Tree-width, clique-width and fly-automata

Bruno Courcelle

Bordeaux University, LaBRI (CNRS laboratory)

References : B.C, Irène Durand: Automata for the verification of monadic second-order graph properties, *J. Applied Logic* 10 (2012) 368-409

B.C.: From tree-decompositions to clique-width terms, *Discrete Applied Maths*, 2017, in press.

B.C.: Fly-automata for checking MSO2 graph properties, *Discrete Applied Maths*, 2017, in press.

Topics

Fixed-parameter tractable (FPT) graph algorithms for monadic second-order (MSO) expressible problems,

for graphs of bounded tree-width (twd) or clique-width (cwd),

based on automata running on algebraic terms denoting the (decomposed) input graphs.

Can compute values, not only *True / False* answers.

Tools: Fly-automata (FA): they *compute* their transitions, to overcome the "huge size problem",

Tree-decompositions encoded by clique-width terms,

Linear bounds on cwd in terms of twd for sparse graphs.

The basic theorem : Each MSO property of graphs of cwd or twd at most *k* is decidable in time *f(k)* **x** *number of vertices*.

Facts: Extends to MSO properties expressed with edge set quantifications, for graphs of bounded tree-width (*not* bounded cwd).

Graphs given with relevant decompositions, of “small width”.

Optimal decompositions are difficult to construct (NP-complete problems). But optimality is not essential.

Computation of graph evaluations

P(X) is a property of tuples X of sets of vertices (usually MSO expressible).

∃ X.P(X) : the basic, "Boolean evaluation".

X.P(X) : number of satisfying tuples X.

Sp X.P(X) : spectrum = the set of tuples of cardinalities of the components of the tuples X that satisfy P(X).

MinCard X.P(X) : minimum cardinality of X satisfying P(X).

Informal review of definitions and basic facts.

1) Graphs are finite, simple, loop-free, directed or not.

A graph G can be given by the logical structure

$$(V_G , edg_G(.,.)) = \text{(vertices, adjacency relation)}$$

2) Monadic second-order (MSO) formulas can express p-colorability (and variants), transitive closure, properties of paths, connectedness, planarity (via Kuratowski), *etc…*

Examples : *3-colorability :*

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

The graph is not connected :

$$\exists Z\ (\ \exists x \in Z\ \wedge\ \exists y \notin Z\ \wedge\ (\forall u,v\ (u \in Z\ \wedge\ edg(u,v) \Rightarrow v \in Z)\ )$$

Planarity is MSO-expressible (no minor K_5 or $K_{3,3}$).

3) Alternative description of graphs :

Inc(G) := (V_G U E_G , $inc_G(.,.)$)

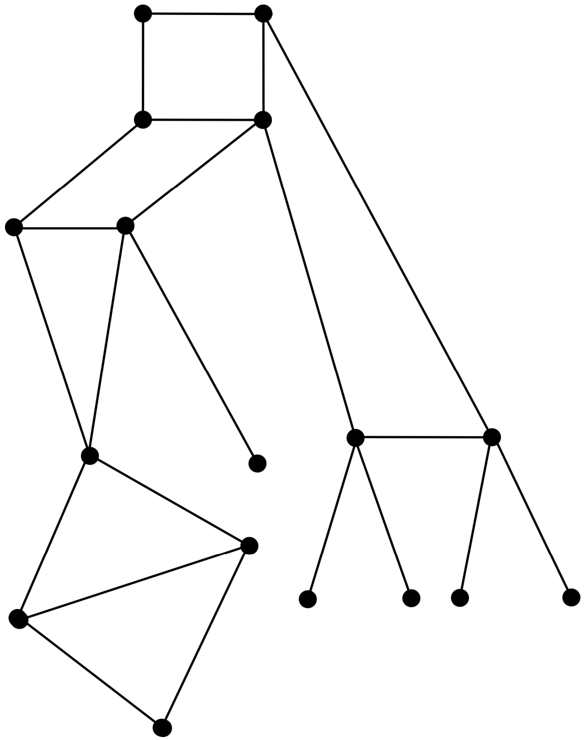
= (vertices *and edges*, incidence relation)

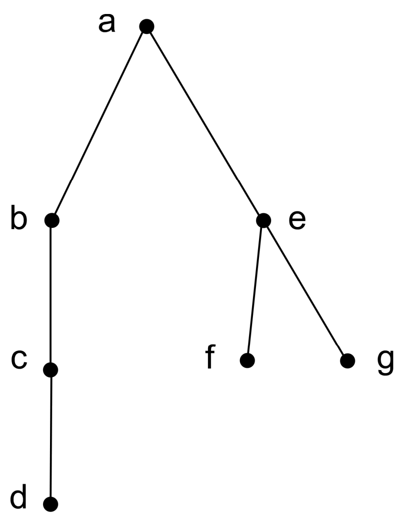
→ the bipartite *incidence graph* of G.

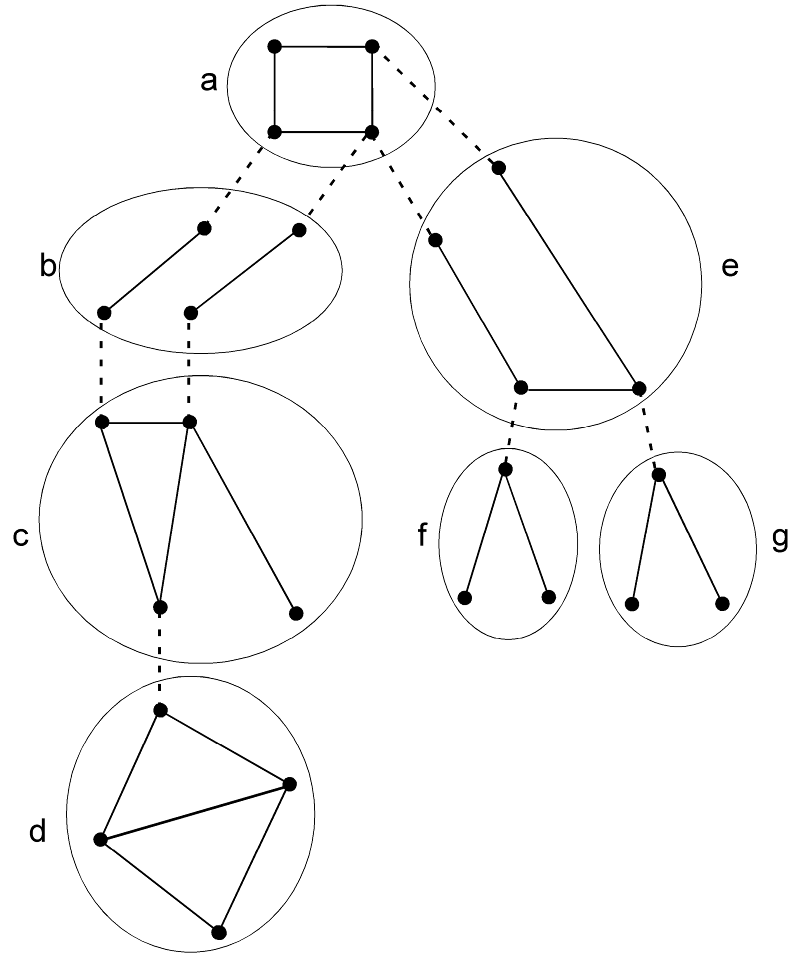
MSO formulas on Inc(G) can use quantifications on sets of edges of the considered graph G.

