Definability of Recursive Predicates in the Induced Subgraph Order

Ramanathan S. Thinniyam(✉)

The Institute of Mathematical Sciences,
CIT Campus, Taramani, Chennai 600113, India
thinniyam@imsc.res.in

Abstract. Consider the set of all finite simple graphs $\mathcal{G}$ ordered by the induced subgraph order $\leq_i$. Building on previous work by Wires [14] and Jezek and Mckenzie [5–8], we show that every recursive predicate over graphs is definable in the first order theory of $(\mathcal{G}, \leq_i, P_3)$ where P_3 is the path on 3 vertices.

1 Introduction

Finite graphs and graph theory have become of central importance with the advent of computer science since many computational problems can be modelled using them. Alongside this, the logical study of graphs has gained importance.

The "graph as a model" way of looking at graphs is the flourishing field of descriptive complexity, which has had success in creating logical objects equivalent to computational complexity classes. However, we will use a different way of looking at graphs. We will study the set of all isomorphism types of simple finite graphs (referred to as "graphs" from here on and denoted $\mathcal{G}$) with a single relation on $\mathcal{G}$, namely the induced subgraph relation (please see Fig. 1). This and other such relations such as the subgraph relation and the minor relation form interesting partial orders and their first order theory has been studied [13,14].

Note in particular that we do not have explicit access to the edge relation inside a particular graph, since we only have the single order relation as the vocabulary. Inspite of this, many graph families such as paths, cycles, cliques etc. and graph theoretical concepts such as connectivity, maximum degree etc. can be expressed in the first order theory of such objects, though in an indirect way. Thus we continue the exploration of the definablity and decidability in these first order theories (and their fragments).

Our work can be considered as continuing that of Jezek and Mckenzie [5–8], who studied the substructure orderings on various kinds of finite objects such as posets, lattices etc. This was later extended to the induced subgraph order by Wires [14]. The primary objective of these model theoretic studies is the determination of the automorphism group of these objects. On the other hand, our motivation is to explore the computational content of these objects.

To further this aim, we prove that the set of all recursive predicates is definable in the object $(\mathcal{G}, \leq_i, P_3)$ i.e. the induced subgraph order with a constant P_3

S. Ghosh and S. Prasad (Eds.): ICLA 2017, LNCS 10119, pp. 211–223, 2017.
DOI: 10.1007/978-3-662-54069-5_16

for the path on three vertices. The notion of recursive predicate we use is that of recognizability by a Turing machine of the encodings of graphs as numbers, for a fixed encoding that we define. We obtain the result by combining classical results on arithmetical definability and previous work by Jezek and McKenzie, and Wires.

Other work on orders on combinatorial objects includes that by Kunos [11] on the subdigraph order; and on word orders by Kuske [12].

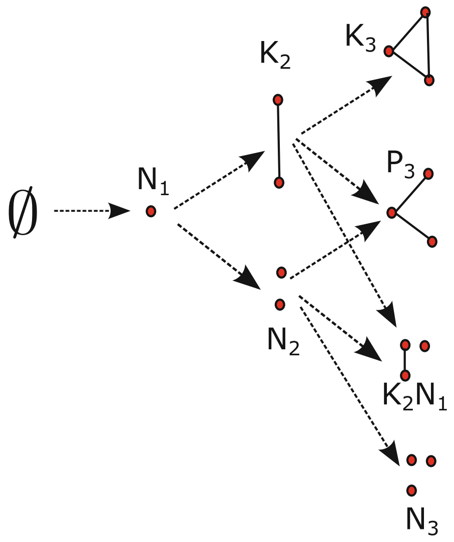

Fig. 1. The first few levels of the induced subgraph order. The arrows indicate the covering relation. $\emptyset$ is the empty graph.

While we are able to answer the question about definability of recursive predicates, our methods are too coarse to handle questions of definability of complexity classes (which are of course a strict subset of recursive predicates), say the set of all PTIME predicates over graphs. In addition, we do not take up the problem of precisely determining the logical resources required for the result. This paper is part of a preliminary investigation of the strength of such theories of combinatorial objects.

