Graphs and Monadic Second Order Logic

Ritam M Mitra

April 2024

1 Introduction

1.1 Graphs

A graph G is a pair $(V(G), E(G))$, where $V(G)$ is a finite set whose elements are called vertices and $E(G) \subseteq \binom{V(G)}{2}$ is a set of unordered pairs of vertices, which are called edges. Hence graphs in this paper are always finite, undirected, and simple, where simple means that there are no loops or parallel edges. If $e = \{u, v\}$ is an edge, we say that the vertices u and v are adjacent, and that both u and v are incident with e. A graph H is a subgraph of a graph G (we write $H \subseteq G$) if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If $E(H) = E(G) \cap \binom{V(G)}{2}$, then H is an induced subgraph of G.

Occasionally, we consider (vertex) labelled graphs. A labelled graph is a tuple $G = (V(G), E(G), P_1(G), ..., P_l(G))$, where $P_i(G) \subseteq V(G)$ for all $i \in [l]$. The symbols P_i are called labels, and if $v \in P_i(G)$ we say that v is labelled by P_i. Subgraphs, union, and intersection extend to labelled graphs in a straightforward manner. The underlying graph of a labelled graph G is $(V(G), E(G))$.

G denotes the class of all graphs. For every class C of graphs, we let C_{lb} be the class of all labelled graphs whose underlying graph is in C. A **graph invariant** is a mapping defined on the class G of all graphs invariant under isomorphisms. All graph invariants considered in this paper are integer valued. For a graph invariant $f : G \to Z$ and a class C of graphs, we say that C **has bounded** f if there is a $k \in Z$ such that $f(G) \leq k$ for all $G \in C$.

1.2 Logic

Let us briefly review the syntax and semantics of **first-order logic FO** and **monadic second-order logic MSO**. We assume that we have an infinite supply of individual variables, usually denoted by the lowercase letters x, y, z, and an infinite supply of set variables, usually denoted by uppercase letters X, Y, Z. First-order formulas in the language of graphs are built up from atomic formulas $E(x, y)$ and $x = y$ by using the usual Boolean connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication), and $\leftrightarrow$ (bi-implication) and existential quantification $\exists x$ and universal quantification $\forall x$ over individual variables. Individual variables range over vertices of a graph. The atomic formula

$E(x, y)$ expresses adjacency, and the formula $x = y$ expresses equality. From this, the semantics of first-order logic is defined in the obvious way.

First order formulas over labelled graphs may contain additional atomic formulas $P_i(x)$, meaning that x is labelled by P_i. If a label P_i does not appear in a labelled graph G, then we always interpret $P_i(G)$ as the empty set. In monadic second-order formulas, we have additional atomic formulas $X(x)$ for set variables X and individual variables x, and we admit existential and universal quantification over set variables. Set variables are interpreted by sets of vertices, and the atomic formula $X(x)$ means that the vertex x is contained in the set X.

The free individual and set variables of a formula are defined in the usual way. A sentence is a formula without free variables. We write $\varphi(x_1, ..., x_k, X_1, ..., X_l)$ to indicate that φ is a formula with free variables among $x_1, ..., x_k, X_1, ..., X_l$. We use this notation to conveniently denote substitutions and assignments to the variables. If $G = (V, E)$ is a graph, $v_1, ..., v_k \in V$, and $W_1, ..., W_l \subseteq V$, then we write $G \models \varphi(v_1, ..., v_k, W_1, ..., W_l)$ to denote that $\varphi(x_1, ..., x_k, X_1, ..., X_l)$ holds in G if the variables x_i are interpreted by the vertices v_i and the variables X_i are interpreted by the vertex sets W_i.

