. The introductory part of this paper is a short survey of the state-of-the-art in the quest for a logic capturing polynomial time.

The main part of the paper is concerned with classes of graphs defined by excluding induced subgraphs. Two of the most fundamental such classes are the class of chordal graphs and the class of line graphs. We prove that capturing polynomial time on either of these classes is as hard as capturing it on the class of all graphs. In particular, this implies that fixed-point logic with counting does not capture polynomial time on these classes. Then we prove that fixed-point logic with counting does capture polynomial time on the class of all graphs that are both chordal and line graphs.

1 The quest for a logic capturing PTIME

Descriptive complexity theory started with a theorem Fagin proved in 1974, stating that existential second-order logic *captures* the complexity class NP. This means that a property of finite structures is decidable in nondeterminisctic polynomial time if and only if it is definable in existential second order logic. Similar logical characterisations where later found for most other complexity classes. For example, in 1982 Immerman [43] and independently Vardi [58] characterised the class PTIME (polynomial time) in terms of least fixed-point logic, and in 1983 Immerman [45] characterised the classes NLOGSPACE (nondeterministic logarithmic space) and LOGSPACE (logarithmic space) in terms of transitive closure logic and its deterministic variant. However, these logical characterisations of the classes PTIME, NLOGSPACE, and LOGSPACE, and all other known logical characterisations of complexity classes contained in PTIME, have a serious drawback: They only apply to properties of *ordered structures*, that is, relational structures with one distinguished relation that is a linear order of the elements of the structure. It is still an open question whether there are logics that characterise these complexity classes on arbitrary, not necessarily ordered structures. We focus on the class PTIME from now on. In this section, which is an updated version of [32], we a short survey of the quest for a logic capturing PTIME.

1.1 Logics capturing PTIME

The question of wether there is a logic that characterises, or *captures*, PTIME is subtle. If phrased naively, it has a trivial, but completely uninteresting positive answer. Yuri Gurevich [36] was the first to give a precise formulation of the question. Instead of arbitrary finite structures, we restrict our attention to graphs in this paper. This is no serious restriction, because the question of wethere there is a logic that captures PTIME on arbitrary structures is equivalent to the restriction of the question to graphs. We first need to define what constitutes a logic. Following Gurevich, we take a very liberal, semantically oriented approach. We identify *properties* of graphs with classes of graphs closed under isomorphism. A logic L (on graphs) consists of a computable set of *sentences* together with a semantics that associates a property $\mathscr{P}_\varphi$ of graphs with each sentence φ. We say that a graph G *satisfies* a sentence φ, and write $G \models \varphi$, if $G \in \mathscr{P}_\varphi$. We say that a property $\mathscr{P}$ of graphs is *definable* in L if there is a sentence φ such that $\mathscr{P}_\varphi = \mathscr{P}$. A logic L *captures* PTIME if the following two conditions are satisfied:

(G.1) Every property of graphs that is decidable in PTIME is definable in L.

(G.2) There is a computable function that associates with every L-sentence φ a polynomial $p(X)$ and an algorithm A such that A decides the property $\mathscr{P}_\varphi$ in time $p(n)$, where n is the number of vertices of the input graph.

While condition (G.1) is obviously necessary, condition (G.2) may seem unnecessarily complicated. The natural condition we expect to see instead is the following condition (G.2'): Every property of graphs that is definable in L is decidable in PTIME. Note that (G.2) implies (G.2'), but that the converse does not hold. However, (G.2') is too weak, as the following example illustrates:

Example 1.1. Let $\mathscr{P}_1, \mathscr{P}_2, \ldots$ be an arbitrary enumeration of all polynomial time decidable properties of graphs. Such an enumeration exists because there are only countably many Turing machines and hence only countable many decidable properties of graphs. Let L$'$ be the "logic" whose sentences are the natural numbers and whose semantics is defined by letting sentence i define property $\mathscr{P}_i$. Then L$'$ is a logic according to our definition, and it does satisfy (G.1) and (G.2'). But clearly, L$'$ is not a "logic capturing PTIME" in any interesting sense.

Let me remark that most natural logics that are candidates for capturing PTIME trivially satisfy (G.2). The difficulty is to prove that they also satisfy (G.1), that is, define all PTIME-properties.

There is a different route that leads to the same question of whether there is a logic capturing PTIME from a database-theory perspective: After Aho and Ullman [2] had realised that SQL, the standard query language for relational databases, cannot express all database queries computable in polynomial time, Chandra and Harel [10] asked for a recursive enumeration of the class of all relational database queries computable in polynomial time. It turned out that Chandra and Harel's question is equivalent to Gurevich's question for a logic capturing PTIME, up to a minor technical detail.[1]

The question of whether there is a logic that captures PTIME is still wide open, and it is considered one of the main open problems in finite model theory and database theory today. Gurevich conjectured that there is no logic capturing PTIME. This would not only imply that PTIME $\neq$ NP — remember that by Fagin's Theorem there is a logic capturing NP — but it would actually have interesting consequences for the structure of the complexity class PTIME. Dawar [15] proved a dichotomy theorem stating that, depending on the answer to the question, there are two fundamentally different possibilities: If there is a logic for PTIME, then the structure of PTIME is very simple; all PTIME-properties are variants or special cases of just one problem. If there is no logic for PTIME, then the structure of PTIME is so complicated that it eludes all attempts for a classification. The formal statement of the first possibility is that there is a complete problem for PTIME under first-order reductions. The formal statement of the second possibility is that the class of PTIME-properties is not recursively enumerable.

1.2 Fixed-point logics

Fixed-point logics play an important role in finite-model theory, and in particular in the quest for a logic capturing PTIME. Very briefly, the fixed-point logics considered in this context are extensions of first-order logic by operators that formalise inductive definitions. We have already mentioned that *least fixed-point logic* LFP captures polynomial time on ordered structures; this result is known as the *Immerman-Vardi Theorem*. For us, it will be more convenient to work with *inflationary fixed-point logic* IFP, which was shown to have the same expressive power as LFP on finite structures by Gurevich and Shelah [38] and on infinite structures by Kreutzer [49].

IFP does not capture polynomial time on all finite structures. The most immediate reason is the inability of the logic to count. For example, there is no IFP-sentence stating that the vertex set of a graph has even

[1] In Chandra and Harel's version of the question, condition (G.2) needs to be replaced by the following condition (CH.2): There is a computable function that associates with every L-sentence φ an algorithm A such that A decides the property $\mathscr{P}_\varphi$ in polynomial time. The difference between (G.2) and (CH.2) is that in (CH.2) the polynomial bounding the running time of the algorithm A is not required to be computable from φ.

cardinality; obviously, the graph property of having an even number of vertices is decidable in polynomial time. This led Immerman [44] to extending fixed-point logic by "counting operators". The formal definition of fixed-point logic with counting operators that we use today, *inflationory fixed-point logic with counting* IFP+C, is due to Grädel and Otto [28]. IFP+C comes surprisingly close to capturing PTIME. Even though Cai, Fürer, and Immerman [9] gave an example of a property of graphs that is decidable in PTIME, but not definable in IFP+C, it turns out that the logic does capture PTIME on many interesting classes of structures.

1.3 Capturing PTIME on classes of graphs

Let $\mathscr{C}$ be a class of graphs, which we assume to be closed under isomorphism. We say that a logic L captures PTIME *on* $\mathscr{C}$ if it satisfies the following two conditions:

(G.1)$_{\mathscr{C}}$ For every property $\mathscr{P}$ of graphs that is decidable in PTIME there is an L-sentence φ such that for all graphs $G \in \mathscr{C}$ it holds that $G \models \varphi$ if and only if $G \in \mathscr{P}$.

(G.2)$_{\mathscr{C}}$ There is a computable function that associates with every L-sentence φ a polynomial $p(X)$ and an algorithm A such that given a graph $G \in \mathscr{C}$, the algorithm A decides if $G \models \varphi$ in time $p(n)$, where n is the number of vertices of G.

Note that these conditions coincide with conditions (G.1) and (G.2) if $\mathscr{C}$ is the class of all graphs.

The first positive result in this direction is due to Immerman and Lander [47], who proved that IFP+C captures PTIME on the class of all trees. In 1998, I proved that IFP+C captures PTIME on the class of all planar graphs [30] and around the same time, Julian Mariño and I proved that IFP+C captures PTIME on all classes of structures of bounded tree width [34]. In [31], I proved the same result for the class of all graphs that have no complete graph on five vertices, K_5, as a minor. A *minor* of graph G is a graph H that can be obtained from a subgraph of G by contracting edges. We say that a class $\mathscr{C}$ of graphs *excludes a minor* if there is a graph H that is not a minor of any graph in $\mathscr{C}$. Very recently, I proved that IFP+C captures PTIME on all classes of graphs that exclude a minor [33].

In the last few years, maybe as a consequence of Chudnowsky, Robertson, Seymour, and Thomas's [11] proof of the strong perfect graph theorem, the focus of many graph theorists has shifted from graph classes with excluded minors to graph classes defined by excluding subgraphs. One of the most basic and important example of such a class is the class of *chordal graphs*. A cycle C of a graph G is *chordless* if it is an induced subgraph. A graph is *chordal* (or *triangulated*) if it has no chordless cycle of length at least four. Figure 1.1(a) shows an example of a chordal graph. All chordal graphs are *perfect*, which means that the graphs themselves and all their induced subgraphs have the same chromatic number and clique number. Chordal graphs have a nice and simple structure; they can be decomposed into a tree of cliques. A second important example is the class of *line graphs*. The line graph of a graph G is the graph $L(G)$ whose vertices are the edges of G, with two edges being adjacent in $L(G)$ if they have a common endvertex in G. Figure 1.1(b) shows an example of a line graph. The class of all line graphs is closed under taking induced subgraphs. Beineke [5] gave a characterisation of the class of line graphs (more precisely, the class of all graphs isomorphic to a line graph) by a family of nine excluded subgraphs. An extension of the class of line graphs, which has also received a lot of attention in the literature, is the class of *claw-free* graphs. A graph is claw-free if it does not have a vertex with three pairwise nonadjacent neighbours, that is, if it does not have a *claw* (displayed in Figure 1.2) as an induced subgraph. It is easy to see that all line graphs are claw-free. Recently, Chudnowsky and Seymour (see [12]) developed a structure theory for claw-free graphs.

