DEFINABILITY AND DECIDABILITY IN FIRST ORDER THEORIES OF GRAPH ORDER

By

RAMANATHAN THINNIYAM SRINIVASAN

MATH10201204005

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

February, 2019

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Ramanathan Thinniyam Srinivasan entitled "Definability and Decidability in First Order Theories of Graph Order" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

Kamal Lodaya
Chairman - Prof. Kamal Lodaya — Date: 12/2/19

[illegible]
Guide/Convenor - Prof. R. Ramanujam — Date: 12/2/2019

Supratik Chakraborty
Examiner - Prof. Supratik Chakraborty — Date: 12/2/2019

[illegible]
Member 1 - Prof. Meena Mahajan — Date: 12/2/2019

Suresh
Member 2 - Prof. S. P. Suresh — Date: 12.02.2019

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 12 Feb 2019

Place: Chennai

[illegible]
Prof. R. Ramanujam(Guide)

STATEMENT BY AUTHOR

Ramanathan Thinniyam Srinivasan

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Ramanathan Thinniyam Srinivasan

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. “Defining recursive predicates in graph order”, R. S. Thinniyam, in Logical Methods in Computer Science (LMCS), **2018** September, Volume 14, Issue 3.

Conferences

1. “Definability in first order theories of graph orderings”, R. Ramanujam, R. S. Thinniyam, in International Symposium on Logical Foundations of Computer Science (LFCS), **2016** January, (pp. 331-348). Springer, Cham.
2. “Definability of recursive predicates in the induced subgraph order”, R. S. Thinniyam, in Indian Conference on Logic and Its Application, **2017** January, (pp. 211-223). Springer, Berlin, Heidelberg.
3. “Decidability in the substructure ordering of finite graphs”, A. Padmanabha, R. S. Thinniyam, presented at Logic Colloquium, 23 to 28 July **2018**, Udine, Italy.

Ramanathan Thinniyam Srinivasan

DEDICATIONS

To the one who came before me

and to the one who will come after.

ACKNOWLEDGEMENTS

I would like to acknowledge the support I have received and continue to receive from my parents, without which it would have been impossible for me to find my bearings. I am grateful to my guide Prof R. Ramanujam who was more generous with his time than I could have imagined and gave me the space to explore freely. Thanks is also due to the faculty of the TCS department who have taught me things both academic and non-academic during the past six years. I am indebted to my friends who ensured that I did not become a vegetable when the going got tough.

Contents

Synopsis

Introduction

Finite graphs are used to model a variety of problems (see book on applications of graph theory by Foulds [15]) and are of central importance in computer science. There are different variants of graphs such as directed graphs, undirected graphs, graphs with edge weights etc. which are used to model different situations. In this thesis, we restrict our study to finite, undirected, simple graphs, that is, ones where the edge relation E is a symmetric and irreflexive binary relation. In particular, we study the first order theory of structures of the kind $(\mathcal{G}, \tau)$ where $\mathcal{G}$ is the set of isomorphism types of finite graphs; and the vocabulary τ contains a binary relation $\leq$ which is interpreted as a partial order. There are several natural candidates for the partial order $\leq$ which arise from graph theory, and we study the induced subgraph, subgraph and minor orders (denoted $\leq_i$, $\leq_s$ and $\leq_m$ respectively) with an emphasis on the induced subgraph and subgraph relations. We will call such theories first order theories of *graph order*.

The study of the first order theory of $(\mathbb{N}, +, \times)$, called first order arithmetic (or simply arithmetic) has a long and storied history (see survey by Bés [3]). Another example of the study of the first order theory of a structure is Tarski's celebrated decision procedure for the first order theory of real arithmetic (see survey by Dries [41]).

These examples follow a particular pattern: collect the objects of interest into one infinite domain D, specify a finite set of operations τ of interest (addition and multiplication in the case of numbers) and study the first order theory of the structure (D, τ). The study of first order theories of graph order is exactly this kind of study where the objects of interest are graphs. There is much less literature dealing with graph orders as compared to arithmetic.

At this juncture, we would like to make a note of the difference between a major field of logical study of finite graphs, namely *Finite Model Theory* (FMT) and the study of graph orders. In FMT (and in its associated field of *Descriptive Complexity*), the approach taken is that a single finite graph $g = (V, E)$ is the model and the vocabulary is the edge relation $\{E\}$ (sometimes extended by other operators such as a total order on the finite domain). We do not have explicit access to the edge relation in theories of graph order, though as we shall see, the 'internal structure' of graphs can be accessed in an indirect fashion using graph order.

We now describe two bodies of literature which are closely related to this thesis. The first body of work is the study of substructure orderings on various kinds of finite objects such as semi-lattices [20], posets [23], distributive lattices [21] and lattices [22] initiated by Jězek and McKenzie. This was extended by Wires [43] to the case of graphs, where the notion of substructure is the induced subgraph order. One of the main objectives in Wires' study is to understand the lattice $\mathscr{S}_g$ of classes of graph axiomatized by universal theories. It turns out that the induced subgraph order is closely related to this structure and the definability of constants in the induced subgraph order implies definability of the finitely generated order-ideals of the induced subgraph order in the structure $\mathscr{S}_g$.

Theorem (Wires [43])**.** *Every element is definable as a constant in the first order theory of ($\mathcal{G}, \leq_i, P_3$), where P_3 is a constant interpreted as the path on three vertices. The only non-trivial automorphism of ($\mathcal{G}, \leq_i$) is the automorphism mapping every graph to its complement.*

On the way to proving the above theorem, Wires shows that many important graph theoretical predicates such as connectivity, disjoint union of graphs, independence number etc. are definable in this first order theory which we find to be of independent interest. The undecidability of the theory follows from the following fact:

Theorem (Wires [43])**.** *First order arithmetic is interpretable in the first order theory of* $(\mathcal{G}, \leq_i, P_3)$*.*

However, the computational content of the first order theory of induced subgraph is not studied by Wires. Understanding the computational content of graph orders is one of the primary motives of this thesis. In our attempt to do so, we crucially use the notion of *o-presentations* introduced by Jĕzek and McKenzie and defined for graphs by Wires. An o-presentation of a graph g is a representation of an ordered version of g as another graph $\tilde{g}$. This can be used as a tool to relate edge information of g which is not available to us in the vocabulary with induced subgraph information about g'.

The second body of work related to this thesis is the study of various kinds of orderings on terms (which can be seen as ranked trees) and words (which can be seen as symmetric labelled paths) which is a part of the term rewriting literature [9, 7]. Different kinds of orders such as lexicographic path order, infix order and subword order have been studied in this literature, of which the subword order is closest in spirit to the graph orders considered. An early result which considers the subword order is by Kuske:

Theorem (Kuske [30])**.** *The* Σ_3 *fragment of the subword order is undecidable, and the* Σ_1 *fragment is decidable.*

Recently, the subword order has regained interest. The question about the decidability of the Σ_2 fragment which was left open by Kuske was shown to be undecidable by Karandikar and Schnoebelen [24]. This result was further strengthened by Halfon et al. to show the following surprising result.

Theorem (Halfon et al. [18])**.** *The* Σ_1 *theory of the subword order expanded with constants is undecidable.*

There was also work done on finite variable fragments.

Theorem (Karandikar and Schnoebelen [24]). *The FO^3 fragment of the subword order expanded by constants is undecidable. The FO^2 fragment of the subword order with constants is decidable.*

The techniques used to obtain the above results are based on automata theory. Unfortunately, automata theory for tree and graphs is not as well understood as that for words.

Contributions of the Thesis

We first summarize the main results on definability in graph order.

We show that arithmetic can be interpreted in each of the graph orders considered in this thesis.

Theorem. *The first order theories of each of the structures $(\mathcal{G}, \leq_i, P_3)$,$(\mathcal{G}, \leq_s)$ and $(\mathcal{G}, \leq_m)$ is mutually interpretable with first order arithmetic.*

We then define the notion of a capable structure $(\mathcal{G}, \tau)$ over graphs (see Definition 4.6) and characterize the relations definable in such a structure.

Theorem. *Let $(\mathcal{G}, \tau)$ be a capable structure over graphs. A relation is definable in a capable structure if and only if it is arithmetical.*

The capability of the induced subgraph order can be seen as a corollary of results obtained by Wires [9], but we provide explicit formulae for the same. The main technical contribution of this thesis is the proof of capability of the subgraph order, during the course of which we show the definability of natural graph theoretic predicates (such as counting the number of edges of a graph and the disjoint union of graphs) in the subgraph order.

As a corollary we also obtain the capability of the minor order when expanded by the predicate $sameSize(x, y)$ which holds if and only if x and y have the same number of edges.

Theorem. *Each of the structures* $(\mathcal{G}, \leq_i, P_3)$,$(\mathcal{G}, \leq_s)$ *and* $(\mathcal{G}, \leq_m, sameSize)$ *is capable.*

The above theorem gives us a characterization of the relations definable in graph order and concludes the main results on definability in graph order. We prove some results regarding the decidability of syntactic fragments of the induced subgraph order. These results are summarized in the following table.

Table 1: Summary of decidability results for the induced subgraph order. A † indicates the result also holds for the subgraph and minor orders.

	Existential	FO^1	FO^2	FO^3
Pure	**NP**-complete†	-NA-	?	?
With constants	Undecidable	in $\mathbf{NP}^{\mathbf{NP}} \cap \mathbf{coNP}^{\mathbf{NP}}$ †	?	Undecidable
Positive with constants	**NP**-complete†	in $\mathbf{NP}^{\mathbf{NP}} \cap \mathbf{coNP}^{\mathbf{NP}}$ †	Decidable	?

In the next section, we describe the organization of this thesis and discuss future directions.

Organization of the Thesis

This thesis is comprised of the following chapters:

1. Introduction: We give an informal introduction to graph orders, summarize relevant literature and describe the organization of the thesis.
2. Preliminaries: This chapter contains a brief summary of first order logic, definitions related to graph orders and notions of *recursively enumerable relations over graphs* and *arithmetical relations over graphs*. We introduce notation which will be used throughout the thesis and the particular encodings of graphs as strings which allow

us to talk about recursively enumerable predicates over graphs using the Turing Machine model. We also define a particular total order on the set of all finite graphs $\mathcal{G}$ with respect to which the structures over graphs are arithmetical.

3. Mutual Interpretability of Arithmetic and Graph Orders: In this chapter we define many graph families and graph relations in each of the three graph orders studied in this thesis: the subgraph, the induced subgraph and the minor orders. These basic definability results allow us to interpret arithmetic in a uniform way in graph order. Though the converse is not unexpected, we show the definability of graph orders in arithmetic.
4. Defining Arithmetical Predicates in Arbitrary Structures over Graphs: We give a proof of the well known result that any recursively enumerable relation over numbers is definable in first order arithmetic to ensure that the abstract definability result in the second section of this chapter is self-contained. We then introduce the concept of a *capable* structure over graphs, which is one which can interpret arithmetic and define three particular graph theoretic predicates related to o-presentations. We then show that any capable structure can define any relation which is arithmetical. An important corollary is the definability of every recursively enumerable predicate in capable structures. We then show that the induced subgraph order is a capable structure. The capability of the induced subgraph order follows as a corollary from the results in Wires [43] and our alternate development extensively uses the basic predicates defined by Wires in the induced subgraph order.
5. Defining Arithmetical Predicates in the Subgraph Order: The capability of the subgraph order uses the definability of two natural graph theoretic predicates of independent interest not previously defined in literature: a ternary predicate $disjointUnion(z, x, y)$ if and only if z is the disjoint union of x and y, and a binary predicate $sameSize(x, y)$ if and only if x and y have the same number of edges. The proof is broken up into two parts. In the first, we show the capability of $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ and in the second we show the definability of

$disjointUnion$ and $sameSize$ in the subgraph order. As a corollary, we also obtain the capability of the expansion of the minor order with the predicate $sameSize$. This chapter may be considered the main technical contribution of this thesis.

6. Decidability in Graph Order: This chapter contains decidability results concerning syntactic fragments of the induced subgraph order. The fragments considered are of two kinds: expansions of the existential theory and finite variable fragments. The existential theory of $(\mathcal{G}, \leq_i)$ is shown to be $\mathsf{NP}-$complete and the existential theory on expansion with constants, denoted $(\mathcal{G}, \leq_i, \mathscr{C}_g)$, is shown to be undecidable. In the case of finite variable restrictions, the FO^3 fragment of $(\mathcal{G}, \leq_i, \mathscr{C}_g)$ is shown to be undecidable and the FO^1 fragment decidable. We also show that the $FO^{2(+)}$ fragment of $(\mathcal{G}, \leq_i, \mathscr{C}_g)$, which is the restriction of FO^2 to only positive formulae, is also decidable. The decidability of the FO^2 fragment is left open. The techniques used by Karandikar and Schnoebelen [25] to show the decidability of the FO^2 fragment of the subword order implicitly contain a strategy for showing the decidability of the FO^2 fragment of the theory of any poset satisfying certain conditions. We separate the logical and combinatorial concerns and define the notion of a locally recursive poset for which this strategy works.
7. Future Work and Conclusion: We go through the results obtained in the thesis and discuss possible extensions and strengthenings. Of particular interest is the definability of recursively enumerable predicates using only existential formulae, which we call the MRDP conjecture for graphs (the analog of MRDP theorem for numbers [34]). On the decidability front, we have concentrated on syntactic fragments of the induced subgraph order. The main open problem arising from syntactic restrictions that may be taken up next is the decidability of the two variable fragment of the induced subgraph order. We discuss work to be done on the other graph orders as well as domain restrictions and structures over graphs with different vocabularies.

List of Figures

Chapter 1

Introduction and Literature Survey

In this thesis we study the first order theories of *structures over graphs* with an emphasis on *graph orders*. Structures over graphs are structures of the kind $(\mathcal{G}, \tau)$ where $\mathcal{G}$ is the set of all finite, undirected graphs, and τ some set of relations and constants. Graph orders are structures over graphs where τ contains a partial order $\leq$. The partial orders we consider are the subgraph, induced subgraph and minor relations (see Figure 1.1 for an initial segment of the induced subgraph order).

The questions about graph orders that we address are of two kinds. The first is that of *definability*, which asks what kind of relations among graphs are definable using first order formulae over graph order. The second is that of *decidability*, which asks for programs which can decide the truth of first order sentences or fragments thereof.

The study of decidability and definability in first order theories over natural numbers has a long history (see Bés [3] for a survey). Similar studies have been carried out for real arithmetic, and include celebrated results such as Tarski's quantifier elimination procedure for real closed fields (see Arnon [1]) for a survey). However, the study of combinatorial theories i.e. theories where the objects of interest are words, trees, graphs etc. has assumed more importance more recently, since the advent of computer science.

One approach to the logical study of graphs is that taken by Finite Model Theory (FMT for short, see Libkin [32]) and the related field of Descriptive Complexity (see Immerman [19]). What follows is a simplistic account of FMT as is revelant to the study of finite graphs. A fixed finite graph $G = (V, E)$ is taken as the model with finite domain (vertex set) V and the vocabulary consisting of the edge relation E (sometimes the vocabulary may involve other predicates such an order $\leq$ on the vertices of the graph). Thus the quantification in FMT is over the finite set V with variables denoting vertices.

In this thesis however, the domain is the set $\mathcal{G}$ and the quantification is over all graphs with a variable denoting a graph. In particular, we do not have access to the edge relation or 'internal structure' of a graph (see Figure 1.2). There is also a large difference in the expressive power of first order logic (FOL) considered in these two diferent contexts. While FOL in the FMT setting defines subsets of $\mathcal{G}$, FOL over graph order can be used to define relations of any arity over $\mathcal{G}$.

FMT studies the set of all finite structures while the study of graph order is the study of a single infinite structure $(\mathcal{G}, \leq)$. There are two kinds of structures studied in literature which are similar in nature to graph orders. The first are the substructure orderings of finite structures which are not graphs, for example, the set of all finite posets. The second kind of structures are orderings on words and terms. We give an overview of these two bodies of work in the next two sections, and in the last section we discuss the organization of this thesis.

1.1 Studies of Substructure Orderings over Finite Structures

In a series of papers, Ježek and McKenzie studied the substructure orderings over finite objects such as semi-lattices [20], posets [23], distributive lattices [21] and lattices [22].

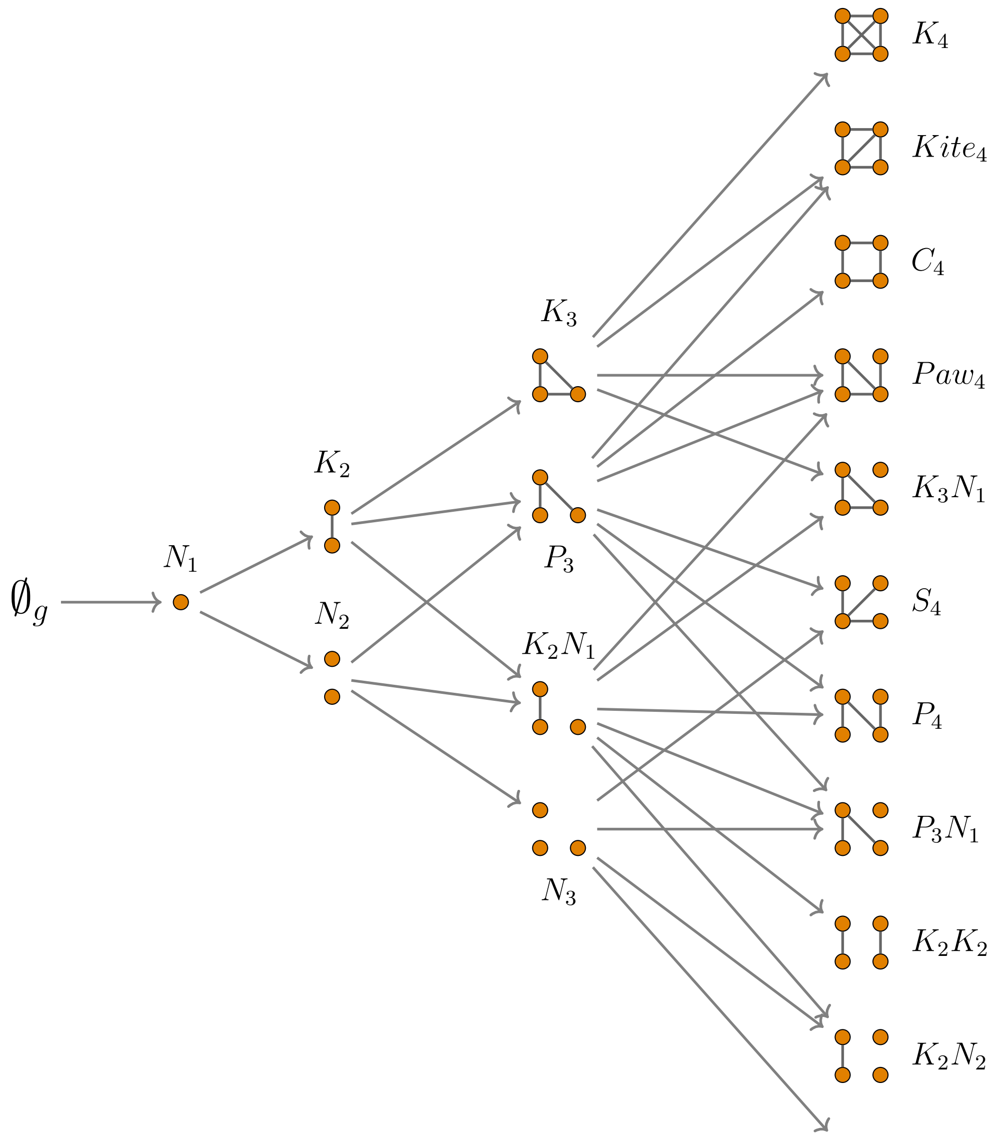


Figure 1.1: The first few levels of the induced subgraph order $\leq_i$. Note the symmetry arising from the automorphism $f(g) = g^c$.

$$\begin{tikzcd}
 & & & K_3 \\
 & & K_2 \arrow[ur] \arrow[r] \arrow[ddr] & P_3 \\
\emptyset_g \arrow[r] & N_1 \arrow[ur] \arrow[dr] & & \\
 & & N_2 \arrow[uur] \arrow[r] \arrow[dr] & K_2N_1 \\
 & & & N_3
\end{tikzcd}$$

Figure 1.2: Initial layers of the object $(\mathcal{G}, \leq_i)$. Note that the edge relation i.e. the "internal structure" is not available to us directly, nor are the names indicated inside the nodes. The arrows indicate the upper cover relation.

Wires [43] extended this work to the substructure order on graphs, also known as the induced subgraph order. A related work by Kunos [29] studies directed graphs under the subdigraph order.

One of the objectives in these studies is to understand universal classes of objects, not necessarily finite. For instance, let $\mathcal{K}$ be a class of semi-lattices described by some set of universal sentences. Ordering all such $\mathcal{K}$ by inclusion gives the lattice $\mathcal{U}$. Let $\mathcal{S}$ be the set of all finite semi-lattices. Then the map sending $\mathcal{K}$ to $\mathcal{K} \cap \mathcal{S}$ is an isomorphism between $\mathcal{U}$ and the set of all order-ideals of $\mathcal{S}$. Ježek and McKenzie observe that the definability in $\mathcal{U}$ of the set of $\mathcal{K}$ which contain only finitely many elements (in other words, the set of finitely generated order ideals of $\mathcal{S}$) can be reduced to showing the definability in $\mathcal{S}$ of every domain element as a constant. They proceed to show that this latter fact holds.

The equivalent question for the case of graphs is whether every graph is definable in the first order theory of the structure $(\mathcal{G}, \leq_i)$. However, this is not possible due to the fact that the map sending every graph g to its complement g^c is an automorphism of the induced subgraph order. This 'natural automorphism' turns out to be the only one, and it can be broken by addition of a single constant. Wires shows that every constant can be defined in the first order theory of $(\mathcal{G}, \leq_i, P_3)$ where P_3 is a constant for the path on three vertices. In the course of doing so, he establishes the definability of many natural graph theoretical predicates such as cardinality, disjoint union etc. as well as graph families such as paths, cycles, stars, trees etc. in this structure. An important corollary obtained is the undecidability of the first order theory of the induced subgraph order via interpretation of arithmetic.

Wires also shows that the set of predicates definable in $(\mathcal{G}, \leq_i, P_3)$ is exactly the set of isomorphism invariant predicates definable in the first order theory of a simple expansion $\mathcal{CG}'$ of the small category of graphs $\mathcal{CG}$. The category $\mathcal{CG}$ can be thought of as a two-sorted structure with domain $\mathcal{G}^* \cup \mathscr{F}$. The elements of $\mathcal{G}^*$ are ordered graphs which are those whose vertex set is an initial segment of $\mathbb{N}$. The elements in $\mathscr{F}$ are the arrows of the category and are homomorphisms between elements of $\mathcal{G}^*$. A member of $\mathscr{F}$ is a triple (g, f, g') where f is a morphism from g to g', where $g, g' \in \mathcal{G}^*$. The composition relation between arrows forms the vocabulary. The structure $\mathcal{CG}'$ should be thought of as a second order structure since we are allowed to quantify over $\mathscr{F}$. The fact that first order definability in the induced subgraph order is equivalent to second order definability is a strong result which implies our results regarding the induced subgraph order in Section 4.3.

The definability results obtained in this thesis differ from the above in their connection to computation. We show that every relation over graphs which is arithmetical is definable in graph order. Since the predicates in the vocabulary are recursive (and hence arithmetical), this gives us a characterization: the predicates definable in graph order are exactly those which are arithmetical. Our techniques rely crucially on the construction of objects known as *o-presentations* introduced first by Ježek and Mckenzie [23]. In the case of graphs, Wires defines o-presentations which are formed by attaching large cycles to the vertices of the original graph g to obtain a new graph g' (see Figure 4.1 for an example). While originally used to show that every graph can be defined as a constant, o-presentations are used in this thesis to define the edge relation in an indirect way using FOL over graph order.

Another point of distinction compared to Wires [43] is that other graph orders of significance in graph theory, such as the subgraph and minor orders, were not studied by him. The work in this thesis highlights some of the similarities between these orders: for instance, we give a uniform way to encode arithmetic in all three graph orders. However, there are also interesting differences in the difficulty of defining different graph theoretical predicates in these orders. For instance, the cardinality of a graph is easily defined in the subgraph order as the largest member of the family $\mathcal{N}$ (of graphs consisting of only isolated points) which is a subgraph; but it takes much more work in the induced subgraph order. In the end, the results of this thesis indicate that the different graph orders considered are equally powerful in terms of definability when we consider full FOL.

The results in this thesis also indicate that theories of graph order may be used to formalize graph theoretic statements just as arithmetic (both first and second order variants) may be used to formalize statements in number theory. Many theorems in an introductory text on graph theory (such as Diestel [10]) may be written in theories of graph order. There do exist important statements such as the Graph Minor Theorem [37] which quantify over infinite subsets of $\mathcal{G}$ and are not expressible in theories of graph order.

In the next two sections we discuss the literature related to the work in this thesis.

1.2 Term Rewriting Literature and the Subword Order

This thesis can also be thought of as extending the study of structures of order over words and trees to graphs. While structures with the domain set Σ^* of finite words over an alphabet Σ have been studied for some time (see Quine [35] on $(\Sigma^*, .)$ where . is a binary function symbol for concatenation), the study of order theories of combinatorial objects is more recent and arose as part of the term rewriting literature [42, 9, 7]. Treinen and Comon

[8] studied the lexicographic path ordering on words. Other orders such as subword, infix etc. were studied by Kuske [30]. The decidability of restricted fragments of FOL over different orders has been of interest (see Comon and Treinen [7] for a survey).

Out of the various orderings on words, the subword order $(\Sigma^*, \leq_{sw})$ (where $\Sigma = \{a, b\}$) is the closest in nature to the orders studied in this thesis. In particular, we show that there is a quantifier-free interpretation of the subword order in the induced subgraph order if we have access to a constant for each domain element. Thus the subword order is a subposet of the induced subgraph order. This allows us to transfer many of the undecidability results for the subword order to the induced subgraph order. We now give an overview of what has been accomplished in literature with regards to decidability in the subword order.

The study of decidability of the subword order has concentrated on syntactic fragments. Work by Kuske [30] showed that the Σ_3 fragment is undecidable while the Σ_1 fragment is NP-complete. This left open the question of the decidability of the Σ_2 fragment, which was shown to be undecidable by Karandikar and Schnoebelen [24]. In the same paper, the FO^2 fragment of the subword order was shown to be decidable, even on addition of all regular languages as unary predicates to the vocabulary. In a very surprising result, Halfon et al [18] showed the undecidability of the Σ_1 fragment of the subword order when expanded by constants for all words.

The upper bound on complexity given by Karandikar and Schnoebelen [24] was non-elementary due to the techniques used, which involved complementation of automata. In an attempt to improve the bound, the same authors introduced new techniques combined with a deeper understanding of the combinatorial properties of *piecewise testable* (p.t.) languages to give a $3EXPTIME$ decision procedure in [25]. The p.t. sets are a subset of regular languages of words which have been extensively studied and form the first level of what is called the *dot-depth hierarchy* (introduced by Brzozowski and Cohen[6]). The p.t. sets were studied by Imre Simon in his PhD thesis [38]. We quote Denis Thérien on Imre Simon's work: "His graduate work had major impact on algebraic theory of automata..."

[39].

The new techniques used by Karandikar and Schnoebelen involve the understanding of various kinds of closure operations on p.t. languages. Such an understanding of closure operations on infinite posets leads to an abstract decidability result of the FO^2 fragment of infinite posets which satisfy said closure properties. We exploit these techniques to give some results in the case of graph order.

We would also like to note that Kudinov et al. [28] show that the subword order can define any relation which is arithmetical. The paper by Kudinov et al is unrelated to the term-rewriting literature above, and is part of the program on computability in abstract structures stemming from Barwise's treatise on admissible sets [2] and continued by researchers from the Novosibirsk school [14].

1.3 Results and Thesis Organization

We recall that the graph orders studied in this thesis are the induced subgraph, subgraph and minor orders. In the case of the induced subgraph order, the vocabulary is expanded by adding a constant P_3 for the path on three vertices. This is to break the automorphism $f : \mathcal{G} \to \mathcal{G}$ defined as $f(g) = g^c$ i.e. every graph is mapped to the graph obtained by exchanging edges and non-edges (see Figure 1.1). We note that without the constant P_3, there is no way to distinguish between a graph and its complement using first order logic. In the case of the subgraph and minor orders, the constant P_3 is not required as there do not exist any non-trivial automorphisms. In the rest of this section, when we talk of the induced subgraph order, we assume that the constant P_3 is also available to us unless explicitly left out.

Chapter 1: Introduction and Literature Survey

This chapter expands on the introduction in this synopsis, explaining in more detail the place of this thesis with respect to the literature.

Chapter 2: Preliminaries

We recall basic definitions from first order logic and graph theory. We also introduce encodings of graphs as strings and vice versa. These are used to give formal definitions for the notion of 'recursively enumerable predicate over graphs' used in this thesis. We also define the notion of an 'arithmetical predicate over graphs'.

Chapter 3: Mutual Interpretability of Arithmetic and Graph Order

This chapter contains many basic constructions which culminate in the definability of arithmetic in the subgraph and minor orders. In the case of the induced subgraph order, many of the graph theoretic predicates required to carry out the constructions in this thesis are already present in Wires' work [43]. We also show the definability of graph order in arithmetic.

Many interesting predicates are definable using simple formulae with just an order relation.

Theorem 1.1. *The following predicates are definable in the subgraph and minor orders:*

1. *Covering relation: $x \lessdot y$ if and only if $x < y$ and there is no graph strictly between them.*
2. *Constants N_1, K_2, K_3, S_4, P_4 for the graph with one vertex, clique on two vertices, clique on three vertices, star on four vertices, path on four vertices.*
3. *The family $\mathcal{N}$ of graphs with isolated points.*

4. *The cardinality of a graph: $|x| = |y|$ if and only if x and y have the same number of vertices.*

The ability to define constants combined with atomic negation is powerful: for instance, the formula $S_4 \not\leq_s x$ defines the family pac of graphs which are disjoint unions of paths and cycles by avoiding degree 3 vertices. For a fixed cardinality n of the vertex set, collecting the maximal graphs in the family pac under the subgraph order gives us the family soc of disjoint unions of cycles. Bootstrapping on these basic constructions allows us to define many families of graph theoretic interest.

Theorem 1.2. *The families $\mathcal{N}, \mathcal{K}, forest, \mathcal{T}, \mathcal{C}, \mathcal{S}, \mathcal{P}, conn$ which denote isolated points, cliques, forests, trees, cycles, stars, paths and connected graphs respectively (see Figure 2.1), are definable in the subgraph and minor orders.*

The maximum degree and maximum path length of a graph are definable in the subgraph and minor orders.

The family $\mathcal{N}$ is used to represent numbers, and numerical parameters of a graph such as maximum degree are interpreted as belonging to this family. For instance, the predicate $maxDeg(x, n)$ which holds if and only if $n \in \mathcal{N}$ and the maximum degree of a vertex in x is $|n|$, is definable in the subgraph and minor orders. The definability of the predicate $maxDeg(x, n)$ is simply referred to as definability of maximum degree in the above theorem.

At this point, we can construct gadgets to define arithmetic operations over the set $\mathcal{N}$ used to represent numbers. For instance, given any n, there is a maximum tree t_n under the subgraph order which has maximum path subgraph P_5 and maximum degree n; and the cardinality of t_n is $n^2 + 1$. This gadget t_n is used to define the arithmetical predicate $square(m, n)$ if and only if $n, m \in \mathcal{N}$ and $|n| = |m|^2$. Together with a gadget for defining addition, we are able to define arithmetic.

Theorem 1.3. *First order arithmetic is definable in the subgraph and minor orders.*

We note that the result above does not immediately follow from the corresponding result for the induced subgraph order obtained by Wires [43].

Theorem 1.4. *The induced subgraph, subgraph and minor orders are mutually interpretable with arithmetic.*

Chapter 4: Defining Arithmetical Predicates in Arbitrary Structures over Graphs

In this chapter, we establish some sufficient conditions for any arithmetical structure over graphs $(\mathcal{G}, \tau)$ to be able to define every arithmetical predicate over graphs. We call such structures *capable* structures over graphs. Since every relation definable in any arithmetical structure is also arithmetical, this gives a characterization of the relations definable in a capable structure: a relation is definable in a capable structure if and only if it is arithmetical. We recall the notion of an arithmetical structure and arithmetical relation ober graphs.

Definition 1.5. *A relational structure $(\mathcal{G}, \tau)$ over graphs is called arithmetical if there exists a total order $\leq_t$ on $\mathcal{G}$ such that ($\mathcal{G}, \leq_t$) is isomorphic to ($\mathbb{N}, \leq$) and all relations $R \in \tau$ are arithmetical. In other words, if $plus_t$ and $times_t$ are the ternary addition and multiplication relations with respect to the order $\leq_t$, then every $R \in \tau$ is definable in ($\mathcal{G}, plus_t, times_t$).*

An arithmetical relation over graphs is one definable in ($\mathcal{G}, plus_t, times_t$).

In order to state what a capable structure is, we need the notion of an o-presentation.

Definition 1.6 (o-presentation)**.** *An o-presentation of $g \in \mathcal{G}$ is another graph g' constructed as follows: Fix a vertex ordering $v_1 < v_2 < \cdots < v_n$ of vertices of g. Let g'' be the graph formed by the disjoint union of g and the cycles C_{n+i+2} for each $1 \leq i \leq n$. Add n additional edges to g'' connecting each cycle C_{n+i+2} to the corresponding vertex v_i. The resulting graph is g'.*

The cycles of size $n + i + 2$ for $1 \leq i \leq n$ are called indicator cycles of a graph g on n vertices.

The graph g' as constructed above is not unique (see Figure 4.1); in fact there is a bijective correspondence between vertex orderings of g and the set of o-presentations of g. We write $g' \in \tilde{g}$ to denote that g' is an o-presentation of g.

Definition 1.7 (Capable Structure over Graphs)**.** *We call an arithmetical structure ($\mathcal{G}, \tau$) a capable structure over graphs if it satisfies the following three conditions:*

($C1$) Arithmetic can be defined in ($\mathcal{G}, \tau$), in particular, the following predicates are definable:

1. *The family $\mathcal{N}$ of graphs which do not contain edges i.e. are made of isolated points.*
2. *The predicate $plus(z, x, y)$ which holds if and only if $x, y, z \in \mathcal{N}$ and $|x| + |y| = |z|$.*
3. *The predicate $times(z, x, y)$ which holds if and only if $x, y, z \in \mathcal{N}$ and $|x| \times |y| = |z|$.*

($C2$) The following predicates related to o-presentations are definable in ($\mathcal{G}, \tau$):

1. *$\psi_{opres}(x, y)$ which holds if and only if x is an o-presentation of y, also written $x \in \tilde{y}$.*
2. *$\psi_{edgeOP}(x, i, j)$ which holds if and only if there exists a graph y such that $x \in \tilde{y}$ and in the vertex ordering induced on y by the o-presentation x, the vertices v_i and v_j in y have an edge.*

($C3$) The predicate $sameCard(x, y)$ which holds if and only if x and y have the same number of vertices, is definable in ($\mathcal{G}, \tau$).

Theorem 1.8. *For any ($\mathcal{G}, \tau$) which is a capable structure over graphs, every arithmetical predicate $R \subseteq \mathcal{G}^n$ over graphs is definable in ($\mathcal{G}, \tau$).*

The proof of the above theorem has two ingredients: (1) arithmetical definability and (2) the ability to define a binary predicate $\psi_{enc}(x, y)$ which represents an injective map

$enc : \mathcal{G} \to \mathcal{N}$ such that $\psi_{enc}(x, y)$ is true iff $enc(x) = y$. Together, these two ingredients allow us to go from a graph g to its corresponding image $enc(g)$ which resides in $\mathcal{N}$ and perform the required computation in the isomorphic copy of arithmetic ($\mathcal{N}, plus, times$) residing inside the graph order.

The rest of the chapter involves showing that the induced subgraph order is a capable structure over graphs. The fact that the induced subgraph order is a capable structure is a corollary of results obtained by Wires. We give an alternate presentation with explicit formulae establishing that the induced subgraph order is capable.

Theorem 1.9. *Every arithmetical predicate is definable in the induced subgraph order.*

Chapter 5: Defining Arithmetical Predicates in the Subgraph Order

This chapter contains the constructions required to prove condition (C2) for the subgraph order. There are subtle differences between the induced subgraph and subgraph orders. For instance, in the induced subgraph order an o-presentation $\tilde{g}$ is constructed from g by first constructing the graph $g \uplus \biguplus_{i=1}^{|g|} P_{|g|+i+1}$ which is the disjoint union of g with a set of paths and then adding the rest of the vertices and edges. In contrast, we construct the graph $g \uplus \biguplus_{i=1}^{|g|} C_{|g|+i+2}$ which is the disjoint union of g with a set of cycles and add the remaining edges in the case of the subgraph order.

We reduce the problem to that of defining three other intermediate predicates. To state the technical lemma, we need some definitions.

Definition 1.10. *The graph $C_{i\to 1}$ is the connected graph formed by adding one new vertex and one new edge to the cycle of cardinality i denoted C_i.*

Given a graph g with $|g| = n$, an indicator cycle is a cycle C_{n+i+2} where $1 \leq i \leq n$. Note that these are the cycles attached to vertices of g to form an o-presentation.

Lemma 1.11. *The predicates ψ_{opres} and ψ_{edgeOP} are definable in the subgraph order assuming the definability of the following predicates:*

1. *$CP_4C(x,i,j)$ holds if and only if $i,j \in \mathcal{N}, 3 < i < j$ and x is constructed from the graph $C_{i\to 1} \uplus C_{j\to 1}$ by adding one additional edge between the unique degree one vertices of $C_{i\to 1}$ and $C_{j\to 1}$.*
2. *$\tilde{\mathcal{G}}(x)$ holds if and only if x is an o-presentation of some graph.*
3. *$constructFromCycles(x,y)$ holds if and only if y is constructed by adding $|x|$ edges to the graph g which is the disjoint union of x and all indicator cycles corresponding to x.*

Of the three intermediate predicates, CP_4C is the easiest, followed by $\tilde{\mathcal{G}}$ and $constructFromCycles(x,y)$ the most difficult. We break the proof up into two parts. First we show that the structure $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ is capable, where $disjointUnion(z,x,y)$ is a ternary predicate which holds if and only if z is the disjoint union of the graphs x and y and $sameSize(x,y)$ is a binary predicate which holds if and only if x and y have the same number of edges. Second, we show that $disjointUnion$ and $sameSize$ are definable in the subgraph order.