Expressing Hamiltonicity of G is possible by an MSO formula on Inc(G) but not on G (edge set quantifications are needed).

4) Tree-width (twd(G)) is well-known.







width of decomposition : 3

dotted lines : equal vertices

5) Clique-width : algebraic construction of graphs

Vertices are labelled by *a,b,c, ...* . A vertex labelled by *a* is an *a-vertex*.

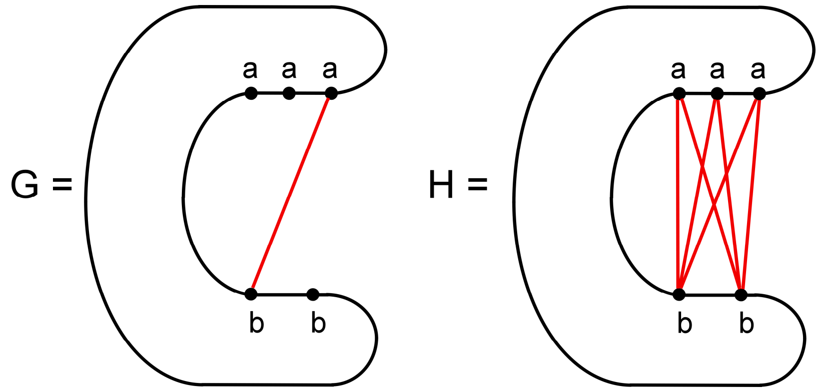
Binary operation: disjoint union : $\oplus$

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

$Add_{a,b}$(G) is G augmented

with (un)directed edges from (between)

every *a*-vertex to (and) every *b*-vertex.



vertex relabellings :

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

Basic graphs : **a** denotes a vertex labelled by *a*

The clique-width of G, denoted by cwd(G), is the smallest k such that G is defined by a term using k labels.

Such a term is a decomposition of G as a gluing of complete bipartite graphs. k indicates the "complexity of gluings", not size of components.

Classes of bounded clique-width:

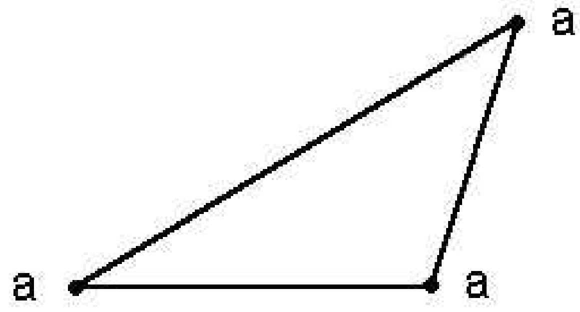
cographs, cliques, complete bipartite graphs, trees,

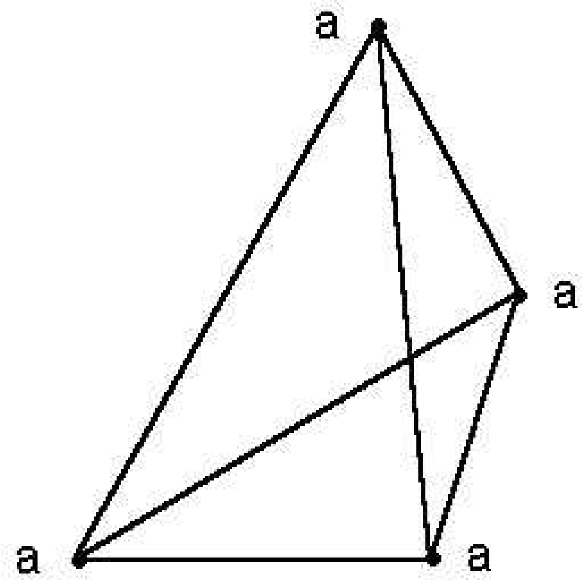
any class of bounded tree-width.

Classes of unbounded clique-width:

Planar graphs, chordal graphs.

Example 1 : Cliques (a-labelled) have clique-width 2 and unbounded tree-width.





K_n is defined by t_n where $t_1 = \mathbf{a}$

$t_{n+1} = Relab_{b \longrightarrow a}(Add_{a,b} (t_n \oplus \mathbf{b}))$

Example 2 : Cographs (a-labelled) are generated by $\oplus$ and $\otimes$ defined by:

$G \otimes H = Relab_{b \longrightarrow a}(Add_{a,b} (G \oplus Relab_{a \longrightarrow b}(H)))$

$= G \oplus H$ with "all edges" between G and H.

Remark : An algebraic expression of tree-width is possible, by using *parallel composition* $G // H$ instead of disjoint union $G \oplus H$.

This operation glues G and H by fusing, for each label a, the (*unique*) a-vertex of G and the (*unique*) a-vertex of H.

