Fly-automata for checking MSO 2 graph properties

Bruno Courcelle

▶ **To cite this version:**

Bruno Courcelle. Fly-automata for checking MSO 2 graph properties. 2015. hal-01234622v2

HAL Id: hal-01234622
https://hal.science/hal-01234622v2

Preprint submitted on 8 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fly-automata for checking MSO_2 graph properties.

Bruno Courcelle
LABRI*, Bordeaux University, Talence, France

December 8, 2015

Abstract

A more descriptive but too long title would be : *Constructing fly-automata to check properties of graphs of bounded tree-width expressed by monadic second-order formulas written with edge quantifications*. Such properties are called MSO_2 in short. *Fly-automata* (*FA*) run bottom-up on terms denoting graphs and compute "on the fly" the necessary states and transitions instead of looking into huge, actually unimplementable tables. In previous works, we have constructed FA that process terms denoting graphs of bounded clique-width, in order to check their monadic second-order (MSO) properties (expressed by formulas without edge quantifications). Here, we adapt these FA to *incidence graphs*, so that they can check MSO_2 properties of graphs of bounded tree-width. This is possible because: (1) an MSO_2 property of a graph is nothing but an MSO property of its incidence graph and (2) the clique-width of the incidence graph of a graph is linearly bounded in terms of its tree-width. Our constructions are actually implementable and usable. We detail concrete constructions of automata in this perspective.

keywords: monadic second-order logic, edge quantification, tree-width, clique-width, algorithmic meta-theorem, fly-automaton.

1 Introduction

Graphs are finite and, either directed or undirected. Our goal is to check their *monadic second-order* (*MSO*) properties by using finite automata running on the terms that denote input graphs, and to obtain in this way *fixed-parameter tractable* (*FPT*) algorithms whose parameters are tree-width or clique-width. We want these automata to be constructable automatically from logical formulas and practically usable. We recall the following algorithmic meta-theorem [7,10][1]; the notions it uses will be reviewed in Section 2.

*This work has been supported by the French National Research Agency (ANR) in the IdEx Bordeaux program "Investments for the future", CPU, ANR-10-IDEX-03-02.

[1]The presentation in these two works is much better than those in the usually quoted original references [3, 11].

Theorem 1 : (a) For every integer k and every MSO sentence φ, there exists a linear time algorithm that checks the validity of φ in any graph of clique-width at most k given by a relevant term.

(b) For every integer k and every MSO_2 sentence φ, there exists a linear time algorithm that checks the validity of φ in any graph of tree-width at most k given by a relevant tree-decomposition.

A *sentence* is a logical formula without free variables. An MSO sentence can only use quantifications over vertices and sets of vertices, whereas an MSO_2 sentence can also use quantifications over edges and sets of edges. The *incidence graph* $Inc(G)$ of a graph G is a bipartite graph constructed as follows: its vertices are those of G together with new vertices representing its edges; its adjacency relation is the incidence relation of G (relating an edge and its ends). Assertion (b)[2] is actually a consequence of (a) because:

(1) an MSO_2 property of a graph G is nothing but an MSO property of its incidence graph $Inc(G)$ and

(2) if G has tree-width k, then $Inc(G)$ has clique-width at most $f(k)$ for some fixed linear function f,

(3) a tree-decomposition of G of width k can be converted in linear time (for fixed k) into a clique-width term of width at most $f(k)$ that defines $Inc(G)$.

Point (1) is just a matter of definitions. Point (2), in particular the fact that f is linear ($f(k) \leq 2k+5$, see details in Section 3) together with the linear time transformation of (3) make practically usable this reduction of (b) to (a).

Automata constructions.

The classical proof of Assertion (a) constructs from an MSO sentence φ and a positive integer k a finite automaton $\mathcal{A}_{\varphi,k}$ that takes as input a term t of width at most k that denotes an input graph assumed of clique-width $\leq k$. This automaton recognizes the terms that denote graphs satisfying φ. The construction is done by induction on the structure of φ. First, one constructs automata for the atomic formulas that are of only two kinds[3] : $X \subseteq Y$ (set inclusion) and $edg(X,Y)$ (meaning that $X = \{x\}, Y = \{y\}$ and there is an edge from x to y, or between them if the graph is undirected). As we will recall, MSO formulas can be written with set variables only and these two types of atomic formulas. From automata $\mathcal{A}_{\varphi,k}$ and $\mathcal{A}_{\psi,k}$, one can build automata for $\mathcal{A}_{\varphi\vee\psi,k}$ and $\mathcal{A}_{\varphi\wedge\psi,k}$ by taking the product of $\mathcal{A}_{\varphi,k}$ and $\mathcal{A}_{\psi,k}$ equipped with appropriate accepting states. Other constructions handle negation and existential quantification. See [7, 10].

It is actually useful to precompute automata $\mathcal{A}_{\varphi,k}$ for some formulas φ expressing basic MSO properties, such as $Partition(X_1, ..., X_p)$ (meaning that $X_1, ..., X_p$ is a partition of the vertex set), *stability* of X (denoted by $St(X)$ and meaning that there are no edges between vertices in X, equivalently, that X is an

[2] By a result of Bodlaender (see [1, 12, 13]), a tree-decomposition of G of width k can be computed in linear time if there exists one. Hence the variant of (b) where a tree-decomposition is not given but must be computed also holds, but this variant is not a consequence of (a). Furthermore, the linear time decomposition algorithm is not practically implementable.

[3] Automata $\mathcal{A}_{\varphi,k}$ can also be defined for formulas φ with free variables. See Theorem 5.

independent set), connectedness of the considered graph, existence of directed or undirected cycles, degree bounds etc. See [7]. The logical expression of a property can be made simpler as we allow, in some sense, more atomic formulas. For example, 3-colorability is expressed by : $\exists X_1, X_2, X_3.(Partition(X_1, X_2, X_3) \wedge St(X_1) \wedge St(X_2) \wedge St(X_3))$.

However, even with this technique, this construction is intractable because $\mathcal{A}_{\varphi,k}$ has in most cases, even for $k = 2$, so many states that it cannot be implemented in the classical way. This is not avoidable [15]. The notion of a *fly-automaton* (an FA in short) remedies to this fact. Its states are not listed in huge tables. Although numerous, they have a common syntactic structure and can be described by words[4]. The necessary transitions are not stored in tables but computed by (small) programs. A deterministic FA having $2^{2^{10}}$ states needs and computes only 100 states on a term of size 100. The maximum size of a state (the number of bits for encoding it) used on an input term is more important to bound the computation time than the total number of states. Implementation of FA has been tested in significant cases [7, 8].

The present contribution :

Our objective is to apply to Assertion (b) the tools developped in [7, 8]. However, the reduction of (b) to (a) necessitates some work on automata. For instance, the clique-width operations allow to write terms that do not define incidence graphs. For example, if some vertex intended to represent an edge has degree more than 2, the defined graph is not an incidence graph. We will define an automaton $\mathcal{A}_{CT}$ to check whether a given term is *correct*, i.e, defines an incidence graph.

Furthermore, in MSO_2 logic seen as MSO logic on incidence graphs, the property $edg(X, Y)$ is no longer atomic. It is expressed by:

> "there exists a vertex representing an edge that is adjacent to the unique vertex of X and to the unique vertex of Y",

hence, the automaton for this formula is more complicated than the one for $edg(X, Y)$ in the case of MSO logic because of the quantification on vertices representing edges. We will detail the construction of such an FA for $edg(X, Y)$ and of other FA for related properties such as domination. We will also construct FA for checking Hamiltonicity, a graph property that is MSO_2-expressible but not MSO-expressible.

We consider *directed graphs* because they are in general algorithmically more difficult to handle than undirected ones. In the present setting they are no more

[4]FA can have infinitely many states: a state can record the (unbounded) number of occurrences of a particular symbol. We can thus construct fly-automata that check properties that are not MSO expressible (for example that a graph is regular or can be partitioned into p disjoint regular graphs). These automata yield FPT or XP algorithms [12,13] for clique-width as parameter. By equipping fly-automata with output functions, we can make them compute values attached to graphs: for example, assuming that the input graph is s-colorable, the minimum size of X_1 in an s-coloring $(X_1, \ldots, X_s)$. We can compute the number of p-vertex colorings, and also the number of so-called of *acyclic* 4-colorings of Petersen's graph: 10800. The number of acyclic 3-colorings of McGee's graph is 57024. See [8].

difficult to deal with, and it is easy to simplify our automata, constructed for directed graphs so as they handle undirected graphs.

Our method, that consists in computing automata from logical formulas, yields usable algorithms very quickly as it consists in combining predefined automata by means of a few standard procedures operating on FA. It has been implemented[5]. The produced algorithms follow the general scheme of dynamic programming over graph decomposition, however, specialized algorithms for particular problems designed with *ad hoc* data structures can be quicker. Our purpose is to obtain by meta-algorithmic tools, usable algorithms with reasonable, but not necessarly optimal, time complexity.

Section 2 reviews definitions and constructions of automata. Although this article is a continuation of [7], we recall most definitions to make it readable independently. In Section 3, we adapt definitions to incidence graphs. Our constructions of automata for the correctness of terms and for properties based on adjacency are in Section 4. Other properties, in particular Hamiltonicity, are discussed in Section 5. We review related work in our conclusion.

2 Definitions and background constructions.

First, some notation : $\mathcal{P}(X)$ denotes the powerset of a set X, $\mathcal{P}_f(X)$, the set of its finite subsets and $\mathcal{P}_{\leq m}(X)$ the set of its subsets of cardinality at most m and $[X \to Y]_f$ the set of partial functions : $X \to Y$ having a finite domain.

A *(functional) signature* is a set F of function symbols, where $f \in F$ has arity $\rho(f) \in \mathbb{N}$ and $T(F)$ is the set of (finite) terms over F. The set of positions of a term t is $Pos(t)$ and positions are formally denoted by Dewey words. They are considered as the nodes of a labelled tree, and the terminology of trees will be used for expressing properties of positions in terms. The positions of $t = f(a, g(a, b))$ are the words ε that it is the root of t, also denoted by $root_t$ and the unique occurrence of f, 1 and 21, the occurrences of a, 2 the occurrence of g and 22, the occurrence of b. If t' is the subterm t/u of t issued from position u, then $Pos(t')$ is the set of words w such that $uw \in Pos(t)$. Positions of t are partially ordered by $\leq_t$ such that $u \leq_t v$ if and only if v is a prefix of u, i.e., v is an ancestor of u or is equal to it.

By a *language*, we mean a set of words or of terms, not to be confused with a *logical language* such as *monadic second-order* (*MSO* in short) logic.

Graphs and MSO logic.

A *simple graph* is, here, a finite directed graph without parallel edges and loops. These graphs will be used as *incidence graphs* of more general graphs, to be defined in the next section. We identify a simple graph G to the logical

[5]The system AUTOGRAPH that builds and runs FA is written in LISP [6] and http://dept-info.labri.u-bordeaux.fr/~idurand/autograph

structure $\langle V_G, edg_G\rangle$ whose domain is V_G, the set of vertices, and where edg_G is the binary relation such that $(x,y) \in edg_G$ if and only if there is an edge from x to y, which we denote by $x \rightarrow_G y$, or by $x \rightarrow y$ if G is clear from the context.

MSO logic uses individual variables $(x, y, z, ...)$ to denote vertices, and set variables $(X, Y, Z, ...)$ to denote set of vertices. Quantifications over binary relations is not allowed. Rather than giving a formal syntax (see [10]), we take the example of 3-vertex colorability, expressed by the MSO formula $\exists X, Y.\alpha(X,Y)$ where $\alpha(X,Y)$ is the formula

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

The formula $\alpha(X,Y)$ expresses that X, Y and $V_G - (X \cup Y)$ are the three color classes of a 3-vertex coloring of G. More generally, p-vertex colorability is expressible in a similar way. Other MSO expressible graph properties are connectedness, strong connectedness, planarity (via forbidden minors) and the existence of directed cycles.

