Graph algorithms based on *fly-automata*:

logical descriptions and usable constructions

Bruno Courcelle

(joint work with Irène Durand)

Bordeaux University, LaBRI (CNRS laboratory)

Topics

Fixed-parameter tractable (FPT) algorithms based on graph decompositions + logic + *infinite* automata on terms called fly-automata.

Graph decompositions = tree structuring of graph in terms of "small" graphs and composition operations

Graph structure theory :

tree-decomposition for the Graph Minor Theorem,

modular decomposition for comparability graphs,

ad hoc decompositions for the Perfect Graph Theorem.

Algorithmic meta-theorems give FPT algorithms for parameters *tree-width* and *clique-width* based on graph decompositions; properties to check are expressed in *monadic second-order logic (MSO)*. (Definitions will be given soon).

Theorem : For each k, every MSO graph property P can be checked in (FPT) time $O(f(k).n)$ where n = number of vertices, k = tree-width or clique-width of the input graph, given by a relevant decomposition. This decomposition is formalized by an algebraic term over operations that build graphs (generalizing concatenation of words).

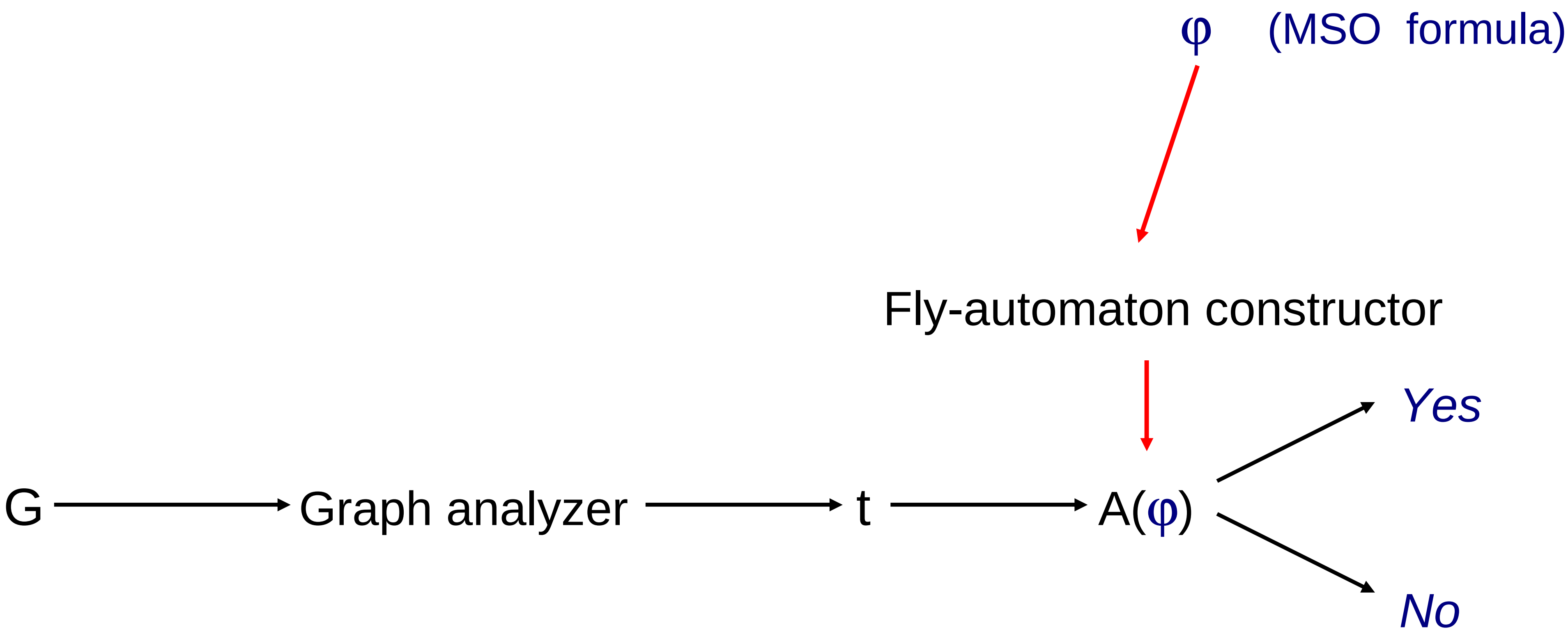
Method : From k and φ expressing P, one builds a finite automaton $A(\varphi,k)$ to recognize the terms that represent decompositions of width at most k and define graphs satisfying P.

Difficulty : The *finite* automaton A(φ,k) is much too large 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 transitions necessary for an input term are computed "on the fly". Sets of states can be infinite and fly-automata can compute values, *e.g.*, the number of p-colorings or of acyclic p-colorings of a graph. This is a theoretical view of dynamic programming.

