Model-Checking by Infinite Fly-Automata

Bruno Courcelle and Irène Durand

Université Bordeaux-1, LaBRI, CNRS
351, Cours de la Libération
33405, Talence, France
{courcell,idurand}@labri.fr

Abstract. We present logic based methods for constructing XP and FPT graph algorithms, parameterized by tree-width or clique-width. We will use *fly-automata* introduced in a previous article. They make it possible to check properties that are *not* monadic second-order expressible because their states may include counters, so that their set of states may be infinite. We equip these automata with *output functions*, so that they can compute values associated with terms or graphs. We present tools for constructing easily algorithms by combining predefined automata for basic functions and properties.

1 Introduction

Finite automata on terms that denote graphs of bounded tree-width or clique-width can be used to check monadic second-order properties of the denoted graphs. However, these automata have in most cases so many states that their transition tables cannot be built [13,15]. In the article [4] we have introduced automata called *fly-automata* whose states are described (but not listed) and whose transitions are computed on the fly (and not tabulated). Fly-automata can have infinite sets of states. For example, a state can record, among other things, the (unbounded) number of occurrences of a particular symbol. We exploit this feature in the construction of fly-automata that check properties that are *not monadic second-order (MS) expressible*. Furthermore, we equip automata with *output functions*, which map accepting states to some effectively given domain $\mathcal{D}$ (e.g., the set of integers, or of pairs of integers, or the set of words over a fixed alphabet). Hence, a fly-automaton $\mathcal{A}$ defines a mapping from $T(F)$ (the set of terms over the signature F) to $\mathcal{D}$, and we construct automata that yield polynomial-time algorithms for these mappings. The height $ht(t)$ of a term t and the number $|t|$ of its positions are obviously computable in this way. The *uniformity* of a term, *i.e.*, the property that all maximal branches of its syntactic tree have the same length, can be checked by a polynomial-time fly-automaton (but not by a finite automaton). (*Symbolic automata* [16] have "small" sets of states and "large" sets of symbols. Symbols are described by properties rather than listed. Fly-automata have, to the opposite, "small" sets of symbols and "large" sets of states).

Our main interest is actually in the case where F is the signature F_∞ of "clique-width graph operations", and for fly-automata that define mappings from

T. Muntean, D. Poulakis, and R. Rolland (Eds.): CAI 2013, LNCS 8080, pp. 211–222, 2013.

the graphs defined by terms in $T(F_\infty)$ to $\mathcal{D}$. We construct fly-automata that yield FPT and XP algorithms [10,12] for clique-width as parameter. Since the clique-width $cwd(G)$ of a simple graph G is bounded in terms of its tree-width $twd(G)$ (we have $cwd(G) \leq 2^{2twd(G)+2}+1$, [7], Proposition 2.114, and [3]), all our results for graphs of bounded clique-width apply immediately to graphs of bounded tree-width. The graphs of clique-width at most k are those denoted by the terms in $T(F_k)$ where F_k is a finite subset of F_∞. As in [4], we construct elementary fly-automata for basic functions and properties, e.g., the degree of a vertex or the regularity of graph. Then, we consider more complex functions and properties written with these functions and properties (and the basic MS properties of [4]) and functional and logical constructors. For example, $\exists X, Y.(Partition(X,Y)$ $\wedge Reg[X] \wedge Reg[Y])$ expresses that the graph is the union of two disjoint regular graphs with possibly some edges between them. Here are some typical examples of questions and functions that we can handle in this way:

(1) Is it possible to cover a graph with s cliques?

(2) Does there exist an equitable s-coloring? *Equitable* means that the sizes of any two color classes differ by at most 1 (see [11]). We express this property by: $\exists X_1, \ldots, X_s.(Partition(X_1, \ldots, X_s) \wedge St[X_1] \wedge ... \wedge St[X_s]$
$\wedge |X_1| = \ldots = |X_{i-1}| \geq |X_i| = \ldots = |X_s| \geq |X_1| - 1)$
where $St[X]$ means that $G[X]$ is *stable*, *i.e.,* has no edge.

(3) Assuming that the graph is s-colorable, what is the minimum size of X_1 in an s-coloring $(X_1, \ldots, X_s)$?

(4) Which sets X such that $G[X]$ and $G[V_G - X]$ are connected, minimize the number of edges between X and $V_G - X$?

More generally, let $P(X_1, ..., X_s)$ be a property of vertex sets $X_1, ..., X_s$. Everywhere in the sequel, we denote $(X_1, ..., X_s)$ by $\overline{X}$ and $t \models P(\overline{X})$ means that $\overline{X}$ satisfies P in the graph $G(t)$ defined by t; this writing does not assume that P is written in any particular logical language. We are interested, not only to check the validity of $\exists \overline{X}.P(\overline{X})$ in some term t, but also to compute from t the following objects:

$\#\overline{X}.P(\overline{X})$, 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 $t \models P(\overline{X})$,
$\mathrm{Sat}\overline{X}.P(\overline{X})$ as the set of assignments $\overline{X}$ such that $t \models P(\overline{X})$.

Each prefix $\#\overline{X}, \mathrm{Sp}\overline{X}.$ etc. can be considered as a generalized quantifier that binds the variables of $\overline{X}$. The associated values (numbers or sets of tuples of numbers) can be computed from $\mathrm{Sat}\overline{X}.P(\overline{X})$, a set of s-tuples of subsets of $Pos(t)$ (the set of *positions* of t, i.e., of nodes of the syntactic tree of t) that may be of exponential cardinality $2^{s.|t|}$, hence, not computable by a polynomial-time algorithm.