It would be tempting to use this structure theory for claw free graphs, or at least the simple treelike structure of chordal graphs, to prove that IFP+C or captures PTIME on these classes in a similar way as the structure theory for classes of graphs with excluded minors is used to prove that IFP+C captures PTIME on classes with excluded minors. Unfortunately, this is only possible on the very restricted class of graphs that are both chordal and line graphs (an example of such a graph is shown in Figure 4.1 on p.11). We prove

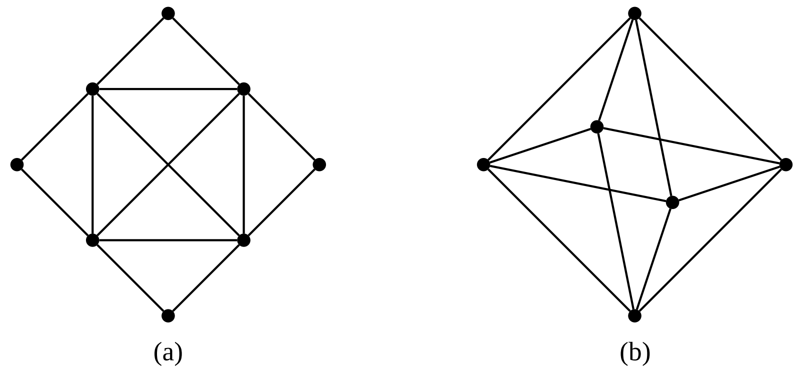

Figure 1.1. (a) a chordal graph, which is not a line graph, and (b) the line graph of K_4, which is not chordal

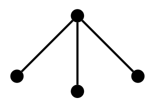

Figure 1.2. A claw

the following theorem:

Theorem 1.2.

(1) IFP+C *does not capture* PTIME *on the class of chordal graphs or on the class of line graphs.*

(2) IFP+C *captures* PTIME *on the class of chordal line graphs.*

Our construction to prove (1) is so simple that it will apply to any reasonable logic, which means that if a "reasonable" logic captures PTIME on the class of chordal graphs or on the class of line graphs, then it captures PTIME on the class of all graphs.

To conclude our discussion of classes of graphs on which IFP+C captures PTIME, let me mention a result due to Hella, Kolaitis, and Luosto [40] stating that IFP+C captures PTIME on almost all graphs, in a precise technical sense. Thus it seems that the results for specific classes of graphs are not very surprising, but it should be mentioned that almost all graphs are neither chordal nor line graphs nor do they exclude any specific graph as a minor.

Instead of capturing all PTIME on a specific class of structures, Otto [53, 54, 55] studied the question of capturing all PTIME properties satisfying certain invariance conditions. Most notably, he proved that bisimulation-invariant properties are decidable in polynomial time if and only if they are definable in the *higher-dimensional μ-calculus.*

1.4 Isomorphism testing and canonisation

As an abstract question, the question of whether there is a logic capturing polynomial time is linked to the graph isomorphism and canonisation problems. Specifically, if there is a polynomial time canonisation algorithm for a class $\mathscr{C}$ of graphs, then there is a logic that captures polynomial time on this class $\mathscr{C}$. This follows from the Immerman-Vardi Theorem. To explain this, let us consider graphs and assume that we represent them by their adjacency matrices. A *canonisation mapping* gets as argument some adjacency matrix representing a graph and returns a *canonical* adjacency matrix for this graph, that is, it maps *isomorphic* adjacency matrices to *equal* adjacency matrices. As an adjacency matrix for a graph is completely fixed once we specify the ordering of the rows and columns of the matrix, we may view a canonisation as a mapping associating with each graph a canonical ordered copy of the graph. Now we can apply the Immerman-Vardi Theorem to this ordered copy.

Clearly, if there is a polynomial time canonisation mapping for a class of graphs (or other structures) then there is a polynomial time isomorphism test for this class. It is open whether the converse also holds.

It is also open whether the existence of a logic for polynomial time implies the existence of a polynomial time isomorphism test or canonisation mapping.

Polynomial time canonisation mappings are known for many natural classes of graphs, for example planar graphs [41, 42], graphs of bounded genus [25, 52], graphs of bounded eigenvalue multiplicity [3], graphs of bounded degree [4, 51], and graphs of bounded tree width [8]. Hence for all theses classes there are logics capturing PTIME. However, the logics obtained through canonisation hardly qualify as natural logics. If a logic is to contribute to our understanding of the complexity class PTIME— and from my perspective this is the main reason for being interested in such a logic — we have to look for natural logics that derive their expressiveness from clearly visible basic principles like inductive definability, counting or other combinatorial operations, and maybe fundamental algebraic operations like computing the rank or the determinant of a matrix. If such a logic captures polynomial time on a class of structures, then this shows that all polynomial time properties of structures in this class are based on the principles underlying the logic. Thus even for classes for which we know that there is a logic capturing PTIME through a polynomial-time canonisation algorithm, I think it is important to find "natural" logics capturing PTIME on these classes. In particular, I view it as an important open problem to find a natural logic that captures PTIME on classes of graphs of bounded degree. It is known that IFP+C does not capture PTIME on the class of all graphs of maximum degree at most three.

Most known capturing results are proved by showing that there is a canonisation mapping that is definable in some logic. In particular, all capturing results for IFP+C mentioned above are proved this way. It was observed by Cai, Fürer, and Immerman [9] that for classes $\mathscr{C}$ of structures which admit a canonisation mapping definable in IFP+C, a simple combinatorial algorithm known as the Weisfeiler-Leman (WL) algorithm [23, 24] can be used as a polynomial time isomorphism test on $\mathscr{C}$. Thus the the WL-algorithm correctly decides isomorphism on the class of chordal line graphs and on all classes of graphs with excluded minors. A refined version of the same approach was used by Verbitsky and others [35, 48, 59] to obtain parallel isomorphism tests running in polylogarithmic time for planar graphs and graphs of bounded tree width.

1.5 Stronger logics

Early on, a number of results regarding the possibility of capturing polynomial time by adding Lindström quantifiers to first-order logic or fixed-point logic were obtained. Hella [39] proved that adding finitely many Lindström quantifiers (or infinitely many of bounded arity) to fixed-point logic does not suffice to capture polynomial time (also see [17]). Dawar [14] proved that if there is a logic capturing polynomial time, then there is such a logic obtained from fixed-point logic by adding one vectorised family of Lindström quantifiers. Another family of logics that have been studied in this context consists of extensions of fixed-point logic with nondeterministic choice operators [1, 18, 26].

Currently, the two main candidates for logics capturing PTIME are *choiceless polynomial time with counting* CP+C and *inflationary fixed-point logic with a rank operator* IFP+R. The logic CP+C was introduced ten years ago by Blass, Gurevich and Shelah [6] (also see [7, 19]). The formal definition of the logic is carried out in the framework of *abstract state machines* (see, for example, [37]). Intuitively CP+C may be viewed as a version of IFP+C where quantification and fixed-point operators not only range over elements of a structure, but instead over all objects that can be described by $O(\log n)$ bits, where n is the size of the structure. This intuition can be formalised in an expansion of a structure by all hereditarily finite sets which use the elements of the structure as atoms. The logic IFP+R, introduced recently [16], is an extension of IFP by an operator that determines the rank of definable matrices in a structure. This may be viewed as a higher dimensional version of a counting operator. (Counting appears as a special case of diagonal $\{0,1\}$-matrices.)

Both CP+C and IFP+R are known to be strictly more expressive than IFP+C. Indeed, both logics can express the property used by Cai, Fürer, and Immerman to separate IFP+C from PTIME. For both logics it is open whether they capture polynomial time, and it is also open whether one of them semantically contains the other.

2 Preliminaries

$\mathbb{N}_0$, and $\mathbb{N}$ denote the sets of nonnegative integers and natural numbers (that is, positive integers), respectively. For $m,n \in \mathbb{N}_0$, we let $[m,n] := \{\ell \in \mathbb{N}_0 \mid m \leq \ell \leq n\}$ and $[n] := [1,n]$.

We often denote tuples $(v_1,\dots,v_k)$ by $\vec{v}$. If $\vec{v}$ denotes the tuple $(v_1,\dots,v_k)$, then by $\tilde{v}$ we denote the set $\{v_1,\dots,v_k\}$. If $\vec{v} = (v_1,\dots,v_k)$ and $\vec{w} = (w_1,\dots,w_\ell)$, then by $\vec{v}\vec{w}$ we denote the tuple $(v_1,\dots,v_k,w_1,\dots,w_\ell)$. By $|\vec{v}|$ we denote the length of a tuple $\vec{v}$, that is, $|(v_1,\dots,v_k)| = k$.

2.1 Graphs

Graphs in this paper are always finite, nonempty, and simple, where simple means that there are no loops or parallel edges. Unless explictly called "directed", graphs are undirected. The vertex set of a graph G is denoted by $V(G)$ and the edge set by $E(G)$. We view graphs as relational structures with $E(G)$ being a binary relation on $V(G)$. However, we often find it convenient to view edges (of undirected graphs) as 2-element subsets of $V(G)$ and use notations like $e = \{u,v\}$ and $v \in e$. Subgraphs, induced subgraphs, union, and intersection of graphs are defined in the usual way. We write $G[W]$ to denote the induced subgraph of G with vertex set $W \subseteq V(G)$, and we write $G \setminus W$ to denote $G[V(G) \setminus W]$. The set $\{w \in V(G) \mid \{v,w\} \in E(G)\}$ of *neighbours* of a node v is denoted by $N^G(v)$, or just $N(v)$ if G is clear from the context, and the *degree* of v is the cardinality of $N(v)$. The *order* of a graph, denoted by $|G|$, is the number of vertices of G. The class of all graphs is denoted by $\mathcal{G}$. A *homomorphism* from a graph G to a graph H is a mapping $h : V(G) \to V(H)$ that preserves adjacency, and an *isomorphism* is a bijective homomorphism whose inverse is also a homomorphism.

For every finite nonempty set V, we let $K[V]$ be the *complete graph* with vertex set V, and we let $K_n := K\big[[n]\big]$. A *clique* in a graph G is a set $W \subseteq V(G)$ such that $G[W]$ is a complete graph. *Paths* and *cycles* in graphs are defined in the usual way. The *length* of a path or cycle is the number of its edges. *Connectedness* and *connected* components are defined in the usual way. A set $W \subseteq V(G)$ is *connected* in a graph G if $W \neq \emptyset$ and $G[W]$ is connected. For sets $W_1, W_2 \subseteq V(G)$, a set $S \subset V(G)$ *separates* W_1 *from* W_2 if there is no path from a vertex in $W_1 \setminus S$ to vertex in $W_2 \setminus S$ in the graph $G \setminus S$.

A *forest* is an undirected acyclic graph, and a *tree* is a connected forest. It will be a useful convention to call the vertices of trees and forests *nodes*. A *rooted tree* is a triple $T = (V(T), E(T), r(T))$, where $(V(T), E(T))$ is a tree and $r(T) \in V(T)$ is a distinguished node called the *root*.