Lemma 1.12. *The predicate $constructFromCycles$ is definable in the subgraph order assuming the definability of the following predicates:*

1. *$csum(x,n)$ holds if and only if $x = \biguplus_{i=1}^{n} C_{n+i+2}$.*
2. *$disjointUnion(z,x,y)$ holds if and only if z is the disjoint union of the graphs x and y.*
3. *$countEdges(x,n)$ holds if and only if $n \in \mathcal{N}$ and x has $|n|$ many edges.*

The remainder of the chapter is devoted to the construction of gadgets to enable the definition of disjoint union and edge counting. We sum up the results of this chapter and the previous one with the formal statement of definability of arithmetical predicates in graph order:

Theorem 1.13 (Arithmetical Predicates in Graph Orders)**.** *For a structure $\mathcal{A}$, let $Def(\mathcal{A})$*

be the set of all predicates definable in $\mathcal{A}$. Then

$$Def(\mathcal{G}, plus_t, times_t) = Def(\mathcal{G}, \leq_s) = Def(\mathcal{G}, \leq_i, P_3) = Def(\mathcal{G}, \leq_m, sameSize).$$

Chapter 6: Decidability in Graph Order

In this chapter, we take up the issue of decidability. The ability to encode arithmetic implies that the full first order theory of graph order is undecidable. We concentrate on examining syntactic fragments, namely the existential fragment and the finite variable fragments; with an emphasis on the induced subgraph order. Atomic negation and constants contribute significant power towards definability of predicates. Hence we explore the consequences of extending the vocabulary by constants and removal of negation in the syntactic fragments. In Table 1.1 we summarize the decidability results obtained. We discuss some of them below.

Table 1.1: Summary of decidability results for the induced subgraph order. A † indicates the result also holds for the subgraph and minor orders.

	Existential	FO^1	FO^2	FO^3
Pure	**NP**-complete†	-NA-	?	?
With constants	Undecidable	in $\mathbf{NP}^{\mathbf{NP}} \cap \mathbf{coNP}^{\mathbf{NP}}$ †	?	Undecidable
Positive with constants	**NP**-complete†	in $\mathbf{NP}^{\mathbf{NP}} \cap \mathbf{coNP}^{\mathbf{NP}}$ †	Decidable	?

We first describe the results on the existential fragment. The following result is a consequence of the fact that each of the graph orders is universal for finite posets.

Theorem 1.14. *The existential theories of each of the induced subgraph, subgraph and minor orders is* NP*-complete.*

But on addition of constants to the vocabulary, we get an undecidable theory.

Theorem 1.15. *The existential theory of the induced subgraph order with a constant symbol for each graph (denoted $\exists^*(\mathcal{G}, \leq_i, \mathscr{C}_g)$) is undecidable.*

The result is obtained via a quantifier free interpretation of the subword order in the subgraph order; the analogous result for subword order having already been established by Halfon et al. [18]. However, it is not clear whether it is possible to interpret the subword order in the subgraph and minor orders. On further restriction to only the positive fragment, we get back decidability.

Theorem 1.16. *Let $\leq$ denote any one of $\leq_i, \leq_s, \leq_m$. The theory $\exists^{*(+)}(\mathcal{G}, \leq, \mathscr{C}_g)$, which has formulae given by the grammar*

$$\phi := c\Box x \mid x\Box y \mid \exists y\ \phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

where $\Box \in \{<, >, =\}$ is NP*-complete.*

Next we describe the results for finite variable fragments. The Σ_0 interpretation of the subword in the induced subgraph order also gives the following result.

Theorem 1.17. *The $FO^3 \cap \Sigma_2$ fragment of the induced subgraph order with constants is undecidable.*

Since the single variable fragment with constants can be shown to be in $\mathsf{NP}^{\mathsf{NP}} \cap \mathsf{coNP}^{\mathsf{NP}}$, this leaves open the question of decidability of the FO^2 fragment. We note that many constants are already definable (upto automorphism of the structure) in the FO^2 fragment, and thus with regards to decidability, there is unlikely to be any difference on adding or removing constants from the vocabulary. For this reason we concentrate on the expansion with constants.

Lemma 1.18. *Every graph on at most 4 vertices is definable upto automorphism in the FO^2 fragment of the induced subgraph order.*

The decidability of FO^2 with constants in the case of the subword order does not carry over in any obvious way to the case of graphs. By analyzing the proof by Karandikar and Schnoebelen [25], we are able to separate the logical and combinatorial concerns. In particular, it is clear that the techniques used by them are relevant for a broad class of infinite posets which we call *locally recursive*.

In the case of such locally recursive posets, the quantifier free formulae in one free variable define sets which are finite unions of *universes* which are solution sets $[\![\phi(x)]\!]$ of quantifier free formulae of the form $\phi(x) = c \leq x \wedge \bigwedge_{d\in D} d \not\leq x$; where D is a finite set. We call a finite union of universes a *multiverse*.

We have the following result:

Theorem 1.19. *Let ($\mathscr{P}, \leq_P$) be a locally recursive poset. Given a set $S \subseteq \mathscr{P}$ we define the unary operations $S \Downarrow = \{p \mid \exists p' \in S \ p <_P p'\}$ and $S|| = \{p \mid \exists p' \in S \ p \not\leq_P p' \ \wedge \ p' \not\leq_P p\}$ which map subsets of $\mathscr{P}$ to subsets of $\mathscr{P}$. If the multiverses of $\mathscr{P}$ are closed under $S \Downarrow$ and $S||$, and furthermore, a representation for the resulting multiverse can be effectively computed from a representation of the input multiverse, then the FO^2 theory of ($\mathscr{P}, \leq_P, \mathscr{C}_P$) is decidable.*

The conditions of closure in the above theorem in the case where $\mathscr{P}$ is a graph order are related to generalizations of the Ramsey theorem. We do not attack this problem in this thesis.

Chapter 7: Future Work and Conclusion

We recognize that there are two immediate technical questions for the short term. The first is the possibility of strengthening Theorem 1.13; in particular, we raise the question of the existence of a vocabulary τ so that the existential theory of $(\mathcal{G}, \tau)$ is capable of defining every recursively enumerable predicate. This can be thought of as the analog in graphs to the MRDP (Matiyasevich-Robinson-Davis-Putnam) theorem [34], which states that the definable sets in the existential theory of arithmetic are exactly the recursively enumerable sets of numbers. Secondly we discuss the combinatorial results required to prove the decidability of the FO^2 fragment left open.

A longer term project is the understanding of how various complexity classes fit into theories of graph order. This involves developing the graph analog to the Bounded Arithmetic theories (see Buss [5]) shown to characterize complexity classes. Unfortunately the vocabulary of graph order is too coarse since the atomic predicates are already NP-complete and hence we need to identify the right vocabulary.

Chapter 2

Preliminaries

2.1 First Order Logic

For the sake of completeness, we give here a basic account of the syntax and semantics of first order logic (FOL). Subleties such as variable capture and renaming of variables are not dealt with here and the reader may consult a standard text such as Enderton [11] for the same. Readers conversant with FOL may skip ahead to Section 2.2.

FOL uses symbols from a *vocabulary* $\tau = (\mathscr{F}, \mathscr{R}, \mathscr{C})$ comprised of countable sets of function symbols $\mathscr{F}$, relation (aka predicate) symbols $\mathscr{R}$ and constant symbols $\mathscr{C}$, a countable set $\mathscr{V}$ of variables and *logical* symbols $\vee, \neg, \exists, =$. These symbols are used to construct $\tau-$*terms* and $\tau-$*formulae*. Where τ is clear from the context, we just say terms and formulae.

A term is specified by the grammar:

$$t := c \mid x \mid f(t_1, t_2, ..., t_{r_f}).$$

That is, any constant c or variable x is a term, and if $t_1, t_2, ..., t_{r_f}$ are terms then $f(t_1, t_2, ..., t_{r_f})$ is a term (where r_f is the *arity* of f).

A formula is specified by the grammar :

$$\phi := R(t_1, t_2, .., t_{r_R}) \mid (t_1 = t_2) \mid (\phi_1 \vee \phi_2) \mid (\neg\phi) \mid (\exists x\ \phi(x))$$

where R is a relation symbol of arity r_R.

The logical symbols $\wedge, \supset, \forall, \iff$ are defined as:

$\phi_1 \wedge \phi_2 := \neg(\phi_1 \vee \phi_2)$

$\phi_1 \supset \phi_2 := \neg\phi_1 \vee \phi_2$

$\phi_1 \iff \phi_2 := (\phi_1 \supset \phi_2)\ \wedge\ (\phi_2 \supset \phi_1)$

$\forall x\ \phi(x) := \neg\exists x\ \neg\phi(x)$

$\exists! x\ \phi(x) := \exists x\ \phi(x)\ \wedge\ \forall y\ \phi(y) \supset x = y$

We will also sometimes use bounded quantifers of the form $\forall x\ 1 \leq x \leq y\ \phi(x, y)$ which is to be read $\forall x\ (1 \leq x \leq y) \supset \phi(x, y)$. Similarly, $\exists x\ (1 \leq x \leq y)\ \phi(x, y)$ is to be read $\exists x\ (1 \leq x \leq y)\ \wedge\ \phi(x, y)$.

Formulae of the form $R(t_1, t_2, .., t_{r_R})$ and $(t_1 = t_2)$ are called *atomic* formulae. Compound formulae are generated from atomic formulae by use of logical symbols.

A *quantifier-free* formula is one which does not contain the symbols $\exists$ and $\forall$. A *free variable* is one which is not quantified. A formula with no free variables is called a *sentence*.

A first order *structure* $\mathcal{A}$ with vocabulary τ is a tuple (D, I) where D is a *domain* set and I an *interpretation* which assigns the following:

1. For each function symbol $f \in \mathscr{F}$ of arity r_f an r_f-ary function $f^I : D^{r_f} \rightarrow D$.
2. For each $R \in \mathscr{R}$ of arity r_R a subset R^I of D^{r_R}.
3. For each constant $c \in \mathscr{C}$ a domain element $c^I \in D$.

An assignment $\sigma : \mathscr{V} \rightarrow D$ is a function mapping every variable to an element of the domain D of a $\tau-$structure $\mathcal{A} = (D, I)$. The assignment function is extended to terms in the obvious way and for a term t we will write $\sigma(t)$ to denote the domain element assigned to t by σ. Consider a $\tau-$structure $\mathcal{A}$ together with an assignment σ. For any $\tau-$formula $\phi(\bar{x})$ where $\bar{x} = (x_1, x_2, ..., x_n)$, let $\phi(\bar{x})[\bar{x} \leftarrow \bar{c}]$ be the $\tau'-$formula produced by replacing each free variable x_i by a new constant symbol $c_i \notin \tau$. The tuple $(\mathcal{A}, \sigma)$ is said to *model* a formula $\phi(\bar{x})$ where $\bar{x} = (x_1, x_2, ..., x_n)$, written $\mathcal{A}, \sigma \models \phi(\bar{x})$ using the following inductive definition:

- For an atomic formula $R(t_1, t_2, ..., t_{r_R})$, consider the *expansion* of $\mathcal{A}' = (D, I')$ with vocabulary $\tau \cup \{c_1, c_2, .., c_n\}$ of $\mathcal{A} = (D, I)$. Each c_i is a new constant symbol not present in τ and the interpretation $c_i^{I'}$ of each c_i is given by $\sigma(x_i)$. I' agrees with I on the interpretation of all symbols in τ. The sentence $R(t_1, t_2, ..., t_{r_R})[\bar{x} \leftarrow \bar{c}]$ is a τ' sentence obtained by replacement of every variable x_i by the corresponding constant symbol c_i.

$$\mathcal{A}, \sigma \models R(t_1, t_2, ..., t_{r_R}) \text{ if and only if } R^{I'}(t_1, t_2, ..., t_{r_R})[\bar{x} \leftarrow \bar{c}] \text{ holds}$$

  Similarly for atomic formulae of the form $(t_1 = t_2)$,

$$\mathcal{A}, \sigma \models (t_1 = t_2) \text{ if and only if } \sigma(t_1) = \sigma(t_2)$$

- For compound formulae of the form $\phi(\bar{x}) = \phi_1(\bar{x}) \vee \phi_2(\bar{x})$ and $\phi(\bar{x}) = \neg\phi'(\bar{x})$ we have, respectively:

$$\mathcal{A}, \sigma \models \phi(\bar{x}) \text{ if and only if } (\mathcal{A}, \sigma \models \phi_1(\bar{x}) \text{ or } \mathcal{A}, \sigma \models \phi_2(\bar{x})), \text{and}$$

$$\mathcal{A}, \sigma \models \phi(\bar{x}) \text{ if and only if } \mathcal{A}, \sigma \not\models \phi'(\bar{x}).$$

  The symbol $\not\models$ denotes ‘does not model’.

- For a compound formula of the form $\phi(\bar{x}) = \exists y \ (\phi'(\bar{x}, y))$

$$\mathcal{A}, \sigma \models \phi(\bar{x}) \text{ if and only if there is an expansion } \mathcal{A}'' \text{ of } \mathcal{A} \text{ with vocabulary}$$
$$\tau \cup \{c\} \text{ and it is the case that } \mathcal{A}'', \sigma \models \phi'(\bar{x}, y)[y \leftarrow c].$$

Note that in the above definition, the constant symbol c is not contained in τ.

In case ϕ is a sentence, no assignment function is required and simply write $\mathcal{A} \models \phi$. The phrases 'entails', 'holds of' or 'is true of' are also used in place of 'models' to denote that a property holds of a structure.

Example 2.1. *1. Every natural number is the sum of four squares:*

$(\mathbb{N}, +, \times) \models \forall x \ \exists y_1 \ \exists y_2 \ \exists y_3 \ \exists y_4 \ x = y_1^2 + y_2^2 + y_3^2 + y_4^2$

2. There are no $0-$*divisors in arithmetic:*

$(\mathbb{N}, \times, 0) \not\models \exists x \ \exists y \ x \neq 0 \ \wedge \ y \neq 0 \ \wedge x \times y = 0$

3. Three is less than four:

$(\mathbb{N}, +), (3, 4) \models \exists y \ x_1 + y = x_2$

In the last example, we need an assignment function because $\exists y \ x_1 + y = x_2$ is a formula. Often, free variables of a formula are listed as a tuple $x_1, x_2, ...$ and on the left hand side, we specify the assignments in the appropriate order. In this case, 3 is assigned to x_1 and 4 is assigned to x_2.

Definition 2.2 (Theories)**.** *A theory $\mathscr{T}$ is a set of $\tau-$sentences for some fixed vocabulary τ.*

The first order theory of a $\tau-$structure $\mathcal{A}$ is the set of all sentences that are true of $\mathcal{A}$. We denote this set by $Th_{FO}(\mathcal{A})$.

$$Th_{FO}(\mathcal{A}) := \{\phi \mid \mathcal{A} \models \phi\}$$

We simply say theory of a structure since the theories dealt with in this thesis are all *contained in the first order theory.*

Fix a set $\bar{x} = \{x_1, x_2, ..., x_k\}$ of variables. Let FO^k denote the subset of first order formulae whose variables are a subset of $\bar{x}$. The $k-$variable fragment of the first order theory of $\mathcal{A}$ is $FO^k(\mathcal{A}) = Th_{FO}(\mathcal{A}) \cap FO^k$.

Existential formulae are those which are of the form $\exists x_1\ \exists x_2\ ...\exists x_n\ \alpha(x_1, ..., x_n)$ where α is a quantifier free formula. The existential theory of $\mathcal{A}$ is the restriction of $Th_{FO}(\mathcal{A})$ to existential sentences, and is denoted $\exists^*(\mathcal{A})$.

The Σ_2 theory of a structure $\mathcal{A}$ is the restriction of $Th_{FO}(\mathcal{A})$ to sentences of the form $\exists x_1\ \exists x_2\ ...\ \exists x_n\ \forall y_1\ \forall y_2\ ...\ \forall y_m\ \phi(\bar{x}, \bar{y})$ where $\phi(\bar{x}, \bar{y})$ is a quantifier-free formula and $n, m \in \mathbb{N}$.

When referring to definability in the first order theory of a structure $\mathcal{A}$, we will often just say 'definable in the theory of $\mathcal{A}$' or 'definable in $\mathcal{A}$'.

Example 2.1(3) leads us naturally to the related notion of *solution sets of formulae*, which is the set of tuples of domain elements making a formula true. Fixing $(\mathbb{N}, +)$ as our structure of study, the tuple $(3, 4)$ is an assignment which makes the formula true while $(4, 3)$ is a tuple which makes the formula false. In this example, the solution set is the set of all tuples (m, n) such that $m \leq n$. We say the formula $\exists y\ x_1 + y = x_2$ *defines* the order relation between numbers.

Definition 2.3. *The solution set of a $\tau-$formula $\phi(\bar{x})$ with k free variables in a $\tau-$structure $\mathcal{A}$, denoted $[\![\phi(\bar{x})]\!]_{\mathcal{A}}$, is the set of $k-$tuples of domain elements of $\mathcal{A}$ which make the formula true in $\mathcal{A}$.*

$$[\![\phi(\bar{x})]\!]_{\mathcal{A}} := \{\bar{a} \mid \mathcal{A}, \bar{a} \models \phi(\bar{x})\}$$

In most cases, the structure $\mathcal{A}$ of interest is clear from the context and we drop the subscript $\mathcal{A}$ and just write $[\![\phi]\!]$.

Definition 2.4 (Definability of Relations). *An n-ary relation R is definable in a τ-structure*

$\mathcal{A}$ if there is a formula $\phi(\bar{x})$ with n free variables such that for any n-tuple $\bar{a} \in \mathcal{A}$,

$$R(\bar{a}) \iff \mathcal{A}, \bar{a} \models \phi(\bar{x})$$

When we talk of the definability of a function f in this thesis, we mean the definability of the associated relation R_f with arity one greater than the arity of f:

$$f(\bar{x}) = y \iff R_f(\bar{x}, y)$$

Remark 2.5. *We will abuse notation by talking of definability of a relation R in a theory $\mathscr{T}$. The intended meaning of such a statement is that R is definable by a formula $\phi(\bar{x})$ such that $\bar{Q}\bar{x}\ \phi(\bar{x})$ belongs to $\mathscr{T}$, where $\bar{Q}$ is any sequence of quantifiers.*

We also talk of the definability of an element of the domain.

Definition 2.6 (Definability of Constants)**.** *Let d be an element of the domain of a $\tau-$structure $\mathcal{A}$. We say that d is definable if there exists a $\tau-$formula $\phi_d(x)$ in one free variable such that $\mathcal{A}, d \vDash \phi_d(x)$ and for any $d' \neq d$ in the domain of $\mathcal{A}$, $\mathcal{A}, d' \nvDash \phi_d(x)$.*

We use d as a constant symbol for the domain element d with the understanding that an equivalent formula can be written without the use of this constant symbol.

Definition 2.7. *The first order theory of the structure $(\mathbb{N}, +, \times)$ is called first order arithmetic. A relation definable in this structure is called an arithmetical relation.*

We will often just say ‘definable in arithmetic’ to mean definable in $(\mathbb{N}, +, \times)$.

Remark 2.8. *In this thesis, we will not make a distinction between definability in $(\mathbb{N}, +, \times)$ and definability in $(\mathbb{N}, plus, times)$ which is the structure obtained by replacing $+, \times$ by ternary relations $plus, times$ respectively where $plus(z, x, y)$ holds if and only if $x + y = z$ and similarly for $times$.*

Definition 2.9 (Interpretability of Structures). *A relational $\tau-$structure $\mathcal{A}$ is said to be interpretable in a relational $\tau'-$structure $\mathcal{B}$ if there exist the following:*

1. *an injective map f from the domain A of $\mathcal{A}$ to the domain B of $\mathcal{B}$,*
2. *a formula $\phi_A(x)$ in the vocabulary τ' such that for any element $b \in B$ it is the case that $\phi_A(b)$ holds if and only if b belongs to the image set $f(A)$, and*
3. *for each $k-$ary relation $R \in \tau$, there is a formula $\phi_R(\bar{x})$ in the vocabulary τ' such that for any $k-$tuple $\bar{a}$ of elements of A, it is the case that $R(\bar{a})$ holds if and only if $\phi_R(f(\bar{a}))$ holds.*

Two structures $\mathcal{A}$ and $\mathcal{B}$ are said to be mutually interpretable if each is interpretable in the other.

The definition above can extended to vocabularies containing functions by considering the definability of the $(k+1)-$ary relation $R_f(\bar{x}, y)$ corresponding to a $k-$ary function $f(\bar{x}) = y$.

2.2 Structures over Graphs and Graph Orders

To clarify what we mean by the isomorphism type of a graph, we formally define the notion of an *ordered* graph.

Definition 2.10. *An ordered (finite) graph is a structure (V, E) where $V = [n] = \{1, 2, 3, ..., n\}$ for some $n \in \mathbb{N}$, and E is a symmetric, irreflexive binary relation. For any two ordered graphs $g = (V, E), g' = (V', E')$, we say they are isomorphic if $V(g) = V(g') = [n]$ for some n and there exists a function $\eta : [n] \to [n]$ such that for any $i, j \in [n]$ $E(i, j) \iff E'(\eta(i), \eta(j))$. We write $g \simeq_g g'$ to denote that two ordered graphs are isomorphic.*

We will write $V = \{v_1, v_2, ..., v_n\}$ instead of using the set $[n]$ to avoid confusion between numbers used to represent vertices and numbers in themselves.

Definition 2.11. *A graph g is an equivalence class (aka isomorphism type) of ordered graphs under the equivalence relation $\simeq_g$. We write $g = [g']$ to denote that g' is an ordered graph belonging to the equivalence class denoted by g.*

In some proofs, it is easier to write arguments about the structure of a graph g by fixing an ordered graph g' with $g \in [g']$. Where the ordering on vertices is not required, we will use u, v to denote vertices. Similarly, e is used to represent a particular edge and uv denotes the edge between vertices u and v.

Definition 2.12 (Structure over Graphs)**.** *A structure over graphs is one which has as its domain the set $\mathcal{G}$ and a set $\tau = [\mathscr{F}, \mathscr{R}, \mathscr{C}]$ of functions $\mathscr{F}$, relations $\mathscr{R}$ and constants $\mathscr{C}$ in $\mathcal{G}$. We will denote the structure by $(\mathcal{G}, \tau)$.*

The structures over graphs of interest in this thesis always include a binary relation $\leq$ as part of the vocabulary which is interpreted as a partial order. We call such structures graph orders.

The graph orders studied in this thesis are the induced subgraph, subgraph and minor orders.

Definition 2.13 (Graph Partial Orders)**.** *Consider the following operations on graphs:*

- *A1. Deletion of a vertex (and all the edges incident on that vertex).*
- *A2. Deletion of an edge.*
- *A3. Contraction of an edge (given an edge $e = uv$, delete both u and v and introduce a new vertex w not in $V(g)$; connect all vertices which were adjacent to either u or v to w).*

For graphs g and g', g can be obtained from g' by a finite sequence of the operations

1. *A1,A2 and A3 if and only if $g \leq_m g'$(g is a minor of g');*
2. *A1 and A2 if and only if $g \leq_s g'$(g is a subgraph of g'); and*
3. *A1 if and only if $g \leq_i g'$(g is an induced subgraph of g').*

When both A1 and A2 are part of the set $\mathscr{O}$ of allowed operations, we can replace A1 by the operation A1' which is the deletion of an isolated vertex to obtain an equivalent set of operations $\mathscr{O}'$. In some cases, it will be more convenient to consider the latter.

The following structures play a prominent role in this thesis:

1. $(\mathcal{G}, \leq_i, P_3)$: the induced subgraph order with a constant symbol P_3 for the path on three vertices.
2. $(\mathcal{G}, \leq_s)$: the subgraph order.
3. $(\mathcal{G}, \leq_m, sameSize)$: the minor order with an additional binary relation $sameSize(x, y)$ which holds if and only if x and y have the same number of edges.
4. We also consider graph orders expanded by the set $\mathscr{C}_g$ which contains a constant symbol for each element in $\mathcal{G}$. Corresponding to each of the three graphs orders we have $(\mathcal{G}, \leq_i, \mathscr{C}_g)$, $(\mathcal{G}, \leq_s, \mathscr{C}_g)$ and $(\mathcal{G}, \leq_m, \mathscr{C}_g)$.

The constant P_3 is used to break the symmetry of the induced subgraph order which by itself cannot distinguish between a graph and its complement since the map sending a graph to its complement is an automorphism of the induced subgraph order. The subgraph order does not have any symmetry; this fact follows from the fact that every constant is definable in its first order theory. It seems likely that the minor order also has no symmetry, this has not been proved in this thesis.

Remark 2.14. *We describe a convention we follow throughout this thesis regarding the order of variables used to define predicates over graphs. Let $R(x_1, x_2, ..., x_k)$ be a $k-$ary relation defined using a formula $\psi_R(x_1, x_2, ..., x_k)$ over graph order. The order of the variables is $x_1 < x_2 < ... < x_k$ in this instance. The convention adopted states that if $R(g_1, g_2, ..., g_k)$ implies that $g_i \leq g_j$ for some $i, j \in \mathbb{N}$, then it must be the case that $i < j$. For a more concrete example, consider the predicate $comp(x, y)$ which holds if and only if y is a component of x. If $comp(g_1, g_2)$ holds, then so does $g_1 \leq_i g_2$. We could have chosen to define the predicate as: $comp(x, y)$ holds if and only x is a component of y, but the former definition is used in accordance with the convention. We hope this will help the*

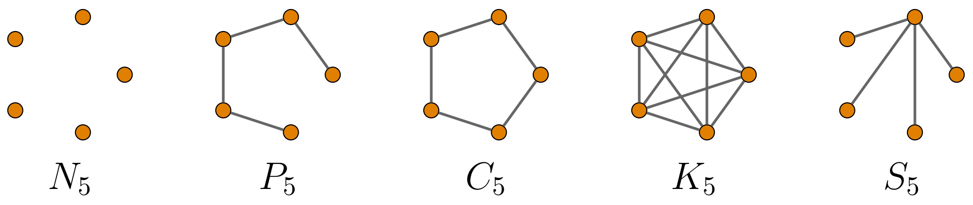


Figure 2.1: Isolated points, path, cycle, clique and star of order 5 from left to right.

reader in parsing the formulae.

Remark 2.15. *We make a note of the different symbols used in this thesis in the list below. Subscripts and superscripts used with a symbol represent the same kind of objects as do the symbol (with the exception that w_i represents the i^{th} letter of a word w).*

- *x, y, z are variables representing graphs in graph formulae and numbers in arithmetic formulae.*
- *g, h are used to represent particular graphs.*
- *$\mathcal{F}, \mathcal{G}$ represent families of graphs.*
- *N_i, K_i, C_i, S_i, P_i represent the graph consisting of i isolated vertices, the i-clique, the cycle on i vertices, the star on i vertices and the path on i vertices respectively (see Figure 2.1).*
- *$\mathcal{N}, \mathcal{K}, \mathcal{C}, \mathcal{S}, \mathcal{P}$ the corresponding families of isolated vertices, cliques, cycles, stars and paths.*
- *i, j, k, l, m, n are used for natural numbers (also on occasion, members of the $\mathcal{N}$ family).*
- *$\downarrow, \Downarrow, \uparrow, \Uparrow$ represent downclosure, strict downclosure, upclosure and strict upclosure in a poset respectively.*

Given a graph g, we use the following notation to represent parameters or contructions which use g as indicated:

- *$V(g)$ stands for the vertex set of g.*
- *$E(g)$ stands for the edge set of g.*

- $|g|$ *stands for the number of vertices of* g *(also called the order of* g*).*
- $||g||$ *stands for the number of edges of* g *(also called the size of* g*).*
- $|g|_g$ *stands for the graph consisting of only isolated vertices which has the same number of vertices as* g*.*
- $g \uplus h$ *stands for the disjoint union of the graphs* g *and* h*.*
- *Given a subset* $V_0 \subseteq V(g)$*,* $g[V_0]$ *denotes the graph induced by the vertices* V_0*.*

Remark 2.16. *We will often deal with interpretations of arithmetic in structures* $(\mathcal{G}, \tau)$ *over graphs in this thesis. When such an interpretation is possible, there exists a formula* ψ^{trans} *over the vocabulary* τ *which is the translation of a formula* ϕ *over the vocabulary* $(+, \times)$*. The relation defined by* ψ^{trans} *will be called a number-theoretic relation in order to avoid confusion with the notion of an arithmetical relation.*

2.3 Computability and Arithmetic

In this section, we introduce two notions which give us a way to measure the power of FOL over graph orders in its ability to define relations over the set $\mathcal{G}$. The first is the notion of a computable relation which comes from the Turing Machine model of computation and the second is that of arithmetic predicates over graphs.

2.3.1 Turing Machines and Representations

Turing Machines are widely accepted as the standard model of computation. A Turing Machine M can be in one of a finite set of *states* Q, has an infinite tape comprised of cells indexed by $\mathbb{N}$ with each cell containing an *alphabet* from a finite set Σ of alphabets and has a *head* which is positioned over a particular tape cell. A configuration of the machine is its state together with the tape contents and the position of the head. The *initial* configuration of the machine is one where the state is a designated *start* state, the head is positioned at the first cell and the tape contents contain the *input* string. The machine transitions from one configuration to another according to a *transition function*. There is also a designated final state f such that there are no transitions possible from f. The machine halts if no transition is possible. The machine accepts an input x if and only if it halts after reaching the final state. The *language* $L(M)$ accepted by M is the set of all strings accepted by M.

The formal definition of a Turing Machine is given below.

Definition 2.17. *A Turing Machine M is a 5-tuple $(Q, \Sigma, \delta, s, f)$ where*

- *Q is a finite, nonempty set of states,*
- *Σ is a finite, nonempty set of alphabets,*
- *$\delta : Q \setminus \{f\} \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is a transition function which is intended to represent a move of M. A transition $(q, a) \to (q', a', d)$ allows the machine to move from a state q on reading alphabet a to a state q' after writing a' in place of a and moving its head in the direction d; where $d = L$ denotes movement of the head one cell to the left and $d = R$ one cell to the right.*
- *$s \in Q$ is the designated start state.*
- *$f \in Q$ is the designated final state.*

A halting Turing Machine M is one which runs for finitely many steps on any input and halts in one of two designated states: a state f which is an accept state and a state r which is a reject state. The language $L(M)$ of a halting Turing Machine M is the set of strings on which M halts in state f.

Definition 2.18. *A recursively enumerable (r.e.) set $R \subseteq \Sigma^*$ of strings is one such that there exists a Turing Machine M with $L(M) = R$.*

A recursive set $R \subseteq \Sigma^*$ of strings is one such that there exists a halting Turing Machine M with $L(M) = R$.

The notion of an r.e. set (respectively recursive set) can be extended to that of an r.e. relation (respectively recursive relation) by giving as input to the Turing Machine a tuple of strings encoded as a single string using a special separating alphabet.

A formula (or sentence) in FOL can be thought of as a finite string which can be given to a Turing Machine as input in order to decide some property of the input formula.

Definition 2.19. *The decision problem associated with a theory $\mathscr{T}$ is whether a given input sentence ϕ belongs to $\mathscr{T}$ or not. If there exists a halting Turing Machine for this decision problem, we say the theory $\mathscr{T}$ is decidable, else it is undecidable.*

The decision problem for a theory $\mathscr{T}$ will on occasion also be referred to as the *truth problem* for the theory $\mathscr{T}$.

The input to a Turing Machine is always a string. Hence, in order to use this notion to talk of a recursively enumerable set of graphs, it is necessary to encode a graph as a string. Representing graphs as numbers also gives a representation as strings via the usual binary representation of numbers as strings, which we proceed to do below.

Definition 2.20 (Number Representation of a Graph)**.** *If g is either $\emptyset_g$ or N_1, it is represented by the numbers 0 and 1 respectively. A number representation of a graph g which is not $\emptyset_g$ or N_1 is defined using the following procedure.*

1. *Choose an ordered graph g' such that $g = [g']$. The order on vertices given by $L_{g'}$ induces an order $\leq_e$ on set S of all tuples of vertices (v_j, v_i) of g with $i < j$. Let (v_j, v_i) and (v_l, v_k) belong to S (i.e. $i < j, k < l$). Define $(v_j, v_i) \leq_e (v_l, v_k)$ if and only if $j < l$ or $j = l, i < k$. Thus the smallest tuple is (v_2, v_1) while the largest is $(v_{|g|}, v_{|g|-1})$.*
2. *Arrange all the tuples belonging to S in descending order by $\leq_e$ to form the sequence seq_g.*
3. *Create the number m whose binary expansion is $\binom{n}{2} + 1$ bits long and has the following property: the i^{th} most significant bit is 0 or 1 according to whether the i^{th} smallest tuple in seq_g corresponds to a non-edge or edge (respectively) of the ordered graph g'.*

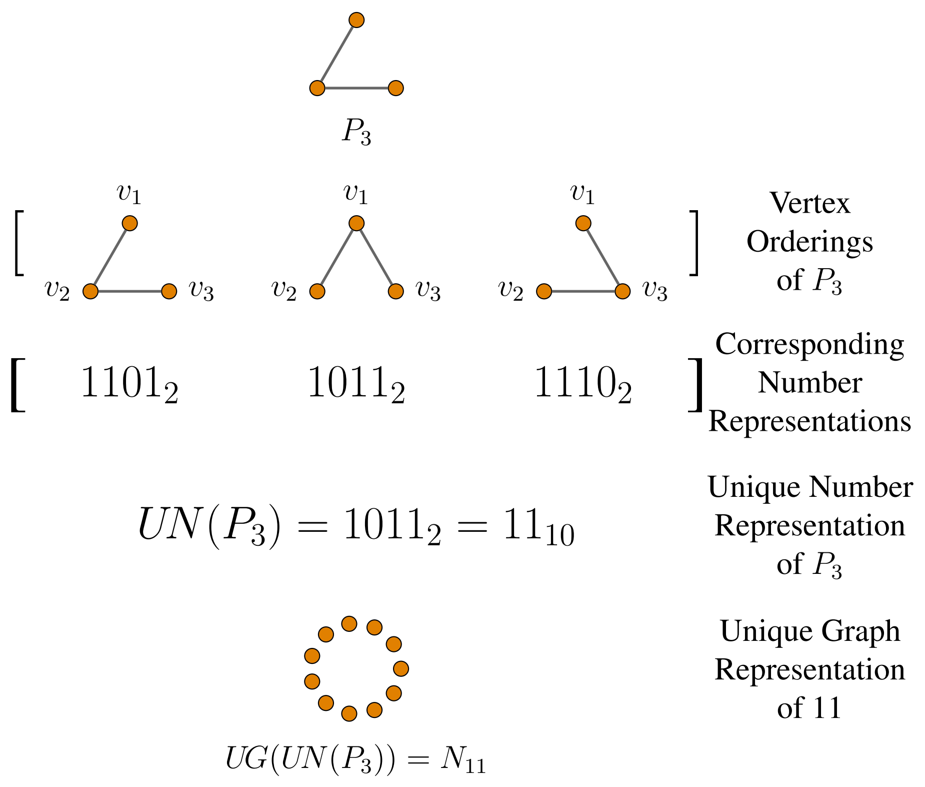

P_3
v_1 v_2 v_3
Vertex Orderings of P_3
1101_2 1011_2 1110_2
Corresponding Number Representations
$UN(P_3) = 1011_2 = 11_{10}$
Unique Number Representation of P_3
Unique Graph Representation of 11
$UG(UN(P_3)) = N_{11}$

Figure 2.2: From top to bottom we see how to obtain the graph $UG(UN(g)) \in \mathcal{N}$ from any graph $g \in \mathcal{G}$ using $g = P_3$ as an example. The subscripts 2 and 10 in the numbers correspond to the base used.

The number m is called a number representation of the graph g. We will denote the set of all number representations by NR. We can also think of $NR : \mathbb{N} \to \mathcal{G}$ as a partial function from numbers to graphs.

Given a number n, we will use the notation $g_{(n)}$ to denote the graph $NR(n)$.

Definition 2.21 (Unique Number Representation of a Graph aka UN)**.** *The unique number representation of a graph g is the least number m such that it is a number representation of g and is denoted $UN(g)$. Note that the map $UN : \mathcal{G} \to \mathbb{N}$ is a one-one map (see Figure 2.2 for an example). Tuples of graphs $\bar{g}$ are represented by tuples of numbers in the usual way. The map UN can also be thought of as a map from graphs to strings by considering the binary representation of numbers.*

Observation 2.22. *The unique representation $UN(g)$ of a graph g induces a particular* *ordering on the vertices of the graph g since every representation $NR(n)$ which maps n to g can be associated with a particular ordering of $V(g)$.*

The formal definition of a recursively enumerable predicates over graphs.

Definition 2.23. *We say a predicate $R \subseteq \mathcal{G}^n$ is r.e. if there exists a Turing Machine M such that*

$$R(\bar{g}) \iff UN(\bar{g}) \in L(M)$$

i.e. the Turing Machine M accepts exactly the tuples of strings which correspond to UN representations of tuples belonging to R.

We will also need to encode numbers as graphs.

Definition 2.24 (Unique Graph Representation of a Number aka *UG*)**.** *Let $\mathcal{N}$ be the family of graphs which consists of graphs with no edges. This family contains exactly one graph of cardinality k for any fixed $k \in \mathbb{N}$, denoted by N_k.*

The one-one map $UG : \mathbb{N} \to \mathcal{G}$ sends a number k to the graph N_k which is called the unique graph representation of k.

2.3.2 Arithmetical Predicates over Graphs

The second natural measure of definability we use is related to arithmetical definability. The sets $\mathcal{G}$ and $\mathbb{N}$ are both countably infinite and in the case where τ_0 is the empty vocabulary, $(\mathcal{G}, \tau_0)$ and $(\mathbb{N}, \tau_0)$ are identical structures. The distinction between structures over graphs and number-theoretic structures comes from our notion of 'natural' operations or relations on graphs and numbers respectively.

Consider a total order $\leq_t$ on $\mathcal{G}$ such that $f_{iso} : (\mathcal{G}, \leq_t) \to (\mathbb{N}, \leq)$ is an isomorphism. A graph order such as $\leq_s$ on $\mathcal{G}$ can be thought of as an 'unnatural' partial order on $\mathbb{N}$ via the isomorphism f_{iso}. Conversely, the natural operations $+_t, \times_t$ which are addition and multiplication with respect to $\leq_t$ are 'unnatural' operations on graphs, giving us the structure $(\mathcal{G}, +_t, \times_t)$ which is isomorphic to $(\mathbb{N}, +, \times)$ via the isomorphism f_{iso}. This enables us to transfer results and concepts from numbers to graphs and vice versa once a total order $\leq_t$ is fixed.

We define the notion of an *arithmetical* structure over graphs.

Definition 2.25. *A relational structure $(\mathcal{G}, \tau)$ over graphs is called arithmetical if there exists a total order $\leq_t$ on $\mathcal{G}$ such that ($\mathcal{G}, \leq_t$) is isomorphic to ($\mathbb{N}, \leq$) and all relations $R \in \tau$ are arithmetical. In other words, if $plus_t$ and $times_t$ are the ternary addition and multiplication relations with respect to the order $\leq_t$, then every $R \in \tau$ is definable in ($\mathcal{G}, plus_t, times_t$).*

Observation 2.26. *Every predicate definable in an arithmetical structure is also arithmetical.*

Remark 2.27. *The more general definition of an arithmetical structure $\mathcal{A}$ which is not necessarily a structure over graphs used by Kudinov and Selivanov [27] is different from that given above in the following ways:*

1. *Instead of an ordering, Kudinov and Selivanov consider a numbering function $num : \mathbb{N} \to \mathcal{A}$ which is required to be onto.*
2. *The relations in τ as well as the equality relation are arithmetical modulo the numbering.*

Definition 2.28. *An arithmetical structure ($\mathcal{G}, \tau$) over graphs is said to have the maximal definability property if every arithmetical predicate (w.r.t. $\leq_t$) is definable in the first order theory of ($\mathcal{G}, \tau$).*

It is clear that the maximal definability property for an arithmetical structure $(\mathcal{G}, \tau)$ gives a characterization of the definable predicates of the structure i.e. a predicate is definable in $(\mathcal{G}, \tau)$ iff it is arithmetical (w.r.t. $\leq_t$). The set of arithmetical predicates w.r.t. any other total order $\leq_{t'}$ on $\mathcal{G}$ remains the same as long as $\leq_{t'}$ is arithmetical w.r.t. $\leq_t$.

