Comparing tree-width and clique-width for degree-constrained graphs

Bruno Courcelle

Bordeaux University, LaBRI (CNRS laboratory)

Topics

Bounds on clique-width in view of constructing usable "fly"-automata for checking MSO properties (and doing other computations) by FPT algorithms parameterized by tree-width or clique-width.

Automata for MSO model-checking.

How to implement the following algorithmic meta-theorem ?

Theorem : (1) For every k, every MSO graph property can be checked in linear time for graphs of *clique-width* at most k, given by terms witnessing this bound.

(2) For every k, every MSO2 (= MSO with edge quantifications) graph property can be checked in linear time for graphs of *tree-width* at most k, (*preferably* given by a term witnessing this bound).

In both cases, a *finite* automaton can be constructed to recognize:

- the terms written over the *finite set* of operations
- that define graphs satisfying the considered property.

Two (well-known) difficulties:

Parsing input graphs (finding the terms, equivalently, the corresponding decompositions): NP-complete problems;

Huge sizes of the automata: unavoidable (Frick-Grohe 2004).

To remedy the size problem:

Irène Durand and myself have introduced fly-automata (in French "automates programmés") whose states and transitions are described and not tabulated.

A deterministic "finite" fly-automaton with 2^{1000} states computes "on the fly" only 100 states and transitions on an input term of size 100.

As states are not listed, a fly-automaton can use an infinite set of states : a state may include counters – and thus compute values :
e.g., the number of p-colorings, or of "acyclic" p-colorings, of a graph.

The system AUTOGRAPH by Irène Durand is based on clique-width.

[B.C.&I.D.: *Automata for the verification of monadic second-order graph properties*, J. Applied Logic, 10 (2012) 368-409].

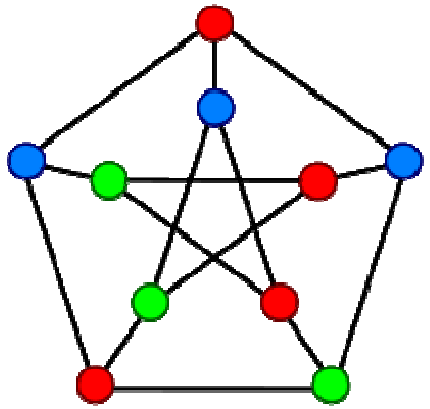
Successful computations: (1) Number of 3-colorings of the 6 x 525 grid (of clique-width 8) in 10 minutes.

(2) 4-acyclic-colorability of the Petersen graph (clique-width 5) in 1.5 minute, from a term in $T(F_6)$.

(3-colorable but not acyclically;

red and **green** vertices

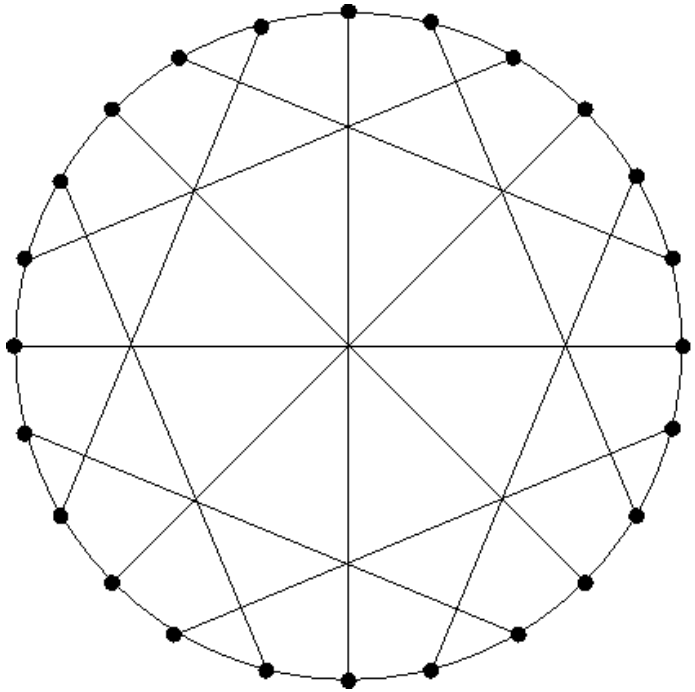
induce a cycle).



