# Query evaluations by fly-automata

*Bruno Courcelle*

*Irène Durand*

Bordeaux University, LaBRI (CNRS laboratory)

Reference : B.C, I.D. : Computations by fly-automata beyond MSO logic, *Theor. Comput. Sci.* Vol. 619 (2016)

# Topics

Fixed-parameter tractable (FPT) graph algorithms for monadic second-order (MSO) problems based on infinite "fly"-automata (FA) running on clique-width terms denoting the input graphs.

Generic, algebraic constructions of FA : for example, the number of accepting runs of a nondeterministic FA

Meta-dynamic-programming: constructions from logical descriptions of problems.

Here : Numerical evaluations on MSO queries.

# Review of definitions and basic facts.

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

A graph G is given by the logical structure

$( V_G , edg_G(.,.) )$ = (vertices, adjacency relation)

Monadic second-order (MSO) formulas $\varphi$ can express

p-colorability (and variants), transitive closure, properties of paths, connectedness, planarity (via Kuratowski), *etc…*

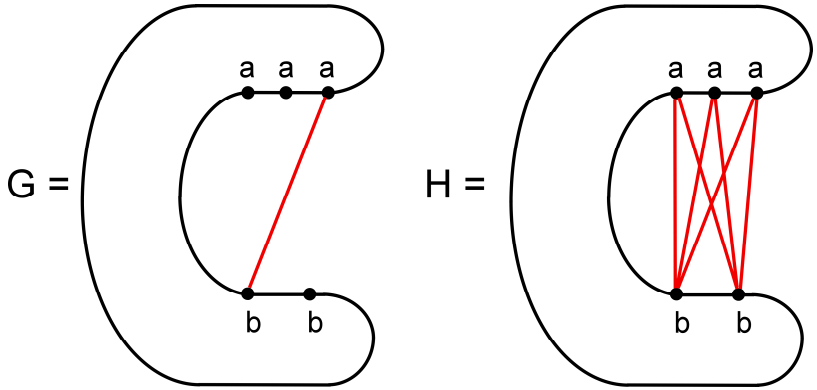
# Clique-width (denoted by cwd(G)).

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.

Theorem (B.C.): Each MSO property $\varphi$ can be checked in time f(k).n by a finite automaton A($\varphi$,k) taking as input a term denoting a graph of clique-width $\leq$ k having n vertices.

Difficulty : The *finite* automaton A(φ,k) is much too large as soon as

$k \geq 2$ : $2^{(2^{(\ldots 2^k)..)})}$ 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 of a graph.

# Fly-automaton (FA)

$A$ = < F, Q, δ, Out > to compute 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),