To maintain some of the expressiveness of second order logic, while retaining the ease of decidability of first order logic, we introduce monadic second order logic. In monadic second order logic, we allow for quantification over variables and sets of variables. Monadic here refers to the ability to quantify over predicates of a single variable (monadic predicates), which is exactly equivalent to quantifying over sets. This added expressiveness allows us to express properties we couldn't in first order logic. For instance, k-colorability can be stated in monadic second order logic (here we show 2-colorability):

$$\exists X \forall x \forall y [\mathbf{adj}(x, y) \Rightarrow (x \in X \wedge y \notin X) \vee (x \notin X \wedge y \in X)]$$

adj can be expressed in terms of **edg**:

$$\mathbf{adj}_G(x, y) \iff \exists e[\mathbf{edg}_G(e, x, y)]$$

Another predicate $\mathbf{Card}_{p,n}$, given by

$$\mathbf{Card}_{p,n}(X) \iff |X| \equiv p (\text{mod } n)$$

1.2.1 Dominating Set of G

A dominating set in a graph $G = (V, E)$ is a set $S \subseteq V$ such that for every $v \in V$, either v is in S or v is adjacent to a vertex in S. The following first-order sentence dom_k says that a graph has a dominating set of size k:

$$dom_k = \exists x_1 ... \exists x_k (\bigwedge_{1 \leq i \leq j \leq k} x_i \neq x_j \wedge \forall y \bigvee_{i=1}^{k} (y = x_i \vee E(y, x_i)))$$

The following formula dom(X) says that X is a dominating set:

$$dom(X) = \forall y(X(y) \vee \exists z(X(z) \wedge E(z,y)))$$

1.2.2 Vertex Cover of Size K

$$\exists x_1, ..., x_K \forall e[\bigvee_{i=1}^{k} \mathbf{inc}(e, x_i)]$$

We define $\mathbf{inc}(e,x) := \exists y[\mathbf{edg}(e,y,x_i)]$, expressing that an edge e is incident to some vertex x. Notice that the length of the expression depends on K, we will see this with graph properties which are parameterized by some positive integer.

1.2.3 Domatic Number K

$$\exists X_1, ..., X_K[\mathbf{partition}(X_1, ..., X_K) \vee \bigvee_{i=1}^{K} \text{“}X_i \text{ is a dominating set”}]$$

The formula $\mathbf{partition}(X_1, ..., X_K)$ states that the vertex sets $X_1, ..., X_K$ form a partition of the vertices, and is defined as

$$\mathbf{partition}(X_1, ..., X_K) := \forall x[\bigvee_{i=1}^{K} x \in X_i] \vee \neg\exists x[\bigvee_{i \neq j}^{K}(x \in X_i \wedge x \in X_j)]$$

The sentence “X_i is a dominating set” can be easily constructed by modifying the formula for dominating set shown above. Its also worth noting that this is the first formula where we've used set quantifications (the previous formulas were all first order formulas). Here it is necessary, since we don't know the size of the dominating sets that form the partition of the graph.

1.2.4 $K-$colorability

$$\exists X_1, ..., X_K[\mathbf{partition}(X_1, ..., X_K) \wedge \forall x, y[\mathbf{adj}(x,y) \rightarrow \bigwedge_{i=1}^{K} \neg(x \in X_i \wedge y \in X_i)]]$$

2 Advanced Techniques

2.1 Reflexive, Transitive Closure

An important fact about monadic second order logic is that if we can express some binary relation of variables in monadic second order logic, then we can also express the reflexive, transitive closure of that relation. We will see that this is made possible by the ability to quantify over sets, and thus cannot be expressed in first order logic.

Theorem : Let R be a binary relation on a set D. Then if R is expressible in monadic second order logic then so is the transitive, reflexive closure of R.

Let R^+ denote the transitive reflexive closure of R. We say a set $X \subseteq D$ is R-closed if for any $x, y \in D$ such that $x \in X$ and xRy, we have $y \in X$.