(3) The McGee graph (of clique-width 8)

defined by a term in $T(F_{10})$

of size 99 and height 76.



It is 3-acyclically colourable.

Checked in 40 minutes.

(Even in 2 seconds by enumerating the accepting runs, and stopping as soon as a success is found).

Various facts

(1) What matters in a fly-automaton is not the number of states but the maximum size of a state in a run on the given term (this size determines the time taken to compute each transition).

(2) Theorem (2) for MSO2 and bounded tree-width reduces to Theorem (1) for MSO and bounded clique-width because we can replace:

G → *Inc*(G) (its incidence graph)

MSO2 on G = MSO on *Inc*(G),

twd(G) = k $\Rightarrow$ cwd(*Inc*(G)) $\leq$ k+3 (T. Bouvier, 2014; see below).

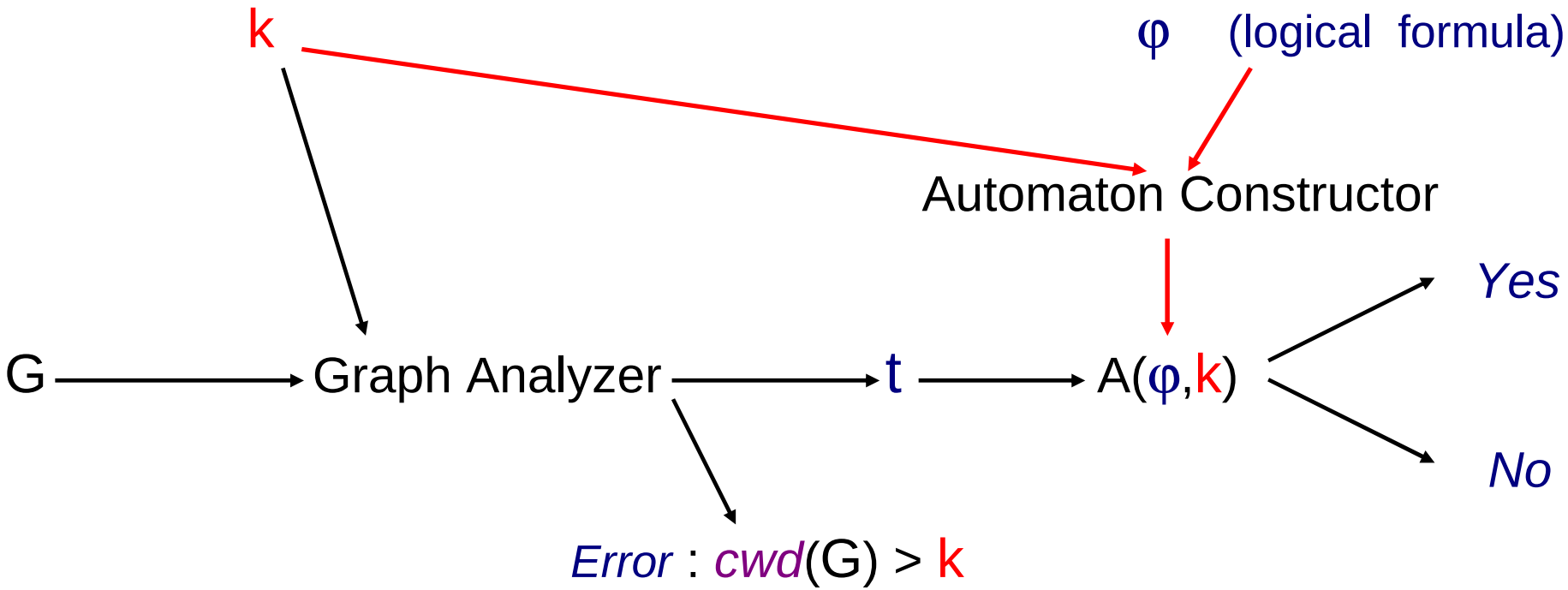
No exponential jump !