We occasionally have to deal with *directed graphs*. We allow directed graphs to have loops. We use standard graph theoretic terminology for directed graphs, without going through it in detail. Homomorphisms and isomorphisms of directed graphs preserve the direction of the edges. Paths and cycles in a directed graph are always meant to be directed; otherwise we will call them "paths or cycles of the underlying undirected graph". Note that cyles in directed graphs may have length 1 or 2. For a directed graph D and a vertex $v \in V(D)$, we let $N^D(v) := \big\{w \in V(D) \;\big|\; (v,w) \in E(D)\big\}$. *Directed acyclic graphs* will be of particular importance in this paper, and we introduce some additional terminology for them: Let D be a directed acyclic graph. A node w is a *child* of a node v, and v is a *parent* of w, if $(v,w) \in E(D)$. We let $\trianglelefteq^D$ be the reflexive transitive closure of the edge relation $E(D)$ and $\triangleleft^D$ its irreflexive version. Then $\trianglelefteq^D$ is a partial order on $V(D)$.

A *directed tree* is a directed acyclic graph T in which every node has at most one parent, and for which there is a vertex r called the *root* such that for all $t \in V(t)$ there is a path from r to t. There is an obvious one-to-one correspondence between rooted trees and directed trees: For a rooted tree T with root $r := r(T)$ we define the corresponding directed tree T' by $V(T') := V(T)$ and $E(T') := \big\{(t,u) \;\big|\; \{t,u\} \in E(T)$ and t occurs on the path $rTu\big\}$. We freely jump back and forth between rooted trees and directed trees, depending on which will be more convenient. In particular, we use the terminology introduced for directed acyclic graphs (parents, children, the partial order $\trianglelefteq$, et cetera) for rooted trees.

2.2 Relational structures

A *relational structure* A consists of a finite set $V(A)$ called the *universe* or *vertex set* of A and finitely many relations on A. The only types of structures we will use in this paper are *graphs*, viewed as structures $G = \big(V(G), E(G)\big)$ with one binary relation $E(G)$, and *ordered graphs*, viewed as structures $G = \big(V(G), E(G), \leqslant(G)\big)$ with two binary relations $E(G)$ and $\leqslant(G)$, where $\big(V(G), E(G)\big)$ is a graph and

$\leqslant(G)$ is a linear order of the vertex set $V(G)$.

2.3 Logics

We assume that the reader has a basic knowledge in logic. In this section, we will informally introduce the two main logics IFP and IFP+C used in this paper. For background and a precise definition, I refer the reader to one of the textbooks [21, 27, 46, 50]. It will be convenient to start by briefly reviewing *first-order logic* FO. Formulas of first-order logic in the language of graphs are built from atomic formulas $E(x,y)$ and $x = y$ expressing adajacency and equality of vertices by the usual Boolean connectives and existential and universal quantifiers ranging over the vertices of a graph. First-order formulas in the language of ordered graphs may also contain atomic formulas of the form $x \leqslant y$ with the obvious meaning, and formulas in other languages may contain atomic formulas defined for these languages. We write $\varphi(x_1,\ldots,x_k)$ to denote that the free variables of a formula φ are among $x_1,\ldots,x_k$. For a graph G and vertices $v_1,\ldots,v_k$, we write $G \models \varphi[v_1,\ldots,v_k]$ to denote that G satisfies φ if x_i is interpreted by v_i, for all $i \in [k]$.

Inflationary fixed-point logic IFP is the extension of FO by a fixed-point operator with an inflationary semantics. To introduce this operator, let $\varphi(X,\vec{x})$ be a formula that, besides a k-tuple $\vec{x} = (x_1,\ldots,x_k)$ of free *individual variables* ranging over the vertices of a graph, has a free k-ary *relation variable* ranging over k-ary relations on the vertex set. For every graph G we define a sequence $R_i = R_i(G,\varphi,X,\vec{x})$, for $i \in \mathbb{N}_0$, of k-ary relations on $V(G)$ as follows:

$$R_0 := \emptyset$$
$$R_{i+1} := R_i \cup \left\{\vec{v} \;\middle|\; G \models \varphi[R_i,\vec{v}]\right\} \qquad \text{for all } i \in \mathbb{N}_0.$$

Since we have $R_0 \subseteq R_1 \subseteq R_2 \subseteq \cdots \subseteq V(G)^k$ and $V(G)$ is finite, the sequence reaches a fixed-point $R_n = R_{n+1} = R_i$ for all $i \geq n$, which we denote by $R_\infty = R_\infty(G,\varphi,X,\vec{x})$. The *ifp-operator* applied to $\varphi,X,\vec{x}$ defines this fixed-point. We use the following syntax:

$$\underbrace{\mathrm{ifp}\left(X \leftarrow \vec{x} \;\middle|\; \varphi\right)\vec{x}'}_{=:\psi(\vec{x}')}. \tag{2.1}$$

Here $\vec{x}'$ is another k-tuple of individual variables, which may coincide with $\vec{x}$. The variables in the tuple $\vec{x}'$ are the free variables of the formula $\psi(\vec{x}')$, and for every graph G and every tuple $\vec{v} \in V(G)^k$ of vertices we let $G \models \psi[\vec{v}] \iff \vec{v} \in R_\infty$. These definitions can easily be extended to a situation where the formula φ contains other free variables than X and and the variables in $\tilde{x}$; these variables remain free variables of ψ. Now formulas of inflationary fixed-point logic IFP in the language of graphs ar built from atomic formulas $E(x,y)$, $x = y$, and $X\vec{x}$ for relation variables X and tuples of individual variables $\vec{x}$ whose length matches the arity of X by the usual Boolean connectives and existential and universal quantifiers ranging over the vertices of a graph, and the ifp-operator.

Example 2.1. The IFP-sentence

$$\mathsf{conn} := \forall x_1 \forall x_2 \ \mathrm{ifp}\left(X \leftarrow (x_1,x_2) \;\middle|\; x_1 = x_2 \vee E(x_1,x_2) \vee \exists x_3\big(X(x_1,x_3) \wedge X(x_3,x_2)\big)\right)(x_1,x_2)$$

states that a graph is connected.

Inflationary fixed-point logic with counting, IFP+C, is the extension of IFP by counting operators that allow it to speak about cardinalities of definable sets and relations. To define IFP+C, we interprete the logic IFP over two sorted extensions of graphs (or other relational structures) by a numerical sort. For a graph G, we let $N(G)$ be the initial segement $\big[0,|G|\big]$ of the nonnengative integers. We let G^+ be the two-sorted structure $G \cup (N(G),\leq)$, where $\leq$ is the natural linear order on $N(G)$. To avoid confusion, we always assume that $V(G)$ and $N(G)$ are disjoint. We call the elements of the first sort $V(G)$ *vertices* and the elements of the second sort $N(G)$ *numbers*. Individual variables of our logic range either over the set $V(G)$ of vertices of G or over the set $N(G)$ of numbers of G. Relation variables may range over mixed relations, having certain places for vertices and certain places for numbers. Let us call the resulting logic, inflationary fixed-point logic over the two-sorted extensions of graphs, IFP^+. We may still view IFP^+ as a logic over plain graphs, because the extension G^+ is uniquely determined by G. More precisely, we say that a sentence φ of IFP^+ is satisfied by a graph G if it $G^+ \models \varphi$. *Inflationary fixed-point logic with counting* IFP+C is the extension of IFP^+ by *counting terms* formed as follows: For every formula φ and tuple $\vec{x}$ of vertex variables we add a term $\#\vec{x}\ \varphi$; the value of this term is the number of assigments to $\vec{x}$ such that φ is satisfied.

With each IFP+C-sentence φ in the language of graphs we associate the graph property $\mathscr{P}_\varphi := \{G \mid G \models \varphi\}$. As the set of all IFP+C-sentences is computable, we may thus view IFP+C as an abstract logic according to the definition given in Section 1.1. It is easy to see that IFP+C satisfies condition (G.2) and therefore condition (G.2)$_\mathscr{C}$ for every class $\mathscr{C}$ of graphs. Thus to prove that IFP+C captures PTIME on a class $\mathscr{C}$ it suffices to verify (G.1)$_\mathscr{C}$.

In the following examples, we use the notational convention that x and variants such as x_1, x' denote vertex variables and that y and variants denote number variables.

Example 2.2. The IFP+C-term $\mathsf{0} := \#x\ \neg x = x$ defines the number $0 \in N(G)$. The formula

$$\mathsf{succ}(y_1,y_2) := y_1 \leq y_2 \land \neg y_1 = y_2 \land \forall y(y \leq y_1 \lor y_2 \leq y)$$

defines the successor relation associated with the linear order $\leq$. The following IFP+C-formula defines the set of even numbers in $N(G)$:

$$\mathsf{even}(y) := \mathrm{ifp}\left(Y \leftarrow y \;\middle|\; y = \mathsf{0} \lor \exists y' \exists y'' \big(Y(y') \land \mathsf{succ}(y',y'') \land \mathsf{succ}(y'',y)\big)\right) y.$$

Example 2.3. A *Eulerian cycle* in a graph is a closed walk on which every edge occurs exactly once. A graph is *Eulerian* if it has a Eulerian cycle. It is a well-known fact that a graph is Eulerian if and only if it is connected and every vertex has even degree. Then the following IFP+C-sentence defines the class of Eulerian graphs:

$$\mathsf{eulerian} := \mathsf{conn} \land \forall x_1\ \mathsf{even}\big(\#x_2\ E(x_1,x_2)\big),$$

where conn is the sentence from Example 2.1 and even(y) is the formula from Example 2.2. By standard techniques from finite model theory, it can be proved that the class of Eulerian graphs is neither definable in IFP nor in the counting extension FO+C of first-order logic.

2.4 Syntactical interpretations

In the following, L is one of the logics IFP+C, IFP, or FO, and λ, μ are relational lanuages such as the language $\{E\}$ of graphs or the language $\{E, \leqslant\}$ of ordered graphs. An L$[\lambda]$*-formula* is an L-formula in the language λ, and similarly for μ. Furthermore, we let $\ell \in \mathbb{N}$. We need some additional notation. Let $\approx$ be an equivalence relation on a set U. For every $u \in U$, by $u/_\approx$ we denote the $\approx$-equivalence class of u, and we let $U/_\approx := \{u/_\approx \mid u \in U\}$ be the set of all equivalence classes. For a tuple $\vec{u} = (u_1,\ldots,u_k) \in U^k$ we let $\vec{u}/_\approx := (u_1/_\approx,\ldots,u_k/_\approx)$, and for a relation $R \subseteq U^k$ we let $R/_\approx := \{\vec{u}/_\approx \mid \vec{u} \in R\}$.