We provide logic based methods for proving the existence of FPT and XP algorithms for terms and graphs. We generalize constructions of [1,2,8]. These constructions have been implemented and tested [4,5,6].

Lacking of space (see [5] for details and proofs), we do not review MS logic and clique-width. We only recall that edges are introduced by means of operations on vertex labeled graphs. A vertex labeled by a is an *a-port*. Notation is as in [4,7]. If $t \in T(F)$, i.e., is a term over a signature F, we let $Sig(t)$ be the set of symbols of F that occur in t.

Tuples of Sets of Positions in Terms
Let F be a signature and s be a positive integer. We let $F^{(s)}$ be the set $F \times \{0,1\}^s$ made into the signature such that the arity $\rho((f,w))$ of (f,w) is $\rho(f)$. We let $pr_s : F^{(s)} \to F$ be the relabeling that deletes the second component of a symbol (f,w). To every term $t \in T(F^{(s)})$ corresponds the term $pr_s(t)$ in $T(F)$ and the s-tuple $\nu(t) = (X_1, ..., X_s)$ of subsets of $Pos(t) = Pos(pr_s(t))$ such that $u \in X_i$ if and only if $w[i] = 1$ where the symbol at position u in t is (f,w). Conversely, if $t \in T(F)$ and $(X_1, ..., X_s)$ is an s-tuple of sets of positions of t, then there is a unique term $t' \in T(F^{(s)})$ such that $pr_s(t') = t$ and $\nu(t') = (X_1, ..., X_s)$.We will denote this term by $t * (X_1, ..., X_s)$ or by $t * \overline{X}$.

A property $P(\overline{X})$ of sets of positions of terms over a signature F is characterized by the language $T_{P(\overline{X})}$ over $F^{(s)}$ defined as $\{t * \overline{X} \mid t \models P(\overline{X})\}$. A key fact about pr_s is that $T_{\exists \overline{X}.P(\overline{X})} = pr_s(T_{P(\overline{X})})$. A function α whose arguments are t and $\overline{X}$ such that $t \in T(F)$ and $\overline{X}$ is an s-tuple of positions of t, and whose values are in a set $\mathcal{D}$ corresponds to a function $\overline{\alpha} : T(F^{(s)}) \to \mathcal{D}$ such that $\overline{\alpha}(t * \overline{X}) = \alpha(t, \overline{X})$.

Tuples of sets of vertices
The operations defining clique-width form a countably infinite signature F_∞. Those using only labels in $[k]$ form F_k. The nullary symbols $\mathbf{a}$ (for vertex labels a) denote the vertices of the graph $G(t)$ defined by t. The same technique as above applies to tuples of sets of vertices of graphs defined by terms in $T(F_\infty)$. In particular, we define $F_\infty^{(s)}$ from F_∞ by replacing all nullary symbols $\mathbf{a}$ by the nullary symbols $(\mathbf{a}, w)$ for all $w \in \{0,1\}^s$. (The other symbols are not changed).

2 Polynomial-Time Fly-Automata

All automata run bottom-up (or frontier-to-root) on terms without ε-transitions.

Definitions 1: *Fly-automata recognizing languages.*

(a) Let F be a finite or infinite (effectively given) signature. A *fly-automaton* over F (in short, an FA over F) is a 4-tuple $\mathcal{A} = \langle F, Q_\mathcal{A}, \delta_\mathcal{A}, Acc_\mathcal{A} \rangle$ such that $Q_\mathcal{A}$ is the finite or infinite, effectively given set of *states*, $Acc_\mathcal{A}$ is a computable mapping $Q_\mathcal{A} \to \{True, False\}$ so that $Acc_\mathcal{A}^{-1}(True)$ is the set of *accepting states*, and $\delta_\mathcal{A}$ is a computable function that defines the *transition rules*: for each tuple $(f, q_1, \ldots, q_m)$ with $q_1, \ldots, q_m \in Q_\mathcal{A}$, $f \in F$, $\rho(f) = m \geq 0$, $\delta_\mathcal{A}(f, q_1, \ldots, q_m)$

is a finite set of states. We will write $f[q_1, \ldots, q_m] \to_{\mathcal{A}} q$ (and $f \to_{\mathcal{A}} q$ if f is nullary) to mean that $q \in \delta_{\mathcal{A}}(f, q_1, \ldots, q_m)$. Each set $\delta_{\mathcal{A}}(f, q_1, \ldots, q_m)$ is linearly ordered for some fixed (say lexicographic) linear order on Z^* where Z is the alphabet used to encode states. We say that $\mathcal{A}$ is *finite* if F and $Q_{\mathcal{A}}$ are finite. If furthermore, $Q_{\mathcal{A}}$, its accepting states and its transitions are listed in tables, it is called a *table-automaton*.

(b) *Runs* and *recognized languages* are defined as usual. A *deterministic* FA $\mathcal{A}$ ("deterministic" will mean "deterministic and complete") has a unique run on each term t, denoted by $run_{\mathcal{A},t}$; we let also $q_{\mathcal{A}}(t) := run_{\mathcal{A},t}(root_t)$. The mapping $q_{\mathcal{A}}$ is computable and the membership in $L(\mathcal{A})$ of a term t is decidable.