The situation is analogous to the considerations of string encodings of graphs encountered in the previous section.

Next we define the total order $\leq_t$ with respect to which the structures we consider in this thesis are arithmetical. In the rest of this thesis, we shorten the phrase 'arithmetical with respect to $\leq_t$' to just 'arithmetical' with the understanding that the underlying order $\leq_t$ is the following:

Definition 2.29. *Define $g_1 \leq_t g_2$ for $g_1, g_2 \in \mathcal{G}$ if and only if $UN(g_1) \leq UN(g_2)$. Note that $\leq_t$ is a total order on $\mathcal{G}$ by the uniqueness of the map UN.*

This concludes the definitions we require to parse the results in this thesis. The next four chapters form the technical content of this thesis. We take up the interpretability of arithmetic in graph order in the next chapter.

Chapter 3

Mutual Interpretability of Arithmetic and Graph Order[1]

In this chapter, we show the mutual interpretability of arithmetic [2] with the induced subgraph, subgraph and minor orders. The chapter is divided into three sections. In the first section, we define some basic predicates and graph families in the three different graph orders. In the second, we construct gadgets based on the results of the first section to define arithmetic in graph orders. In the third, we define graph orders in arithmetic.

3.1 Defining Basic Graph Theoretic Predicates in Graph Order

The predicates defined in this section are basic in two ways: first, they correspond to graph families (such as trees, cycles, cliques etc) and numerical parameters (such as maximum degree) encountered in initial chapters of standard graph theory text books ; second, they

[1]The results in this chapter are from Ramanujam and Thinniyam [36].

[2]Mutual interpretability is a weaker notion than bi-interpretability. Unfortunately, our paper [36] wrongly uses the word bi-interpretable instead of mutually interpretable.

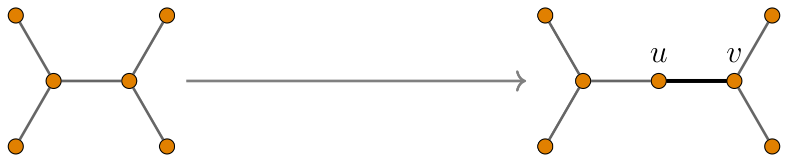


Figure 3.1: Contraction of the edge uv which lies on a path between high degree vertices cannot be simulated using deletion. The arrow is the covering relation under the minor order.

will be used in multiple chapters of this thesis.

We will take up the subgraph, minor and induced subgraph orders, in that order, in the three sections of this chapter. Many of the basic predicates in the case of the induced subgraph order are present in Wires [43]. The defining formulae used in the subgraph order can be transferred to the minor order in some special cases using the following observations.

Observation 3.1. *The downclosures of S_4, P_4, K_3 are identical under the subgraph and minor orders.*

Under some natural restrictions, the subgraph and minor orders are the same.

Observation 3.2. *If $|x| = |y|$ then $x \leq_s y$ iff $x \leq_m y$ and $y \lessdot_s x$ iff $y \lessdot_m x$. Since the contraction operation reduces the number of vertices, restricting the orders to tuples of the same cardinality makes minor and subgraph equivalent.*

Another restriction which makes the subgraph and minor orders equivalent is as follows.

Lemma 3.3. *Let x be a tree with at most one degree 3 vertex and no vertex of degree 4 or more. Then for any other graph y, $x \leq_m y$ iff $x \leq_s y$.*

Proof. It suffices to prove the only if direction since any subgraph is also a minor. The idea is that edges which are contracted between high degree nodes cannot be replaced by deletions to obtain the same graph (see Figure 3.1), but otherwise this is possible.

We observe that there is a normal form for any sequence of minor operations. Let $x = x_n$ and $y = x_0$ and $x_n \leq_m x_0$ via a sequence of minor operations $o_1, o_2, ..., o_n$, then there exists a series of minor operations $o'_1, ..., o'_m$ on x_0 resulting in x_n such that no deletion operation occurs after a contraction operation and the number of contraction operations in the sequence $o'_1, ..., o'_m$ is at most the number of contractions in the original sequence $o_1, ..., o_n$. This is because for any two operations o_i, o_{i+1} where o_i is a contraction and o_{i+1} a deletion, the only case in which the operations do not commute is when o_{i+1} is the deletion of the new vertex v formed after the contraction o_i. But since we only allow deletion of isolated vertices, it must be the case that o_i involves the contraction of an edge $u_k u_l$ which forms a component. We can replace o_i, o_{i+1} by o'_i, o'_{i+1}, o'_{i+2} which are deletion of the edge $u_k u_l$ followed by deletion of u_k and deletion of u_l. We assume this normal form for sequences of operations in the rest of the proof.

We prove the result by induction on the number of contraction operations in transforming x_0 to x_n.

Base Case: There are no contraction operations, there is nothing to be done.

For the induction step there are two cases we consider:

Case 1 : x_n has no degree 3 node. Let x_n be a path $u_0, u_1, ..., u_m$ with edges $u_i u_{i+1}$ for $0 \leq i \leq m - 1$. Let $o_1, .., o_n$ be the sequence of minor operations in normal form with x_i being obtained from x_{i-1} via operation o_i. The last operation o_n must be a contraction operation (else all operations are deletions and we are done). Therefore x_{n-1} is either a path of length $m + 1$ or a graph such that $V(x_{n-1}) = V(x_n) \cup \{u'\}$ and there exists an i with $E(x_{n-1}) = E(x_n) \cup \{u'u_i\}$ or $E(x_{n-1}) = E(x_n) \cup \{u'u_i, u'u_{i+1}\}$. In the cases where x_{n-1} is a path or $E(x_{n-1}) = E(x_n) \cup \{u'u_i\}$, we can delete an edge and a vertex in order to obtain x_n. Thus there is a sequence $o_1, .., o_{n-1}, o'_n, o'_{n+1}$ where the last two are deletion operations to obtain x_n from x_0. In the third case of $E(x_{n-1}) = E(x_n) \cup \{u'u_i, u'u_{i+1}\}$, we have to delete two edges and a vertex. In all three cases, since the new sequence has a smaller number of contractions, by induction hypothesis, x_n is a subgraph of x_0.

Case 2 : x_n has exactly one degree three node. Let x_n consist of a degree 3 node u with paths p_1, p_2, p_3 incident on u. As before, consider the sequence of minor operations. In one case x_{n-1} is a graph with a degree 3 node attached to three paths exactly one of which has length one more than previously. We can delete the end point and incident edge of the appropriate path to get x_n from x_{n-1}. Another possibility is that x_{n-1} is a graph with a vertex $u' \notin V(x_n)$ such that u' is attached to either u or a vertex v in one of the paths p_1, p_2, p_3 or two adjacent vertices in one of the paths. As before, we can delete u' and one or two incident edges to get x_n from x_{n-1}. Then by induction hypothesis x_n is a subgraph of x_{n-1}. □

Before taking up the graph orders, we have some useful observations which hold of any partial order.

Definition 3.4 (Covering Relation of a Poset)**.** *Given elements p, p' of a poset $(\mathscr{P}, \leq_P)$ we define the covering relation $p \lessdot_P p'$ as*
$p \lessdot_P p'$ if and only if $p <_P p'$ and there exists no element p'' of P such that $p <_P p'' <_P p'$.

Observation 3.5. *The covering relation of a poset $(\mathscr{P}, \leq_P)$ is first order definable using $\leq_P$:*

$$x \lessdot_P y := x <_P y \wedge \forall z \neg(x <_P z <_P y)$$

Observation 3.6. *Given a poset $(\mathscr{P}, \leq_P)$, let $\mathcal{F}_0 \subseteq \mathscr{P}$ be a definable family of graphs and $\leq'$ be a definable order on $\mathcal{F}_0$ such that $(\mathcal{F}_0, \leq')$ is isomorphic to $(\mathbb{N}, \leq)$. Then every element of $\mathcal{F}_0$ is definable.*

By assumption there is a minimum element of $\mathcal{F}_0$ under the order $\leq$ and by repeated use of the covering relation $\lessdot_P$, we can inductively construct every member of $\mathcal{F}_0$.

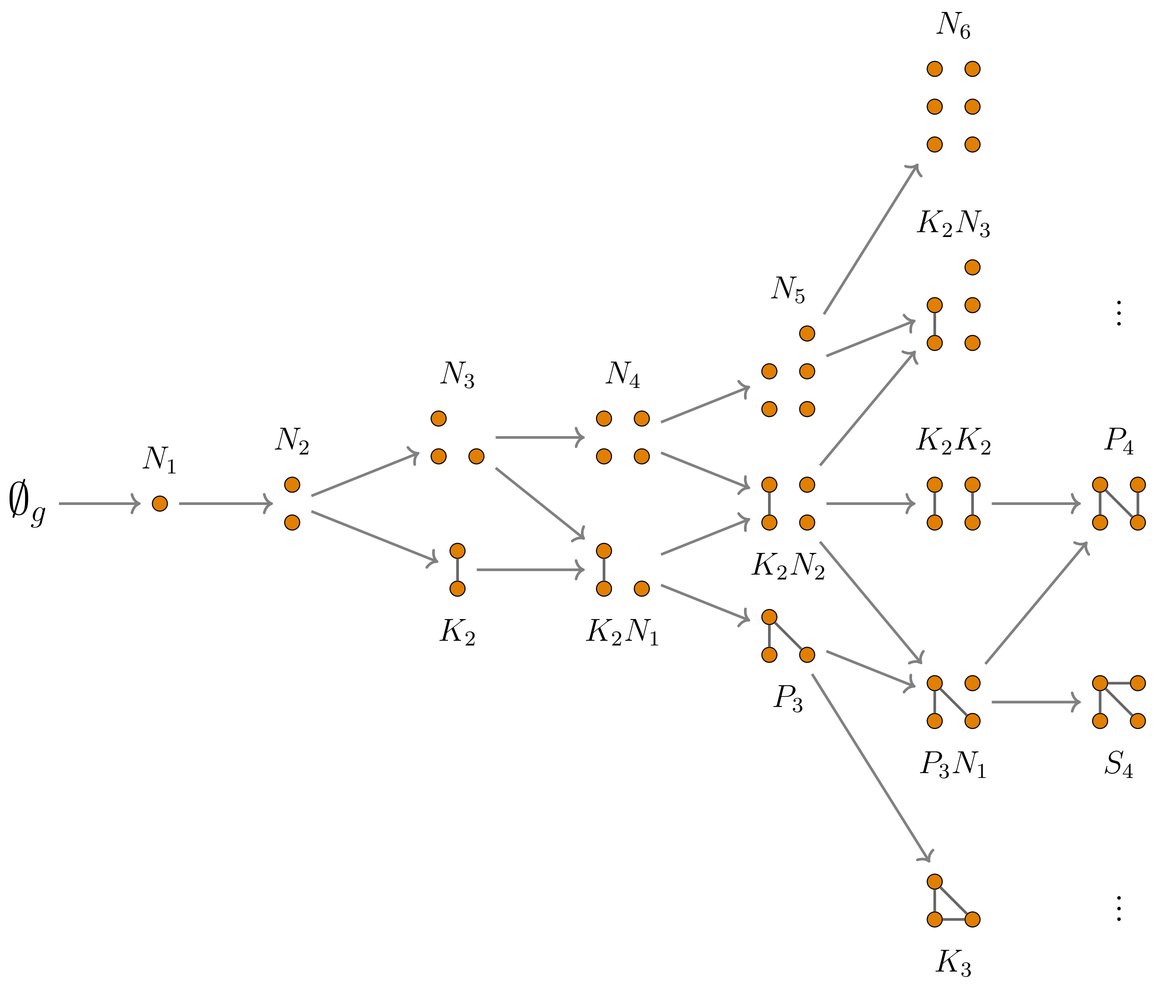


Figure 3.2: The first few layers of the subgraph order $\leq_s$. Note the lack of symmetry as compared to the induced subgraph order. The last layer containing S_4 and P_4 is incomplete.

3.1.1 Basic Predicates in the Subgraph Order

We will first define a few graphs of small cardinality. The application of atomic negation to formulae containing constants helps us define cardinality of a graph.

Lemma 3.7. *The covering relation, the order of a graph, the family $\mathcal{N}$ and the graphs N_1, K_2, K_3, S_4, P_4 are definable in the subgraph order.*

The definability of the covering relation for subgraph follows from Observation 3.5.

The following graphs in the first few layers of the subgraph order are definable. Refering to Figure 3.2, the following formulae can easily be verified:

1. $\emptyset_g(x) := \forall y\ x \leq_s y$
2. $N_1(x) := \emptyset \lessdot_s x$
3. $N_2(x) := N_1 \lessdot_s x$
4. $K_2(x) := N_2 \lessdot_s x \wedge \exists y\ x \lessdot_s y\ \wedge\ \forall z\ x \lessdot_s z \supset z = y$
5. $N_3(x) := N_2 \lessdot_s x\ \wedge\ x \neq K_2$
6. $K_2N_1(x) := K_2 \lessdot_s x$
7. $K_2N_2(x) := K_2N_1 \lessdot_s x\ \wedge\ N_4 \lessdot_s x$
8. $P_3(x) := \exists! y\ y \lessdot_s x \wedge y = K_2N_1$
9. $P_3N_1(x) := P_3 \lessdot_s x \wedge K_2N_2 \lessdot_s x\ \wedge \forall y\ y \lessdot_s x \supset (y = P_3\ \vee\ y = K_2N_2)$
10. $S_4(x) := P_3N_1 \lessdot_s x\ \wedge\ \forall y\ y \lessdot_s x \supset y = P_3N_1$
11. $K_3(x) := \exists! y\ y \lessdot_s x\ \wedge\ y = P_3$
12. $P_4(x) := P_3N_1 \lessdot_s x\ \wedge\ x \neq S_4$

The family of isolated points is now easily seen to be definable via: $\mathcal{N}(x) := K_2 \nleq_s x$. In addition, using the family $\mathcal{N}$ as a "yardstick", we can capture the cardinality (order) of a graph. The predicate $card(x, n)$ which holds if and only if $n \in \mathcal{N}$ and $|x| = |n|$ can be defined:

$$card(x, n) := \mathcal{N}(n)\ \wedge\ \forall m\ (\mathcal{N}(m)\ \wedge\ m \leq_s x) \supset m \leq_s n$$

For definable numerical predicates such as cardinality, we will simply use them as functions instead of predicates to simplify notation from here on. For instance, $|x|_g$ in a formula in the vocabulary of graph order will denote the member of $\mathcal{N}$ whose cardinality is the same as that of x. Thus we will write formulae such as $|x|_g = |y|_g$ which denotes the relation which holds if and only if x and y have the same cardinality. This relation can be defined using $card(x, n)$

$$|x|_g = |y|_g := \exists n\ card(x, n) \wedge card(y, n)$$

Next we show that several natural graph families are definable in the subgraph order.

Lemma 3.8. *The families $\mathcal{K}, \mathcal{P}, \mathcal{C}, forest, \mathcal{T}, \mathcal{S}$ are definable in the subgraph order.*

Cliques: Any graph to which an edge can be added contains at least two upper covers. The unique upper cover of a clique is formed by adding an isolated point to it.

$$\mathcal{K}(x) := \exists! y \; x \lessdot_s y$$

Paths

In order to define paths, we need to define a few additional families :

1. Disjoint unions of paths and cycles (denoted pac)
2. Disjoint unions of cycles i.e. sums of cycles (denoted soc)
3. Disjoint unions of paths i.e. forest of paths(denoted fop)

Assuming these, we can define paths :

$$\mathcal{P}(x) := fop(x) \;\wedge\; \forall y \; |x|_g = |y|_g \;\wedge\; fop(y) \supset y \leq_s x.$$

Out of all the fops of the same order n, the graph P_n forms the maximum element. Clearly by adding appropriate edges to a fop of the same order, one can form P_n. Adding any more edges to P_n gives a non-fop.

A graph is a disjoint union of paths and cycles if and only if it has maximum degree at most two:

$$pac(x) := S_4 \not\leq_s x$$

Assuming soc, fop can be defined :

$$fop(x) := pac(x) \wedge (\forall y \; soc(y) \supset y \not\leq_s x)$$

if: x is clearly a pac. Since x does not have any cycles as subgraph, it cannot have any soc as a subgraph.

only if: Let $x = c_1 \uplus c_2 \uplus ... \uplus c_n$ where c_i is either a path or a cycle, for all i. Suppose

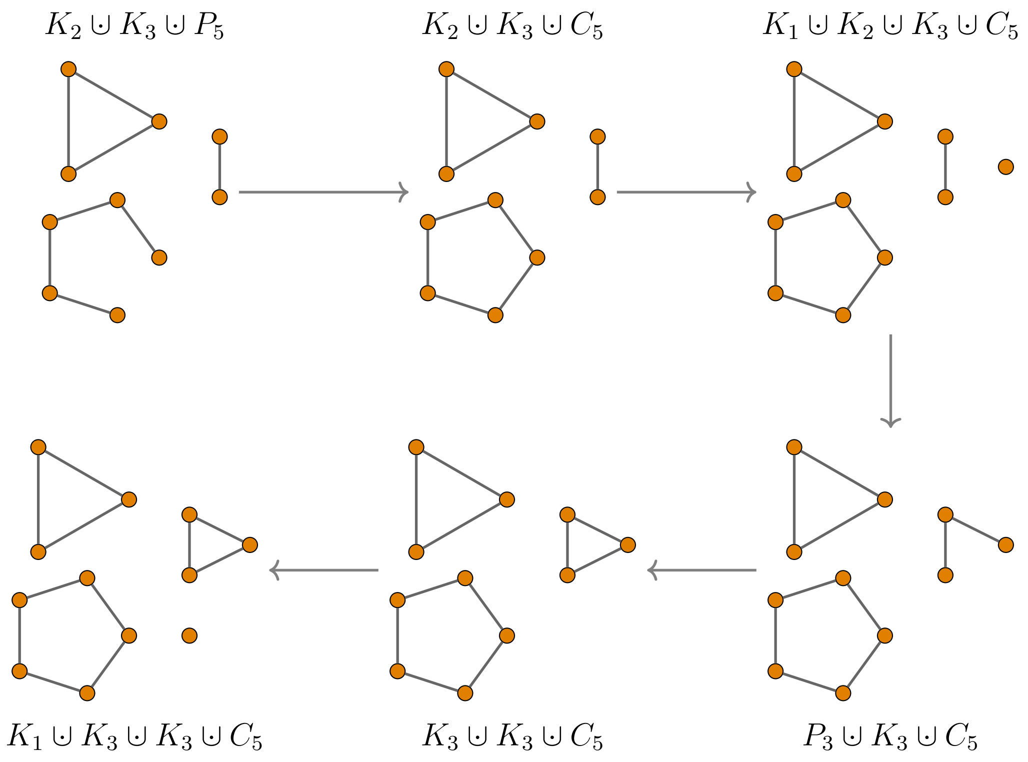

$K_2 \dot\cup K_3 \dot\cup P_5$
$K_2 \dot\cup K_3 \dot\cup C_5$
$K_1 \dot\cup K_2 \dot\cup K_3 \dot\cup C_5$
$K_1 \dot\cup K_3 \dot\cup K_3 \dot\cup C_5$
$K_3 \dot\cup K_3 \dot\cup C_5$
$P_3 \dot\cup K_3 \dot\cup C_5$

Figure 3.3: All graphs above belong to the family pac. Only $K_2 \dot\cup K_3 \dot\cup \mathsf{C}_5$, $K_1 \dot\cup K_3 \dot\cup K_3 \dot\cup C_5$, $K_3 \dot\cup K_3 \dot\cup C_5$ belong to soc' and only $K_3 \dot\cup K_3 \dot\cup C_5$ is a soc. Arrows indicate the covering relation under the subgraph order.

there is an i with c_i cycle. Then clearly $c_i \leq_s x$ but c_i is also a soc, which is a contradiction. Hence all components are paths and x is a fop.

It is only left to define disjoint unions (sums) of cycles i.e. soc:

$$soc(x) = soc'(x) \wedge \forall y\, (x \lessdot_s y \wedge pac(y)) \supset soc'(y)$$

where

$$soc'(x) := x \neq \emptyset_g \;\wedge\; pac(x) \wedge\; \forall y\, (|y|_g = |x|_g \wedge pac(y)) \supset \neg x <_s y$$

Claim 3.9. *$soc'(x)$ holds if and only if x is not the empty graph and every component of x is a cycle, N_1 or K_2 and x contains at most one copy of N_1 or one copy of K_2 but not both.*

Proof. <u>if</u>: Clearly x is a pac. Suppose there exists a pac y of the same order as x and $x <_s y$. We can obtain y from x by addition of edges. But addition of any edge would introduce a degree three node, thus such a y cannot exist.
<u>only if</u>: Let $x = c_1 \uplus c_2 \uplus ... \uplus c_n$ where c_i is either a cycle or a path. Suppose there is an i such that c_i is a path of cardinality at least three. Let c_i' be the cycle formed by joining the ends of c_i. Now $x' = c_1 \uplus ...c_{i-1} \uplus c_i' \uplus c_{i+1}... \uplus c_n$ is also a pac, $|y| = |x|$ and x can obtained from y by deleting the newly added edge to get c_i from c_i'. Thus no path of cardinality more than two can exist (see Figure 3.3 for examples). Similarly, we can obtain a contradiction in the following cases by appropriately constructing x':

1. There are two copies of K_2 in x. Join the two copies end to end to form a path of cardinality four, to get x'.
2. There are two copies of N_1 in x. Join the copies by an edge to get x'.
3. There is a K_2 and an N_1 as components in x. Join N_1 by an edge to K_2 to get a path of cardinaltiy three, to get x'.

□

Now we show the correctness of $soc(x)$.

if: Clearly x is a soc'. The only upper cover of x which is a pac is $x \uplus N_1$ since adding any more edges would lead to a degree three node. $x \uplus N_1$ is a soc'.
only if: Let $x = c_1 \uplus c_2 \uplus ... \uplus c_n$ and x is a soc'. Suppose there is i such that c_i is K_2. Consider $x' = x \uplus N_1$. The graph x' is an upper cover of x, is a pac but is not a soc' because it has an N_1 and a K_2 as components. Similarly we can rule out N_1 as a component of x.

Cycles, Forests, Trees, Stars

$$\begin{aligned}
\mathcal{C}(x) :=& pac(x) \ \wedge \ \exists y \ \mathcal{P}(y) \ \wedge |x|_g = |y|_g \ \wedge \ y \lessdot_s x \\
forest(x) :=& \forall y \ \mathcal{C}(y) \supset y \nleq_s x \\
\mathcal{T}(x) :=& forest(x) \ \wedge \ \forall y \ (forest(y) \ \wedge \ |x|_g = |y|_g) \supset \neg x <_s y \\
\mathcal{S}(x) :=& \mathcal{T}(x) \ \wedge \ P_4 \nleq_s x
\end{aligned}$$

$\mathcal{C}(x)$ i.e. cycles: It is clear that by deleting any edge from a cycle, we get a path which is a lower cover of the same order.
Conversely, consider any upper cover of a path with the same order. Adding an edge which joins the degree one vertices of the path gives a cycle, but adding an edge any where else creates a degree three vertex, which violates the condition that x is a pac. Thus only a cycle fulfills all the conditions.

$forest(x)$: A forest is a graph which contains no cycles.

$\mathcal{T}(x)$ i.e. trees: Of all forests with the same cardinality, a tree is a maximal element since adding another edge gives a cycle. A non-tree forest can be made into a tree of same order by adding appropriate edges.

$\mathcal{S}(x)$ i.e. stars: A star is a tree which does not contain a path on four vertices as subgraph.

We next take up the definability of some natural graph theoretic predicates.

Lemma 3.10. *Connectivity, maximum degree and maximum path length are definable in the subgraph order.*

Connectivity

The predicate $conn(x)$ holds if and only if x is a connected graph.

$$conn(x) := \exists y\ \mathcal{T}(y)\ \wedge\ y \leq_s x\ \wedge\ |x| = |y|$$

A graph is connected iff it has a spanning tree.

Maximum path

$maxPath(x, n)$ holds if and only if $n \in \mathcal{N}$ and the largest path which is a subgraph of x is P_n.

$$maxPath(x, n) := \mathcal{N}(n)\ \wedge\ \exists y\ \mathcal{P}(y)\ \wedge\ y \leq_s x\ \wedge |y|_g = n\ \wedge$$
$$\forall z\ (\mathcal{P}(z)\ \wedge\ z \leq_s x) \supset z \leq_s y$$

Maximum degree

$maxDeg(x, n)$ holds if and only if $n \in \mathcal{N}$ and the maximum degree of x is $|n|$.

$$maxDeg(x, n) := \mathcal{N}(n)\ \wedge\ \exists y\ \mathcal{S}(y)\ \wedge\ y \leq_s x\ \wedge\ n \lessdot_s |y|_g\ \wedge$$
$$\forall z\ (\mathcal{S}(z)\ \wedge\ z \leq_s x) \supset z \leq_s y$$

The maximum degree of x is one less than the order of the largest star which is a subgraph of x.

3.1.2 Basic Predicates in the Minor Order

Lemma 3.11. *The covering relation, the cardinality of a graph, the family $\mathcal{N}$ and the graphs N_1, K_2, K_3, P_4, S_4 are definable in the minor order.*

Proof. The definability of the covering relation follows from Observation 3.5.

By use of Observations 3.1, and the fact that the defining formulae used in Lemma 3.7 only refer to graphs in the common initial segment, we see that the subgraph order replaced by the minor order in the defining formulae define the corresponding constants in the minor order. In particular, note that the defining formulae for graphs of cardinality greater than 3 only involve atomic formulae where the free variable x always appears in the form $c \lessdot_s x, y \lessdot_s x$ where c is some constant; and for smaller graphs, $x \lessdot_s y$ implies that y is contained in the initial segment referred to in Observation 3.1.

By Lemma 3.3, $K_2 \not\leq_m x$ is equivalent to $K_2 \not\leq_s x$ and hence defines the family $\mathcal{N}$. Similarly, replacing the subgraph by the minor order gives us the defining formula for the cardinality of a graph. □

Lemma 3.12. *The families $\mathcal{K}, \mathcal{P}, \mathcal{C}, forest, \mathcal{T}, \mathcal{S}$ are definable in the minor order.*

Firstly note that a graph contains a cycle as subgraph iff it contains K_3 as a minor (by contraction along the cycle after deleting the rest of the graph). Hence forests are defined by:

$$forest(x) := K_3 \not\leq_m x$$

By Lemma 3.3, we can replace subgraph by minor in the defining formula for disjoint unions of paths and cycles (pac):

$$pac(x) := S_4 \not\leq_m x.$$

For forest of paths (fop), we can restrict a pac to be a forest, giving a fop:

$$fop(x) := forest(x) \ \wedge \ pac(x)$$

Now, using Observation 3.2, we can immediately get paths, cliques, cycles and trees; and stars by Lemma 3.3:

$$\begin{aligned}
\mathcal{P}(x) :=& fop(x) \ \wedge\ \forall y\ (|x|_g = |y|_g \ \wedge\ fop(y)) \supset y \leq_m x \\
\mathcal{K}(x) :=& \forall y\ |y|_g = |x|_g \supset y \leq_m x \\
\mathcal{C}(x) :=& pac(x) \ \wedge\ \exists y\ (\mathcal{P}(y) \ \wedge |x|_g = |y|_g \ \wedge\ y \lessdot_m x) \\
\mathcal{T}(x) :=& forest(x) \ \wedge\ \forall y\ (forest(y) \ \wedge\ |x|_g = |y|_g) \supset \neg x <_m y \\
\mathcal{S}(x) :=& \mathcal{T}(x) \ \wedge\ P_4 \not\leq_m x
\end{aligned}$$

Lemma 3.13. *Connectivity, maximum degree and maximum path length are definable in the minor order.*

Proof. Application of Observation 3.2 and Lemma 3.3 give defining formulae for connectivity and $maxPath$:

$$\begin{aligned}
conn(x) :=& \exists y\ \mathcal{T}(y) \ \wedge\ y \leq_m x \ \wedge\ |x|_g = |y|_g \\
maxPath(x,n) :=& \mathcal{N}(n) \ \wedge\ \exists y\ \mathcal{P}(y) \ \wedge\ y \leq_m x \ \wedge |y|_g = n \ \wedge \\
& \forall z\ (\mathcal{P}(z) \ \wedge\ z \leq_m x) \supset z \leq_m y
\end{aligned}$$

It remains to define $maxDeg(x,n)$ which holds if and only if $n \in \mathcal{N}$ and the maximum degree of x is $|n|$.

Here we need to do some more work since the largest star which is a minor of x may be much larger than the maximum degree of the graph. In order to apply Observation 3.2, we construct the following family:

$\mathcal{S} \uplus \mathcal{N}(x)$ holds if and only if x is formed by addition of some arbitrary number of isolated vertices to a star.

$$\begin{aligned}
\mathcal{S} \uplus \mathcal{N}(x) :=& \mathcal{F}(x) \ \wedge \ \exists y \ hasStarComp(x,y) \ \wedge \ onlyStar(x,y) \\
& \text{where} \\
hasStarComp(x,y) :=& \mathcal{S}(y) \ \wedge \ y \leq_m x \wedge \ \forall z \ conn(z) \ \wedge \ z \leq_m x \supset z \leq_m y \\
onlyStar(x,y) :=& \forall x' \ \mathcal{F}(x') \ \wedge \ |x'|_g = |x|_g \ \wedge \ x \lessdot_m x' \supset \\
& \forall y' \ (conn(y') \ \wedge \ y' \leq_m x') \supset \ |y|_g \lessdot_m |y'|_g
\end{aligned}$$

$hasStarComp$ asserts that y is a star minor of x and in addition, every connected minor of x is also a minor of y. To fulfill this condition, x has to contain y as a connected component. However, x could contain multiple copies of subgraphs of y.

$onlyStar$ asserts that any forest x' which is formed by adding an edge to x (by Observation 3.2) has the property that all its connected minors have order one more than the order of y. Clearly, any graph formed by adding isolated vertices to a star has these properties. Conversely, consider a graph g satisfying the formula $\mathcal{S} \uplus \mathcal{N}(x)$. Then g has a connected component c_0 which is a star such that $hasStarComp(g, c_0)$ holds. Suppose g had another component $c_1 \neq N_1$. Then by adding an edge between c_0 and c_1, we get a graph g' such that $|g'| > |c_0| + 1$ and g' is a minor of g'' which is a forest formed by adding an edge to g. This contradicts the formula $onlyStar$, and hence g is the union of c_0 with some number of isolated points.

$$\begin{aligned}
maxDeg(x,n) :=& \exists y \ \mathcal{S} \uplus \mathcal{N} subgraph(x,y) \ \wedge \ \forall z \ \mathcal{S} \uplus \mathcal{N} subgraph(x,z) \supset z \leq_m y \\
& \wedge \ \exists z' \ \mathcal{S}(z') \ \wedge \ n \lessdot_m |z'|_g \ \wedge \ z' \leq_m y \\
& where \\
\mathcal{S} \uplus \mathcal{N} subgraph(x,y) :=& \mathcal{S} \uplus \mathcal{N}(y) \ \wedge \ |x|_g = |y|_g \ \wedge \ y \leq_m x
\end{aligned}$$

$\mathcal{S} \uplus \mathcal{N} subGraph$ states that there is a subgraph y of x which is a $\mathcal{S} \uplus \mathcal{N}$ of same cardinality as x. Note that for $S_n \uplus N_m$ and $S_{n'} \uplus N_{m'}$ with $n + m = n' + m'$, $S_n \uplus N_m \leq_m S_{n'} \uplus N_{m'}$

iff $n \leq n'$. Thus the maximal y satisfying the formula $\mathcal{S} \uplus \mathcal{N} subGraph$ contains the largest star occuring as a subgraph of x. We extract the star from this object to obtain the maximum degree of x. $\square$

3.1.3 Basic Predicates in the Induced Subgraph Order

In this subsection, we summarize some definability results from Wires [43] needed for the next section on arithmetical definability in graph order.

Lemma 3.14 (Wires [43]). *The following predicates are definable in the induced subgraph order:*

1. *The families $\mathcal{N}, \mathcal{T}, \mathcal{P}$ of isolated points, trees and paths respectively.*
2. *$|x| = |y|$ if and only if x and y have the same cardinality (i.e. same number of vertices, also known as order of the graph).*

We need to define the family of stars.

Observation 3.15. *The predicate $\mathcal{S}(x)$ which holds if and only if x is a star, is definable in the induced subgraph order.*

$$\mathcal{S}(x) := \mathcal{T}(x) \ \wedge \ P_4 \not\leq_i x$$

As usual, it is easy to see that the conditions specified are necessary. In a tree, a path is present as a subgraph iff it is present as an induced subgraph. Any graph containing P_4 as a subgraph cannot be a star.

3.2 Defining Arithmetic in Graph Order

We use the family $\mathcal{N}$ to represent numbers. This is a natural choice which is very convenient in the case of the subgraph order. Recall (see Lemma 3.7) that this immediately allows us to define cardinality in the subgraph and minor orders. However, consider the same formula in the induced subgraph order:

$$\alpha(x) = \alpha(y) := \forall z\, \mathcal{N}(z) \supset (z \leq_i x \iff z \leq_i y)$$

This defines the binary predicate $\alpha(x) = \alpha(y)$ which holds if and only if the graphs x and y have the same independence number. It turns out that defining $|x| = |y|$ in the induced subgraph order requires much more work. Wires defines addition initially using the family $\mathcal{P}$ of paths instead [43]; though he shows later that one can also use the family $\mathcal{N}$. Since we will be constructing explicit formulae which use arithmetical definability in later chapters, it is convenient to work with the same representation of numbers for the sake of uniformity.

Definition 3.16. *By definability of arithmetic in graph order we mean the existence of defining formulae for the following predicates:*

1. *$\mathcal{N}(x)$ holds if and only if x belongs to $\mathcal{N}$.*
2. *$plus(z, x, y)$ holds if and only if $x, y, z \in \mathcal{N}$ and $|x| + |y| = |z|$.*
3. *$times(z, x, y)$ holds if and only if $x, y, z \in \mathcal{N}$ and $|x| \times |y| = |z|$.*

Since the formula $K_2 \not\leq x$ defines the family $\mathcal{N}$ in each of the graph orders, we need to exhibit defining formulae for $plus$ and $times$, which we take up in the next two subsections.

3.2.1 Defining Addition in Graph Order

The definability of addition in the induced subgraph order has already been established :

Lemma 3.17 (Wires [43])**.** *The predicate $plus(z, x, y)$ if and only if $x, y, z \in \mathcal{N}$ and $|x| + |y| = |z|$ is definable in the induced subgraph order.*

We take up the definability of addition in the subgraph and minor orders.

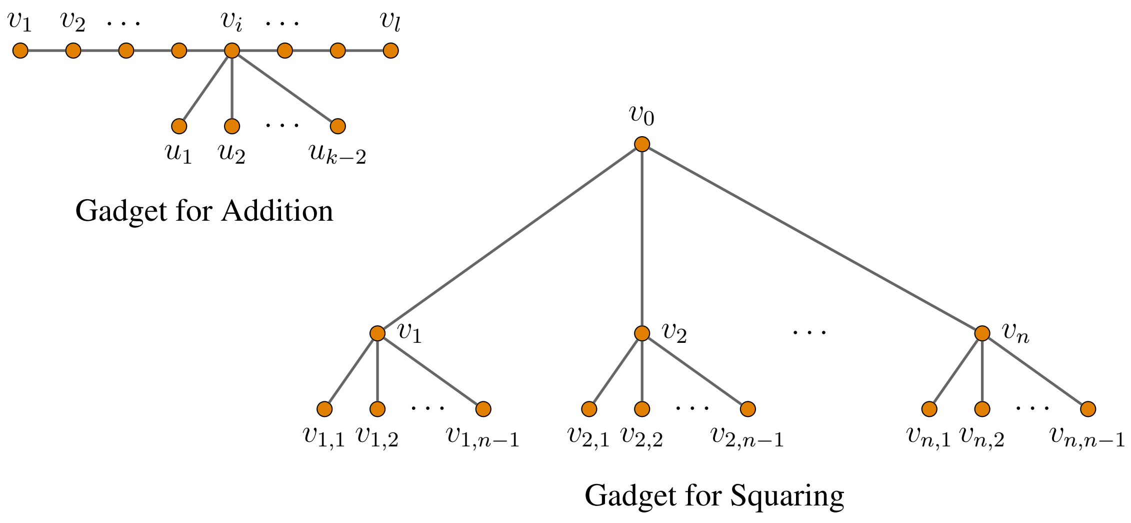
Lemma 3.18. *The predicate* $plus(z, x, y)$ *if and only if* $x, y, z \in \mathcal{N}$ *and* $|x| + |y| = |z|$ *is definable in the subgraph and minor orders.*

Proof.

$$\begin{aligned}
plus(m, k, l) :=& \mathcal{N}(k) \ \wedge\ \mathcal{N}(l) \ \wedge\ \mathcal{N}(m) \wedge \\
& (initial(m, k, l) \vee (N_3 \leq_s k \ \wedge\ N_3 \leq_s l \wedge \\
& \quad \exists x \ starTail(k, l, x) \ \wedge\ plus2(m, x))); \text{ where} \\
starTail(x, k, l) :=& starTail'(x, k, l) \ \wedge\ \forall x' \ (starTail'(x', k, l) \supset |x|_g \leq_s |x'|_g) \\
starTail'(x, k, l) :=& \mathcal{T}(x) \ \wedge\ maxDeg(x) = k \ \wedge\ maxPath(x) = l \\
plus2(x, m) :=& \exists m' \ x \lessdot_s |m'|_g \ \wedge\ |m'|_g \lessdot_s m_g \\
initial(m, k, l) :=& (k = \emptyset_g \ \wedge\ m = l) \vee \ (l = \emptyset_g \ \wedge\ m = k) \vee \\
& (k = N_1 \ \wedge\ l \lessdot_s m) \vee \ (l = N_1 \ \wedge\ k \lessdot_s m) \vee \\
& (k = N_2 \ \wedge\ \exists m' \ l \lessdot_s |m'|_g \ \wedge\ |m'|_g \lessdot_s m) \vee \\
& (l = N_2 \ \wedge\ \exists m' \ k \lessdot_s |m'|_g \ \wedge\ |m'|_g \lessdot_s m)
\end{aligned}$$

When either k or l are strictly less than two, we hardcode the function value using $initial$. When both are at least three, consider a tree with maximum degree k and maximum path l. A tree of least order with these properties is formed from a path by choosing some degree two vertex v_i of the path $v_1, v_2, ..., v_i, v_{i+1}, ..., v_l$, adding $k - 2$ new vertices $u_1, u_2, ..., u_{k-2}$ and adding the edges $u_1v_i, u_2v_i, ..., u_{k-2}v_i$ (see Figure 3.4). The order of this tree is $k + l - 2$. This is captured in the formula $starTail$ and in $plus2$ we add two to its cardinality to get $k + l$.