The MSO meta-theorem through *fly-automata*



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

Computations using fly-automata (by Irène Durand)

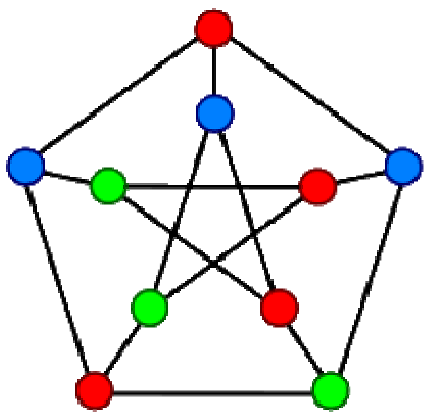
Number of 3-colorings of the 6 x 525 rectangular grid (of clique-width 8) in 10 minutes.

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

(3-colorable but not acyclically;

red and **green** vertices

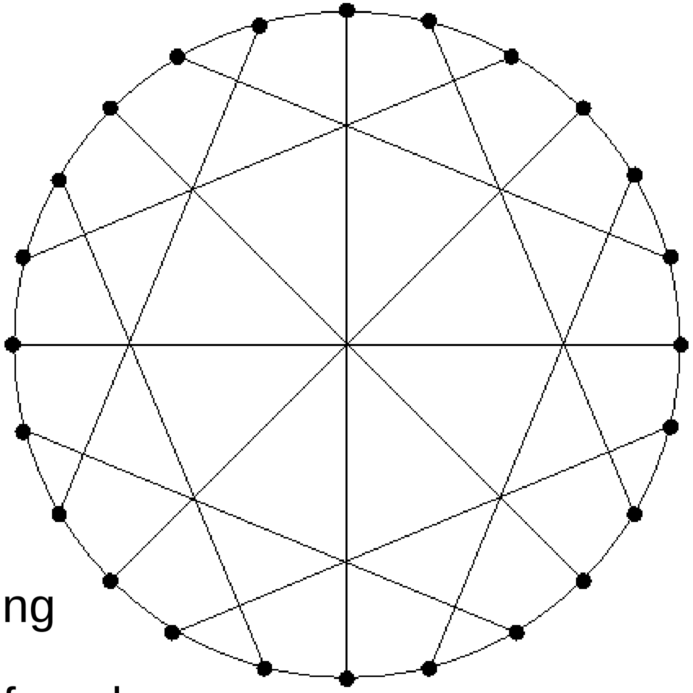
induce a cycle).



The McGee graph

is defined by a 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.

Definition 1 : Monadic Second-Order Logic

First-order logic extended with (quantified) variables denoting subsets of the domains.

A graph G is given by the logical structure

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

Property P is MSO expressible : $P(G) \Leftrightarrow G \models \varphi$

MSO expressible properties : transitive closure, properties of paths, connectedness, planarity (via Kuratowski), p-colorability.

Examples : G is 3-colorable :

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

G 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 $K_{3,3}$).

Edge quantifications (MSO_2 graph properties)

If $G = (V_G, edg_G(.,.))$, its *incidence graph* is defined as

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

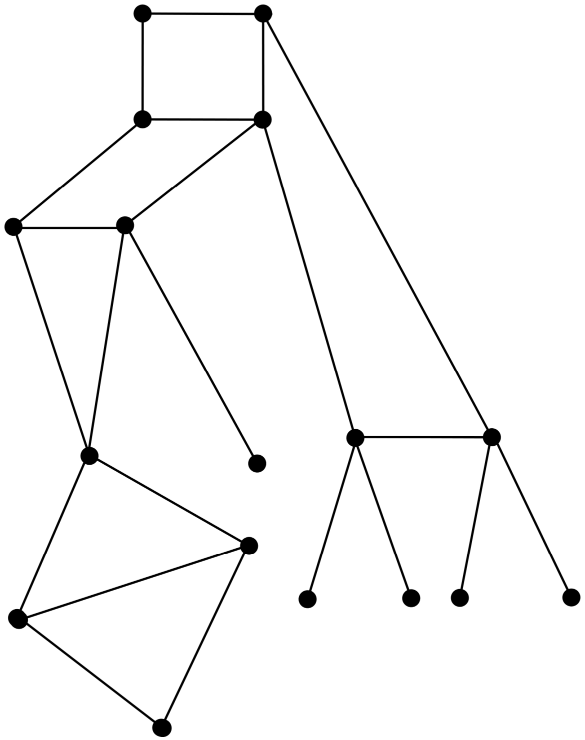
$inc_G(u,e) \Leftrightarrow$ u is the *tail* of edge e,

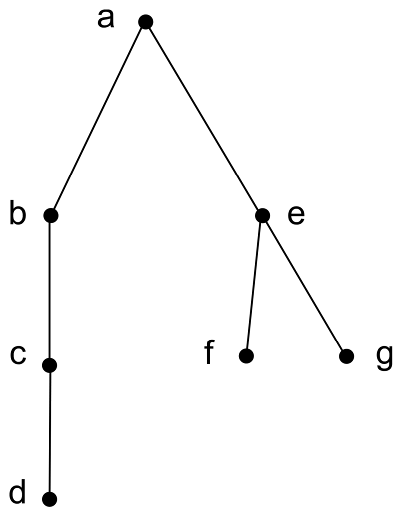
$inc_G(e,u) \Leftrightarrow$ u is the *head* of edge e. (G is directed).