Definition 2.4. (1) An ℓ*-ary* L*-interpretation of* μ *in* λ is a tuple

$$\Gamma(\vec{x}) = \Big(\gamma_{app}(\vec{x}), \gamma_V(\vec{x},\vec{y}), \gamma_\approx(\vec{x},\vec{y}_1,\vec{y}_2), \big(\gamma_R(\vec{x},\vec{y}_R)\big)_{R\in\mu}\Big),$$

of L$[\lambda]$-formulas, where $\vec{x}, \vec{y}, \vec{y}_1, \vec{y}_2$, and $\vec{y}_R$ for $R \in \mu$ are tuples of individual variables such that $|\vec{y}| = |\vec{y}_1| = |\vec{y}_2| = \ell$ and $|\vec{y}_R| = k \cdot \ell$ for all k-ary $R \in \mu$. Furthermore, the tuple $\vec{x}$ consists entirely of vertex variables. The tuples $\vec{y}, \vec{y}_1, \vec{y}_2$, and $\vec{y}_R$ for $R \in \mu$ either all consist entirely of vertex variables or they all consist entirely of number variables.[2] In the latter case, we call $\Gamma(\vec{x})$ a *numerical* interpretation.

In the following, let $\Gamma(\vec{x})$ be an ℓ-ary L-interpretation of μ in λ. Let G be a λ-structure and $\vec{v} \in V(G)^{|\vec{x}|}$:

(3) $\Gamma(\vec{x})$ is *applicable* to $(G,\vec{v})$ if $G \models \gamma_{app}[\vec{v}]$.

[2] Of course we can relax this restriction by admitting both vertex and number variables in the tuples and making sure that the types match where necessary. However, the restricted definition given here is usually sufficient.

(4) If $\Gamma(\vec{x})$ is applicable to $(G,\vec{v})$, we let $\Gamma[G;\vec{v}]$ be the μ-structure with vertex set

$$V\big(\Gamma[G;\vec{v}]\big) := \gamma_V[G;\vec{v},\vec{y}]/_{\approx},$$

where $\approx$ is the reflexive, symmetric, transitive closure of $\gamma_{\approx}[G;\vec{v},\vec{y}_1,\vec{y}_2]$ viewed as a binary relation on $V(G)^\ell$. Furthermore, for k-ary $R \in \mu$, we let

$$R\big(\Gamma[G;\vec{v}]\big) := \Big(\gamma_R[G;\vec{v},\vec{y}_R] \cap \gamma_V[G;\vec{v},\vec{y}]^k\Big)\Big/_{\approx}.$$

Here we view the $k\cdot\ell$-ary relation $\gamma_R[G;\vec{v},\vec{y}_R]$ as a k-ary relation on $V(G)^\ell$.

Syntactical interpretations map λ-structures to μ-structures. The crucial observation is that they also induce a reverse translation from $\mathsf{L}[\mu]$-formulas to $\mathsf{L}[\lambda]$-formulas.

Fact 2.5 (Lemma on Syntactical Interpretations). *Let $\Gamma(\vec{x})$ be an ℓ-ary L-interpretation of μ in λ. Then for every $\mathsf{L}[\mu]$-sentence φ there is an $\mathsf{L}[\lambda]$-formula $\varphi^{-\Gamma}(\vec{x})$ such that the following holds for all λ-structures G and all tuples $\vec{v} \in V(G)^{|\vec{x}|}$: If $\Gamma(\vec{x})$ is applicable to $(G,\vec{v})$, then*

$$G \models \varphi^{-\Gamma}[\vec{v}] \iff \Gamma[G;\vec{v}] \models \varphi.$$

A proof of this fact for first-order logic can be found in [22]. The proof for the other logics considered here is an easy adaptation of the one for first-order logic.

2.5 Definable canonisation

A *canonisation mapping* for a class of $\mathscr{C}$ graphs associates with every graph $G \in \mathscr{C}$ an *ordered copy* of G, that is, an ordered graph $(H,\leq)$ such that $H \cong G$. We are interested in canonisation mappings definable in the logic IFP+C by syntactical interpretations of $\{E,\leqslant\}$ in $\{E\}$. The easiest way to define a canonisation mapping is by defining a linear order $\leq$ on the universe of a structure G and then take $(G,\leq)$ as the canonical copy. However, defining an ordered copy of a structure is not the same as defining a linear order on the universe, as the following example illustrates:

Example 2.6. Let $\mathscr{K}$ be the class of all complete graphs. It is easy to see that there is no IFP+C-formula $\varphi(x_1,x_2)$ such that for all $K \in \mathscr{K}$ the binary relation $\varphi[K;x_1,x_2]$ is a linear order of $V(K)$.

However, there is an FO+C-definable canonisation mapping for the class $\mathscr{K}$: Let

$$\Gamma = \big(\gamma_{app}, \gamma_V(\vec{y}), \gamma_{\approx}(y_1,y_2), \gamma_E(y_1,y_2), \gamma_{\leqslant}(y_1,y_2)\big)$$

be the numerical FO+C-interpretation of $\{E,\leqslant\}$ in $\{E\}$ defined by:

- $\gamma_{app} := \forall x\, x = x$;
- $\gamma_V(y) := 1 \leq y \wedge y \leq \mathsf{ord}$, where $\mathsf{ord} := \#x\, x = x$;
- $\gamma_{\approx}(y_1,y_2) := y_1 = y_2$;
- $\gamma_E(y_1,y_2) := \neg y_1 = y_2$;
- $\gamma_{\leqslant}(y_1,y_2) := y_1 \leq y_2$.

It is easy to see that the mapping $K \mapsto \Gamma[K]$ is a canonisation mapping for the class $\mathscr{K}$.

Our notion of *definable canonisation* slightly relaxes the requirement of defining a canonisation mapping; instead of just one ordered copy, we associate with each structure a parameterised family of polynomially many ordered copies.

Definition 2.7. (1) Let $\Gamma(\vec{x})$ be an L-interpretation of $\{E,\leqslant\}$ in $\{E\}$. Then $\Gamma(\vec{x})$ *canonises* a graph G if there is at least one tuple $\vec{v} \in V(G)^{|\vec{x}|}$ such that $\Gamma(\vec{x})$ is applicable to $(G,\vec{v})$, and for all tuples $\vec{v} \in V(G)^{\vec{x}}$ such that $\Gamma(\vec{x})$ is applicable to $(G,\vec{v})$ it holds that $\Gamma[G;\vec{v}]$ is an ordered copy of G.

(2) A class $\mathscr{C}$ of graphs admits L-*definable canonisation* if there is an L-interpretation $\Gamma(\vec{x})$ of $\{E, \leqslant\}$ in $\{E\}$ that canonises all $G \in \mathscr{C}$.

The following well-known fact is a consequence of the Immerman-Vardi Theorem. It is used, at least implicitly, in [30, 31, 34, 47]:

Fact 2.8. *Let $\mathscr{C}$ be a class of graphs that admits* IFP+C*-definable canonisation. Then* IFP+C *captures* PTIME *on $\mathscr{C}$.*

3 Negative results

In this section, we prove that IFP+C does not capture PTIME on the classes of chordal graphs and line graphs. Actually, our proof yields a more general result: Any logic that captures PTIME on any of these two classes and that is "closed under first-order reductions" captures PTIME on the class of all graphs. It will be obvious what we mean by "closed under first-order reductions" from the proofs, and it is also clear that most "natural" logics will satisfy this closure condition. It follows from our constructions that if there is a logic capturing PTIME on one of the two classes, then there is a logic capturing PTIME on all graphs.

Our negative results for IFP+C are based on the following theorem:

Fact 3.1 (Cai, Fürer, and Immerman [9]). *There is a* PTIME*-decidable property $\mathscr{P}_{\text{CFI}}$ of graphs that is not definable in* IFP+C*.*

Without loss of generality we assume that all $G \in \mathscr{P}_{\text{CFI}}$ are connected and of order at least 4.

3.1 Chordal graphs

Let us denote the class of chordal graphs by $\mathscr{CD}$.

For every graph G, we define a graph $\hat{G}$ as follows:

- $V(\hat{G}) := V(G) \cup \{v_e \mid e \in E(G)\}$, where for each $e \in E(G)$ we let v_e be a new vertex;
- $E(\hat{G}) := \binom{V(G)}{2} \cup \big\{\{v, v_e\} \bigm| v \in V(G), e \in E(G), v \in e\big\}$.

The following lemmas collect the properties of the transformation $G \mapsto \hat{G}$ that we need here. We leave the straighforward proofs to the reader.

Lemma 3.2. *For every graph G the graph $\hat{G}$ is chordal.*

Note that for the graphs K_2 and $I_3 := \big([3], \emptyset\big)$ it holds that $\hat{K}_2 \cong \hat{I}_3 \cong K_3$. It turns out that K_2 and I_3 are the only two nonisomorphic graphs that have isomorphic images under the mapping $G \mapsto \hat{G}$. It is easy to verify this by observing that for G with $|G| \geq 4$ and $v \in V(\hat{G})$, it holds that $v \in V(G)$ if and only if $\deg(v) \geq 3$. Let $\hat{\mathscr{G}}$ be the class of all graphs H such that $H \cong \hat{G}$ for some graph G.

Lemma 3.3. *The class $\hat{\mathscr{G}}$ is polynomial time decidable. Furthermore, there is a polynomial time algorithm that, given a graph $H \in \hat{\mathscr{G}}$, computes the unique (up to isomorphism) graph $G \in \mathscr{G} \setminus \{K \mid K \cong K_2\}$ with $\hat{G} \cong H$.*

Lemma 3.4. *There is an* FO*-interpretation $\hat{\Gamma}$ of $\{E\}$ in $\{E\}$ such that for all graph G it holds that $\hat{\Gamma}[G] \cong \hat{G}$.*

Theorem 3.5. IFP+C *does not capture* PTIME *on the class $\mathscr{CD}$ of chordal graphs.*

Proof. Let $\mathscr{P}_{\text{CFI}}$ be the graph property of Fact 3.1 that separates PTIME from IFP+C. Note that $K_2 \notin \mathscr{P}_{\text{CFI}}$ by our assumption that all graphs in $\mathscr{P}_{\text{CFI}}$ have order at least 4. By Lemma 3.3, the class $\hat{\mathscr{P}} := \{H \mid H \cong \hat{G}$ for some $G \in \mathscr{P}_{\text{CFI}}\}$ is a polynomial time decidable subclass of $\mathscr{CD}$.