2 Preliminaries

First we give some definitions regarding graphs.

Definition 1 (Labelled Graph). *A (finite) labelled graph g is a structure (V_g, E_g, L_g) with*

1. finite domain (aka vertex) set V_g, *2. a symmetric binary relation $E_g \subseteq V_g \times V_g$ which is the edge set of the graph, and*

3. a bijective function $L_g : V_g \rightarrow [n]$ where $[n]$ stands for the initial segment $\{1, 2, 3..., n\}$ of the natural numbers with $n = |V_g|$ i.e. n is the number of vertices in the graph.

We will write v_i to denote the vertex whose image under L_g is i. We will write v_iv_j to denote the edge (if it exists) between v_i and v_j. In addition, we restrict ourselves to simple graphs i.e. graphs which dont have edges of the form (v_i, v_i) for some $v_i \in V_g$.

Definition 2. *An isomorphism between two labelled graphs g_1 and g_2 is a bijection $\eta : V_{g_1} \rightarrow V_{g_2}$ such that for any two vertices v_i, v_j of g_1, the edge v_iv_j exists if and only if there is an edge between vertices $\eta(v_i), \eta(v_j)$ in g_2.*

We say g_1 is isomorphic to g_2 if there is an isomorphism between them, and write $g_1 \simeq g_2$. The relation $\simeq$ is an equivalence relation on the set of all finite labelled graphs.

Definition 3 (Graph). *By a graph g, we mean an equivalence class under the relation $\simeq$ over the set of all finite labelled graphs. The set of all graphs will be denoted $\mathcal{G}$.*

We will write $g = [g']$ to denote that the graph g is the isomorphism type of the labelled graph g'.

All variables x, y, z occuring in formulae denote graphs and not labelled graphs. However, we will however need to talk about specific vertices or edges inside a graph and thus will require a labelling. So we will abuse notation and use u, v to talk of vertices of a graph (not a labelled one), uv for the edge joining u and v, and e to denote the edge of a graph. We denote graphs by g, h, and graph families by caligraphic letters such as $\mathcal{P}, \mathcal{C}$.

We will denote by N_i, K_i, C_i, S_i, P_i 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 Fig. 2); and by $\mathcal{N}, \mathcal{K}, \mathcal{C}, \mathcal{S}, \mathcal{P}$ the corresponding families of isolated vertices, cliques, cycles, stars and paths. We denote the cardinality (number of vertices) of a graph g by $|g|$, and the disjoint union of graphs g and h by $g \cup h$.

Next we need some definitions regarding the first order structures we study and definability in them.

For the standard syntax and semantics of first order logic, we refer the reader to Enderton [2].

Definition 4 (Induced Subgraph Order). *We consider the first order theory of the structure $(\mathcal{G}, \leq_i, P_3)$ where P_3 is a constant symbol for the path on three vertices and the $\leq_i$ is the induced subgraph order which is defined as: $g \leq_i g'$ iff g can be obtained from g' by deleting some (arbitrarily many) vertices of g'.*

The constant symbol 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 order.

Definition 5 (Arithmetic). *By arithmetic, we mean the first order theory of the structure* $(\mathbb{N}, \phi_+, \phi_\times)$ *where* $\mathbb{N}$ *is the set of all natural numbers and* $\phi_+, \phi_\times$ *are ternary predicates for addition and multiplication respectively.*

We will also use variables x, y, z to denote numbers in arithmetical formulae; and lower case letters k, l, m, n to denote numbers.

Definition 6 (Constant Definability). *Fix a first order language* $\mathcal{L}$*. Let* e *be an element of the domain of an* $\mathcal{L}$*-structure* $\mathcal{A}$*. We say that* e *is definable in* $\mathcal{A}$*, if there exists an* $\mathcal{L}$ *formula* $\alpha_e(x)$ *in one free variable, such that* $\mathcal{A}, e \vDash \alpha_e(x)$ *and for any* $e' \neq e$ *in the domain of* $\mathcal{A}$*,* $\mathcal{A}, e' \nvDash \alpha_e(x)$*.*

For any definable domain element e, we use e as a constant symbol representing the domain element because an equivalent formula can be written in the language $\mathcal{L}$ via use of the defining formula α_e.