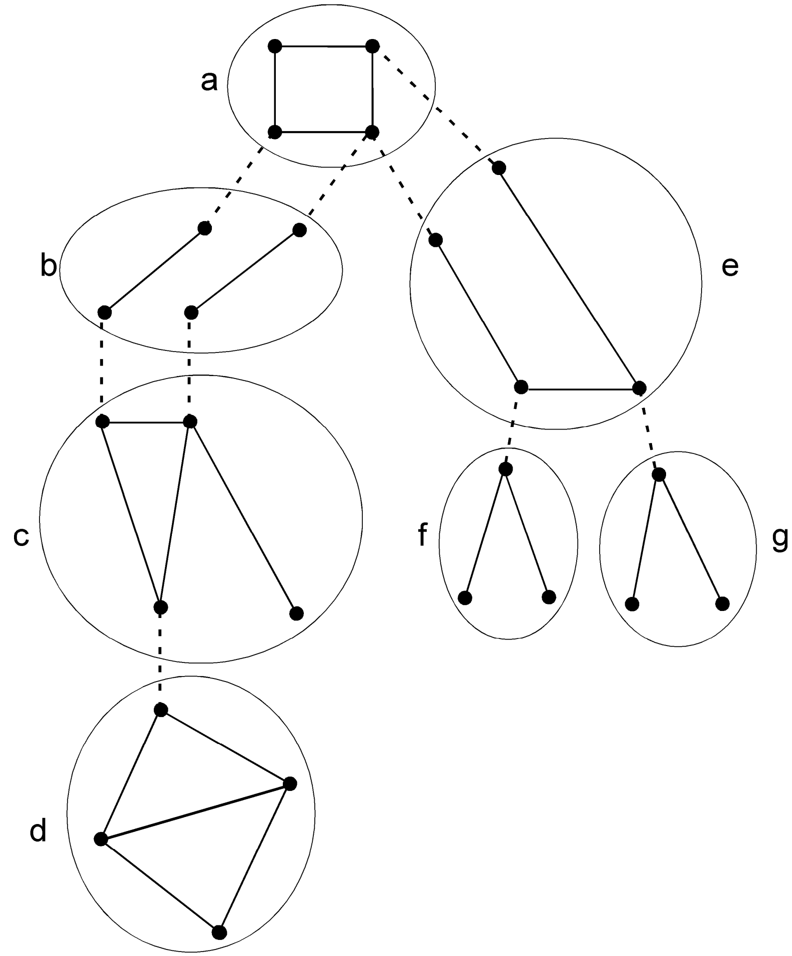
MSO formulas over Inc(G) can use quantifications on edges and express more properties than those over G. MSO_2 graph properties of G are expressed by MSO formulas over Inc(G).

That G is isomorphic to some $K_{p,p}$ is MSO_2 expressible but not MSO expressible.

Definition 2 : Tree-decomposition, tree-width (denoted by twd(G)).







Graph G a decomposition of G of width 3 (= 4-1)

Definition 3 : Clique-width (denoted by cwd(G)).

Defined from graph operations. Graphs are simple, directed or not, vertices are labelled by *a,b,c, ...* . A vertex labelled by *a* is an *a-vertex*.

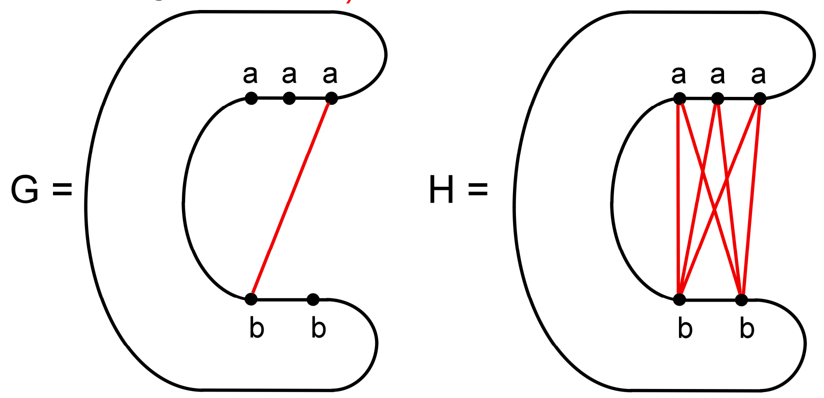
One binary operation: disjoint union : ⊕

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.