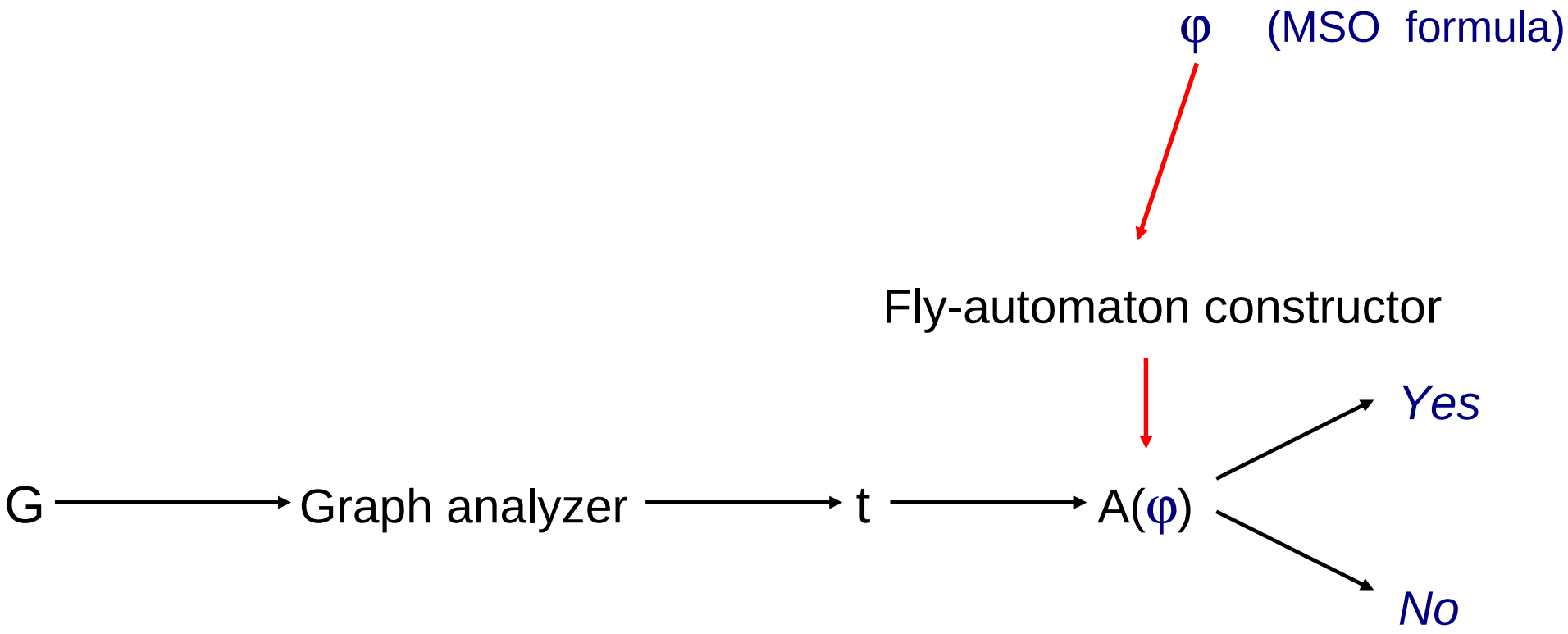
Out : Q → $D$ , computable ($D$ is effective, coded by finite words).

δ : computable (bottom-up) transition function

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

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

# The MSO meta-theorem through *fly-automata*




A(φ): *unique 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.

# Computation time of a fly-automaton

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.

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

# Computing graph evaluations

$P(\underline{X})$ is a property of tuples $\underline{X}$ of sets of vertices (usually MSO expressible).

$\exists\, \underline{X}.P(\underline{X})$ : the basic, "Boolean evaluation".

Sat $\underline{X}.P(\underline{X})$ : the set of all $\underline{X}$ that satisfy $P(\underline{X})$: the "maximal evaluation".

$\#\, \underline{X}.P(\underline{X})$ : number of satisfying tuples $\underline{X}$.

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

MSp $\underline{X}.P(\underline{X})$ : multispectrum = the corresponding multiset.

(for $\underline{X} = X$ : the set of pairs $(m, i)$ such that $i > 0$ is the number of sets $X$ of cardinality $m$ that satisfy $P(X)$ ).

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

SetVal-$\alpha$(X)/P(X) : the set of values of $\alpha$(X)

for the tuples X that satisfy P(X).

A general presentation for all these cases and others :

f(G) := f(SatX.P(X)) where

f is a computable function: $P(P(V_G)^p) \rightarrow D$

*Theorem* (Flum and Grohe) : One can compute Sat X.P(X) in time f(k)(n+ size of the result) where cwd(G) $\leq$ k and n is the size of the term.

Question : How to "shortcut" the computation of Sat X.P(X) ?

## *Review* : inductive construction for $\exists \underline{X}.P(\underline{X})$ based on an MSO formula $\varphi(\underline{X})$ that defines $P(\underline{X})$

Atomic formulas in $\varphi(\underline{X})$ : direct constructions

$\neg$ P (negation) : FA are run deterministically (see below), 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 and $\exists \underline{X}.P(\underline{X})$ ?