(3) We construct fly-automata in uniform ways from logical descriptions of the problems.

(4) Parsing is difficult in general but the graphs arising from concrete problems are not random. They usually have "natural" hierarchical decompositions from which terms of small tree-width or clique-width are not too hard to construct.

This situation arises in compilation (flow-graphs of structured programs), in linguistics and in chemistry. It is thus interesting to develop specific parsing algorithms for graph classes relevant to particular applications.

The MSO meta-theorem through *finite* automata: the basic scheme

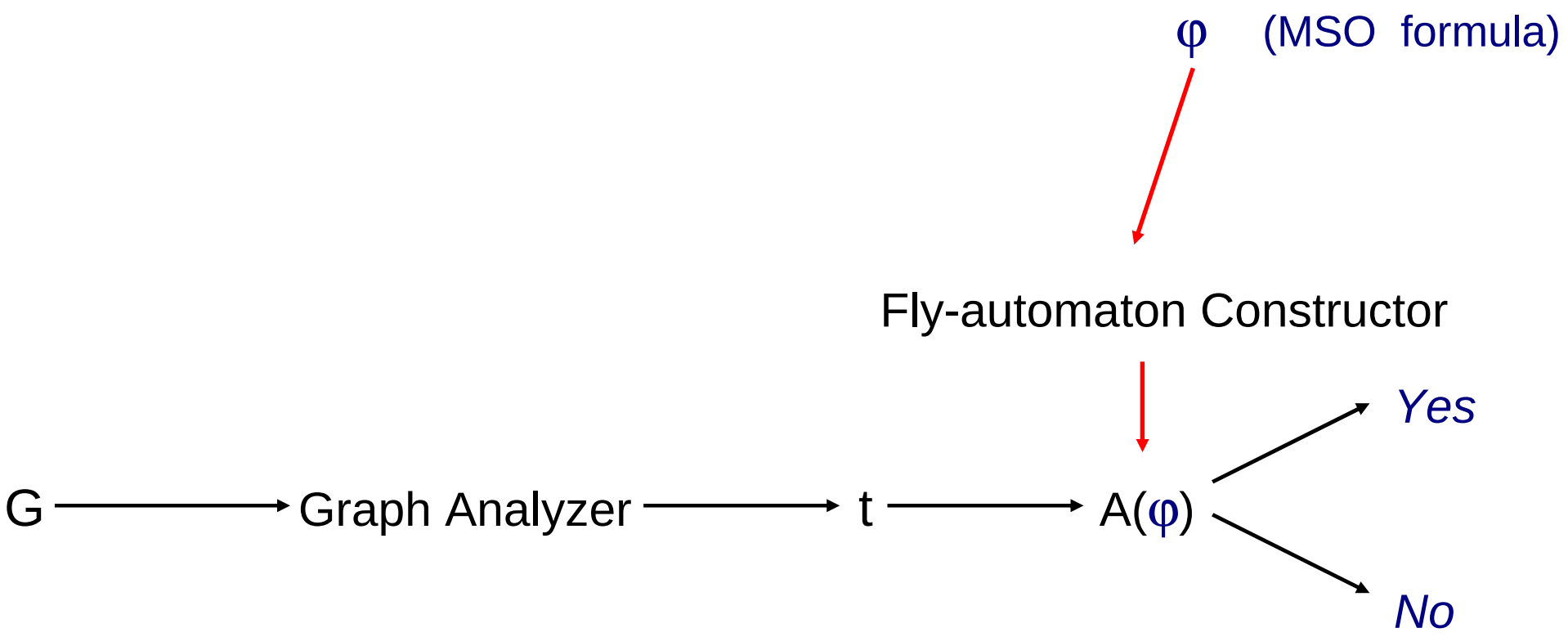


Steps ⟶ are done "once for all", independently of G

A(φ,k): "finite" automaton, running on terms t

cwd can be replaced by tree-width or rank-width.

