Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

Model Checking with Fly-Automata

Bruno Courcelle*[a] Irene Durand[b]
[a]LaBRI, CNRS, Bordeaux University, Talence, France
[b]LaBRI, CNRS, Bordeaux University, Paris, France

Q1

Keywords Model checking • Monadic second-order logic • Tree-width • Clique-width • Fixed-parameter tractable algorithm • Automaton on terms • Fly-automaton

Summarized Original Work

2012, 2013; Courcelle, Durand

Problem Definition

The verification of monadic second-order (MSO) graph properties, equivalently, the model-checking problem for MSO logic over finite binary relational structures, is fixed-parameter tractable (FPT) where the parameter consists of the formula that expresses the property and the tree-width or the clique-width of the input graph or structure. How to build usable algorithms for this problem? The proof of the general theorem (an *algorithmic meta-theorem*, cf. [12]) is based on the description of the input by algebraic terms and the construction of finite automata that accept the terms describing the satisfying inputs. But these automata are in practice much too large to be constructed [11, 14]. A typical number of states is $2^{2^{10}}$, and lower bounds match this number. Can one use automata and overcome this difficulty?

Key Results

We propose to use *fly-automata* (*FA*) [3]. They are automata whose states are *described* and not *listed* and whose transitions are *computed on the fly* and not *tabulated*. When running on a term of size 1,000, a fly-automaton with $2^{2^{10}}$ states computes only 1,000 transitions if it is deterministic. FA can have infinitely many states. For example, a state can record, among other things, the (unbounded) number of occurrences of a particular symbol in the input term. FA can thus check certain graph properties that are *not monadic second-order expressible*. An example is *regularity*, the fact that all vertices have the same degree. Furthermore, an FA equipped with an *output function* that maps the set of accepting states to an effectively given domain $\mathcal{D}$ can compute a value, for example, the number of k-colorings of the given graph G or the minimum cardinality of one of the k color classes if G is k-colorable (this number measures how close this graph is to be $(k-1)$-colorable). We have implemented and tested an FA that computes the number of 3-colorings of a graph.

*E-mail: courcell@labri.fr

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

Tree-width and *clique-width* are graph complexity measures that serve as parameters in many FPT algorithms [7, 8, 10]. Both are based on hierarchical decompositions of graphs that can be expressed by terms written with the operation symbols of appropriate *graph algebras* [6]. The model-checking automata take such terms as inputs. We will present results concerning graphs of bounded clique-width. The similar results for graphs of bounded tree-width reduce to them as we will explain at the end of this section.

Q2

Graphs and Monadic Second-Order Logic

Graphs are finite, undirected, and without loops and multiple edges. The extension to directed graphs, possibly with loops and/or labels, is straightforward. A graph G is identified with the relational structure $\langle V_G, edg_G\rangle$ where edg_G is a binary symmetric relation representing adjacency.

Rather than giving a formal definition of *monadic second-order* (MSO) logic, we present the closed formula expressing 3-colorability (an NP-complete property). It is $\exists X, Y.Col(X, Y)$ where $Col(X, Y)$ is the formula

$$X \cap Y = \emptyset \wedge \forall u, v.\{edg(u, v) \Longrightarrow [\neg(u \in X \wedge v \in X) \wedge \neg(u \in Y \wedge v \in Y) \wedge \neg(u \notin X \cup Y \wedge v \notin X \cup Y)]\}.$$

This formula expresses that X, Y and $V_G - (X \cup Y)$ are the three color classes of a 3-coloring. The corresponding colors are respectively 1, 2, and 3.

Definition 1 (The graph algebra $\mathcal{G}$).

(a) We will use $\mathbb{N}_+$ as a set of labels called *port labels*. A *p-graph* is a triple $G = \langle V_G, edg_G, \pi_G\rangle$ where π_G is a mapping: $V_G \to \mathbb{N}_+$. If $\pi_G(x) = a$, we say that x is an *a-port*. The set $\pi(G)$ of port labels of G is its *type*. By using a default label, say 1, we make every nonempty graph into a p-graph of type $\{1\}$.