By Observations 3.1 and 3.2; and Lemma 3.3, the defining formula above can be transferred to the minor order simply by replacing the subgraph order by the minor order. $\square$



v_1 v_2 ... v_i ... v_l
u_1 u_2 ... u_{k-2}
Gadget for Addition
v_0
v_1 v_2 ... v_n
$v_{1,1}$ $v_{1,2}$ ... $v_{1,n-1}$
$v_{2,1}$ $v_{2,2}$ ... $v_{2,n-1}$
$v_{n,1}$ $v_{n,2}$ ... $v_{n,n-1}$
Gadget for Squaring

Figure 3.4: Top Left: minimum cardinality tree with maximum degree k and maximum path subgraph P_l. Bottom Right : tree t_n containing $n^2 + 1$ vertices.

3.2.2 Defining Multiplication in Graph Order

Instead of the formula $times(z, x, y)$, we can equivalently define the square predicate $square(x, y)$ if and only if $x, y \in \mathcal{N}$ and $|x| = |y|^2$. Definability of multiplication follows from definability of addition and the equality

$$(n + m)^2 = n^2 + m^2 + 2mn$$

To define squaring, we construct a tree t_n given a numerical parameter n such that its cardinality is $n^2 + 1$.

Definition 3.19. *Define the tree t_n by*

$V(t_n) = \{v_0\} \cup \{v_1, v_2, ..., v_n\} \cup \bigcup_{i=1}^{n} \{v_{i,1}, v_{i,2}, ..., v_{i,n-1}\}$,

$E(t_n) = \{v_0v_1, v_0v_2, ..., v_0v_n\} \cup \bigcup_{i=1}^{n} \{v_iv_{i,1}, v_iv_{i,2}, ..., v_iv_{i,n-1}\}$.

The predicate $stree(x, n)$ holds if and only if $x = t_n$.

Lemma 3.20. *If the predicates $\mathcal{N}$, $|x| = |y|$, $|x| = |y| + 1$ and $stree(x, n)$ are definable in a structure $(\mathcal{G}, \tau)$ over graphs, then the predicate $square(x, y)$ which holds if and only if $x, y \in \mathcal{N}$ and $|x| = |y|^2$ is definable in $(\mathcal{G}, \tau)$.*

Proof. The defining formula for the predicate $square(x, y)$ is given below.

$$square(x, y) := \mathcal{N}(x) \ \wedge \ \mathcal{N}(y) \ \wedge \ \exists z \ stree(z, y) \ \wedge \ |z|_g = x + 1$$

The tree t_n has a root v_0 which has degree n and each of its neighbours also has degree n. It is easy to see that $|t_n| = 1 + n + n(n-1) = n^2 + 1$ (see Figure 3.4). □

Lemma 3.21. *Multiplication is definable in the induced subgraph, subgraph and minor orders.*

Proof. We note that $x + 1 = y$ for $x, y \in \mathcal{N}$ can be defined by the formula $x \lessdot y$ in each of the three graph orders. Since all the predicates in the hypothesis of the above lemma except for $stree$ are definable in each graph order, the definability of squaring (and thus multiplication) is reduced to defining $stree$.

The predicate $stree$ can be defined in the induced subgraph order in two steps. First we define $stree'$:

$$\begin{aligned} stree'(x, n) :=& maxDegInTrees(x, n) \ \wedge \ maxPath5(x) \\ & where \\ maxDegInTrees(x, n) :=& \mathcal{T}(x) \ \wedge \ S_n \leq_i x \ \wedge \ \forall m \ (\mathcal{N}(m) \ \wedge \ m > n \supset S_m \not\leq_i x) \\ maxPath5(x) :=& \mathcal{T}(x) \ \wedge \ P_5 \leq_i x \ \wedge \ \forall m \ (\mathcal{N}(m) \ \wedge \ m > 5 \supset P_m \not\leq_i x) \end{aligned}$$

The maximum degree of a forest is one less than the largest star which is an induced subgraph. Let x be a tree satisfying $stree'(n, x)$. Fix the vertex v_0 as root of this tree. The degree condition implies that v_0 has at most n neighbouring vertices and the $maxPath5$ condition ensures that the maximum depth of the tree is two. In the second step, we define $stree(x, n)$ using $stree'$:

$$stree(x, n) := stree'(x, n) \ \wedge \ \forall y \ stree'(y, n) \supset |y| \leq_i |x|$$

The tree of maximum cardinality satisfying $stree'$ is exactly the tree t_n.

In the case of the subgraph and minor orders, we already have the definability of the predicates $maxDeg, maxPath$ from Lemma 3.14:

$$stree'(x,n) := \mathcal{T}(x) \ \wedge \ maxDeg(x,n) \ \wedge \ maxPath(x, N_5)$$
$$stree(x,n) := stree'(x,n) \ \wedge \ \forall y \ stree'(y,n) \supset |y|_g \leq_s |x|_g$$

The formula for the minor order is obtained simply by replacing occurrences of the subgraph order by the minor order. □

Thus by Lemmas 3.17, 3.18 and 3.21 we have the following theorem.

Theorem 3.22. *First order arithmetic is definable in the induced subgraph, subgraph and minor orders.*

3.3 Defining Graph Orders in Arithmetic

We show that the structures ($\mathcal{G}, \leq_s$), ($\mathcal{G}, \leq_i, P_3$) and ($\mathcal{G}, \leq_m$) can be interpreted in first order arithmetic. While this is not unexpected, it completes the proof of mutual interpretability. This section is broken up into two subsections. In the first subsection we define some basic arithmetic predicates which allow us to manipulate the binary encodings of numbers. This allows us to define the number representation NR introduced in Definition 2.20 in which multiple numbers represent the same graph. We then define some relations in arithmetic related to permutations on the set $[n]$. This makes it possible for us to define the representation UN introduced in the Preliminaries 2.21. To distinguish between NR and UN we will use the phrases 'represent' and 'uniquely represent' respectively when talking about representations. In the second subsection, we use the basic predicates defined in the first to define the graph orders.

3.3.1 Basic Predicates in Arithmetic

We give defining formulae for some basic predicates in arithmetic.

Lemma 3.23. *The following predicates are definable in first order arithmetic.*

1. $nchoose2(x, n)$ *holds if and only if* $x = \binom{n}{2}$ *where* $\binom{n}{2} = n \times (n-1)/2$.
2. $div(x, y, n)$ *holds if and only if* n *is the quotient when* x *is divided by* y*; denoted* $n = \lfloor \frac{x}{y} \rfloor$.
3. $x|y$ *holds if and only if there exists* n *such that* $n \times x = y$ *i.e.* x *divides* y.
4. $prime(x)$ *holds if and only if* x *is a prime.*
5. $rem(x, y, n)$ *holds if and only if* n *is the remainder when* x *is divided by* y*; denoted* $n = rem(x, y)$.
6. $exp(x, y, z)$ *holds if and only if* $x^y = z$.
7. $pow2(x, i)$ *holds if and only if* $x = 2^i$.
8. $bit(x, i)$ *holds if and only if the* i^{th} *bit (counting from the least significant bit) of the binary representation of* x *is a 1.*
9. $length(x, n)$ *holds if and only if the length of the binary representation of* x *is* n*. We will denote this unique* n *by* $|x|$.
10. *For any binary representation* w*, let* $w[i, j]$ *for* $1 \le i \le j \le |w|$ *be the infix (i.e. contiguous segment) of* w *beginning at the* i^{th} *most significant bit and ending at the* j^{th} *most significant bit. The predicate* $window(k, i, x, n)$ *holds if and only if the number whose binary representation (ignoring leading* 0*'s) is* $x[|x| - ki, |x| - 1 - k(i-1)]$ *is* n *and* $|x| \ge ki + 1$.

    By $window(k, i, x)$ *we mean the unique* n *such that* $window(k, i, x, n)$ *holds.*
11. $prime(x, i)$ *if and only if* x *is the* i^{th} *prime. We denote it by* p_i.

Proof.

$$nchoose2(x, n) := 2 \times x + n = n^2$$

$$div(x, y, n) := \exists z \; x = y \times n + z \;\wedge\; z < y$$

$$x|y := \exists n\ n \times x = y$$

$$prime(x) = \forall y\ y|x \supset (x = y \vee y = 1)$$

$$rem(x, y, n) := \exists z\ x = y \times z + n\ \wedge\ n < y$$

$$\begin{aligned} exp(x, y, z) :=& \exists x_1\ \exists x_2\ pseq(x_1, x_2, 0, 1)\ \wedge\ pseq(x_1, x_2, y, z)\ \wedge \\ & \forall y_1\ \forall y_2\ (y_1 < y\ \wedge\ pseq(x_1, x_2, y_1, y_2)) \supset pseq(x_1, x_2, y_1 + 1, xy_2) \\ & \text{where} \\ pseq(x, y, n, w) :=& rem(x, y(n + 1) + 1, w) \end{aligned}$$

The intended meaning of the above formula is that there exists a sequence of numbers $i_1, i_2, ..., i_y$ with $i_1 = 1$ and $i_y = z$ and consecutive numbers i_j, i_{j+1} are related by $xi_j = i_{j+1}$. We refer the reader to the text by Kaye [26] for the correctness of the formula.

$$pow2(x, i) := exp(2, i, x)$$

$$bit(x, i) := \frac{rem(x, 2^i) - rem(x, 2^{i-1})}{2^{i-1}} = 1$$

$$length(x, n) := bit(x, n)\ \wedge\ \forall n'\ n < n' \supset \neg bit(x, n')$$

$$\begin{aligned} window(k, i, x, n) :=& |x| \geq ki + 1\ \wedge\ |n| \leq k\ \wedge \\ & \forall j\ 1 \leq j \leq k \supset\ (bit(n, j) \iff bit(x, |x| - ki - 1 + j)) \end{aligned}$$

$$\begin{aligned}
prime(x,i) :=& prime(x) \ \wedge\ \exists y\ |y| = 1 + 2|x|x \ \wedge\ \forall 1 \leq j \leq x \\
& window(|x|, 2j-1, y) = j \ \wedge\ window(|x|, 2, y) = 0 \\
& \wedge\ ((prime(window(|x|, 2j+1, y)) \ \wedge\ j < x) \\
& \quad \supset window(|x|, 2j, y) + 1 = window(|x|, 2j+2, y)) \\
& \wedge\ window(|x|, 2x, y) = i
\end{aligned}$$

The formula above encodes a naive algorithm for finding the i^{th} prime, namely run through all the numbers up to x and increment a counter every time a prime is found. The number y represents this computation via its binary encoding which is $1 + 2|x|x$ bits long. Starting from the most significant digit of y, every window of size $|x|$ encodes a number. The windows in the odd positions correspond to numbers from 1 to x while the even position windows represent the counter, with window at $i = 2$ initialized to 0. We check that the final window representing the value of the counter at the end of the process is i. $\square$

Using the above predicates, we can now define some of the required graph predicates in arithmetic.

Lemma 3.24. *The following predicates are definable in arithmetic:*

1. *$NRinv(x)$ holds if and only if x is a number which represents a graph i.e. there exists an image $g = NR(x)$ under the map NR.*
2. *$graphOrder(x, n)$ holds if and only if $NR(x)$ holds and $|g_{(x)}| = n$.*
3. *$edgeExists(x, i, j)$ holds if and only if $NR(x)$ holds and $v_i v_j \in E(g_{(x)})$.*

Proof. Recall that we use $g_{(x)}$ to denote the graph $NR(x)$ and whenever we talk of a vertex v_i of $g_{(x)}$ we mean the i^{th} vertex in the order induced by the representation x on $g_{(x)}$.

$$NRinv(x) := x = 0 \ \vee\ x = 1 \ \vee\ \exists n\ |x| = (1 + \binom{n}{2})$$

$$graphOrder(x,n) := NRinv(x) \ \wedge \ 2 \times |x| = 2 + n \times (n-1)$$

$$\begin{aligned} edgeExists(x,i,j) :=& \exists n \ graphOrder(x,n) \wedge \\ & ((1 \le i < j \le n \ \wedge \ bit(x, ((j^2 - 3j + 2)/2) + i) \\ & \vee (1 \le j < i \le n \ \wedge \ bit(x, ((i^2 - 3i + 2)/2) + j) \end{aligned}$$

To assert that there is an edge from v_i to v_j in $g_{(x)}$ (where $i < j$), note that the tuple (v_j, v_i) occurs at bit position $\Sigma_{k=1}^{j-1}(k-1) + i = \frac{j^2-3j+2}{2} + i$. □

Any permutation of vertices of a ordered graph induces a permutation on the edges of a graph. To identify all numbers which represent the same graph under the map NR, we will need to represent permutations on $[n]$ and their actions. This helps us define the map UN as well as to identify when a graph is a subgraph of another using number representations.

Definition 3.25. *We represent a permutation by a bit string which is of length $n \times |n| + 1$. The most significant bit is ignored, after which every block of $|n|$ bits represents a number from 1 to n. Additionally, every number in $[n]$ occurs exactly in one block. The permutation sends $i \in [n]$ to the number represented by the i^{th} block from the left. Let $perm(x,n)$ hold if and only if x represents a permutation on $[n]$.*

Lemma 3.26. *The following predicates are definable in arithmetic:*

1. *$perm(x,n)$ holds if and only if x represents a permutation on $[n]$.*
2. *$applyperm(x,i,j,n)$ holds if and only if x is a permutation on $[n]$ and sends i to j for $i,j \in [n]$.*
3. *$sameGraph(x,y)$ holds if and only if $g_{(x)} = g_{(y)}$.*
4. *$UNinv(x)$ holds if and only if $NRinv(x)$ and x is the smallest number representing $g_{(x)}$.*

Proof.

$$perm(x,n) := |x| = 1 + n \times |n| \ \wedge \ \forall i \; 1 \le i \le n \; \exists! j \; 1 \le j \le n$$
$$i = window(j, |n|, x)$$

$$applyperm(x,i,j,n) := perm(x,n) \ \wedge \ i \le n \ \wedge \ j \le n \wedge$$
$$j = window(i, |n|, x)$$

$$sameGraph(x,y) := \exists n \; |x| = |y| = 1 + \binom{n}{2} \ \wedge \ \exists z \; perm(z,n) \wedge$$
$$\forall i \, \forall j \; 1 \le i < j \le n \supset \ (edgeExists(x,i,j) \iff (\exists i' \exists j' \; applyPerm(z,i,i',n)$$
$$\wedge \ applyPerm(z,j,j',n) \ \wedge edgeExists(y,i',j') \ ))$$

The formula above states that for x and y to represent the same graph, there must exist a permutation z such that for any tuple (v_i, v_j) of vertices of x, $v_iv_j \in E(g_{(x)})$ iff $v_{z(i)}v_{z(j)} \in E(g_y)$.

$$UNinv(x) := NRinv(x) \ \wedge \ \forall y \; sameGraph(x,y) \supset x \le y$$

□

3.3.2 Defining Graph Orders in Arithmetic

We have all the intermediate predicates required to define the subgraph and induced subgraph orders in arithmetic, which we proceed to do now.

Lemma 3.27. *The following predicates are definable in arithmetic:*

1. *$subGraph(x,y)$ holds if and only if x, y uniquely represent graphs and $g_{(x)}$ is a*

subgraph of g_y.

2. *$inSubGraph(x, y)$ holds if and only if x, y uniquely represent graphs and $g_{(x)}$ is an induced subgraph of y.*
3. *$P_3(x)$ holds if and only if $x = UN(P_3)$.*

Proof.

$$subGraph(x, y) := UNinv(x) \ \wedge \ UNinv(y) \ \wedge \ |g_{(x)}| \leq |g_y| \wedge$$
$$\exists z \ sameGraph(y, z) \ \wedge \ \forall k \ (1 \leq k \leq |x| - 1) \ \supset \ (bit(x, k) \supset bit(z, k))$$

If $g_{(x)} \leq_s g_{(y)}$, then there must exist a permutation of vertices of $g_{(y)}$ (w.r.t to the order specified by y) to give another representation z of g_y such the vertices forming the witness for the subgraph relation are an initial segment of $[|g_{(y)}|]$ and moreover, the ordering among these is the same as the ordering imposed by x. This implies that the binary expansion of x (with the most significant bit removed) occurs as an initial segment of the binary expansion of z.

We can define induced subgraph by a small modification in the subgraph formula as follows:

$$inSubGraph(x, y) := UNinv(x) \ \wedge \ UNinv(y) \ \wedge \ |g_{(x)}| \leq |g_y| \wedge$$
$$\exists z \ sameGraph(y, z) \ \wedge \ \forall k \ (1 \leq k \leq |x| - 1) \ \supset \ (bit(k, x) \iff bit(k, z))$$

The formula for P_3 just requires us to identify $UN(P_3)$.

$$P_3(x) := (x = 11)$$

The correctness of the above formula is clear from Figure 2.2. $\square$

In order to define the minor relation in arithmetic, we use the following alternate definition of the minor relation (see Diestel [10]):

Definition 3.28. *Let x be a graph on the vertex set $V = \{v_1, v_2, ..., v_n\}$ and y be a graph on the vertex set $U = \{u_1, u_2, ..., u_m\}$. Then, x is a minor of y iff there exist disjoint subsets $S_1, S_2, ..., S_n$ of U such that the graph $y[S_i]$ induced by each S_i is connected and for any edge v_iv_j of x, there exists $u_k \in S_i, u_l \in S_j$ such that u_ku_l is an edge of y.*

In order to use the above definition, we need a way to encode sets of vertices, the notion of an element in a set etc. to get a formula in the language of arithmetic.

Definition 3.29. *A nonempty sequence of numbers $i_1, i_2, ..., i_n$ is encoded as the number $2^{i_1+1} \times 3^{i_2+1} \times ...p_j^{i_j+1}...p_n^{i_n+1}$ where p_j stands for the j^{th} prime. Corresponding to this definition of sequence we have the predicate $seq(x)$ if and only if x is a number which encodes a sequence.*

Lemma 3.30. *The following predicates are definable in arithmetic:*

1. *$seq(x)$ holds if and only if x is a number which encodes a sequence.*
2. *$index(x, i, p)$ holds if and only if x is a sequence and p is a prime such that the largest power of p dividing x is $p^i + 1$. We will write $i \in x$ if and only if $\exists p\ index(x, i, p)$.*
3. *$set(x)$ holds if and only if x is a sequence, any number occurs only once in the sequence and for any pair of numbers $m < n$ occuring in the sequence x, m occurs before n. We will write $s \in x$ to denote that s belongs to the set x.*
4. *$extract(s, i, n)$ holds if and only if s is a set and the i^{th} smallest element of s is n. We denote the unique n by $s(i)$.*
5. *$cardSet(s, n)$ holds if and only if s is a set and its cardinality is n. We will denote the unique n satisfying $cardSet(s, n)$ by $|s|_{set}$.*

Proof. We give the defining formulae below, the proof of correctness is straightforward.

$$seq(x) := x \neq 0 \ \wedge \ x \neq 1 \ \wedge \ \forall p\, prime(p) \ \wedge \ p|x \supset$$
$$\forall p' \ (prime(p') \ \wedge \ p' < p) \supset p'|x$$

$$index(x,i,p) := seq(x) \ \wedge \ prime(p) \ \wedge \ p^{i+1}|x \ \wedge \ p^{i+2} \not| x$$

$$\begin{aligned} set(x) :=& seq(x) \ \wedge \ \forall i \, \forall j \ (i \in x \wedge \ j \in x) \supset \\ & (unique(i,x) \ \wedge \ ordered(i,j,x)) \\ & \text{where} \\ unique(i,x) :=& \exists! p \; index(x,i,p) \\ ordered(i,j,x) :=& \exists p \, \exists p' \; index(x,i,p) \ \wedge \ index(x,j,p') \ \wedge (i < j \supset p < p') \end{aligned}$$

$$extract(s,i,n) := set(s) \ \wedge \ index(s,n,p_i)$$

$$cardSet(s,n) := set(s) \ \wedge \ p_n|s \ \wedge \ p_{n+1} \not| s$$

□

Lemma 3.31. *The predicate* $minor(x,y)$ *if and only if* $g_{(x)} \ \leq_m \ g_{(y)}$ *is definable in arithmetic.*

Proof. First we need the following intermediate predicate:
$induced(s,x,y)$ if and only if x, y represent graphs, s is a set which is a subset of $[|g_{(x)}|]$ and $g_{(y)} = x[s]$ where $x[s]$ is the graph induced by s on x.

$$\begin{aligned} induced(s,x,y) :=& set(s) \ \wedge \ NRinv(x) \ \wedge \ NRinv(y) \ \wedge \ |g_{(y)}| = |s|_{set} \wedge \\ & \forall i_1 \, \forall i_2 (1 \leq i_1 < i_2 \leq |g_{(y)}|) \supset \\ & (edgeExists(i_1, i_2, y) \iff \quad edgeExists(s(i_1), s(i_2), x)) \end{aligned}$$

We can now define the predicate $minor(x, y)$.

$$\begin{aligned}
minor(x, y) :=& UNinv(x) \ \wedge \ UNinv(y) \ \wedge \ \exists s \ set(s) \ \wedge \ \forall q \in s \ set(q) \wedge \\
& disjoint(x, y, s) \ \wedge \ edges(x, y, s) \ \wedge \ connCond(x, y, s) \\
& \text{where} \\
disjoint(x, y, s) :=& \forall q_1, q_2 \in s(\forall i \ (1 \leq i \leq |g_{(x)}|) \ i \in q_1 \supset i \notin q_2) \wedge \\
& \forall j \ (j \in q_1) \supset (1 \leq j \leq |g_{(x)}|) \\
edges(x, y, s) :=& \forall i_1 \ \forall i_2 \ (1 \leq i_1 < i_2 \leq |g_{(x)}|) \ edgeExists(i_1, i_2, x) \supset \\
& \exists j_1 \in s(i_1) \ \exists j_2 \in s(i_2) \ edgeExists(j_1, j_2, y) \\
connCond(x, y, s) :=& \forall q \in s \ \exists z \ induced(q, y, z) \ \wedge \ conn(g_z)
\end{aligned}$$

The set s is a set of subsets $q_1, q_2, ...$ of the vertex set of g_y. The formula $disjoint(x, y, s)$ ensures that the sets q_i of s are all disjoint. The formula $edges(x, y, s)$ ensures that if there is an edge between two vertices of $g_{(x)}$ then there is an edge between the appropriate sets of vertices in s and $connCond(x, y, s)$ ensures that the graph induced by each of the q_i is connected. Note that the existence of a formula $conn(g_z)$ which states that the graph g_z is connected follows from two facts: first that connectivity is definable using the subgraph order by Lemma 3.10 and second that the subgraph order is definable in arithmetic by Lemma 3.27. □

Using Lemmas 3.31 and 3.27 we have the following theorem:

Theorem 3.32. *The structures $(\mathcal{G}, \leq_m)$, $(\mathcal{G}, \leq_s)$ and $(\mathcal{G}, \leq_i, P_3)$ are definable in $(\mathbb{N}, +, \times)$.*

Combining the above result with Theorem 3.22 we have:

Theorem 3.33. *The structures $(\mathcal{G}, \leq_m)$, $(\mathcal{G}, \leq_s)$ and $(\mathcal{G}, \leq_i, P_3)$ are mutually interpretable with first order arithmetic.*

Chapter 4

Defining Arithmetical Predicates in Arbitrary Structures over Graphs[1]

This chapter is divided into three sections. In the first section, we give a proof of the fact that every recursively enumerable relation over numbers is definable in first order arithmetic. In the second section, we define the notion of a *capable* structure over graphs which are arithmetical structures which can interpret arithmetic and define three particular predicates over graphs. It is then shown that such capable structures have the maximal definability property (m.d.p.), that is, a relation is definable in such structures iff it is arithmetical. The third section is devoted to showing that the induced subgraph order is a capable structure and thus has the m.d.p. An important corollary is the definability of every recursively enumerable predicate over graphs in the induced subgraph order.

[1]The results in this chapter are from Thinniyam [40].

4.1 Defining Recursively Enumerable Predicates in Arithmetic

The following theorem is in fact a weakening of the MRDP theorem [34] which states that every recursively enumerable relation over numbers is existentially definable in arithmetic, but we give a proof here so that the abstract definability result in the next section (Theorem 4.8) has a self contained proof.

Theorem 4.1. *Every recursively enumerable relation over numbers is definable in first order arithmetic.*

To simplify the presentation, we will consider $R \subseteq \mathbb{N}$ which is an r.e. set of numbers. The essence of the proof remains the same for a relation of arity more than one. By assumption, there is a Turing Machine $M_R = (Q, \Sigma, \delta, s, f)$ with finite set of states Q, alphabet $\Sigma = \{0, 1\}$, transition function $\delta : Q \setminus \{f\} \times \{0, 1\} \to Q \times \{0, 1\} \times \{L, R\}$, start state s and final state f such that the language $L(M_R)$ accepted by the Turing Machine is R.

We need to construct a formula $\phi_R(x)$ in one free variable such that

$$n \in L(M_R) \iff (\mathbb{N}, +, \times) \models \phi_R(n)$$

The accepting run of M_R on x is a sequence of configurations $c_0, c_1, ..., c_{n_2}$ of the machine M_R where c_0 is the initial configuration and the final configuration c_{n_2} is in the accepting state f. This accepting run can be encoded as the binary representation of a number y, which is used to construct the required formula ϕ_R.

Let $Q = \{q_1, q_2, ..., q_k\}$ where $q_1 = s$ and $q_2 = f$. Let $\Sigma' = \{a_1, a_2, ..., a_{k+5}\}$ where $a_1 = 0, a_2 = 1, a_3 = \#, a_4 = 0', a_5 = 1'$ and $a_{i+5} = q_i$ for $1 \leq i \leq k$. Let $n_0 = \lfloor log(\Sigma') \rfloor + 1$. The symbol $\#$ is used to separate consecutive configurations, the alphabets $0'$ and $1'$ are intended to represent the fact that the letter being read by the head of the Turing Machine is 0 and 1 respectively, and the rest of Σ' is used to represent the states of M_R. The number n_0 is the number of bits required to represent one letter of Σ' and is independent of the input x. Suppose the machine M uses $n_1 - 2$ tape cells and runs for time n_2 before accepting the input x, then the binary representation of y (disregarding the most significant bit) can be thought of as $c_0\#c_1\#...\#c_{n_2}\#$ where a configuration c_i is a string $qb_1b_2....b_{n_1-2}$ with q the state and $b_1...b_{n_1-2}$ the tape contents. The b_i are either 0 or 1 except for a unique b_j which is either $0'$ or $1'$ indicating the current position of the head of the machine.

$$\phi_R(x) := \exists y\ \exists n_0\ \exists n_1\ \exists n_2\ |y| = 1 + n_0n_1n_2\ \wedge\ init(x, y, n_0, n_1, n_2) \wedge$$
$$checkConfig(y, n_0, n_1, n_2)\ \wedge\ validMoves(y, n_0, n_1, n_2)\ \wedge\ accept(y, n_0, n_1, n_2)$$

The formula $checkConfig$ makes sure that the binary representation of y is of the form $c_0\#c_1\#...\#c_{n_2}\#$, $init$ makes sure the initial configuration is correct, $accept$ makes sure that M_R is in the state f at the end of the accepting run and $validMoves$ makes sure that consecutive configurations c_i and c_{i+1} are related by the transition function δ.

To define the above formulae, we need some intermediate predicates:

$$a_i(j, z, n_0) := window(n_0, j, z) = i$$

$$config(i, y, n_0, n_1) := 2^{n_0n_1} + window(n_0n_1, i, y)$$

$$head(i_0, i, x, n_0, n_1) := a_4(i_0, config(i, y, n_0, n_1), n_0) \vee a_5(i_0, config(i, y, n_0, n_1), n_0)$$

In the above, the formula $a_i(j, z, n_0)$ states that z when seen as a word over Σ' has letter a_i in position j when read left to right. We use the predicate $window$ defined in Lemma 3.23. The second formula has been written as a term where the term $config(i, y, n_0, n_1)$ is intended to mean a number of total bit length $1 + n_0n_1$ which when seen as a word over $\{0, 1\}$ (disregarding the most significant bit) is the i^{th} subword of y of window length

$n_0 n_1$ when y is read left to right. The formula $head$ is intended to capture the position of the head of the Turing Machine and says that the head is at position i_0 along the word $config(i, y, n_0, n_1)$ read from left to right. We can now write down the required formulae:

$$\begin{aligned}
init(x, y, n_0, n_1, n_2) := a_6(1, y, n_0)\ \wedge \forall i\ (2 \leq i \leq |x|)\ \supset \\
&[bit(|x| - i + 1, x) \supset a_2(i + 1, y, n_0)\ \wedge\ (\neg bit(|x| - i + 1, x) \supset a_1(i + 1, y, n_0)) \\
&\wedge\ (bit(|x|, x) \supset a_5(2, y, n_0))\ \wedge\ (\neg bit(|x|, x) \supset a_4(2, y, n_0)) \\
&\wedge\ (\forall i\ (|x| + 1 \leq i \leq n_1) \supset\ a_1(i, y, n_0))]
\end{aligned}$$

The above formula checks that the first letter in y corresponds to the initial state, the second letter is either $0'$ or $1'$ depending on whether the most significant bit of the input x is 0 or 1 respectively, and the letters from positions 2 to $|x|$ are 0 or 1 depending on whether the corresponding bit of x is 0 or 1 and the letters in positions $|x| + 1$ to n_1 are all 0.

$$\begin{aligned}
checkConfig(y, n_0, n_1, n_2) := \forall i\ (1 \leq i \leq n_2) \supset\ & state(y, i, n_0, n_1) \wedge \\
& hashMarker(y, i, n_0, n_1)\ \wedge\ \exists i_0\ head(i_0, i, y, n_0, n_1) \\
& \wedge\ \forall i_1 (2 \leq i_1 \leq n_1\ \wedge\ i_1 \neq i_0) \supset \\
& \qquad tapeContents(i_1, i, y, n_0, n_1)
\end{aligned}$$

where

$$\begin{aligned}
state(y, i, n_0, n_1) &:= \bigvee_{k+5 \geq j \geq 6} a_j(1, config(i, y, n_0, n_1), n_0) \\
hashMarker(y, i, n_0, n_1) &:= a_3(n_1, config(i, y, n_0, n_1), n_0) \\
tapeContents(i_1, i, y, n_0, n_1) &:= a_1(i_1, config(i, y, n_0, n_1), n_0) \vee \\
& \quad a_2(i_1, config(i, y, n_0, n_1), n_0)
\end{aligned}$$

The formula $checkConfig$ checks that the first letter in every window is a state, the last letter in every window is a #, there is a unique position i_0 in every window which indicates the head position and the rest of the letters in a window are either 0 or 1.

$$\begin{aligned}
\mathit{validMoves}(y, n_0, n_1, n_2) :=& \forall i\ (1 \leq i \leq n_2 - 1) \supset [\\
& \bigvee_{(q_j,b)\to(q_{j'},b',d)\in\delta} \exists i_0\ \mathit{moveHead}(i_0, i, y, n_0, n_1)\\
& \wedge\ \mathit{stateChange}(i_0, i, y, n_0, n_1)\\
& \wedge\ \mathit{letterUnderOldHead}(i_0, i, y, n_0, n_1)\\
& \wedge\ \mathit{letterUnderNewHead}(i_0, i, y, n_0, n_1)\\
& \wedge\ \mathit{otherLetters}(i_0, i, y, n_0, n_1)]\\
& \text{where}\\
\mathit{moveHead}(i_0, i, y, n_0, n_1) :=& \mathit{head}(i_0, i, y, n_0, n_1) \wedge\\
& \mathit{head}(i_0 + d, i + 1, y, n_0, n_1)\\
\mathit{stateChange}(i_0, i, y, n_0, n_1) :=& a_{j+5}(1, \mathit{config}(i, y, n_1))\\
& \wedge\ a_{j'+5}(1, \mathit{config}(i, y, n_0, n_1), n_0)\\
\mathit{letterUnderOldHead}(i_0, i, y, n_0, n_1) :=& a_{b+4}(i_0, \mathit{config}(i, y, n_0, n_1), n_0)\\
& \wedge\ a_{b'+1}(i_0, \mathit{config}(i, y, n_1))\\
\mathit{letterUnderNewHead}(i_0, i, y, n_0, n_1) :=& a_{b+1}(i_0 + d, \mathit{config}(i, y, n_0, n_1), n_0) \wedge\\
& a_{b'+4}(i_0 + d, \mathit{config}(i, y, n_1))\\
\mathit{otherLetters}(i_0, i, y, n_0, n_1) :=& \forall i\ (1 \leq i' \leq n_1\ \wedge\ i' \neq i_0 \wedge i' \neq i_0 + d) \supset\\
a_1(i', \mathit{config}(i, y, n_0, n_1), n_0) \iff& a_1(i', \mathit{config}(i + 1, y, n_0, n_1), n_0)
\end{aligned}$$

By a transition $(q_j, b) \to (q_{j'}, b', d)$ we mean that the machine M_R from a state q_j on reading a bit b, changes its state to $q_{j'}$ after replacing b by the bit b' and moves its head in the direction d, where $d = 1$ if the machine head moves right and $d = -1$ if it moves left. The formula $\mathit{validMoves}$ checks that for every pair of consecutive configurations, there exists a transition $(q_j, b) \to (q_{j'}, b', d)$ such that movement of the head, the state change, the letter under the head before and after the move are in accordance with the transition and the remaining letters remain the same.

$$accept(y, n_0, n_1, n_2) := a_7(1, config(n_2, y, n_0, n_1).n_0)$$

The formula $accept$ checks if the final configuration is in the accepting state f. This concludes the proof of Theorem 4.1.

4.2 Capable Structures

In this section, we introduce the notion of a capable structure over graphs which is an arithmetical structure with the ability to define arithmetic and certain predicates related to o-presentations. We then show that any capable structure can define every arithmetical predicate over graphs. We note that the graph orders considered in this thesis are arithmetical; the abstract definability results of this chapter will be used to give a characterization of the predicates definable in graph orders in the next chapter.

Lemma 4.2. *The structures ($\mathcal{G}, \leq_i, P_3$), ($\mathcal{G}, \leq_s$) and ($\mathcal{G}, \leq_m, sameSize$) are arithmetical.*

Proof. The fact that $\leq_i, \leq_s, \leq_m, P_3, sameSize$ are definable in ($\mathcal{G}, plus_t, times_t$) follows from the fact that they are recursively enumerable predicates and using
Theorem 4.1. □

In order to introduce the notion of a capable structure, we need the notion of an *o-presentation*, first introduced by Ježek and McKenzie [23] in the setting of posets and defined by Wires [43] for graphs. An o-presentation is the representation of an ordered version of a graph g by another graph g'. The definability of predicates related to such o-presentations allows us to access the internal structure of a graph i.e. the edge relation, which is not available in the vocabulary.

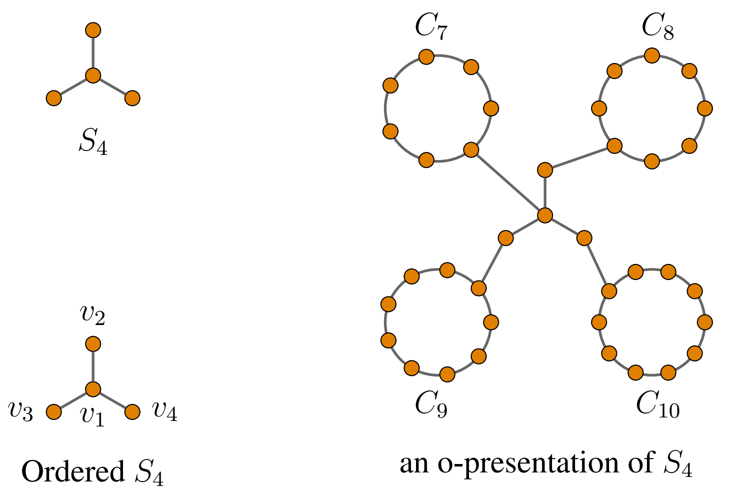


Figure 4.1: Top left: the graph S_4. Bottom left: A vertex ordering of S_4. Right: The o-presentation corresponding to the given vertex ordering.

Definition 4.3 (o-presentation)**.** *An o-presentation of $g \in \mathcal{G}$ is another graph g' constructed as follows: fix an ordering $v_1 < v_2 < ... < v_n$ of vertices of g. Let g'' be the graph formed by the disjoint union of g and the cycles C_{n+i+2} for each $1 \leq i \leq n$. Add n additional edges to g'' connecting each cycle C_{n+i+2} to the corresponding vertex v_i. The resulting graph is g'.*

Remark 4.4. *Note that each vertex ordering of a graph g leads to a (possibly) different o-presentation. We will refer to the set of o-presentations of g by $\tilde{g}$ and write $g' \in \tilde{g}$ to indicate that g' is an o-presentation of g. The example in Figure 4.1 clarifies the bijective correspondence between o-presentations and labellings of a graph.*

Definition 4.5 (Indicator Cycle)**.** *Given a graph g, a cycle C is called an indicator cycle of g if $|g| + 3 \leq |C| \leq 2|g| + 2$.*

We will often just call C an indicator cycle if the graph g is understood from the context.

Definition 4.6 (Capable Structure over Graphs)**.** *We call an arithmetical structure $(\mathcal{G}, \tau)$ a capable structure over graphs if it satisfies the following three conditions:*

($C1$) Arithmetic can be defined in ($\mathcal{G}, \tau$), in particular, the following predicates are definable:

1. *The family $\mathcal{N}$ of graphs which do not contain edges i.e. are made of isolated points.*
2. *The predicate $\psi_{+}(z, x, y)$ holds if and only if $x, y, z \in \mathcal{N}$ and $|x| + |y| = |z|$.*
3. *The predicate $\psi_{\times}(z, x, y)$ holds if and only if $x, y, z \in \mathcal{N}$ and $|x| \times |y| = |z|$.*

($C2$) The following predicates related to o-presentations are definable in ($\mathcal{G}, \tau$):

1. *$\psi_{opres}(x, y)$ holds if and only if x is an o-presentation of y, also written $x \in \tilde{y}$.*
2. *$\psi_{edgeOP}(x, i, j)$ holds if and only if $i, j \in \mathcal{N}$ and there exists a graph y such that $x \in \tilde{y}$ and in the vertex ordering induced on y by the o-presentation x, the vertices $v_{|i|}$ and $v_{|j|}$ in y have an edge.*

($C3$) The predicate $sameCard(x, y)$ holds if and only if x and y have the same number of vertices.

Observation 4.7. *The binary relation $card(x, y)$ which holds if and only if $y = |x|_g$ is definable :*

$$card(x, y) := \mathcal{N}(y) \wedge sameCard(x, y)$$

Recall that an arithmetical structure over graphs is said to have the maximal definability property if it can define every arithmetical predicate over graphs.

Theorem 4.8. *Any ($\mathcal{G}, \tau$) which is a capable structure over graphs has the maximal definability property.*

Let n_x be the number assigned to a graph x in accordance with the order $\leq_t$. It is sufficient to show that the following predicates are definable in any capable structure:

1. $plus_t(z, x, y)$ if and only if $n_x + n_y = n_z$.
2. $times_t(z, x, y)$ if and only if $n_x \times n_y = n_z$.