Suppose for contradiction that IFP+C captures polynomial time on $\mathscr{CD}$. Then by $(\text{G.2})_{\mathscr{CD}}$ there is an IFP+C-sentence φ such that for all chordal graphs G it holds that $G \models \varphi \iff G \in \hat{\mathscr{P}}$. We apply the Lemma on Syntactical Interpretations to φ and the interpretation $\hat{\Gamma}$ of Lemma 3.4 and obtain an IFP+C-sentence $\varphi^{-\hat{\Gamma}}$ such that for all graphs G it holds that

$$G \models \varphi^{-\hat{\Gamma}} \iff \hat{G} \cong \hat{\Gamma}[G] \models \varphi.$$

Thus $\varphi^{-\hat{\Gamma}}$ defines $\mathscr{P}_{\text{CFI}}$, which is a contradiction. $\square$

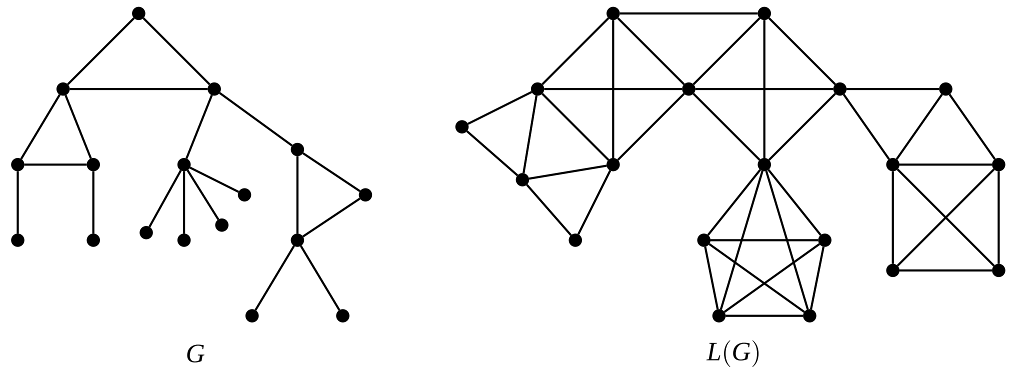

Figure 4.1. A graph G and its line graph $L(G)$, which is chordal

3.2 Line graphs

Let $\mathscr{L}$ denote the class of all line graphs, or more precisely, the class of all graphs L such that there is a graph G with $L \cong L(G)$. Observe that a triangle and a claw have the same line graph, a triangle. Whitney [60] proved that for all nonisomorphic connected graphs G, H except the claw and triangle, the line graphs of G and H are nonisomorphic. The following fact, corresponding to Lemma 3.3, is essentially an algorithmic version of Whitney's result:

Fact 3.6 (Roussopoulos [57]). *The class $\mathscr{L}$ is polynomial time decidable. Furthermore, there is a polynomial time algorithm that, given a connected graph $H \in \mathscr{L}$, computes the unique (up to isomorphism) graph $G \in \mathscr{G} \setminus \{K \mid K \cong K_3\}$ with $L(G) \cong H$.*

Lemma 3.7. *There is an* FO*-interpretation Λ of $\{E\}$ in $\{E\}$ such that for all graph G it holds that $\Lambda[G] \cong L(G)$.*

Proof. We define $\Lambda := \big(\lambda_{\text{app}}, \lambda_V(y_1,y_2), \lambda_\approx(y_1,y_2,y_1',y_2'), \lambda_E(y_1,y_2,y_1',y_2')\big)$ by:

- $\lambda_{\text{app}} := \forall x\, x = x$;
- $\lambda_V(y_1,y_2) := E(y_1,y_2)$;
- $\lambda_\approx(y_1,y_2,y_1',y_2') := (y_1 = y_1' \wedge y_2 = y_2') \vee (y_1 = y_2' \wedge y_2 = y_1')$;
- $\lambda_E(y_1,y_2,y_1',y_2') := (y_1 = y_1' \wedge \neg y_2 = y_2') \vee (y_2 = y_2' \wedge \neg y_1 = y_1') \vee (y_1 = y_2' \wedge \neg y_2 = y_1') \vee (y_2 = y_1' \wedge \neg y_2 = y_1')$.

□

Theorem 3.8. IFP+C *does not capture* PTIME *on the class $\mathscr{L}$ of line graphs.*

Proof. The proof is completely analogous to the proof of Theorem 3.5, using Fact 3.6 and Lemma 3.7 instead of Lemmas 3.3 and 3.4. □

4 Capturing polynomial time on chordal line graphs

In this section, we shall prove that IFP+C captures PTIME on the class $\mathscr{CD} \cap \mathscr{L}$ of graphs that are both chordal and line graphs. As we will see, such graphs have a simple treelike structure. We can exploit this structure and canonise the graphs in $\mathscr{CD} \cap \mathscr{L}$ in a similar way as trees or graphs of bounded tree width.

Example 4.1. Figure 4.1 shows an example of a chordal line graph.

4.1 On the structure of chordal line graphs

It is a well-known fact that chordal graphs can be decomposed into cliques arranged in a tree-like manner. To state this formally, we review tree decompositions of graphs. Let G be a graph. A *tree decomposition* of a graph G is a pair (T,β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$ is a mapping such that the following two conditions are satisfied:

(T.1) For every $v \in V(G)$ the set $\{t \in V(T) \mid v \in \beta(t)\}$ is connected in T.

(T.2) For every $e \in E(G)$ there is a $t \in V(T)$ such that $e \subseteq \beta(t)$.

The sets $\beta(t)$, for $t \in V(T)$, are called the *bags* of the decomposition. It will be convenient for us to always assume the tree T in a tree decompition to be rooted. This gives us the partial tree order $\trianglelefteq^T$. We introduce some additional notation. Let (T,β) be a tree decomposition of a graph G. For every $t \in V(T)$ we let:

$$\gamma(t) := \bigcup_{u \in V(T) \text{ with } t \trianglelefteq^T u} \beta(u),$$

The set $\gamma(t)$ is called the *cone* of (T,β) at t. It easy to see that for every $t \in V(T) \setminus \{r(T)\}$ with parent s the set $\beta(t) \cap \beta(s)$ separates $\gamma(t)$ from $V(G) \setminus \gamma(t)$. Furthermore, for every clique X of G there is a $t \in V(T)$ such that $X \subseteq \beta(t)$. (See Diestel's textbook [20] for proofs of theses facts and background on tree decompositions.) Another useful fact is that every tree decomposition (T,β) of a graph G can be transformed into a tree decomposition (T',β') such that for all $t' \in V(T')$ there exists a $t \in V(T)$ such that $\beta'(t') = \beta(t)$, and for all $t,u \in V(T')$ with $t \neq u$ it holds that $\beta'(t) \not\subseteq \beta'(u)$.

Fact 4.2. *A nonempty graph G is chordal if and only if G has a tree decomposition into cliques, that is, a tree decomposition (T,β) such that for all $t \in V(T)$ the bag $\beta(t)$ is a clique of G.*

For a graph G, we let $MCL(G)$ be the set of all maximal cliques in G with respect to set inclusion. If we combine Fact 4.2 with the observations about tree decomposition stated before the fact, we obtain the following lemma:

Lemma 4.3. *Let G be a nonempty chordal graph. Then G has a tree decomposition (T,β) with the following properties:*

(i) For every $t \in V(T)$ it holds that $\beta(t) \in MCL(G)$.

(ii) For every $X \in MCL(G)$ there is exactly one $t \in V(T)$ such that $\beta(t) = X$.

We call a tree decomposition satisfying conditions (i) and (ii) a good tree decomposition *of G.*

Let us now turn to line graphs. Let $L := L(G)$ be the line graph of a graph G. For every $v \in V(G)$, let $X(v) := \{e \in E(G) \mid v \in e\} \subseteq V(L)$. Unless v is an isolated vertex, $X(v)$ is a clique in L. Furthermore, we have

$$L = \bigcup_{v \in V(G)} L[X(v)].$$

Observe that for all $v,w \in V(G)$, if $e := \{v,w\} \in E(G)$ then $X(v) \cap X(w) = \{e\}$, and if $\{v,w\} \notin E(G)$ then $X(v) \cap X(w) = \emptyset$. The following proposition, which is probably well-known, characterises the line graphs that are chordal:

Proposition 4.4. *Let $L = L(G) \in \mathscr{L}$. Then*

$$L \in \mathscr{CD} \iff \text{all cycles in } G \text{ are triangles}.$$

Note that on the right hand side, we do not only consider chordless cycles.

Proof. For the forward dircetion, suppose that $L \in \mathscr{CD}$, and let $C \subseteq G$ be a cycle. Then $L[E(C)]$ is a chordless cycle in L. Hence $|C| \leq 3$, that is, C is a triangle.

For the backward direction, suppose that all cycles in G are triangles, and let $C \subseteq L$ be a chordless cycle of length k. Let $e_1, \dots, e_k$ be the vertices of C in cyclic order. To simplify the notation, let $e_0 := e_k$. Then for all $i \in [k]$ it holds that $\{e_{i-1}, e_i\} \in E(L)$ and thus $e_{i-1} \cap e_i \neq \emptyset$. Let $v_0, v_1 \in V(G)$ such that $e_1 = \{v_0, v_1\}$, and for $i \in [2,k]$, let $v_i \in e_i \setminus e_{i-1}$. Then $v_i \neq v_j$ for all $j \in [i-2]$, and if $i < k$ even for $j \in [0, i-2]$, because the cycle C is chordless and thus $e_i \cap e_j = \emptyset$. Furthermore, $v_k = v_0$. Thus $\{v_1, \ldots, v_k\}$ is the vertex set of a cycle in G, and we have $k = 3$. □

Lemma 4.5. *Let $L = L(G) \in \mathscr{CD} \cap \mathscr{L}$, and let $X \in MCL(L)$ and $e = \{v, w\} \in X$. Then $X = X(v)$ or $X = X(w)$ or there is an $x \in V(G)$ such that $\{x, v\}, \{x, w\} \in E(G)$ and $X = \big\{e, \{x, v\}, \{x, w\}\big\}$.*

Proof. For all $f \in X$, either $v \in f$ or $w \in f$, because f is adjacent to e. Hence $X \subseteq X(v) \cup X(w)$. If $X \subseteq X(v)$, then $X = X(v)$ by the maximality of X. Similarly, if $X \subseteq X(w)$ then $X = X(w)$. Suppose that $X \setminus X(v) \neq \emptyset$ and $X \setminus X(w) \neq \emptyset$. Let $f \in X \setminus X(v)$ and $g \in X \setminus X(w)$. As X is a clique, we have $\{f, g\} \in E(L)$ and thus $f \cap g \neq \emptyset$. Hence there is an $x \in V(G)$ such that $f = \{x, w\}$ and $g = \{x, v\}$. Furthermore, $X = \{e, f, g\}$. To see this, let $h \in X$. Then $\{h, e\} \in E(L)$ and thus $v \in h$ or $w \in h$. Say, $v \in h$. If $w \in h$, then $h = e$. Otherwise, we have $x \in h$, because h is adjacent to g. Thus $h = g$. □