But the construction of an automaton running on terms over // denoting graphs G of twd $\leq k$ intended to check an MSO property of Inc(G) is more complicated because of these fusions. The basic fact for $\oplus$ is : $G \oplus H \models \varphi(X)$ if and only if

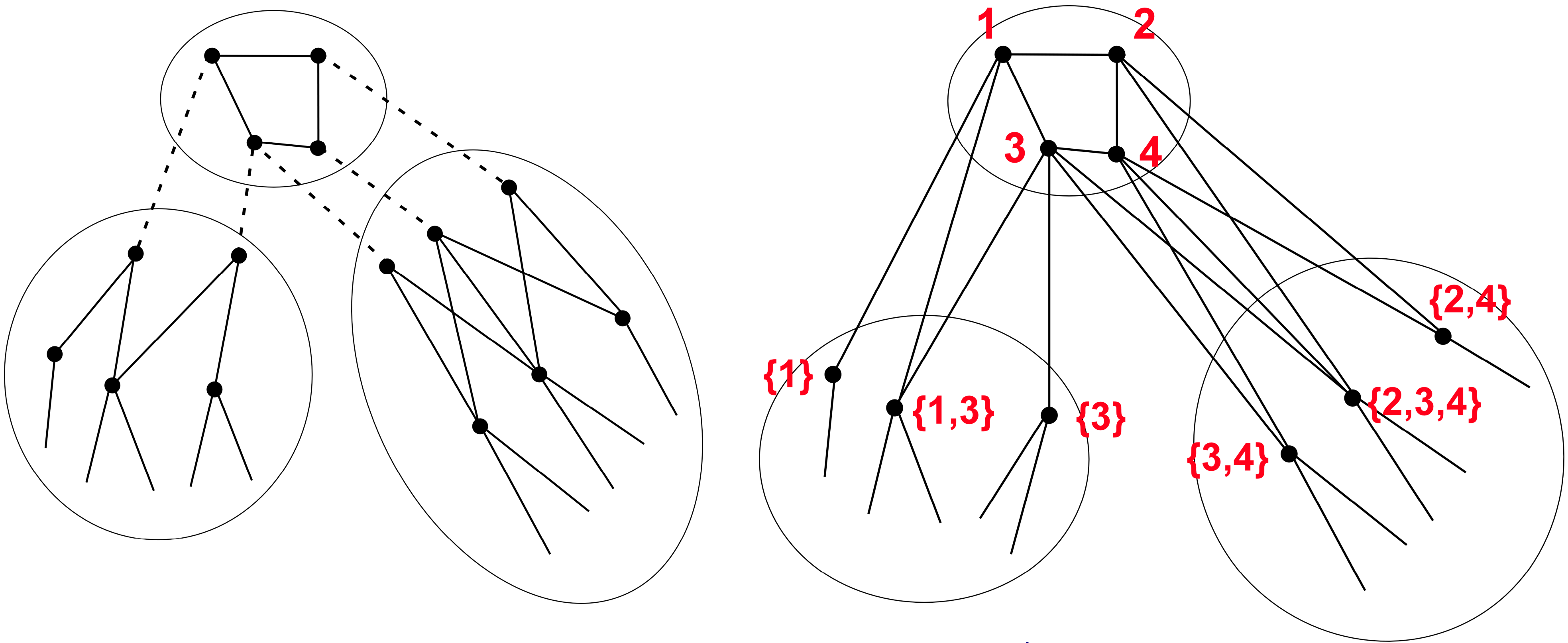
$$G \models \psi_1(X \cap V_G) \text{ and } H \models \theta_1(X \cap V_H)$$

$$\text{or } G \models \psi_2(X \cap V_G) \text{ and } H \models \theta_2(X \cap V_H) \quad \ldots$$

$$\text{or } G \models \psi_p(X \cap V_G) \text{ and } H \models \theta_p(X \cap V_H)$$

Comparing tree-width and clique-width (undirected graphs)

cwd (G) $\leq$ 3. 2 $^{\mathbf{twd(G) - 1}}$ (Corneil & Rotics, the exponential is not avoidable)



If a box of the tree-dec has k vertices, then 2^k-1 labels may be necessary to specify how the vertices below it are linked to its vertices.

For which classes do we have $cwd(G) = O(twd(G)^{\mathbf{c}})$ for fixed **c** ?

Graph class	cwd(G) where k = twd(G)
planar	6k – 9 (32k – 57 if directed)
degree < d	k.d + 1
incidence graph	k + 3 (2k + 4 if directed)
1-planar	O(k)
p-planar	**O(k) ?**
at most **q**. n edges for n vertices	$O(k^{\mathbf{q}})$ where q << k

These results hold for directed graphs.

Remark : About incidence graphs of graphs of bounded tree-width and MSO_2 properties.

MSO_2 means expressed by an MSO formula using edge set quantifications.

Example : There exists a set of edges forming a perfect matching, or forming a Hamiltonian path. Not possible without such quantifications.

1) From of a tree-decomposition of G of width k, we construct a clique-width term t for Inc(G) of "small" width k+3 (or 2k+4); no exp. !

2) We translate an MSO_2 formula φ for G into an MSO formula θ for Inc(G).

3) The corresponding automaton A(θ) takes term t as input.

More remarks to come.

Proof method for making tree-decompositions into cwd terms

For a graph G and Y a set of vertices :

$\mu_G(Y)$:= the number of sets $N_G(x) \cap Y$ for $x \notin Y$. ($N_G(x)$: neighbours of x)

Lemma : If $twd(G) \leq k$, and $\mu_G(Y) \leq m$ whenever $|Y| \leq k + 1$,

then $cwd(G) \leq m + 1$.

For each graph class, we bound $\mu_G(Y)$ in terms of $|Y|$.

For planar graphs, we use the bound 3n - 6 on the number of edges ;

for q-sparse graphs, we use an orientation of indegree at most q.

In all cases we transform a tree-decomposition into a clique-width term based on the same tree.

Proof sketch for planar graphs.

Enough to consider bipartite graph with vertex set X U Y and $|Y| = k$.

There are at most k+1 sets $N_G(x)$ for x of degree 0 or 1 $(x \in X)$.

There are at most 3k-6 sets $N_G(x)$ for x of degree 2 : each of them corresponds to an edge of a planar graph with vertex set Y.

There are at most 2k-4 vertices x of degree > 2 : let Z be these vertices : $3.|Z| \leq |E| \leq 2.(|Z| + k) - 4$ (planar bipartite).