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

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

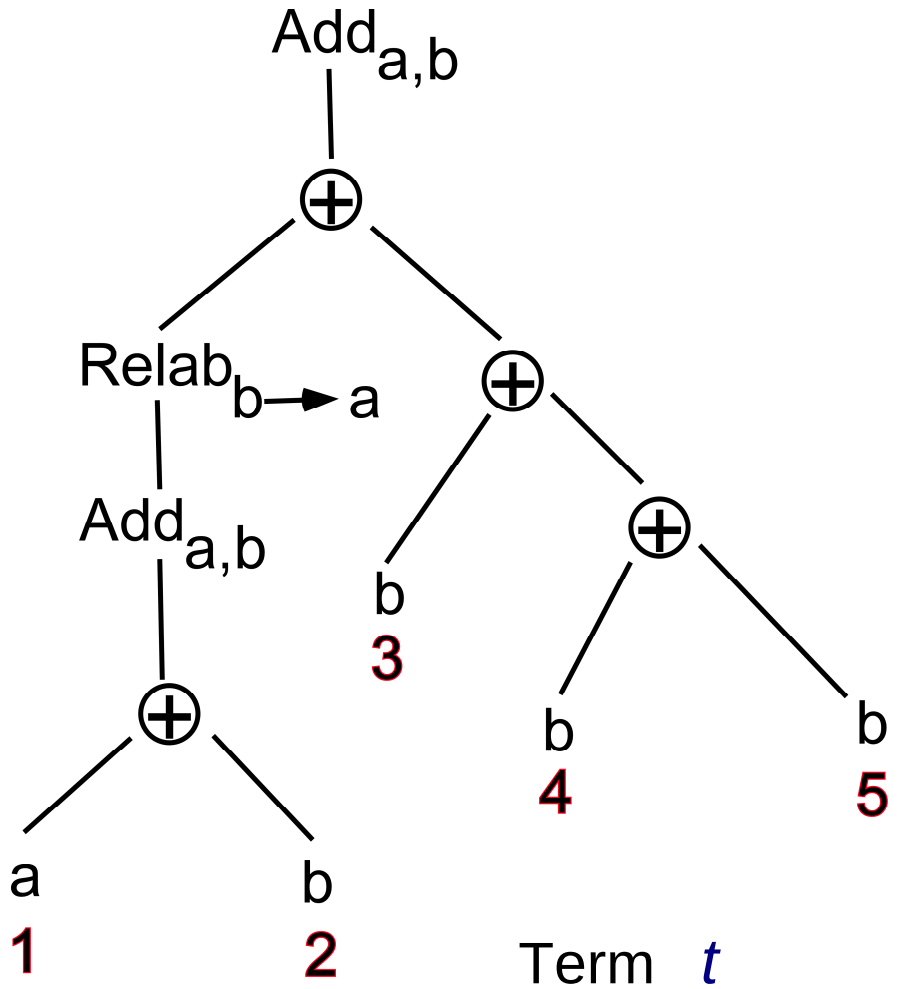
2) a term $s$ in $\mathbf{T}(F^{(p)})$ encodes a term $t$ in $\mathbf{T}(F)$ and an assignment of sets $V_1,\ldots,V_p$ to the set variables $X_1,\ldots,X_p$ :

if $u$ is an occurrence of ($\mathbf{a}$, $(w_1,..,w_p)$), then

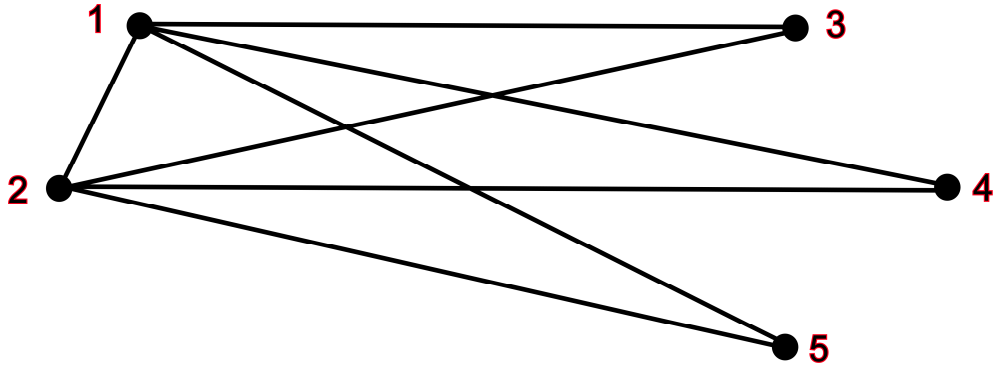
$w_i = 1$ if and only if $u \in V_i$ .