Every MSO formula can be written so as to use only set variables and the atomic formulas $X \subseteq Y$ (for set inclusion) and $edg(X,Y)$ (meaning that $X = \{x\}, Y = \{y\}$ and $x \rightarrow y$): for doing that, we replace the atomic formula $x \in Y$ by $X \subseteq Y$ where X denotes a singleton set. The property that X is singleton, denoted by $Sgl(X)$, can be expressed itself in terms of inclusions. It is however useful to allow $X = Y, X = \emptyset$, $Sgl(X)$, $Partition(X_1, ..., X_p)$ (meaning that $(X_1, ..., X_p)$ is a partition of the vertex set, where some sets X_i may be empty) as atomic formulas. Furthermore, universal quantifications $\forall X.\varphi$ are replaced by $\neg\exists X.\neg\varphi$. These syntactic constraints facilitate the construction of automata. Details are in [10].

A *sentence* is a formula without free variables. It expresses a property of the considered graph.

Clique-width

Clique-width is a graph complexity measure, comparable to tree-width, that is defined from operations that construct simple graphs equipped with vertex labels. Let C be a finite or countable set of labels. A *C-graph* is a triple $G = \langle V_G, edg_G, \pi_G\rangle$ where π_G is a mapping: $V_G \rightarrow C$. If $\pi_G(x) = a$ we say to be short that x is an *a-vertex*.

We let F_C be the following set of operations on C-graphs:

$\oplus$ is the union of two disjoint C-graphs,

$relab_{a\rightarrow b}$, for $a \neq b$, $a, b \in C$, is the the unary operation that changes every vertex label a into b,

$\overrightarrow{add}_{a,b}$, for $a \neq b$, $a, b \in C$, is the unary operation that adds an edge from each a-vertex x to each b-vertex y (unless we already have an edge from x to y),

$\varnothing$ is a nullary symbol denoting the empty graph,

and for each $a \in C$, the nullary symbol $\mathbf{a}$ denotes an isolated a-vertex.

The set F_C is finite if C is finite.

Every term t in $T(F_C)$ defines a C-graph $G = val(t)$ (a formal definition is in [7, 10]). Its vertex set is the set of positions in t of the nullary symbols $\mathbf{a}$, $a \in C$. The clique-width of a graph G, denoted by $cwd(G)$, is the least cardinality of a set of labels C such that G is isomorphic, up to vertex labels, to $val(t)$ for some t in $T(F_C)$. It is clear that $cwd(G) \leq |V_G|$.

Let t' be a subterm t/u of $t \in T(F_C)$ for $u \in Pos(t)$. The graph $val(t')$ is, up to vertex labels, isomorphic to a subgraph G' of $G = val(t)$. More precisely, let $i_u : Pos(t') \to Pos(t)$ map w to uw (we recall that positions are Dewey words). Then $i_u(Pos(t')) = \{v \in Pos(t) \mid v \leq_t u\}$ is the set of vertices of G' and i_u is the isomorphism : $val(t') \to G'$. If $x \to_{val(t')} y$, then $i_u(x) \to_G i_u(y)$, however we may have $i_u(x) \to_G i_u(y)$ without having $x \to_{val(t')} y$ because an edge from $i_u(x)$ to $i_u(y)$ can be added to G' by an operation $\overrightarrow{add}_{a,b}$ above u in t. To simplify notation, we will use in the above case the following:

Convention about subterms: we will forget i_u and consider $val(t')$ as identical to the subgraph G' of $G = val(t)$.

Hence, a node w of $val(t')$ will be identified to the node $i_u(w) = uw$ of G.

A term t in $T(F_C)$ is *irredundant* if an operation $\overrightarrow{add}_{a,b}$ never "tries to create" an edge from an a-vertex x to a b-vertex y such that we already have $x \to y$. In particular, t has no subterm of the form $\overrightarrow{add}_{a,b}(t')$ such that $val(t')$ has an a-vertex x and a b-vertex y such that $x \to_{val(t')} y$. (The formal definition is more complicated because of relabellings). Every term can be made irredundant by an easy preprocessing [7, 10] consisting in removing some operations $\overrightarrow{add}_{a,b}$.

Lemma 2 : Let $t \in T(F_C)$ be irredundant, $u \in Pos(t), t' = t/u$. If a vertex x of $val(t')$ has indegree p in this graph, then it has indegree $p + q$ in $val(t)$, where q is nonnegative and depends only on t, u and the label of x in $val(t')$. The same holds for outdegrees.

This statement uses our *Convention about subterms*. Its proof is straightforward from the definition (Definition 5 of [7]). It follows that if t, t', x, q are as in the statement and x' is another vertex of $val(t')$ with same label and of indegree p', then it has indegree $p' + q$ in $val(t)$.

We denote by $twd(G)$ the tree-width of a graph G [1, 10, 12, 13].

Proposition 3 : For every simple graph G, we have $cwd(G) \leq 2^{2twd(G)+2} + 1$, and $cwd(G)$ is not polynomially bounded in terms of $twd(G)$, as it can be at least 2^{k-1} for some graphs of tree-width $2k$.

References for the proofs and comments are in [10], Proposition 2.114.

Finally, we explain how a term in $T(F_C)$ can be enriched so as to define, not only a graph, but also vertex sets $X_1, ..., X_p$ of this graph. We replace in F_C each nullary symbol $\mathbf{a}$ by the nullary symbols $(\mathbf{a}, w)$ for all $w \in \{0,1\}^p$. We obtain a set of operations $F_C^{(p)}$.

Let $t \in T(F_C)$, $X_1, ..., X_p \subseteq V_{val(t)}$ and $x \in V_{val(t)}$. We let $w(x)$ be the sequence in $\{0,1\}^p$ whose i-th element $w(x)[i]$ is 1 if $x \in X_i$ and 0 otherwise. We replace in t a nullary symbol $\mathbf{a}$ at position u by $(\mathbf{a}, w(u))$ (u is a vertex of $val(t)$). We obtain a term in $T(F_C^{(p)})$ that defines the graph $val(t)$ and also the p-tuple $(X_1, ..., X_p)$. We denote this term by $t * (X_1, ..., X_p)$. Conversely, every term t' in $T(F_C^{(p)})$ is of this form, and we will also denote by $val(t')$ the graph $val(t)$.

If P is a property of graphs[6], then $L_{P,C}$ is defined as the set of terms $t \in T(F_C)$ such that $val(t)$ satisfies P. More generally, if $P(X_1, ..., X_p)$ is a property of vertex sets $X_1, ..., X_p$ of graphs, then $L_{P(X_1,...,X_p),C}$ is the set of terms

$t * (X_1, ..., X_p) \in T(F_C^{(p)})$ such that $P(X_1, ..., X_p)$ holds in $val(t)$. If $P(X_1, ..., X_p)$ is expressed by an MSO formula $\varphi(X_1, ..., X_p)$, we denote by $L_{\varphi(X_1,...,X_p),C}$ the language $L_{P(X_1,...,X_p),C}$.

Automata.

If $\varphi(X_1, ..., X_p)$ is an MSO formula and C is finite, then $L_{\varphi(X_1,...,X_p),C}$ is a recognizable language, hence, is definable by a finite automaton. However, if C is infinite, $L_{\varphi(X_1,...,X_p),C}$ can be defined by an infinite automaton that is usable in algorithms.

Definition 4: *Fly-automata.*

(a) A set is *effectively given* if, either it is finite and the list of its elements is known, or it is countably infinite and its elements are bijectively encoded by words over $\{0,1\}$ (or another finite alphabet) that form a decidable set. Through this encoding, we have the notion of *computable functions* on effectively given sets. If $\mathcal{D}$ is effectively given, then so are $\mathcal{D}^s$ for $s \geq 2$, $\mathcal{P}_f(\mathcal{D})$, the set of finite subsets of $\mathcal{D}$ and the set $[\mathcal{D}' \to \mathcal{D}]_f$ of partial functions having a finite domain, where $\mathcal{D}'$ is effectively given. A signature F is *effectively given* if the set F is effectively given, the arity mapping ρ is computable and the least upper bound $\rho(F)$ of the arities is finite.

(b) Let F be an 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 an effectively given set called the set of *states*, $Acc_\mathcal{A}$ is a decidable subset of $Q_\mathcal{A}$ called the

[6]Saying that P is a property of graphs means that it is invariant under isomorphisms and arbitrary vertex relabellings. Labels are used for constructing graphs, but at the end, they are forgotten.

set of *accepting states*[7], and $\delta_{\mathcal{A}}$ is a computable function such that, for each tuple $(f, q_1, \ldots, q_{\rho(f)})$ such that $q_1, \ldots, q_{\rho(f)} \in Q_{\mathcal{A}}$ and $f \in F$, the set of states $\delta_{\mathcal{A}}(f, q_1, \ldots, q_{\rho(f)})$ is finite. The *transitions* are the pairs $f[q_1, \ldots, q_{\rho(f)}] \to_{\mathcal{A}} q$ (and $f \to_{\mathcal{A}} q$ if f is nullary) such that $q \in \delta_{\mathcal{A}}(f, q_1, \ldots, q_{\rho(f)})$. We say that $f[q_1, \ldots, q_{\rho(f)}] \to_{\mathcal{A}} q$ is a transition that *yields* q. We say that $\mathcal{A}$ is *finite* if F and $Q_{\mathcal{A}}$ are finite, and we get the usual notion of a finite automaton.

(c) A *run* of an FA $\mathcal{A}$ on a term $t \in T(F)$ is a mapping $r : Pos(t) \to Q_{\mathcal{A}}$ such that :

> if u is an occurrence of a function symbol $f \in F$ and $u_1, ..., u_{\rho(f)}$ is the sequence of sons of u, then $f[r(u_1), \ldots, r(u_{\rho(f)})] \to_{\mathcal{A}} r(u)$; (if $\rho(f) = 0$, the condition reads $f \to_{\mathcal{A}} r(u)$).

For $q \in Q_{\mathcal{A}}$, we define $L(\mathcal{A}, q)$ as the set of terms t in $T(F)$ on which there is a run r of $\mathcal{A}$ such that $r(root_t) = q$. A run r on t is *accepting* if the state $r(root_t)$ is accepting. We define $L(\mathcal{A}) := \bigcup\{L(\mathcal{A}, q) \mid q \in Acc_{\mathcal{A}}\} \subseteq T(F)$. It is the *language accepted* (or *recognized*) by $\mathcal{A}$. A state q is *accessible* if $L(\mathcal{A}, q) \neq \emptyset$. We denote by $Q_{\mathcal{A}} \restriction t$ the set of states that occur in the runs on t and on its subterms, and by $Q_{\mathcal{A}} \restriction L$ the union of the sets $Q_{\mathcal{A}} \restriction t$ for $t \in L \subseteq T(F)$.

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. This mapping is computable and the membership in $L(\mathcal{A})$ of a term in $T(F)$ is decidable because $t \in L(\mathcal{A})$ if and only if the set $run^*_{\mathcal{A},t}(root_t)$ contains an accepting state. Hence, whether all states of an FA are accessible or not does not affect the membership algorithm: the inaccessible states simply never appear in any run. There is no need to try to remove them, which is actually impossible in general, unless if $\mathcal{A}$ is finite. Removing them in the case of a "small" finite automaton permits to reduce the size of small the transition table.

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 $Pos(t)$. We have $ndeg_{\mathcal{A}}(t) \leq |Q_{\mathcal{A}} \restriction t|$.

A *sink* is a state s such that, for every transition $f[q_1, \ldots, q_m] \to_{\mathcal{A}} q$, we have $q = s$ if $q_i = s$ for some i. If F has at least one symbol of arity at least 2, then an automaton has at most one sink. A state named *Error* (resp. *Success*) will always be a nonaccepting (resp. accepting) sink.

We say that an FA $\mathcal{B} = \langle H, Q_{\mathcal{B}}, \delta_{\mathcal{B}}, Acc_{\mathcal{B}}\rangle$ is a *subautomaton* of an FA $\mathcal{A}$ if $H \subseteq F$, $Q_{\mathcal{B}} \subseteq Q_{\mathcal{A}}$, $Acc_{\mathcal{B}} = Acc_{\mathcal{A}} \cap Q_{\mathcal{B}}$ and $\delta_{\mathcal{B}}(f, q_1, \ldots, q_{\rho(f)}) = \delta_{\mathcal{A}}(f, q_1, \ldots, q_{\rho(f)})$ if $f \in H$ and $q_1, \ldots, q_{\rho(f)} \in Q_{\mathcal{B}}$. Then $L(\mathcal{B}) = L(\mathcal{A}) \cap T(H)$.