(b) We let F_k be the following finite set of *operations* on p-graphs of type included in $C := \{1, \ldots, k\} \subseteq \mathbb{N}_+$:

- The binary symbol $\oplus$ denotes the union of two *disjoint* p-graphs,
- The unary symbol $relab_{a\to b}$ denotes the *relabelling* that changes every port label a into b (where $a, b \in C$),
- The unary symbol $add_{a,b}$, for $a < b$, $a, b \in C$, denotes the *edge addition* that adds an edge between every a-port x and every b-port y (unless there is already an edge between them, our graphs have no multiple edges),
- For each $a \in C$, the nullary symbol **a** denotes an isolated a-port.

(c) Every term t in $T(F_k)$ (the set of finite terms written with F_k) is called a *k-expression*. Its *value* is a p-graph, $val(t)$, that we now define. For each position u of t (equivalently, each node u of the syntax tree of t), we define a p-graph $val(t)/u$, whose vertex set is the set of leaves of t below u. The definition of $val(t)/u$ is, for fixed t, by bottom-up induction on u:

- If u is an occurrence of **a**, then $val(t)/u$ has vertex u as an a-port and no edge,

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

- If u is an occurrence of $\oplus$ with sons u_1 and u_2, then $val(t)/u := val(t)/u_1 \oplus val(t)/u_2$ (note that $val(t)/u_1$ and $val(t)/u_2$ are disjoint),
- If u is an occurrence of $relab_{a\to b}$ with son u_1, then $val(t)/u := relab_{a\to b}(val(t)/u_1)$,
- If u is an occurrence of $add_{a,b}$ with son u_1, then $val(t)/u := add_{a,b}(val(t)/u_1)$.

Finally, $val(t) := val(t)/root_t$. Its vertex set is the set of all leaves (occurrences of nullary symbols). For an example, let

$$t := add^1_{b,c}(add^2_{a,b}(\mathbf{a}^3 \oplus^4 \mathbf{b}^5) \oplus^6 relab^7_{b\to c}(add^8_{a,b}(\mathbf{a}^9 \oplus^{10} \mathbf{b}^{11})))$$

where the superscripts 1–11 number the positions of t. The p-graph $val(t)$ is $3_a - 5_b - 11_c - 9_a$ where the subscripts a, b, c indicate the port labels. (For clarity, port labels are letters in examples.) If $u := 2$ and $w := 8$, then $t/u = t/w = add_{a,b}(\mathbf{a} \oplus \mathbf{b})$; however, $val(t)/u$ is the p-graph $3_a - 5_b$ and $val(t)/w$ is $9_a - 11_b$, isomorphic to $val(t)/u$.

(d) The *clique-width* of a graph G, denoted by $cwd(G)$, is the least integer k such that G is isomorphic to $val(t)$ for some t in $T(F_k)$. We denote by $\mathcal{G}_k$ the set $val(T(F_k))$ of p-graphs that are the value of a term over F_k. We let F be the union of the sets F_k and $\mathcal{G}$ be the union of the sets $\mathcal{G}_k$. Every p-graph is isomorphic to a graph in $\mathcal{G}$, hence, has a clique-width.

(e) An *F-congruence* is an equivalence relation $\approx$ on p-graphs such that:

- Two isomorphic p-graphs are equivalent, and
- If $G \approx G'$ and $H \approx H'$, then $\pi(G) = \pi(G')$, $G \oplus H \approx G' \oplus H'$, $add_{a,b}(G) \approx add_{a,b}(G')$ and $relab_{a\to b}(G) \approx relab_{a\to b}(G')$.

(f) A set of graphs L is *recognizable* if it is a union of classes of an F-congruence such that, for each finite type $C \subseteq \mathbb{N}_+$, the number of equivalence classes of p-graphs of type C is finite.

Definition 2 (Fly-automata).

(a) Let H be a finite or countable, effectively given, signature. A *fly-automaton over* H (in short, an *FA over* H) is a 4-tuple $\mathcal{A} = \langle H, Q_\mathcal{A}, \delta_\mathcal{A}, Acc_\mathcal{A}\rangle$ such that $Q_\mathcal{A}$ is the finite or countable, effectively given, set of *states*; $Acc_\mathcal{A}$ is the set of *accepting states,* a decidable subset of $Q_\mathcal{A}$; and $\delta_\mathcal{A}$ is a computable function that defines the *transition rules*: for each tuple $(f, q_1, \ldots, q_m)$ with $q_1, \ldots, q_m \in Q_\mathcal{A}$, $f \in H$, $\rho(f) = m \geq 0$, $\delta_\mathcal{A}(f, q_1, \ldots, q_m)$ is a finite set of states. We write $f[q_1, \ldots, q_m] \to q$ (and $f \to q$ if f is nullary) to mean that $q \in \delta_\mathcal{A}(f, q_1, \ldots, q_m)$. We say that $\mathcal{A}$ is *finite* if H and $Q_\mathcal{A}$ are finite.