Definition 7 (Covering Relation of a Poset). *Given elements* x, y *of a poset* $(P, \leq)$ *we define the covering relation* $x \lessdot y$ *as* $x \lessdot y$ *iff* $x < y$ *and there exists no element* z *of* P *such that* $x < z < y$*. This can easily written using a first order formula in the vocabulary of* $\{\leq\}$*.*

Definition 8 (Definability of Predicates). *We say a predicate is* definable in arithmetic *iff it is definable in* $(\mathbb{N}, \phi_+, \phi_\times)$ *and a predicate is* definable in graph theory *iff it is definable in* $(\mathcal{G}, \leq_i, P_3)$*.*

We use the symbol ϕ for arithmetical formulae and ψ for graph theory formulae to aid the reader.

Observation 1. *For any definable family* $\mathcal{F}$ *of* $(\mathcal{G}, \leq_i, P_3)$ *which forms a total order under* $\leq_i$*, every member of* $\mathcal{F}$ *is definable as a constant.*

To see this, first observe that there exists a minimum element f_1 in $\mathcal{F}$ by well foundedness of the order $\leq_i$.

$$f_1(x) := \mathcal{F}(x) \wedge (\forall y\, \mathcal{F}(y) \supset (y \leq_i x))$$

Assuming f_n (the n^{th} smallest element of $\mathcal{F}$) has been defined, f_{n+1} can be defined as the unique cover of f_n in $\mathcal{F}$.

Next we have the definitions we need to formalize the meaning of "recursive predicate over graphs." There exist notions of computability and recursive predicates over abstract structures (see [3]), but these are fairly technical. For our purposes, we use a fixed encoding of graphs as strings so that the standard notion of a computable predicate as one accepted by a Turing machine can be used. We encode graphs as numbers (equivalently binary strings). These encodings were originally introduced by us in previous work [13].

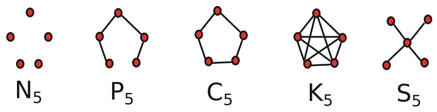

Fig. 2. Isolated points, path, cycle, clique and star of order 5 from left to right.

Definition 9 (Number Representation of a Graph). *A number representation of a graph g is defined using the following procedure.*

1. *Choose an labelled 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_i, v_j) of g with $j < i$. Let (v_i, v_j) and (v_k, v_l) belong to S (i.e. $j < i, l < k$). Then $(v_i, v_j) \leq_e (v_k, v_l)$ iff $i < k$ or $i = k, j < l$.*
2. *Arrange all the tuples belonging to S in descending order by $\leq_e$ to form the sequence seq.*
3. *Create the number m whose binary expansion is $\binom{n}{2} + 1$ bits long and has the following property: the most significant bit is 1 (always true for a number). The i^{th} most significant bit is 0 or 1 according to whether the $i - 1^{th}$ tuple in seq corresponds to a non-edge or edge (respectively) of the labelled graph g'.*

The number m is called a number representation of the graph g.

Definition 10 (Unique Number Representation). *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 $UN : \mathcal{G} \rightarrow \mathbb{N}$ is a one-one map. (See Fig. 3 for an example.)*

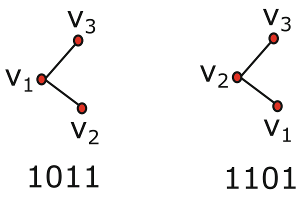

Fig. 3. Two different number representations of P_3 corresponding to two different labellings. The one on the left (i.e. 1011 in binary which is the number 11) is $UN(P_3)$.

Observation 2. *The representation UN induces an ordering on the vertices of the graph which comes from the underlying labelled graph.*

We can finally state what we mean by recursive predicates over graphs.