(d) *Deterministic automata.* An FA $\mathcal{A}$ is *deterministic* (implicitly, *and complete*) and if all sets $\delta_{\mathcal{A}}(f, q_1, \ldots, q_{\rho(f)})$ have cardinality 1. A deterministic

[7] Fly-automata can also compute functions if one replaces the accepting states by a computable mapping from states to some effective domain. See [8].

FA $\mathcal{A}$ has, on each term t, a unique run denoted by $run_{\mathcal{A},t}$; we let $q_{\mathcal{A}}(t) := run_{\mathcal{A},t}(root_t)$. The mapping $q_{\mathcal{A}}$ is computable.

The computation time of $\mathcal{A}$ on a term t is bounded by $p.|t|$ where p bounds the time used for computing a transition.

For every fly-automaton $\mathcal{A}$, there exists a deteministic FA $\mathcal{B}$ denoted by $\det(\mathcal{A})$ such that $Q_{\mathcal{B}} = \mathcal{P}_f(Q_{\mathcal{A}})$, $run_{\mathcal{B},t} = run^*_{\mathcal{A},t}$ and $L(\mathcal{B}) = L(\mathcal{A})$. (See [7], Proposition 45(2)). A run of $\mathcal{B}$ on a term t, that is, a bottom-up computation of $run_{\mathcal{B},t}$, is called *the determinized run* of $\mathcal{A}$. The maximal size of a state of $\det(\mathcal{A})$ on a term t is bounded by $ndeg_{\mathcal{A}}(t).s$ where s is the maximal size of a state in $Q_{\mathcal{A}} \restriction t$.

If the state *Error* is met at any point of the computation of a deterministic FA, the term can be immediately rejected. If the state *Success* is met at any point of the bottom-up computation of an FA, then the term can be immediately accepted. Hence, using sinks *Success* and *Error* in this way can shorten some computations.

(e) *Direct and inverse images of automata.*

Let H and F be effectively given signatures. Let $h : H \to F$ be a *relabelling*, i.e., an arity preserving mapping having a *computable inverse*, that is, such that $h^{-1}(f) \subseteq H$ is finite and computable for each f. Let h be extended from $T(H)$ to $T(F)$ in the obvious way. If $L \subseteq T(H)$, then $h(L) := \{h(t) \mid t \in L\}$. If $\mathcal{A}$ is an FA over H, then $h(\mathcal{A})$ is the fly-automaton over F ([7], Proposition 45) obtained from $\mathcal{A}$ by replacing each transition $f[q_1, \cdots, q_{\rho(f)}] \to_{\mathcal{A}} q$ by $h(f)[q_1, \cdots, q_{\rho(f)}] \to q$. We say that $h(\mathcal{A})$ is the *image of* $\mathcal{A}$ under h. It is not deterministic in general, even if $\mathcal{A}$ is. We have $h(L(\mathcal{A})) = L(h(\mathcal{A}))$ (because $h(\mathcal{A})$ has the same accepting states as A).

Let now $h : T(H) \to T(F)$ be a computable relabelling. If $K \subseteq T(F)$, then $h^{-1}(K) := \{t \in T(H) \mid h(t) \in K\}$. If $\mathcal{A}$ is an FA over F, we let $h^{-1}(\mathcal{A})$ be the FA over H with transitions of the form $f[q_1, \cdots, q_{\rho(f)}] \to q$ whenever $h(f)[q_1, \cdots, q_{\rho(f)}] \to_{\mathcal{A}} q$. We have $L(h^{-1}(\mathcal{A})) = h^{-1}(L(\mathcal{A}))$. We call $h^{-1}(\mathcal{A})$ the *inverse image* of $\mathcal{A}$ under h ([7], Definition 17(h)). It is deterministic if $\mathcal{A}$ is so.

Automata from checking graph properties.

Theorem 5 : Let $\varphi(X_1, ..., X_p)$ be an MSO formula and C be an effectively given set of labels.

(1) There exists a deterministic FA $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ over F_C that recognizes the language $L_{\varphi(X_1,...,X_p),C}$. This automaton can be constructed by an algorithm.

(2) If $C' \subseteq C$ is effectively given, then $\mathcal{A}_{\varphi(X_1,...,X_p),C'}$ is a subautomaton of $\mathcal{A}_{\varphi(X_1,...,X_p),C}$.

(3) If C is finite, then $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ is finite.

Proof : In [7, 10] and related articles, we construct, for each finite set C, a finite deterministic automaton $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ that accepts the recognizable

language $L_{\varphi(X_1,...,X_p),C} \subseteq T(F_C^{(p)})$. The construction is by induction on the structure of $\varphi(X_1, ..., X_p)$. We first construct automata for the atomic formulas, $X \subseteq Y$ and $edg(X, Y)$ and also for some useful MSO properties such as $X = \emptyset$, $Sgl(X)$, $Partition(X_1, \ldots, X_p)$, $Conn(X)$ expressing that the induced subgraph with vertex set X is connected, $DirCycle$ expressing that the graph has an directed cycle. These constructions are based on our understanding of the considered properties.

Operations on FA "implement" the logical connectives $\vee, \wedge, \neg, \exists$ and variable substitutions. The operation reflecting existential quantification consists in taking an image automaton, followed by a determinization. We have :

$$\mathcal{A}_{\exists X_p.\varphi(X_1,...,X_p),C} = \det(h(\mathcal{A}_{\varphi(X_1,...,X_p),C}))$$

where $h : F_C^{(p)} \to F_C^{(p-1)}$ deletes the last Boolean of w in each nullary symbol $(\mathbf{a}, w)$.

The transformation reflecting substitutions of variables consists in taking an inverse image: if $\psi(X_1, ..., X_p)$ is defined as $\varphi(X_{i_1}, ..., X_{i_m})$ from $\varphi(Y_1, ..., Y_m)$, then the automaton $\mathcal{A}_{\psi(X_1,...,X_p),C}$ is an inverse image of $\mathcal{A}_{\varphi(Y_1,...,Y_m),C}$ ([7], Lemma 13 and Section 4.2.1). For an example, if $\psi(X_1, X_2, X_3)$ is defined as $edg(X_3, X_1)$, we obtain $\mathcal{A}_{\psi(X_1,X_2,X_3),C} = h^{-1}(\mathcal{A}_{edg(X_1,X_2),C})$ where $h((\mathbf{a}, w_1w_2w_3))$ $= (\mathbf{a}, w_3, w_1)$ for all $w_1, w_2, w_3 \in \{0, 1\}$. If $\varphi(X_1, ..., X_p)$ is the conjunction of two formulas for which we know automata, then the automaton for $\varphi(X_1, ..., X_p)$ is the product of these two automata, provided they are built for formulas having the same list of free variables. Variable substitution is useful to insure this technical point[8].

As explained in [7], Section 7.3.1, these constructions work for infinite sets of labels. The three assertions of Theorem 5 can be proved by induction on the structure of φ. To be more precise[9], we define, for each set X and integer $i \geq 0$, the set $\mathcal{L}_i(X)$ as follows:

$$\mathcal{L}_0(X) := X,$$
$$\mathcal{L}_{i+1}(X) := \mathcal{L}_i(X) \cup \mathcal{P}_f(\mathcal{L}_i(X)) \cup (\mathcal{L}_i(X) \times \mathcal{L}_i(X)).$$

It is clear that $X \subseteq Y$ implies $\mathcal{L}_i(X) \subseteq \mathcal{L}_i(Y)$ and that $\mathcal{L}_i(X)$ is finite if X is finite. The set $\mathcal{L}_i(X)$ is effectively given if X is.

For every MSO formula $\varphi(X_1, ..., X_p)$, one can define a finite set B and an integer i such that, for each effectively given set C of labels, one can construct a deterministic FA $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ over $F_C^{(p)}$ that recognizes the language $L_{\varphi(X_1,...,X_p),C}$ and whose set of states $Q(C)$ satisfies the following properties:

(i) $Q(C) \subseteq \mathcal{L}_i(B \cup C)$,
(ii) $Q(C') = Q(C) \cap \mathcal{L}_i(B \cup C')$ if $C' \subseteq C$,

[8] We use this observation in Section 4.2.

[9] This definition is used in [9] to prove that the *(Strong) Recognizability Theorem,* Theorem 5.68(1) of [10], can be proved via the construction of automata.

(iii) if $\ell : C \to C'$ is a bijection, then $\mathcal{A}_{\varphi(X_1,...,X_p),C'}$ is the FA obtained from $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ by replacing in its states and transitions a by $\ell(a)$ for each $a \in C$ (informally, ℓ yields an isomorphism of automata : $\mathcal{A}_{\varphi(X_1,...,X_p),C} \to \mathcal{A}_{\varphi(X_1,...,X_p),C'}$).

For an example, the automaton $\mathcal{A}_{edg(X,Y),C}$ constructed in [7,10] has states in $\{Error, Ok\} \cup (P_{\leq 1}(C) \times P_{\leq 1}(C))$ that is a subset of $\mathcal{L}_2(\{Error, Ok\} \cup C)$. (As they are defined in [7,10], states are in $\mathcal{L}_1(B \cup C)$ for some finite set B.)

It follows from Properties (i)-(iii) that we need only construct a single FA $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ over $F_C^{(p)}$ where C is an infinite effectively given set of labels. We will take $C = \mathbb{N}_+$. Running $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ on terms in $T(F_{[k]}^{(p)})$, where $[k] := \{1, ..., k\}$, is the same as running $\mathcal{A}_{\varphi(X_1,...,X_p),[k]}$. An implementation of $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ yields a single program that works for all terms in $T(F_{[k]}^{(p)})$, $k \geq 1$, hence for graphs of all clique-widths.

The membership in $L_{\varphi(X_1,...,X_p),C}$ of a term $t \in T(F_{[k]}^{(p)})$ can be decided in (FPT) time $f(k).|t|$ for a function f depending only on $\varphi(X_1, ..., X_p)$. This function bounds the time for computing transitions. This gives a linear time recognition algorithm for graphs of clique-width at most some fixed k given by terms in $T(F_{[k]})$, but finding such terms takes (presently) cubic time. See [10] for details and references.

Properties of induced subgraphs

We apply to the *relativization* of a property P to a set X the notion of inverse image of an FA. If P is a graph property, we denote by $P[X]$ the property of (G, X) where X is a set of vertices of G such that the induced subgraph $G[X]$ satisfies P.

Let $t \in T(F_C)$ define a graph $G := val(t)$ and X be a set of vertices of G. (We recall that X is a set of occurrences of the nullary symbols $\mathbf{a}$ for $a \in C$.) Let $t' \in T(F_C)$ be obtained from t by replacing each $\mathbf{a}$ by the nullary symbol $\varnothing$ (that denotes the empty graph) at its occurrences not in X. Then $val(t')$ is $G[X]$, the induced subgraph of G with vertex set X. (If t is irredundant, then so is t').

Then the subset $L_{P[X],C}$ of $T(F_C^{(1)})$ is $h^{-1}(L_{P,C})$ where $h : T(F_C^{(1)}) \to T(F_C)$ replaces in a term of $T(F_C^{(1)})$ each nullary symbol $(\mathbf{a}, 0)$ by $\varnothing$ and $(\mathbf{a}, 1)$ by $\mathbf{a}$. Hence, if P is defined by an MSO sentence φ, then $P[X]$ is defined by an MSO formula $\varphi[X]$ with free variable X (we need not write this formula) and $\mathcal{A}_{\varphi[X],C}$ is the inverse image by h of $\mathcal{A}_{\varphi,C}$. See Definition 4(e) and Lemma 15 of [7], or [10].

Irredundant terms

The set of irredundant terms in $T(F_C)$ or in $T(F_C^{(p)})$, denoted by L_{Irr}, is recognized by an FA $\mathcal{I}$ ([7]), whose states are $Error$ and the finite subsets of C^2. For each term t, $q_{\mathcal{I}}(t) = Error$ if t is not irredundant and is $\{(\pi_{val(t)}(x), \pi_{val(t)}(y)) \mid x \rightarrow_{val(t)} y\}$ if t is irredundant. Its transitions are easy to define.