Every fly-automaton $\mathcal{A}$ over F can be *determinized* as follows. For every term $t \in T(F)$, we denote by $run^*_{\mathcal{A},t}$ the mapping: $Pos(t) \to \mathcal{P}_f(Q_{\mathcal{A}})$ that associates with every position u, the finite set of states of the form $r(root_{t/u})$ for some run r on the subterm t/u of t issued from u. The run of $\det(\mathcal{A})$ on t is called *the determinized run* of $\mathcal{A}$ and we define $ndeg_{\mathcal{A}}(t)$, the *nondeterminism degree of* $\mathcal{A}$ *on* t, as the maximal cardinality of $run^*_{\mathcal{A},t}(u)$ for u in $Pos(t)$. The mapping $run^*_{\mathcal{A},t}$ is computable and the membership in $L(\mathcal{A})$ of a term in $T(F)$ is decidable: clearly, $t \in L(\mathcal{A})$ if and only if the set $run^*_{\mathcal{A},t}(root_t)$ contains an accepting state.

Definitions 2: *Fly-automata computing functions.*
A fly-automaton over F *with output function* is a 4-tuple $\mathcal{A} = \langle F, Q_{\mathcal{A}}, \delta_{\mathcal{A}}, Out_{\mathcal{A}} \rangle$ as above except that $Acc_{\mathcal{A}}$ is replaced by a total and computable *output function* $Out_{\mathcal{A}}$: $Q_{\mathcal{A}} \to \mathcal{D}$ where $\mathcal{D}$ is an effectively given domain. If $\mathcal{A}$ is deterministic, the *function computed by* $\mathcal{A}$ is $Comp(\mathcal{A}) : T(F) \to \mathcal{D}$ such that $Comp(\mathcal{A})(t) := Out_{\mathcal{A}}(q_{\mathcal{A}}(t))$. If $\mathcal{A}$ is not deterministic, we let $\mathcal{B}$ be $\det(\mathcal{A})$ equipped with output function $Out_{\mathcal{B}}$:$\mathcal{P}_f(Q_{\mathcal{A}}) \to \mathcal{P}_f(\mathcal{D})$ such that $Out_{\mathcal{B}}(R) := \{Out_{\mathcal{A}}(q) \mid q \in R\}$. Then, we define $Comp(\det(\mathcal{A}))$ as $Comp(\mathcal{B})$. (In some cases, we may take a computable function $Out_{\mathcal{B}} : \mathcal{P}_f(Q_{\mathcal{A}}) \to \mathcal{D}'$ where $\mathcal{D}'$ is another effectively given domain).

Definitions 3: *Polynomial-time fly-automata and related notions*

(a) A fly-automaton $\mathcal{A}$ over a signature F, possibly with output, is a *polynomial-time fly-automaton* (a *P-FA*) if it is deterministic and there is a polynomial p such that its computation time on any term $t \in T(F)$ is at most $p(\|t\|)$, where $\|t\|$ is the size of t, written as a word; the operation symbols are encoded by words of non constant length. This time includes the time taken by the output function. We call p a *bounding polynomial* for $\mathcal{A}$.

(b) A fly-automaton $\mathcal{A}$ as above is an *XP fly-automaton* (an *XP-FA* in short) if, for each finite subsignature F' of F, $\mathcal{A} \restriction F'$ (the subautomaton of $\mathcal{A}$ induced by F') is a P-FA. It is an *FPT fly-automaton* (an *FPT-FA* in short) if, for each finite subsignature F' of F, $\mathcal{A} \restriction F'$ is a P-FA with bounding polynomial whose degree does not depend on F'. We have the inclusions of classes of automata:

P-FA $\subseteq$ FPT-FA $\subseteq$ XP-FA with equalities for finite signatures.

Lemma 4: Let $\mathcal{A}$ be a nondeterministic fly-automaton over a signature F.

(1) The fly-automaton $\det(\mathcal{A})$ is a P-FA if and only if there are polynomials $p_1, ..., p_4$ such that, in the determinized computation of $\mathcal{A}$ on any term $t \in T(F)$, $p_1(\|t\|)$ bounds the time for firing the next transition (and recognizing that there is no next transition), $p_2(\|t\|)$ bounds the size of a state, $p_3(\|t\|)$ bounds the time for checking if a state is accepting or for computing the output and $p_4(\|t\|)$ bounds the nondeterminism degree of $\mathcal{A}$ on t.

(2) The fly-automaton $\det(\mathcal{A})$ is an XP-FA if and only if, for each finite subsignature F' of F, there are polynomials $p_1, ..., p_4$ that bound as above the computations on terms in $T(F')$. It is an FPT-FA if and only if, for each finite subsignature F' of F, there are polynomials $p_1, ..., p_4$ that bound as above the computations on terms in $T(F')$ and whose degrees are independent of F'.

Definition 5: *Functions computable by fly-automata.*

A function $\alpha : T(F) \to \mathcal{D}$ *is P-FA computable* (or is a *P-FA function* for short) if it is computable by a P-FA over F that we have constructed or that we know how to construct by an algorithm. For a property P, we say that it is *P-FA decidable*. In this definition, F can be $H^{(s)}$ for some signature H, hence, a P-FA computable function or property can take as arguments, not only a term, but also a tuple of sets of positions or of vertices.

It is well-known that every MS property P of a term over a finite signature is P-FA decidable. The cardinality of a set and the height of a term are P-FA functions. We will construct an FPT-FA to check if a graph is regular (this not an MS property).