Total : k+1 + 3k-6 + 2k -4 = 6k - 9.

How to specify tree-decompositions ?

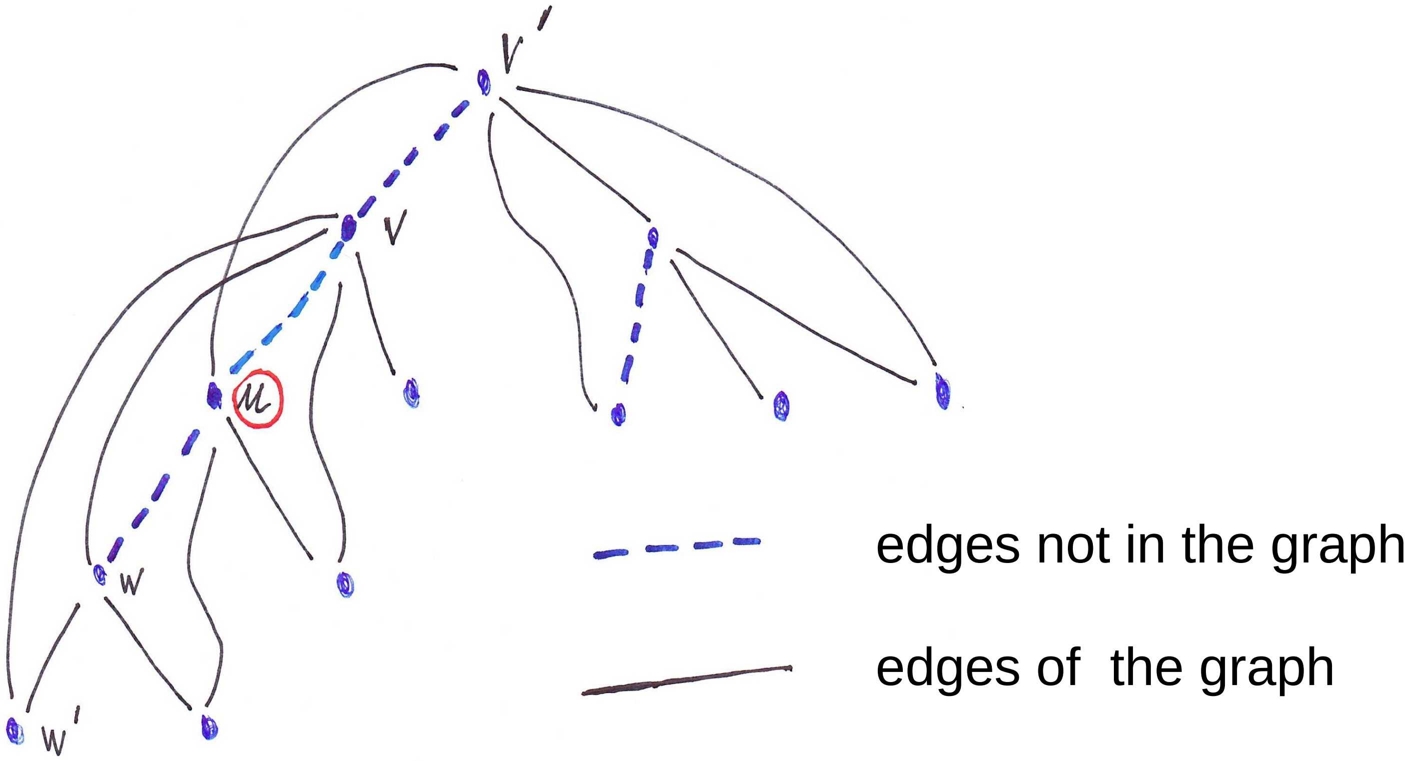
Instead of the classical definition (T,f), we use *partial k-trees* in the following way. A *normal tree* T for a graph G is :

rooted, its nodes *are the vertices* of the graph and adjacent vertices of the graph are *comparable* for the ancestor relation of T.

Then T is the tree of a tree-decomposition (T, f_T) where :

$$f_T(u) := \{u\} \cup \{ v >_T u \ / \ v \text{ is adjacent to some } w \leq_T u \}.$$

Every tree-dec (T ', f) can be made (T, f_T) of same width for a normal tree T (by contracting edges in T ' and inserting nodes on edges; no complicated transformation).



f_T (u) := {u, v, v'} : the edges from w, w' "jump" over u

Remarks :

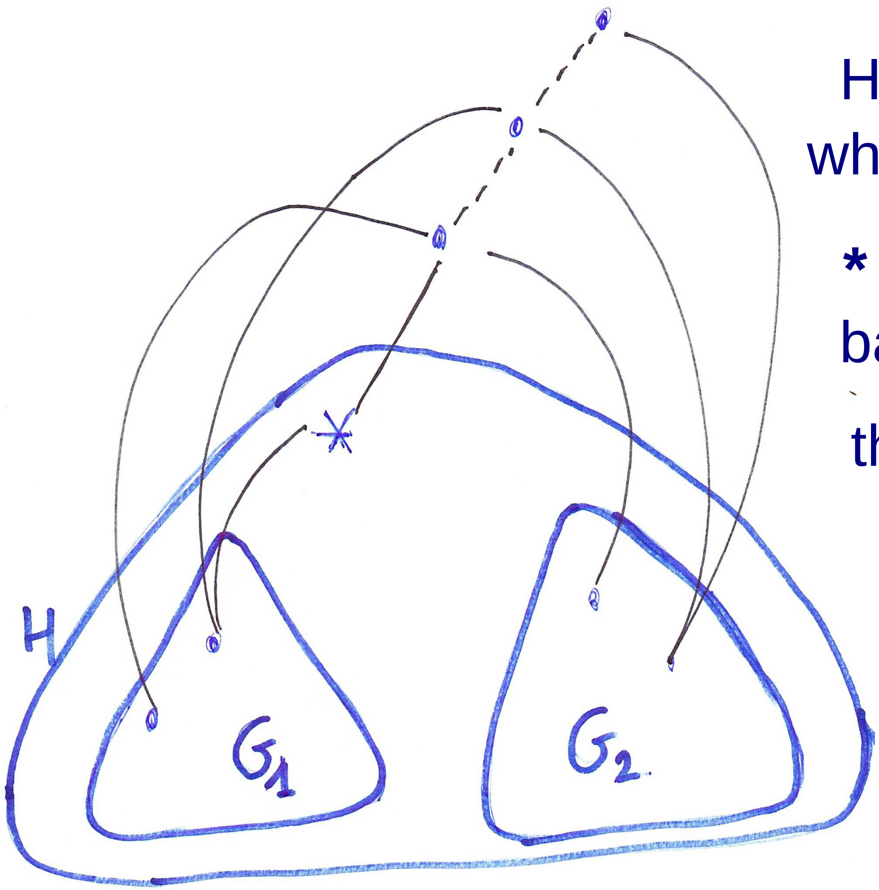
1. We get a *compact data structure* for the graph and a tree-decomposition : $R = (V_G, edg_G, parent_T)$

from which f_T (describing the "bags") is easily computable.