We will construct automata $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ intended to work correctly for irredundant terms. This means that $L(\mathcal{A}_{\varphi(X_1,...,X_p),C}) \cap L_{Irr} = L_{\varphi(X_1,...,X_p),C} \cap L_{Irr}$. If an input term is not irredundant, the answer of $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ may be incorrect. By taking the product of $\mathcal{A}_{\varphi(X_1,...,X_p),C}$ and $\mathcal{I}$, we can get an automaton that recognizes exactly $L_{\varphi(X_1,...,X_p),C} \cap L_{Irr}$. However, we will assume that the verification of irredundancy is done by the *parsing algorithm* that builds the term denoting the input graph to be processed by the automaton.

3 Incidence graphs

When the domain of the relational structure representing a graph is its vertex set, and since quantifications over binary relations are not allowed, MSO formulas cannot use quantified variables denoting sets of edges. In the incidence graph $Inc(G)$ of a graph G, each edge of G is made into a vertex, hence an MSO formula over $Inc(G)$ can be seen as an MSO formula using quantifications on sets of edges and it is called an MSO_2 formula intended to express a property of G. The formula : "there exists a set of edges that induces a directed cycle and goes through all vertices", expresses that the considered graph has a directed Hamiltonian cycle. This property is not expressible in MSO logic over the relational $\langle V_G, edg_G \rangle$ [10].

Assertion (b) of Theorem 1 does not extend to inputs $Inc(G)$ for G of clique-width at most k because the clique-width of such graphs $Inc(G)$ is unbounded. It can be used for graphs G of tree-width at most k. Proposition 3 gives $cwd(Inc(G)) = 2^{O(twd(G))}$ (because $twd(Inc(G)) \leq twd(G) + 1$) however $cwd(Inc(G)) = O(twd(G))$ as we will see.

Definition 6 *: Incidence graphs.*

Let G be a directed graph that can have parallel edges and loops. We define it as a triple $\langle V_G, E_G, inc_G \rangle$ such that V_G is its set of vertices, E_G is its set of edges (of course $V_G \cap E_G = \emptyset$) and $inc_G \subseteq (V_G \times E_G) \cup (E_G \times V_G)$ is such that, for $u, v \in V_G$ and $e \in E_G$, we have $(u, e) \in inc_G$ and $(e, v) \in inc_G$ if e is an edge $u \rightarrow v$. We define $Inc(G)$ as the simple, directed and bipartite graph identified to the relational structure $\langle V_G \cup E_G, inc_G, isv_G \rangle$ where isv is unary and $isv_G(u)$ holds if and only if $u \in V_G$. This relation is useful to distinguish the vertices from the edges in an arbitrary relational structure $\langle X, inc, isv \rangle$ isomorphic to $\langle V_G \cup E_G, inc_G, isv_G \rangle$. It follows that G can be reconstructed up to isomorphism from any structure $\langle X, inc, isv \rangle$ isomorphic to $\langle V_G \cup E_G, inc_G, isv_G \rangle$.

If G is undirected, then we define $inc_G \subseteq E_G \times V_G$ where $(e, u) \in inc_G$ if u is an end of e. In this case, the predicate isv_G is not necessary because the

vertices of G are the elements x of the domain such that $inc(x, y)$ holds for no y. We will only discuss directed graphs in the sequel, leaving to the reader the easy task of simplifying our constructions so as to handle undirected graphs.

We will use some adaptations of MSO formulas and clique-width operations. We will consider a relational structure $S = \langle D_S, inc_S, isv_S \rangle$ as a simple graph, also denoted by S, whose vertex set is D_S, adjacency relation is inc_S and that is equipped with a distinguished set of vertices isv_S. In such a structure isomorphic to an incidence graph $\langle V_G \cup E_G, inc_G, isv_G \rangle$, we will call *v-vertices* the vertices of S that satisfy isv and represent the vertices of G and *e-vertices* the others, that represent its edges. It is FO expressible whether a relational structure $S = \langle D_S, inc_S, isv_S \rangle$ is $Inc(G)$ for some directed graph G. If this is true, then G is unique and its vertex set is isv_G.

For expressing properties of such structures S, we will write MSO formulas with two types of set variables : $X, Y, ...$ to denote sets of v-vertices and U, V, W to denote sets of *e-vertices*[10]. The atomic formulas will be of the forms $X \subseteq Y, U \subseteq V, inc(X, U)$ and $inc(U, X)$; a formula $X \subseteq U$ or $inc(X, Y)$ is not allowed, and the predicate isv will not occur in formulas, as it is replaced by the typing of variables, forcing them to denote either sets of vertices or sets of edges.

For constructing incidence graphs with clique-width operations, we will use pairs (C, D) of disjoint, effectively given sets of labels. Those in C will define the v-vertices, and those in D the e-vertices. The operations $relab$ and $\overrightarrow{add}$ will be those from $F_{C \cup D}$ such that:

no label in C can be changed to a label in D, and vice-versa,

no edge-addition $\overrightarrow{add}_{a,b}$ can be used with $a, b \in C$ or $a, b \in D$.

We obtain an effectively given signature $F_{C,D}$. Every term $t \in T(F_{C,D})$ defines a simple bipartite graph $val(t) = H = \langle V_H, inc_H, isv_H \rangle$ where $isv_H(x)$ holds if and only if x is defined by $\mathbf{a}$ for some $a \in C$. We have $inc_H(x, y)$ if and only if $x \rightarrow_{val(t)} y$. We say that t is *correct* if $H = Inc(G)$ for some graph G, whose vertex set is then necessarily isv_H. This is the case if and only if each vertex having a label in D has indegree and outdegree 1.

Proposition 7 : If a directed graph G has tree-width k, then $Inc(G)$ is defined by a term in $T(F_{C,D})$ such that $|C| = 2$ and $|D| = 2k+3$. If G is undirected, then $Inc(G)$ is defined by a term in $T(F_{C,D})$ such that $|C| = 2$ and $|D| = k+2$. Terms witnessing these bounds can be constructed in linear time from tree-decompositions of G of width k. Conversely, $twd(G) = O(cwd(Inc(G)))$.

Proof : See [2,5] for the first three assertions. We have actually $cwd(Inc(G)) \leq 2k+4$ for G directed if we allow relabellings $a \rightarrow d$ such that $a \in C$ and $d \in D$ in terms defining $Inc(G)$. Similarly, $cwd(Inc(G)) \leq k + 3$ if G is undirected.

[10]These formulas could be alternatively defined as MSO formulas where each variable X (resp. U) comes with the condition that the elements x of X (resp. of U) satisfy $isv(x)$ (resp. $\neg isv(x)$).

The last assertion holds because $twd(G) \leq twd(Inc(G))$ and, by [16] (also in [10], Proposition 2.115), since $Inc(G)$ has no subgraph isomorphic to $K_{3,3}$, we have $twd(Inc(G)) \leq 6.cwd(Inc(G)) - 1$.□

By the last assertion, the reduction of (b) to (a) in Theorem 1 does not work for the verification of MSO_2 properties of graphs of bounded clique-width. This is not a surprise because there are MSO_2 properties that are not FPT for clique-width unless FPT = W[1] ([14]) which is unlikely, similarly to N = NP (see [12, 13] for the classes FPT and W[1]).

In order to represent in terms sets of vertices $X_1, ..., X_p$ and sets of edges $U_1, ..., U_m$, we replace in $F_{C,D}$ each nullary symbol $\mathbf{a}$, for $a \in C$, by the nullary symbols $(\mathbf{a}, w)$ for $w \in \{0,1\}^p$ and each nullary symbol $\mathbf{d}$, for $d \in D$, by the symbols $(\mathbf{d}, w)$ for $w \in \{0,1\}^m$. We obtain a signature $F_{C,D}^{(p,m)}$ and, for each MSO formula $\varphi(X_1, ..., X_p, U_1, ..., U_m)$, a representing language $L_{\varphi(X_1,...,X_p,U_1,...,U_m),C,D} \subseteq T(F_{C,D}^{(p,m)})$ consisting of the correct terms t that encode an assignment to $X_1, ..., X_p$, $U_1, ..., U_m$ for which φ holds in $val(t)$. We will construct FA that recognize these languages.

Denotation of subgraphs

Let G be a graph, $X \subseteq V_G$ and $U \subseteq E_G$. We let $Inc(G)[X, U] := \langle X \cup U, inc_G \cap (X \cup U)^2, isv_G \cap (X \cup U)\rangle$. This structure is an incidence graph $Inc(H)$ if and only if the ends of all "edges" $u \in U$ are in X. If this is the case, then $V_H = X$, $E_H = U$ and H is a subgraph of G. We call $Subgraph(X, U)$ this property of (G, X, U). Assume now that $Inc(G) = val(t)$ where $t \in T(F_{C,D})$, and X, U as above are sets of occurrences in t of nullary symbols respectively in $\mathbf{C}$ and in $\mathbf{D}$. Let $t[X, U]$ be obtained from t by replacing by $\varnothing$ (denoting the empty graph) the nullary symbols at their occurrences not in $X \cup U$. It is clear that $val(t[X, U]) = Inc(G)[X, U]$.

Then $t[X, U]$ is a correct term if and only if $Inc(G)[X, U]$ is an incidence graph. The set of correct terms $t[X, U] \in T(F_{C,D}^{(1,1)})$ is $h^{-1}(L)$ where L is the set of correct terms in $T(F_C)$ and h maps $(\mathbf{a}, 1)$ to $\mathbf{a}$ and $(\mathbf{a}, 0)$ to $\varnothing$ for $a \in C \cup D$. We will define L by an FA $\mathcal{A}_{CT}$. The property $Subgraph(X, U)$ will thus be defined by its inverse image, $h^{-1}(\mathcal{A}_{CT})$.

4 Automata

In this section, we construct FA to check MSO_2 properties of directed graphs. It is easy to modify them in order to check similar properties of undirected graphs. These automata will be deterministic and designed so as to work correctly on irredundant terms in $T(F_{C,D})$ for pairs (C, D) of disjoint effectively given sets of labels.

They will be *linear FPT-FA*, meaning that their computation times are linear in the size of input terms over fixed finite subsignatures of $F_{C,D}$.

Their constructions are the same for $C \cup D$ either finite or infinite, as explained after Theorem 5. We will construct FA for unspecified pairs C, D, either $C = \mathbb{N}_+$ and $D = \{-n \mid n \in \mathbb{N}_+\}$ (for being concrete) or finite subsets of them. The complexities of our automata will appear from their numbers of states when C and D are finite. An FA constructed from a formula φ will be denoted by $\mathcal{A}_\varphi$ without reference to C, D.

4.1 Correct terms

Every term $t \in T(F_{C,D})$ defines a simple bipartite graph $val(t)$, and is correct if and only if $val(t)$ is an incidence graph. A C-vertex of $val(t)$ is a vertex having a label in C, a D-vertex is one having a label in D. These definitions apply even if t is not correct.

We describe an FA $\mathcal{A}_{CT}$ that checks the correctness of terms in $T(F_{C,D})$, assumed to be irredundant. Its states are the sink *Error* and the 6-tuples $(\gamma_1, \gamma_2, \delta_{00}, \delta_{01}, \delta_{10}, \delta_{11}) \in \mathcal{P}_f(C)^2 \times \mathcal{P}_f(D)^4$ such that $\gamma_1 \cap \gamma_2 = \emptyset$ and $\delta_{00}, \delta_{01}, \delta_{10}, \delta_{11}$ are pairwise disjoint. We will denote $(\delta_{00}, \delta_{01}, \delta_{10}, \delta_{11})$ by $\overrightarrow{\delta}$.

At the root of a term $t \in T(F_{C,D})$, $\mathcal{A}_{CT}$ reaches the state $(\gamma_1, \gamma_2, \overrightarrow{\delta})$ if and only if, we have in $val(t)$:

(i) γ_1 is the set of labels $a \in C$ that label a single C-vertex.

(ii) γ_2 is the set of labels $a \in C$ that label at least two C-vertices,

(iii) $\overrightarrow{\delta} = (\delta_{00}, \delta_{01}, \delta_{10}, \delta_{11})$ where for $i, j \in \{0, 1\}$, δ_{ij} is the set of labels of D-vertices of indegree i and outdegree j,

(iv) the sets $\delta_{00}, \delta_{01}, \delta_{10}, \delta_{11}$ defined in (iii) are pairwise disjoint,

(v) no D-vertex has indegree or outdegree 2 or more.