Lemma 4.6. *Let $L \in \mathscr{CD} \cap \mathscr{L}$, and let $X_1, X_2 \in MCL(L)$ be distinct. Then $|X_1 \cap X_2| \leq 2$.*

Proof. Let $L = L(G)$ for some graph G. Suppose for contradiction that $|X_1 \cap X_2| \geq 3$. Then $|X_1|, |X_2| \geq 4$, because X_1 and X_2 are distinct maximal cliques. By Lemma 4.5, it follows that there are vertices $v_1, v_2 \in V(G)$ such that $X_1 = X(v_1)$ and $X_2 = X(v_2)$, which implies $|X_1 \cap X_2| \leq 1$. This is a contradiction. □

Lemma 4.7. *Let $L \in \mathscr{CD} \cap \mathscr{L}$, and let $X_1, X_2, X_3 \in MCL(L)$ be pairwise distinct such that $X_1 \cap X_2 \cap X_3 \neq \emptyset$. Then there are i, j, k such that $\{i, j, k\} = [3]$ and $X_i \subseteq X_j \cup X_k$ and $|X_i| = 3$.*

Proof. Let $L = L(G)$ for some graph G. Let $e \in X_1 \cap X_2 \cap X_3$. Suppose that $e = \{v, w\} \in E(G)$. As the cliques X_1, X_2, X_3 are distinct, it follows from Lemma 4.5 that there is an $i \in [3]$ and an $x \in V(G)$ such that $X_i = \big\{e, \{x, v\}, \{x, w\}\big\}$. Choose such i and x.

Claim 1. For all $j \in [3] \setminus \{i\}$, either $X_j = X(v)$ or $X_j = X(w)$.

Proof. Suppose for contradiction that $X_j \neq X(v)$ and $X_j \neq X(w)$. Then by Lemma 4.5, there exists a $y \in V(G)$ such that $\{y, v\}, \{y, w\} \in E(G)$ and $X_j = \big\{e, \{y, v\}, \{y, w\}\big\}$. But then

$$L\big[\{y, v\}, \{v, x\}, \{x, w\}, \{w, y\}\big]$$

is a chordless cycle in L, which contradicts L being chordal. ⌟

Thus there are j, k such that $\{i, j, k\} = [3]$ and $X_j = X(v)$ and $X_k = X(w)$. Then $X_i \subseteq X_j \cup X_k$. □

Lemma 4.8. *Let $L \in \mathscr{CD} \cap \mathscr{L}$. Then every good tree decomposition (T, β) of L satisfies the following conditions (in addition to conditions (i) and (ii) of Lemma 4.3):*

(iii) For all $t \in V(T)$,

- *either $|\beta(t)| = 3$ and t has at most three neighbours in T (the* neighbours *of a node are its children and the parent),*
- *or for all distinct neighbours u, u' of t in T it holds that $\beta(u) \cap \beta(u') = \emptyset$.*

(iv) For all $t, u \in V(T)$ with $t \neq u$ it holds that $|\beta(t) \cap \beta(u)| \leq 2$.

Proof. Let (T, β) be a good tree decomposition of L. Such a decomposition exists because L is chordal. As all bags of the decomposition are maximal cliques of L, condition (iii) follows from Lemma 4.7 and condition (iv) follows from Lemma 4.6. □

4.2 Canonisation

Theorem 4.9. *The class $\mathscr{CD} \cap \mathscr{L}$ of all chordal line graphs admits* IFP+C*-definable canonisation.*

Corollary 4.10. IFP+C *captures* PTIME *on the class of all chordal line graphs.*

Proof of Theorem 4.9. The proof resembles the proof that classes of graphs of bounded tree width admit IFP+C-definable canonisation [34] and also the proof of Theorem 7.2 (the "Second Lifting Theorem") in [31]. Both of these proofs are generalisations of the simple proof that the class of trees admits IFP+C-definable canonisation (see, for example, [29]). We shall describe an inductive construction that associates with each chordal line graph G a canonical copy G' whose universe is an initial segment of the natural numbers. For readers with some experience in finite model theory, it will be straightforward to formalise the construction in IFP+C. We only describe the canonisation of *connected* chordal line graphs that are not complete graphs. It is easy to extend it to arbitrary chordal line graphs. For complete graphs, which are chordal line graphs, cf. Example 2.6

To describe the construction, we fix a connected graph $G \in \mathscr{CD} \cap \mathscr{L}$ that is not a complete graph. Note that this implies $|G| \geq 3$. Let (T, β^T) be a good tree decomposition of G. As G is not a complete graph, we have $|T| \geq 2$. Without loss of generality we may assume that the root $r(T)$ has exactly one child in T, because every tree has at least one node of degree at most 1 and properties (i), (ii) of a good decomposition do not depend on the choice of the root. It will be convenient to view the rooted tree T as a directed graph, where the edges are directed from parents to children.

Let U be the set of all triples $(u_1, u_2, u_3) \in V(G)^3$ such that $u_3 \neq u_1, u_2$ (possibly, $u_1 = u_2$), and there is a unique $X \in MCL(G)$ such that $u_1, u_2, u_3 \in X$. For all $\vec{u} = (u_1, u_2, u_3) \in U$, let $A(\vec{u})$ be the connected component of $G \setminus \{u_1, u_2\}$ that contains u_3 (possibly, $A(\vec{u}) = G \setminus \{u_1, u_2\}$). We define mappings $\sigma^U, \alpha^U, \gamma^U, \beta^U : U \to 2^{V(G)}$ as follows: For all $\vec{u} = (u_1, u_2, u_3) \in U$, we let $\sigma^U(\vec{u}) := \{u_1, u_2\}$ and $\alpha^U(\vec{u}) := V(A(\vec{u}))$. We let $\gamma^U(\vec{u}) := \sigma^U(\vec{u}) \cup \alpha^U(\vec{u})$, and we let $\beta^U(\vec{u})$ the unique $X \in MCL(G)$ with $u_1, u_2, u_3 \in X$. We define a partial order $\trianglelefteq$ on U by letting $\vec{u} \trianglelefteq \vec{v}$ if and only if $\vec{u} = \vec{v}$ or $\alpha(\vec{u}) \supset \alpha(\vec{v})$. We let F be the successor relation of $\trianglelefteq$, that is, $(\vec{u}, \vec{v}) \in F$ if $\vec{u} \triangleleft \vec{v}$ and there is no $\vec{w} \in U \setminus \{\vec{u}, \vec{v}\}$ such that $\vec{u} \triangleleft \vec{w} \triangleleft \vec{v}$. Finally, we let $D := (U, F)$. Then D is a directed acyclic graph. It is easy to verify that for all $\vec{u} \in U$ we have

$$\beta^U(\vec{u}) = \gamma^U(\vec{u}) \setminus \bigcup_{\vec{v} \in N^D(\vec{u})} \alpha^U(\vec{v}), \tag{4.1}$$

where $N^D(\vec{u}) = \{\vec{v} \in U \mid (\vec{u}, \vec{v}) \in F\}$.

Recall that we also have mappings $\beta^T, \gamma^T : V(T) \to 2^{V(G)}$ derived from the tree decomposition. We define a mapping $\sigma^T : V(T) \to 2^{V(G)}$ as follows:

- For a node $t \in V(T) \setminus \{r(T)\}$ with parent s, we let $\sigma^T(t) := \beta^T(t) \cap \beta^T(s)$.
- For the root $r := r(T)$, we first define a set $S \subseteq V(G)$ by letting $S := \beta^T(r) \setminus \beta^T(t)$, where t is the unique child of r. (Remember our assumption that r has exactly one child.) Then if $|S| \geq 2$, we choose distinct $v, v' \in S$ and let $\sigma^T(r) := \{v, v'\}$, and if $|S| = 1$ we let $\sigma^T(r) := S$.

Note that $\beta^T(t) \setminus \sigma^T(t) \neq \emptyset$ and $1 \leq |\sigma^T(t)| \leq 2$ for all $t \in V(T)$. For the root, this follows immediately from the definition of $\sigma^T(t)$, and for nodes $t \in V(T) \setminus \{r(T)\}$ it follows from Lemma 4.8. We define a mapping $\alpha^T : V(T) \to 2^{V(G)}$ by letting $\alpha^T(t) := \gamma^T(t) \setminus \sigma^T(t)$ for all $t \in V(T)$. We define a mapping $g : V(T) \to U$ by choosing, for every node $t \in V(T)$, vertices u_1, u_2 such that $\sigma^T(t) = \{u_1, u_2\}$ (possible $u_1 = u_2$) and a vertex $u_3 \in \beta(t) \setminus \sigma(t)$ and letting $g(t) := (u_1, u_2, u_3)$. Note that $(u_1, u_2, u_3) \in U$, because $\beta^T(t)$ is the uniqe maximal clique in $MCL(G)$ that contains u_1, u_2, u_3.

Claim 1. The mapping g is a directed graph embedding of T into D. Furthermore, for all $t \in V(T)$ it holds that $\alpha^T(t) = \alpha^U(g(t))$, $\beta^T(t) = \beta^U(g(t))$, $\gamma^T(t) = \gamma^U(g(t))$, and $\sigma^T(t) = \sigma^U(g(t))$.

Proof. We leave the straightforward inductive proof to the reader. ⌟

Let $\vec{u}_0 := g(r(T))$, and let U_0 be the subset of U consisting of all $\vec{u} \in U$ such that $\vec{u}_0 \trianglelefteq \vec{u}$. Let F_0 be the restriction of F to U_0 and $D_0 := (U_0, F_0)$. Note that U_0 is upward closed with respect to $\trianglelefteq$ and that $g(T) \subseteq D_0$.

Claim 2. There is mapping $h : U_0 \to V(T)$ such that h is a directed graph homomorphism from D_0 to T and $h \circ g$ is the identity mapping on $V(T)$. Furthermore, for all $\vec{u} \in U_0$ it holds that $\alpha^U(\vec{u}) = \alpha^T(h(\vec{u}))$, $\beta^U(\vec{u}) = \beta^T(h(\vec{u}))$, $\gamma^U(\vec{u}) = \gamma^T(h(\vec{u}))$, and $\sigma^U(\vec{u}) = \sigma^T(h(\vec{u}))$.

