Monadic second-order logic for graphs. Algorithmic and language theoretical applications*

Bruno Courcelle

Université Bordeaux-1, LaBRI, CNRS
Institut Universitaire de France
351, Cours de la Libération
33405, Talence cedex, France
courcell@labri.fr

Abstract. This tutorial will present an overview of the use of Monadic Second-Order Logic to describe sets of finite graphs and graph transformations, in relation with the notions of tree-width and clique-width. It will review applications to the construction of algorithms, to Graph Theory and to the extension to graphs of Formal Language Theory concepts.

We first explain the role of Logic. A graph, either finite or infinite, can be considered as a logical structure whose domain (the ground set of the logical structure) consists of the set of vertices ; a binary relation on this set represents adjacency. Graph properties can be expressed by logical formulas of different languages and classified accordingly. First-order formulas are rather weak in this respect because they can only express *local properties*, like having degree or diameter at most k for fixed k. Most properties of interest in Graph Theory can be expressed in second-order logic (this language allows quantifications on relations of arbitrary but fixed arity), but unfortunately, little can be obtained from such expressions.

Monadic second-order formulas are second-order formulas that only use quantifications on unary relations, i.e., on sets. They can express many basic and useful graph properties like connectivity, k-colorability, planarity and minor inclusion, just to take a few examples. These properties are said to be *monadic second-order expressible* and the corresponding sets of graphs are *monadic second-order definable*. Many algorithmic properties follow from such logical descriptions. In particular, every monadic second-order definable set of *finite* graphs of bounded *tree-width* has a linear time recognition algorithm ([1], [2], [4], [5], [6]).

Monadic second-order formulas are also used in Formal Language Theory to describe *languages*, i.e., sets of words or terms. A fundamental result in this field is that monadic second-order formulas and finite automata have the same expressive power. It is fundamental for the theory and practice of model-checking,

* Supported by the GRAAL project of "Agence Nationale pour la Recherche".

and, in Language Theory, it helps to analyze and classify the *regular languages*, i.e., those defined by finite automata. Monadic second-order formulas are even more important for describing sets of graphs than for describing languages because there is no convenient notion of graph automaton. They can also replace finite automata for defining graph transformations, that we call *transductions*, as in Language Theory.

However, monadic second-order logic alone yields no interesting results. As mentioned above every monadic second-order graph property has, for each k, a polynomial time checking algorithm for graphs of tree-width at most k. No such algorithm can exist for arbitrary graphs, otherwise P=NP because 3-colorability is expressible by a monadic second-order formula. The checking problem for every monadic second-order graph property is nevertheless *fixed parameter tractable* (FPT) for tree-width as parameter. (See [5], [6] on this notion). Hence in order to be useful, the expression of a graph property by a monadic second-order formula must be coupled with constraints on the considered graphs like having bounded tree-width, or with other constraints that are also based on appropriate types of hierarchical decompositions. *Clique-width* is another graph parameter, based on certain graph decompositions, that yields FPT algorithms for monadic second-order expressible properties ([2], [3], [7]). Tree-width and clique-width are important for language theoretical issues as well as for the construction of polynomial algorithms.

It is convenient to formalize the *hierarchical graph decompositions* that yield the notions of tree-width and clique-width and fit with monadic second-order logic, as algebraic terms written with appropriate *graph operations* that generalize the concatenation of words. The value of such a term t is a finite graph G and t is one of its hierarchical decompositions of the considered type. The subterms of t define (roughly speaking) combinations of larger and larger subgraphs of G. This fact justifies the use of the word "hierarchical" in the above description.

Sets of finite graphs can be described as subsets of certain graph algebras, by means of notions of Universal Algebra that are already known in Language Theory. Two of these notions are those of an *equational* and of a *recognizable* set of elements of an algebra. In the monoid of words, the corresponding classes of sets are respectively those of context-free and of regular languages. The notion of an equational set generalizes to arbitrary algebras the well-known *Least-Fixed Point Characterization* of context-free languages, and that of a recognizable set generalizes the characterization of regular languages in terms of finite congruences. Many properties of equational and recognizable sets of graphs are just particular instances of results that hold in arbitrary algebras, but others are particular to the considered algebras of graphs.

Finite graphs can thus be handled in two ways. The "logical way" characterizes graphs "from inside", that is, in terms of what they are made of and contain: vertices, edges, paths, minors, subgraphs. The "algebraic way" characterizes sets of graphs in a global way : a graph is treated as an element of an algebra and related with other elements of the same algebra, that are not necessarily among its subgraphs. Two important theorems relate these two ap-

proaches. The *Recognizability Theorem* says that every set of finite graphs that is monadic second-order definable is recognizable. The *Equationality Theorem* says that a set of finite graphs is equational if and only if it is the set of graphs "definable inside finite trees" by a fixed finite tuple of monadic second-order formulas : we will say : "is the image of a set of finite trees under a *monadic second-order transduction*".