It reaches the state *Error* if (iv) or (v) does not hold.

The accepting states are the tuples $(\gamma_1, \gamma_2, \overrightarrow{\delta})$ such that $\delta_{00} = \delta_{01} = \delta_{10} = \emptyset$.

For any subterm t of a correct term, Condition (iv) is necessary by Lemma 2 and Condition (v) also, because the D-vertices in an incidence graph represent edges. Transitions are listed in Table 1. In order to simplify the table, we use the following notations and conventions. First, we do not list the transitions with *Error* on the left side as they always yield *Error* (cf. Definition 4(c)). Furthermore,

a, b denote elements of C and d, e denote elements of D,

$d \in \overrightarrow{\delta}$ means $d \in \delta_{00} \cup \delta_{01} \cup \delta_{10} \cup \delta_{11}$,

$Disj(\overrightarrow{\delta})$ means that $\delta_{00}, \delta_{01}, \delta_{10}, \delta_{11}$ are pairwise disjoint,

$\overrightarrow{\delta} \cup \overrightarrow{\delta'} := (\delta_{00} \cup \delta'_{00}, \delta_{01} \cup \delta'_{01}, \delta_{10} \cup \delta'_{10}, \delta_{11} \cup \delta'_{11})$,

$\gamma[a \to b]$ is the set γ where a is replaced by b, and similarly for $\delta[d \to e]$,

$\overrightarrow{\delta}[d \to e] := (\delta_{00}[d \to e], \delta_{01}[d \to e], \delta_{10}[d \to e], \delta_{11}[d \to e])$,

$\overrightarrow{\emptyset}$ denotes a sequence of empty sets of appropriate length.

Transitions	Conditions
$\varnothing \to \overrightarrow{\emptyset}$	
$\mathbf{a} \to (\{a\}, \overrightarrow{\emptyset})$	
$\mathbf{d} \to (\emptyset, \emptyset, \{d\}, \overrightarrow{\emptyset})$	
$relab_{a\to b}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta})$	$a \notin \gamma_1 \cup \gamma_2$
$relab_{a\to b}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\overline{\gamma}_1, \overline{\gamma}_2, \overrightarrow{\delta})$	$\{a, b\} \subseteq \gamma_1 \cup \gamma_2$, $\overline{\gamma}_1 = \gamma_1 - \{a, b\}$ and $\overline{\gamma}_2 = \gamma_2 \cup \{b\} - \{a\}$
$relab_{a\to b}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\overline{\gamma}_1, \overline{\gamma}_2, \overrightarrow{\delta})$	$a \in \gamma_1 \cup \gamma_2$, $b \notin \gamma_1 \cup \gamma_2$, $\overline{\gamma}_1 = \gamma_1[a \to b]$ and $\overline{\gamma}_2 = \gamma_2[a \to b]$
$relab_{d\to e}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta}[d \to e])$	$Disj(\overrightarrow{\delta}[d \to e])$
$relab_{d\to e}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Error$	$\neg Disj(\overrightarrow{\delta}[d \to e])$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta})$	$a \notin \gamma_1 \cup \gamma_2$ or $d \notin \overrightarrow{\delta}$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Error$	($a \in \gamma_2$ and $d \in \overrightarrow{\delta}$) or $a \in \gamma_1$ and $d \in \delta_{10} \cup \delta_{11}$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta'})$	$a \in \gamma_1$ and $d \in \delta_{00} \cup \delta_{01}$, $\delta'_{00} = \delta_{00} - \{d\}, \delta'_{01} = \delta_{01} - \{d\}$, $\delta'_{10} =$ **if** $d \in \delta_{00}$ **then** $\delta_{10} \cup \{d\}$ **else** δ_{10}, $\delta'_{11} =$ **if** $d \in \delta_{01}$ **then** $\delta_{11} \cup \{d\}$ **else** δ_{11}.
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), (\gamma'_1, \gamma'_2, \overrightarrow{\delta'})] \to (\overline{\gamma}_1, \overline{\gamma}_2, \overrightarrow{\delta} \cup \overrightarrow{\delta'})$	$Disj(\overrightarrow{\delta} \cup \overrightarrow{\delta'})$, $\overline{\gamma}_2 = \gamma_2 \cup \gamma'_2 \cup (\gamma_1 \cap \gamma'_1)$, $\overline{\gamma}_1 = (\gamma_1 - (\gamma'_1 \cup \gamma'_2)) \cup (\gamma'_1 - (\gamma_1 \cup \gamma_2))$.
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), (\gamma'_1, \gamma'_2, \overrightarrow{\delta'})] \to Error$	$\neg Disj(\overrightarrow{\delta} \cup \overrightarrow{\delta'})$

Table 1: Some transitions of $\mathcal{A}_{CT}$.

Remarks: (1) We do not list the transitions relative to $\overrightarrow{add}_{d,a}$ because they are similar to those of $\overrightarrow{add}_{a,d}$.

(2) For the transitions relative to $\overrightarrow{add}_{a,d}$, there are three cases. It is clear that the conditions on a and d are mutually exclusive and cover all possibilities.

(3) The transition $\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Error$ is correct because the input term is assumed irredundant. Without irredundancy, it may happen that $a \in \gamma_1$ and $d \in \delta_{11} \cup \delta_{10}$ but $\overrightarrow{add}_{a,d}$ has no effect because there are already edges from the a-vertex to the d-vertex (or to several d-vertices), so that the d-vertex (or vertices) still have indegree 1. In that case, the transition should yield $(\gamma_1, \gamma_2, \overrightarrow{\delta})$.

(4) If we replace the transitions $\mathbf{a} \to (\{a\}, \overrightarrow{\emptyset})$ and $\mathbf{d} \to (\emptyset, \emptyset, \{d\}, \overrightarrow{\emptyset})$ by $(\mathbf{a}, w) \to (\{a\}, \overrightarrow{\emptyset})$ and $(\mathbf{d}, w') \to (\emptyset, \emptyset, \{d\}, \overrightarrow{\emptyset})$ respectively for $w \in \{0,1\}^p$ and $w' \in \{0,1\}^m$, we obtain an automaton that checks the correctness of a term in $T(F^{(p,m)}_{C,D})$.

(5) Some states $(\gamma_1, \gamma_2, \overrightarrow{\delta})$ are not accessible: for example, those such that $\gamma_1 \cup \gamma_2 = \emptyset$ and $\delta_{01} \cup \delta_{10} \cup \delta_{11} \neq \emptyset$.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is $3^k.5^\ell + 1$ and the size of a state is $O(k + \ell)$. Each transition is computable in time $O(k + \ell)$ (our time complexity evaluations are based on straightforward data structures). We obtain a linear FPT-FA.

As noted previously, $h^{-1}(\mathcal{A}_{CT}) = \mathcal{A}_{Subgraph(X,U)}$ where h maps $(\mathbf{a}, 1)$ to $\mathbf{a}$ and $(\mathbf{a}, 0)$ to $\varnothing$ for each $a \in C \cup D$.

4.2 Adjacency in incidence graphs

In the case of MSO graph properties reviewed in Section 2 the atomic formula $edg(X, Y)$ is checked by an FA over $F_C^{(2)}$ that has $k^2 + k + 3$ states if C is finite of cardinality k. This construction is easily applicable to the atomic formula $inc(X, U)$ (resp. $inc(U, X)$) stating that X consists of one vertex x and U of one edge u whose tail (resp. head) is x. We will describe $\mathcal{A}_{inc(X,U)}$ at the end of the section for purpose of comparison.

In MSO formulas expressing properties of $Inc(G)$, the property $edg(X, Y)$ is no longer atomic; it is expressed by $\exists U.(inc(X, U) \wedge inc(U, Y))$. We can apply the general construction of [7, 10] to this formula, but, in view of practical constructions, it is useful to define directly an automaton $\mathcal{A}_{edg(X,Y)}$ over $F_{C,D}^{(2,0)}$.

The FA $\mathcal{A}_{edg(X,Y)}$

We now construct an FA $\mathcal{A}_{edg(X,Y)}$ intended to run on correct and irredundant terms in $T(F_{C,D}^{(2,0)})$. As for irredundancy, we will assume that correctness is guaranteed by the parsing algorithm.

The states of $\mathcal{A}_{edg(X,Y)}$ are Ok, $Error$ and the tuples $(\gamma_1, \gamma_2, \overrightarrow{\delta}) \in \mathcal{P}_{\leq 1}(C)^2 \times \mathcal{P}_f(D)^3$ such that the components of $\overrightarrow{\delta} = (\delta, \delta_1, \delta_2)$ verify the condition $\delta_1 \cup \delta_2 \subseteq \delta$.

Let $t \in T(F_{C,D}^{(2,0)})$ be a correct and irredundant term. It defines an incidence graph $val(t) = Inc(G)$ and two sets of vertices X, Y of G. Every subterm t' of t is irredundant and defines a bipartite graph $val(t')$ that we consider as a subgraph of $val(t)$ (by our *Convention about subterms*, cf. Section 2). It may not be an incidence graph, because some D-vertices may be of indegree or outdegree 0. Let $X' = X \cap V_{val(t')}, Y' = Y \cap V_{val(t')}$.Their sets of labels are $\pi_{val(t')}(X')$ and $\pi_{val(t')}(Y')$, both subsets of C. To simplify notation we will denote $\pi_{val(t')}$ by π and $\rightarrow_{val(t')}$ (the edge relation) by $\rightarrow$. At the root of t', $\mathcal{A}_{edg(X,Y)}$ reaches the following state:

> $Error$ if and only if X' or Y' has cardinality 2 or more,
>
> Ok if and only if $X' = \{x\}$, $Y' = \{y\}$ for some x, y such that $x \rightarrow u \rightarrow y$ for some D-vertex u (so that $x \rightarrow_G y$),

$(\gamma_1, \gamma_2, \overrightarrow{\delta})$ otherwise, and we have :

$\gamma_1 = \pi(X')$ and $|X'| \leq 1$,

$\gamma_2 = \pi(Y')$ and $|Y'| \leq 1$,

δ is the set of labels of D-vertices,

$\delta_1 = \pi(Out(X')) \subseteq \delta$,

$\delta_2 = \pi(In(Y')) \subseteq \delta$,

where $Out(X')$ (resp. $In(Y')$) is the set of D-vertices u such that $X' \to u$ (resp. $u \to Y'$).

The accepting state is Ok.

Transitions	Conditions
$\varnothing \to \overrightarrow{\emptyset}$	
$(\mathbf{a}, 00) \to \overrightarrow{\emptyset}$	
$(\mathbf{a}, 10) \to (\{a\}, \overrightarrow{\emptyset})$	
$(\mathbf{a}, 01) \to (\emptyset, \{a\}, \overrightarrow{\emptyset})$	
$(\mathbf{a}, 11) \to (\{a\}, \{a\}, \overrightarrow{\emptyset})$	
$\mathbf{d} \to (\emptyset, \emptyset, \{d\}, \overrightarrow{\emptyset})$	
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta})$	$a \notin \gamma_1$ or $d \notin \delta$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Ok$	$a \in \gamma_1$ and $d \in \delta_2$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \delta, \delta'_1, \delta_2)$	$a \in \gamma_1$, $d \in \delta - \delta_2$, $\delta'_1 := \delta_1 \cup \{d\}$
$\overrightarrow{add}_{a,d}[Ok] \to Ok$	
$relab_{a\to b}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma'_1, \gamma'_2, \overrightarrow{\delta})$	$\gamma'_1 := \gamma_1[a \to b], \gamma'_2 := \gamma_2[a \to b]$
$relab_{a\to b}[Ok] \to Ok$	
$relab_{d\to e}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta'})$	$\overrightarrow{\delta'} := \overrightarrow{\delta}[d \to e]$
$\oplus[Ok, Ok] \to Error$	
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), Ok] \to Error$	$\gamma_1 \neq \emptyset$ or $\gamma_2 \neq \emptyset$
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), Ok] \to Ok$	$\gamma_1 = \gamma_2 = \emptyset$
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), (\gamma'_1, \gamma'_2, \overrightarrow{\delta'})] \to Error$	$\lvert\gamma_1\rvert + \lvert\gamma'_1\rvert \geq 2$ or $\lvert\gamma_2\rvert + \lvert\gamma'_2\rvert \geq 2$
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), (\gamma'_1, \gamma'_2, \overrightarrow{\delta'})] \to (\gamma_1 \cup \gamma'_1, \gamma_2 \cup \gamma'_2, \overrightarrow{\delta} \cup \overrightarrow{\delta})$	otherwise

Table 2 : Some transitions of $\mathcal{A}_{edg(X,Y)}$

Remarks : (1) The transitions $\oplus[(\emptyset, \emptyset, \overrightarrow{\delta}), Ok] \to Ok$ "loose" the value $\overrightarrow{\delta}$. This value is not needed after Ok is obtained. The only thing that remains to be checked is that the sets X, Y are singletons. It follows that we have transitions such as $\oplus[Ok, Ok] \to Error$ and $\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), Ok] \to Error$ if $\gamma_1 \neq \emptyset$ or $\gamma_2 \neq \emptyset$. The state Ok is not a sink.