We will instead show that any recursive relation $R \subseteq \mathbb{N}^k$ over graphs is definable via a

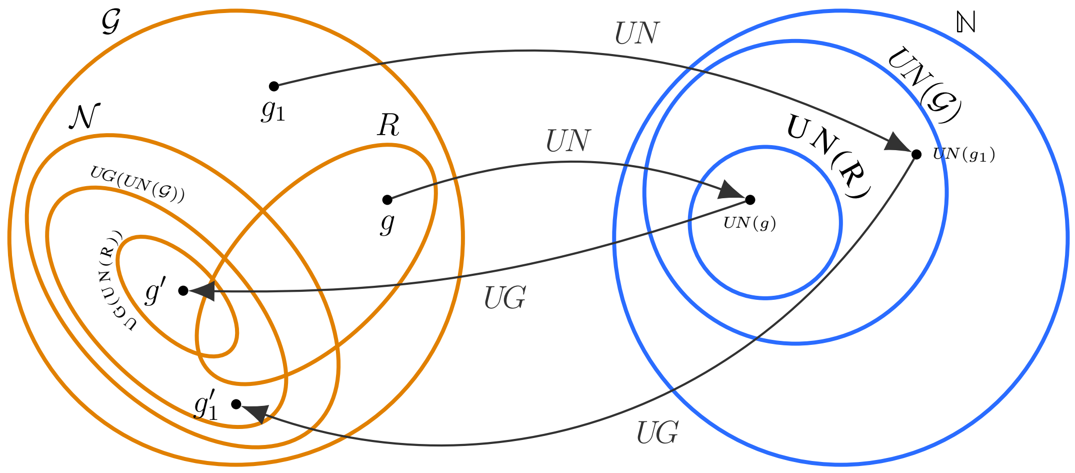


Figure 4.2: Maps UN and UG and how they act on $R \subseteq \mathcal{G}$. For any graph $g \in R$ it is the case that $UG(UN(g)) = g' \in UG(UN(R))$. For any graph $g_1 \notin R$ correspondingly $UG(UN(g_1)) = g'_1 \in (UG(UN(\mathcal{G})) \setminus UG(UN(R)))$.

formula ψ_R in a capable structure. This implies the definability of $plus_t$ and $times_t$ since they are both recursive predicates. The proof is carried out in two steps which we outline below:

Step 1 :

Corresponding to every graph g there exists the graph $g' = UG(UN(g)) \in \mathcal{N}$. From the definitions of the maps UN and UG it is clear that the map $UG \circ UN$ is a bijection between the sets $\mathcal{G}$ and $UG(UN(\mathcal{G}))$. The fact that $UG(UN(\mathcal{G})) \subseteq \mathcal{N}$ allows us to use the definability of arithmetic to capture the image $UG(UN(R))$ of the relation R using a formula $\psi^{trans}_{UG(UN(R))}$ over τ which is the translation of an arithmetic formula $\phi_{UN(R)}$.

Step 2 :

What remains to be done is to show that the predicate $\psi_{enc}(x, y)$ which holds if and only if $y = UG(UN(x))$ is definable in $(\mathcal{G}, \tau)$. Once this is done, we immediately get the defining formula ψ_R by moving over from the input $k-$tuple $\bar{x}$ to the $k-$tuple $\bar{y}$ and verifying that $\bar{y}$ belongs to $UG(UN(R))$

We now give details of the first step whose aim is to define $\psi^{trans}_{UG(UN(R))}$.

Since R is a recursive predicate, by Definition 2.23 there exists a machine M which accepts the UN encodings of the set of $k-$tuples of graphs which belong to R.

$$R(\bar{g}) \iff UN(\bar{g}) \in L(M)$$

By Theorem 4.1, there is an arithmetic formula $\phi_{UN(R)}(\bar{x})$ such that for any $k-$tuple $\bar{n}$ of numbers,

$$(\mathbb{N}, +, \times) \models \phi_{UN(R)}(\bar{n}) \iff \bar{n} \in UN(R)$$

The condition ($C1$) that arithmetic is definable in any capable structure gives the following corollary:

Corollary 4.9. *For every formula $\phi(\bar{x})$ in arithmetic there is a formula $\psi^{trans}_{UG}(\bar{x})$ in $(\mathcal{G}, \tau)$ such that for any $k-$tuple $\bar{n}$ of numbers,*

$$(\mathbb{N}, +, \times) \models \phi(\bar{n}) \iff (\mathcal{G}, \tau) \models \psi^{trans}_{UG}(UG(\bar{n}))$$

Applying this translation to $\phi_{UN(R)}$ gives us the graph formula $\psi^{t}_{UG(UN(R))}$. This completes the first step of the proof.

For the second step, we show how to define ψ_{enc}. To do so, we need arithmetical predicates to identify the image set $UN(\mathcal{G}) \subseteq \mathbb{N}$, identify the cardinality of g and draw out the edge information $E(g)$ from the unique number representation $UN(g)$.

The following lemma is a corollary of Theorem 4.1.

Lemma 4.10. *The following predicates are definable in arithmetic:*

1. *$\phi_{UN}(x)$ holds if and only if x is a number which uniquely represents a graph (see Definition 2.21).*
2. *$\phi_{edge}(x, i, j)$ holds if and only if there is a graph g such that $x = UN(g)$ and $v_i v_j$ is an edge in the order induced by the map UN.*
3. *$\phi_{length}(x, n)$ holds if and only if the length of the binary representation of x is n. We*

will just write $length(x)$ *to denote* n.

We can now define the binary relation $y = UG(UN(x))$ by the formula $\psi_{enc}(x, y)$:

$$\psi_{enc}(x,y) := \mathcal{N}(y) \ \wedge \ \psi^{trans}_{graphOrder}(y, |x|_g) \ \wedge \ \psi^{trans}_{UN}(y) \ \wedge$$
$$\exists z \ [z \in \tilde{x} \ \wedge \ \forall \ N_1 \leq i < j \leq |x|_g$$
$$\psi^{trans}_{edge}(y, i, j) \iff \psi_{edgeOP}(z, i, j)]$$

We explain the various subformulae used in the formula above:

1. ψ^{trans}_{UN} and ψ^{trans}_{edge} are translations of arithmetical formulae by use of Theorem 4.10.
2. $z \in \tilde{x}$ which holds if and only if z is an o-presentation of x and $\psi_{edgeOP}(z, i, j)$ are from condition $C2$.
3. The cardinality $|x|$ (represented by $|x|_g$ which is the corresponding member of $\mathcal{N}$) of a graph x uses condition $C3$.
4. $\psi^{trans}_{graphOrder}$ is the translation of the following arithmetic formula:

$$\phi_{graphOrder}(n, m) := length(n) = 1 + m(m-1)/2$$

Given a graph x, $UN(x) = n_x$ is a number which has bit length $1 + |x|(|x| - 1)/2$. Applying UG to n_x should give us y. Thus the number $n = UG^{-1}(y)$ should have bit length $1+|x|(|x|-1)/2$. This condition is taken care of by the formula $\psi^{trans}_{graphOrder}(y, |x|)$. Effectively, what this amounts to is identifying the cardinality of x. The formula $\psi^{trans}_{UN}(y)$ verifies that n belongs to the set $UN(\mathcal{G})$. Finally, we need to check that the edge information is correct to conclude that $n_x = n$. This is done using the fact that there is a witnessing o-presentation z of x such that the edge information in z matches with the edge information in n (which is accessed via y in the formula). This concludes the definability of $\psi_{enc}(x, y)$.

We can now write the required formula ψ_R in $(\mathcal{G}, \tau)$.

$$\psi_R(\bar{x}) := \exists \bar{y} \bigwedge_{i=1}^{k} \psi_{enc}(x_i, y_i) \ \wedge\ \psi^{trans}_{UG(UN(R))}(\bar{y})$$

This concludes the proof of Theorem 4.8. In the next section, we show that the induced subgraph order is a capable structure.

4.3 The Induced Subgraph Order is Capable

The fact that the induced subgraph order is a capable structure is already implicit in the work of Wires [43] and follows from the result therein that a relation over graphs is definable in the induced subgraph order if and only if it is definable in the associated small category of graphs. In fact, most of the formulae required are already explicitly present in Wires; for the few which are not, we give explicit formulae. In particular condition $C1$ is by Lemma 3.14 (2) and $C3$ is by Lemmas 3.14 (1), 3.17 and 3.21. It remains to give explicit formulae for condition $C2$.

In other words, we need to show the definability of the predicate $x \in \tilde{y}$ which stands for 'x is an o-presentation of y' and the predicate $\psi_{edgeOP}(x, i, j)$ which holds if and only if x is an o-presentation and there is an edge between vertices v_i and v_j in the corresponding ordered graph. The formulae required to do this make extensive use of number-theoretic predicates and we will freely use formulae such as $|x|^2 + |y| = |z| + 3$ with the understanding that appropriate formulae can be written in the original vocabulary; in particular we drop the subscript $|x|_g$ and use $|x|$ to denote the graph $N_{|x|}$.

We will first define the set $\tilde{\mathcal{G}}$ of all o-presentations and then use it to define $x \in \tilde{y}$. We recall some definability results from Wires [43].

Lemma 4.11 (Wires [43]). *The following predicates are definable in ($\mathcal{G}, \leq_i, P_3$).*

1. *The family $\mathcal{C}$ of cycles.*

2. *$maximalComp(x, y)$ if and only if y is a maximal connected component of x.*
3. *$cover(y, x, n)$ holds if and only if there are exactly $n - 1$ graphs between x and y in the order and $x \leq_i y$. Also denoted $x \lessdot_i^n y$.*
4. *$\mathcal{C}_{\to 1}(x)$ holds if and only if x is the connected graph formed by adding one extra vertex and one extra edge to a cycle.*
5. *$conn(x)$ holds if and only if x is a connected graph.*
6. *$disjointUnion(z, x, y)$ (also written $z = x \uplus y$) holds if and only if z is the disjoint union of x and y.*
7. *$\mathcal{C}_{\to 2}(x)$ holds if and only if x is formed from the graph g which satisfies $C_{\to 1}(g)$ by adding an additional vertex and joining it to the unique vertex in g which has degree 1.*
8. *$pointedCycleSum(z, x, y)$ holds if and only if x and y are incomparable cycles and z is formed by starting with the graph $x \cup y$ and adding one extra vertex v and two extra edges, one from v to any vertex of x and another from v to any vertex of y. We will write $z = x +_p y$ for short.*
9. *$CP_4C(x, i, j)$ holds if and only if $i, j \in \mathcal{N}, 3 < i < j$ and x is formed by adding to the graph $C_i \uplus C_j$ two additional vertices v_1, v_2 and the edge $v_1 v_2$, one edge between C_i and v_1 and one edge between C_j and v_2. We denote x by $CP_4C(i, j)$.*

Notice that from the definability of $\mathcal{C}_{\to 1}(x)$ we also have definability of the graph $C_{j\to 1}$ which stands for the member of $\mathcal{C}_{\to 1}$ of order $j + 1$ because the family is totally ordered by number of vertices and thus Observation 3.6 can be applied to $C_{j\to 1}$ to define each member of the family as a constant. Additionally, given a parameter n, we can obtain $C_{n\to 1}$. For the same reason, $C_{n\to 2}$ is also definable given n.

We need one additional predicate not explicitly defined by Wires.

Lemma 4.12. *The family $bicycle(x)$ which holds if and only if x is formed by adding an edge between two unequal cycles, is definable in the induced subgraph order.*

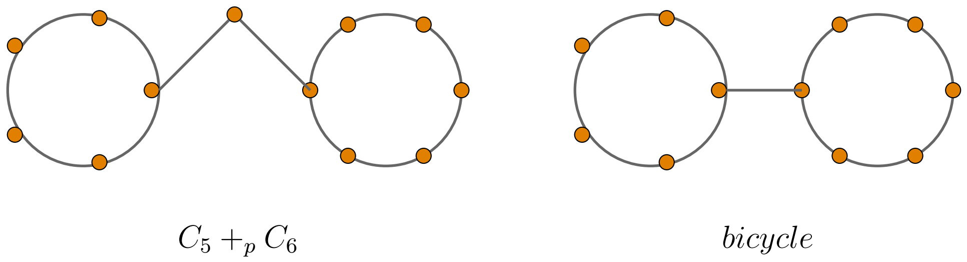


Figure 4.3: Left: a pointed cycle sum. Right: a bicycle.

Proof.

$$\begin{aligned} \mathit{bicycle}(x) :=& \mathit{conn}(x) \ \wedge \ \exists y \, \exists z \, \mathcal{C}(y) \ \wedge \ \mathcal{C}(z) \ \wedge \ y \neq z \\ & \wedge \ |x| = |y| + |z| \ \wedge \ y \leq_i x \ \wedge \ z \leq_i x \ \wedge \\ & \forall w \ (w \neq C_{|y| \to 2} \ \wedge \ C_{|y| \to 1} \lessdot_i w) \supset w \not\leq_i x \\ & \wedge \ \forall w \ (w \neq C_{|z| \to 2} \ \wedge \ C_{|z| \to 1} \lessdot_i w) \supset w \not\leq_i x \end{aligned}$$

Since the two cycles y and z are induced subgraphs and the cardinality constraint implies that there are no other vertices apart from the cycle vertices, x is restricted to graphs which are formed by adding edges between z and y. There is at least one edge due to the connectedness constraint. We avoid multiple edges by avoiding induced subgraphs which contain two edges incident on either cycle. □

We can now define the set of all o-presentations.

Lemma 4.13. *The predicate $\tilde{\mathcal{G}}(x)$ which holds if and only if x is an o-presentation is definable in the induced subgraph order.*

Proof.

$$\begin{aligned}
\tilde{\mathcal{G}}(x) :=& \exists n\ cardCond(x,n)\ \wedge\ hasC1s(x,n)\ \wedge \\
& hasUnionOfCycles(x,n)\ \wedge\ noMultiEdge(x,n) \\
& \wedge\ noPointedCycleSums(x,n)\ \wedge\ noBicycles(x,n) \\
& \text{where} \\
cardCond(x,n) :=& \mathcal{N}(n)\ \wedge\ |x| = n^2 + n(n+1)/2 + 3n \\
hasC1s(x,n) :=& \forall i\ (1 \leq i \leq n) \supset C_{i+n+2\to 1} \leq_i x \\
hasUnionOfCycles(x,n) :=& \dot{\bigcup_{i=1}^{n}} C_{n+i+2} \leq_i x \\
noMultiEdge(x,n) :=& \forall i\ (1 \leq i \leq n) \supset\ [\forall z\ (C_{n+i+2\to 2} \neq z\ \wedge \\
& C_{n+i+2\to 1} \lessdot_i z) \supset z \nleq_i x] \\
noPointedCycleSums(x,n) :=& \forall i\ \forall j\ (1 \leq i < j \leq n) \supset C_{n+i+2} +_p C_{n+j+2} \nleq_i x \\
noBicycles(x,n) :=& \forall y\ (bicycle(y)\ \wedge\ |y| > 2n) \supset y \nleq_i x
\end{aligned}$$

In order to show that a graph x is an o-presentation, we need to show that the vertex set V of x can be partitioned into two sets V_1 and V_2 such that:

1. Let $|V_2| = n$. The graph g induced on V_1 is $\dot{\bigcup}_{i=1}^{n} C_{n+i+2}$. We denote the vertex set of each large cycle C_{n+i+2} by V_i' and $\dot{\bigcup}_i V_i' = V_1$.
2. There is a bijection $f : \{V_1', V_2', ..., V_n'\} \to V_2$ such that there is an edge from a unique vertex of V_i' to $f(V_i')$ and there are no other edges between the large cycles in x and the vertex set V_2.

The formula $cardCond$ states that the graph has as many vertices as required to contain as induced subgraph a graph on n vertices and cycles of order $n+i+2$ for each i between 1 and n.

$hasUnionOfCycles$ states that the disjoint union of all the required cycles is an induced subgraph. Because of the cardinality constraint already imposed, this implies that there is a unique copy of each cycle in x. Let V_1 be the set of vertices which induce the graph $\dot\bigcup_{i=1}^{n} C_{n+i+2}$. The remaining vertices of x form the set V_2, which has cardinality n. No restriction is placed on the edges between the non-cycle vertices V_2. It remains to place appropriate restrictions on V_1 in order to make sure that the resulting graph x is of the required form. The formula $hasC1s$ states that the $C_{\rightarrow 1}$ are induced subgraphs, thus there is at least one edge from every indicator cycle to the rest of the graph. The formula $noMultiEdge$ ensures that there are no multiple edges between an indicator cycle and the rest of the graph while $noPointedCycleSums$ ensures that two different indicator cycles do not have an edge to a vertex v external to the two cycles. At this point, the constraints ensure that there is exactly one edge incident on each indicator cycle which has its other end elsewhere. But this does not rule out the possibility of two cycles directly connected by an edge. To rule this out, we have $noBicycles$. Note that by design all indicator cycles are of different length. Thus the edge enforced via $hasC1s$ must be between a indicator cycle and a vertex in V_2. Together, this implies the existence of the bijection f between indicator cycles and vertices in V_2. □

Towards defining ψ_{opres}, we define two more intermediate predicates.

Lemma 4.14. *The following predicates are definable in the induced subgraph order:*

1. $csum(x, n)$ *holds if and only if* $n \in \mathcal{N}$ *and* $x = \dot\bigcup_{i=1}^{n} C_{n+i+2}$.
2. $psum(x, n)$ *holds if and only if* $x = \dot\bigcup_{i=1}^{n} P_{n+i+1}$.

Proof. We can construct the object $\dot\bigcup_{i=1}^{n} C_{n+i+2}$ as follows:

$$\begin{aligned} csum(x,n) :=& \forall z\ maximalComp(x,z) \supset \mathcal{C}(z) \wedge \\ & cardCond(x,n) \wedge allCycles(x,n) \\ & \text{where} \\ cardCond(x,n) :=& \mathcal{N}(n) \wedge |x| = n^2 + n(n+1)/2 + 3n \\ allCycles(x,n) :=& \forall m\ (n+3 \le m \le 2n+2) \supset C_m \le_i x \end{aligned}$$

If $x = \dot\bigcup_{i=1}^{n} C_{n+i+2}$ then it clearly satisfies the formula $csum(x,n)$. Suppose x satisfies $csum(x,n)$. Therefore it contains every cycle C_m for $n+3 \le m \le 2n+2$ as induced subgraph. Suppose a copy of C_m and $C_{m'}$ (where $m \neq m'$ and $C_m, C_{m'}$ are both indicator cycles) present in x share a common vertex v. Consider the subgraph g formed by the vertices of these copies: it is connected. Hence the component of x containing g is not a cycle, which contradicts $\forall z\ maximalComp(x,z) \supset \mathcal{C}(z)$. Hence any copy of C_m is disjoint from a copy of $C_{m'}$ present inside x. But the cardinality condition imposed by $cardCond$ implies that there is a unique copy of each C_m. In fact, there are no other vertices in x apart from vertices belonging to the cycles C_m. There cannot be any edges between different cycles, again because of the $maximalComp$ condition. Hence x is exactly the graph $\dot\bigcup_{i=1}^{n} C_{n+i+2}$.

Next we show how to construct the graph $\dot\bigcup_{i=1}^{n} P_{n+i+1}$, which is formed by deleting one vertex from each cycle in the graph $\dot\bigcup_{i=1}^{n} C_{n+i+2}$.

$$psum(x,n) := \exists y\ y = \dot\bigcup_{i=1}^{n} C_{n+i+2} \wedge x <_i^n y \wedge \forall z\, \mathcal{C}(z) \supset z \not\le_i x$$

i.e. we get the appropriate graph by enforcing the condition that no cycle is an induced subgraph. □

We can now define $\psi_{opres}(y,x)$ which holds if and only if y is an o-presentation of x.

$$\begin{aligned} \psi_{opres}(y,x) :=& \tilde{\mathcal{G}}(y) \ \wedge \ constructFromPaths(y,x) \\ & \wedge \ cardCond(y,x) \\ & where \\ cardCond(y,x) :=& |y| = |x|^2 + |x|(|x|+1)/2 + 3|x| \\ constructFromPaths(y,x) :=& \exists z \ z = x \uplus \biguplus_{i=1}^{|x|} P_{|x|+1+i} \ \wedge \ z \lessdot_i^{|x|} y \end{aligned}$$

The formula ψ_{opres} states that y is an o-presentation of appropriate order and deletion of $|x|$ vertices from y gives the disjoint union of x with paths of size $|x|+2$ to $2|x|+1$. Let y be an o-presentation obtained by by addition of $|x|$ new vertices $V_{new} = \{v_1, v_2, ..., v_{|x|}\}$ to z. Let $V(z) = V(x) \cup V_P$ where V_P are the vertices of $\biguplus_{i=1}^{|x|} P_{|x|+1+i}$. The cardinality of y is $|x|^2 + |x|(|x|+1) + 3|x|$ and hence it must contain each cycle in the set $\{C_{|x|+i+2} \mid 1 \leq i \leq |x|\}$ of indicator cycles as induced subgraph. None of these cycles is present in z as induced subgraph since neither x nor any of the graphs $P_{|x|+1+i}$ contain them. Hence every such cycle has to be created in y through addition of new edges between the newly added vertices V_{new} and $V(z)$. Since the creation of each indicator cycle requires at least one new vertex and the number of new vertices is equal to the number of indicator cycles, the only way to get $\biguplus_{i=1}^{|x|} C_{|x|+i+2}$ as an induced subgraph of y is to add two edges connecting the ends of the path $P_{|x|+i+1}$ to v_i, for every i. This gives us the graph $y' = x \uplus \biguplus_{i=1}^{|x|} C_{|x|+2+i}$ where $V(y') = V(z) \cup V_{new}$. We need to add some more edges between V_{new} and the vertices $V(z)$ in y' to get an o-presentation. But by the properties of o-presentations, there is exactly one more edge between each vertex in V_{new} and the vertices $V(x)$ of z. Thus the graph y must be an o-presentation of x.

We now take up the definition of the predicate $\psi_{edgeOP}(x,i,j)$.

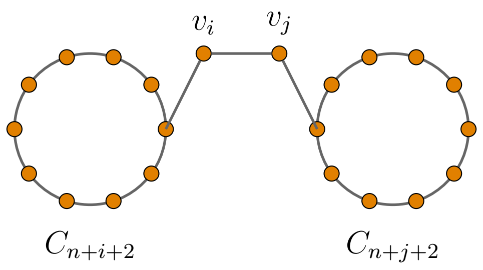


Figure 4.4: The CP_4C graph corresponding to an edge between vertices v_i and v_j.

$$\psi_{edgeOP}(x,i,j) := \exists y\ x \in \tilde{y}\ \wedge\ \exists m\ (|x|_g = m^2 + m(m+1)/2 + 3m) \wedge$$
$$CP_4C(m+i+2, m+j+2) \leq_i x$$

The existence of an edge between vertices v_i and v_j in the graph x is captured by the presence of a CP_4C induced subgraph in y (which is an o-presentation of x) with appropriate parameters and this is stated by the formula ψ_{edgeOP}.

This concludes the proof that $(\mathcal{G}, \leq_i, P_3)$ satisfies condition $C2$.

Lemma 4.15 (Wires [43])**.** *The structure $(\mathcal{G}, \leq_i, P_3)$ is capable.*

By Theorem 4.8 and the above result, we get the following result.

Theorem 4.16. *The structure $(\mathcal{G}, \leq_i, P_3)$ has the maximal definability property.*

Chapter 5

Defining Arithmetical Predicates in the Subgraph Order[1]

The main result of this chapter is that the subgraph order is a capable structure. The proof of the induced subgraph order being capable in the previous chapter was presented bottom-up since many of the intermediate predicates required were already available in literature. Since we need to define all the predicates for the subgraph order and the details are tricky, we present a top-down approach for the analogous result. Examining the defining formulae for the relations ψ_{opres} and ψ_{edgeOP} in the induced subgraph order, we see that the result follows from the definability of three intermediate predicates: $\tilde{\mathcal{G}}$, CP_4C and $constructFromPaths$. The difference in the case of the subgraph is that instead of $constructFromPaths$ we use a relation $constructFromCycles$, along with $\tilde{\mathcal{G}}$ and CP_4C. The $constructFromCycles$ relation is used to capture the construction of o-presentations in the subgraph order. An o-presentation g'' of a graph g is constructed from the graph $g' = g \mathbin{\dot\cup} \dot\bigcup_{i=1}^{|g|} C_{|g|+i+2}$ which is the disjoint union of g with its indicator cycles, by addition of $|g|$ many edges to g'. This difference arises from the fact that the covering relation in the subgraph order is equivalent to deletion of a single edge or isolated

[1]The results in this chapter are from Thinniyam [40].

vertex, as opposed to deletion of a vertex along with all its incident edges in the induced subgraph order.

The definability of $constructFromCycles$ is more involved than that of $\tilde{\mathcal{G}}$ and CP_4C and is facilitated by the definition of two natural graph theoretic predicates of independent interest : a ternary relation $disjointUnion(x, y, z)$ if and only if $x \uplus y = z$, and a binary predicate $sameSize(x, y)$ if and only if x and y have the same number of edges. Thus the proof can be split up into two steps: first we show that the structure $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ is capable; second, we show that $disjointUnion$ and $sameSize$ are definable in $(\mathcal{G}, \leq_s)$, completing the proof that the subgraph order is capable. The two steps form the first two sections of this chapter. As a corollary we obtain the fact that the structure $(\mathcal{G}, \leq_m, sameSize)$ is also capable in the third section.

5.1 $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ is Capable

We first note the importance of the $sameSize$ predicate. Recall that the number of edges of x is denoted by $||x||$. Representing the number of edges of a graph x as a member of $\mathcal{N}$ allows us to perform arithmetic operations on $||x||$ and by an abuse of notation we will write $||x||$ in graph formulae to denote the corresponding member of $\mathcal{N}$. This enables us to write formulae such as $||x|| + |y| = ||z||$ which will be critically used in this section.

As in the case of the induced subgraph order, condition $C1$ follows from Lemmas 3.7, 3.18 and 3.21 and condition $C3$ from Lemma 3.7 for the subgraph order. We show that condition $C2$ holds of $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$. Continuing our top-down approach, we first show that $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ is capable assuming the definability of the relations CP_4C, $\tilde{\mathcal{G}}$ and $constructFromCycles$. This is followed by three subsections in which we show the definability of these three relations.

Observation 5.1. *The following predicates are known to be definable in the subgraph order by the results in Section 3.1:*

1. *$x \lessdot_s y, x \lessdot_{sv} y, x \lessdot_{se} y$ if and only if y is an upper cover, an upper cover formed by adding a single vertex, an upper cover formed by adding a single edge (respectively) to x.*
2. *The family $soc(x)$ of graphs where x is made of disjoint unions of cycles.*
3. *The families $\mathcal{N}, \mathcal{T}, \mathcal{K}, \mathcal{P}, \mathcal{C}, \mathcal{S}$ (isolated points, trees, cliques, paths, cycles, stars respectively).*
4. *$conn(x)$ if and only if x is a connected graph.*

Explicit formulae for $\lessdot_{sv}$ and $\lessdot_{se}$ were not given previously, we give them here:

$$x \lessdot_{sv} y := x \lessdot_s y \;\wedge\; |x| \neq |y|$$

$$x \lessdot_{se} y := x \lessdot_s y \;\wedge\; |x| = |y|$$

It is clear that for any fixed n, $x \lessdot_{sv}^{n} y$ if and only if y is formed by addition of n isolated vertices to y and similarly $x \lessdot_{se}^{n} y$ can be defined in an analogous manner.

Lemma 5.2. *The predicates ψ_{opres} and ψ_{edgeOP} are definable in $(\mathcal{G}, \leq_s,\ disjointUnion, sameSize)$ assuming the definability of the following predicates:*

1. *$CP_4C(x, i, j)$ if and only if $i, j \in \mathcal{N}, 3 < i < j$ and x is constructed from the graph $C_{i \to 1} \uplus C_{j \to 1}$ by adding one additional edge between the unique degree one vertices of $C_{i\to 1}$ and $C_{j \to 1}$.*
2. *$\tilde{\mathcal{G}}(x)$ if and only if x is an o-presentation of some graph.*
3. *$constructFromCycles(y, x)$ if and only if y is constructed by adding $|x|$ edges to the graph g which is the disjoint union of x and all indicator cycles corresponding to x.*

Proof.

$$\psi_{opres}(x, y) := \tilde{\mathcal{G}}(y) \;\wedge\; constructFromCycles(y, x)$$

The proof of correctness of the formula is similar to that in the induced subgraph case. Let

g be the graph $x \uplus \biguplus_i^{|x|} C_{|x|+i+2}$. The only way to get an o-presentation from the graph g by adding $|x|$ edges is to connect each indicator cycle to a vertex of x. The resulting graph has to be an o-presentation of x.

$$\psi_{edgeOP}(x,i,j) := \exists y \; x = \tilde{y} \;\wedge\; \exists m \; (|x| = m^2 + m(m+1)/2 + 3m) \;\wedge$$
$$CP_4C(m+i+2, m+j+2) \leq_s x$$

The formula ψ_{edgeOP} we use in the subgraph order is just the formula used in the induced order with $\leq_i$ replaced by $\leq_s$. The correctness of the formula follows from the fact that from the construction of o-presentations, it is clear that an indicator cycle (and the CP_4C graphs) is a subgraph of an o-presentation iff it is an induced subgraph of that o-presentation. $\square$

We now take up the definability of the relations CP_4C, $\tilde{\mathcal{G}}$ and $constructFromCycles$ in that order.

5.1.1 Defining CP_4C

In this subsection we define the relation $CP_4C(x,i,j)$. We will need the following intermediate predicates:

Lemma 5.3. *The following predicates are definable in the subgraph order:*

1. $maximalComp(y,x)$ *if and only if* x *is a maximal component of* y *under the subgraph order.*
2. $addVert(x,y)$ *if and only if* y *is a connected graph and* x *is a connected graph formed by adding one additional vertex and one additional edge to* y.
3. $\mathcal{C}_{\rightarrow 1}(x)$ *if and only if* x *is the connected graph formed by adding one additional vertex and one additional edge to a cycle.*

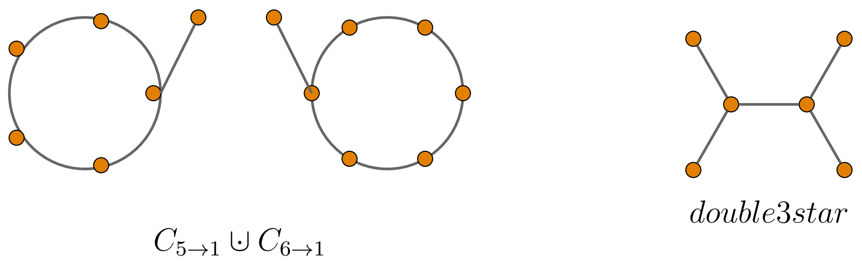


Figure 5.1: Left: a graph from the family $twoC1s$. Right: the graph $double3star$.

4. *$\mathcal{C}_{\to 2}(x)$ if and only if x is a graph which is formed by a taking a graph g from the family $\mathcal{C}_{\to 1}$ and adding an additional vertex and connecting it to the unique degree one vertex in g.*
5. *$twoC1s(x, i, j)$ if and only if $3 < i < j$ and x is a graph formed from the graph $C_i \uplus C_j$ by addition of two new vertices v_1, v_2 and two new edges e_1, e_2 where e_1 joins v_1 to C_i and e_2 joins v_2 to C_j. For fixed values of i, j there is a unique graph x satisfying the formula and we will denote it by $C_{i\to 1} \uplus C_{j\to 1}$ (see Figure 5.1).*

Proof. The proof of correctness of the defining formulae for the first three predicates is straightforward and essentially follows from their definitions.

$$maximalComp(y, x) := conn(x) \ \wedge \ x \leq_s y \ \wedge \ \forall z \; conn(z) \ \wedge \ z \leq_s y \supset \neg(x <_s z)$$

$$addVert(x, y) := conn(x) \ \wedge \ conn(y) \ \wedge \ \exists z \; y \lessdot_{sv} z \ \wedge \ z \lessdot_{se} x$$

$$\mathcal{C}_{\to 1}(x) := \exists y \; \mathcal{C}(y) \ \wedge \ addVert(x, y)$$

We need to define a particular constant graph to define $\mathcal{C}_{\to 2}$. The graph $double3star$ (see Figure 5.1) has vertex set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and edge set $E = \{v_1v_2, v_3v_2, v_2v_4, v_4v_5, v_4v_6\}$. The definability of this constant is straighforward:

$$double3star(x) := \mathcal{T}(x) \ \wedge \ |x| = 6 \ \wedge \ P_4 \leq_s x \ \wedge \ P_5 \not\leq_s x \ \wedge \ S_5 \not\leq_s x.$$

$$\mathcal{C}_{\to 2}(x) := \exists y\, \mathcal{C}_{\to 1}(y) \;\wedge\; addVert(x,y) \;\wedge\; P_{|x|} \leq_s x \;\wedge\; double3star \not\leq_s x$$

Let g be a graph satisfying the formula $\mathcal{C}_{\to 2}$. It is a connected graph formed by adding one more vertex and one more edge to a graph $g' \in \mathcal{C}_{\to 1}$. The formula states that a path of the same order as g is a subgraph of g. This implies that we can delete one edge from g to get $P_{|g|}$. There are only two possible such graphs. One of them is the graph we require and the other is the graph g'' which is formed from g' as follows. Let v be the vertex of degree three in g' and u be a vertex of degree two which is adjacent to v. Add a new vertex v' and the edge uv' to get g''. By deleting the edge uv from g'' we can get $P_{|g|}$. But this graph contains $double3star$ as a subgraph, which is disallowed by the formula. The correctness of the formula follows.

$$\begin{aligned} 2C1s(x,i,j) :=& |x| = i + j + 2 \;\wedge\; C_{i\to 1} \leq_s x \;\wedge\; C_{j\to 1} \leq_s x \\ & \wedge\; \forall z\; maximalComp(x,z) \supset \mathcal{C}_{\to 1}(z) \end{aligned}$$

We now show the correctness of the above formula. Let x, i, j be a tuple satisfying the formula. Every maximal component of x is a graph from the family $\mathcal{C}_{\to 1}$. Since the members of the $\mathcal{C}_{\to 1}$ family are pairwise incomparable under the subgraph order, any $C_{k\to 1}$ which is a subgraph of x must be a component of x. Hence $C_{i\to 1}$ and $\mathcal{C}_{j\to 1}$ are components of x. But according to condition $|x| = i + j + 2$, there cannot be any other vertices in x except for those belonging to the copy of $C_{i\to 1}$ or to the copy of $C_{j\to 1}$. Hence x is exactly the graph $C_{i\to 1} \uplus C_{j\to 1}$. $\square$

Now we can define CP_4C:

Lemma 5.4. *The predicate $CP_4C(x,i,j)$ if and only if $i, j \in \mathcal{N}, 3 < i < j$ and x is constructed from the graph $C_{i\to 1} \cup C_{j\to 1}$ by adding one additional edge between the unique degree one vertices of $C_{i\to 1}$ and $C_{j\to 1}$ is definable in the subgraph order.*

Proof.

$$\begin{aligned} CP_4C(x,i,j) :=& conn(x) \ \wedge \ \mathcal{N}(i) \ \wedge \ \mathcal{N}(j) \ \wedge \ 3 < i < j \ \wedge \\ & C_{i\to 1} \uplus C_{j\to 1} \lessdot_{se} x \ \wedge \ \forall z \ z \leq_s x \ \wedge \\ & (addVert(z, C_{i\to 1}) \vee addVert(z, C_{j\to 1})) \supset \mathcal{C}_{\to 2}(z) \end{aligned}$$

Let (x, i, j) be a tuple satisfying the above formula. We start with $C_{i\to 1} \uplus C_{j\to 1}$ and add an edge to get a connected graph. Suppose we connect a vertex from $C_{i\to 1}$ of degree more than 1 to a vertex of $C_{j\to 1}$. The resulting graph contains a graph g as subgraph which satisfies $addVert(g, C_{i\to 1})$ but does not satisfy $\mathcal{C}_{\to 2}(g)$. This contradicts the last condition of the formula. Hence the edge has to be added between the unique degree one vertices of $C_{i\to 1}$ and $C_{j\to 1}$. □

5.1.2 Defining $\tilde{\mathcal{G}}$

We take up the definability of the set $\tilde{\mathcal{G}}$ of all o-presentations in ($\mathcal{G}, \leq_s, sameSize$) in this subsection.

Lemma 5.5. *The following predicates are definable in the subgraph order:*

1. $soc2(x, i, j)$ *if and only if* x *is made of the disjoint union of the cycles* C_i *and* C_j.
2. $bicycle(x, i, j)$ *if and only if* x *is the connected graph formed by adding an edge to the graph* $g = C_i \uplus C_j$.
3. $pointedCycleSum(x, i, j)$ *if and only if* x *is formed from the graph* $g = C_i \cup C_j$ *by addition of a vertex* v*, an edge connecting* v *to* C_i *and another edge connecting* v *to* C_j*. We will denote the unique* x *which is the pointed cycle sum of* C_i *and* C_j *by* $C_i +_p C_j$.
4. $csum(x, n)$ *if and only if* $x = \bigcup_{i=1}^{n} C_{n+i+2}$.

Proof.

$$\begin{aligned} \mathit{soc2}(x,i,j) :=& \mathcal{N}(i) \ \wedge \ \mathcal{N}(j) \ \wedge \ \mathit{soc}(x) \ \wedge \ |x| = i+j \\ & \wedge \ \forall y \ (\mathcal{C}(y) \ \wedge \ y \leq_s x) \supset (y = C_i \vee y = C_j) \ \wedge \\ & C_i \leq_s x \ \wedge \ C_j \leq_s x \end{aligned}$$

Let (x,i,j) be a tuple satisfying the formula. Then x is a soc on $i+j$ vertices. Suppose $i \neq j$, then C_i and C_j are incomparable. Then it is sufficient to state that both C_i and C_j are subgraphs of x, because $C_i \uplus C_j$ is also forced as a subgraph of x. Along with the cardinality constraint, this implies that x is exactly $C_i \uplus C_j$.

If $i = j$ and $i = n_1 \times n_2$, then the graph $C_i \uplus C_{n_1} \uplus ... \uplus C_{n_1}$ i.e. the disjoint union of C_i with n_2 copies of C_{n_1} also satisfies the conditions enforced so far. Enforcing the condition that every cycle subgraph must be either C_i or C_j disallows this latter graph, thus completing the proof of correctness.

$$\mathit{bicycle}(x,i,j) := \mathit{conn}(x) \ \wedge \ \exists y \ \mathit{soc2}(y,i,j) \ \wedge \ y \lessdot_{se} x$$

The proof of correctness of the above formula follows from the definition.

$$\begin{aligned} \mathit{pointedCycleSum}(x,i,j) :=& \mathit{conn}(x) \ \wedge \ \exists y \ y = C_i \uplus C_j \ \wedge \ \exists z \ y \lessdot_{sv} z \\ & \wedge \ z \lessdot_{se}^2 x \ \wedge \ \forall w \ \mathit{bicycle}(w) \supset w \nleq_s x \end{aligned}$$

Let (x,i,j) be a tuple satisfying the formula. It is constructed from the graph $C_i \uplus C_j$ by adding one more vertex v and two more edges e_1, e_2. Suppose without loss of generality that e_1 connects v to the cycle C_i since to form a connected graph, v must be connected to one of the cycles. Then e_2 must join the other cycle C_j to either v or C_i. Suppose it connects C_i and C_j, then there is a bicycle subgraph of x which is disallowed. Therefore e_2 must connect C_j to v. Hence the graph x is the required graph.