Proof. We define h by induction on the partial order $\trianglelefteq$. The unique $\trianglelefteq$-minimal element of U_0 is $\vec{u}_0$. We let $h(\vec{u}_0) := r(T)$. Now let $\vec{v} = (v_1, v_2, v_3) \in U_0$, and suppose that $h(\vec{u})$ is defined for all $\vec{u} \in U_0$ with $\vec{u} \triangleleft \vec{v}$. Let $\vec{u} \in U_0$ such that $(\vec{u}, \vec{v}) \in F_0$, and let $s := h(\vec{u})$. By the induction hypothesis, we have $\alpha^U(\vec{u}) = \alpha^T(s)$, $\beta^U(\vec{u}) = \beta^T(s)$, $\gamma^U(\vec{u}) = \gamma^T(s)$, and $\sigma^U(\vec{v}) = \sigma^T(s)$. The set $\alpha^U(\vec{v})$ is the vertex set of a connected component of $G \setminus \sigma^U(\vec{v})$ which is contained in $\alpha^U(\vec{u}) \subseteq \gamma^U(\vec{u}) = \gamma^T(s)$, and by (4.1) it holds that $\alpha^U(\vec{v}) \cap \beta^U(\vec{u}) = \emptyset$. Hence there is a child t of s such that $\alpha^U(\vec{v}) \subseteq \alpha^T(t)$. Let $\vec{v}' := g(t)$. If $\alpha^U(\vec{v}) \subset \alpha^T(t) = \alpha^U(\vec{v}')$, then $\vec{u} \triangleleft \vec{v}' \triangleleft \vec{v}$, which contradicts $(\vec{u}, \vec{v}) \in F$. Hence $\alpha^U(\vec{v}) = \alpha^T(t)$ and thus $\sigma^U(\vec{v}) = \sigma^T(t)$. This also implies $\gamma^U(\vec{v}) = \gamma^T(t)$ and $\beta^U(\vec{v}) = \beta^T(t)$. We let $h(\vec{v}) := t$.

To prove that h is really a homomorphism, it remains to prove that for all $\vec{u}' \in U_0$ with $(\vec{u}', \vec{v}) \in F_0$ we also have $h(\vec{u}') = s$. So let $\vec{u}' \in U_0$ with $(\vec{u}', \vec{v}) \in F_0$, and let $s' = h(\vec{u}')$. Suppose for contradiction that $s \neq s'$. If $s' \triangleleft^T s$ then $\alpha^U(\vec{u}') \supset \alpha^U(\vec{u})$ and thus $\vec{u}' \triangleleft \vec{u}$, which contradicts $(\vec{u}', \vec{v}) \in F_0$. Thus $s' \not\trianglelefteq^T s$, and similarly $s \not\trianglelefteq^T s'$. But then both $\sigma^T(s)$ and $\sigma^T(s')$ separate $\gamma^T(s)$ from $\gamma^T(s')$ in G. This contradicts $\alpha^U(\vec{v}) \subseteq \alpha^T(s) \cap \alpha^T(s') \subseteq \big(\gamma^T(s) \cap \gamma^T(s')\big) \setminus \big(\sigma^T(s) \cup \sigma^T(s')\big)$. ⌟

Thus essentially, the "treelike" decomposition (D_0, β^U) is the same as the tree decomposition (T, β^T). However, the decomposition (D_0, β^U) is IFP-definable with three parameters fixing the tuple $\vec{u}_0 = g(r(T))$.

Let us now turn to the canonisation. For every $\vec{u} \in U_0$, we let $G(\vec{u}) := G[\gamma(\vec{u})]$. Then $G = G(\vec{u}_0)$. We inductively define for every $\vec{u} = (u_1, u_2, u_3) \in U_0$ a graph $H(\vec{u})$ with the following properties:

(i) $V\big(H(\vec{u})\big) = [n_{\vec{u}}]$, where $n_{\vec{u}} := |\gamma(\vec{u})| = \big|V\big(G_{\vec{u}}\big)\big|$.

(ii) There is an isomorphism $f_{\vec{u}}$ from $G(\vec{u})$ to $H(\vec{u})$ such that if $u_1 \neq u_2$ it holds that $f_{\vec{u}}(u_1) = 1$ and $f_{\vec{u}}(u_2) = 2$, and if $u_1 = u_2$ it holds that $f_{\vec{u}}(u_1) = 1$.

For the induction basis, let $\vec{u} \in U_0$ with $N^{D_0}(\vec{u}) = \emptyset$. Then $\gamma^U(\vec{u}) = \beta^U(\vec{u})$, and $G(\vec{u}) = K[\beta^U(\vec{u})]$. We let $n := n_{\vec{u}} = |\beta^U(\vec{u})|$ and $H(\vec{u}) := K_n$. The (i) and (ii) are obviously satisfied.

For the induction step, let $\vec{u} \in U_0$ and $N^{D_0}(\vec{u}) = \{\vec{v}^1, \ldots, \vec{v}^n\} \neq \emptyset$. It follows from Claim 2 that for all $i, j \in [n]$, either $\gamma(\vec{v}^i) = \gamma(\vec{v}^j)$ or $\gamma(\vec{v}^i) \cap \gamma(\vec{v}^j) = \sigma(\vec{v}^i) \cap \sigma(\vec{v}^j) \subseteq \beta(\vec{u})$. We may assume without loss of generality that there are $i_1, \ldots, i_m \in [n]$ such that $i_1 < i_2 < \ldots < i_m$ and for all $j, j' \in [m]$ with $j \neq j'$ we have $\gamma(\vec{v}^{i_j}) \neq \gamma(\vec{v}^{i_{j'}})$ and for all $j \in [m]$, $i \in [i_j, i_{j+1} - 1]$ we have $\gamma(\vec{v}^i) = \gamma(\vec{v}^{i_j})$. Here and in the following we let $i_{m+1} := n + 1$.

The class of all graphs whose vertex set is a subset of $\mathbb{N}$ may be ordered lexicographically; we let $H \leq_{\text{s-lex}} H'$ if either $V(H)$ is lexicographically smaller than $V(H')$, that is, the first element of the symmetric difference $V(H) \triangle V(H')$ belongs to $V(H')$, or $V(H) = V(H')$ and $E(H)$ is lexicographically smaller than $E(H')$ with respect to the lexicographical ordering of unordered pairs of natural numbers, or $H = H'$. Without loss of generality we may assume that for each $j \in [m]$ it holds that

$$H(\vec{v}^{i_j}) \leq_{\text{s-lex}} H(\vec{v}^{i_j+1}) \leq_{\text{s-lex}} H(\vec{v}^{i_j+2}) \leq_{\text{s-lex}} \ldots \leq_{\text{s-lex}} H(\vec{v}^{i_{j+1}-1})$$

and, furthermore,

$$H(\vec{v}^{i_1}) \leq_{\text{s-lex}} H(\vec{v}^{i_2}) \leq_{\text{s-lex}} \ldots \leq_{\text{s-lex}} H(\vec{v}^{i_m}) \tag{4.2}$$

Note that, even though the graphs $G(\vec{v}^{i_1}), G(\vec{v}^{i_2}), \ldots, G(\vec{v}^{i_m})$ are vertex disjoint subgraphs of $G(\vec{u})$, they may be isomorphic, and hence not all of the inequalities in (4.2) need to be strict. For all $j \in [m]$, let $\vec{v}_j := \vec{v}^{i_j}$ and $G_j := G(\vec{v}_j)$ an $H_j := H(\vec{v}_j)$. Then $H_1 \leq_{\text{s-lex}} H_2 \leq_{\text{s-lex}} \ldots \leq_{\text{s-lex}} H_m$. Let $j_1, \ldots, j_\ell \in [m]$ such that $j_1 < j_2 < \ldots < j_\ell$ and $H_j = H_{j_i}$ for all $i \in [\ell]$, $j \in [j_i, j_{i+1} - 1]$, where $j_{\ell+1} = m + 1$, and $H_{j_i} \neq H_{j_{i+1}}$ for all $i \in [\ell - 1]$. For all $i \in [\ell]$, let $J_i := H_{j_i}$. Furthermore, let $n_i := |J_i|$ and $k_i := j_{i+1} - j_i$ and $q_i := |\sigma^U(\vec{v}^{i_j})|$ and

$$q := \left|\beta^U(\vec{u}) \setminus \bigcup_{j=1}^{m} \beta^U(\vec{v}_j)\right|.$$

Case 1: For all neighbours t,t' of $h(\vec{u})$ in the undirected tree underlying T it holds that $\beta^T(t)\cap\beta^T(t')=\emptyset$. We define $H(\vec{u})$ by first taking a complete graph K_q, then k_1 copies of J_1, then k_2 copies of J_2, et cetera, and finally k_ℓ copies of J_ℓ. The universes of all these copies are disjoint, consecutive intervals of natural numbers. Let K be the union of $[q]$ with the first q_i vertices of each of the k_i copies of J_i for all $i\in[\ell]$. Then K is the set of vertices of $H(\vec{u})$ that corresponds to the clique $\beta(\vec{u})$. We add edges among the vertices in K to turn it into a clique. It is not hard to verify that the resulting structure satisfies (i) and (ii).

Case 2: There are neighbours t,t' of $h(\vec{u})$ in the undirected tree underlying T such that $\beta^T(t)\cap\beta^T(t')\neq\emptyset$. Then by Lemma 4.8(iii) we have $|\beta^U(\vec{u})|=3$, and $h(\vec{u})$ has at most two children. Hence $m\leq 2$, and essentially this means we only have two possibilities of how to combine the parts H_1,H_2 to the graph $H(\vec{u})$; either H_1 comes first or H_2. We choose the lexicographically smaller possibility. We omit the details.

This completes our description of the construction of the graphs $H(\vec{u})$.