Let $\phi(.,.)$ be a monadic second order logic formula which defines R. First, we write a formula which expresses that that a set X is R-closed.

$$\psi(X) := \forall x[x \in X \to \forall y[\phi(x,y) \to y \in X]]$$

Using the above claim, we write a formula defining the transitive, reflexive closure of R:

$$\phi^+(x,y) := \forall X[(\psi(X) \wedge x \in X \wedge \forall Y[a \in Y \wedge \psi(Y) \to \text{"}X \subseteq Y\text{"}]) \to y \in X]$$

As far as we are concerned, the main application of this theorem is to describe connectivity in graphs. Two vertices are connected if there is some chain of vertices between the two which are all adjacent. And any vertex is trivially connected to itself. In other words, connectedness is the transitive, reflexive closure of the adjacency relation.

Corollary : The following properties are expressible in monadic second order logic:
1. Two vertices x and y are connected by a path.
2. A graph is connected.
3. An edge set U forms a path between vertices x and y.
4. An edge set C forms a cycle.

2.2 Transductions

One of the most powerful tools in writing logical formulas is that of transductions. Broadly, transductions allow you to describe a graph within (N disjoint copies of) a different graph. Transductions can be used to describe properties of subgraphs, supergraphs, and various graph transformations.

Definition (Transduction Definition Scheme). Let $N > 0$ and $\mathcal{W}$ be a set of variables. A transduction definition scheme is a triple ,

$$\Delta = \langle \phi, (\psi_i)_{i=1}^3, (\mathbf{edg}_{i,j,k})_{i,j,k=1}^3 \rangle,$$

where

- An monadic second order formula ϕ with free variables in $\mathcal{W}$.
- monadic second order formulas ψ_i with free variables in $\mathcal{W} \cup \{x\}$.
- monadic second order formulas $\mathbf{edg}_{i,j,k}$ with free variables in $\mathcal{W} \cup \{x_1, x_2, x_3\}$

3 Advanced Properties

3.1 Subgraph with property P

Suppose that P is some property which can be expressed in monadic second order logic. We can describe arbitrary subgraphs with the following transduction definition scheme:

- $\mathcal{W} := \{V, E\}$
- $\phi := \forall e[e \in E \to \exists x, y[x, y \in V \wedge \mathbf{edg}(e, x, y)]]$
- $\psi_1(x) := x \in V \vee x \in E$
- $\mathbf{edg}_{1,1,1}(e, x, y) := \mathbf{edg}(e, x, y)$

With this, we can quantify over all subgraphs of a graph and determine whether any have property P.

Examples of properties which can be described in monadic second order logic and whose corresponding subgraph problem is NP-complete include:

- Bipartite
- Maximum degree less than d
- Planar: Planar graphs can be characterized by forbidden minors according to Kuratowski's theorem, and graph minors can be expressed in monadic second order logic
- Edge graph (edge graphs can be characterized by forbidden subgraphs)
- Transative: $\forall x, y, z[\mathbf{adj}(x, y) \wedge \mathbf{adj}(y, z) \to \mathbf{adj}(x, z)]$

3.2 Partition into K subgraphs with property P

Suppose that P is some monadic second order expressible property. We can describe an induced subgraph with the following transduction definition scheme:

- $\mathcal{W} := \{V\}$
- $\phi := \mathbf{true}$
- $\psi_i(x) := x \in V \vee \exists y, z[y, z \in V \wedge \mathbf{edg}(x, y, z)]$
- $\mathbf{edg}_{1,1,1}(e, x, y) := \mathbf{edg}(e, x, y)$

With this, we can quantify over all partitions of a graph into K subgraphs and determine whether any are such that each of the partitions have property P.

$$\exists X_1, ..., X_K[\textbf{partition}(X_1, ...X_K) \wedge \bigwedge_{i=1}^{K} \text{"}X_i\text{induces a graph with property P]"}$$

Examples of P which are expressible in monadic second order logic, and whose corresponding partition problem is NP-complete, include :

- Hamiltonicity
- Complete: $\forall x, y[\textbf{adj}(x, y)]$

3.3 Perfect Graph

A graph G is perfect if, for any set $X \subseteq V(G)$, the chromatic number and max clique size of the induced subgraph $G[X]$ are equal. This characterization will not help us in writing a formula, since monadic second order logic is unable to do arithmetic (specifically equality of size of sets). There is another characterization of perfect graphs, however, which will prove useful to us. We recall that the complement of a graph G is the graph whose edge set consists of edges not in G.

