Definability of Recursive Predicates in the Induced Subgraph Order

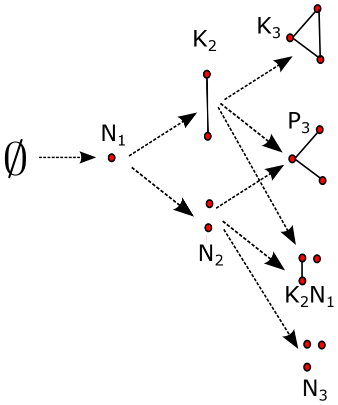
Ramanathan S. Thinniyam

Institute of Mathematical Sciences, Chennai

11 October, 2016

Graph Orders

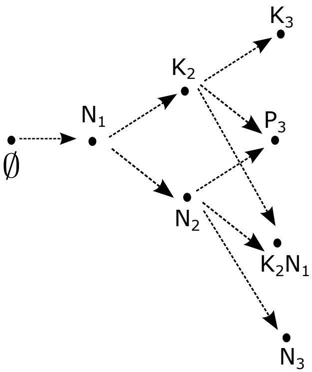
- $\mathcal{G}$ is the set of all isomorphism types of simple finite graphs.
- For $g, g' \in \mathcal{G}$, $g \leq g'$ iff g is an induced subgraph of g'.
- Other orders such as subgraph and minor can also be studied.



Objective

- Study logical theories of such objects.
- In this talk we will concentrate on the induced subgraph order with an additional constant P_3 for the path on three vertices : $(\mathcal{G}, \leq, P_3)$.
- In particular, definability of predicates and decidability of fragments.
- Different as compared to study of graphs as done in finite model theory (graph as a model).

Induced Subgraph as FO Structure



Constants No (direct) access to the edge relation E i.e. the internal structure of a graph.

Literature on Combinatorial Theories of Order

- Series of papers by Jezek and McKenzie "Definability in Substructure Orderings" I to IV (2009,2010) studies substructure order over finite posets, semilattices, lattices, distributive lattices. Emphasis on constant definability.
- Extended by Wires (2012) to induced subgraph. $(\mathcal{G}, \leq, P_3)$ interprets arithmetic (predicate version $(\mathbb{N}, \phi_+, \phi_\times)$).
- Word orders such as subword, infix, lpo studied by Kuske (2006). Emphasis in Kuske's work on decidability.
- Previous work by R. Ramanujam and R.T. on mutual interpretability of induced subgraph, subgraph and minor orders with arithmetic.

Definability in the Induced Subgraph Order : Wires's Work

- Wires proves that various graph families (such as cycles, paths, stars denoted $\mathcal{C}, \mathcal{P}, \mathcal{S}$ respecively) and important graph theoretical predicates such as connectivity, order of a graph etc. can be defined.
- Emphasis in Wires' paper on constants definability and finding set of all automorphisms.
- Our concern : computational content of these objects.

Main Result

Theorem

The set of all recursive predicates over graphs is definable in $(\mathcal{G}, \leq, P_3)$. □

The result is obtained by combining multiple modules which deal with definability in arithmetic and graphs.

Example of Definability I

Lemma (Wires)

The family $\{K_n : 1 \leq n\} \cup \{N_n : 1 \leq n\}$ *comprised of all cliques and isolated points is definable.*

$$KN(x) := \neg(\exists y\ \exists z\ y \neq z\ \wedge\ y \not\leq z\ \wedge\ z \not\leq y\ \wedge\ y < x\ \wedge\ z < x)$$

Above formula says "Downclosure of x under $\leq$ is a chain".
Clearly the family satisfies the property.

Example of Definability I

For the reverse direction, consider any graph g not of the family. There are vertices u, v, x, y in g such that $|\{u, v, x, y\}| \geq 3$ and $\neg Exy$ and Euv.
Thus both K_2 and N_2 are induced subgraphs of g but these are incomparable graphs.

Example of Definability II

Lemma (Wires)

All graphs of cardinality at most 4 are definable as constants.