The mapping $\mathrm{Sat}X.P(X)$ is not P-FA computable, and not even XP-FA computable in general for the obvious reason that its output is not always of polynomial size (take $P(X)$ always true).

Proposition 6: Let F be a signature. Every **P**-computable (resp. **FPT**- computable or **XP**-computable) function α on $T(F)$ is computable by a P-FA (resp. by an FPT-FA or an XP-FA).

Hence, our three notions of FA may look trivial. Actually, we will be interested by giving effective constructions of P-FA, FPT-FA and XP-FA from logical expressions of functions and properties. These constructions will apply to properties that are *not MS expressible* but are decidable in polynomial time on graphs of bounded tree-width or clique-width.

3 Fly-Automata for Logically Defined Functions and Properties

Proposition 7: (1) If $\alpha_1, ..., \alpha_r$ are P-FA functions of same type and g is a **P**-computable function (or relation) of appropriate type, the function (or the property) $g \circ (\alpha_1, ..., \alpha_r)$ is P-FA computable (or decidable).

(2) If P and Q are P-FA properties of same type, then, so are $\neg P$, $P \vee Q$ and $P \wedge Q$.

(3) The same properties hold with FPT-FA and XP-FA.

The proof is based on easy constructions like taking a product of automata.

First-order constructions
We now consider the more delicate case of existential quantifications. We define one more construction: if $\alpha(\overline{X})$ is a function (relative to a term t), we define $\mathrm{SetVal}\overline{X}.\alpha(\overline{X})$ as the set of values $\alpha(\overline{X}) \neq \bot$ ($\bot$ stands for undefined) for all relevant tuples $\overline{X}$. We let $\exists x_1, ..., x_s.P(x_1, ..., x_s)$ (also written $\exists \overline{x}.P(\overline{x})$) abbreviate $\exists X_1, ..., X_s.(P(X_1, ..., X_s) \wedge Sgl(X_1) \wedge ... \wedge Sgl(X_s))$ (where $Sgl(X)$ means that X is singleton) and similarly, $\mathrm{SetVal}(x_1, ..., x_s).\alpha(x_1, ..., x_s)$ (also written $\mathrm{SetVal}\overline{x}.\alpha(\overline{x})$) is the set of well-defined values of $\alpha(X_1, ..., X_s)$ such that: $Sgl(X_1) \wedge ... \wedge Sgl(X_s)$.

Proposition 8: (1) If $P(\overline{X})$ is a P-FA property, then the property $\exists \overline{x}.P(\overline{x})$ is P-FA decidable and the functions $\mathrm{Sat}\overline{x}.P(\overline{x})$ and $\#\overline{x}.P(\overline{x})$ are P-FA computable.
(2) If $\alpha(\overline{X})$ is a P-FA function, then the function $\mathrm{SetVal}\overline{x}.\alpha(\overline{x})$ is P-FA computable.
(3) The same implications hold for the classes FPT-FA and XP-FA.

Proof Sketch: If $\mathcal{A}$, deterministic, checks $P(\overline{x})$, then the nondeterministic automaton $pr_s(\mathcal{A})$ defines $\exists \overline{x}.P(\overline{x})$ with nondeterminism degree bounded by the polynomial $p(n) = 1 + (n+1)^s$ that does not depend on $Sig(t)$. Hence $\det(pr_s(\mathcal{A}))$ is a P-FA, an FPT-FA or an XP-FA by Lemma 4 if $\mathcal{A}$ is so. □

These results remain valid if each condition $Sgl(X_i)$ is replaced by the condition $Card(X_i) = c_i$ or $Card(X_i) \leq c_i$ for fixed integers c_i. For example, we can compute

$$\#(X_1, ..., X_s).P(X_1, ..., X_s) \wedge Card(X_1) \leq c_1 \wedge ... \wedge Card(X_s) \leq c_s.$$

The exponents in the bounding polynomial become larger, but they still depend only on the numbers $c_1, ..., c_s$. This does not work for $Card(X_i) \geq c_i$ because the bound would not be polynomial.

Monadic second-order constructions
We recall that for finite signatures, the notions of P-FA, FPT-FA and XP-FA coincide. We let $P(\overline{X})$ be a property of terms in $T(F)$ with s set arguments and $\alpha(\overline{X})$ be similarly a function. The relabeling pr: $F^{(s)} \to F$ has a computable inverse. We consider infinite signatures F. Our main application will be to the infinite signature F_∞ that generates all finite graphs. We will use $Sig(t)$, the set of symbols that occur in a term t, as a parameter for FPT and XP algorithms. If $t \in T(F_\infty)$, this is equivalent to taking as parameter the minimal k such that $t \in T(F_k)$, hence, the clique-width of the considered graph because clique-width can be approximated in cubic time.