Directed edges are defined similarly.



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

Vertex relabellings :

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

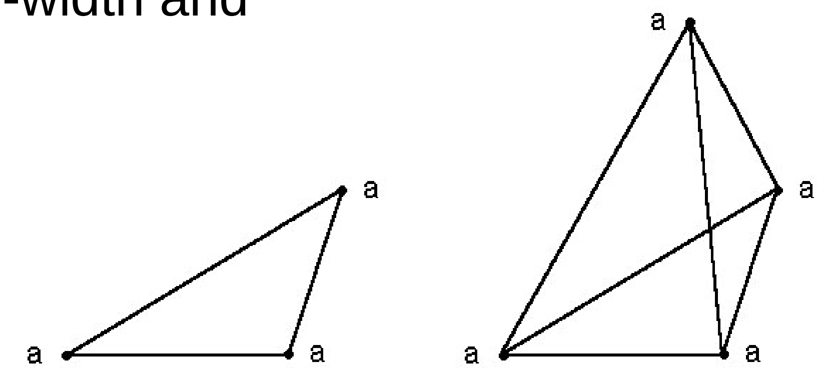
Basic graphs : **a** , a vertex labelled by *a*.

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

Example : Cliques have unbounded tree-width and clique-width 2.

$K_{\mathbf{n}}$ is defined by $t_{\mathbf{n}}$ where $t_{\mathbf{n+1}}$ =

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



Meta-theorems : FPT time f(wd(G)).n

(1) MSO properties of graphs of bounded cwd,

(2) MSO_2 properties of graphs of bounded twd.

Notes: - MSO expressible $\Rightarrow$ MSO_2 expressible and

bounded twd $\Rightarrow$ bounded cwd.

(2) reduces to (1) because MSO_2 on G = MSO on Inc(G)

and cwd(Inc(G)) = O(twd(Inc(G))) = O(twd(G))

avoiding the exponential jump $cwd(G) = 2^{O(twd(G))}$

Next : only MSO formulas and clique-width

Definition 4 : Fly-automaton (FA)

$\mathcal{A}$ = < F, Q, δ, Out >

F : finite or countable (effective) 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* (a set of finite words), computable.

δ : computable (bottom-up) transition function

Nondeterministic case : δ is *finitely multi-valued*, which implies that determinization works.

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

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

Computation time is f(k).n, n = size of term, k = number of labels in t.

Note : Graphs are handled through terms or labelled trees (cf. tree-decompositions) describing them.

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

Proof sketches : To check a property P(G), for G = G(t), t $\in$ T(F).

For each *labelled* graph G, we define a piece of information q(G) that encodes properties of G and values attached to G, so that:

(i) inductive behaviour of q : for f $\in$ F and graphs G,H:

$$q(f(G,H)) = f^{q}(q(G), q(H))$$

for some computable function f^{q} .

(ii) P(G) can be decided from q(G).

Then q(G(t/u)) is computed bottom-up in a term t, for each node u. This information is relative to the graph G(t/u) (a subgraph of G) defined by the subterm t/u of t issued from u.

q(G(t/u)) is a state of a *finite or infinite deterministic bottom-up automaton*.

These automata formalize some form of dynamic programming.

Constructions: "Direct" for a well-understood graph property or "automatic" from an MSO formula.

Computation time of a fly-automaton

F : all graph 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 :

$(k + n)^c$ → 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 (XP : see Downey and Fellows).

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

Direct construction 1 : Connectedness.

The state at node u is the *set of types (sets of labels)* of the connected components of the graph G(t/u). For k labels (k = bound on clique-width), the set of states has size $\leq 2 ^\wedge (2 ^\wedge k)$.

Proved lower bound : $2 ^\wedge (2 ^\wedge k/2)$.

→ Impossible to "compile" the automaton (*i.e.,* to list the transitions) .

Example of a state : q = { {a}, {a,b}, {b,c,d}, {b,d,f } }, (a,b,c,d,f : labels).

Some transitions :

$Add_{a,c}$: q ⟶ { {a,b,c,d}, {b,d,f } },

$Relab_{a \to b}$: q ⟶ { {b}, {b,c,d}, {b,d,f } }

Transitions for ⊕ : union of sets of types.

Note : Also state (p,p) if G(t/u) has ≥ 2 connected components, all of type p.

In a *fly-automaton,* states and transitions are *computed* and not tabulated. We can allow fly-automata with *infinitely* many states and with *outputs* : numbers, finite sets of tuples of numbers, etc.

Example continued : For computing the number of connected components, we use states such as :

q = { ({a}, 4), ({a,b}, 2), ({b,c,d},2), ({b,d,f },3) },

where 4, 2, 2, 3 are the numbers of connected components of respective types {a}, {a,b}, {b,c,d}, {b,d,f }.