2. This triple is also a *convenient logical structure* : the bags can be described by an MSO formula $\varphi(u,X)$ saying $X = f_T(u)$ (in R)

3. This description corresponds to the notion of a partial k-tree, obtained by edge deletions from a k-tree.

Bottom-up inductive construction of a clique-width term from a normal tree-decomposition.



H = RELAB (ADD($G_1 \oplus G_2 \oplus *$))
where ADD adds the edges between $*$ and the vertices in $G_1 \oplus G_2$, on the basis of the labels in $G_1 \oplus G_2$ that encode subsets of $f_T(*)$.
The number of such labels is bounded by $\mu_G(f_T(*))$.
RELAB : relabellings to update the labels in $G_1 \oplus G_2$ and change $*$ into the correct label for H.

Remark : In this construction, $add_{a,b}$ only creates "stars" $K_{1,p}$, but no $K_{q,p}$. The full power of edge addition is not used. We do not get optimal clique-width (as examples can show).

Conclusion: From a "good" tree-decomposition of a sparse graph (planar, bounded degree, etc…), we can get a "good" clique-width term, of comparable width (avoiding the general exponential jump).

There are many algorithms that construct "good" (not optimal) tree-decompositions, but not so many that construct "good" clique-width terms. Algorithms based on rank-width do not give "good terms".

Clique-width terms yield easier constructions of fly-automata than tree-decompositions.

Fly-automata for the verification of MSO graph properties

Standard proof of the basic theorem : one constructs, for each MSO formula φ and integer k, a finite automaton $A(\varphi,k)$ that takes as input a term denoting a graph G of clique-width $\leq k$ and answers in time $f(k).n$ whether $G \models \varphi$ (where n is the number of vertices).

The construction is by induction on the structure of φ.

Difficulty : The *finite* automaton A(φ,k) is too large to be implemented by a transition table as usual as soon as k $\geq$ 2 : 2^(2^(…2^k)..)) states, because of quantifier alternations.

To overcome this difficulty, we use fly-automata whose states and transitions are described and not tabulated. Only the (say 100) transitions necessary for an input term (say of size 100) are computed "on the fly".

Sets of states can be infinite and fly-automata can compute values, for example, the number of p-colorings of a graph.

Fly-automaton (FA)

Definition : A = < F, Q, δ, Out > (FA that computes a function).

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

Q : finite or countable (effective) set of states (integers, pairs of integers, *etc. :* states are encoded by finite words),

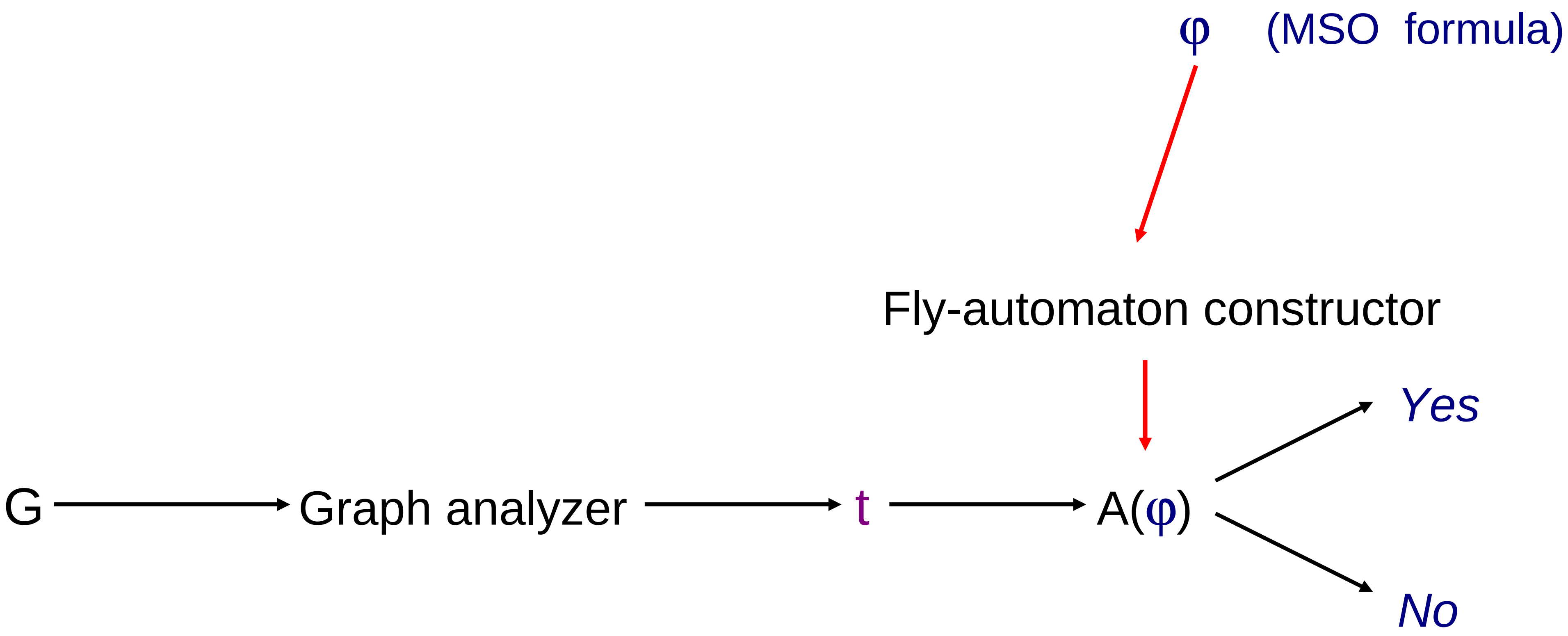
Out : Q → *D* , computable (*D* is an effective set, coded by finite words).