First define the covering relation:

$$x \lessdot y := x \leq y \ \wedge \ \forall z \, \neg(x < z < y)$$

By repeated use of the covering relation we can define for every fixed $k > 0$, the relation $x \lessdot^k y$ iff there are exactly k graphs between x and y.

$$\emptyset(x) := \forall y \; x \leq y \quad N_1(x) := \emptyset \lessdot x$$

▸ InducedSubgraph

Example of Definability II

$$\{K_3, N_3\}(x) := KN(x) \ \wedge \ \emptyset \lessdot^2 x$$

$$\{P_3, K_2 N_1\}(x) := \neg KN(x) \ \wedge \ \emptyset \lessdot^2 x$$

Since we already have P_3, we can get $K_2 N_1$.

Recursive Predicates on Graphs

- To talk about graph properties accepted by Turing machines, we need to encode graphs as strings.
- We will use a specific encoding of graphs as numbers (equivalently, binary strings) for our purposes, which we call *UN* (unique number).
- $UN : \mathcal{G} \to \mathbb{N}$ is a 1-1 map which fixes a vertex ordering of the graph.

Definition

A predicate $R \subseteq \mathcal{G}^n$ (for some n) is said to be recursive if there is a Turing machine M such that $L(M) = UN(R)$.

Detailed Statement of Main Result

We first restate the main result : For every recursive predicate $R \subseteq \mathcal{G}^n$, there is a formula $\phi_{R,\mathcal{G}}(\bar{x})$ in the language of graphs such that for any $n - tuple$ $\bar{g}$ of graphs,

$$(\mathcal{G}, \leq, P_3) \models \phi_{R,\mathcal{G}}(\bar{g}) \iff \bar{g} \in R$$

Important Remarks

- Mutual interpretability with arithmetic does not automatically give the result i.e. definability of recursive predicates.
- A key ingredient required is the ability to access the internal structure of a graph in order to do computation on it.
- Builds on the work by Jezek and McKenzie and by Wires.

Proof Sketch : Definability of Recursive Predicates in Arithmetic

Given recursive predicate $R \subseteq \mathcal{G}^n$, the definition gives us a Turing machine M which recognises $UN(R)$. By a classical theorem, there is a formula $\phi_{UN(R),\mathbb{N}}(\bar{x})$ in the language of numbers (i.e. using predicates ϕ_+ and $\phi_\times$) such that

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

Proof Sketch : Definability of Arithmetic in Graphs

Theorem (Wires)

Consider the map $UG : \mathbb{N} \to \mathcal{G}$ which sends every number n to the graph N_n made of n isolated points. We denote the image of a tuple $\bar{n}$ of numbers under this map by $UG(\bar{n})$.
$UG(\mathbb{N})$ is a definable family in the induced order.
There are formulae in $\phi_{\mathcal{G}(+)}(x, y, z)$ and $\phi_{\mathcal{G}(\times)}(x, y, z)$ over graphs such that for any three tuple of numbers (n_1, n_2, n_3),

$$(\mathbb{N}, \phi_+, \phi_\times) \models \phi_+(n_1, n_2, n_3)$$
$$\iff$$
$$(\mathcal{G}, \leq, P_3) \models \phi_{\mathcal{G}(+)}(UG(n_1), UG(n_2), UG(n_3))$$

Similarly for $\phi_\times(x, y, z)$. □

Proof Sketch : Translation of Arithmetical Formulae into Graph Theory

Corollary: For every arithmetical formula $\phi_{\mathbb{N}}(\bar{x})$ there is a graph formula $\phi_{\mathcal{G}(\mathbb{N})}(\bar{x})$ such that for

$$(\mathbb{N}, \phi_+, \phi_\times) \models \phi_{\mathbb{N}}(\bar{n})$$

$$\iff$$

$$(\mathcal{G}, \leq, P_3) \models \phi_{\mathcal{G}(\mathbb{N})}(UG(\bar{n}))$$

Proof Sketch : Applying the Translation

Applying the above translation to $\phi_{UN(R),\mathbb{N}}(\bar{x})$, we get $\phi_{\mathcal{G}(UN(R),\mathbb{N})}(\bar{x}))$ in the language of graphs.