It follows from the Recognizability Theorem that, for a graph G defined by a term t (relative to an appropriate graph algebra), one can check in time $O(|t|)$ whether or not G satisfies a fixed monadic second-order property. It follows also that the graphs of an equational set that satisfy a fixed monadic second-order property (like planarity) form an equational set. We call this result the *Filtering Theorem*. It generalizes the classical fact that the intersection of a context-free language with a regular one is context-free. Since the emptiness of an equational set is decidable, we get as a corollary that the *monadic second-order satisfiability problem* is decidable for every equational set L . This means that one can decide whether or not a given monadic second-order formula is satisfied by some graph in L. The Equationality Theorem entails that the family of equational sets of graphs is preserved under monadic second-order transductions. This corollary is similar to the fact that the image of a context-free language under a rational transduction is context-free. (A rational transduction can be specified by a nondeterministic finite-state transducer, and if the image of every word is a finite set, then it is also a monadic second-order transduction). The corollary follows from the fact that the class of monadic second-order transductions is closed under composition.

The Recognizability and the Equationality Theorem contribute to establishing the foundations of a sound and robust extension of the theory of formal languages intended to cover descriptions of sets of finite graphs, and in which monadic second-order logic plays a major role. From the above informal statements, this extension may seem to be straightforward. However, general graphs are intrinsically more complex than words and terms, and some results do not extend. Let us give two examples. The set of all finite graphs is not equational, whereas the set of all words on a finite alphabet is (trivially) context-free. There are uncountably many recognizable sets of graphs, and this fact forbids any characterization of these sets in terms of graph automata, that would generalize a classical characterization of regular languages. These examples reflect the fact that the sets of graph operations upon which Recognizability and Equationality are based are infinite, and that this infiniteness is unavoidable.

According to the above presentation, monadic second-order formulas expressing graph properties do not use edge set quantifications. This is due to the chosen representation of a graph by a relational structure. If we replace a graph G by its incidence (bipartite) graph $Inc(G)$, where each edge is made into a vertex and where the adjacency relation $edg_{Inc(G)}(e, v)$ expresses that a vertex v of G is an end of edge e, then monadic second-order formulas to be interpreted in $Inc(G)$ can use edge set quantifications. A graph property is MS_2-*expressible* if it is expressible by a monadic second-order formula on incidence graphs. This variant of

monadic second-order logic has strictly more expressive power than the initially defined language. The existence of a perfect matching is MS_2-expressible but not monadic second-order expressible. However, the two variants of monadic second-order logic have the same expressive power on words, on trees and on certain classes of graphs like those of planar graphs or, for each k, of graphs of degree at most k.

The Recognizability Theorem and the Equationality Theorem have thus two versions, relative to the two possible representations of graphs by relational structures and to two different graph algebras. The graph algebra corresponding to MS_2-formulas, called the *HR algebra*, is the one with graph operations that express tree-decompositions and characterize tree-width. The one corresponding to monadic second-order formulas without edge set quantifications is called the *VR algebra*, it defines clique-width. The acronyms HR and VR refer respectively to *hyperedge replacement* and *vertex replacement* because the equational sets of the HR and of the VR algebras are the sets of graphs generated by certain context-free graph grammars called respectively *hyperdege replacement* and *vertex replacement graph grammars*. (See [1] and the first two chapters of the same volume.)

These results are also interesting for Structural Graph Theory, i.e., for the study of graph decompositions, of embeddings of graphs on surfaces, of forbidden configuration characterizations, of colorings expressed as homomorphisms between graphs. Quick proofs that certain graph classes have bounded tree-width or clique-width can be obtained from the Equationality Theorem.

All these definitions and results are in a book in preparation [2].

References

1. B. Courcelle, The Expression of Graph Properties and Graph Transformations in Monadic Second-Order Logic, in *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. G. Rozenberg ed., World Scientific, 1997, pp. 313-400
2. B. Courcelle, Graph Structure and Monadic Second-order Logic, book in preparation, to be published by Cambridge University Press, readable on http://www.labri.fr/perso/courcell/ActSci.html
3. B. Courcelle, J. Makowsky, U. Rotics, Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width, *Theory Comput. Syst.* **33** (2000) 125-150
4. B. Courcelle, M. Mosbah, Monadic Second-order Evaluations on Tree-decomposable Graphs. *Theor. Comput. Sci.* **109** (1993) 49-82
5. R. Downey, M. Fellows, *Parameterized Complexity*, Springer, 1999
6. J. Flum, M. Grohe, *Parameterized Complexity Theory*, Springer, 2006
7. J. Makowsky, Algorithmic uses of the Feferman-Vaught Theorem, *Ann. Pure Appl. Logic* **126** (2004) 159-213