Definitions 9: *Multisets of tuples of numbers; a semi-ring.*

(a) If μ and μ' are two mappings $\mathbb{N}^s \to \mathbb{N}$, we define $\mu+\mu'$ and $\mu*\mu' : \mathbb{N}^s \to \mathbb{N}$ by:

$(\mu + \mu')(n_1, ..., n_s) := \mu(n_1, ..., n_s) + \mu'(n_1, ..., n_s)$, and
$(\mu * \mu')(n_1, ..., n_s) := \sum_{0 \leq p_i \leq n_i} \mu(p_1, ..., p_s).\mu'(n_1 - p_1, ..., n_s - p_s).$

$[\mathbb{N}^s \to \mathbb{N}]_f$ is the set of *finite* mappings: $\mathbb{N}^s \to \mathbb{N}$, i.e., with value 0 almost everywhere. The functions $\mu + \mu'$ and $\mu * \mu'$ are finite if μ and μ' are. The operations $+$ and $*$ are associative and commutative. The constant mapping: $\mathbb{N}^s \to \mathbb{N}$ with value 0 is denoted by $\mathbf{0}$. If $w \in \{0,1\}^s$, we let M_w: $\mathbb{N}^s \to \mathbb{N}$ be such that $M_w(\overline{n})$:=`if` $\overline{n} = w$ `then` 1 `else` 0. We have $\mu + \mathbf{0} = \mu$, $\mu * \mathbf{0} = \mathbf{0}$ and $\mu * M_{0...0} = \mu$. Since $*$ is distributive over $+$, we get that $\langle [\mathbb{N}^s \to \mathbb{N}]_f, +, *, \mathbf{0}, M_{0...0} \rangle$ is a semi-ring; $\mu \in [\mathbb{N}^s \to \mathbb{N}]_f$ is (represents) a finite multiset of s-tuples of integers.

(b) If E is a set and $Z \subseteq \mathcal{P}_f(E)^s$, we define $\mathrm{MSp}(Z)$ as the mapping: $\mathbb{N}^s \to \mathbb{N}$ such that $\mathrm{MSp}(Z)(n_1, ..., n_s)$ is the number of tuples $(X_1, \ldots, X_s) \in Z$ such that $n_i = |X_i|$ for each i, hence is the multiset $\{(|X_1|, ..., |X_s|) \mid (X_1, \ldots, X_s) \in Z\}$. If Z and $Z' \subseteq \mathcal{P}_f(E)^s$ are disjoint, then $\mathrm{MSp}(Z \cup Z') = \mathrm{MSp}(Z) + \mathrm{MSp}(Z')$. If $Z \subseteq \mathcal{P}_f(E)^s$ and $Z' \subseteq \mathcal{P}_f(E')^s$ with $E \cap E' = \emptyset$, and if $W = \{(X_1 \cup Y_1, \ldots, X_s \cup Y_s) \mid (X_1, \ldots, X_s) \in Z, (Y_1, \ldots, Y_s) \in Z'\}$, then $\mathrm{MSp}(W) = \mathrm{MSp}(Z) * \mathrm{MSp}(Z')$.

Definition 10: A fly-automaton $\mathcal{A}$ over F has an *FPT-bounded nondeterminism degree* (cf. p_4 in Lemma 4) if, for every $t \in T(F)$, $ndeg_{\mathcal{A}}(t) \leq f(Sig(t)) \cdot \|t\|^a$ for some fixed function f and constant a. It has an *XP-bounded nondeterminism degree* if $ndeg_{\mathcal{A}}(t) \leq f(Sig(t)) \cdot \|t\|^{g(Sig(t))}$ for some fixed functions f and g, equivalently, if $\mathcal{A} \restriction H$ has a polynomially bounded nondeterminism degree for each finite subsignature H of F.

Proposition 11: (1) If $P(\overline{X})$ is decided by a P-FA (resp. FPT-FA, resp. XP-FA) $\mathcal{A}$ over $F^{(s)}$ such that the FA $pr(\mathcal{A})$ has a polynomially bounded (resp. FPT-bounded, resp. XP-bounded) nondeterminism degree, then the property $\exists \overline{X}.P(\overline{X})$ is P-FA (resp. FPT-FA, resp. XP-FA) decidable, and the function $\mathrm{MSp}\overline{X}.P(\overline{X})$ is P-FA (resp. FPT-FA, resp. XP-FA) computable. These results also hold for $\mathrm{Sp}\overline{X}.P(\overline{X})$, $\#\overline{X}.P(\overline{X})$, $\mathrm{MinCard}X.P(X)$ and $\mathrm{MaxCard}X.P(X)$.

(2) If $\alpha(\overline{X})$ is computed by a P-FA (resp. FPT-FA, resp. XP-FA) $\mathcal{A}$ such that $pr(\mathcal{A})$ has a polynomially bounded (resp. FPT-bounded, resp. XP-bounded) nondeterminism degree, then the function $\mathrm{SetVal}\overline{X}.\alpha(\overline{X})$ is P-FA (resp. FPT-FA, resp. XP-FA) computable.

Proof Sketch: We start from a deterministic automaton $\mathcal{A}$ over $F^{(s)}$ that defines $P(\overline{X})$. Let $t \in T(F)$. For each state q and position u of t, we let $Z(q,u)$ be the set of s-tuples $\overline{X} \in (\mathcal{P}_f(Pos(t)/u))^s$ (where $Pos(t)/u$ is the set of positions of t below or equal to u; to be distinguished from $Pos(t/u)$) such that $run_{\mathcal{A},t*\overline{X}}(u) = q$. At the root, these sets define $\mathrm{Sat}\overline{X}.P(\overline{X})$. We extract information from $Z(q,u)$ and make it into an *attribute* of q. Depending on the case, this attribute may be a Boolean for emptiness of $Z(q,u)$ (for $\exists\overline{X}$), its cardinality (for $\#\overline{X}$), the multiset $\mathrm{MSp}(Z(q,u))$ (for $\mathrm{MSp}\overline{X}$). We focus on the last case.