δ : computable (bottom-up) transition function

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

An FA defines a computable function : T(F) → *D* , hence, a decidable property if *D* = {*True, False*}.

The MSO meta-theorem through *fly-automata*



A(φ): *a single infinite fly-automaton*. The time taken by A(φ) is f(k).n where k depends on the operations occurring in t and bounds the tree-width or clique-width of G.

Computation time of a fly-automaton (FA)

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

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

takes time bounded by :

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

$f(k).\ n^c$ → it is an FPT-FA,

$a.\ n^{g(k)}$ → it is an XP-FA.

The associated algorithm is polynomial-time, FPT or XP for clique-width as parameter. (The important notion is the max. size of a state.)

All dynamic programming algorithms based on clique-width terms can be described by FA.

Fly-automata can be constructed :

- either "directly", from our understanding of the considered graph properties,

- or "automatically" from a logical description,

- or by combining previously constructed automata.

Example of a direct construction for p-coloring :

Checking that a "guessed" p-coloring is good: a state is a set of pairs (a, j) where a is a label and j is a color (among 1, …, p) or Error.

Checking the existence of a good p-coloring : a set of such states, in practice not of maximal (exponential) size.

Combinations and transformations of fly-automata.

*Product o*f A and B : states are pairs of a state of A and one of B.

Determinization of A : states of Det(A) are finite sets of states of A because the transition is *finitely* multi-valued. At each position in the term, Det(A) gives the finitely many states that can in some computation (the automaton A can be infinite).

Counting determinization of A, yielding CDet(A): a state of CDet(A) is a finite multi-set of states of A (giving the *number of runs* that can yield a state of A, not only the existence).

Inductive construction for $\exists \underline{X}.\ \varphi(\underline{X})$ with $\varphi(\underline{X})$ MSO formula

Atomic formulas (for example $X \subseteq Y$, edg(X,Y)) : direct constructions

$\neg$ P (negation) : as FA are run deterministically (by computing at each position the finite set of reachable states), it suffices to exchange accepting and non-accepting states.

$P \wedge Q$, $P \vee Q$: products of automata.

How to handle free variables for queries $\varphi(\underline{X})$ and for $\exists \underline{X}.\varphi(\underline{X})$?

Terms are equipped with Booleans that encode assignments of vertex sets $V_1,\ldots,V_p$ to the free set variables $X_1,\ldots,X_p$ of MSO formulas (*formulas are written without first-order variables*):

1) we replace in F each **a** by the nullary symbol

(**a**, $(w_1,\ldots,w_p)$), $w_i \in \{0,1\}$: we get $F^{(p)}$ (*only nullary symbols are modified*);

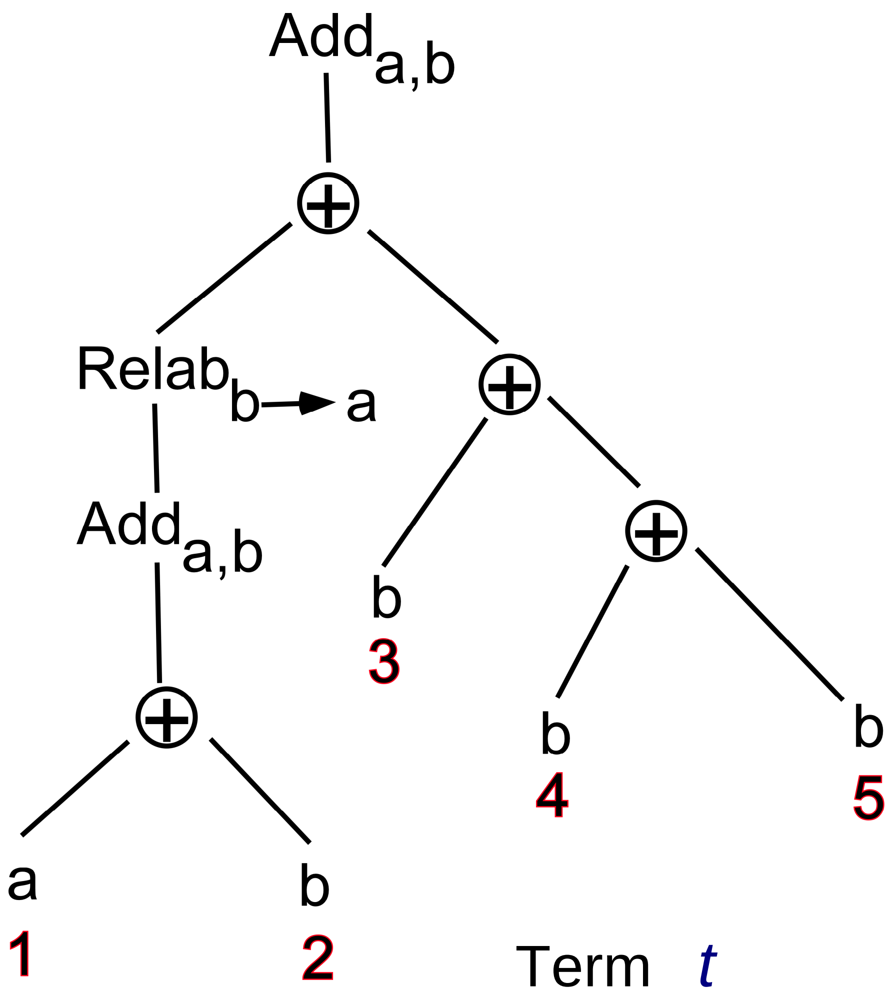
2) a term *s* in $\mathbf{T}(F^{(p)})$ encodes a term *t* in $\mathbf{T}(F)$ and an assignment of sets $V_1,\ldots,V_p$ to the set variables $X_1,\ldots,X_p$:

if *u* is an occurrence of (**a**, $(w_1,..,w_p)$), then

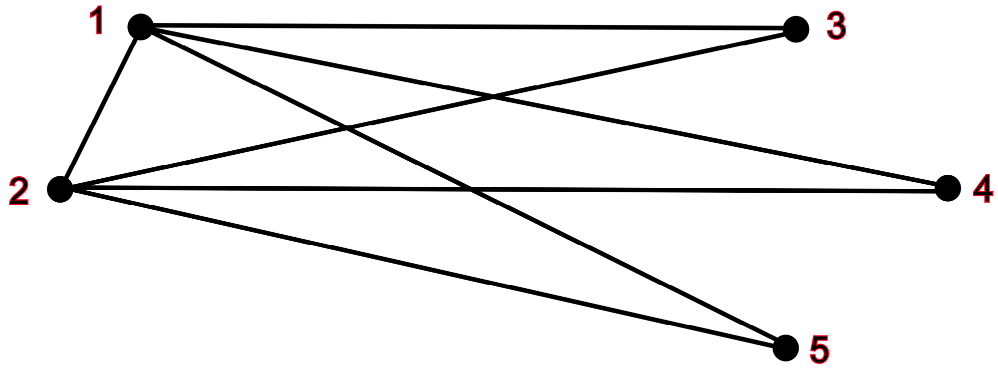
$w_i = 1$ if and only if $u \in V_i$.

3) *s* is denoted by $t * (V_1,\ldots,V_p)$

Example

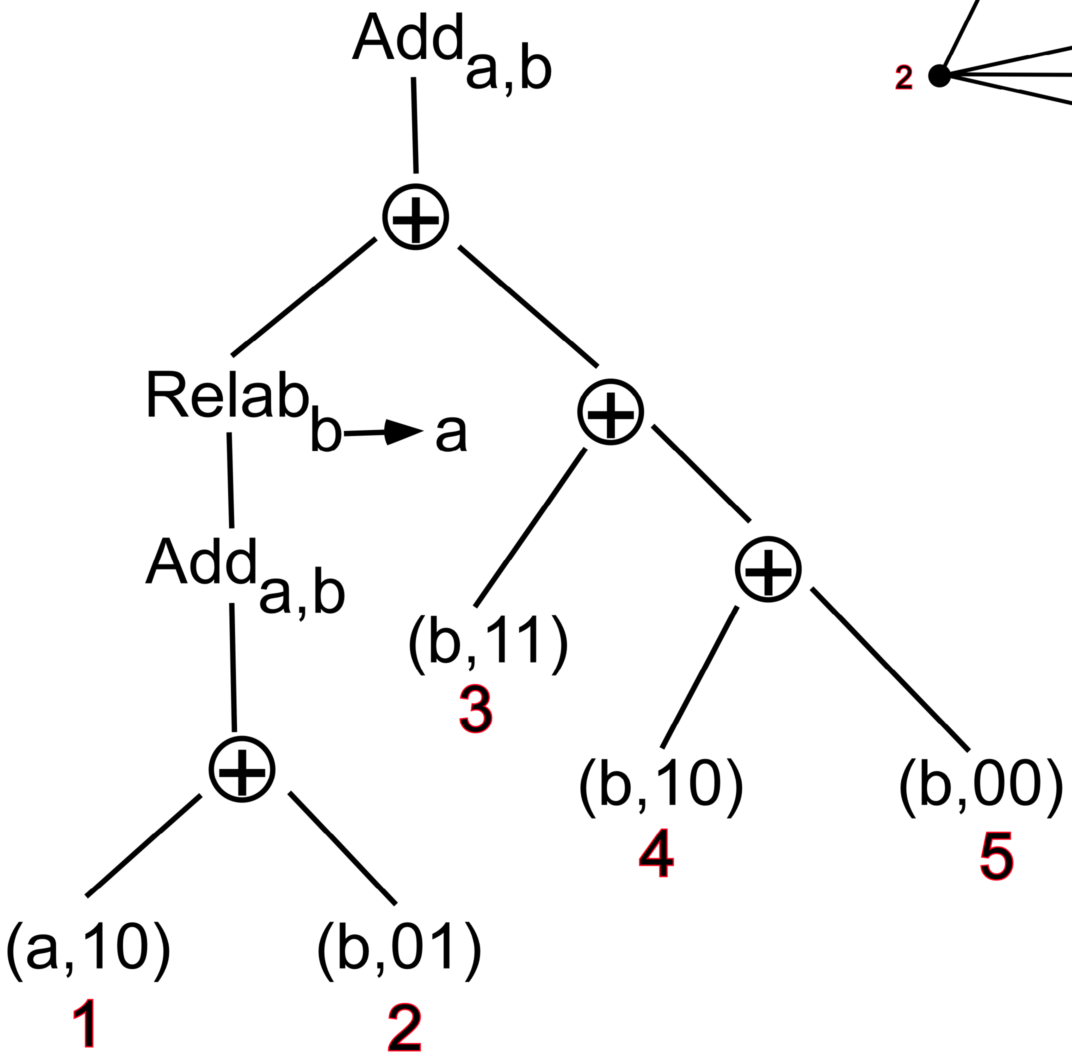


Term *t*

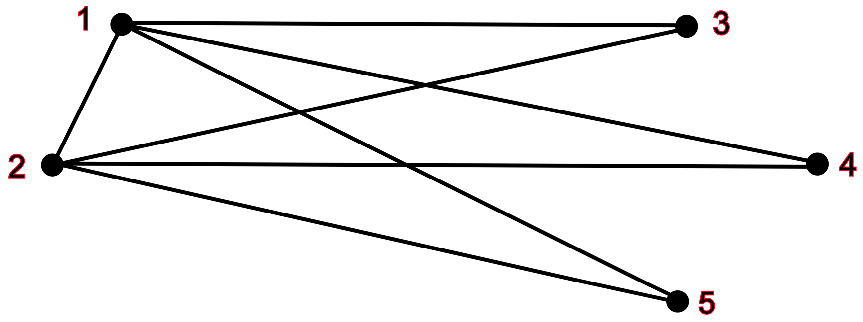


Graph G(*t*)

Example (continued)