Theorem(Strong Perfect Graph Theorem). Let G be a graph. Then G is perfect if and only if G contains neither C_{2k+1} nor C_{2k+1} as an induced subgraph for $k \geq 2$.

We first write a formula which describes graphs which are odd cycles of length at least 5 :

$$\text{“connected”} \wedge \text{“2-regular”} \wedge \forall X[(\forall x[x \in X]) \rightarrow \textbf{Card}_{1,2}(X) \wedge \text{"}|X| \geq 5\text{"}]$$

We've previously established that all the expressions written in quotes above can be expressed in monadic second order logic. Then using the transduction in section 3.2, we can write the formula

$$\forall X[\text{“X does not induce } C_{2k+1} \text{ for } k \geq 2\text{"}]$$

It remains to show that we can express that a graph does not contain $\overline{C_{2k+1}}$ as an induced subgraph. If we can define a transduction definition scheme for the complement of a graph, then we are done. We have to be careful when defining this transduction. How many copies are necessary to describe the edges we are adding? Consider a totally disconnected graph with n vertices $-$ the complement would have $\frac{n^2-n}{2}$ edges. In the worse case scenario, we would need n copies of the graph to perform the transduction. In other words, the size of the formula would depend on the size of the graph, rendering moot any potential complexity results.

Since the complement of the graph has the same number of vertices as the original graph, we only require one copy of the graph to perform the transduction. We define the following transduction scheme to describe the complement of a graph :

- $\mathcal{W} := \varnothing$
- $\phi :=$ **true**
- $\psi_1(x) :=$ **true**
- $\mathrm{adj}_{1,1}(x, y) := \neg\exists e[\mathbf{edg}(e, x, y)]$

Lastly, we need to confirm that we can apply this transduction to the above formula; that the above formula does not require quantification over edges or edge sets. Connectedness is the transitive closure of the **adj** operator, so it $MS1$ expressible. That a graph is 2-regular can be written without edge quantifications thusly :

$$"d(x) \leq 2" := \forall y_1, y_2, y_3[\bigwedge_{i=1}^{3} \mathbf{adj}(x, yi) \rightarrow \bigvee_{i \neq j}^{3} y_i = y_j]$$

$$"d(x) \geq 2" := \exists y_1, y_2[\mathbf{adj}(x, y_1) \wedge \mathbf{adj}(x, y_2) \wedge y_1 \neq y_2]$$

$$"\text{2-regular}" := \forall x["d(x) \leq 2" \wedge "d(x) \geq 2"]$$

The rest of the formula trivially does not require edge quantification. Therefore, that a graph induces $\overline{C_{2k+1}}$ for $k \geq 2$ can be described in monadic second order logic. Then with the strong perfect graph theorem we can write a formula which describes that a graph is perfect:

$$\forall X[\text{“X does not induce } C_{2k+1}, k \geq 2\text{”} \wedge \backslash X \text{ does not induce } C_{2k+1}, k \geq 2\text{”}.$$

4 References

[1] Logic and Automata, History and Perspective. Edited by Jorg Flum Erich Gradel Thomas Wilke.

[2] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pages 193–242. Elsevier Science Publishers, Amsterdam, 1990.

[3] B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Inf. Comput., 85(1):12–75, 1990.

[4] B. Courcelle. The monadic second-order logic of graphs vii: Graphs as relational structures. Theor. Comput. Sci., 101(1):3–33, 1992.

[5] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, Handbook of Graph Grammars, pages 313–400. World Scientific, 1997.