The MSO meta-theorem through *fly-automata*: a simpler scheme



A(φ): *infinite fly-automaton* over the countable set of all graph operations that define clique-width. The time taken by A(φ) depends on the labels that occur in t, not only on the size of G or of t.

Definition : Fly-automaton (FA)

$A = < F, Q, \delta,$ Out $>$

F : finite or countable (effective) signature (set of operations),

Q : finite or countable (effective) set of states (integers, pairs of integers, finite sets of integers: states can be encoded as finite words, integers in binary),

Out : Q → *D* (an effective domain, i.e., set of finite words), computable.

δ : computable (bottom-up) transition function

Nondeterministic case : δ is *finitely multi-valued*.

This automaton defines a computable function : T(F) → *D*

(or : T(F) → *P*(*D*) if it is not deterministic)

If *D* = { *True, False* }, it defines a decidable property, equivalently, a decidable subset of T(F).

Deterministic computation of a nondeterministic FA :

bottom-up computation of *finite* sets of states (classical simulation of the determinized automaton): these states are the useful ones of the *determinized automaton*; these sets are *finite* because the transition function is finitely multivalued.

Fly-automata are "implicitly determinized" and they run deterministically.

Examples of computations by fly-automata:

- the number of connected components,
- the number of p-colorings
- more generally : the number of accepting runs of a nondeterministic automaton,
- whether a graph is regular (not an MSO property).

Computation time of a fly-automaton

F : all "clique-width" operations , F_k : those using labels 1, …, k.

On term $t \in T(F_k)$ defining G(t) with n vertices, if a fly-automaton takes time bounded by :

$(k + n)^c \Rightarrow$ it is a P-FA (a polynomial-time FA),

$f(k).n^c \Rightarrow$ it is an FPT-FA,

$a.n^{g(k)} \Rightarrow$ it is an XP-FA.

The associated algorithm is, respectively, polynomial-time, FPT or XP for clique-width as parameter.

Theorem [B.C & I.D.] : For each MSO property P, one can construct a single (infinite) FPT-FA over F that recognizes the terms t in T(F) such that P(G(t)) holds.

For each k, its restriction to the finite signature F_k is finite.

Consequence : The same automaton (the same model-checking program) can be used for graphs of any clique-width.

Comparing tree-width and clique-width and the corresponding decompositions

Objectives : to understand the properties of these parameters,

to transform tree-decompositions (for which many algos exist) into clique-width terms,

to use fly-automata to check MSO2 properties of graphs of bounded tree-width.

We discuss simple undirected graphs. All results have easy extensions to directed graphs with similar bounds.

Comparing tree-width (twd(G)) and clique-width (cwd(G)).

All graphs :

cwd(G) $\leq$ $3.2^{twd(G)-1}$ with ETT (Corneil&Rotics)

No : cwd(G) = *poly*(twd(G))

No : twd(G) = f(cwd(G)).

ETT means Easy Transformation of underlying Trees

Graphs of degree $\leq$ d :

cwd(G) $\leq$ 20.d.(twd(G) +1) +2 *without* ETT (yet)

proof uses "tree-partition-width" (D. Wood, to be cleaned up).

twd(G) $\leq$ 3.d.cwd(G) -1 with ETT (Gurski&Wanke)

Hence : tree-width and clique-width are *linearly equivalent*.

Planar graphs : We known that cwd(G) $\leq$ f(twd(G)) for some exponential function f.

New : cwd(G) = O(twd(G)) with ETT (proof to be written)

twd(G) $\leq$ 6.cwd(G) -1 with ETT (G&W)

Uniformly q-sparse graphs (also said q-degenerated).

Question : cwd(G) = *poly*(twd(G)) ? (we have $\leq$ f'(twd(G))

twd(G) $\leq$ 6.q.cwd(G) -1 with ETT (G&W)

Incidence graphs, edge quantifications and tree-width.

For $G = (V_G , edg_G(.,.))$, Inc(G) := $(V_G \cup E_G , inc_G(.,.))$

is its *incidence graph*: $inc_G(u,e) \Leftrightarrow$ u is an end of e.