We can use the same formula we used in Lemma 4.14 with the formula $allCycles'$ being the formula obtained by replacing occurrences of $\leq_i$ in $allCycles$ by $\leq_s$:

$$csum(x,n) := \mathcal{N}(n) \ \wedge \ \forall z \ maximalComp(x,z) \ \supset \ \mathcal{C}(z) \wedge$$
$$cardCond(x,n) \ \wedge \ allCycles'(x,n)$$

where

$$allCycles'(x,n) := \forall m \ (n+3 \leq m \leq 2n+2) \ \supset \ C_m \leq_s x$$

$$cardCond(x,y) := |y| = |x|^2 + |x|(|x|+1)/2 + 3|x|$$

The formula $maximalComp$ is from Lemma 5.3 and $cardCond$ is definable by conditions $C1$ and $C3$. This is because the combination of $allCycles'$ which states that every cycle is present as a subgraph is equivalent to every cycle being present as an induced subgraph under the condition that every maximal component is a cycle.

□

We now exhibit the defining formula for $\tilde{\mathcal{G}}(x)$.

Lemma 5.6. *The family $\tilde{\mathcal{G}}(x)$ if and only if x is an o-presentation is definable in ($\mathcal{G}, \leq_s$ $, sameSize$).*

Proof. For any o-presentation $x \in \tilde{g}$, its cardinality is related to the cardinality of g by $|x| = |g|^2 + |g|(|g|+1)/2 + 3|g|$. In order for an arbitrary x be an o-presentation, it is necessary and sufficient to show that the graph x is formed by adding some number of edges to $g' = \biguplus_{i=1}^{|g|} C_{|g|+i+2\rightarrow 1}$ such that each new edge is constrained to be between the degree 1 vertices of the $C_{|g|+i+2\rightarrow 1}$. We will first construct g' given a parameter $|g|$ and use it to define $\tilde{\mathcal{G}}$.

The predicate $csumHook(x,n)$ which holds if and only if $x = \biguplus_{i=1}^{n} C_{n+i+2\rightarrow 1}$ is defined by the following formula:

$$csumHook(x,n) := \mathcal{N}(n) \ \wedge\ \exists y\ csum(y,n)\ \wedge\ |x| = |y| + n\ \wedge\ ||x|| = ||y|| + n$$
$$\wedge\ y \leq_s x\ \wedge\ \forall i\ (1 \leq i \leq n) \supset\ C_{n+i+2\to 1} \leq_s x.$$

The only way to get every graph $C_{n+i+2\to 1}$ as a subgraph by adding n vertices and n edges to $\dot\bigcup_{i=1}^{n} C_{n+i+2}$ is by connecting each new vertex to exactly one of the cycles.

The defining formula for $\tilde{\mathcal{G}}$ is given below:

$$\begin{aligned}
\tilde{\mathcal{G}}(x) :=& \exists n \in \mathcal{N}\ cardCond(x,n)\ \wedge\ \dot\bigcup_{i=1}^{n} C_{n+i+2\to 1} \leq_s x \\
&\wedge\ indicatorsInduced(x,n)\ \wedge\ noBicycles(x,n) \\
&\wedge\ noPointedCycleSum(x,n) \\
&\text{where} \\
cardCond(x,y) :=& |y| = |x|^2 + |x|(|x|+1)/2 + 3|x| \\
indicatorsInduced(x,n) :=& \forall y\ \forall i\ ((1 \leq i \leq n)\ \wedge C_{n+i+2\to 1} \leq_{se} y) \supset \neg(y \leq_s x) \\
noBicycles(x,n) :=& \forall y\ \forall i\ \forall j\ ((1 \leq\ i,j\ \leq n)\ \wedge bicycle(y,i,j))\ \supset y \not\leq_s x) \\
noPointedCycleSum(x,n) :=& \forall i\ \forall j\ (1\ \leq i,j\ \leq n) \supset\ C_{n+i+2} +_p C_{n+j+2} \not\leq_s x
\end{aligned}$$

The formula $cardCond$ enforces a cardinality condition that ensures that the every vertex of x has to be used to witness the fact that $\dot\bigcup_{i=1}^{n} C_{n+i+2\to 1}$ is a subgraph. We now need to make sure that none of the extra edges occur and that $\dot\bigcup_{i=1}^{n} C_{n+i+2\to 1}$ occurs as an induced subgraph. The edges inside a particular $C_{n+i+2\to 1}$ are forbidden by $indicatorsInduced$, edges between two indicator cycles are forbidden by $noBicycles$ and $noPointedCycleSum$ ensures that two cycles are not connected to the degree 1 vertex in a $C_{n+i+2\to 1}$. Thus the only extra edges allowed are between the degree 1 vertices of the $C_{n+i+2\to 1}$. □

Remark 5.7. *We note that it is possible to define $\tilde{\mathcal{G}}$ without having to resort to the use* *of the $sameSize$ predicate, but the construction and proof of correctness become more complicated.*

5.1.3 Defining $constructFromCycles$

The relation $constructFromCycles(y, x)$ states that y is formed by adding $|x|$ number of edges to the graph obtained by the disjoint union of x with all its indicator cycles. The construction of the disjoint union of a set of indicator cycles has already been shown in Lemma 5.5, but defining arbitrary disjoint union and counting the number of edges of a graph take more work.

We observe that $sameSize$ and edge counting are equidefinable under the subgraph order.

Observation 5.8. *The predicate $sameSize$ if and only if x and y have the same number of edges allows us to define the related predicate $countEdges(x, n)$ if and only if $n \in \mathcal{N}$ and x has $|n|$ edges. We denote the unique n satisfying $countEdges(x, n)$ for a fixed x by $||x||$.*

$$countEdges(x, n) := n \in \mathcal{N} \ \wedge \ sameSize(x, n)$$

It is also clear that $sameSize(x, y)$ is definable in ($\mathcal{G}, \leq_s, countEdges$):

$$sameSize(x, y) := \exists n \ countEdges(x, n) \ \wedge \ countEdges(y, n)$$

We can now define $constructFromCycles$.

Lemma 5.9. *The predicate $constructFromCycles$ is definable in* $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$.

Proof.

$$constructFromCycles(y, x) := \exists z\ z = x \mathbin{\dot\cup} \dot{\bigcup_{i=1}^{|x|}} C_{|x|+i+2} \wedge |z| = |y| \wedge ||z|| + |x| = ||y||$$

□

This concludes the proof that condition $C2$ holds for the structure $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$, giving us the following result.

Lemma 5.10. *The structure* $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ *is capable.*

5.2 The Subgraph Order is Capable

In this section, we show that the predicates $disjointUnion$ and $sameSize$ are definable in the subgraph order. By the results of the previous section, we then obtain the fact that the subgraph order is a capable structure.

5.2.1 Defining Disjoint Union in the Subgraph Order

We outline the strategy employed for defining disjoint union. Consider $g = g_1 \mathbin{\dot\cup} g_2$. The connected components of g must occur either in g_1 and g_2. Further, for any component c of g, the number of times it occurs as a component is equal to the sum of the number of times it occurs as a component in g_1 and g_2. Checking the two conditions is necessary and sufficient to verify that a given graph g' is indeed the disjoint union of g_1 and g_2. However, we do not have a definition for the predicate $comp(y, x)$ which holds if and only if x is a component of y. Hence we will enforce a condition which turns out to be equivalent when quantified over all graphs.

For any connected graph g_0 which is a subgraph of g, let $maxCopies(g, g_0, n)$ be true if and only if g contains n disjoint copies of g_0 as a subgraph, but not $n + 1$ disjoint copies of g_0. We verify that that the number of copies of any such connected graph g_0 present in g is the sum of the number of copies of g_0 in g_1 and g_2. We show how to define $maxCopies$ first, and then use it to define disjoint union.

Lemma 5.11. *The following predicates are definable in the subgraph order:*

1. *$mult(x, y, n)$ holds if and only if y is a connected graph and x is the disjoint union of n many components, each of which is y.*
2. *$maxCopies(x, y, n)$ holds if and only if y is connected and the graph g satisfying $mult(g, y, n)$ is a subgraph of x but not the graph g' satisfying $mult(g', y, n + 1)$.*

We will refer to the unique n satisfying $maxCopies(x, y, n)$ (where x and y are fixed graphs) as the "the number of copies of y in x" and write $maxCopies(x, y)$ to denote n.

Proof. It is sufficient to define the relation $mult(x, y)$ which holds if and only if x is made of up some arbitrary number of components, each of which is y. Using $mult(x, y)$ we can define $mult(x, y, n)$ if and only if x is made of exactly n many y. This is because we can use arithmetic to get $n = \frac{|x|}{|y|}$.

$$\begin{aligned} mult(x, y) := & zero(x) \ \vee \ (conn(y) \ \wedge \ maximumComp(x, y) \ \wedge \\ & uniqueCompCard(x, y) \ \wedge \ edgeMaximal(x, y)) \end{aligned}$$

where

$$zero(x) := x = \emptyset_g$$

$$maximumComp(x, y) := maximalComp(x, y) \ \wedge \ \forall z \ maximalComp(x, z) \supset z \leq_s y$$

$$uniqueCompCard(x,y) := \forall z\ x \lessdot_{se} z \supset (\forall z'\ maximalComp(z,z') \supset (|z'| = |y| \ \vee\ |z'| = 2|y|))$$

$$edgeMaximal(x,y) := \forall z\ x \lessdot_{se} z \supset \neg maximumComp(z,y)$$

The subformula $zero$ is necessary for the boundary case where $x = \emptyset_g$ with x made of zero number of copies of y. The defining formula for $mult(x,y)$ enforces three conditions:

1. $maximumComp(x,y)$: y is a maximum component of x i.e. for any component c of x, it is the case that $c \leq_s y$.
2. $uniqueCompCard(x,y)$: Adding any edge to x gives a graph z such that every maximal component of z has cardinality either $|y|$ or $2|y|$.
3. $edgeMaximal(x,y)$: Adding any edge to x ensures that y is no longer the maximum component of the new graph.

The proof of correctness of the formulae enforcing these three conditions is straightforward. We now show that they suffice to define $mult$. Let the tuple (x,y) satisfy the defining formula. We have to show that x is the disjoint union of some arbitrary number of y.

Suppose x has a component c which is not y and therefore is a strict subgraph of y. Therefore $y \uplus c$ is an induced subgraph of x since y is also a component of x. Adding an edge between y and c gives us an edge cover z of x which has a component c' of order $|y| + |c|$. Since y is the maximum component of x, c' is a maximal component of z. Hence by $uniqueCompOrder$ which states that any edge cover of x has the property that every maximal component has order $|y|$ or $2|y|$, we get $|c| = |y|$. But we chose an arbitrary c, so all components in x have order $|y|$. Thus any component c of x is obtained by deletion of some number of edges from y while making sure the resulting graph is still connected.

Now, let c be a component which is a strict subgraph of y. We can add an edge to c to get c' which is still a subgraph of y. The edge cover of x thus formed still has as its maximum component y, which contradicts the formula $edgeMaximal$. Hence c is not a strict subgraph of y. Hence all components in x are in fact y. We can now define $maxCopies$:

$$maxCopies(x, y, n) := \exists z\ mult(z, y, n)\ \wedge\ z \leq_s x\ \wedge\ \exists z'\ mult(z, y, n{+}1)\ \wedge\ \neg z' \leq_s x$$

$\square$

Next we use $maxCopies$ to define disjoint union.

Lemma 5.12. *The predicate $disjointUnion(z, x, y)$ if and only if $z = x \uplus y$ is definable in the subgraph order.*

Proof.

$$\begin{aligned} disjointUnion(z, x, y) :=& \forall z'\ conn(z')\ \wedge compUnionInZ(z, x, y, z') \wedge \\ & compZInUnion(z, x, y, z') \\ & \text{where} \\ compUnionInZ(z, x, y, z') :=& ((z' \leq_s x \vee z' \leq_s y) \supset \\ & maxCopies(x, z') + maxCopies(y, z') = maxCopies(z, z')) \\ compZInUnion(z, x, y, z') :=& z' \leq_s z \supset (maxCopies(x, z') + maxCopies(y, z') \\ & = maxCopies(z, z')) \end{aligned}$$

Consider any component c of $x \uplus y$. Such a c must be a subgraph of either x or y since any component of $x \uplus y$ must be a component of either x or y (or both). Hence by $compUnionInZ$, the number of copies of c in $x \uplus y$ which is the same as the sum of the number of copies of c in x and y, is the number of copies of c in z.
By a similar arguement, $compZInUnion$ enforces that the number of copies of a component c in z is the sum of number of its copies in x and y i.e. the number of copies in $x \uplus y$. We now observe that:

Observation 5.13. *For any two graphs g_1 and g_2, $g_1 = g_2$ iff for any component c of g_1, $maxCopies(g_1, c) = maxCopies(g_2, c)$ and for any component c' of g_2, $maxCopies(g_1, c') = maxCopies(g_2, c')$.*

The forward direction is obvious, we consider the reverse. Let $g_1 \neq g_2$ be two graphs of smallest order which form a counterexample. Let c be a maximal component of g_1, then it must also be a maximal component of g_2; otherwise either there exists a component c' of g_2 which is a supergraph of c or c is not a subgraph of g_2. In the first case $maxCopies(g_2, c') > 0$ but $maxCopies(g_1, c') = 0$, in the second $maxCopies(g_2, c) = 0$ but $maxCopies(g_1, c) > 0$. Further, $maxCopies(g_1, c) = maxCopies(g_2, c)$ by assumption and let this number be n_c. Clearly $g_1 = n_c c \uplus g_1'$ and $g_2 = n_c c \uplus g_2'$ i.e. g_1 is the disjoint union of a graph g_1' which does not contain c as subgraph with the graph $n_c c$ which is the disjoint union of n_c maxCopies of c. Similarly for g_2. This implies that $g_1' \neq g_2'$ which contradicts the assumption that g_1, g_2 form the smallest counterexample and the observation follows.

We have shown above that $x \uplus y$ and z satisfy the property in the above observation and hence $z = x \uplus y$. This concludes the proof of definability of disjoint union in the subgraph order. $\square$

The following corollary will be of use in the next section.

Corollary 5.14. *The predicate $comp(y, x)$ which holds if and only if x is a component of y is definable in the subgraph order.*

$$comp(y, x) := conn(x) \ \wedge \ \exists z \ x \uplus z = y$$

5.2.2 Defining $sameSize$ in the Subgraph Order

By Observation 5.8, it is sufficient to define the predicate $countEdges$ which counts the number of edges of a graph. We will break this problem up into two parts: counting of edges n of a connected graph x using the predicate $countEdgesConn(x, n)$, and counting the number of components n of a graph x using the predicate $countComps(x, n)$.

Let a graph g have m_2 many components. The minimum number of edges to be added to g to get a connected graph g' is $m_2 - 1$. It is clear that the number of edges in g and g' are related by $||g|| + m_2 - 1 = ||g'||$. Thus using $countEdgesConn$ and $countComps$ we can define the required predicate $countEdges$.

Lemma 5.15. *The predicate $countEdgesConn(x, n)$ which holds if and only if x is a connected graph and n is the number of edges of x, is definable in the subgraph order.*

Proof. We construct a gadget to define $countEdgesConn(x, n)$. For any given graph g, we construct a graph g' which has the following properties:

1. g is a component of g'.
2. Every component of g' is formed by adding some number of edges to g.
3. For every component c of g', there is a component c' of g' such that c' is an edge-cover of c, or it is the case that c is a clique.
4. g' is a minimal element under the subgraph order of the set of graphs satisfying the above three properties.

It is easy to see that any such $g' = c_0 \uplus c_1 \uplus_2 ...c_m$ where $c_0 = g, c_m = K_{|g|}$ and for every $i > 0$, c_i is an edge-cover of c_{i-1} (see Figure 5.2 for an example). Hence using some arithmetic, we can relate the cardinality of g' to the number of edges of g and retrieve the latter : $|g|(\binom{|g|}{2} - ||g|| + 1) = |g'|$. Having given the proof idea, we give the formulae below:

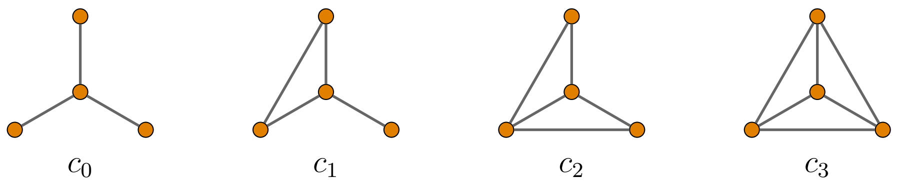


Figure 5.2: The gadget g' for counting number of edges of the connected graph $g = S_4$. Note that $g' = c_0 \uplus c_1 \uplus c_2 \uplus c_3$ with $c_0 = S_4$ and $c_3 = K_4$

First we have a formula which enforces only the first two conditions on g'.

$$\begin{aligned} \mathit{countEdgesGadget}'(y,x) :=& \mathit{conn}(x) \ \wedge \ \mathit{comp}(y,x) \ \wedge \ \forall z \ \mathit{comp}(y,z) \supset \\ & [x \leq_s z \ \wedge \ |x| = |z| \wedge \\ & (\mathcal{K}(z) \vee \exists z' \ (\mathit{comp}(y,z') \ \wedge \ z \lessdot_{se} z'))] \end{aligned}$$

The third condition of minimality is enforced:

$$\begin{aligned} \mathit{countEdgesGadget}(y,x) :=& \mathit{countEdgesGadget}'(y,x) \\ & \wedge \ \forall z \ \mathit{countEdgesGadget}'(z,x) \supset \neg(z <_s y) \end{aligned}$$

We use the constructed gadget to do the edge counting:

$$\mathit{countEdgesConn}(x,n) := \exists y \ \mathit{countEdgesGadget}(x,y) \ \wedge \ |x|(\binom{|x|}{2} - n + 1) = |y|$$

$\square$

The other predicate needed to define $\mathit{countEdges}$ is $\mathit{countComps}$.

Lemma 5.16. *The predicate $\mathit{countComps}(x,n)$ if and only if x contains n components is definable in the subgraph order.*

Proof. We show that for any graph g, we can construct the graph g' which is obtained by adding all the edges uv such that u, v belong to the same component of g (see Figure 5.3 for an example). Such a graph is a disjoint union of cliques and we will call this family of

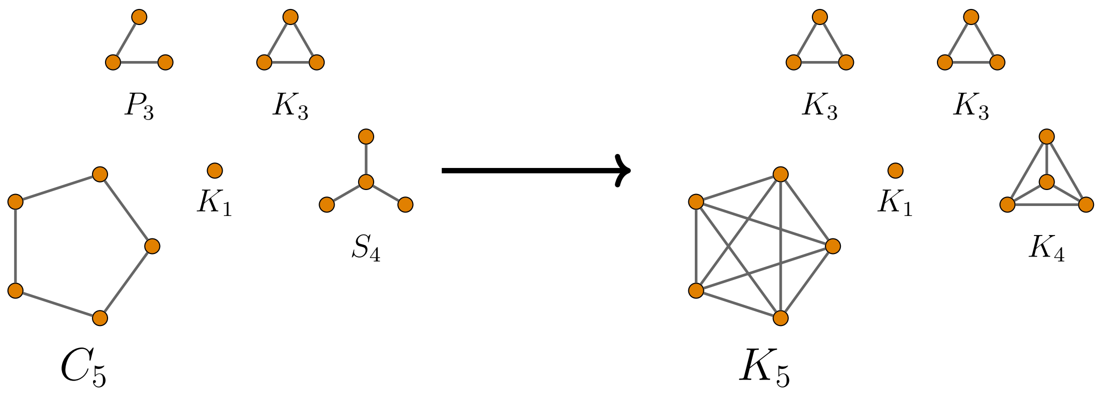


Figure 5.3: The map f_K taking the graph $g = K_1 \uplus P_3 \uplus K_3 \uplus S_4 \uplus C_5$ to the graph $f_K(g) = K_1 \uplus K_3 \uplus K_3 \uplus K_4 \uplus K_5$ belonging to the family $unionOfCliques$.

graphs $unionOfCliques$. The number of components in both g and g' is the same. We then show how to use arithmetic to count components assuming that the input graph is always from the family $unionOfCliques$.

Definition 5.17 (Extend to Cliques)**.** *The map $f : \mathcal{G} \to \mathcal{G}$ takes a graph $g = c_0 \uplus c_1 \cdots \uplus c_n$ (where each c_i is a component) to the graph $f_K(g) = K_{|c_0|} \uplus K_{|c_1|} \uplus \cdots \uplus K_{|c_m|}$.*

It is easy to define the family $unionOfCliques$:

$$unionOfCliques(x) := \forall y\ comp(x, y) \supset \mathcal{K}(y)$$

The predicate $extendToCliques(y, x)$ holds if and only if $y = f_K(x)$ and is defined by the formula

$$\begin{aligned} extendToCliques(y, x) :=& unionOfCliques(y) \wedge \\ & \forall z\ (unionOfCliques(z)\ \wedge\ x \leq_s z) \supset \neg(y <_s z) \end{aligned}$$

We show the correctness of the defining formula. The formula states that y is a member of $unionOfCliques$ which is a minimal element under the subgraph order of the set S of all graphs which are supergraphs of x and also belong to $unionOfCliques$. Clearly the graph $y = f_K(x)$ belongs to S. We claim that it is the unique minimal element of S.

Consider any $y' \in S$. Since y' is a supergraph of $x = c_0 \uplus c_1 \uplus ... \uplus c_n$, there are vertex sets $V_0, V_1, ..., V_n$ such that $V_i \subset V(y')$, $V_i \cap V_j = \emptyset$ for all $i \neq j$ and $x[V_i]$ is a supergraph of c_i with $|V_i| = |c_i|$. Thus $x[V_i]$ is also a connected graph. But this implies that $x[V_i]$ is equal to $K_{|c_i|}$. Hence $f_K(x)$ is also a subgraph of y' and hence is the unique minimal element of S.

Next we show that through the use of arithmetic, we can extract the number of components from a graph of the family $unionOfCliques$.

Arithmetic allows us to create sequences of numbers which can be stored as a single number and manipulate this sequence.

Observation 5.18. *The following predicates are definable in arithmetic:*

1. *$\phi_{sequence}(n, i, j)$ holds if and only if the largest power of the i^{th} prime which divides n is j. We will abuse notation in the sequel and refer to the number $2^{n_1} \times 3^{n_2} \times ... \times p_k^{n_k}$ as the sequence $(n_1, n_2, ..., n_k)$.*
2. *$\phi_{sequenceSum}(n, m)$ holds if and only if the sum of the exponents of all primes dividing n is m i.e. $n = 2^{n_1} \times 3^{n_2} \times ... \times p_k^{n_k}$ and $m = n_1 + n_2 + ... + n_k$.*

The intended usage of $\phi_{sequence}(n, i, j)$ is that n is a sequence of numbers and we can extract the i^{th} number in the sequence if we are given the index i.
Suppose we are able to create a sequence l_x which contains the number of times each clique occurs as a component in a given member x of $unionOfCliques$. For example, if $x = K_2 \uplus K_2 \uplus K_4 \uplus K_5$ then the sequence l_x corresponding to it would be $(0, 2, 0, 1, 1)$ since there are no K_1 components, two K_2 components etc. However, what we can easily obtain using the predicates we have defined is the sequence l'_x corresponding to the numbers $maxCopies(K_i, x)$. In this case, the sequence is $(13, 6, 2, 2, 1)$.

In the general case, we have the following definition which relates these two sequences:

Definition 5.19. *Let $A \subseteq \mathbb{N}^*$ be the subset of sequences of natural numbers such that for any $l \in A$ with $l = (n_1, n_2, ..., n_k)$, it is the case that $n_k > 0$.*
We define a map $f_1 : A \to \mathbb{N}^*$ as follows:

For $l = (n_1, n_2, ..., n_k) \in A$, $f_1(l) = l' = (m_1, m_2, ..., m_k)$ where

$$m_i = \Sigma_{j=1}^{k} \lfloor \frac{j}{i} \rfloor \times n_j$$

Let $B \subseteq \mathbb{N}^*$ be the image set under the map f_1. We can define the inverse map $f_1^{-1} : B \to A$ of the map f_1. Given $l' = (m_1, m_2, ..., m_k)$, it is possible to compute n_i if we know the value of $m_i, n_{i+1}, n_{i+2}, ..., n_k$:

$$\begin{aligned} n_k &= m_k \\ n_i &= m_i - \Sigma_{j=i+1}^{k} \lfloor \frac{j}{i} \rfloor \times n_j \end{aligned}$$

From the above definition, it is clear that the inverse map f_1^{-1} is well defined and is recursive and hence, definable in arithmetic.

Observation 5.20. *The predicate $\phi_{sequenceConvert}(m, n)$ if and only if n, m are sequences such that $n = f_1^{-1}(m)$ is definable in arithmetic.*

Let $makeSequenceFromUOC(x, n)$ hold if and only if n is a sequence, $x \in unionOfCliques$ and for every i, the i^{th} member of the sequence n is the number j such that $maxCopies(x, K_i, j)$ is true. The predicate $makeSequenceFromUOC(x, n)$ is definable in the subgraph order:

$$\begin{aligned} makeSequenceFromUOC(x, n) :=& unionOfCliques(x) \ \wedge \ \forall i, j \in \mathcal{N} \\ & (\psi_{sequence}^{trans}(n, i, j) \iff copies(K_i, x) = j) \end{aligned}$$

Note that $\psi_{sequence}^{trans}$ is a formula in the vocabulary of the subgraph order which is the translation of the arithmetic formula $\phi_{sequence}$ (see Corollary 4.9).

We can define $countComps$ by creating the sequence of number of copies, translating it under the map f_1^{-1} and adding up the elements of the latter sequence:

$$
\begin{aligned}
countComps(x,n) :=\, & \exists y\ extendToCliques(y,x)\ \wedge\ \exists m_1, m_2 \in \mathcal{N}\ \wedge \\
& makeSequenceFromUOC(y, m_1)\ \wedge\ \psi^{trans}_{sequenceConvert}(m_1, m_2) \\
& \wedge\ \psi^{trans}_{sequenceSum}(m_2, n)
\end{aligned}
$$

This concludes the proof of Lemma 5.16. □

We can now define $countEdges$.

Lemma 5.21. *The predicate* $countEdges(x,n)$ *which holds if and only if* x *has* n *edges is definable in the subgraph order.*

Proof. Using Lemma 5.15 (definability of $countEdgesConn$) and Lemma 5.16 (definability of $countComps$), we can define $countEdges$:

$$
\begin{aligned}
countEdges(x,n) :=\, & \exists y\ conn(y)\ x \leq_s y\ \wedge\ \forall z\ conn(z)\ \wedge\ x \leq_s z \supset \neg(z <_s y)\ \wedge \\
& \exists m_1, m_2 \in \mathcal{N}\ countEdgesConn(y, m_1)\ \wedge\ countComps(x, m_2) \\
& \wedge\ n = m_1 - m_2 + 1
\end{aligned}
$$

□

By Observation 5.8, we get :

Corollary 5.22. *The predicate* $sameSize$ *is definable in the subgraph order.*

Combining the capability of $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ proved in Lemma 5.10 with the definability of disjoint union and edge counting proved in Lemma 5.12 and Corollary 5.22 respectively, we conclude that that the subgraph order is a capable structure.

Lemma 5.23. *$(\mathcal{G}, \leq_s)$ is a capable structure.*

5.3 $(\mathcal{G}, \leq_m, sameSize)$ is Capable

The following lemma implies that $(\mathcal{G}, \leq_m, sameSize)$ is a capable structure.

Lemma 5.24. *The subgraph order is definable in the structure $(\mathcal{G}, \leq_m, sameSize)$.*

Proof.

$$x \leq_s y := \exists z \; x \leq_m z \;\wedge\; sameSize(x, z) \;\wedge\; z \leq_m y \;\wedge\; |z| = |y|$$

Suppose $x \leq_s y$. Then y can be constructed from x in two steps. The first step involves addition of an arbitrary number of vertices to give a graph z. The second step involves addition of an arbitrary number of edges to the graph z to get y. Note that this two step construction characterizes the subgraph order. The formula captures this two step construction and its correctness follows from the following observation:

Observation 5.25. *Let $g_1, g_2 \in \mathcal{G}$. If g_1, g_2 have the same number of edges or same cardinality, then*

$$g_1 \leq_s g_2 \iff g_1 \leq_m g_2.$$

This is due to the fact that a contraction operation decreases both the number of edges as well as the number of vertices in a graph. Thus constraining either one of these parameters to remain constant means that contraction operations cannot be used. □

We conclude this section with the formal statement of the main result of this thesis:

Theorem 5.26 (Arithmetical Predicates in Graph Orders)**.** *Each of the structures $(\mathcal{G}, \leq_i, P_3)$, $(\mathcal{G}, \leq_s)$ and $(\mathcal{G}, \leq_m, sameSize)$ has the maximal definability property. In other words, let $Def(\mathcal{A})$ be the set of all definable predicates of a structure $\mathcal{A}$; then*

$$Def(\mathcal{G}, plus_t, times_t) = Def(\mathcal{G}, \leq_s) = Def(\mathcal{G}, \leq_i, P_3) = Def(\mathcal{G}, \leq_m, sameSize).$$

Corollary 5.27. *Every recursively enumerable predicate over graphs is definable in each of the structures* $(\mathcal{G}, \leq_i, P_3)$, $(\mathcal{G}, \leq_s)$ *and* $(\mathcal{G}, \leq_m, sameSize)$.

Chapter 6

Decidability in Graph Order[1]

The arithmetical interpretability results of Chapter 3 show that the full first order theory of graph order is undecidable. We would like to identify fragments of the theory which are decidable. These fragments may be obtained by application of different kinds of restrictions to the first order theory. These restrictions are of three kinds:

1. Vocabulary restrictions: The results of Chapter 4 show that any arithmetical relation $R \subseteq \mathcal{G}$ is definable in the full first order theory of graph order. The theory of $(\mathcal{G}, \tau)$ can be seen as a fragment of the theory of graph order for any set τ of arithmetical relations over graphs.
2. Domain restrictions : Let $\mathcal{G}_0 \subseteq \mathcal{G}$ be a family of graphs definable in graph order. One could consider studying the first order theory $Th(\mathcal{G}_0, \leq)$.
3. Syntactic restrictions: These restrictions are placed on the syntax of the logic, for instance, by considering only some strict subset of the logical symbols. Another way to place a syntactic restriction is by defining a numerical parameter P which maps every FOL formula ϕ to a number $P(\phi)$ and considering only the set of formulae $S_k = \{\phi \mid P(\phi) \leq k\}$ for some $k \in \mathbb{N}$.

[1]The results in this chapter will be presented at Logic Colloquium 2018 to be held at Udine, Italy from 23 to 28 July 2018.

Of the three kinds of restrictions mentioned above, the choice of τ in the first and $\mathcal{G}_0$ in the second depends largely on what is considered important in graph theory; whereas in the third we may direct our attention towards fragments which are analogs of those studied in the satisfiability problem for FOL (see the text The Classical Decision Problem [4]).

In this chapter, we primarily study decidability in the induced subgraph order, with a focus on syntactic restrictions; in some cases the proof obtained goes through in a uniform way for all three graph orders. In particular, we consider the existential fragment and the finite variable fragments in the first two sections of this chapter. In each of these two cases, we further consider what happens when constants are added to the vocabulary; and what happens when negation is disallowed.

At this juncture, we make a couple of remarks regarding negation-free fragments and concerns of measuring the size of a formula which contains constants.

Remark 6.1. *We make a note here about different syntactic forms of the atomic formulae with regards to negation. It is possible to define the following new atomic predicates using* $\{\leq, \nleq\}$*:*

$$x = y := x \leq y \wedge y \leq x$$

$$x < y := x \leq y \wedge x \neq y$$

$$x || y := x \nleq y \wedge y \nleq x$$

The predicates $\{<, >, =, ||\}$ *partition the space* $\mathcal{G} \times \mathcal{G}$ *and are mutually exclusive. Further, they can be taken as atomic predicates and used to define* $\leq, \nleq$*:*

$$x \leq y := x < y \vee x = y$$

$$x \nleq y := y < x \vee x || y$$

When we talk of fragments without negation, we actually mean that the atomic predicates are taken to be $\{<, >, =\}$ *i.e. we exclude* $||$*. Note that* $<$ *cannot be defined using only* $\leq$

without using negation. When dealing with fragments without the positivity constraint, we can choose either of the two sets $\{<, >, ||, =\}$ *or* $\{\leq, \nleq, \geq, \ngeq\}$ *as is convenient. It is convenient to use the symbol* $\nparallel$ *for the comparability relation defined by*

$$x \nparallel y := x \leq y \vee y \leq x.$$

Remark 6.2. *In the case of a formula over a vocabulary which does not contain constants, the size of the formula is taken to be the number of symbols occuring in it. This implicitly assumes that every symbol has the same size. When constant symbols are used for each graph* g*, we define the size of a formula* ϕ *as the sum of the sizes of the symbols occuring in it, where a constant symbol* g *has size* $|g| + ||g||$ *and every non-constant symbol has unit size.*

The results in this chapter on the computational complexity of the fragments of graph order considered assume that a formula ϕ *is represented as a string* s_ϕ *such that the representation of constants* c *contained in the formula is of size* $O(|c| + ||c||)$*. For instance, we could use the representation NR, but any other representation which is polynomially bigger will also work.*

6.1 Existential Fragments

In this section, we will see that the theory $\exists^*(\mathcal{G}, \leq_i)$ is $\mathsf{NP}-$complete, but $\exists^*(\mathcal{G}, \leq_i, \mathscr{C}_g)$ is undecidable. These two results form the first two subsections of this section. In the third subsection, we show that restricting $\exists^*(\mathcal{G}, \leq_i, \mathscr{C}_g)$ to its negation-free fragment gives the theory $\exists^{*(+)}(\mathcal{G}, \leq_i, \mathscr{C}_g)$ which is $\mathsf{NP}-$complete. These results further validate the power of negation applied to atomic formulae containing constants. Many families important from the viewpoint of graph theory have simple definitions using negated atomic formulae; recall that the formulae $K_2 \nleq_i x$ and $K_3 \nleq_m x$ define the families $\mathcal{N}$ and fop respectively.

6.1.1 Existential Fragment without Constants

We show that each of the existential theories $\exists^*(\mathcal{G}, \leq_i), \exists^*(\mathcal{G}, \leq_s), \exists^*(\mathcal{G}, \leq_m)$ of the three graph orders is NP-complete.

The result follows easily from the following lemma which shows that each graph order is universal for finite posets: every finite poset is embeddable in each of the three graph orders.

Lemma 6.3. *Let $P = (\{p_1, p_2, ..., p_n\}, \leq_p)$ be a partial order on n elements. Then P can be embedded into $(\mathcal{G}, \leq)$ where $\leq$ is $\leq_i, \leq_s$ or $\leq_m$.*

Proof. We map each p_i to a graph g_i which is constructed as follows:
Start with a path of length $n + 2$ with vertex set $V = \{v', v_0, v_1, ...v_n, v''\}$ and edge set $E = \{v'v_0, v_nv''\} \cup \{v_iv_{i+1} | 0 \leq i \leq (n-1)\}$.
Add the following vertices and edges to the above path to get g_i:
Add vertices u', u'' and edges $u'v_0, u''v_0$.
For every $0 < j \leq n$ add a new vertex u_j and an edge u_jv_j iff $p_j \leq_p p_i$.
The resulting graph is shown in Fig. 6.1.
Claim: $(\{g_i\}_{1 \leq i \leq n}, \leq_i)$ is isomorphic to $(P, \leq_p)$.
$\underline{p_i \leq_p p_j \Rightarrow g_i \leq_i g_j}$: Let u_k be present in g_i. This implies that $p_k \leq_p p_i$. But since P is a partial order and $p_i \leq_p p_j$, therefore $p_k \leq_p p_j$. Thus by construction u_k is also present in g_j. Delete all vertices in g_j not present in g_i to get g_i from g_j i.e. $g_i \leq_i g_j$.
$\underline{g_i \leq_i g_j \Rightarrow p_i \leq_p p_j}$: First we observe that none of the vertices v', v'', v_i can be deleted since there is a unique longest path in all of the graphs constructed which must be retained. Since g_i can be obtained from g_j by deletion vertices and g_i contains u_i, it must be the case that g_j also contains u_i. But by construction this implies that $p_i \leq_p p_j$. This proves the claim and it follows that the induced subgraph order is universal for finite posets.

For the other two orders $\leq_s$ and $\leq_m$, the direction $p_i \leq_p p_j \Rightarrow g_i \leq g_j$ follows from the fact that an induced subgraph is also a subgraph and minor. For the direction

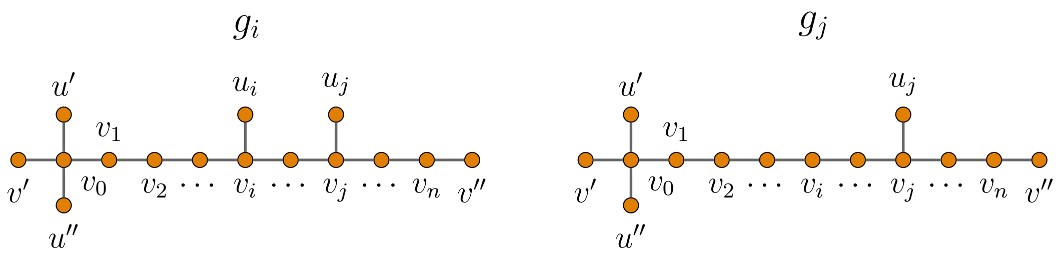


Figure 6.1: Embedding finite posets in graph order : graphs g_i, g_j corresponding to p_i, p_j such that $p_j <_P p_i$.

$g_i \leq g_j \Rightarrow p_i \leq_p p_j$, we observe that the subgraph and induced subgraph orders are identical when restricted to trees. In the case of the minor order, contraction of any edge in the longest path decreases the longest path, contraction of edges from u', u'' decreases the maximum degree to less than 4, and contraction of an edge from u_i to v_i is the same as deletion of the vertex u_i. Thus the result also holds for the minor order. $\square$

The above lemma immediately gives the following result.

Theorem 6.4. *The existential theories of each of the induced subgraph, subgraph and minor orders is* NP*-complete.*

Proof. We demonstrate the assertion for the induced subgraph order. The argument is similar for the other two orders. Any purely existential statement $\exists \bar{x}\ \phi(\bar{x})$ on $|\bar{x}| = n$ variables in the induced subgraph order asserts the existence of a substructure with certain conditions on how the variables are ordered. Since $\leq_i$ contains every n-element partial order and every substructure must be a partial order, it is necessary and sufficient to verify that there exists an n-element partial order $\mathscr{P}$ consistent with the conditions in the quantifier free formula $\phi(\bar{x})$. We can guess such a $\mathscr{P}$ and verify in polynomial time that it is consistent with ϕ.