(2) Let us comment on δ_2. A D-vertex u such that $u \to Y'$ corresponds either to a partially defined edge with head in Y' and whose tail is not yet found or to an edge from a vertex not in X' to a vertex in Y'. The set δ_2 is the set of their labels. The transition $\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Ok$ when $\gamma_1 = \{a\}$ and $d \in \delta_2$ is correct because, since t is assumed to be a correct term, a D-vertex u with label d cannot correspond to an edge from a vertex not in X' to a vertex in Y'. Hence, $u \to y \in Y'$ and the operation $\overrightarrow{add}_{a,d}$ creates an edge from the vertex of X' to y.

(3) Some states $(\gamma_1, \gamma_2, \overrightarrow{\delta})$ are not accessible, for example, those such that $\gamma_1 = \emptyset$ and $\delta_1 \neq \emptyset$.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is $(k+1)^2.5^\ell + 2$ and the size of a state is $O(\log(k) + \ell))$. Each transition is computable in time $O(\log(k) + \ell))$. We obtain a linear FPT-FA.

Comparison with the automaton constructed with Theorem 4.

The automaton $\mathcal{A}_{inc(X,U)}$ over $F^{(1,1)}_{C,D}$ obtained by a straightforward adaptation of the automaton $\mathcal{A}_{edg(X,Y)}$ over $F^{(2)}_{C}$ defined in [7, 10]. It has states Ok, *Error* and the pairs (γ, δ) in $\mathcal{P}_{\leq 1}(C) \times \mathcal{P}_{\leq 1}(D)$.

Let $t * (X, U) \in T(F^{(1,1)}_{C,D})$ be correct and irredundant. It defines an incidence graph $val(t) = Inc(G)$, a set X of C-vertices and a set U of D-vertices. Every subterm t' of t defines a bipartite graph $val(t')$. Let $X' = X \cap V_{val(t')}, U' = U \cap V_{val(t')}$. At the root of $t' * (X', U')$, $\mathcal{A}_{inc(X,U)}$ reaches the following state:

Error if and only if X' or U' has cardinality 2 or more,

Ok if and only if $X' = \{x\}$, $U' = \{u\}$ and $x \to_{val(t')} u$,

(γ, δ) otherwise where

$\gamma = \pi_{val(t')}(X')$ and $|X'| \leq 1$,

$\delta = \pi_{val(t')}(U')$ and $|U'| \leq 1$.

The accepting state is Ok and the transitions are easy to define. If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is $(k + 1).(\ell + 1) + 2$.

Similarly, we have $\mathcal{A}_{inc(U,Y)}$ with same set of states, except that we write $(\delta, \gamma) = (\pi_{val(t')}(U'), \pi_{val(t')}(Y')) \in \mathcal{P}_{\leq 1}(D) \times \mathcal{P}_{\leq 1}(C)$ instead of (γ, δ).

Let us now consider $edg(X, Y)$ defined by $\exists U.(inc_1(X, U, Y) \wedge inc_2(X, U, Y))$ where $inc_1(X, U, Y)$ is defined as $inc(X, U)$ with extra (useless) variable Y and $inc_2(X, U, Y))$ where $inc_2(X, U, Y)$ is defined as $inc(U, Y)$. These useless variables are added so that the two parts of the conjunction $inc_1(X, U, Y) \wedge inc_2(X, U, Y)$ have the same free variables.

The automaton $\mathcal{B} := \mathcal{A}_{inc_1(X,U,Y) \wedge inc_2(X,U,Y)}$ over $F^{(2,1)}_{C,D}$ is thus the product of $\mathcal{A}_{inc_1(X,U,Y)}$ and $\mathcal{A}_{inc_2(X,U,Y)}$. Furthermore, $\mathcal{A}_{inc_1(X,U,Y)}$ and $\mathcal{A}_{inc_2(X,U,Y)}$ have the sames states as, respectively, $\mathcal{A}_{inc(X,U)}$ and $\mathcal{A}_{inc(U,Y)}$. (See the remarks on variable substitutions after Theorem 5). The construction of Theorem 5 replaces $\mathcal{B}$ by a nondeterministic FA $\mathcal{C}$ over $F^{(2,0)}_{C,D}$ that characterizes

$\exists U.(inc_1(X,U,Y) \wedge inc_2(X,U,Y))$. The FA $\mathcal{B}$ has $((k+1).(\ell+1)+2)^2$ states if $k = |C|$ and $\ell = |D|$, which gives more than $2^{k^2.\ell^2}$ states for the deterministic automaton $\det(\mathcal{C})$ obtained from $\mathcal{C}$. However, some states of $\mathcal{B}$ are inaccessible, for instance, those of the form $((\gamma,\delta),(\delta',\gamma'))$ where $\delta \neq \delta'$. Furthermore, the states of $\det(\mathcal{C})$ are more complicated to write than those of $\mathcal{A}_{edg(X,Y)}$ are in the set $\{Ok, Error\} \cup (\mathcal{P}_{\leq 1}(C)^2 \times \mathcal{P}_f(D)^3)$ whereas those of $\det(\mathcal{C})$ are in $\mathcal{P}_f([\{Ok, Error\} \cup (\mathcal{P}_{\leq 1}(C) \times \mathcal{P}_{\leq 1}(D))]^2)$.

This observation motivates our interest for "direct constructions" of FA for properties related to adjacency, as these properties are defined with $edg(X,Y)$.

4.3 Links and domination

We consider the following four properties based on adjacency, ordered by increasing complexity, measured by the sizes of the automata we will construct (G is the graph whose incidence graph is $val(t)$ and t the given correct term) :

$Link^{\exists\exists}(X,Y)$ meaning that $X \rightarrow_G Y$, i.e., $x \rightarrow_G y$ for some $x \in X$ and $y \in Y$,

$Link^{\forall\exists}(X,Y)$ meaning that for all $x \in X$ there is $y \in Y$ such that $x \rightarrow_G y$, (Y *dominates* X for $\leftarrow_G$),

$Link^{\forall\forall}(X,Y)$ meaning that $x \rightarrow_G y$ for all $x \in X$ and $y \in Y$,

$Link^{\exists\forall}(X,Y)$ meaning that there is $x \in X$ such that $x \rightarrow_G y$ for all $y \in Y$ (x *dominates* Y for $\rightarrow_G$).

The property that X induces a complete directed graph (with loops on all vertices) is expressed by $Link^{\forall\forall}(X,X)$. That X is stable, i.e., that the induced graph $G[X]$ has no edge, is expressed by $\neg Link^{\exists\exists}(X,X)$.

The logical expressions of these properties by MSO formulas interpreted in $\langle V_G \cup E_G, inc_G\rangle$ have the following respective quantifier structures: $\exists\exists\exists, \forall\exists\exists$, $\forall\forall\exists$ and $\exists\forall\exists$ with 0, 1,1 and 2 quantifier alternations. We will construct FA or sketch their constructions. Their sets of states will reflect the differences of quantifier alternations. Notation is as for the definition of $\mathcal{A}_{edg(X,Y)}$. All FA will be intended to run on correct and irredundant terms in $T(F^{(2,0)}_{C,D})$.

The FA $\mathcal{A}_{Link^{\exists\exists}(X,Y)}$

It is similar to $\mathcal{A}_{edg(X,Y)}$ with some interesting differences. It is simpler to describe because it need not check that X and Y are singletons, however, it has more states. Its states are the accepting sink $Success$ and the tuples $(\gamma_1, \gamma_2, \overrightarrow{\delta}) \in \mathcal{P}_f(C)^2 \times \mathcal{P}_f(D)^3$ such that $\delta_1 \cup \delta_2 \subseteq \delta$. As above for $\mathcal{A}_{edg(X,Y)}$, if $t' * (X',Y')$ is a subterm of a correct irredundant term $t * (X,Y) \in T(F^{(2,0)}_{C,D})$ where $X' = X \cap V_{val(t')}$ and $Y' = Y \cap V_{val(t')}$, then $\mathcal{A}_{Link^{\exists\exists}(X,Y)}$ reaches the following state at the root of $t' * (X',Y')$:

Success if and only if $x \to u \to y$ for some $x \in X'$ and $y \in Y'$ and u (so that $X' \to_G Y'$); otherwise

$(\gamma_1, \gamma_2, \overrightarrow{\delta})$ where :

$\gamma_1 = \pi(X') \subseteq C$,

$\gamma_2 = \pi(Y') \subseteq C$,

δ is the set of labels of D-vertices,

$\delta_1 = \pi(Out(X')) \subseteq \delta$,

$\delta_2 = \pi(In(Y')) \subseteq \delta$.

Some transitions are listed in Table 3 (the others are easy to define or similar to those for $\mathcal{A}_{edg(X,Y)}$).

Transitions	Conditions
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \overrightarrow{\delta})$	$a \notin \gamma_1$ or $d \notin \delta$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to Success$	$a \in \gamma_1$ and $d \in \delta_2$
$\overrightarrow{add}_{a,d}[(\gamma_1, \gamma_2, \overrightarrow{\delta})] \to (\gamma_1, \gamma_2, \delta, \delta_1 \cup \{d\}, \delta_2)$	$a \in \gamma_1$, $d \in \delta - \delta_2$
$\oplus[(\gamma_1, \gamma_2, \overrightarrow{\delta}), (\gamma'_1, \gamma'_2, \overrightarrow{\delta'})]$ $\to (\gamma_1 \cup \gamma'_1, \gamma_2 \cup \gamma'_2, \overrightarrow{\delta} \cup \overrightarrow{\delta'})$	

Table 3: Some transitions of $\mathcal{A}_{Link^{\exists\exists}(X,Y)}$.

The accepting state is the sink *Success*. All transitions with *Success* on the left yield *Success*. (In $\mathcal{A}_{edg(X,Y)}$, we use a state *Ok*, that looks like *Success* but is not a sink.) See Remark (2) relative to $\mathcal{A}_{edg(X,Y)}$ to verify the validity of the transition to *Success* in this table. There is no *Error* state.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is $4^k.5^\ell + 1$. These states have size $O(k + \ell)$. Each transition is computed in time $O(k + \ell)$. We obtain a linear FPT-FA.

The FA $\mathcal{A}_{Link^{\forall\forall}(X,Y)}$

Notation is as for the previous automaton. The state at the root of t' will contain the relation :

$$\theta := \{(\pi(x), \pi(y)) \mid x \in X', y \in Y' \text{ and } \nexists u.(x \to u \to y)\},$$

and will be accepting if and only if this relation is empty, because θ indicates the existence of "missing" edges. However, additional information will be needed for the construction of transition rules.

A state of $\mathcal{A}_{Link^{\forall\forall}(X,Y)}$ is a 4-tuple of finite sets $(\delta, \Delta, \Lambda, \Theta)$ such that:

$\delta \subseteq D$,
$\Delta \subseteq C \times \mathcal{P}_f(\delta)$, $\Lambda \subseteq \mathcal{P}_f(\delta) \times C$ and
$\Theta \subseteq \Delta \times \Lambda \subseteq C \times \mathcal{P}_f(D) \times \mathcal{P}_f(D) \times C$.

The state at the root of $t' * (X', Y')$ is $(\delta, \Delta, \Lambda, \Theta)$ such that :

δ is the set of labels of D-vertices,
$\Delta = \{(\pi(x), \pi(Out(x)) \mid x \in X'\}$,
$\Lambda = \{(\pi(In(y)), \pi(y)) \mid y \in Y'\}$,
$\Theta = \{(\pi(x), \pi(Out(x)), \pi(In(y)), \pi(y)) \mid x \in X',$
$y \in Y'$ and $\nexists u.(x \to u \to y)\}$.