Definition 11. *We say a predicate $R \subseteq \mathcal{G}^n$ is recursive if there exists a Turing machine M such that*

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

i.e. the Turing machine accepts exactly the tuples of strings which correspond to UN encodings of tuples belonging to R.

3 Main Result

We note that the richness of a structure (for instance, its ability to interpret arithmetic) does not automatically imply the obtained result. Something more is required: the ability of the structure to perform operations on its elements, and in some sense, access its own internal structure in a way that is first order definable.

We will state the main result and show how the various modules come together to form the proof. Some of the details are postponed to make the presentation more understandable.

Theorem 1. *Every recursive predicate $R \subseteq \mathcal{G}^n$ on graphs is definable in $(\mathcal{G}, \leq_i, P_3)$.*

We need to show that for every recursive predicate $R \subseteq \mathcal{G}^n$ over graphs, there exists a formula $\psi_R(\bar{x})$ (where $|\bar{x}| = n$) in graph theory such that for any n-tuple of graphs $\bar{g}$,

$$R(\bar{g}) \iff (\mathcal{G}, \leq_i, P_3) \models \psi_R(\bar{g})$$

Since R is a recursive predicate, by Definition 11 there exists a machine M which accepts the UN number encodings of the set of graphs which belong to R.

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

The following is a classical theorem (see Appendix for a proof sketch):

Theorem 2. *Every recursive predicate R on numbers is definable in arithmetic.*

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

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

Next we recall that

Theorem 3 (Wires [14]). *Arithmetic i.e. $(\mathbb{N}, \phi_+, \phi_\times)$, is definable in graph theory i.e.,$(\mathcal{G}, \leq_i, P_3)$.*

In particular, the image set of the following map from numbers to graphs is definable:

$$UG : \mathcal{G} \to \mathbb{N}$$

UG takes a number n to the graph N_n which consists of n isolated points. There also exist defining formulae in graph theory for the predicates:

$$\psi_{(+)}(x,y,z) \text{ iff } ; x,y,z \in \mathcal{N} \text{ and } |x|+|y| = |z|.$$
$$\psi_{\times}(x,y,z) \text{ iff } ; x,y,z \in \mathcal{N} \text{ and } |x| \times |y| = |z|$$

We will write $|x|$ to denote the graph $N_{|x|}$ since there is a formula which defines the binary predicate $Norder(x,y)$ iff $|x| = |y|$ and $y \in \mathcal{N}$.

We will abuse notation by writing i instead of N_i and expressions such as $i+j, ij$ will be taken to mean the member of $\mathcal{N}$ such that its order equals $i+j, i \times j$ respectively. Similarly, since the order relation $<$ over the naturals is definable using addition, we will use quantifiers such as $\forall 1 < j < n$ in graph theory whose meaning is really $\forall j\, \mathcal{N}(j) \wedge N_1 \leq_i j \wedge j \leq_i N_n$.

Observation 3. *For every formula $\phi(\bar{x})$ in arithmetic there is a formula $\psi^t(\bar{x})$ in graph theory such that*

$$(\mathbb{N}, \phi_+, \phi_\times) \models \phi(\bar{n}) \iff (\mathcal{G}, \leq_i, P_3) \models \psi^t(UG(\bar{n}))$$

Applying this translation to the formula $\phi_{UN(R)}(\bar{n})$ gives us the graph formula $\psi^t_{UN(R)}(UG(\bar{n}))$.

Given a graph g, the above formula essentially states what we require but in terms of the graph $UG(UN(g))$. If there were a definable way to go between these two graphs inside the induced order, we could potentially "do the computation inside arithmetic and come back". This is essentially what we do to get the formula we require. In order to do this we require two things:

1. Acess to the edge relation inside arithmetic so that we can carry out the required computation inside arithmetic.
2. The ability to access the internal structure of a graph using the induced subgraph order.