Term $t * (V_1,V_2)$



$V_1 = \{1,3,4\}$, $V_2 = \{2,3\}$

By an induction on φ, we construct, for each $\varphi(\underline{X})$, $\underline{X}=(X_1,\ldots,X_p)$, a fly-automaton $A(\varphi(\underline{X}))$ that recognizes:

$$L(\varphi(\underline{X})) := \{ t * (V_1,\ldots,V_p) \in \mathbf{T}(F^{(p)}) \ / \ (G(t), V_1,\ldots,V_p) \models \varphi \}$$

Quantifications: Formulas are written without $\forall$

$$L(\exists X_{p+1} . \varphi(X_1, ..., X_{p+1})) = pr_{p+1}(L(\varphi(X_1, ..., X_{p+1})))$$

$$A(\exists X_{p+1} . \varphi(X_1, ..., X_{p+1})) = pr_{p+1}(A(\varphi(X_1, ..., X_{p+1})))$$

where pr_{p+1} is the *projection* that eliminates the last Boolean;

→ a *non-deterministic* FA denoted by $pr_{p+1}(A(\varphi(X_1, ..., X_{p+1})))$, to be run deterministically.

Remark: If a graph is denoted by a clique-width term *t*, then each of its vertices is represented in *t* at a single position (an occurrence of a nullary symbol).

If the operation // is also used (G // H is obtained from disjoint G and H by fusing some vertices of G to some vertices of H, in a precise way fixed by labels), then a vertex of G//H is represented by several positions of the term.

The automaton that checks a property $\varphi(X_1, ..., X_p)$ of G denoted by a term *t* must also check that the Booleans that specify $(X_1, ..., X_p)$ agree on all positions of *t* that specify a same vertex of G.

We have no such difficulty if we use disjoint union instead of //. Hence, for representing tree-decompositions, clique-width terms may be more convenient.

Computations using fly-automata (by Irène Durand)

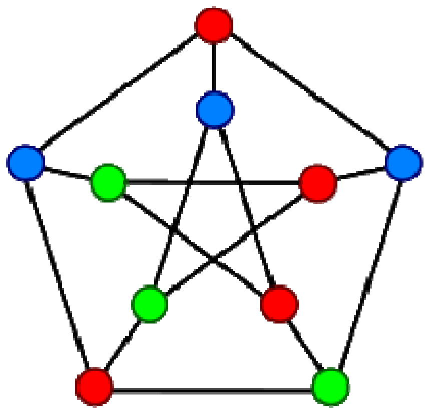
Number of 3-colorings of the 6 x 525 rectangular grid (of clique-width 8) in 10 minutes.

4-acyclic-colorability of the Petersen graph (clique-width 5) in 1.5 minutes.

(3-colorable but not acyclically;

red and **green** vertices

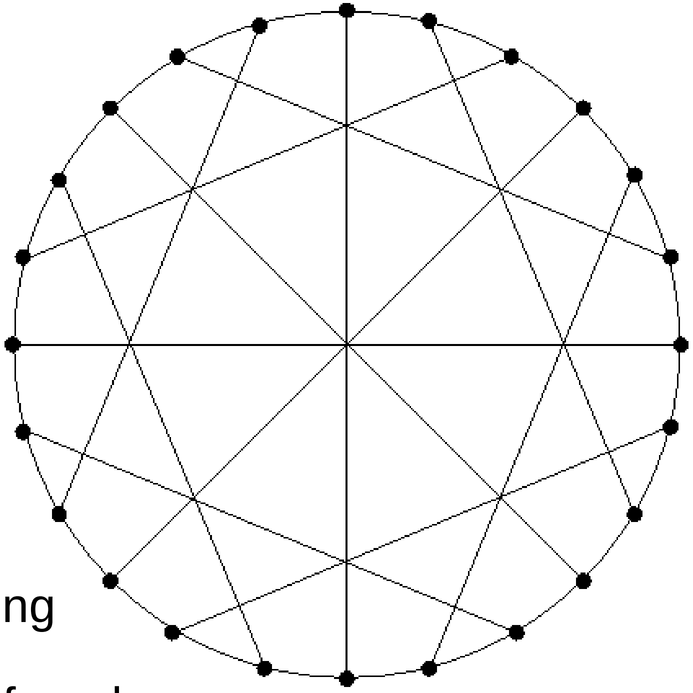
induce a cycle).



The McGee graph

is defined by a clique-width term

of size 99 and depth 76.



This graph is 3-acyclically colorable.

Checked in 40 minutes.

Even in 2 seconds by enumerating the accepting

runs, and stopping as soon as a success is found.

Application to MSO_2 properties of graphs of bounded tree-width *via* incidence graphs.

1) *Recall :* From of a tree-decomposition of G of width k, we construct a term t for Inc(G) of "small" clique-width k+3 (or 2k+4).

2) *Recall :* We translate an MSO_2 formula φ for G into an MSO formula θ for Inc(G).

3) The corresponding automaton **A**(θ) takes term t as input. But an atomic formula edg(X,Y) of φ is translated into $\exists$U. inc(X,U) $\wedge$ inc(U,Y) in θ which adds one level of quantification.

Fact : The automaton **A**(θ) remains manageable.

For certain graph properties P, for example "connectedness", "contains a directed cycle", "outdegree < p", we have :

$$P(G) \Leftrightarrow P(Inc(G)).$$

The automaton for graphs G defined by clique-width terms can be used "directly" for the clique-width terms that define the graphs Inc(G).

Summary : Checking properties of G of tree-width < k

MSO property	MSO_2 property
cwd term for G of width O(k) or $O(k^q)$ in "good cases" and exponential in bad ones	cwd term for Inc(G) of width O(k); more complicated automaton in some cases, because of edg(X,Y)

General conclusion

1) By uniform constructions, we get dynamic programming algorithms based on fly-automata, that can be quickly constructed from logical descriptions → *flexibility*.
2) It is hard to obtain upper-bounds to time computations. We do not get better algorithms than the specific ones that have been developed.

3) Even for graphs given by tree-decompositions, clique-width terms are appropriate because of two facts:

   (a) Fly-automata are simpler to construct and

   (b) it is practically possible to translate tree-decompositions of "certain" sparse graphs into clique-width terms.

4) Fly-automata are implemented. Tests have been made for colorability and connectedness problems.