The operation + sums the multisets coming from the different runs of $pr(\mathcal{A})$ that reach q at u. The operation $*$ combines the attributes at the sons of u in each run that reaches q at u. We obtain a nondeterministic automaton $\mathcal{B}$ whose states are pairs (q, α) where α is an attribute. Then $\det(\mathcal{B})$ is a deterministic FA that computes $\mathrm{MSp}\overline{X}.P(\overline{X})$ which is equal to the sum of the multisets $\mathrm{MSp}(Z(q, root_t))$ for q accepting.

We consider the case where $P(\overline{X})$ is MS expressible and F is finite. Then $\mathcal{A}$ is finite. A state of $\det(\mathcal{B})$ can be implemented as the finite set of tuples $(q, n_1, ..., n_s, m)$ such that $q \in Q_{\mathcal{A}}$, $m = \alpha(n_1, ..., n_s) \neq 0$ where α is the attribute of q (at some position). Then $\det(\mathcal{B})$ is a P-FA because its states (on a term with n nodes) can be encoded by words of length $\leq |Q_{\mathcal{A}}|.(n+1)^s.\log(2^{s.n}) = O(n^{s+1})$ (the numbers $\alpha(n_1, ..., n_s)$ being written in binary). Computing the transitions and the output takes polynomial time. The proof extends to infinite F as stated. The cases of $\mathrm{Sp}\overline{X}.P(\overline{X})$ etc. are even simpler because we have less information from $Z(q, u)$ to encode. The computation of $\mathrm{SetVal}\overline{X}.\alpha(\overline{X})$, the set of all values of α, is based on $\det(\mathcal{A})$.

If $\mathrm{Sat}\overline{X}.P(\overline{X}))$ is not empty, a P-FA (resp. an FPT-FA, resp. an XP-FA) can compute one of its tuples (but not all of them in general). □

4 Properties of Terms and Functions on Terms

The height of a term t can be computed by a P-FA $\mathcal{A}_{ht}$ whose states are positive integers (the state at u is $ht(t/u)$ where $ht(a) = 1$ for a nullary symbol a). A term t is *uniform* (this property is denoted by $Unif(t)$) if and only if any two leaves of its syntactic tree are at same distance to the root. This is equivalent to the condition that for every position u with sons u' and $u"$, the subterms t/u' and $t/u"$ have same height. The automaton $\mathcal{A}_{ht}$ can be modified into a P-FA $\mathcal{A}_{Unif}$ that decides uniformity. Its set of states is $\mathbb{N}_+ \cup \{Error\}$ and $q_{\mathcal{A}_{Unif}}(t) = ht(t)$ if t is uniform, $= Error$ if t is not uniform.

Definition 12: *An extension of MS logic on terms.*

We consider properties and functions constructed in the following way:

(a) We use free set variables X_1, ..., X_s (that will not be quantified), first-order (FO) variables, $y_1, ..., y_m$ and set terms over $X_1, ..., X_s, \{y_1\}, ..., \{y_m\}$.

(b) As basic properties, we use *Unif* and all properties P expressible by MS formulas (that can use other bound variables than X_1, ..., X_s, $y_1, ..., y_m$). As basic functions, we use ht, $Card$ (that yields the cardinality of a set of positions).

(c) We construct properties from already constructed properties P, Q, ... and from functions α,$\alpha_1, ..., \alpha_r, ...$ by the following compositions:

$P \wedge Q, P \vee Q, \neg P$,
$R \circ (\alpha_1, ..., \alpha_r)$ where R is an r-ary **P**-decidable relation on $\mathcal{D}$,
$P(S_1, ..., S_p)$ where $S_1, ..., S_p$ are set terms over $X_1, ..., X_s, \{y_1\}, ..., \{y_m\}$ (*set terms* are built with union, intersection and complementation; see [4]).
$\exists\overline{y}.P(\overline{y})$ where $\overline{y}$ is a tuple of variables among $y_1, ..., y_m$.

(d) Similarly, we construct functions in the following ways:

$g \circ (\alpha_1, ..., \alpha_r)$ where g is **P**-computable: $\mathcal{D} \to \mathcal{D}^r$,
$\alpha(S_1, ..., S_p)$ where $S_1, ..., S_p$ are set terms over $X_1, ..., X_s, \{y_1\}, ..., \{y_m\}$,
SetVal$\overline{y}.\alpha(\overline{y})$ (the set of values of α), $\#\overline{y}.P(\overline{y})$ and Sat$\overline{y}.P(\overline{y})$ where $\overline{y}$ is a tuple of variables among $y_1, ..., y_m$.

We assume that we have for R in (c) and g in (d) a certified polynomial-time algorithm. This is necessary to build automata. We denote by $\mathcal{PF}(F)$ the set of all these formulas.

Theorem 13: Every property (or function) defined by a formula of $\mathcal{PF}(F)$ is decidable (or computable) by a P-FA over F. Such an automaton can be constructed from automata for the basic properties and functions.

Our language $\mathcal{PF}(F)$ does not exhaust the possibilities of extension of MS logic that yield P-FA computable properties and functions. We can for example introduce a *relativized height* $ht(t, X)$ for $t \in T(F)$ and $X \subseteq Pos(t)$, defined as the maximal number of elements of X on a branch of the syntactic tree of t. However, we cannot use set quantifications.

5 Properties and Functions on Graphs

1. *Degrees of vertices*
For a directed graph G, we generalize the notion of *outdegree* by defining $e(X_1, X_2)$ as the number of edges from X_1 to X_2 if X_1 and X_2 are disjoint sets of vertices and as $\perp$ otherwise. Hence $e(\{x\}, V_G - \{x\})$ is the outdegree of x in G (all graphs are loop-free). Note that $e(X_1, X_2)$ *is not* of the form $\#Y.P(Y, X_1, X_2)$ for an MS property P as we do not allow edge set quantification.