The first of these has already been accomplished in previous work:

Theorem 4 ([13]). *The following predicates are definable in arithmetic:*

1. *$\phi_{UN}(x)$ iff x is a number which represents an isomorphism type of a graph as given in Definition 10.*
2. *$\phi_{edge}(x,i,j)$ iff x is a number representation of graph g_x and $v_i v_j \in E_{g_x}$.*
3. *$\phi_{length}(n,x)$ iff the length of the binary representation of x is n. We will just write $length(x)$ to denote n.*

Now we tackle the second problem i.e. that of accessing the internal structure of a graph. This is accomplished by using definable "vertex labelled representations" of graphs (which are themselves graphs), called *o-presentations*. This was first introduced by Jezek and Mckenzie, and defined for graphs by Wires.

Definition 12 (o-presentation). *An o-presentation of $g \in \mathcal{G}$ is another graph $\tilde{g}$ defined as follows: Fix an enumeration $v_1, v_1, .., v_n$ 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 denoted $\tilde{g}$.*

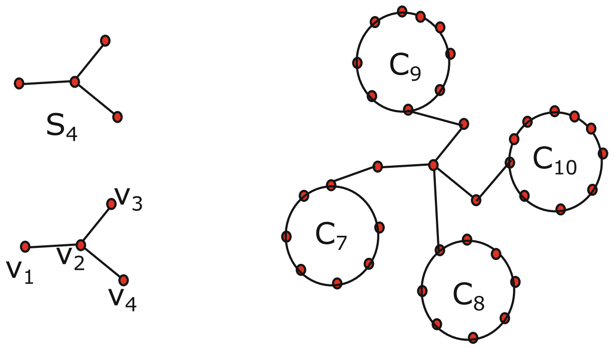

Fig. 4. Top left: the star graph S_4. Bottom left: a vertex labelling of S_4. Right: the o-presentation corresponding to the vertex labelling.

Given a graph g, an o-presentation can be regarded as the representation of a labelled graph g' with $g = [g']$, as another graph $\tilde{g}$. From the example in the Fig. 4, it is clear that there is a bijective correspondence between o-presentations and labellings of a graph.

The proof of the following lemma is deferred to the end of the section so as to not obstruct the flow of the main proof:

Lemma 1. *The following predicates are definable in* $(\mathcal{G} \leq_i, P_3)$*:*

1. *The set of all o-presentations, denoted* $\tilde{\mathcal{G}}$ *i.e.* $\tilde{\mathcal{G}}(x)$ *holds iff there is a graph* y *such that* x *is an o-presentation of* y.
2. *The predicate* $\psi_{opres}(x, y)$ *iff* y *is an o-presentation* $\tilde{x}$ *of* x*, also written* $y = \tilde{x}$ *for short.*
3. $\psi_{edgeOP}(x, i, j)$ *iff there exists a graph* y *such that* $y = \tilde{x}$ *and in the vertex labelling corresponding to the o-presentation, there is an edge between vertices* v_i *and* v_j *in the graph* y.

Using Lemma 1 and Theorem 4 we can now define the binary relation $n = UG(UN(x))$ by the formula $\psi_{enc}(x, n)$:

$$\begin{aligned}\psi_{enc}(x, n) := n \in \mathcal{N} \wedge \exists y\, y = \tilde{x} \wedge \psi^t_{graphOrder}(n, |x|) \\ \wedge\ \psi^t_{UN}(n) \wedge \forall 1 \leq i < j \leq |x| \\ \psi^t_{edge}(n, i, j) \iff \psi_{edgeOP}(y, i, j)\end{aligned}$$

The formula asserts that n is a trivial graph (has no edges) and there exists an o-presentation y of x such that there exists an edge between vertices v_i and v_j in the enumeration of the graph g corresponding to the o-presentation if and only if there is an edge between vertices v_i and v_j in an enumeration of the graph which is consistent with the UN representation.