Direct construction 2 : Regularity (not MSO)

A state is a tuple of counters that indicates, for each label *a*:

the number of *a*-vertices and

the common degree of all *a*-vertices.

The state is *Error* if two *a*-vertices have different degrees: the edge addition operations will add the same numbers of edges (some technical details are omitted) to these vertices, hence the considered graph cannot be regular.

Inductive construction based on an MSO formula

Atomic formulas : direct constructions (examples to come)

$\neg$ P (negation) : FA are run deterministically, it suffices to exchange accepting/non-accepting states.

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

How to handle free variables and $\exists$ X,Y.P(X,Y) ?

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

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

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

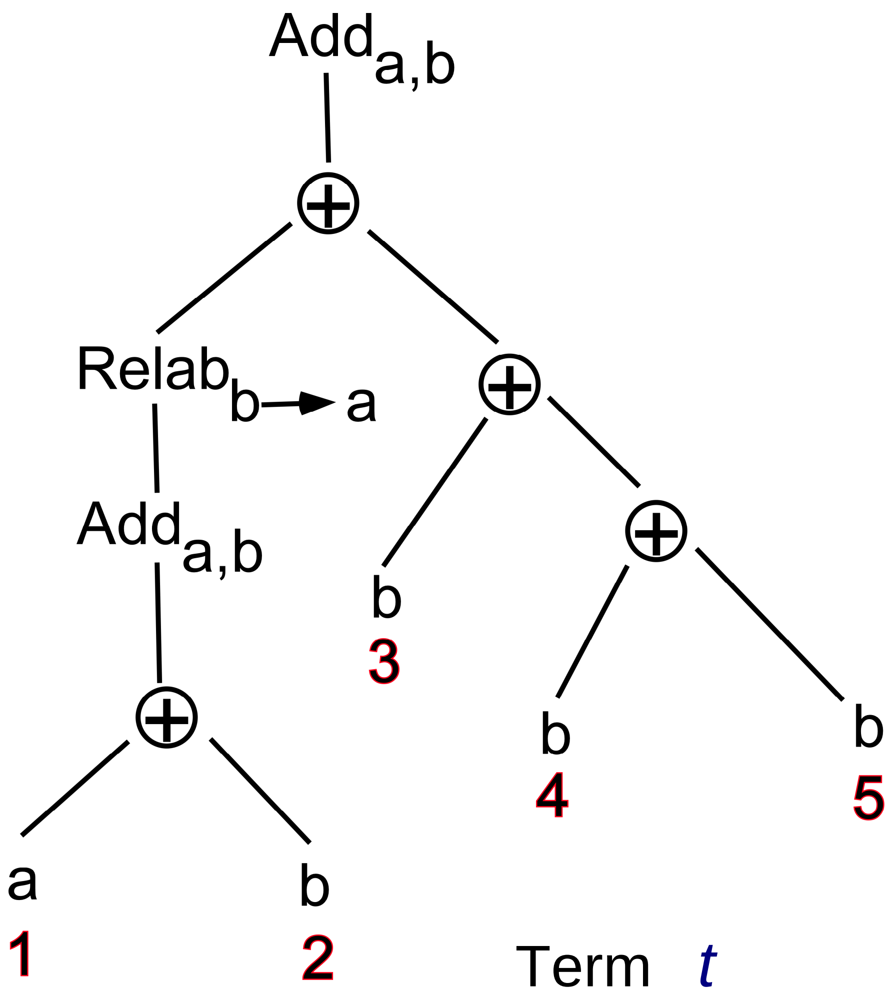
2) a term *s* in $\mathbf{T}(F^{(n)})$ encodes a term *t* in $\mathbf{T}(F)$ and an assignment of sets $V_1,\ldots,V_n$ to the set variables $X_1,\ldots,X_n$:

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

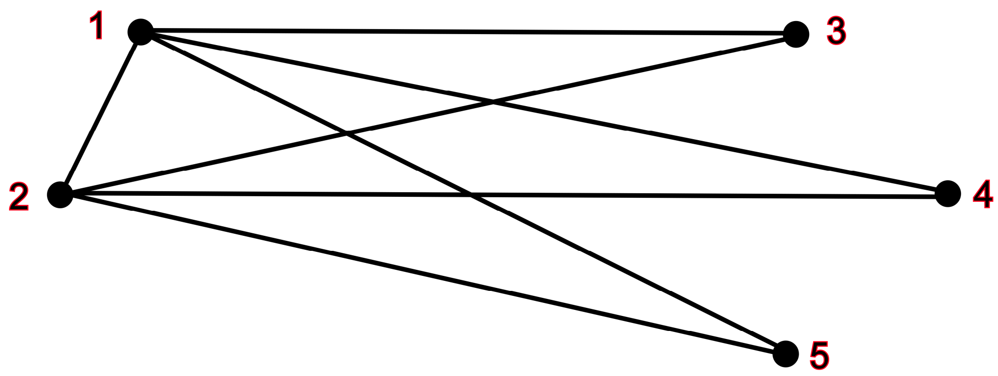
$w_i = 1$ if and only if $u \in V_i$.