To prove hardness, we reduce from $3-\mathsf{SAT}$. Let $\phi(\bar{x})$ be a $3-\mathsf{SAT}$ instance with n variables. Replace each x_i occuring in positive form by $x_i \leq z$ and each negative occurence by $x_i > z$ to form $\phi'(\bar{x}, z)$.

$$(\mathcal{G}, \leq_i) \models \exists z\ \exists \bar{x}\ \phi'(\bar{x}, z) \iff \phi(\bar{x}) \text{ is satisfiable.}$$

If there is an assignment σ to $\bar{x}$ making $\phi(\bar{x})$ true, let Y be the set of variables set to 1 by the assignment and Z be the set of variables set to 0 by the assignment. Then assign every variable $x_i \in Y$ and z to the graph $\emptyset_g$; assign every variable $x_i \in Z$ to the graph N_1. This gives a witness for $(\mathcal{G}, \leq_i) \models \exists z\ \exists \bar{x}\ \phi'(\bar{x}, z)$. Conversely, given a witness assigning variables to graphs g_i for each x_i and g for z, for any $g_i \leq_i g$ set $\sigma(x_i)$ to 1 and set all other variables to 0. This gives a satisfying assignment for the original $3 - \mathsf{SAT}$ formula. $\square$

In the next subsection, we show that an attempt to strengthen the decidability result by adding constants to the vocabulary fails.

6.1.2 Existential Fragment with Constants

We show that the existential fragment of the induced subgraph order extended by constants is undecidable. The proof is via reduction to the corresponding fragment of the subword order shown to be undecidable by Halfon etal. [18].

Definition 6.5. *A graph is a threshold graph if it does not contain P_4, K_2K_2, C_4 as induced subgraph and it is not the empty graph. We denote the family of threshold graphs by Thr.*

Observation 6.6. *The family Thr is definable in the induced subgraph order with constants using a quantifier-free formula.*

$$Thr(x) := P_4 \not\leq_i x\ \wedge\ C_4 \not\leq_i x\ \wedge\ K_2K_2 \not\leq_i x\ \wedge\ x \neq \emptyset_g$$

The following lemma is a known property of threshold graphs (see Mahadev and Peled [33]), but we give a proof here for the sake of completeness.

Lemma 6.7. *Every threshold graph contains either a dominating vertex (a vertex adjacent to every other vertex) or isolated vertex (a vertex adjacent to no other vertex).*

Proof. Suppose $g \in Thr$ is a minimal graph which does not contain either a dominating vertex or isolated vertex. Pick any vertex $v_0 \in g$. Let U be the subset of vertices of $V(g)$ to which v_0 has an edge and let $V = V(g) \setminus (U \cup \{v_0\})$. By assumption, both U and V are non-empty sets.

Case $|U| = 1$: Suppose $U = \{u\}$. Let $v_i \in V$ be a vertex not connected to u (else u is a dominating vertex in g). But since g does not have isolated vertices, v_i is connected to some $v_j \in V$. If uv_j is present then v_0, u, v_i, v_j form an induced P_4, else they form an induced K_2K_2.

Case $|V| = 1$: Suppose $V = \{v\}$. Let $u_i \in U$ be a vertex adjacent to v (since v is not isolated). There exists u_j such that u_iu_j is not an edge (else u_j is dominating). If vu_j is an edge then v_0, v, u_i, u_j form an induced C_4 else they form an induced P_4.

Case $|U|, |V| \geq 2$: Because g is a minimal counterexample, there is either a vertex $u_1 \in U$ which is an isolated vertex in $g[U \cup V]$ or there is a vertex $v_1 \in V$ which is a dominating vertex in $g[U \cup V]$. Note that a dominating vertex belonging to U in $g[U \cup V]$ is also a dominating vertex in g and for similar reasons an isolated vertex in V is also not possible. We also note that there must be an edge between U and V. Otherwise, g has multiple components (and no isolated vertices) i.e. there exists an induced K_2K_2. We have the following subcases:

- Subcase u_1 is isolated in $g[U \cup V]$: pick $u_2 \in U$ and $v_i \in V$ such that u_2v_i is an edge. Then u_1, v_0, u_2, v_i form an induced P_4.
- Subcase v_1 is a dominating vertex in $g[U \cup V]$: for any two vertices $u_i, u_j \in U$, v_1u_i and v_1u_j are edges. If u_iu_j is a non-edge, then v_0, u_i, v_1, u_j form an induced C_4. Hence u_iu_j is forced, and since we considered two arbitrary vertices of U, $g[U]$ is a clique. There must exist v_2 such that u_iv_2 is a nonedge (else u_i is dominanting in g).

But this implies $v_0 u_i v_1 v_j$ is an induced P_4.

$\square$

Definition 6.8. *We define the following unary graph operations which take an input graph g:*

- *(AddIsoV) Disjoint union with N_1: $AddIsoV(g) = g \uplus N_1$.*
- *(Complement) Complementation: $Complement(g) = g^c$ which is defined by vertex set $V(g^c) = V(g)$ and edge set $E(g^c) = \{uv \mid uv \notin E(g)\}$.*
- *(AddDomV) Join with N_1: $AddDomV(g) = (g^c \uplus N_1)^c$.*

Theorem 6.9 (Halfon etal.[18]). *Let $\Sigma = \{a, b\}$. The existential theory of the subword order with constants, denoted $\exists^*(\Sigma^*, \leq_{sw}, \mathscr{C}_w)$, is undecidable.*

Lemma 6.10. *The structure $(Thr, \leq_i)$ is isomorphic to $(\Sigma^*, \leq_{sw})$.*

Proof. We define the map $f : \Sigma^* \to \mathcal{G}$ inductively as follows: $f(\epsilon) = N_1$, $f(wa) = AddIsoV(f(w))$, $f(wb) = AddDomV(f(w))$. We need to prove that f is an order preserving bijection between $(\Sigma^*, \leq_{sw})$ and $(Thr, \leq_i)$. For any g, it is clear that $|g| + 1 = |f(g)|$ i.e. the number of vertices in g is the same as one more than the number of letters in the word $f(g)$. Let f_i be the restriction of f to words of cardinality at most i; then $f = \bigcup_{i<\omega} f_i$. We prove by induction on i that f_i is an order preserving bijection between $\Sigma^*_{\leq i} = \{w \mid 0 \leq |w| \leq i\}$ and $Thr_{\leq i+1} = \{g \in Thr \mid 1 \leq |g| \leq i + 1\}$.

<u>Base case:</u> f_0 maps ϵ to N_1. There are no other graphs of cardinality one and the property holds trivially.

<u>Induction:</u> Let f_{i-1} satisfy the required property. We need to show that f_i is an order preserving bijection between $\Sigma^*_{\leq i} = \{w \mid 0 \leq |w| \leq i\}$ and $Thr_{\leq i+1} = \{g \in Thr \mid 1 \leq |g| \leq i + 1\}$.

First, we prove that f_i is a surjection. To do so, we need to show that any graph $g \in Thr_{\leq i}$ is the image of some word w. By Lemma 6.7, we know that g has to contain a dominating or isolated vertex. Let us assume that $g = g' \uplus N_1$. But by induction hypothesis, there exists a word w' such that $f(w') = g'$. Clearly $f(w'a) = g$; the case where g contains a dominating vertex is similar.

Showing that f_i is order preserving also implies that it is an injection. This is because if $f(w) = f(w')$ for $w \neq w'$ with $|w| = |w'| = i$ then $w \not\leq_{sw} w'$ but $f(w) \leq_i f(w')$ which contradicts order preservation. We show that the map f_i preserves order next.

Let w, w' be two words of cardinality at most i. In the case where both of the words have cardinality strictly smaller than i, the requirement follows from induction hypothesis. If $|w| = |w'| = i$ then $w \leq_{sw} w' \iff w = w'$. But $w = w' \Rightarrow f(w) = f(w')$. Conversely, if $f(w) = f(w')$, let $w = va$ and $w' = v'a$ since if w, w' have different last letters then one has a dominating vertex and the other an isolated vertex leading to a contradiction. Then $f(va) = f(v) \uplus N_1, f(v'a) = f(v') \uplus N_1$. But this implies $f(v) = f(v')$ and by induction hypothesis, $v = v' \Rightarrow w = w'$. The argument when the last letter is b is similar.

The remaining case is $|w| = j < i, |w'| = i$. Let $w = w_1w_2...w_j$, $w' = w'_1w'_2...w'_i$ and $w_j \neq w'_i$. Then $w \leq_{sw} w' \iff w \leq_{sw} w''$ where $w'' = w'_1w'_2...w'_{i-1}$. By induction hypothesis $w \leq_{sw} w'' \iff f(w) \leq_i f(w'')$. But by construction $f(w) \leq_i f(w'') \Rightarrow f(w) \leq_i f(w')$ and $f(w) \leq_i f(w') \Rightarrow f(w) \leq_i f(w'')$ by the assumption that $w_j \neq w'_i$. Hence $w \leq_{sw} w'' \iff f(w) \leq_i f(w')$.

Otherwise we have $|w| = j < i, |w'| = i, w_j = w'_i$. Let $w'_i = a$ giving $w = va$, $w' = v'a$. Then $w \leq_{sw} w' \iff v \leq_{sw} v' \iff f(v) \leq_i f(v')$ (by induction hypothesis); but then $f(v) \leq_i f(v') \iff f(va) \leq_i f(v'a)$. The argument for the case of appending b is similar. This completes the proof of order preservation and the lemma follows. □

Theorem 6.11. *The existential theory of the induced subgraph order with a constant symbol for each graph, denoted $\exists^*(\mathcal{G}, \leq_i, \mathscr{C}_g)$, is undecidable.*

Proof. By Observation 6.6, the family of threshold graphs is definable via a quantifier-free formula in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$. The theorem follows from Theorem 6.9 and Lemma 6.10. Given a sentence $\exists \bar{x}\ \psi(\bar{x})$ in the vocabulary of the subword order expanded by constants, we construct the sentence $\exists \bar{x}\ \bigwedge_{x \in \bar{x}} Thr(x) \wedge \psi'(\bar{x})$ in the vocabulary of the induced subgraph order expanded by constants where $\psi'(\bar{x})$ is the formula obtained from $\psi(\bar{x})$ by replacing all occurences of $\leq_{sw}$ by $\leq_i$ and constants from $\mathscr{C}_w$ by constants from $\mathscr{C}_g$ using the map f defined in Lemma 6.10. It is clear that $\exists \bar{x}\ \psi(\bar{x})$ is true of the subword order with constants if and only if $\exists \bar{x}\ \bigwedge_{x \in \bar{x}} Thr(x) \wedge \psi'(\bar{x})$ is true of the induced subgraph order with constants and the construction of the latter formula is recursive. $\square$

Remark 6.12. *The above reduction from the existential fragment of the subword order to the induced subgraph order respects the automorphisms of the two structures. In particular, note that the map f' sending any word w to the word $w_{a \leftrightarrow b}$ which has an a in every position of w which has a b and vice versa for b, is an automorphism of the subword order. The map f in the theorem above sends the word $w_{a \leftrightarrow b}$ to the graph $f(w)^c$ which is the complement of the image of the word w.*

The subgraph and minor orders do not have any automorphisms and it is not clear if it is possible to embed the subword order into either of them.

6.1.3 Positive Existential Fragment with Constants

Next we show that removing negation from the existential fragment with constants gives decidability again. We first define some notions related to posets.

Definition 6.13. *Let $(\mathscr{P}, \leq_P)$ be a poset. A filter F, also known as an upclosed set, is one such that for any $p, p' \in \mathscr{P}$ with $p \leq_P p'$ it is the case that $p \in F$ implies $p' \in F$. Given a set S, we define its upclosure $S \uparrow$ by $S \uparrow = \{p \mid \exists p' \in S\ p' \leq_P p\}$. The strict upclosure $S \Uparrow$ is defined by replacing $\leq_P$ by $<_P$ in the definition of upclosure. We define downclosure of a set by replacing $\leq_P$ by $\geq_P$ in the definition of upclosure and similarly define strict downclosure. When the set S is a singleton, we will write $p \uparrow$ instead of $\{p\} \uparrow$.*

Theorem 6.14. *Let $\leq$ denote any one of $\leq_i, \leq_s, \leq_m$. The theory $\exists^{*(+)}(\mathcal{G}, \leq, \mathscr{C}_g)$, which has formulae given by the grammar*

$$\phi := c\Box x \mid x\Box y \mid \exists y\ \phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

where $\Box \in \{<, >, =\}$ is NP*-complete.*

Proof. The $\mathsf{NP}-$hardness result follows from the fact that the quantifier-free fragment contains formulae of the form $c_1 \leq c_2$ which corresponds to the problem of deciding when a graph is a subgraph, induced subgraph or minor of another graph. These problems are known to be $\mathsf{NP}-$hard problems (see Garey and Johnson [17] for the orders $\leq_i, \leq_s$ and Eppstein [12] for $\leq_m$).

We will show that there exists a certificate for any yes instance of the problem which is polynomial in the input size and verifying that the certificate is valid can be done in time polynomial in the input size.

First we describe the construction of the certificate $Cert_\phi$ for an input sentence assumed to be in prenex form $\exists \bar{x}\ \phi(\bar{x})$ where $\phi(\bar{x})$ is quantifier free. Consider the formula $\phi(\bar{x})$. The atomic formulae in ϕ can be thought of as propositions and thus there exists a quantifier free formula ϕ' which is equivalent to ϕ such that ϕ' is in Disjunctive Normal Form (DNF). Let $\phi' = t_1 \vee t_2 \vee ...t_l$ where each t_i is a *term* [2] of the form $l_1 \wedge l_2 \wedge \cdots \wedge l_n$ with each l_j an atomic formula of the form $x\Box y$ or $c\Box x$. The set of atomic subformulae of ϕ' is contained in the set of atomic subformulae of ϕ. Note that ϕ' may be exponentially bigger than ϕ, but we do not need to construct it. We appeal to the equivalence $\phi \equiv \phi'$ only for the proof of the fact that $Cert_\phi$ is indeed a certificate.

We need to define some parameters of ϕ' to construct the certificate. Let $C = \{c_1, c_2, ..., c_m\}$ be the set of constants used in ϕ', $\bar{x} = \{x_1, x_2, ..., x_n\}$ the set of vari-

[2] Unfortunately, this is standard terminology and is not to be confused with terms in FOL. Note that there are no function symbols in the vocabulary being considered here.

ables in ϕ' and $S = C \downarrow$ the downclosure of the set of contants. The truth of $\exists \bar{x}\ \phi'(\bar{x})$ requires the assignment $\bar{g}$ of graphs to the variables so that $\phi'(\bar{g})$ holds. Fix an assignment $\bar{g}$ making ϕ' true. This happens if and only if there exists a term t of ϕ' such that $t(\bar{g})$ is true. Let t be such a term which witnesses the satisfiability of ϕ'. The size of any term t is at most the size of ϕ since every literal l contained in t is an atomic subformula of ϕ. Let $At(t)$ be the set of literals of t. Define the set $X_t \subseteq \{x_1, x_2, ..., x_n\}$ of *bounded variables* as the smallest set of variables which satisfy the following conditions:

1. For any x if there exists some $c \in C$ such that one of $x < c$, $x = c$ belongs to t, then $x \in X_t$.
2. If $x_i \in X_t$ and one of $x_j < x_i$, $x_j = x_i$ belongs to t then $x_j \in X_t$.

We see that the assignment to any variable in X_t must come from S since the term t imposes a bound on these variables and hence this part of the assignment is polynomial in the size of the input.

This leaves the assignment to the remaining variables, which we will show need not be guessed. Let $X'_t = \bar{x} \setminus X_t$ be the *unbounded variables*. Fix a variable x' in X'_t. Any such variable x' does not occur in an atomic formula of the form $x' < c$, $x' = c$, $x' = y$ or $x' < y$ where y is a bounded variable in the term t. This implies that $\bigwedge_{x \in X_t} x < x'$ is consistent with the term t for any $x' \in X'_t$. Let $g_0 = N_1 \uplus \biguplus_{c \in C} c$. If $g_0 \leq x'$ then $\bigwedge_{x \in X_t} x < x'$ holds because g_0 is a supergraph of every $c \in C$.

The equality relation imposed by t breaks X'_t up into equivalence classes $E_1, E_2, ..., E_k$. Let $X''_t \subseteq X'_t$ be a set of representatives for these equivalence classes. The conditions in t impose a partial order $\leq_t$ on X''_t. Note that since $||$ is not part of the vocabulary, we cannot force any two unbounded variables to be incomparable and this is a critical fact. Let $\leq_{tot}$ be a total order which extends the partial order $\leq_t$. Suppose $X''_t = \{x'_1, x'_2, ..., x_l\}$ where $x'_i <_{tot} x'_j$, then the assignment $g_0 \uplus N_{i+1}$ to the variable x'_i is consistent with the term t. This follows from the consistency of this assignment with the total order $\leq_{tot}$ and the fact that $\bigwedge_{x \in X_t} x < x'$ holds for this assignment. The actual assignment of graphs to the unbounded variables is not part of the certificate; the total order $\leq_{tot}$ suffices since we know that an appropriate assignment exists for each such total order. We are now in a position to describe the certificate.

The polynomial size certificate $Cert_\phi$ consists of the following :

1. The set X_t, an assignment function $\sigma : X_t \to S$ for each $x \in X_t$. The size of X_t is $O(|\phi|)$ and that of σ is $O(|\phi|^2)$.
2. The equivalence classes $E_1, E_2, ..., E_K$ and the order $\leq_{tot}$. The size of this part of the certificate is $O(|\phi| log(|\phi|))$.
3. The set $At(t)$ of atomic subformulae of ϕ which form the set of literals of the term t. This is of size $O(|\phi|)$.
4. Once we have fixed the above elements of the certificate, we can obtain a formula ϕ_σ which is obtained from ϕ by substituting the appropriate assignments to the variables in X_t. We now guess the maps which witness atomic formulae of the form $c\Box c'$ for each such atomic formula in ϕ_σ, depending on which graph order is being considered. For example, if σ assigns g_1 to x and g_2 to y, and there exists an atomic formula $x <_s y$ in $At(t)$, then we need to guess a map $f : V(g_1) \to V(g_2)$ which witnesses the fact that g_1 is a subgraph of g_2. The maps are linear in the size of the graphs and hence in the size of ϕ and there are $At(t)$ many of them. The size of this part of the certificate is $O(|\phi|^2)$.

The process of verifying the certificate is as follows.

1. Plug in the assignment σ contained in $Cert_\phi(1)$ to the variables X_t to produce the formula ϕ_σ.
2. For each atomic formula $\alpha \in At(t)$, there exists a corresponding formula α_σ in ϕ_σ which can be obtained from ϕ. We now have to check that:
   - If α_σ is of the form $c\Box c'$, then verify that the map guessed in $Cert_\phi(4)$ for this formula is correct.
   - If α_σ is of the form $x' < x''$ for $x', x'' \in X_t'$, then verify that the total order

$\leq_{tot}$ in $Cert_\phi(2)$ is consistent with α_σ.

- If α_σ is of the form $x' = x''$ for $x', x'' \in X'_t$, then verify that there exists an equivalence class E_i in $Cert_\phi(2)$ such that $x, x' \in E_i$.

3. Set all atomic formulae not in $At(t)$ to false and those in $At(t)$ to true, and verify that the formula ϕ evaluates to true.

The certificate as outlined is polynomial in the size of the input; the correctness of the certificate follows from the equivalence of $\exists \bar{x}\ \phi(\bar{x})$ and $\exists \bar{x}\ \phi'(\bar{x})$; and the verification process is in polynomial time.

□

We note that if one allows atomic formulae of the form $x_i || x_j$ for two unbounded variables x_i, x_j then the assignment of graphs of the form $g_0 \uplus N_i$ to unbounded variables is no longer consistent with the formula. Hence the above proof does not work for the existential fragment with constants.

This concludes our study of the existential fragment and we take up the finite variable fragments in the next section.

6.2 Finite Variable Fragments

The finite variable fragment FO^k is defined to be the fragment of FOL which only employs variables from $\{x_1, x_2, .., x_k\}$. Nesting of quantifiers allows the reuse of variables, for example $\exists x_1\ (\phi_1(x_1)\ \wedge\ (\forall x_2\ (\phi_2(x_1, x_2)\ \wedge\ \exists x_1\ \phi_3(x_1, x_2))))$ is a sentence in FO^2.

In this section, we will see that the FO^1 fragment of graph order with constants is decidable for each graph order and the FO^3 fragment of the induced subgraph order is undecidable. These two results form the first two subsections of this section. In the third subsection we show that the negation-free fragment of $FO^2(\mathcal{G}, \leq_i)$ is decidable.

6.2.1 The FO^3 Fragment of the Induced Subgraph Order

Theorem 6.15 (Karandikar and Schnoebelen [24]). *The $FO^3 \cap \Sigma_2$ fragment of the subword order with constants is undecidable.*

The quantifier-free interpretation of the subword order in the induced subgraph order from Observation 6.6 and Lemma 6.10 combined with the above theorem also gives us the following undecidability result for the FO^3 fragment of the induced subgraph order with constants.

Theorem 6.16. *The $FO^3 \cap \Sigma_2$ fragment of the induced subgraph order with constants is undecidable.*

It would be interesting to see if the above theorem can be strengthened by getting rid of the constants in the vocabulary. One way to do so would be to show that constants are definable in the $FO^3 \cap \Sigma_2$ fragment of the subword order.

We take up the decidability of the FO^1 fragment next.

6.2.2 FO^1 fragment of graph order

We note that without constants, restriction to a single variable does not make sense since the vocabulary consists of a single binary relation. The quantifier free part of the FO^1 fragment with constants consists only of statements such as $c_1 \leq c_2$. This quantifier-free fragment is known to be NP-complete for each of the three orders. In the case of the induced subgraph order and subgraph order, the independent set problem and the Hamiltonian cycle problem are well known special cases which are already NP-complete (see Garey and Johnson [16]) and similarly finding clique minors is known to be NP-complete (see Eppstein [12]). Due to closure under boolean operations, the FO^1 fragment also contains coNP-complete problems. We give an upper bound for this fragment next.

Formally, the formulae in FO^1 are given by the grammar

$$\phi := c_1 \Box c_2 \mid c \Box x \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \exists x\ \phi \mid \forall x\ \phi,$$

where $\Box \in \{\leq, \geq, \nleq, \ngeq\}$.

Theorem 6.17. *The truth problem for the FO^1 fragment of each of the structures $(\mathcal{G}, \leq_i, \mathscr{C}_g), (\mathcal{G}, \leq_s, \mathscr{C}_g), (\mathcal{G}, \leq_m, \mathscr{C}_g)$ is contained in* $\mathsf{NP}^{\mathsf{NP}} \cap \mathsf{coNP}^{\mathsf{NP}}$.

Proof. We observe that any FO^1 sentence ψ can be written in polynomial time as conjunctions and disjunctions of subformulae of one of the forms

1. $c_1 \Box c_2$ for $\Box \in \{\leq, \geq, \nleq, \ngeq\}$.
2. $\exists x\ \phi(x)$, where $\phi(x)$ is quantifier free and contains atomic formulae of the kind $c \Box x$ for $\Box \in \{\leq, \geq, \nleq, \ngeq\}$.
3. $\forall x\ \phi(x)$, where $\phi(x)$ is quantifier free and contains atomic formulae of the kind $c \Box x$ for $\Box \in \{\leq, \geq, \nleq, \ngeq\}$.

The above normal form follows from pulling out subformulae which are not contained in the scope of a quantifier. Further, ψ is a boolean combination of subformulae of the kind $c_1 \Box c_2$ and $\exists x\ \phi(x)$ since $\forall x\ \phi(x)$ can be rewritten using negation and $\exists x\ \phi(x)$. Alternately, we have a dual representation of ψ as a boolean combination of only $c_1 \Box c_2$ and $\forall x\ \phi(x)$ subformulae. We will show that using the subformulae of the kind $c_1 \Box c_2$ and $\exists x\ \phi(x)$ which we will call *propositions*, we have a procedure which is in $\mathsf{NP}^{\mathsf{NP}}$. By using the dual representation, we also obtain a $\mathsf{coNP}^{\mathsf{NP}}$ procedure, which gives us the result that the problem is in $\mathsf{coNP}^{\mathsf{NP}} \cap \mathsf{NP}^{\mathsf{NP}}$.

We now demonstrate the $\mathsf{NP}^{\mathsf{NP}}$ procedure. The truth of propositions of the form $c_1 \Box c_2$ can be determined using an NP oracle. Suppose we have a $\mathsf{NP}^{\mathsf{NP}}$ procedure which solves the truth problem for propositions of the kind $\exists x\ \phi(x)$, then we can obtain the truth values of all the propositions of ψ in $\mathsf{NP}^{\mathsf{NP}}$, followed by a simple circuit evaluation which is linear time, which completes the $\mathsf{NP}^{\mathsf{NP}}$ procedure. It is sufficient to prove the following claim to obtain an $\mathsf{NP}^{\mathsf{NP}}$ procedure which solves the truth problem for propositions of the kind $\exists x\ \phi(x)$.

Claim: If $\exists x\ \phi(x)$ is true, then there is a graph g whose cardinality is polynomial in $|\phi|$ such that $\phi(g)$ is true. We call g a polynomial-size witness for $\exists x\ \phi(x)$.

Assuming the above claim gives us a polynomial-size witness g; the problem of evaluating the truth of $\phi(g)$ can be done by using an NP oracle which is used to evaluate the truth of subformulae of the form $g\Box c$ contained in $\phi(g)$ followed by a circuit evaluation. The guess of the witness g followed by calls to an NP oracle results in an $\mathsf{NP}^{\mathsf{NP}}$ procedure.

We will now prove the claim that there is a polynomial-size witness g for $\exists x\ \phi(x)$. As in the proof of Theorem 6.1.3, there exists a quantifier-free formula $\phi'(x)$ such that $\phi(x) \equiv \phi'(x)$ with ϕ' in DNF. It is sufficient to prove that a polynomial-size witness exists for sentences of the kind $\exists x\ \phi'(x)$ with $\phi'(x) = l_1 \wedge l_2 \cdots l_n$ where each l_i is a *literal* of the kind $x\Box c$ for $\Box \in \{\leq, \geq, \not\leq, \not\geq\}$.

If there exists a literal $x \leq c$ in ϕ' then we immediately obtain a polynomial bound on x and therefore we can assume this is not the case. Suppose there is a literal of the form $x \not\leq c$ in ϕ'. Since $[\![x \not\leq c]\!] = \mathcal{G}_{|c|+1} \cup S_0$, where S_0 is the finite set of graphs with cardinality smaller than $|c|$ which are incomparable to c , the claim follows if there exists a solution for x which is in S_0. Thus we can replace $x \not\leq c$ by $\bigvee_{c' \in \mathcal{G}_{=|c|+1}} c' \leq x$ and assume that atomic formulae that occur in ϕ' are of the form $c \leq x, c \not\leq x$ only.

Let C, D be the sets of constants that occur in ϕ' which occur in the atomic forms $c \leq x, d \not\leq x$ respectively i.e. $\phi' = \bigwedge_{c \in C} c \leq x \wedge \bigwedge_{d \in D} d \not\leq x$. Let $g \in [\![\phi']\!]$ be a graph that contains every c as induced subgraph and does not contain any d as induced subgraph. The property of not containing d as an induced subgraph is downclosed i.e. any induced subgraph of g also does not contain d (this remains true for the other graph orders). Let $V_0 \subseteq V(g)$ be the set of vertices such that $g[V_0]$ contains every $c \in C$ as induced subgraph. Clearly $|V_0| \leq \Sigma_{c \in C}|c|$ and thus we have a witness for x which is polynomial in the size of the formula and this proves the claim.

This concludes the proof that the truth problem for the FO^1 fragment with constants is in $\mathsf{NP}^{\mathsf{NP}} \cap \mathsf{coNP}^{\mathsf{NP}}$. $\square$

Remark 6.18. *We leave the lower bound problem open for the case of FO^1. Clearly the problem is both* NP*−hard and* coNP*−hard since the quantifer-free fragment of FO^1 contains hard problems of the form $c_1 \square c_2$.*

These results motivate the question of whether the FO^2 fragment is decidable. While we do not have a decision procedure on hand for the FO^2 fragment, analogous results in the case of words and partial results in the case of graphs give evidence for believing that the FO^2 fragment is decidable.

In the next subsection, we give a decision procedure for the positive fragment of the FO^2 theory of the induced subgraph order without constants, denoted
$FO^{2(+)}(\mathcal{G}, \leq_i)$.

6.2.3 $FO^{2(+)}$ Fragment of the Induced Subgraph Order

We give a proof of decidability of the negation-free fragment of the FO^2 theory of the induced subgraph order without constants denoted $FO^{2(+)}(\mathcal{G}, \leq_i)$ in this section. Note that we do not add the constant P_3 to the vocabulary and in this section, by the induced subgraph order we mean the structure $(\mathcal{G}, \leq_i)$. The strategy employed may lead to analogous results for the subgraph and minor orders, but we do not pursue this. We believe the decidability of the FO^2 fragment can be obtained via similar means and we discuss this in the next subsection. We make the following observation regarding the automorphism of the induced subgraph order which maps every graph g to its complement g^c. Note that we use the notation $\mathcal{G} \setminus S$ for set complementation and this is not to be confused with graph complementation.

Observation 6.19. *Every set definable in the first order theory of induced subgraph order is closed under graph complement i.e. for any formula $\phi(\bar{x})$ with n free variables, the n-tuples defined by it have the property that if $\bar{g} = (g_1, g_2, ..., g_n)$ satisfies ϕ, then so does $\bar{g}^c = (g_1^c, g_2^c, ..., g_n^c)$.*

We now describe the strategy employed to show the decidability of $FO^{2(+)}(\mathcal{G}, \leq_i)$. Consider a sentence $\psi \in FO^{2(+)}$. There are no quantifier free sentences in this fragment and so we can assume $\psi = Qx\ \phi(x)$ where Q is either $\exists$ or $\forall$ and $\phi(x)$ is a formula with one free variable. The truth problem for $FO^{2(+)}(\mathcal{G}, \leq_i)$ reduces to checking the emptiness or universality of the solution set $[\![\phi(x)]\!]$ depending on whether Q is the existential or universal quantifier respectively.

We show that any solution set $[\![\phi(x)]\!]$ of a formula in $FO^{2(+)}$ can be effectively and finitely encoded. The resulting finite encoding s_ϕ has the following properties:

1. Given an encoding s_ϕ, it is possible to decide whether $[\![\phi(x)]\!] = \emptyset$ and whether $[\![\phi(x)]\!] = \mathcal{G}$.
2. An encoding s_ϕ can be computed for $[\![\phi(x)]\!]$ given encodings for its strict subformulae.

We now take up the details of how the above is accomplished.

Definition 6.20. *A principal filter is a set $S \subseteq \mathcal{G}$ defined as the upclosure of a single element i.e. $S = g \uparrow$. A special multifilter S' is a finite union of principal filters that the following additional properties:*

1. *S' does not contain either $\emptyset_g$ or N_1.*
2. *If for any $i \in \mathbb{N}$, $N_i \in S'$, then it is the case that S contains all graphs on i vertices, denoted $\mathcal{G}_{=i}$. We call any set satisfying this condition a convex set.*

An almost special multifilter S'' is the union of a special multifilter S_1 with a set S_2 where $S_2 \subseteq \{\emptyset_g, N_1\}$.

Note that we consider $S' = \emptyset$ also as a special multifilter since it is the finite union of zero many principal filters.

We will also need to refer to the following kinds of sets in our proof.

Definition 6.21. *Let $S \subseteq \mathcal{G}$. A layer of S is a set $S_{=n} = \{g \in S \ : \ |g| = n\}$ for some $n \in \mathbb{N}$. We also define $S_n = \{g \in S \ : \ |g| \geq n\}$ and $S_{\leq n} = \{g \in S \ : \ |g| \leq n\}$.*
In particular, $\mathcal{G}_{=n}$ stands for the set of graphs of cardinality n for some fixed $n \in \mathbb{N}$, $\mathcal{G}_n$ for graphs of cardinality at least n and $\mathcal{G}_{\leq n}$ for graphs of cardinality at most n.

We introduce the encoding we use for almost special multifilters.

Definition 6.22. *A special multifilter S is encoded by the finite set $min(S)$ of its minimal elements. The union of a special multifilter with a finite set S' is encoded by $min(S)$ together with the list of elements present in S'.*

Note that it is possible to encode a special multifilter S by any finite set S' such that $S = S' \uparrow$. Clearly $S = S' \uparrow = min(S) \uparrow$ and $min(S) \subseteq S'$. However, one can always eliminate $S' \setminus min(S)$ by comparing each element of S' with the rest of S' to obtain $min(S)$.

Observation 6.23. *Given the encoding of a set S which is an almost special multifilter, checking emptiness and universality of S are both decidable. In fact, both of these problems are trivial because any S which has a non-empty set of minimal elements is non-empty and only $S = (K_2 \uparrow \cup N_2 \uparrow) \cup \{\emptyset_g, N_1\}$ which is encoded by ($\{K_2, N_2\}, \{\emptyset_g, N_1\}$) is universal.*

Definition 6.24. *$Sol_{FO^{2(+)}}$ is defined to be the set of all solutions sets of $FO^{2(+)}$ formulae in one free variable.*

$$Sol_{FO^{2(+)}} := \{[\![\phi(x)]\!] \mid \phi(x) \in FO^{2(+)}\}$$

Lemma 6.25. *Every set $S \in Sol_{FO^{2(+)}}$ is an almost special multifilter.*

Given a formula $\phi(x) \in FO^{2(+)}(\mathcal{G}, \leq_i)$, an encoding s_ϕ of the solution set $[\![\phi(x)]\!]$ can be effectively computed.

We postpone the proof of the above lemma and introduce some tools used to prove it first.

Definition 6.26. *An $FO^{2(+)}$ formula in one free variable is said to be in Operator Normal Form (ONF) if it can be generated by the following grammar :*

$$\phi(x) := S(x) \mid \phi_1(x) \wedge \phi_2(x) \mid \phi_1(x) \vee \phi_2(x) \mid \forall y\ \phi(y)\ \vee\ x\Box' y \mid \exists y\ \phi(y)\ \wedge\ x\Box y$$

where $\Box \in \{<, >, =\}$; $\Box' \in \{<, >, =, \leq, \geq, \not\parallel\}$ and $S(x)$ is one of the formulae $\exists y\ x <_i y, \exists y\ y <_i x, \forall y\ x \leq_i y$ or $\forall y\ x \leq_i y \vee y <_i x$.

The formulae of the form $S(x)$ in the above definition correspond to formula with a single quantifier and one free variable x over $FO^{2(+)}$ and form the base case of our analysis later in this section.

The usefulness of the ONF lies in the fact that the cases in Definition 6.26 correspond naturally to operations on $FO^{2(+)}$ solution sets which behave well with respect to the encoding of the solution sets. This enables us to compute an encoding of a solution set $[\![\phi(x)]\!]$ given encodings of the solution sets of its subformulae.

Definition 6.27. *Let $O : (2^{\mathcal{G}})^k \to 2^{\mathcal{G}}$ be a $k-$ary operation sending $k-$tuples of subsets of $\mathcal{G}$ to a subset of $\mathcal{G}$.*

The collection of almost special multifilters is said to be closed under an operation O if whenever each of $S_1, S_2, ..., S_k$ is an almost special multifilter, then $O(S_1, S_2, ..., S_k)$ is also an almost special multifilter.

The collection of almost special multifilters is said to be recursively closed under an operation O if it is closed under O and in addition, an encoding of $O(S_1, S_2, ..., S_k)$ can be computed given encodings for each of the S_i.

Every case in the grammar for the ONF in Definition 6.26 corresponds to an operation. For instance $\phi_1(x) \wedge \phi_2(x)$ corresponds to an operation which takes in $[\![\phi_1(x)]\!]$ and $[\![\phi_2(x)]\!]$ as input and produces their intersection as output.

Definition 6.28. *The set $\mathscr{O}_{FO^{2(+)}}$ of $FO^{2(+)}$-operations on a poset ($\mathscr{P}, \leq$) is comprised of the two binary operations union and intersection and the following unary operations which take a set $S \subseteq \mathscr{P}$ as input:*

$$\exists^{\Downarrow}(S) := \{p \mid \exists y\ S(y)\ \wedge\ p < y\}$$

$$\exists^{\Uparrow}(S) = \{p \mid \exists y\ S(y)\ \wedge\ p > y\}$$

$$\forall^{\Uparrow}(S) = \{p \mid \forall y\ S(y)\ \vee p < y\}$$

$$\forall^{\Downarrow}(S) = \{p \mid \forall y\ S(y)\ \vee p > y\}$$

$$\forall^{\uparrow}(S) = \{p \mid \forall y\ S(y)\ \vee p \leq y\}$$

$$\forall^{\downarrow}(S) = \{p \mid \forall y\ S(y)\ \vee p \geq y\}$$

$$\forall^{\nparallel}(S) = \{p \mid \forall y\ S(y)\ \vee p \nparallel y\}$$

Observation 6.29. *Let S, S' be the solution sets of $FO^{2(+)}$ formulae $\phi(x), \phi'(x)$ respectively.*

$$[\![\phi(x) \wedge \phi'(x)]\!] := S \cup S'$$

$$[\![\phi(x) \vee \phi'(x)]\!] := S \cap S'$$

$$[\![\exists y\ \phi(y)\ \wedge\ x < y]\!] := \exists^{\Downarrow}(S)$$

$$[\![\exists y\ \phi(y)\ \wedge\ x > y]\!] := \exists^{\Uparrow}(S)$$

$$[\![\forall y\ \phi(y)\ \vee x < y]\!] := \forall^{\Uparrow}(S)$$

$$[\![\forall y\ \phi(y)\ \vee x > y]\!] := \forall^{\Downarrow}(S)$$

$$[\![\forall y\ \phi(y)\ \vee x \leq y]\!] := \forall^{\uparrow}(S)$$

$$[\![\forall y\ \phi(y)\ \vee x \geq y]\!] := \forall^{\downarrow}(S)$$

$$[\![\forall y\ \phi(y)\ \vee x \nparallel y]\!] := \forall^{\nparallel}(S)$$

Lemma 6.30. *Any $FO^{2(+)}$ formula ψ is equivalent to a formula ψ' in ONF and ψ' can be computed from ψ.*

Proof. The lemma follows from the distribution of the existential quantifier over disjunctions, distribution of the universal quantifier over conjunctions and the pairwise inconsistency of formulae of the form $x\Box y$ for $\Box \in \{<, >, =\}$. $\Box$

We are now ready to prove Lemma 6.25. We will show that the set of almost special multifilters is recursively closed under the set $\mathscr{O}_{FO^{2(+)}}$ of operations and that every set in $Sol_{FO^{2(+)}}$ which is the solution set of a formula $\phi(x)$ with only one quantifier is an almost special multifilter with known encoding. This implies that every set S in $Sol_{FO^{2(+)}}$ is an almost special multifilter and further we can compute an encoding of S.

Proof of Lemma 6.25.

Let $\mathscr{S}_{base} = \{\mathcal{G}, \mathcal{G} \setminus \{\emptyset_g\}, \{\emptyset_g, N_1\}, \{\emptyset_g\}, \{N_1\}\}$.