By use of Lemma 1, we are able to write $y = \tilde{x}$. The formulae ψ^t_{edge} and $\psi^t_{UN}(n)$ are the translations of the formulae from Theorem 4 by using

Observation 3. $\psi^t_{graphOrder}$ is the translation of the following arithmetic formula

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

Note that the arguments in the translated formulae have to be members of the family $\mathcal{N}$ and are applied to the image graph under the map UG while the arithmetic formulae are on the numbers obtained from the inverse of this map. Also note the use of $\forall 1 \leq i < j \leq |x|$ which is syntactic sugar for a more involved formula we can write in graph theory due to definability of arithmetic (recall remarks under Theorem 3). We can now define R in the induced subgraph order:

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

ψ_R essentially inverts the encoding function UN to go back from the number encodings to the graphs.

All that remains to be done is the proof of Lemma 1. In order to do so, we need some machinery:

Lemma 2 (Wires [14]). *The following predicates are definable in* $(\mathcal{G}, \leq_i, P_3)$.

1. *The families* $\mathcal{N}, \mathcal{K}, \mathcal{C}, \mathcal{P}$ *standing for trivial graphs, complete graphs (cliques), cycles and paths respectively.*
2. $|x| = |y|$ *iff* x *and* y *have the same cardinality (i.e. same number of vertices, also known as order of the graph).*
3. $maxComp(x, y)$ *iff* x *is a maximal connected component of* y.
4. $cover(x, y, n)$ *iff there are exactly* $n-1$ *graphs between* x *and* y *in the order and* $x \leq_i y$. *Also denoted* $x \lessdot^n_i y$.

$$cover(x, y, n) := |x| + n = |y|$$

   The order of the graph defines a layering of the induced subgraph order.

5. $z = x \cup y$ *iff* z *is the disjoint union of* x *and* y.
6. $C_{\to 1}(x)$ *iff* x *is the connected graph formed by adding one extra vertex and one extra edge to a cycle.*
7. $conn(x)$ *iff* x *is a connected graph.*
8. $C_{\to 2}(x)$ *iff* x *is graph formed by taking a graph* g *with* $C_{\to 1}(g)$ *and adding an additional vertex and joining it to the unique degree 1 vertex in* g.
9. $pointedCycleSum(x, y, z)$ *iff* 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.*

Notice that from the definability of $C_{\to 1}(x)$ we also have definability of the graph $C_{i \to 1}$ which stands for the member of $C_{\to 1}$ of order $i+1$ because the family is totally ordered by number of vertices and for similar reasons as Observation 1. Additionally, given a parameter n, we can obtain $C_{n \to 1}$.

We are now ready to give a proof of Lemma 1:

Proof of Lemma 1. We recollect the statement. The following are definable in graphs:

1. The set of all o-presentations, denoted $\tilde{\mathcal{G}}$ i.e. $\tilde{\mathcal{G}}(x)$ holds iff there is a graph y such that x is an o-presentation of y.
2. The predicate $\psi_{opres}(x,y)$ iff y is an o-presentation $\tilde{x}$ of x, also written $y = \tilde{x}$ for short.
3. $\psi_{edgeOP}(x,i,j)$ iff there exists a graph y such that $y = \tilde{x}$ and in the vertex labelling corresponding to the o-presentation, there is an edge between vertices v_i and v_j in the graph y.

Proof. We take up the definition of the family $\tilde{\mathcal{G}}$. First we note that given a number n, we can construct the object $\bigcup_{i=1}^{n} C_{n+i+2}$ as follows:

$$csum(n,x) \text{ iff } n \in \mathcal{N} \text{ and } x = \bigcup_{i=1}^{n} C_{n+i+2}.$$

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

The formula constrains every maximal component to be a cycle using the *maxComp* predicate. This forces all the cycles to be disjoint. Enforcing the cardinality condition and the fact that each cycle has to be present (*allCycles*) makes sure that the graph is made up of exactly one copy of each cycle and nothing else.