Given a graph g, suppose we are able to obtain the graph $UG(UN(g))$ (and vice versa) in a definable way inside graph theory, we can do the computation inside arithmetic and come back.

To do this, we need

1. Definable "vertex labelled representations" of graphs (as other graphs) called o-presentations (Jezek and McKenzie, Wires).
2. Access to the edge relation of a graph (represented as a number) inside arithmetic.

Proof Sketch : O-presentations

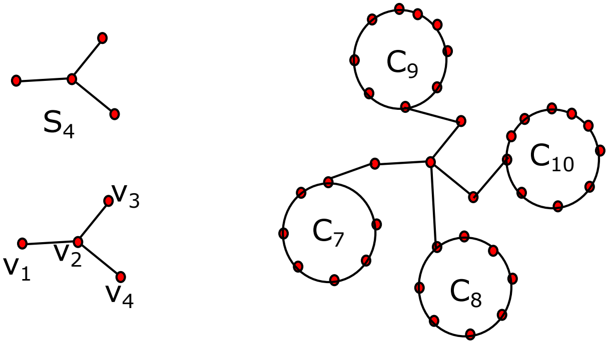


Figure: Top left : the graph S_4. Bottom left: a vertex labelling of S_4. Right: o-presentation of S_4 corresponding to the given vertex labelling.

Proof Sketch: Defining O-presentations in Graphs

Definition

For graphs g, g' we write $g' = \tilde{g}$ iff g' is an o-presentation of g.
The set of all graphs which are o-presentation is denoted $\tilde{\mathcal{G}}$.

Theorem

The following predicates are definable in graphs:

1. *The set of all o-presentations $\tilde{\mathcal{G}}$*
2. *The relation $x = \tilde{y}$ relating a graph and one of its o-presentations.*
3. *The predicate $edgeExistsOP(x, i, j)$ iff there is a graph y with $x = \tilde{y}$ and there is an edge between the vertices v_i and v_j as assigned by the o-presentation.*

Proof Sketch: Edge Relation in Arithmetic

Theorem

The following predicates are definable in arithmetic:

1. $\phi_{UN}(x)$ *iff* x *is a number representing a graph in the chosen encoding.*
2. $\phi_{edgeExists}(x, i, j)$ *iff* $\phi_{UN}(x)$ *holds and there is an edge between vertex* v_i *and vertex* v_j *in the graph represented by* x*.*
3. $\phi_{graphOrder}(n, m)$ *iff the length of the binary representation of* n *is equal to* $1 + \binom{m}{2}$*.*

Proof Sketch: Putting it Together

Theorem

The predicate $\phi_{enc}(x,n)$ iff $n = UG(UN(x))$, is definable in graphs.

$$\begin{aligned}\phi_{enc}(x,n) := n \in \mathcal{N} \;\wedge\; \exists y\; y = \tilde{x} \;\wedge\; \phi_{\mathcal{G}(graphOrder)}(n,|x|) \;\wedge \\ \phi_{\mathcal{G}(UN)}(n) \;\wedge\; \forall 1 \le i < j \le |x| \\ \phi_{\mathcal{G}(edgeExists)}(n,i,j) \iff edgeExistsOP(y,i,j)\end{aligned}$$

□

We now finally get the desired formula for the predicate R:

$$\phi_{R,\mathcal{G}}(\bar{x}) := \exists \bar{y} \bigwedge_{i=1}^{n} \phi_{enc}(x_i, y_i) \;\wedge\; \phi_{\mathcal{G}(UN(R),\mathbb{N})}(\bar{y})$$

Future Directions

- Try to replicate the proof for other graph orders.
- Decidable fragments : Syntactic fragments such as $\exists^*\forall^*$, graph classes such as bounded vertex cover graphs, theory of the covering relation $\mathrm{Th}(\mathcal{G}, \lessdot)$
- Come up with natural computational predicates over graphs (like *bit* in arithmetic) which can be used to produce a simpler proof.
- Characterize computational complexity classes such as PTIME as a fragment of this (or other) theory.

THANK YOU