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

Example

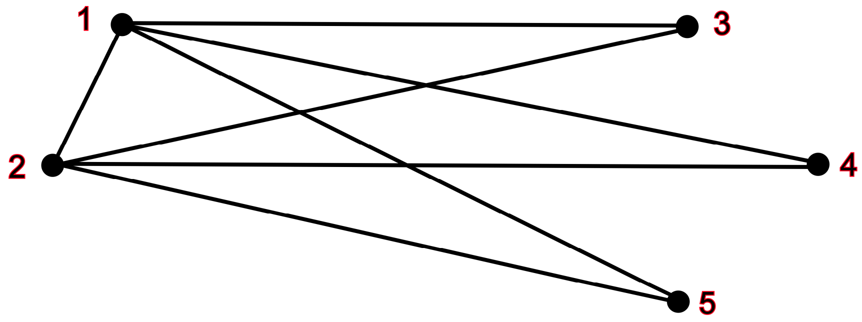


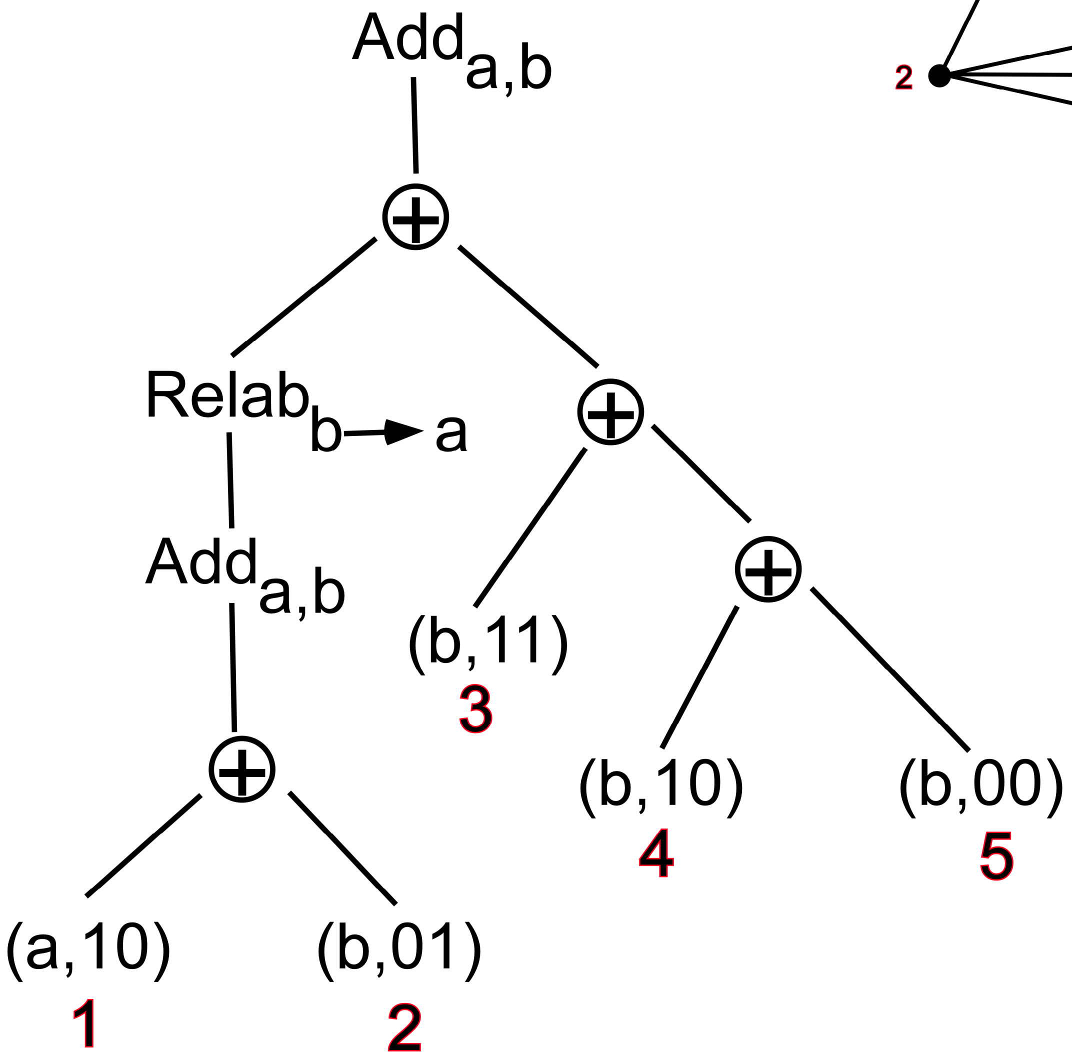
Term *t*



Graph G(*t*)