Now we can define the set of o-presentations as follows:

$$\begin{aligned} \tilde{\mathcal{G}}(x) :=& \exists n\, cardCond(n,x) \wedge hasC1s(n,x) \\ & \wedge\ hasUnionOfCycles(n,x) \\ & \text{where} \\ cardCond(n,x) :=& \mathcal{N}(n) \wedge |x| = n^2 + n(n+1)/2 + 3n \\ hasC1s(n,x) :=& \forall i\, (1 \le i \le n) \supset C_{i+n+2 \to 1} \le_i x \\ hasUnionOfCycles(n,x) :=& \bigcup_{i=1}^{n} C_{n+i+2} \le_i x \end{aligned}$$

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. *hasCycles* states that the $C_{\to 1}$ are induced subgraphs. *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. In addition, there are no chord or edges between the cycles. No restriction is place on the edges between the non-cycle vertices. Thus the resulting graph x is of the required form.

We take up the second predicate, $\psi_{opres}(x, y)$ iff y is an o-presentation of x.

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

The object $\bigcup_{i=1}^{n} P_{n+i+1}$ can be constructed given n by taking the appropriate $\bigcup_{i=1}^{n} C_{n+i+2}$, deleting n vertices from it, and enforcing the condition that no cycles are present.

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 $2|x| + 1$. The only way to get an o-presentation by adding $|x|$ vertices to z is to add two edges between every new vertex and and ends of one of the paths and one edge from the new vertex to a vertex in x. Thus any such y must be an o-presentation of x.

Moving on to the last predicate $\psi_{edgeOP}(x, i, j)$, we first need the following intermediate predicate:

$CP_4C(x, i, j)$ iff $i, j \in \mathcal{N}, 3 < i < j$ and x is formed by adding to the graph $C_i \cup C_j$ two additional vertices v_1, v_2 and the edge v_1v_2, one edge between C_i and v_1 and one edge between C_j and v_2. We denote x by $CP_4C(i, j)$.

$$\begin{aligned}CP_4C(x, i, j) := {}& conn(x) \wedge \mathcal{N}(i) \wedge \mathcal{N}(j) \wedge 3 < i < j \\ & \wedge C_i +_p C_j \not\leq_i x \\ & \wedge C_{i\to 1} \cup C_j \lessdot_i x \wedge C_{j\to 1} \cup C_i \lessdot_i x\end{aligned}$$

From the definition, x has to be obtained by adding one new vertex v to $g_0 = C_{i\to 1} \cup C_j$ and some number of edges which are incident on v. Notice that there is only one copy of C_j present as subgraph in x because of cardinality constraints. Thus there is exactly one edge between v and C_j (connectivity constraint). If there were multiple edges, we cannot get $C_{j\to 1}$ as induced subgraph. Now suppose there is also exactly one edge from v to copy of C_i in g_0, then we can get $C_i +_p C_j$ as induced subgraph, which is not allowed. Suppose there are multiple edges between v and copy of C_i in g_0, then we cannot get $C_{j\to 1} \cup C_i$ as induced subgraph from x by deleting a single vertex (since v remains connected to the rest of the graph not considering C_j on deleting only one vertex). But given the connectivity constraint, there must be an edge from v to the dangling vertex of $C_{i\to 1}$ inside g_0. Thus the graph we get is the required graph.

We can now write

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

The existence of an edge between vertices v_i and v_j in the graph x is captured by the presense 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}. $\square$

This concludes the proof of Lemma 1 and thus completes the proof of Theorem 1.

4 Discussion

Our result leads to a number of interesting questions and potential areas for research.

There has been considerable work in the area of bounded arithmetic systems and their connection to complexity theory [1,10]. An intimate connection has been shown between propositional proof systems, systems of bounded arithmetic and complexity theory. Characterizing complexity classes of graph problems using fragments of the induced subgraph order may prove useful.

The way we have arrived at our result is very roundabout in the sense that we dont use any "natural computational predicates" over graphs. There may be such predicates over graphs which are the equivalent of the *bit* predicate and exponentiation in arithmetic. It is by carefully controlling these two (and further expanding the language) that the bounded arithmetic theories were discovered. In addition, we note that the method of computation we use essentially puts a total order on the vertices of the graph (via the o-presentation). This is closely related to the question of "order-invariant querying" which is of much interest in finite model theory and descriptive complexity [4].