Facts : twd(G) = twd(Inc(G) (for simple loop-free graphs)

MSO formulas over Inc(G) can use quantifications on edges and express more properties. They are MSO2 on G.

Results: 1. $cwd(Inc(G)) \leq twd(G) + 3$ with ETT (T. Bouvier, Ph D, 2014) improves the exponential bound from C&R.

2. $twd(G) \leq 2.cwd(Inc(G)) - 1$ with ETT (B)

3. No : $cwd(Inc(G)) \leq f(cwd(G))$

4. $cwd(G) \leq 2^{cwd(Inc(G))+1} - 1$ but not *poly*$(cwd(Inc(G)))$ (C,B).

Because $cwd(Inc(G)) \leq twd(G) + 3$ avoids exponential blow-up, tools for "bounded clique-width" and MSO formulas are applicable to "bounded tree-width" and MSO2 formulas.

As $twd(G) = O(cwd(Inc(G))$, incidence graphs (for MSO2 model-checking) "only work" for graphs G of bounded tree-width.

Bipartite graphs of *semi-degree* $\leq$ d :

all vertices of one side have degree $\leq$ d.

d = 2 : they are incidence graphs.

d $\geq$ 3 : they are uniformly d-sparse graphs.

cwd(G) $\leq$ f(twd(G)) for some exponential function f,

New : $cwd(G) = O(twd(G)^d)$ (proof to be written)

twd(G)) $\leq$ 6.d.cwd(G) -1 with ETT (G&W)

Bipartite graphs occur in many cases : in particular graph encodings of SAT problems.

Conclusion

MSO / cwd tools are applicable to MSO2 / twd model-checking,

New upper-bounds for clique-width in graph classes of practical interest.

Easy Transformations of decomposition Trees in most cases.

Definition : Clique-width

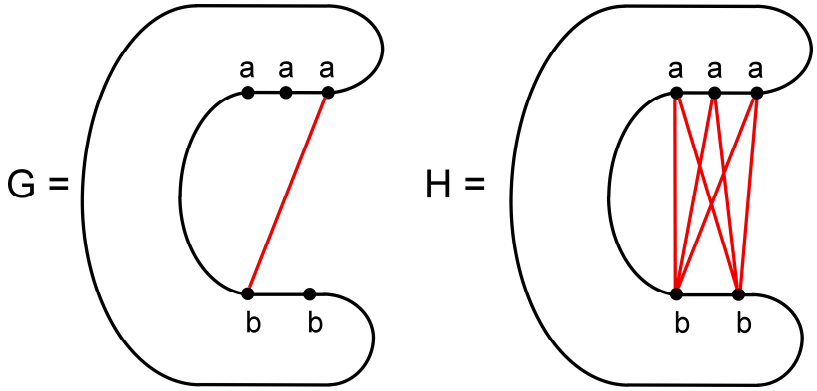
It is defined from graph operations. Graphs are simple, directed or not, and labelled by *a* , *b* , *c, ...* . A vertex labelled by *a* is called an *a-vertex*.

One binary operation: *disjoint union* : $\oplus$

Unary operations: edge addition denoted by $Add_{a,b}$

$Add_{a,b}(G)$ is G augmented with undirected edges between every *a*-vertex and every *b*-vertex.

The number of added edges depends on the argument graph.



H = $Add_{a,b}(G)$; only 5 new edges added

Directed edges can be defined similarly.

Vertex relabellings :

$Relab_{a \longrightarrow b}(G)$ is G with every *a*-vertex is made into a *b*-vertex

Basic graphs : those with a single vertex **a**, labelled by *a*.

Definition: A graph G has clique-width (denoted by cwd(G)) $\leq$ k

$\Leftrightarrow$ G=G(t) is defined by a term t using $\leq$ k labels.

Example : Cliques have clique-width 2.

K_n is defined by t_n where t_{n+1} =

$Relab_{b \longrightarrow a}(Add_{a,b} (t_n \oplus \mathbf{b}))$