It remains to prove that $H(\vec{u})$ is IFP+C-definable. We first define IFP-formulas $\theta_U(\vec{x})$, $\theta_F(\vec{x},\vec{y})$, $\theta_\alpha(\vec{x},y)$, $\theta_\beta(\vec{x},y)$, $\theta_\gamma(\vec{x},y)$, $\theta_\sigma(\vec{x},y)$ such that

$$\begin{aligned} U &= \left\{\vec{u}\in V(G)^3 \;\middle|\; G\models\theta_U[\vec{u}]\right\},\\ F &= \left\{(\vec{u},\vec{v})\in U^2 \;\middle|\; G\models\theta_F[\vec{u},\vec{v}]\right\},\\ \alpha^U(\vec{u}) &= \left\{v\in V(G) \;\middle|\; G\models\theta_\alpha[\vec{u},v]\right\} \qquad\text{for all } \vec{u}\in U,\end{aligned}$$

and similarly for β,γ,σ. Then we define formulas $\theta_U^0(\vec{x}_0,\vec{x})$, $\theta_F^0(\vec{x}_0,\vec{x})$ that define D_0. We have no canonical way of checking that a tuple $\vec{u}_0$ really is the image $g(r(T))$ of the root of a good tree decomposition, but all we need is that the graph $D^0(\vec{u}_0)$ with vertex set $\left\{\vec{u}\in V(G)^3 \;\middle|\; G\models\theta_U^0[\vec{u}_0,\vec{u}]\right\}$ and edge set $\left\{(\vec{u},\vec{v})\in U^2 \;\middle|\; G\models\theta_F[\vec{u}_0,\vec{u},\vec{v}]\right\}$ has the properties we derive from T being a good tree decomposition. In particular, if a node $\vec{u}$ has a child $\vec{v}$ with $\sigma^U(\vec{u})\cap\sigma^U(\vec{v})\neq\emptyset$ or children $\vec{v}_1\neq\vec{v}_2$ with $\sigma^U(\vec{v}_1)\cap\sigma^U(\vec{v}_1)\neq\emptyset$, then $|\beta^U(\vec{u})|\leq 3$. Once we have defined D^0, it is straightforward to formalise the definition of the graphs $H(\vec{u})$ in IFP+C and define an IFP+C-interpretation $\Gamma(\vec{x}_0)$ that canonises G. We leave the (tedious) details to the reader. □

Remark 4.11. Implicitly, the previous proof heavily depends on the concepts introduced in [31]. In particular, the definable directed graph D together with the definable mappings σ and α constitute a *definable tree decomposition*. However, our theorem does not follow directly from Theorem 7.2 of [31].

The class $\mathscr{CD}\cap\mathscr{L}$ of chordal line graphs is fairly restricted, and there may be an easier way to prove the canonisation theorem by using Proposition 4.4. The proof given here has the advantage that it generalises to the class of all chordal graphs that have a good tree decomposition where the bags of the neighbours of a node intersect in a "bounded way". We omit the details.

5 Further research

I mentioned several important open problems related to the quest for a logic capturing PTIME in the survey in Section 1. Further open problems can be found in [32]. Here, I will briefly discuss a few open problems related to classes closed under taking induced subgraphs, or equivalently, classes defined by excluding (finitely or infinitely many) induced subgraphs.

A fairly obvious generalisation of our positive capturing result is pointed out in Remark 4.11, but as far as I can see this does not include any other particularly interesting classes. I conjecture that the theorem generalises to all claw-free chordal graphs, that is, I conjecture that the class of claw-free chordal graphs admits IFP+C-definable canonisation. Another interesting class of chordal graphs closed under taking induced subgraphs is the class of all interval graphs. I also conjecture that the class of interval graphs admits IFP+C-definable canonisation. A very interesing and rich family of classes of graphs closed under taking induced subgraphs is the family of classes of graphs of bounded rank width [56], or equivalently, bounded clique width [13]. It is conceivable that IFP+C captures polynomial time on all classes of bounded rank width, but I have no clear intuition whether this is the case or not. To the best of my knowledge, currently it is not even know whether isomorphism testing for graphs of bounded rank width is in polynomial time.

Acknowledgements

I would like to thank Yijia Chen and Bastian Laubner for valuable comments on a earlier version of this paper.

References

[1] S. Abiteboul and V. Vianu. Non-deterministic languages to express deterministic transformations. In *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pages 218–229, 1990.

[2] A.V. Aho and J.D. Ullman. The universality of data retrieval languages. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 110–120, 1979.

[3] L. Babai, D.Yu. Grigoryev, and D.M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 310–324, 1982.

[4] L. Babai and E.M. Luks. Canonical labeling of graphs. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 171–183, 1983.

[5] L.W. Beineke. Characterizations of derived graphs. *Journal of Combinatorial Theory*, 9:129–135, 1970.

[6] A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100:141–187, 1999.

[7] A. Blass, Y. Gurevich, and S. Shelah. On polynomial time computation over unordered structures. *Journal of Symbolic Logic*, 67:1093–1125, 2002.

[8] H.L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *Journal of Algorithms*, 11:631–643, 1990.

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

[10] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[11] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.

[12] M. Chudnovsky and P. Seymour. The structure of claw-free graphs. In B.S. Webb, editor, *Surveys in Combinatorics*, volume 327 of *London Mathematical Society Lecture Note Series*, pages 153–171. Cambridge University Press, 2005.

[13] B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000.

[14] A. Dawar. Generalized quantifiers and logical reducibilities. *Journal of Logic and Computation*, 5:213–226, 1995.

[15] A. Dawar. A restricted second order logic for finite structures. In D. Leivant, editor, *Logic and Computational Complexity: International Workshop, LCC '94*, volume 960 of *Lecture Notes in Computer Science*, 1995.

[16] A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *Proceedings of the 24th IEEE Symposium on Logic in Computer Science*, pages 113–122, 2009.

[17] A. Dawar and L. Hella. The expressive power of finitely many generalized quantifiers. In *Proceedings of the 9th IEEE Symposium on Logic in Computer Science*, 1994.

[18] A. Dawar and D. Richerby. A fixed-point logic with symmetric choice. In M. Baaz and J.A. Makowsky, editors, *Proceedings of the 17th International Workshop on Computer Science Logic*, volume 2803 of *Lecture Notes in Computer Science*, pages 169–182. Springer-Verlag, 2003.

[19] A. Dawar, D. Richerby, and B. Rossman. Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs: (Extended abstract). *Electronic Notes on Theoretical Compututer Science*, 143:13–26, 2006.

[20] R. Diestel. *Graph Theory*. Springer-Verlag, 3rd edition, 2005.

[21] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 2nd edition, 1999.

[22] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.

[23] S. Evdokimov, M. Karpinski, and I. Ponomarenko. On a new high dimensional Weisfeiler-Lehman algorithm. *Journal of Algebraic Combinatorics*, 10:29–45, 1999.

[24] S. Evdokimov and I. Ponomarenko. On highly closed cellular algebras and highly closed isomorphism. *Electronic Journal of Combinatorics*, 6:#R18, 1999.

[25] I. S. Filotti and J. N. Mayer. A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 236–243, 1980.

[26] F. Gire and H.K. Hoang. An extension of fixpoint logic with a symmetry-based choice construct. *Information and Computation*, 144:40–65, 1998.

[27] E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Springer-Verlag, 2007.

[28] E. Grädel and M. Otto. On Logics with Two Variables. *Theoretical Computer Science*, 224:73–113, 1999.

[29] E. Grädel. Finite model theory and descriptive complexity. In E. Grädel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein, *Finite Model Theory and Its Applications*, chapter 3, pages 125–230. Springer-Verlag, 2007.

[30] M. Grohe. Fixed-point logics on planar graphs. In *Proceedings of the 13th IEEE Symposium on Logic in Computer Science*, pages 6–15, 1998.

[31] M. Grohe. Definable tree decompositions. In *Proceedings of the 23rd IEEE Symposium on Logic in Computer Science*, pages 406–417, 2008.

[32] M. Grohe. The quest for a logic capturing PTIME. In *Proceedings of the 23rd IEEE Symposium on Logic in Computer Science*, pages 267–271, 2008.

[33] M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *Proceedings of the 25th IEEE Symposium on Logic in Computer Science*, 2010. To appear.

[34] M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In C. Beeri and P. Buneman, editors, *Proceedings of the 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 70–82. Springer-Verlag, 1999.

[35] M. Grohe and O. Verbitzky. Testing graph isomorphism in parallel by playing a game. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 3–14. Springer-Verlag, 2006.

[36] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current trends in theoretical computer science*, pages 1–57. Computer Science Press, 1988.

[37] Y. Gurevich. Sequential abstract-state machines capture sequential algorithms. *ACM Transaction on Computational Logic*, 1:77–111, 2000.

[38] Y. Gurevich and S. Shelah. Fixed point extensions of first–order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[39] L. Hella. Definability hierarchies of generalized quantifiers. *Annals of Pure and Applied Logic*, 43:235–271, 1989.

[40] L. Hella, Ph.G. Kolaitis, and K. Luosto. Almost everywhere equivalence of logics in finite model theory. *Bulletin of Symbolic Logic*, 2:422–443, 1996.

[41] J. E. Hopcroft and R. Tarjan. Isomorphism of planar graphs (working paper). In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*. Plenum Press, 1972.

[42] J.E. Hopcroft and J.K. Wong. Linear time algorithm for isomorphism of planar graphs. In *Proceedings of the 6th ACM Symposium on Theory of Computing*, pages 172–184, 1974.

[43] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[44] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proceedings of the 2nd IEEE Symposium on Structure in Complexity Theory*, pages 194–202, 1987.

[45] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, 1987.

[46] N. Immerman. *Descriptive Complexity*. Springer-Verlag, 1999.

[47] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In A. Selman, editor, *Complexity theory retrospective*, pages 59–81. Springer-Verlag, 1990.

[48] J. Köbler and O. Verbitsky. From invariants to canonization in parallel. In E.A. Hirsch, A.A. Razborov, A.L. Semenov, and A. Slissenko, editors, *Proceedings of the Third International Computer Science Symposium in Russia*, volume 5010 of *Lecture Notes in Computer Science*, pages 216–227. Springer-Verlag, 2008.

[49] Stephan Kreutzer. Expressive equivalence of least and inflationary fixed-point logic. *Annals of Pure and Applied Logic*, 130(1-3):61–78, 2004.

[50] L. Libkin. *Elements of Finite Model Theory*. Springer-Verlag, 2004.

[51] E.M. Luks. Isomorphism of graphs of bounded valance can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.

[52] G. L. Miller. Isomorphism testing for graphs of bounded genus. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 225–235, 1980.

[53] M. Otto. *Bounded variable logics and counting – A study in finite models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.

[54] M. Otto. Canonization for two variables and puzzles on the square. *Annals of Pure and Applied Logic*, 85:243–282, 1997.

[55] M. Otto. Bisimulation-invariant PTIME and higher-dimensional μ-calculus. *Theoretical Computer Science*, 224:237–265, 1999.

[56] S.-I. Oum and P.D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96:514–528, 2006.

[57] N.D. Roussopoulos. A $\max\{m,n\}$ algorithm for determining the graph H from its line graph G. *Information Processing Letters*, 2:108–112, 1973.

[58] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[59] O. Verbitsky. Planar graphs: Logical complexity and parallel isomorphism tests. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science*, volume 4393 of *Lecture Notes in Computer Science*, pages 682–693. Springer-Verlag, 2007.

[60] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–168, 1932.