Example (continued)





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

Term $t * (V_1,V_2)$

By an induction on φ, we construct for each $\varphi(X_1,\ldots,X_n)$ an FA $A(\varphi(X_1,\ldots,X_n))$ that recognizes:

$$L(\varphi(X_1,\ldots,X_n)) := \{ t * (V_1,\ldots,V_n) \in \mathbf{T}(F^{(n)}) \;/\; (G(t), V_1,\ldots,V_n) \models \varphi \}$$

Quantifications: Formulas are written without $\forall$

$$L(\exists X_{n+1} . \varphi(X_1, \ldots, X_{n+1})) = pr(L(\varphi(X_1, \ldots, X_{n+1})))$$

$$A(\exists X_{n+1} . \varphi(X_1, \ldots, X_{n+1})) = pr(A(\varphi(X_1, \ldots, X_{n+1})))$$

where pr is the *projection* that eliminates the last Boolean;

→ a *non-deterministic* automaton.

Atomic and basic formulas :

$X_1 \subseteq X_2$, $X_1 = \emptyset$, Single(X_1),

$\text{Card}_{\mathbf{p,q}}(X_1)$: cardinality of X_1 is = p mod. q,

$\text{Card}_{\mathbf{<q}}(X_1)$: cardinality of X_1 is < q.

→ Easy constructions of automata with few states :

respectively 2, 2, 3, q, q+1 states.

Example : for $X_1 \subseteq X_2$, the term must have no constant (**a**, 10).

Atomic formula : edg(X_1,X_2) (means : $X_1 = \{ x \} \wedge X_2 = \{ y \} \wedge$ edg(x, y))

Vertex labels belong to a countable set C of labels.

States : 0, Ok, a(1), a(2), ab, Error, for a,b $\in$ C , a $\neq$ b

Meaning of states (at node u of t ; its subterm t/u defines G(t/u) $\subseteq$ G(t)).

0 : $X_1 = \emptyset$, $X_2 = \emptyset$

Ok *Accepting state* : $X_1 = \{v\}$, $X_2 = \{w\}$, edg(v,w) in G(t/u)

a(1) : $X_1 = \{v\}$, $X_2 = \emptyset$, v has label a in G(t/u)

a(2) : $X_1 = \emptyset$, $X_2 = \{w\}$, w has label a in G(t/u)

ab : $X_1 = \{v\}$, $X_2 = \{w\}$, v has label a, w has label b (hence v $\neq$ w) and $\neg$edg(v,w) in G(t/u)

Error : all other cases

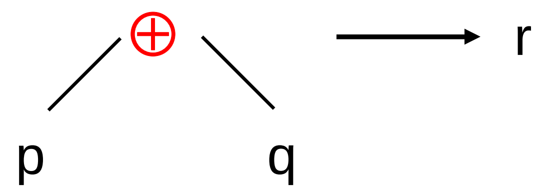
Transition rules

For the constants based on **a** :

(**a**,00) → 0 ; (**a**,10) → a(1) ; (**a**,01) → a(2) ; (**a**,11) → Error

For the binary operation ⊕:

(p,q,r are states)



If p = 0 then r := q

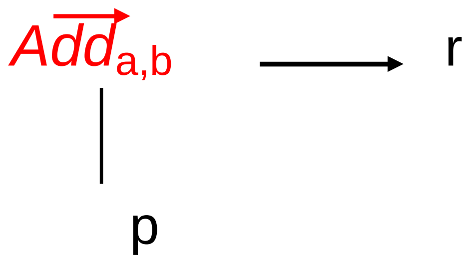
If q = 0 then r := p

If p = a(1), q = b(2) and a ≠ b then r := ab

If p = b(2), q = a(1) and a ≠ b then r := ab

Otherwise r := Error

For unary operations $\overrightarrow{Add}_{a,b}$



If p = ab then r := Ok else r := p

For unary operations $Relab_{a \rightarrow b}$

If p = a(i) where i = 1 or 2 then r := b(i)

If p = ac where c ≠ a and c ≠ b then r := bc

If p = ca where c ≠ a and c ≠ b then r := cb

If p = Error, 0, Ok, c(i), cd, dc where c ≠ a then r := p

Theorem : For each sentence φ, the infinite fly-automaton A(φ) accepts in time f(φ, k). | *t* | the terms *t* in **T**(F) such that G(*t*) $\models \varphi$ where k is the number of labels occurring in *t*.

