Graph Operations and Monadic Second-Order Logic: a Survey

Bruno Courcelle*

LaBRI (CNRS, UMR 5800), Université Bordeaux-I,
351 cours de la Libération,
33405 Talence, France,
email: courcell@labri.u-bordeaux.fr,
WWW: http://dept-info.labri.u-bordeaux.fr/~courcell/ActSci.html.

We handle finite graphs in two ways, as relational structures on the one hand, and as algebraic objects, i.e., as elements of algebras, based on graph operations on the other.

Graphs as relational structures

By considering a graph as a relational structure (consisting typically, of the set of vertices as domain and of a binary relation representing the edges), one can express graph properties in logical languages like First-Order Logic or fragments of Second-Order Logic. The purpose of *Descriptive Complexity* is to relate the complexity of graph properties (or more generally of properties of finite relational structures) with the syntax of their logical expressions, and to characterize complexity classes in logical terms, independently of computation models like Turing machines.

The logical expression of graph properties raises also *satisfiability problems* for specific classes of graph, namely the problems of deciding whether a given formula of a certain logical language is satisfiable by some graph belonging to a fixed class.

Monadic Second-Order Logic

As main logical language, we will consider *Monadic Second-Order Logic*, i.e., the extension of First-Order Logic with variables denoting sets of elements of the considered structures. Despite the fact that it does not correspond exactly to any complexity class, this language enjoys a number of interesting properties.

First, it is rich enough to express nontrivial graph properties like planarity, k-vertex colorability (for fixed k), connectivity, and many others (that are not expressible in First-Order Logic).

Second, it is an essential tool for studying context-free graph grammars. In particular, certain graph transformations expressible by Monadic Second-Order formulas behave very much like Rational Transductions (or Tree Transductions) used in the Theory of Formal Languages.

* This research is supported by the European Community Training and Mobility in Research network GETGRATS.

Third, the verification, optimization and counting problems, expressible in Monadic Second-Order logic are efficiently solvable for certain classes of "hierarchically structured graphs" i.e., of graphs built from finite sets of graphs by means of finitely many graph operations. (This the case of the well-known class of partial k-trees, equivalently, of graphs of tree width at most k). We will discuss this second way of handling graphs shortly. Let us precise here that a *verification* (or *model checking*) *problem* consists in testing whether a given graph from a certain class is a model of a fixed closed logical formula (here a Monadic Second-Order formula). An *optimization problem* consists in computing for a given graph (from a certain class), the minimum (or maximum) cardinality of a set of vertices satisfying a fixed formula with one free set variable, again of Monadic Second-Order Logic. The length of a shortest path between two specified vertices and the maximum size of a planar induced subgraph in a given graph are expresible in this way. A *counting problem* consists in counting the number of sets satisfying a given formula. The number of paths between two specified vertices is of this form.

Graph operations

The algebraic approach to graphs pertains to the extension to sets of finite (and even countably infinite) graphs of several notions of Formal Language Theory based on the monoid structure of words. Two basic such notions are *context-freeness* and *recognizability* (defined in terms of finite congruences). The graph operations we will consider can be seen as generalizations of the concatenation of words, or of the construction of a tree from smaller trees connected by a new root.

From an algebra of finite graphs, generated by finitely many graph operations and basic graphs, one obtains:

- a specification of its graphs by algebraic terms: this yields a linear notation for these graphs, and also a background for inductive definitions and proofs,
- a notion of *context-free graph grammar* formalized in terms of *systems of recursive set equations* having least solutions: this is, by far, the easiest and most general way to handle context-free graph grammars,
- an *algebraic notion of recognizability*, defined in terms of finite congruences; an algebraic notion is useful because there is no appropriate notion of finite-state automaton for graphs except in very special cases; however, algebraic recognizability yields finite-state tree automata processing the syntax trees of the considered graphs.

Provided the graph operations are *compatible with Monadic Second-order logic* (this notion has a precise definition), we also obtain linear algorithms for every verification, optimization or counting problem expressed in Monadic Second-Order Logic, on the graphs of the corresponding algebras. These algorithms are based on tree automata traversing the syntax trees of the given graphs, and these automata exist because Monadic Second-Order Logic is equivalent to recognizability on finite graphs.

A drawback of this theory is that we cannot handle the class of all finite graphs as a single finitely generated algebra based on operations compatible with Monadic Second-Order Logic. But this is unavoidable, unless P = NP.