(b) *Runs* and *recognized languages* are defined as usual; see [1]. A *deterministic* FA $\mathcal{A}$ (by "deterministic" we mean "deterministic and complete") has a unique run on each term t, and $q_\mathcal{A}(t)$ is the state reached at the root of t. The mapping $q_\mathcal{A}$ is computable, and the membership in $L(\mathcal{A})$ of a term $t \in T(H)$ is decidable.

(c) Every FA $\mathcal{A}$ that is not deterministic can be *determinized* by an easy extension of the usual construction, see [3]; it is important that the sets $\delta_\mathcal{A}(f, q_1, \ldots, q_m)$ be finite.

(d) A deterministic FA over H *with output function* is a 4-tuple $\mathcal{A} = \langle H, Q_\mathcal{A}, \delta_\mathcal{A}, Out_\mathcal{A}\rangle$ that is a deterministic FA where $Acc_\mathcal{A}$ is replaced by a total and computable *output function* $Out_\mathcal{A}$: $Q_\mathcal{A} \to \mathcal{D}$ such that $\mathcal{D}$ is an effectively given domain. The *function computed by* $\mathcal{A}$ is $Comp(\mathcal{A}) : T(H) \to \mathcal{D}$ such that $Comp(\mathcal{A})(t) := Out_\mathcal{A}(q_\mathcal{A}(t))$.

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

Example 1. The number of accepting runs of an automaton.

Let $\mathcal{A} = \langle H, Q_{\mathcal{A}}, \delta_{\mathcal{A}}, Acc_{\mathcal{A}} \rangle$ be a nondeterministic FA. We construct a deterministic FA $\mathcal{B}$ that computes the number of accepting runs of $\mathcal{A}$ on any term in $T(H)$. As set of states $Q_{\mathcal{B}}$, we take the set of finite subsets of $Q_{\mathcal{A}} \times \mathbb{N}_+$. The transitions are defined so that $\mathcal{B}$ reaches state α at the root of $t \in T(H)$ if and only if α is the finite set of pairs $(q, n) \in Q_{\mathcal{A}} \times \mathbb{N}_+$ such that n is the number of runs of $\mathcal{A}$ that reach state q at its root. This number is finite and α can be seen as a partial function: $Q_{\mathcal{A}} \to \mathbb{N}_+$ having a finite domain. For a symbol f of arity 2, $\mathcal{B}$ has the transition: $f[\alpha, \beta] \to \gamma$ where γ is the set of pairs (q, n) such that n is the sum of the integers $n_p.n_r$ over all pairs $(p, r) \in Q_{\mathcal{A}} \times Q_{\mathcal{A}}$ such that $(p, n_p) \in \alpha$, $(r, n_r) \in \beta$ and $q \in \delta_{\mathcal{A}}(f, p, r)$. The transitions for other symbols are defined similarly. The function $Out_{\mathcal{A}}$ maps a state α to the sum of the integers n such that $(q, n) \in \alpha \cap (Acc_{\mathcal{A}} \times \mathbb{N}_+)$.□

Example 2. An FA for checking 3-colorability.

In order to construct an FA that accepts the terms $t \in T(F)$ such that $val(t)$ is 3-colorable, we first construct an FA $\mathcal{A}$ for the property $Col(X, Y)$. For this purpose, we transform F into $F^{(2)}$ by replacing each nullary symbol $\mathbf{a}$ by the four nullary symbols $(\mathbf{a}, ij)$, $i, j \in \{0, 1\}$. A term $t \in T(F^{(2)})$ defines, first, the graph $val(t')$ where t' is obtained from t by removing the Booleans i, j from the nullary symbols and, second, the pair (V_X, V_Y) such that V_X is the set of vertices u (leaves of t) that are occurrences of $(\mathbf{a}, 1j)$ for some $\mathbf{a}$ and j and V_Y is the set of those that are occurrences of $(\mathbf{a}, i1)$ for some $\mathbf{a}$ and i. The set of terms $t \in T(F^{(2)})$ such that $Col(V_X, V_Y)$ holds in $val(t')$ is defined by a deterministic FA $\mathcal{A}$ that we now specify. Its states are *Error* and the finite subsets of $\mathbb{N}_+ \times \{1, 2, 3\}$. Their meanings are as follows:

- At position u of t, the automaton reaches state *Error* if and only if $val(t')/u$ has a vertex in $V_X \cap V_Y$ or an edge between two vertices, either both in V_X or both in V_Y or both in $V_G - (V_X \cup V_Y)$, hence of the same color, respectively 1, 2, or 3;
- It reaches state $\alpha \subseteq \mathbb{N}_+ \times \{1, 2, 3\}$ if and only if these conditions do not hold and α is the set of pairs (a, i) such that $val(t')/u$ has an a-port of color i.

All states except *Error* are accepting. Here are the transitions of $\mathcal{A}$:

$$(\mathbf{a}, 00) \to \{(a, 3)\}, (\mathbf{a}, 10) \to \{(a, 1)\}, (\mathbf{a}, 01) \to \{(a, 2)\}, (\mathbf{a}, 11) \to Error.$$

For $\alpha, \beta \subseteq \mathbb{N}_+ \times \{1, 2, 3\}$, $\mathcal{A}$ has transitions:

$$\begin{aligned}
&\oplus[\alpha, \beta] \to \alpha \cup \beta,\\
&add_{a,b}[\alpha] \to Error, \text{ if } (a, i) \text{ and } (b, i) \text{ belong to } \alpha \text{ for some } i = 1, 2, 3,\\
&add_{a,b}[\alpha] \to \alpha, \text{ otherwise },\\
&relab_{a \to b}[\alpha] \to \beta, \text{ obtained by replacing } a \text{ by } b \text{ in each pair of } \alpha.
\end{aligned}$$

Its other transitions are $\oplus[\alpha, \beta] \to Error$ if α or β is *Error*, $add_{a,b}[Error] \to Error$, and $relab_{a \to b}[Error] \to Error$.

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

This FA checks $Col(X, Y)$. To check, $\exists X, Y.Col(X, Y)$, we build a nondeterministic FA $\mathcal{B}$ by deleting the state *Error*, by replacing the first three rules of $\mathcal{A}$ by $\mathbf{a} \to \{(a, 3)\}, \mathbf{a} \to \{(a, 1)\}, \mathbf{a} \to \{(a, 2)\}$, and by deleting those that yield *Error*. All states are accepting, but on some terms, no run can reach the root, and these terms are rejected. Furthermore, the construction of Example 1 shows how to make $\mathcal{B}$ into a deterministic FA that computes the number of 3-colorings, because the 3-colorings of $val(t)$ are in bijection with the accepting runs of $\mathcal{B}$ on t. □

Recognizability Theorem: The set of graphs that satisfy a closed MSO formula φ is F-recognizable.

Weak Recognizability Theorem: For every closed MSO formula φ, for every k, the set of graphs in $\mathcal{G}_k$ that satisfy φ is F_k-recognizable.

Proofs: The Recognizability Theorem is Theorem 5.68 of [6]. Its proof shows that the equivalence defined by the fact that the two considered p-graphs have the same type and satisfy the same closed MSO formulas of quantifier height at most that of φ satisfies the conditions of Definition 1(f). (These formulas have unary predicates for expressing port labels.) The Weak Recognizability Theorem follows from the former one. It can be proved directly by constructing an FA over F [3]. (We construct a single FA, not a particular FA for each subsignature F_k as in Theorem 6.35 of [6].) This construction can be implemented, at least in a number of nontrivial cases. The proof of the strong theorem does not provide any usable automaton.

Counting and Optimizing Automata

Let $P(X_1, \ldots, X_s)$ be an MSO property of vertex sets $X_1, \ldots, X_s$. We denote $(X_1, \ldots, X_s)$ by $\overline{X}$ and $t \models P(\overline{X})$ means that $\overline{X}$ satisfies P in the graph $val(t)$ defined by a term t. We are interested not only to check the validity of $\exists \overline{X}.P(\overline{X})$ but also to compute from a term t the following values:

$\#\overline{X}.P(\overline{X})$, defined as the number of assignments $\overline{X}$ such that $t \models P(\overline{X})$,
$\mathrm{Sp}\overline{X}.P(\overline{X})$, the *spectrum* of $P(\overline{X})$, defined as the set of tuples of the form $(|X_1|, \ldots, |X_s|)$ such that $t \models P(\overline{X})$,
$\mathrm{MSp}\overline{X}.P(\overline{X})$, the *multispectrum* of $P(\overline{X})$, defined as the multiset of tuples $(|X_1|, \ldots, |X_s|)$ such that $t \models P(\overline{X})$.