The only sets that are definable in $FO^{2(+)}$ using a single quantifier are those in $\mathscr{S}_{base}$ and $\mathscr{S}_{base}$ is closed under unions and intersections. Let us call any set S that can be written as $S_1 \cup S_2$ where S_1 is a special multifilter and $S_2 \subseteq \{\emptyset_g, N_1\}$ an *almost special multifilter set*. We note that each of the sets in $\mathscr{S}_{base}$ is an almost special multifilter

$$[\![\exists y\ x <_i y]\!] = \mathcal{G} = (N_2 \uparrow \cup K_2 \uparrow) \cup \{\emptyset_g, N_1\}$$

$$[\![\exists y\ y <_i x]\!] = \mathcal{G} \setminus \{\emptyset_g\} = (N_2 \uparrow \cup K_2 \uparrow) \cup \{N_1\}$$

$$[\![\forall y\ x \leq_i y]\!] = \{\emptyset_g\}$$

$$[\![\forall y\ x \leq_i y \vee y <_i x]\!] = \{\emptyset_g, N_1\}$$

$\{N_1\}$ can be produced by intersection of $\mathcal{G} \setminus \{\emptyset_g\}$ and $\{\emptyset_g, N_1\}$.

All other sets definable in $FO^{2(+)}$ are built from those in $\mathscr{S}_{base}$ using the operations $\mathscr{O}_{FO^{+(2)}}$ by the fact that every formula can be effectively converted to one in ONF by Lemma 6.30. We need to show recursive closure of almost special multifilter sets under the $\mathscr{O}_{FO^{+(2)}}$ operations to complete the proof.

The $\mathscr{O}_{FO^{+(2)}}$ operations satisfy some obvious inclusions which help in case analysis later.

Observation 6.31. *Let $\Box, \Box' \in \{\leq, \geq, <, >, \nparallel\}$ and let $\forall^{\Box}$ and $\forall^{\Box'}$ be the corresponding operations as defined in Definition 6.26. If $x \Box y \supset x \Box' y$, then $\forall^{\Box}(S) \subseteq \forall^{\Box'}(S)$.*

If $S \subseteq S'$, then $\forall^{\Box}(S) \subseteq \forall^{\Box}(S')$.

Observation 6.32. *Any filter $c \uparrow$ contains members of $\mathcal{N}$ iff $c = N_i$ for some i.*

We now take up the recursive closure of almost special multifilters under the $\mathscr{O}_{FO^{+(2)}}$ operations. Let $S = S_1 \cup S_2$ and $S' = S'_1 \cup S'_2$.

<ins>Closure under unions and intersections:</ins>

$S \cup S' = (S_1 \cup S_2) \cup (S'_1 \cup S'_2) = (S_1 \cup S'_1) \cup (S_2 \cup S'_2)$ is of the required form because convexity is preserved under unions.

$S \cap S' = (S_1 \cup S_2) \cap (S'_1 \cup S'_2) = (S_1 \cap S'_1) \cup (S_2 \cap S'_2)$ because $S_1 \cap S'_2 = \emptyset$. We are now required to show that $(S_1 \cap S'_1)$ is a special multifilter The intersection of principal filters gives a finite union of principal filters because of the finite lub property. It is easy to verify that convexity is retained under intersection.

<ins>Closure under the $\exists^{\Box}$ operators:</ins>

$S \Downarrow = (S_1 \cup S_2) \Downarrow = S_1 \Downarrow \cup S_2 \Downarrow$. $S_2 \Downarrow \subseteq \{\emptyset_g, N_1\}$. For any filter $c \uparrow$, it is the case that

$(c\uparrow)\Downarrow = \mathcal{G}$. This is because for any $g \neq c$, $g \cup c \in c\uparrow$ i.e. the disjoint union graph is the required witness. This means any nonempty special multifilter gives $\mathcal{G}$, which is convex.

$S\Uparrow = (S_1 \cup S_2)\Uparrow = S_1\Uparrow \cup S_2\Uparrow$. If S_2 is non-empty, then $S_2\Uparrow$ is either $\mathcal{G} \setminus \{\emptyset_g\} = \mathcal{G}_1$ or $\mathcal{G}_2$; both of which always contain $S_1\Uparrow$ for any choice of S_1 and both are convex.

Let S_1 be represented by $C = \{c_1, c_2, ..., c_n\}$. We can assume that there are no c_i, c_j which are comparable i.e. C is a minimal set. Then $S_1\Uparrow$ is represented by $C' = \{c' \mid \exists c \in C, c' \in UC(c)\}$ where $UC(c)$ is the set of upper covers of c. If C does not contain any N_i, then neither can C' and hence $S\uparrow$ is convex by Observation 6.32. Suppose $N_i \in C$, then for every other $c \in C$, $|c| \leq i$ by convexity. $N_i \notin S\Uparrow$ but $N_{i+1} \in S\Uparrow$. Since $\mathcal{G}_{=i} \subseteq S$, this means that $\mathcal{G}_{=i+1} \subseteq S\Uparrow$ since no element of $\mathcal{G}_{i+1}$ can be minimal in S. Thus $S\Uparrow$ is also convex.

Closure under the $\forall^{\Box}$ operators:

We divide these into cases.

Case 1: S_1 does not contain a filter rooted ar N_i for any i

In this case, $\forall^{\Updownarrow}(S) = \emptyset$ no matter the value of S_2. By Observation 6.31, every operator $\forall^{\Box}(S)$ is also $\emptyset$.

Case 2 : S_1 contains a filter rooted at some N_i

Subcase 2.1 : $S_1 = N_2\uparrow \cup K_2\uparrow$

If $S_2 = \emptyset$, then $\forall^{\Downarrow}(S) = \mathcal{G}_2$, $\forall^{\downarrow}(S) = \mathcal{G}_1$, $\forall^{\Updownarrow}(S) = \mathcal{G}$, $\forall^{\Uparrow}(S) = \emptyset$, $\forall^{\uparrow}(S) = \{\emptyset_g\}$.

If $S_2 = \{\emptyset_g, N_1\}$, then $\forall^{\Box}(S) = \mathcal{G}$ for any value of $\Box$.

This implies that for $S_2 = \{\emptyset_g\}$ and $S_2 = \{N_1\}$, $\mathcal{G}_2 \subseteq \forall^{\Box}(S)$ by Observation 6.31 because it is sandwiched between $S_2 = \emptyset$ and $S_2 = \{\emptyset_g, N_1\}$ above.

Subcase 2.2 : $S_1 \cap \{K_2, N_2\} = \emptyset$

$\forall^{\uparrow}(S) \subseteq \{\emptyset_g, N_1\}$ no matter the value of S_2, because for any graph m, $\{K_2, N_2\} \subseteq m\uparrow$ implies that $m = \emptyset_g$ or $m = N_1$. By Observation 6.31, $\forall^{\Uparrow}(S) \subseteq \{\emptyset_g, N_1\}$.

We claim $\forall^{\nparallel}(S) = \forall^{\nparallel}(S) = \forall^{\downarrow}(S) = \forall^{\Downarrow}(S)$. Since $\{K_2, N_2\} \cap S = \emptyset$ and by assumption N_i is the smallest member of $\mathcal{N}$ present in S, it must be the case that for any $m \in \forall^{\nparallel}(S)$, N_{i-1} and K_{i-1} are part of $m \Downarrow$ where $i \geq 2$. This is because both N_{i-1}, K_{i-1} cannot be in $m \uparrow$. This implies that m also belongs to $\forall^{\Downarrow}(S)$.

Since by assumption S contains filters rooted at N_i and K_i (the latter due to Observation 6.19), by Ramsey theory, $S' = \mathcal{G} \setminus S$ is finite and can be computed. This gives us

$$\forall^{\Downarrow}(S) := \{m \mid \exists g \; g \in lub(S'), g < m\}$$

Thus $\forall^{\Downarrow}(S)$ is a finite union of filters and since $\{K_2, N_2\} \subseteq S'$, $lub(S') \cap (\mathcal{K} \cup \mathcal{N}) = \emptyset$ and thus $\forall^{\Downarrow}(S)$ is convex.

It is clear from the proof of closure above that we can compute an encoding of the new set produced in every case considered. This concludes the proof of recursive closure of almost special multifilter sets under the $\mathscr{O}_{FO^{+(2)}}$ operations. This shows that every set in $Sol_{FO^{2(+)}}$ is an almost special multifilter and also concludes the proof of Lemma 6.25.

Theorem 6.33. *The theory $FO^{2(+)}(\mathcal{G}, \leq_i)$ is decidable.*

Proof. The input is an $FO^{2(+)}$ sentence $\Phi = Qx\ \phi(x)$. We convert $\phi(x)$ to $\phi'(x)$ which is in ONF by Lemma 6.30. Using Lemma 6.25, we get an encoding of the solution set $[\![\phi'(x)]\!]$ and by Observation 6.23 we can decide whether $[\![\phi'(x)]\!]$ is empty or $\mathcal{G}$. $\square$

6.3 Towards FO^2 Decidability

In this section, we introduce the notion of a *locally recursive* poset $(\mathscr{P}, \leq_P)$. Graph orders are locally recursive posets. We also define a normal form for solution sets of quantifier-free formulae with one free variable over the vocabulary $\{\leq_P, \mathscr{C}_P\}$ (where $\mathscr{C}_P$ is a set of constants corresponding to the domain elements in $\mathscr{P}$) for any locally recursive poset. This normal form is called the *Multiverse Normal Form* (MNF) and gives rise to an associated family of solution sets called multiverses which have a natural encoding analogous to the special multifilters introduced in the previous section. Testing for emptiness or universality of (the encoding of) a multiverse is easy and the problem reduces to showing the recursive closure of (the encodings of) multiverses under operations arising from the FO^2 fragment. These operations are analogous to those obtained for $FO^{2(+)}$ in Definition 6.28.

In contrast with special multifilter sets, multiverses can be represented directly because of the presence of constants in the vocabulary and the strategy in the previous section when applied to $FO^2(\mathscr{P}, \leq_P, \mathscr{C}_P)$ can be thought of as a quantifier elimination procedure. The strategy, if successful, would give us the result that every set definable in $FO^2(\mathscr{P}, \leq_P, \mathscr{C}_P)$ is a multiverse whose encoding can be computed.

Resolution of the latter problem regarding recursive closures depends on the specific poset $\mathscr{P}$ under consideration and is a problem which is combinatorial rather than logical in flavour. In the case of graph orders, the combinatorics required can be seen as a generalization of the results obtained for the subword order by Karandikar and Schnoebelen [25] in view of the quantifier-free interpretation of the subword order in the induced subgraph order in Section 6.1.

In the last subsection, we obtain some results related to constant definability in $FO^2(\mathcal{G}, \leq_i)$ which indicate that $FO^2(\mathcal{G}, \leq_i, \mathscr{C}_g)$ may lie on the same side of the decidability-undecidability divide, in contrast to the existential fragment of graph order. In the rest of this section, we assume that constants are part of the vocabulary.

6.3.1 Definability in Locally Recursive Posets

We start with a poset $(\mathscr{P}, \leq_P)$ with a countably infinite domain $\mathscr{P}$. By the structure $(\mathscr{P}, \leq_P, \mathscr{C}_P)$ we mean the poset expanded with constant symbols for each member of the domain.

Definition 6.34. *For any $S \subseteq \mathscr{P}$ for a poset $(\mathscr{P}, \leq_P)$ define $min(S)$ to be the set of minimal elements of S, $max(S)$ to be the set of maximal elements of S and $lub(S) = \{p \mid \forall p' \in S\ p' \leq_P p\}$.*

Definition 6.35. *A poset is called locally recursive if it satisfies the following properties:*
(P1) The downclosure $c\downarrow$ of any element c is finite and computable.
(P2) The set $UC(c)$ of upper covers of an element c is finite and computable.
(P3) The set $lub(c, c') = \{p \mid c < p, c' < p, \neg\exists p'\ (c < p' < p \wedge c' < p' < p)\}$ is finite and computable for any choice of c, c'.
(P4) The set $min(\mathscr{P}) = \{p \mid \neg\exists p'\ p' < p\}$ is finite and computable.

Remark 6.36. *Each of the three graph orders considered in this thesis is a locally recursive poset. The meaning of 'computable' as used in the above definition requires the notion of a representation of the elements of $\mathscr{P}$ as strings similar to our representation UN for graphs.*

Definition 6.37 (Multiverse Normal Form)**.** *A universe $\mathscr{U}_{c,D}$ is the solution set $[\![\phi(x)]\!]$ of the quantifier free formula $\phi(x) = c \leq x \wedge \bigwedge_{d \in D} d \not\leq x$; where D is a finite set. We will write $g \in \mathscr{U}_{c,D}$ instead of $g \in [\![c \leq x \wedge \bigwedge_{d \in D} d \not\leq x]\!]$ for short. We can assume that $c < d_i$ for every d_i for any non-empty universe. A multiverse is a finite union of universes. We denote the the set of all multiverses by $\mathscr{M}$.*

A formula which is a disjunction of formulae of the form $c \leq x \wedge \bigwedge_{d \in D} d \not\leq x$ is said to be in Multiverse Normal Form (MNF).

A universe $\mathscr{U}_{c,D}$ is encoded by the tuple (c, D) and a multiverse is encoded by the collection of encodings of its constituent universes.

In the rest of this subsection, we will talk of recursive closure of multiverses under operations with the understanding that formally we mean the recursive closure of the encoding.

Lemma 6.38. *Multiverses of a locally recursive poset are recursively closed under boolean operations.*

Proof. The recursive closure of multiverses under union follows from definition; we take up recursive closure under intersection. Let $S = S_1 \cup S_2 ... \cup S_n$, $S' = S'_1 \cup S'_2 \cup ... S'_m$ be multiverses. It is sufficient to show that intersection $S_i \cap S'_j$ of atomic universes S_i and S'_j is a multiverse. If (c, D) and (c', D') are encodings for S_i, S'_j then $S_i \cap S'_j = [\![c \leq x \wedge c' \leq x \wedge \bigwedge_{d \in D \cup D'} d \not\leq x]\!] = \bigcup_{c'' \in lub(c,c')} [\![c'' \leq x \wedge \bigwedge_{d \in D \cup D'} d \not\leq x]\!]$, which is a multiverse since $lub(c, c')$ is a finite, computable set by property (P3) of locally recursive posets.

We take up recursive closure under complementation. For any $S = S_1 \cup S_2 \cup ... S_n$ where S_i are universes with encoding (c_i, D_i) respectively, $\mathscr{P} \setminus S = (\mathscr{P} \setminus S_1) \cap (\mathscr{P} \setminus S_2) \cap ... (\mathscr{P} \setminus S_n)$. The set $(\mathscr{P} \setminus S_i) = [\![c_i \not\leq x \vee \bigvee_{d \in D_i} d \leq x]\!]$ is a multiverse with encoding $\{(c', \{c_i\}) \mid c' \in min(\mathscr{P})\} \cup \{(d, \emptyset) \mid d \in D_i\}$, using property (P4). $\square$

Observation 6.39. *Given an encoding (c, D) of a universe, we can decide whether $\mathscr{U}_{c,D} = \emptyset$: this happens if and only if there exists $d \in D$ such that $d \leq c$; this implies that checking whether a given multiverse is $\emptyset$ is also decidable. By closure under complementation proved in the above lemma, checking if $\mathscr{U}_{c,D} = \mathscr{P}$ is also decidable.*

Lemma 6.40. *Let $(\mathscr{P}, \leq_P, \mathscr{C}_P)$ be a locally recursive poset. Any quantifier free formula $\phi(x)$ in one free variable in the vocabulary $\{\leq_P, \mathscr{C}_P\}$ is equivalent to a formula $\phi'(x)$ in MNF and ϕ' can be computed from ϕ.*

Proof. We can assume that the formula $\phi(x)$ is in Disjunctive Normal Form i.e. $\phi(x) = t_1(x) \vee t_2(x) \vee ... \vee t_n(x)$ where $t_i(x)$ is a conjunction of atomic formulae of the kind $c \square x$ for $\square \in \{\leq_P, \not\leq_P, \geq_P, \not\geq_P\}$. We observe that the formula $x = c$ (equality being definable using the partial order) is a universe : $c \leq_P x \wedge \bigwedge_{c <_P c'} c' \not\leq_P x$ by appeal to property (P2).

Consider a formula of the kind $x \leq_P c$. By property (P1) , there are only finitely many c' such that $c' \leq c$ and thus we can write an equivalent formula $\bigvee_{c' \leq_P c} x = c'$. Next consider a formula $x \not\leq_P c$; this is the complement of $x \leq_P c$ and thus is a multiverse by Lemma 6.38. We can thus convert the original formula into one where all atomic formulae can be replaced by multiverses and the proof follows from Lemma 6.38. □

Definition 6.41. *We define the following unary operations on the subsets of $\mathscr{P}$:*

1. *Strict downclosure of a set S, denoted by $S \Downarrow = \{p \mid \exists p' \in S, p <_P p'\}$.*
2. *Strict upclosure of a set S, denoted by $S \Uparrow = \{p \mid \exists p' \in S, p' <_P p\}$*
3. *Parallel closure of a set S, denoted by $S|| = \{p \mid \exists p' \in S, p' \not\leq_P p \wedge p \not\leq_P p'\}$.*

Lemma 6.42. *If the multiverses of a locally recursive poset $\mathscr{P}$ are recursively closed under $S \Downarrow$ and $S||$, then the FO^2 theory of $(\mathscr{P}, \leq_P, \mathscr{C}_P)$ is decidable.*

Proof. By Lemma 6.40, any quantifier free formula in one free variable is equivalent to a multiverse. Hence formulae of FO^2 in one free variable are generated by the grammar

$$\phi(x) := x \in S \mid \phi_1(x) \wedge \phi_2(x) \mid \neg\phi(x) \mid \exists y\ \phi(y)\ \wedge\ x \Box y$$

where $\Box \in \{<, >, ||, =\}$, and S is a multiverse with known encoding.

In the case of FO^2, we have three operations on sets $S \Downarrow, S \Uparrow, S||$ which correspond to the solution sets of the formulae of the form $\exists y\ \phi(y)\ \wedge\ x\Box y$ where $\Box$ is $<, >, ||$ respectively. By Lemma 6.38, we already have recursive closure of multiverses under boolean operations. Recursive closure under $S \Downarrow, S||$ are assumed by the hypothesis of the lemma, so it remains to show the recursive closure of multiverses under $S \Uparrow$.

Closure under $S \Uparrow$

The operator distributes over unions and thus it is sufficient to consider $S \Uparrow$ where S is a universe, say with encoding (c, D). Let $UC(c)$ be the set of upper covers of c. Then $S \Uparrow$ is represented by $\{(c', D') \mid c' \in UC(c), D' = \emptyset\}$. This is because $c \notin S \Uparrow$ (since there is no element less than c in S), but every element strictly more than c must be in $S \Uparrow$. Elements not in $c \Uparrow$ cannot be in $S \Uparrow$.

By an argument similar to that in Theorem 6.33 the FO^2 theory of $(\mathscr{P}, \leq_P, \mathscr{C}_P)$ is decidable. $\square$

6.3.2 Constants in FO^2

The results in this subsection give evidence that from the perspective of decidability, there isnt much of a difference between FO^2 with and without constants over the induced subgraph order. The definability of a graph g gives us immediate access to the filter rooted at g using the upclosure operator. By Observation 6.19, it is not possible to distinguish a graph from its complement in the absence of constants even in the full first order theory of the induced subgraph order.

The best one can hope for is the definability of every pair $\{g, g^c\}$ in $FO^2(\mathcal{G}, \leq_i)$. This leads to the natural question of what kind of finite sets are definable in FO^2, which can be seen as a weakening of the definability of constants. We examine this latter question in this subsection.

Lemma 6.43. *Given any finite set S definable in $FO^2(\mathcal{G}, \leq_i)$ and a number n, the layer $S_{=n}$ is definable. In particular, $\mathcal{G}_{=n}, \mathcal{G}_{\leq n}$ and $\mathcal{G}_n$ are definable for each $n \in \mathbb{N}$.*

Proof. We show that $\mathcal{G}_n$ is definable inductively. We already know that $\mathcal{G}_1 = \mathcal{G} \setminus \{\emptyset_g\}$ is definable using the formula $\exists y\ y <_i x$.

$$\mathcal{G}_{n+1}(x) := \exists y\ \mathcal{G}_n(y) \wedge y <_i x$$

By recursive closure under complementation of multiverses, we have the definability of $\mathcal{G}_{\leq n}$ and by closure under intersection, we get $\mathcal{G}_{=n}$. Finally, $S_{=n} = S \cap \mathcal{G}_{=n}$. $\square$

Lemma 6.44. *Let S be a definable set. Then $min(S), max(S)$ and $lub(S)$ are definable in $FO^2(\mathcal{G}, \leq_i)$.*

Proof. $min(S) = S \setminus (S \Uparrow)$, where the set difference operation $\setminus$ is a boolean operation. Similarly, $max(S) = S \setminus S \Downarrow$.

$S' = lub(S) \uparrow = \forall^{\Downarrow}(S)$; $lub(S) = min(S')$. We note here that if S is an infinite set, then $lub(S)$ is empty. □

Observation 6.45. *FO^2 can define each pair $\{g, g^c\}$ of graphs for $g, g^c \in \mathcal{G}_{\leq 4}$.*

Proof. $\{P_3, K_2N_1\} := lub(\mathcal{G}_{\leq 2})$.
$\mathcal{K} \cup \mathcal{N} := (\{P_3, K_2N_1\} \uparrow)^c$.
$\{N_i, K_i\} := (\mathcal{K} \cup \mathcal{N}) \cap \mathcal{G}_{=i}$.
The above result implies that for all graphs upto layer 2, we have proved that FO^2 without constants can define each pair $\{g, g^c\}$. Since the only elements at layer 3 are $\{N_3, K_3, P_3, K_2N_1\}$, we can take up layer 4 next.

$\mathcal{G}_{=4} = \{K_4, N_4\} \cup \{K_2K_2, C_4\} \cup \{P_4\} \cup \{Kite_4, K_2N_2\} \cup \{Paw_4, P_3N_1\} \cup \{S_4, K_3N_1\}$.
$\{K_4, N_4\}$ is definable as shown above.
$\{K_2K_2, C_4, P_4\}(x) := \mathcal{G}_{=4}(x) \wedge \forall y \in \{K_3, N_3\} y \not\leq_i x$
$P_4(x) := \{K_2K_2, C_4, P_4\}(x) \wedge \forall y \in \{P_3, K_2N_1\} y \leq_i x$
Removing P_4 from $\{K_2K_2, C_4, P_4\}$ we get $\{K_2K_2, C_4\}$.
$\{Paw_4, P_3N_1\}(x) := x \in (lub(\{P_3, K_2N_1\} \cap \mathcal{G}_{=4})$.
Removing all of the above defined elements from $\mathcal{G}_{=4}$ gives $S' = \{Kite_4, K_2N_2\} \cup \{S_4, K_3N_1\}$.
$\{Kite_4, K_2N_2\}(x) := S'(x) \wedge \exists y \in lub(\{K_3, N_3\})\ x \leq_i y$.
Removing $\{Kite_4, K_2N_2\}$ from S' gives us $\{S_4, K_3N_1\}$.
This concludes the proof of the observation. □

Remark 6.46. *Given an encoding (c, D) of a universe $\mathscr{U}_{c,D}$, the definability of each of the constants in the encoding implies the definability of the universe :*

$$\mathscr{U}_{c,D} = c\uparrow \setminus (\bigcup_{d\in D} d\uparrow).$$

Chapter 7

Future Work and Conclusion

In this thesis, we have studied the definability and decidability of FOL over structures of the kind $(\mathcal{G}, \tau)$ where τ includes a partial order. The work on definability forms the larger part of the thesis. Some preliminary results have been obtained on decidability, but there are a number of questions left open. We discuss the results of this thesis and avenues for further work on definability and decidability in graph order in the next two sections.

7.1 Definability in Graph Order

The results related to definability form Chapters 3, 4 and 5 of this thesis. Many of the graph theoretical relations required for the results in this thesis are already defined for the induced subgraph order by Wires; but definability in other graph orders has not been studied before. We will now take up each chapter and discuss the technical details of the results, problems that remain open and possible ways to tackle them.

In Chapter 3, we have shown that many small graphs can be defined in the first order theory of graph order. The availability of constants along with the power of atomic negation is crucial at this juncture to define many important graph families. For instance we are able to define the family of isolated points :

$$\mathcal{N}(x) := K_2 \not\leq x$$

In the above formula, $\not\leq$ can be any of subgraph, induced subgraph or minor. Thus there is a uniform way to define $\mathcal{N}$ in all three graph orders. This uniformity does not extend to other relations such as the predicate $maxDeg(x, n)$ which holds if and only if the maximum degree of x is n. This is because the maximum degree of a graph is one less than the cardinality of the maximum star subgraph, but this fact does not hold of the maximum star minor.

The basic graph predicates defined have been used to construct gadgets which enabled us to interpret arithmetic in graph order. These gadgets are trees and so it seems likely that the structure $(forest, \leq_s)$ which denotes forests under the subgraph order, can also interpret arithmetic. It is possible that with further restriction of the domain to either only fop (disjoint unions of paths) or only trees, we still have definability of arithmetic. In the case of fop, we no longer have access to the tree gadgets used and so new gadgets need to be constructed. In the case of trees, we no longer have definability of the predicate $|x| = |y|$ via the route taken in this thesis because $\mathcal{N}$ is not contained in trees. Thus in either case, more work is needed.

In Chapter 4, we have introduced the notion of a capable structure over graphs, which is one satisfying a set of three definability conditions: definability of arithmetic, definability of cardinality and definability of two graph relations related to o-presentations. An o-presentation $\tilde{g}$ of a graph g converts edge information contained in the latter to the subgraph information about the former by attaching large cycles of different cardinalities to the vertices of g. We have proved a theorem showing that any capable structure has the maximal definability property; in other words, a graph relation is definable in graph order iff it is arithmetical. An important corollary is the definability of every recursively enumerable predicate over graphs in theories of graph order. These two results have been presented modulo a particular encoding of graphs as strings (respectively a particular total ordering on graphs), with the understanding that any other encoding (respectively total order) in the same equivalence class with respect to Turing reducibility (respectively arithmetical definability) leads to the same notion of recursively enumerable predicate (respectively arithmetical relation) over graphs. We have also shown that the induced subgraph order is a capable structure, though this result is already implicit in the work of Wires [43] and many of the relations required to be defined are present explicitly in his work. The definition of a capable structure over graphs seems to have little to do with graphs and we hope that the results in this chapter can be generalized to other finite objects considered by Ježek and McKenzie [20], [23], [22], [21].

In Chapter 5, we have taken up the task of showing that the subgraph order is capable. The strategy employed to do so is the same as that for the induced subgraph order. The task is broken up into the definability of three relations out of which CP_4C and $\tilde{\mathcal{G}}$ are also used in the proof of capability of the induced subgraph order, but the third, namely $constructFromCycles$, is subtly different. The definability of CP_4C follows easily from the basic predicates defined in Chapter 3. We have shown that the definability of $\tilde{\mathcal{G}}$ and $constructFromCycles$ follows assuming predicates for the disjoint union of graphs and edge counting. The capability of the structure $(\mathcal{G}, \leq_s, disjointUnion, sameSize)$ has been shown and the rest of the chapter is dedicated to showing the definability of $disjointUnion$ and $sameSize$ in the subgraph order. We have noted that $(\mathcal{G}, \leq_m, sameSize)$ can define the subgraph order and leave the problem of defining $sameSize$ in the minor order open. We note that showing that the minor order is capable by direct use of o-presentations seems more difficult. The fact that the family $\mathcal{C}$ of cycles forms an infinite antichain under the subgraph and induced subgraph orders is what allows us to represent the vertex ordering of a graph via an o-presentation. In contrast, the celebrated Graph Minor Theorem [37] states that there are no infinite antichains under the minor order. There may well be other ways to represent vertex orderings of graphs via an alternative to o-presentations for the minor order, but we do not study this in this thesis.

Conjecture 7.1. *The minor order is a capable structure and thus has the maximal definability property.*

Showing that the minor order is capable would bring closure to the understanding of definability in the full first order theory of graph order. As regards definability in fragments, there is some evidence that it may be possible to strengthen Corollary 5.27 regarding the definability of r.e. predicates as follows.

Conjecture 7.2 (MRDP for Graphs)**.** *Every recursively enumerable predicate over graphs is definable using an existential formula in the structure* $(\mathcal{G}, \leq_i, \mathscr{C}_g)$*.*

The evidence mentioned is that arithmetic is existentially definable in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$ via the quantifier-free interpretation of the subword order in Theorem 6.11; the analogous result for the subword order has already been shown by Halfon et al [18]. In order to prove the above conjecture, one possible route involves proving two additional facts:

1. Arithmetic is universally definable in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$ i.e. it is $\Delta-$definable in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$.
2. The formula ψ_{enc} is existentially definable in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$.

The fact that this would suffice is clear from the form of the formula $\phi_R(\bar{x})$ for a relation R constructed in the proof of Theorem 4.8.

One variant of this approach involves changing our encoding function UN to one that may be more amenable to $\Delta-$definability of arithmetic in $(\mathcal{G}, \leq_i, \mathscr{C}_g)$. Another variant would be to show the $\Delta-$definability of arithmetic with the domain set used for interpretation being all of $\mathcal{G}$ and not just $\mathbb{N}$. This approach would rid us of the need to define encoding functions. This latter approach is closely connected to the Theory of Numberings introduced by Ershov [13].

We take up the discussion of decidability in graph order in the next section.

7.2 Decidability in Graph Order

Our results on decidability are contained in Chapter 6 and have concentrated on syntactic fragments of the induced subgraph order. In particular, we have shown that the existential theory of $(\mathcal{G}, \leq)$ is $\mathsf{NP}-$complete for each of the three graph orders but the existential theory of $(\mathcal{G}, \leq_i, \mathscr{C}_g)$ is undecidable. The decidability of the existential theory of $(\mathcal{G}, \leq_s, \mathscr{C}_g)$ and $(\mathcal{G}, \leq_m, \mathscr{C}_g)$ is left open.

Conjecture 7.3. *The existential theories of $(\mathcal{G}, \leq_s, \mathscr{C}_g)$ and $(\mathcal{G}, \leq_m, \mathscr{C}_g)$ are both undecidable.*

The other syntactic restriction we have studied is variable restriction, where we have shown that the three variable fragment of $(\mathcal{G}, \leq_i, \mathscr{C}_g)$ is undecidable while the corresponding single variable fragment is in $\mathsf{NP}^{\mathsf{NP}} \cap \mathsf{coNP}^{\mathsf{NP}}$. The decidability of the two variable fragment is left open. The FO^2 problem in the case of the subword order has been shown to be decidable by Karandikar and Schnoebelen and the strategies employed in [25] look promising.

Conjecture 7.4. *The two variable fragment of each of $(\mathcal{G}, \leq_i, \mathscr{C}_g)$,$(\mathcal{G}, \leq_s, \mathscr{C}_g)$ and $(\mathcal{G}, \leq_m, \mathscr{C}_g)$ is decidable.*

In particular, the above problem reduces to proving the recursive closure of multiverses under the operators $S \Downarrow$ and $S||$ for graph orders in view of Lemma 6.42. These questions are related to Ramsey Theory. Consider a universe $U \subseteq \mathcal{G}$ represented by $(\emptyset_g, D)$. Let $max(U)$ be the set of maximal elements of U under the order $\leq_i$. The set $U \Downarrow$ is equal to $U \setminus max(U)$ in this case. In case D contains some K_m and N_k, the Ramsey Theorem states that the entire set U is finite, and hence $U \Downarrow$ is also finite and is thus a multiverse. But when D does not contain any K_m, every clique belongs to U and hence U is infinite. However, it may still be the case that $max(U)$ is finite and this is not implied by the Ramsey Theorem. The finiteness of $max(U)$ immediately implies that $U \Downarrow$ is a multiverse by Lemma 6.38 which shows that multiverses are closed under boolean operations.

The results in Section 6.3.2 indicate that we can define many constants in FO^2 (upto automorphism) and it may in fact be the case that we can define all constants in FO^3. If so, there would be no difference in decidability between FO^3 over graph order with and without constants.

Conjecture 7.5. *The FO^3 fragment of each of the graph orders ($\mathcal{G}, \leq_i$),($\mathcal{G}, \leq_s$) and($\mathcal{G}, \leq_m$) is undecidable.*

This concludes our discussion of the syntactic restrictions. There are two other ways to place restrictions on the first order theory of graph order: domain restrictions and vocabulary restrictions.

Domain restrictions give rise to first order theories of structures of the form ($\mathcal{G}_0, \leq_i$) where $\mathcal{G}_0 \subseteq \mathcal{G}$. Recall from the previous section that arithmetical definability results are hard to extend to the case where $\mathcal{G}_0 = \mathcal{T}$. Even if it is not possible to interpret arithmetic in trees under the subtree order, the structure seems rich enough that we conjecture it has an undecidable theory.

Conjecture 7.6. *The first order theory of ($\mathcal{T}, \leq_i$) is undecidable.*

Changing the vocabulary from graph orders to other relations gives us structures whose theories can be seen as fragments of ($\mathcal{G}, \leq_s$) in view of our definability results because all natural relations over graphs that could be considered are recursive relations over graphs. For instance, one may consider the first order theory of the covering relation ($\mathcal{G}, \lessdot_i$). On the one hand, this theory seems too weak to be undecidable. On the other hand, open problems related to Graph Reconstruction [31] can be stated in it :

$$\forall x\, \forall y\, \forall z\, (z \lessdot_i x \iff z \lessdot_i y) \supset x = y$$

The above sentence states that every graph is characterized by the set of its one-vertex deleted subgraphs. Hence we do not expect a simple decision procedure for this theory, if one exists. We are less sure of the following conjecture:

Conjecture 7.7. *The first order theory of* $(\mathcal{G}, <_i)$ *is decidable.*

The decidability of structures over graphs which use different relations such as disjoint union, edge counting, connectivity etc. can be considered, but the choice of vocabulary is large and *apriori* it is not clear which of these would be the most fruitful. An understanding of how complexity classes such as the set of polynomial time predicates over graphs can be represented as a theory of some structure over graphs would greatly help this choice. In the case of graph orders, the quantifier-free formulae $g_1 \leq g_2$ when seen as computational problems are already $\mathsf{NP}-$complete and thus seem too coarse-grained for such a study by themselves. The exploration of graph relations of smaller complexity possibly qualifies as a long term project in itself.

Bibliography

[1] Dennis S Arnon. A bibliography of quantifier elimination for real closed fields. *Journal of symbolic computation*, 5(1-2):267–274, 1988.

[2] Jon Barwise. *Admissible sets and structures*, volume 7. Cambridge University Press, 2017.

[3] Alexis Bés. A survey of arithmetical definability. 2002.

[4] Egon Börger, Erich Grädel, and Yuri Gurevich. *The classical decision problem*. Springer Science & Business Media, 2001.

[5] Samuel R Buss. *Bounded Arithmetic, Bibliopolis, 1986. Revision of 1985 Princeton University Ph. D.* PhD thesis, thesis.

[6] Rina S Cohen and Janusz A Brzozowski. Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5(1):1–16, 1971.

[7] Hubert Comon and Ralf Treinen. Ordering constraints on trees. In *Colloquium on Trees in Algebra and Programming*, pages 1–14. Springer, 1994.

[8] Hubert Comon and Ralf Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theoretical Computer Science*, 176(1-2):67–87, 1997.

[9] Nachum Dershowitz. Orderings for term-rewriting systems. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 123–131. IEEE, 1979.

[10] Reinhard Diestel. *Graph theory.* Springer, 2005.

[11] Herbert Enderton. *A mathematical introduction to logic.* Academic press, 2001.

[12] David Eppstein. Finding large clique minors is hard. *J. Graph Algorithms Appl.*, 13(2):197–204, 2009.

[13] Yuri L Ershov. Theory of numberings. *Handbook of computability theory*, 140:473–506, 1999.

[14] Yuri Leonidovich Ershov. *Definability and computability.* Springer Science & Business Media, 1996.

[15] Leslie R Foulds. *Graph theory applications.* Springer Science & Business Media, 2012.

[16] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[17] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[18] Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *Logic in Computer Science (LICS), 2017 32nd Annual ACM/IEEE Symposium on*, pages 1–12. IEEE, 2017.

[19] Neil Immerman. *Descriptive complexity.* Springer Science & Business Media, 2012.

[20] Jaroslav Ježek and Ralph McKenzie. Definability in substructure orderings, i: finite semilattices. *Algebra universalis*, 61(1):59–75, 2009.

[21] Jaroslav Ježek and Ralph McKenzie. Definability in substructure orderings, iii: finite distributive lattices. *Algebra universalis*, 61(3-4):283–300, 2009.

[22] Jaroslav Ježek and Ralph McKenzie. Definability in substructure orderings, iv: finite lattices. *Algebra universalis*, 61(3-4):301–312, 2009.

[23] Jaroslav Ježek and Ralph McKenzie. Definability in substructure orderings, ii: finite ordered sets. *Order*, 27(2):115–145, 2010.

[24] Prateek Karandikar and Philippe Schnoebelen. Decidability in the logic of subsequences and supersequences. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 84–97, 2015.

[25] Prateek Karandikar and Philippe Schnoebelen. The Height of Piecewise-Testable Languages with Applications in Logical Complexity. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:22, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[26] Richard Kaye. *Models of Peano arithmetic*. Oxford University Press, USA, 1991.

[27] Oleg V Kudinov and Victor L Selivanov. A gandy theorem for abstract structures and applications to first-order definability. In *Conference on Computability in Europe*, pages 290–299. Springer, 2009.

[28] Oleg V Kudinov, Victor L Selivanov, and Lyudmila V Yartseva. Definability in the subword order. In *Programs, Proofs, Processes*, pages 246–255. Springer, 2010.

[29] Ádám Kunos. Definability in the embeddability ordering of finite directed graphs. *Order*, 32(1):117–133, 2015.

[30] Dietrich Kuske. Theories of orders on the set of words. *RAIRO-Theoretical Informatics and Applications*, 40(01):53–74, 2006.

[31] Josef Lauri and Raffaele Scapellato. *Topics in graph automorphisms and reconstruction*, volume 432. Cambridge University Press, 2016.

[32] Leonid Libkin. *Elements of finite model theory*. Springer Science & Business Media, 2013.

[33] Nadimpalli VR Mahadev and Uri N Peled. *Threshold graphs and related topics*, volume 56. Elsevier, 1995.

[34] Yuri Vladimirovich Matiyasevich. Diophantine representation of enumerable predicates. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 35(1):3–30, 1971.

[35] Willard V Quine. Concatenation as a basis for arithmetic. *The Journal of Symbolic Logic*, 11(4):105–114, 1946.

[36] R Ramanujam and RS Thinniyam. Definability in first order theories of graph orderings. In *Logical Foundations of Computer Science*, pages 331–348. Springer, 2016.

[37] Neil Robertson and Paul D Seymour. Graph minors. xx. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[38] Imre Simon. *Hierarchies of events with dot-depth one*. PhD thesis, Thesis (Ph. D.)–University of Waterloo, 1972.

[39] Denis Thérien. Imre simon: an exceptional graduate student. *RAIRO-Theoretical Informatics and Applications*, 39(1):297–304, 2005.

[40] Ramanathan S. Thinniyam. Defining recursive predicates in graph orders. *Logical Methods in Computer Science*, 14(3), 2018.

[41] Lou Van Den Dries. Alfred tarski's elimination theory for real closed fields. *The Journal of Symbolic Logic*, 53(01):7–19, 1988.

[42] KN Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM (JACM)*, 34(2):492–510, 1987.

[43] Alexander Wires. Definability in the substructure ordering of simple graphs. *Annals of Combinatorics*, 20(1):139–176, 2016.