There are related objects such as the subgraph order and the graph minor order whose expressive power is enough to interpret arithmetic [13] but it is not clear if o-presentations can be defined in them. On the other hand we do not have the tools to tackle the problem of proving inexpressibility in such rich structures. It would be interesting (though doubtful) to see if there are any general methods to generate o-presentations in different types of structures.

Acknowledgment. I would like to thank my guide Prof. R. Ramanujam for his advice and discussions on both technical matter and the presentation of this paper.

Appendix: Proof Sketch of Theorem 2

Theorem Statement: Every recursive predicate R on numbers is definable in first order arithmetic.

Proof (sketch). For simplicity we look at the case of only unary predicates, assume $R \subseteq \mathbb{N}$. Let $M = (Q, \delta, s, F)$ be a turing machine over the alphabet $\{0, 1\}$. First, consider strings over the alphabet $\Sigma = (0, 1, \#, s, q_1, ..., q_n)$ where $Q = \{s, q_1, ..., q_n\}$. They can be encoded as binary strings by using some encoding e.g. 0 is mapped to 01, 1 to 001, # to 0001, s to 00001, q_i to $0^{i+4}1$. Given any input x, we can encode the run of the Turing machine as a number y, which we will think of a string over the extended alphabet Σ (ignoring the 1 in the most significant digit). $y = c_1\#c_2\#...\#c_m$ where each c_i is a string containing exactly one state symbol and remaining 0's and 1's. The placement of the head of the machine is given by the position just after the state symbol. c_1 is sx i.e. the starting state s concatenated with the input x, c_m is a configuration containing a final state and the relationship between any two consecutive configurations is restricted based on the transition function δ. All of this can be written as a formula $\phi_R(x)$ which essentially states "there exists a number y such that the binary encoding of the number represents an accepting run of the machine on x", making crucial use of the *bit* predicate and exponentiation. For details on definability in arithmetic, please see Kaye [9]. □

References

1. Cook, S., Nguyen, P.: Logical Foundations of Proof Complexity. Cambridge University Press, Cambridge (2010)
2. Enderton, H.: A Mathematical Introduction to logic. Academic Press, Burlington (2001)
3. Fitting, M.: Fundamentals of Generalized Recursion Theory. Elsevier, Amsterdam (2011)
4. Grohe, M.: The quest for a logic capturing PTIME. In: 23rd Annual IEEE Symposium on Logic in Computer Science, LICS 2008, pp. 267–271. IEEE (2008)
5. Ježek, J., McKenzie, R.: Definability in substructure orderings, IV: finite lattices. Algebra Univers. **61**(3–4), 301–312 (2009)
6. Ježek, J., McKenzie, R.: Definability in substructure orderings, I: finite semilattices. Algebra Univers. **61**(1), 59–75 (2009)
7. Ježek, J., McKenzie, R.: Definability in substructure orderings, III: finite distributive lattices. Algebra Univers. **61**(3–4), 283–300 (2009)
8. Ježek, J., McKenzie, R.: Definability in substructure orderings, II: finite ordered sets. Order **27**(2), 115–145 (2010)
9. Kaye, R.: Models of Peano arithmetic. Oxford University Press, Oxford (1991)
10. Krajicek, J.: Bounded Arithmetic, Propositional Logic and Complexity Theory. Cambridge University Press, Cambridge (1995)
11. Kunos, Á.: Definability in the embeddability ordering of finite directed graphs. Order **32**(1), 117–133 (2015)
12. Kuske, D.: Theories of orders on the set of words. RAIRO Theor. Inform. Appl. **40**(01), 53–74 (2006)
13. Ramanujam, R., Thinniyam, R.S.: Definability in first order theories of graph orderings. In: Artemov, S., Nerode, A. (eds.) LFCS 2016. LNCS, vol. 9537, pp. 331–348. Springer, Heidelberg (2016). doi:10.1007/978-3-319-27683-0_23
14. Wires, A.: Definability in the substructure ordering of simple graphs. Ann. Comb. **20**(1), 139–176 (2016)