A tuple (a, η, η', b) in Θ encodes the following information about a "missing edge" from some $x \in X'$ to some $y \in Y'$: $a = \pi(x)$, $b = \pi(y)$, η contains the labels of the partially defined edges with tail x and similarly, η' contains the labels of those, partially defined, with head y. An edge from x to y in the graph G whose incidence graph is $val(t)$ can be created by $\overrightarrow{add}_{a,d}$ if $d \in \eta'$ or by $\overrightarrow{add}_{d,b}$ if $d \in \eta$ (assuming that labels a, d are not modified above t').

Remark : The states of $\mathcal{A}_{Link^{\forall\forall}(X,Y)}$ contain more information than those of $\mathcal{A}_{Link^{\exists\exists}(X,Y)}$ because $\pi(X') = \{a \in C \mid (a, \eta) \in \Delta \text{ for some } \eta\}$, $\pi(Out(X'))$ is the union of the sets η such that $(a, \eta) \in \Delta$ for some a and similarly for $\pi(Y')$ and $\pi(In(Y'))$. Hence, the state of $\mathcal{A}_{Link^{\exists\exists}(X,Y)}$ at some position u can be computed from that of $\mathcal{A}_{Link^{\forall\forall}(X,Y)}$ at u. One could formalize that by the existence of a homomorphism : $\mathcal{A}_{Link^{\forall\forall}(X,Y)} \to \mathcal{A}_{Link^{\exists\exists}(X,Y)}$. □

The accepting states are those such that $\Theta = \emptyset$ because, at the root of a correct term t, θ is the set of pairs (a, b) such that $(a, \eta, \eta', b) \in \Theta$ for some η, η'. Transitions are as follows. We begin with the easier cases.

$\varnothing \to \overrightarrow{\emptyset}$
$(\mathbf{a}, 11) \to (\emptyset, \{(a, \emptyset)\}, \{(\emptyset, a)\}, \{(a, \emptyset, \emptyset, a)\})$,
$(\mathbf{a}, 10) \to (\emptyset, \{(a, \emptyset)\}, \overrightarrow{\emptyset})$,
$(\mathbf{a}, 01) \to (\emptyset, \{(\emptyset, a)\}, \overrightarrow{\emptyset})$,
$(\mathbf{a}, 00) \to \overrightarrow{\emptyset}$,
$\mathbf{d} \to (\{d\}, \overrightarrow{\emptyset})$.

Transitions for relabellings are straightforward:
$relab_{x \to y}[(\delta, \Delta, \Lambda, \Theta)] \to (\delta', \Delta', \Lambda', \Theta')$ where $\delta' := \delta[x \to y]$ and similarly for the other components.

Transitions for disjoint union are as follows:

$\oplus[(\delta_1, \Delta_1, \Lambda_1, \Theta_1), (\delta_2, \Delta_2, \Lambda_2, \Theta_2)] \rightarrow$
$(\delta_1 \cup \delta_2, \Delta_1 \cup \Delta_2, \Lambda_1 \cup \Lambda_2, \Theta_1 \cup \Theta_2 \cup \Theta')$ where
$\Theta' := \{(a, \eta, \eta', b) \mid ((a, \eta) \in \Delta_1$ and $(\eta', b) \in \Lambda_2)$
or $((a, \eta) \in \Delta_2$ and $(\eta', b) \in \Lambda_1))\}$.

Transitions for edge addition are as follows:
$\overrightarrow{add}_{a,d}[(\delta, \Delta, \Lambda, \Theta)] \rightarrow q$ where we have:

$q := (\delta, \Delta, \Lambda, \Theta)$ if a does not occur in Δ or $d \notin \delta$,
$q := (\delta, \Delta', \Lambda, \Theta')$ otherwise, where :
Δ' is defined from Δ by replacing each pair (a, η) by $(a, \eta \cup \{d\})$;
Θ' is defined from in Θ as follows :
a tuple of the form (a, η, η', b), for any $b \in C$ is deleted
if $d \in \eta'$; otherwise it is replaced by $(a, \eta \cup \{d\}, \eta', b)$.

In the transitions for $\overrightarrow{add}_{a,d}$, we cannot have a and d occurring both in Δ because the input term t is assumed irredundant and correct. If d belongs to η', then it labels vertices of indegree 0 because t is correct and irredundant.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is at most $2^{\ell}.5^{k^2 4^{\ell}}$. However, there are less accessible states. Counting them precisely seems difficult and is actually not important for our use of FA. The size of a state is $O(k^2.4^{\ell})$. We obtain a linear FPT-FA.

The FA $\mathcal{A}_{Link^{\forall\exists}(X,Y)}$

The construction is similar to the previous one. The set of states is a bit smaller, although of similar type. Notation is as in the previous cases. The states are sets of 5-tuples of sets $(\gamma, \delta, \lambda, \Delta, \Theta) \in \mathcal{P}_f(C) \times \mathcal{P}_f(D)^2 \times \mathcal{P}_f(C \times \mathcal{P}_f(D))^2$ such that:

$\gamma \subseteq C, \delta \subseteq D, \lambda \subseteq \delta$ and $\Theta \subseteq \Delta \subseteq C \times \mathcal{P}_f(\delta)$.

The state at the root of $t' * (X', Y')$ is $(\gamma, \delta, \lambda, \Delta, \Theta)$ such that :

$\gamma = \pi(Y')$,
δ is the set of labels of D-vertices,
$\lambda = \pi(In(Y')) \subseteq \delta$,
$\Delta = \{(\pi(x), \pi(Out(X'))) \mid x \in X'\}$,
$\Theta = \{(\pi(x), \pi(Out(X'))) \mid x \in X'$ and $\nexists u.(x \rightarrow u \rightarrow Y')\}$.

The accepting states are those such that $\Theta = \emptyset$.

Transitions for disjoint union are as follows:

$$\oplus[(\gamma_1, \delta_1, \lambda_1, \Delta_1, \Theta_1), (\gamma_2, \delta_2, \lambda_2, \Delta_2, \Theta_2)] \rightarrow \\ (\gamma_1 \cup \gamma_2, \delta_1 \cup \delta_2, \lambda_1 \cup \lambda_2, \Delta_1 \cup \Delta_2, \Theta'_1 \cup \Theta'_2).$$

where:

$\Theta'_1 :=$ `if` $\gamma_2 = \emptyset$ `then` Θ_1 `else` Δ_1, and
$\Theta'_2 :=$ `if` $\gamma_1 = \emptyset$ `then` Θ_2 `else` Δ_2.

The most complicated transitions are for edge addition:
$\overrightarrow{add}_{a,d}[(\gamma, \delta, \lambda, \Delta, \Theta)] \rightarrow q$ where the following holds:

if a does not occur in Δ or $d \notin \delta$, then $q := (\gamma, \delta, \lambda, \Delta, \Theta)$,

otherwise $q := (\gamma, \delta, \lambda, \Delta', \Theta')$, where Δ', Θ' are defined as follows from Δ and Θ :

a pair (a, η) in Δ is replaced by $(a, \eta \cup \{d\})$,

a pair (a, η) in Θ is deleted if $d \in \lambda$ and replaced by $(a, \eta \cup \{d\})$ otherwise.

$\overrightarrow{add}_{d,b}[(\gamma, \delta, \lambda, \Delta, \Theta)] \rightarrow q$ where the following holds:

if $b \notin \gamma$ or $d \notin \delta$, then $q = (\gamma, \delta, \lambda, \Delta, \Theta)$,

otherwise $q := (\gamma, \delta, \lambda \cup \{d\}, \Delta, \Theta')$ where Θ' is defined from Θ by deleting each pair (a, η) such that if $d \in \eta$.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is bounded by $2^k 3^\ell . 3^{k.2^\ell}$. The size of a state is $O(k.2^\ell)$ and the computation time of a transition is $O(k.2^\ell)$. We obtain a linear FPT-FA.

The FA $\mathcal{A}_{Link^{\exists\forall}(X,Y)}$

This automaton is more complicated than the two previous ones, as it checks a formula with two quantifier alternations instead of one. Its states are triples of finite sets (δ, Λ, Ξ) such that:

$\delta \subseteq D$,
$\Lambda \subseteq \mathcal{P}_f(\delta) \times C$,
$\Xi \subseteq C \times \mathcal{P}_f(\delta) \times \mathcal{P}_f(\Lambda) \subseteq C \times \mathcal{P}_f(D) \times \mathcal{P}_f(\mathcal{P}_f(D) \times C)$.

The state at the root of $t' * (X', Y')$ is (δ, Λ, Ξ) such that :

δ is the set of labels of D-vertices,
$\Lambda = \{(\pi(In(y)), \pi(y)) \mid y \in Y'\}$,
Ξ is the set of triples
$(\pi(x), \pi(Out(x)), \{(\pi(In(y)), \pi(y)) \mid y \in Y', \nexists u.(x \to u \to y)\})$
for all $x \in X'$.

The accepting states are those such that Ξ contains a triple of the form $(a, \Lambda, \emptyset)$. We show some transitions.

$\oplus[(\delta_1, \Lambda_1, \Xi_1), (\delta_2, \Lambda_2, \Xi_2)] \to (\delta_1 \cup \delta_2, \Lambda_1 \cup \Lambda_2, \Xi_1 \cup \Xi_2 \cup \Xi')$
where Ξ' is the set of triples are of the form $(a, \eta, \Phi \cup \Lambda_2)$ for $(a, \eta, \Phi) \in \Xi_1$ and of the form $(a, \eta, \Phi \cup \Lambda_1)$ for $(a, \eta, \Phi) \in \Xi_2$.

The other transitions are easy to define, by the same methods as for the previous automata. If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, the number of states is bounded by $2^{\ell}.2^{k.2^{\ell}}.2^{k.2^{\ell}.2^{k.2^{\ell}}}$. The size of a state is $O(2^{k.2^{\ell}})$ and the computation time of a transition is $O(2^{k.2^{\ell}})$.

Table 4 compares the bounds on the sizes of the states for the linear FPT-FA we just constructed for incidence graphs to the bounds for those constructed in [7] for "ordinary" graphs. For terms obtained by Proposition 6 from a tree-decomposition of width p, we have $k = 2$ and $\ell \leq 2p + 3$.

Property	MSO	MSO_2
edg	$O(\log(k))$	$O(\log(k) + \ell)$
$Link^{\exists\exists}$	$O(k)$	$O(k + \ell)$
$Link^{\forall\exists}$	$O(k)$	$O(k.2^{\ell})$
$Link^{\forall\forall}$	$O(k^2)$	$O(k.2^{\ell})$
$Link^{\exists\forall}$	$O(2^k)$	$O(2^{k.2^{\ell}})$

Table 4: Comparison between MSO and MSO_2-automata.

5 Automata for other properties

5.1 Inc-invariant properties

Definition 8 : *Invariance for taking the incidence graph.*
A graph property P is *Inc-invariant* if $P(G) \Longleftrightarrow P(Inc(G))$ for every graph G.

So are, for instance, for a directed graph G, connectedness and strong connectedness, the properties that all its vertices are of outdegree at most p, that G has a directed cycle, or an undirected cycle (a cycle in which edge directions do not matter), or that G has a path from vertex x to vertex y.

For such a property P, an FA over $F_{C,D}$ intended to check it on incidence graphs defined by correct terms can be constructed from the FA $\mathcal{A}_{P,C}$ over F_C that checks P on "ordinary" graphs G, without needing significant modification: essentially, we replace C by $C \cup D$. Although the FA for the relation $edg(X, Y)$ is more complicated in the case of incidence graphs than for ordinary graphs, this increasing complexity does not extend to all properties expressed by means of $edg(X, Y)$.

5.2 An automaton for Directed Hamiltonian cycle

The property $DirHam$ that a graph G has a directed Hamiltonian cycle is MSO_2 expressible but not MSO expressible. It is expressed in $Inc(G)$ by "there exists a set of edges of G that forms a directed cycle going through all vertices". Without using the corresponding logical expression, we will construct an FA for this property.