These computations can be done by FA. The construction for $\#\overline{X}.P(\overline{X})$ is based on Example 1. We obtain in this way FPT or XP algorithms [8, 10].

Edge Set Quantifications and Tree-Width

The two recognizability theorems and the corresponding constructions of FA yielding FPT and XP algorithms hold and can be done for graphs of bounded tree-width and MSO formulas with edge set quantifications: it suffices to replace a graph G by its incidence graph $Inc(G)$, a bipartite graph whose vertices are those of G and its edges, to observe that the clique-width of $Inc(G)$ is bounded in terms of the tree-width of G and that an MSO formula with edge set quantifications over G can be translated into an MSO formula over $Inc(G)$. Another approach is in [2].

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

Beyond MS Logic

The property that the considered graph is the union of two disjoint regular graphs with possibly some edges between these two subgraphs is not MSO expressible but can be checked by an FA. An FA can also compute the minimal number of edges between X and $V_G - X$ such that $G[X]$ and $G[V_G - X]$ are connected, when such a set X exists.

Open Problems

The parsing problem for graphs of clique-width at most k is NP-complete (with k in the input) [9]. Good heuristics remain to be developed.

Experimental Results

These constructions have been implemented and tested [3–5]. We have computed the number of optimal colorings of graphs of clique-width at most 8 for which the chromatic polynomial is known, which allows to verify the correctness of the automaton. We can verify in, respectively, 35 and 105 min that the 20×20 and the 6×60 grids are 3-colorable. In 29 min, we can verify that the McGee graph (24 vertices) given by a term over F_{10} is acyclically 3-colorable.

A different approach using games is presented in [13].

Recommended Reading

1. Comon H et al (2007) Tree automata techniques and applications. http://tata.gforge.inria.fr/
2. Courcelle B (2012) On the model-checking of monadic second-order formulas with edge set quantifications. Discret Appl Math 160:866–887
3. Courcelle B, Durand I (2012) Automata for the verification of monadic second-order graph properties. J Appl Logic 10:368–409
4. Courcelle B, Durand I (submitted for publication, 2013) Computations by fly-automata beyond monadic second-order logic. http://arxiv.org/abs/1305.7120
5. Courcelle B, Durand I (2013) Model-checking by infinite fly-automata. In: Proceedings of the 5th conference on algebraic informatics, Porquerolles. Lecture notes in computer science, vol 8080, pp 211–222
6. Courcelle B, Engelfriet J (2012) Graph structure and monadic second-order logic, a language theoretic approach. Volume 138 of encyclopedia of mathematics and its application. Cambridge University Press, Cambridge
7. Courcelle B, Makowsky J, Rotics U Linear-time solvable optimization problems on graphs of bounded clique-width. Theory Comput Syst 33:125–150 (2000)
8. Downey R, Fellows M (1999) Parameterized complexity. Springer, New York
9. Fellows M, Rosamond F, Rotics U, Szeider S (2009) Clique-width is NP-complete. SIAM J Discret Math 23:909–939
10. Flum J, Grohe M (2006) Parametrized complexity theory. Springer, Berlin
11. Frick M, Grohe M (2004) The complexity of first-order and monadic second-order logic revisited. Ann Pure Appl Logic 130:3–31

Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_692-1

12. Grohe M, Kreutzer S (2011) Model theoretic methods in finite combinatorics. In: Grohe M, Makowsky J (eds) Contemporary mathematics, vol 558. American Mathematical Society, Providence, pp 181–206
13. Kneis J, Langer A, Rossmanith P (2011) Courcelle's theorem – a game-theoretic approach. Discret Optim 8:568–594
14. Reinhardt K (2002) The complexity of translating logic to finite automata. In: Graedel E et al (eds) Automata, logics, and infinite games: a guide to current research. Lecture notes in computer scienc, vol 2500. Springer, Berlin/Hong Kong, pp 231–238

Q6