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

## *Example*

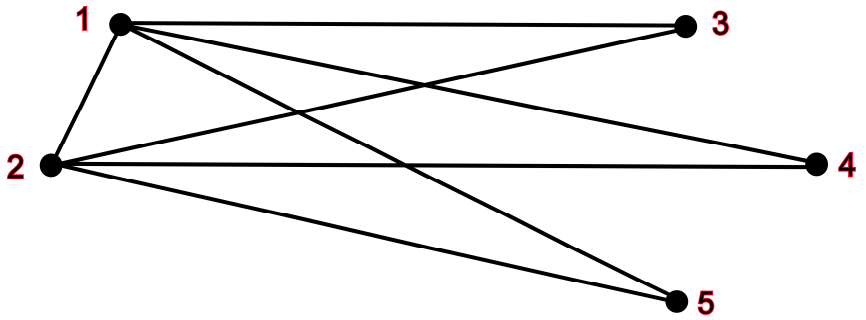


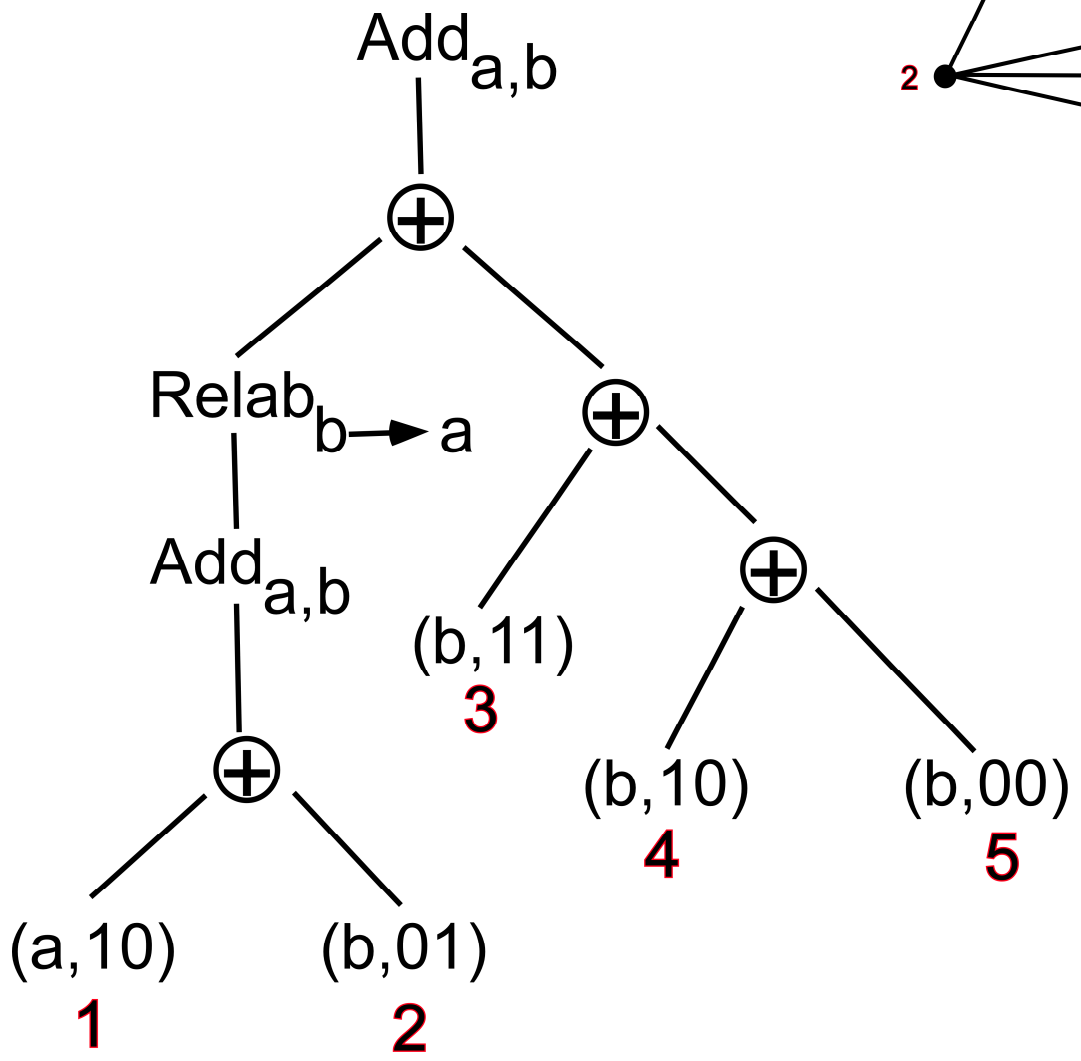

Term *t*




Graph G(*t*)

## *Example (continued)*




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

Term $t * (V_1, V_2)$

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

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

*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.

# Computation of Sat $\underline{X}.P(\underline{X})$

We start from a deterministic FA $A(\varphi(\underline{X}))$ over $F^{(p)}$ that computes $L(\varphi(\underline{X}))$. At position u in a term $t*\underline{X}$, it reaches state $q(u, \underline{X} / u)$.

We make it into a deterministic FA B over $F^{(p)}$ that reaches state $(q(u, \underline{X} / u) , \underline{X} / u )$ at each u. At the *root*, we get:

Sat $\underline{X}.P(\underline{X})$ := the set of all $\underline{X}$ such that $q(root, \underline{X} )$ is accepting.

The determinized run of $C := pr_{All}(B)$ computes at each u the set of all pairs $(q(u, \underline{X} / u) , \underline{X} / u)$ for all $\underline{X} / u$.

We can “factorize” this set as { (q, Sat(u,q)) / Sat(u,q) is the set of tuples $\underline{X} / u$ such that $q = q(u, \underline{X} / u)$ and q is not *Error* }.

Optimizations : How to avoid intermediate computations that do not contribute to the final result.

*Recall* : One can compute Sat $\underline{X}$.P($\underline{X}$) in time f(k)(n+ size of the result) where cwd(G) $\leq$ k and n is the size of the term.

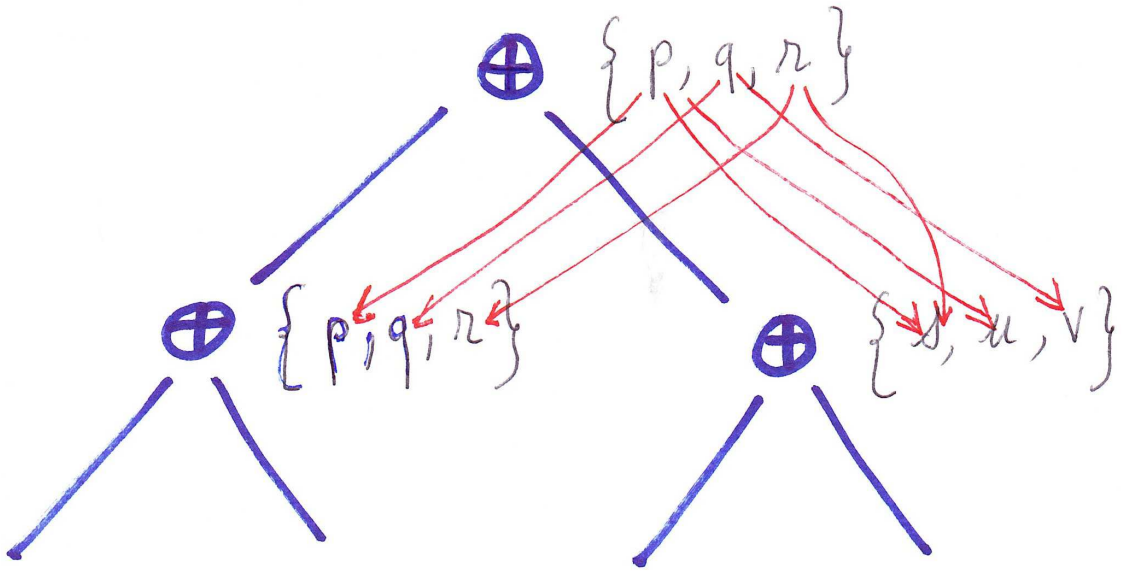
The bottom-up computation of the set of (q, Sat(u,q)) must "know" whether q belongs to any accepting run on the input term.