We can define a deterministic FA $\mathcal{A}_k$ over $F_k^{(2)}$, intended to run on *irredundant terms* (such that no edge is defined twice, see [4]) written with labels in $C := [k]$. Its set of states is $(\mathbb{N} \times [C \to \mathbb{N}] \times [C \to \mathbb{N}]) \cup \{Error\}$. If $X \subseteq V_G$, we denote by λ_X the mapping that gives, for each a, the number of a-ports in X; if $X = V_G$, we denote it by λ_G. We want that $q_{\mathcal{A}_k}(t * (X_1, X_2)) = Error$ if $X_1 \cap X_2 \neq \emptyset$ and $q_{\mathcal{A}_k}(t * (X_1, X_2)) = (e(X_1, X_2), \lambda_{X_1}, \lambda_{X_2})$ otherwise. The transitions are easy to write. For example $\overrightarrow{add}_{a,b}[(m, \lambda_1, \lambda_2)] \to (m + \lambda_1(a).\lambda_2(b), \lambda_1, \lambda_2)$, is correct because t is assumed irredundant.

On a term that denotes a graph with n vertices, each state belongs to the set $([0, n^2] \times [C \to [0, n]] \times [C \to [0, n]]) \cup \{Error\}$ of cardinality less than $(n+1)^{2+2k}$, hence, has size $O(k.\log(n))$ (the integers m and the values of λ_1 and λ_2 are written in binary notation). Transitions and outputs can be computed in time $O(k.\log(n))$. Hence, $\mathcal{A}_k$ is a P-FA. We represent a function $\lambda : C \to \mathbb{N}$ by the set $\{(a, \lambda(a)) \mid \lambda(a) \neq 0\}$. This implies that $\mathcal{A}_k$ is a subautomaton of $\mathcal{A}_{k'}$ if $k < k'$. Hence, the union of the automata $\mathcal{A}_k$ is a P-FA $\mathcal{A}_\infty$ over $F_\infty^{(2)}$. For an undirected graph, we define $e(X_1, X_2)$ as the number of edges between X_1 and X_2 if X_1 and X_2 are disjoint and $\perp$ otherwise. The construction is similar.

2. *Regularity of a graph*
The regularity of an undirected graph is not MS expressible because the complete bipartite graph $K_{n,m}$ is regular if and only if $n = m$ and we apply the arguments of Proposition 5.13 of [7]. That a graph is not regular can be checked by a FA constructed from the formula $\exists X, Y.(P(X,Y) \wedge Sgl(X) \wedge Sgl(Y))$ where $P(X,Y)$ is the property $e(X, X^c) \neq e(Y, Y^c)$. By previous constructions, this property is P-FA decidable, and we can apply Proposition 14(1) to get a P-FA for checking regularity. However, we can construct directly a simpler P-FA without using an intermediate nondeterministic automaton. Its state at position u is *Error* if two a-ports of $G(t/u)$ have different degrees, and otherwise indicates, for each a, the number of a-ports and their common degree. In its run on a term t such that $G(t)$ has n vertices, less than $(n+1)^{2k}$ states occur and these states have size $O(k.\log(n))$. We get a P-FA $\mathcal{A}_{Reg[X]}$. The nondeterminism degree of $pr(\mathcal{A}_{Reg[X]})$ is bounded by $O((n+1)^{2k})$ where the exponent depends on the bound k to the clique-width.

The property $\exists X.(Card_{\leq p}(X) \wedge Reg[X^c])$ expressing that the considered graph becomes regular if we remove at most p vertices, is P-FA decidable by Proposition 11(1) and the remark following it. The function MaxCard$X.Reg[X]$ that defines the maximal cardinality of a regular induced subgraph of the considered graph is XP-FA computable. So is the property that the graph can be partitioned into two regular subgraphs, expressed by $\exists X.(Reg[X] \wedge Reg[X^c])$ (the proof uses the same propositions).

3. *Graph partition problems with numerical constraints*
Many partition problems (cf. also [14]) consist in finding $(X_1, \cdots, X_s)$, an s-tuple satisfying:

$$Partition(X_1, ..., X_s) \wedge P_1(X_1) \wedge ... \wedge P_s(X_s) \wedge R(|X_1|, ..., |X_s|),$$

where, $P_1, ..., P_s$ are properties of sets and R is a **P**-computable arithmetic condition. We may also wish to count the number of such partitions, or to find one that maximizes or minimizes the number $Ext(\overline{X}) := \Sigma_{1 \leq i < j \leq s} e(X_i, X_j)$ of *external edges*, *i.e.*, of edges not in the induced subgraphs $G[X_1], ..., G[X_s]$. This number is P-FA computable.

We can handle partitions in s planar induced subgraphs with an FPT-FA, however, its implementation does not seem doable (planarity is MS expressible, but the formula is complicated).