It gives a *fixed-parameter linear* model-checking algorithm for input *t*, and a *fixed-parameter cubic* one if the graph has to be *parsed*.

The parsing problem is another difficulty. See conclusion.

Other constructions (for computing values)

The number of satisfying assignments : # $\underline{X}$.P($\underline{X}$).

The spectrum Sp$\underline{X}$.P($\underline{X}$) : the set of tuples of cardinalities of the components of the tuples $\underline{X}$ that satisfy P($\underline{X}$).

The multispectrum MSp$\underline{X}$.P($\underline{X}$) is the corresponding multiset of tuples of Sp$\underline{X}$.P($\underline{X}$). For $\underline{X}$ = X (one component):

the set of pairs (*m* ,i) such that i > 0 is

the number of sets X of cardinality *m* that satisfy P(X).

For a p-tuple $\underline{X}$, a multispectrum is a function $[0,n]^p \rightarrow [0,2^{p.n}]$;

it can be encoded in size $O(n^p.\log(2^{p.n})) = O(n^{p+1})$.

The minimum cardinality of X satisfying P(X).

Example: Number of accepting runs of a nondeterministic automat.

Let A = < F, Q, δ, Acc > be finite, nondeterministic.

Then #A := < F, [Q → **N**], $\delta^{\#}$, Out >

[Q → **N**] = the set of total functions : Q → **N**

$\delta^{\#}$ is easy to define such that the state reached at position

u in the input term is the function σ such that σ(q) is

the number of runs reaching q at u.

Out(σ) is the sum of σ(q) for q in Acc.

#A is a fly-automaton obtained by a generic construction that extends to the case of an infinite fly-automaton A.

Some non-MSO examples

(1) *Equitable p-coloring* :

$\exists X_1,\ldots,X_p$ (Partition($X_1,\ldots,X_p$) $\wedge$ Stable[X_1] $\wedge \ldots \wedge$ Stable[X_p]

$\wedge\ |X_1| = \ldots = |X_{i-1}| \geq |X_i| = \ldots = |X_p| \geq |X_1| - 1$).

It is FPT (for fixed p).

(2) *Partition into 2 regular graphs* :

$\exists X$ (Reg[X] $\wedge$ Reg[X^c])

Reg[X] means that the subgraph induced on X is regular;

X^c is the complement of X. It is XP.

(3) Minimizing the use of a particular color: this gives a “distance to *p*-colorability” for a graph that *p+1*-colorable but not p-colorable.

In general, we can handle properties and functions of the forms

$$\exists \underline{X}.P(\underline{X}),\ MSp\ \underline{X}.P(\underline{X}),\ Sp\ \underline{X}.P(\underline{X}),\ \#\ \underline{X}.P(\underline{X})$$

where $P(\underline{X})$ is a Boolean combination of properties for which we have constructed FA (Reg, NoCycle, Stable, etc…).

The system AUTOGRAPH (by I. Durand)

Fly-automata for basic graph properties :

Clique, Stable (no edge), Link(X,Y), NoCycle,

Connectedness, *Regularity*, Partition(X, Y, Z), etc…

and functions :

#Link(X,Y) (number of edges between X and Y),

Maximum degree.

Procedures for combining fly-automata (combinations of descriptions)

product : for $P \wedge Q$, $P \vee Q$, $g(\alpha_1 , \ldots, \alpha_p)$

$A \rightarrow A/X$: for $P \rightarrow P[X]$, (P in induced subgraph on X) and

$A \rightarrow A/(X \cap Y) \cup (Y \cap Z)^c$ for relativization to *set terms.*

image automaton: $A \rightarrow h(A)$: in the transitions of A, each function symbol f is replaced by h(f) ; h(A) is *nondeterministic*: for $P(\underline{X}) \rightarrow \exists \underline{X}.P(\underline{X})$

Procedures to build automata that compute functions:

#X.P(X) : the number of tuples X that satisfy P(X) in the input term (hence, in the associated graph).

SpX.P(X) : the set of tuples of cardinalities of the components of the X that satisfy P(X).

MSpX.P(X) : the corresponding multiset.

SetValX.α(X) / P(X) : the set of values of α(X) for the tuples X that satisfy P(X).

For each case, a procedure transforms FA for P(X) and α(X) into FA that compute the associated functions. (These transformations do not depend on P(X) and α(X).)

Conclusion and future work

The parsing problem : 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 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.

With fly-automata, we get in most cases XP or FPT dynamic programming algorithms, that can be obtained independently.

Fly-automata, can be quickly constructed from logical descriptions → *flexibility*.

These constructions are implemented. Tests have been made mainly for colorability and connectedness problems.

Next step : *enumeration algorithms* based on fly-automata.

Thank you for suggesting interesting problems that could fit in this framework.

Work presented at workshop GROW in Santorini, Greece.