Context-free graph grammars and Monadic Second-Order Logic

There exist only two classes of context-free graph grammars. They are called the *HR Grammars* (HR stands for *hyperedge replacement*) and the *VR Grammars* (VR stands for *vertex replacement*). The corresponding classes of sets of graphs are closed under graph transformations expressible in Monadic Second-Order Logic, and are generated from the set of binary trees by such transformations. Hence, the classes HR and VR are robust and have characterizations independent of any choice of graph operations. This establishes a strong connection between context-free graph grammars and Monadic Second-Order Logic, and, more generally, between the two ways we handle graphs.

Graph operations compatible with Monadic Second-Order Logic

The graph operations we know enjoying the desired "compatibility properties" are the following ones, dealing with k-graphs, i.e., with graphs the vertices of which are colored with k colors (neighbour vertices may have the same color):

(a) disjoint union of two k-graphs,
(b) uniform change of color (i.e., all vertices of the input k-graph colored by p are then colored by q), for fixed colors p, q,
(c) redefinition of the (binary) edge relation of the structure representing the input k-graph by a fixed quantifier-free formula using possibly the k color predicates,
(d) an operation that fuses all vertices of the input k-graph having color p into a single one, for fixed color p.

The edge complement is an example of an operation of type (c) (its definition needs no color predicate).

For generating graphs, the operations of the forms (c) and (d) can be replaced by operations of the restricted form:

(c') addition of edges between any vertex colored by p and any vertex colored by q,

at the cost of using (many) more colors. (See [6]).

The *clique-width* of a graph G is defined as the minimal number k of colors that can be used in an algebraic expression denoting this graph and built from one-vertex graphs and operations of the forms (a), (b), (c') (using only these k colors). This complexity measure is comparable to tree-width, but stronger in the sense that, for a set of finite graphs, bounded tree-width implies bounded clique-width, but not vice versa. (See [1,6,7]).

The HR context-free graph grammars are those defined as systems of equations using operations (a), (b) and (d). The VR context-free graph grammars are those defined as systems of equations using operations of all four types. (See [6]).

Summary of the lecture

In a first part, we will review these notions, we will give various examples of graph operations and of Monadic Second-Order graph properties ([2,5,6,7]). In a second part, we will focus our attention on the "compatibility" conditions mentioned above and on operations of type (c) and (d): we will review results from [5]. In a third part, we will present the following open problems:

1. *The parsing problem*: What is the complexity of deciding whether the clique-width of a given graph is at most k? It is polynomial for k at most 3, NP otherwise ([1]). Is it NP complete for fixed values of k? For the algorithmic applications, one needs an algorithm producing not only a "yes/no" answer but also an algebraic expression in case of "yes" answers.
2. *Alternative complexity measure*: Can one define a complexity measure *equivalent* to clique-width (equivalent in the sense that the same sets of finite graphs have bounded "width"), such that the corresponding parsing problem is polynomial for each value k?
3. *Countable graphs*: From infinite expressions using the operation of types (a), (b), (c'), one can define the clique-width of a countable graph G. It may be finite but strictly larger than the maximum clique-width of the finite induced subgraphs of G. How large can be the gap? Is there an equivalent complexity measure for which there is no gap? Preliminary results can be found in [4].
4. *An open conjecture by Seese*: If a set of finite or countable graphs has a decidable satisfiability problem for Monadic Second-Order Formulas, then it has bounded clique-width. The result of [4] reduces this conjecture to the case of sets of finite graphs, but the hard part remains; partial results have been obtained in [3].

More open problems can be found from: http://dept-info.labri.u-bordeaux.fr/~courcell/ActSci.html. A list of results on clique-width is maintained on the page: http://www.laria.u-picardie.fr/~vanherpe/cwd/cwd.html

References:

[**1**] **D. Corneil, M. Habib, J.M. Lanlignel, B. Reed, U. Rotics**, Polynomial time recognition of clique-width at most 3 graphs, Conference LATIN 2000, to appear.

[**2**] **B. Courcelle**: The expression of graph properties and graph transformations in monadic second-order logic, Chapter 5 of the "Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations", G. Rozenberg ed., World Scientific (New-Jersey, London), 1997, pp. 313-400.

[**3**] **B. Courcelle**: The monadic second-order logic of graphs XIV: Uniformly sparse graphs and edge set quantifications, 2000, submitted.

[**4**] **B. Courcelle**: Clique-width of countable graphs: a compactness property. International Conference of Graph Theory, Marseille, 2000.

[**5**] **B. Courcelle, J. Makowsky**: Operations on relational structures and their compatibility with monadic second-order logic, 2000, submitted.

[**6**] **B. Courcelle, J. Makowsky, U. Rotics**: Linear time solvable optimization problems on certain structured graph families Theory of Computer Science (formerly Mathematical Systems Theory) **33** (2000) 125-150.

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

Full texts of submitted or unpublished papers and other references can be found from: http://dept-info.labri.u-bordeaux.fr/~courcell/ActSci.html.