If $P_i(X_i)$ is stability for each i, (*i.e.*, the induced subgraphs have no edge), we get a *constrained coloring problem* of the form:

$$\exists X_1, ..., X_s.(Partition(X_1, ..., X_s) \wedge St[X_1] \wedge \ ... \ \wedge St[X_s] \wedge R(|X_1|, ..., |X_s|)).$$

An example is the notion of *equitable s-coloring*: condition $R(|X_1|, ..., |X_s|)$ is $\exists i \in [s].(|X_1| = ... = |X_{i-1}| \geq |X_i| = ... = |X_s| \geq |X_1| - 1)$, which means that any two color classes have same cardinality up to 1. The existence of an equitable 3-coloring is not trivial: it holds for the cycles but not for the graphs $K_{n,n}$ for large n. The existence of an equitable s-coloring is W[1]-hard for the

parameter defined as s plus the tree-width [11], hence presumably not FPT for this parameter. Our constructions yield, for each integer s, an FPT-FA for checking the existence of an equitable s-coloring for clique-width as parameter.

6 Implementations

Let $\mathcal{A}_{P(\overline{X})}$ be an automaton recognizing graphs with assignments of sets to the variables of $\overline{X}$. From $\mathcal{A}_{P(\overline{X})}$, we can obtain the automaton $\mathcal{A}_{\Gamma.P(\overline{X})}$ for $\Gamma \in \{\#\overline{X}, \mathrm{Sp}\overline{X}, \mathrm{MSp}\overline{X}, \ \mathrm{MinCard}X, \mathrm{MaxCard}X\}$ for graphs with no assignments. This is done in two steps. The first step is to associate to the automaton an *attribute mechanism* such that the automaton, instead of computing a state q, computes a state $[q, a]$ where a is the attribute to be computed according to Γ, for instance the number of runs yielding q for $\#\overline{X}.P(\overline{X})$. The attribute mechanism is composed of two functions: the first function applies to symbols and yields a function for computing the attribute obtained at $t = f(t_1, \ldots, t_p)$ from the ones obtained for $t_1, \ldots, t_p$ in the deterministic case; sometimes this function is the same for all symbols as for the counting case. The second function is for combining several attributes of identical states accessed with different runs. For $\#\overline{X}.P(\overline{X})$, the first function is the addition function for all symbols and the second is the multiplication function. (The case of $\exists\overline{X}.P(\overline{X})$ is handled by determinizing $pr(\mathcal{A}_{P(\overline{X})})$.)

This can be applied for counting the s-colorings of a graph or for constructing "special" colorings. From an appropriate $\mathcal{A}$, we can obtain an automaton that computes the number of s-colorings as the number of runs of $\mathcal{A}$ on the representing term. This is done by using the attribute mechanism for counting runs. For classic graphs such as Petersen's, with known chromatic polynomials, we can verify the computation. We can also count acyclic-colorings [4]. The number of 4-acyclic colorings of Petersen's graph is 10800. The number of 3-acyclic-colorings of McGee's graph is 57024. We also provide a mechanism for enumerating colorings (more generally, satisfying assignments) [9]. It is also useful for determining "quickly" the existence a coloring (but not for counting them).

7 Conclusion and References

In this communication, we have given logic based methods for proving the existence of FPT and XP algorithms that check properties or compute functions on terms and on graphs defined by terms. These constructions are currently under implementation. They are quite general and flexible and so, they do not give necessarily the best possible time complexities. They generalize constructions of [1,2,8]. Detailed definitions and proofs are in [5]. Implementation issues are described in [6,9].

References

1. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. J. Algorithms 12, 308–340 (1991)
2. Courcelle, B., Mosbah, M.: Monadic second-order evaluations on tree-decomposable graphs. Theor. Comput. Sci. 109, 49–82 (1993)
3. Courcelle, B.: On the model-checking of monadic second-order formulas with edge set quantifications. Discrete Applied Mathematics 160, 866–887 (2012)
4. Courcelle, B., Durand, I.: Automata for the verification of monadic second-order graph properties. J. Applied Logic 10, 368–409 (2012)
5. Courcelle, B., Durand, I.: Computations by fly-automata beyond monadic second-order logic (preprint, June 2013)
6. Courcelle, B., Durand, I.: Infinite transducers on terms denoting graphs In: Proceedings of the 6th European Lisp Symposium, Madrid (June 2013)
7. Courcelle, B., Engelfriet, J.: Graph structure and monadic second-order logic, a language theoretic approach. Encyclopedia of mathematics and its application, vol. 138. Cambridge University Press (June 2012)
8. Courcelle, B., Makowsky, J., Rotics, U.: Linear-time solvable optimization problems on graphs of bounded clique-width. Theory Comput. Syst. 33, 125–150 (2000)
9. Durand, I.: Object enumeration. In: Proceedings of the 5th European LISP Conference, Zadar, Croatia, pp. 43–57 (May 2012)
10. Downey, R., Fellows, M.: Parameterized complexity. Springer (1999)
11. Fellows, M., et al.: On the complexity of some colorful problems parameterized by treewidth. Inf. Comput. 209, 143–153 (2011)
12. Flum, J., Grohe, M.: Parametrized complexity theory. Springer (2006)
13. Frick, M., Grohe, M.: The complexity of first-order and monadic second-order logic revisited. Ann. Pure Appl. Logic 130, 3–31 (2004)
14. Rao, M.: MSOL partitioning problems on graphs of bounded treewidth and clique-width. Theor. Comput. Sci. 377, 260–267 (2007)
15. Reinhardt, K.: 13 the complexity of translating logic to finite automata. In: Grädel, E., Thomas, W., Wilke, T. (eds.) Automata, Logics, and Infinite Games. LNCS, vol. 2500, pp. 231–238. Springer, Heidelberg (2002)
16. Veanes, M., Bjørner, N.: Symbolic automata: The toolkit. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 472–477. Springer, Heidelberg (2012)