We let P mean be that the considered graph is a directed cycle or is empty, and L be the set of correct irredundant terms t in $T(F_{C,D})$ such that $P(val(t))$ holds. We first construct an FA $\mathcal{B}$ over $F_{C,D}$ that recognizes L among correct and irredundant terms, that is, such that $L(\mathcal{B}) \cap L_{Irr} \cap L_{CT} = L$.

Let $t \in L$ and t' be a subterm of t. Then $val(t')$ satisfies one of the following properties:

(a) it is a single directed cycle (and then $val(t') = val(t)$),

(b) it consists of isolated vertices $x_1, ..., x_p, (p \geq 0)$ and of pairwise disjoint paths from y_1 to z_1, y_2 to z_2, ... , y_m to z_m $(m \geq 0)$, such that the labels of $x_1, ..., x_p$, y_1, ..., y_m, z_1,... , z_m are all different.

Each of them implies :

(c) no C-vertex has indegree or outdegree 2 or more.

Since t is assumed correct, every D-vertex has indegree and outdegree at most one. A transition that shows a violation of (c) will yield $Error$.

We define $\mathcal{B}$ with the following states : Ok, $Error$ and the 3-tuples (α, β, Ψ) in $\mathcal{P}_f(C \cup D) \times \mathcal{P}_f(C) \times \mathcal{P}_f((C \cup D)^2)$. At the root of a term $t' \in T(F_{C,D})$ as above, the automaton $\mathcal{B}$ reaches the following state:

Ok if (a) holds,

$Error$ if neither (a) nor (b) holds,

(α, β, Ψ) if (b) holds and :

α is the set of labels of the vertices $x_1, ..., x_p$,

β is the set of labels of the C-vertices of indegree and outdegree 1,

$\Psi = \{(\pi(y_1), \pi(z_1)), ..., (\pi(y_m), \pi(z_m))\}$.

From any finite set $\Psi = \{(a_1, b_1), ..., (a_m, b_m)\} \subseteq (C \cup D)^2$, we define $\Psi_1 := \{a_1, ..., a_m\}$ and $\Psi_2 := \{b_1, ..., b_m\}$.

If $\sigma = (\alpha, \beta, \Psi)$ is reached at the root of t' satisfying (b), then:

(i) the sets α, β, Ψ_1 and Ψ_2 are pairwise disjoint,

(ii) no two pairs in Ψ have a component in common (by (i) this condition reduces to: $a_i \neq a_j$ and $b_i \neq b_j$ for $i \neq j$).

We denote by $D(\sigma)$ the conjunction of these two conditions.

Remarks: When $\mathcal{B}$ reaches state (α, β, Ψ), the labels of the C-vertices of $val(t')$ are all in $\alpha \cup \beta \cup \Psi_1 \cup \Psi_2$, but (α, β, Ψ) does not indicate the set δ of labels of the D-vertices that are on the paths from y_j to z_j but not at their ends. These vertices have indegree and outdegree 1. Since t is assumed correct and irredundant, Lemma 2 yields that $\delta \cap (\alpha \cup \Psi_1 \cup \Psi_2) = \emptyset$.□

The accepting state is Ok. We now describe some transitions.

$\varnothing \to \overrightarrow{\emptyset}$,

$\mathbf{a} \to (\{a\}, \overrightarrow{\emptyset})$ for $a \in C \cup D$,

$\oplus[Ok, \overrightarrow{\emptyset}] \to Ok$,

$\oplus[Ok, q] \to Error$, if $q \neq \overrightarrow{\emptyset}$ (in particular if $q = Ok$),

$\oplus[\sigma, \sigma'] \to \sigma \cup \sigma'$ if $\alpha \cap \alpha' = \emptyset, \Psi \cap \Psi' = \emptyset$ and $D(\sigma \cup \sigma')$ holds,

where, if $\sigma = (\alpha, \beta, \Psi)$ and $\sigma' = (\alpha', \beta', \Psi')$, we have

$\sigma \cup \sigma' := (\alpha \cup \alpha', \beta \cup \beta', \Psi \cup \Psi')$,

$\oplus[\sigma, \sigma'] \to Error$ otherwise.

We denote by a, b any labels in $C \cup D$. The transitions for relabellings $relab_{a \to b}$ in $F_{C,D}$ are:

$relab_{a\to b}[Ok] \to Ok$,

$relab_{a\to b}[\sigma] \to Error$ if $\{a,b\} \subseteq \alpha$ or $D(\sigma')$ does not hold[11]

where σ' is obtained from σ by replacing everywhere a by b;

otherwise $relab_{a\to b}[\sigma] \to \sigma'$.

Let comment on the last case. If $a \in C$ and $a \notin \alpha \cup \beta \cup \Psi_1 \cup \Psi_2$, then $relab_{a\to b}$ has no effect and the last transition yields $\sigma' = \sigma$. If $a \in D$ and $a \notin \alpha \cup \Psi_1 \cup \Psi_2$, it may happen that $a \in \delta$ (cf. the above remarks) hence that $relab_{a\to b}$ might have some effect. But since the input term is irrendundant and correct, Lemma 2 shows that b cannot belong to $\alpha \cup \Psi_1 \cup \Psi_2$. Hence δ is replaced by $\delta[a \to b]$ and nothing else is modified. Hence the transition yields correctly $\sigma' = \sigma$.

Transitions for edge additions are as follows:
$\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Theta)] \to \sigma$ if a or b is not in $\alpha \cup \beta \cup \Psi_1 \cup \Psi_2$,
$\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Theta)] \to Ok$ if $\alpha = \emptyset$ and $\Psi = \{(b,a)\}$,
$\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Theta)] \to (\alpha',\beta',\Psi')$ as described in Table 5.

Condition	α'	β'	Ψ'
$a,b \in \alpha$	$\alpha - \{a,b\}$	β	$\Psi \cup \{(a,b)\}$
$a \in \alpha, (b,c) \in \Psi$	$\alpha - \{a\}$	$\beta \cup \{b\}$	$\Psi \cup \{(a,c)\} - \{(b,c)\}$
$(d,a) \in \Psi, b \in \alpha$	$\alpha - \{b\}$	$\beta \cup \{a\}$	$\Psi \cup \{(d,b)\} - \{(d,a)\}$
$(d,a),(b,c) \in \Psi$	α	$\beta \cup \{a,b\}$	$\Psi \cup \{(d,c)\} - \{(d,a),(b,c)\}$

Table 5 : Transitions $\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Psi)] \to (\alpha',\beta',\Psi')$.

By Properties (i) and (ii), a,b,c,d are pairwise distinct. Finally,

$\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Psi)] \to Error$ in all other cases.

Note that by our usual argument using Lemma 2, if $\overrightarrow{add}_{a\to b}[(\alpha,\beta,\Psi)]$ is to be fired, then b cannot belong to δ.

If $C \cup D$ is finite, $k = |C|$ and $\ell = |D|$, then the number of states is bounded by $3^k.2^\ell.m(k+\ell)$ where $m(n)$ is the number of ordered matchings over n elements (of sets of pairwise disjoint ordered pairs). We have $\lfloor n/2 \rfloor ! < m(n) < 2^{n^2}$. The size of a state is $O((k+\ell)\log(k+\ell))$ and the time for computing a transition is also $O((k+\ell)\log(k+\ell))$.

The property $DirHam$, i.e., the existence of a directed Hamiltonian cycle in the graph G such that $Inc(G) = val(t)$ is expressed by :

"There exists a set U of D-vertices of $val(t)$

such that $P(val(t)[V_{val(t)}, U])$ holds"

[11] That $D(\sigma')$ holds implies that a and b are not both in Ψ_1 and not both in Ψ_2.

(cf. the end of Section 3). Note that $t[V_{val(t)}, U]$ is a correct irredundant term for all sets U if t is so. This property is thus checked by the nondeterministic FA $\mathcal{C}$ obtained from $\mathcal{B}$ by adding the transitions $\mathbf{d} \to \overrightarrow{\emptyset}$ for $d \in D$. These new transitions correspond to the elimination of the edges that will not be on the directed cycle under construction. The FPT-FA $\det(\mathcal{C})$ checks $DirHam$ in time $O(2^{2(k+\ell)^2}.n)$ where n is the number of vertices and edges of the input graph.

6 Conclusion

These results indicate that the tools of [7, 8] can be applied to the verification of MSO_2 properties of graphs of bounded tree-width given by their tree-decompositions. The software AUTOGRAPH[12] can be used basically as it is (up to minor syntactic adaptations) although the algebras of terms describing tree-decompositions and of terms defining clique-width are fairly different (as discussed in [4]).

We have introduced in [4] a variant of tree-width called *special tree-width* intended for the verification of MSO_2 properties. It is weaker than tree-width in the sense that bounded special tree-width implies bounded tree-width but not vice-versa. Its advantage is that *special tree-decompositions* can be formalized in terms of clique-width operations. A decomposition of width p is formalized by a term in $T(F_{[p+2]})$. Hence, this article also provides tools for checking MSO_2 properties by FA based on clique-width operations.

For completeness, we also cite [17] where a completely different method is used to check MSO_2 properties of graphs of bounded tree-width.

Acknowledgements : I thank R. Sampaio for fruitful discussions on these topics. I thank I. Durand for her ongoing collaboration and her implementation of fly-automata.

References

[1] H. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* **25** (1996) 1305-1317.

[2] T. Bouvier, *Graphes et décompositions*, Doctoral dissertation, Bordeaux University, December 2014.

[3] B. Courcelle: The Monadic Second-Order Logic of Graphs I: Recognizable Sets of Finite Graphs. *Inf. Comput.* **85** (1990) 12-75.

[12] The system AUTOGRAPH that builds and runs FA is written in LISP [6] and http://dept-info.labri.u-bordeaux.fr/~idurand/autograph

[4] B. Courcelle, On the model-checking of monadic second-order formulas with edge set quantifications, *Discrete Applied Mathematics* **160** (2012) 866-887.

[5] B. Courcelle, Clique-width and tree-width of sparse graphs. 2015. Submitted for publication.

[6] B. Courcelle and I. Durand, Fly-Automata, their properties and applications. CIAA 2011 (*Conference on Implementation and Application of Automata*), *Lecture Notes in Computer Science* **6807** (2011) 264-272.

[7] B. Courcelle and I. Durand, Automata for the verification of monadic second-order graph properties, *J. Applied Logic* **10** (2012) 368-409.

[8] B. Courcelle and I. Durand, Computations by fly-automata beyond monadic second-order logic, June 2013. To appear in *Theoret. Comput. Sci.* Short version in Proceedings of *Conference on Algebraic Informatics*, *Lecture Notes in Computer Science* **8080** (2013) 211-222.

[9] B. Courcelle and I. Durand, Fly-automata, model-checking and recognizability, Proceedings of the workshop *Frontiers of Recognizability,* CIRM, Marseille, April 2014, http://fr.arxiv.org/abs/1409.5368

[10] B. Courcelle and J. Engelfriet, *Graph structure and monadic second-order logic, a language theoretic approach*, Volume **138** of *Encyclopedia of mathematics and its application*, Cambridge University Press, June 2012.

[11] B. Courcelle, J. Makowsky and U. Rotics, Linear-time solvable optimization problems on graphs of bounded clique-width, *Theory Comput. Syst.* **33** (2000) 125-150.

[12] R. Downey and M. Fellows, *Parameterized complexity*, Springer-Verlag, 1999.

[13] R. Downey and M. Fellows, *Fundamentals of parameterized complexity*, Springer-Verlag, 2013.

[14] F. Fomin, P. Golovach, D. Lokshtanov and S. Saurabh, Algorithmic lower bounds for problems parameterized with clique-width. *Proceedings of SODA (Symposium on Discrete Algorithms)* 2010, pp. 493-502.

[15] M. Frick and M. Grohe, The complexity of first-order and monadic second-order logic revisited, *Ann. Pure Appl. Logic* **130** (2004) 3-31.

[16] F. Gurski and E. Wanke: The Tree-Width of Clique-Width Bounded Graphs Without $K_{n,n}$. Proceedings of WG 2000 (*Graph-Theoretic Concepts in Computer Science, 26th International Workshop*), *Lec. Notes Comput. Sci.* **1928** (2000) 196-205.

[17] J. Kneis, A. Langer, P. Rossmanith: Courcelle's theorem - A game-theoretic approach. *Discrete Optimization* **8** (2011) 568-594.