*Method :* A 3-pass algorithm

1 : determinized bottom-up run keeping pointers showing how states are obtained from others,

2 : top-down run starting from the accepting states at *root* and marking the *useful states*: set Q(u) at u.

3 : bottom-up computation of the pairs (q, Sat(u,q)), only for the useful states q in Q(u).




$\oplus[p, s] \rightarrow p$

$\oplus[p, u] \rightarrow p$

$\oplus[q, v] \rightarrow q$

$\oplus[r, s] \rightarrow r$

# Efficient evaluations

For computing f(Sat $\underline{X}$.P($\underline{X}$)) as opposed to Sat $\underline{X}$.P($\underline{X}$), one can (in good cases) maintain "light" information by replacing in the pairs (q, Sat(u,q)), each tuple $\underline{X}$ /u by some value h($\underline{X}$ /u ).

Good case : f : $P(P(V_G)^p) \rightarrow D$ is goh where h is a semi-ring homomorphism : $P(P(V_G)^p) \rightarrow R$ and g is computable : $R \rightarrow D$.

The two needed operations on $P(P(V_G)^p)$ are disjoint union $\underline{\cup}$ , and:

$\underline{A} \underline{*} \underline{B}$ := the set of pairs $(A1 \cup B1, \ldots, Ap \cup Bp)$ such that :

$(A1, \ldots, Ap) \in \underline{A}$ and $(B1, \ldots, Bp) \in \underline{B}$ , where $Ai \cap Bj = \varnothing$.

Commutative semi-ring structure :

$\underline{\cup}$ and $\underline{*}$ are associative and commutative with respective units $\varnothing$ and $(\varnothing, \ldots, \varnothing)$. Also $\varnothing$ is a zero for $\underline{*}$ and :

$$\underline{A} \underline{*} (\underline{B1} \underline{\cup} \underline{B2}) = \underline{A} \underline{*} \underline{B1} \underline{\cup} \underline{A} \underline{*} \underline{B2} .$$

They are useful because at position u of $\oplus$ :

Sat(u,q) is the disjoint union of the sets

Sat(u1,q1) $*$ Sat(u2,q2) such that we have $\oplus$ [q1,q2] $\rightarrow$ q

At an occurrence of an *Add* or *Relab* operation, only $\cup$ is needed.

We need a semi-ring $R$ = <R, +, $*$ ,0, 1> and h homo : $P(P(V_G)^p) \rightarrow R$.

*Example* : For computing #X.P(X), Sat(u,q) is replaced by its cardinality, informally, the number of X /u that yield state q at u.

Here $R$ = <**N**, +, $*$ , 0, 1>.

| | R | h(A ) | a + b | a ⋆ b |
|---|---|---|---|---|
| Sp$\underline{X}$.P($\underline{X}$) | $P_f(\mathbf{N}^p)$ | Set of p-tuples of cardinalities | a ∪ b | Set of sums $\alpha+\beta$, $\alpha \in$ a, $\beta \in$ b |
| MSp$\underline{X}$.P($\underline{X}$) | Finite multisets over $\mathbf{N}^p$ | Multiset of p-tuples of cardinalities | Union of multisets | Multiset of sums $\alpha+\beta$, $\alpha \in$ a, $\beta \in$ b |

| | R | h(A ) | a + b | a ∗ b |
|---|---|---|---|---|
| MinCardX.P(X) | **N** $\cup\{\propto\}$ | Min. card. of set $\alpha \in A$ ; $\propto$ if $A = \emptyset$ | Min{a, b} | a + b |
| MinSetX.P(X) w.r.t. some partial order $\leq$ | Min($P(P(V_G))$) = antichains in $P(V_G)$ | Set of minimal sets in A | Min($a \cup b$) | Minimal sets in the set of $\alpha\cup\beta$, $\alpha \in$ a, $\beta \in$ b |

Partial orders $X \leq Y$ : $X \subseteq Y$ ; $|X| \leq |Y|$ ;

$|X| < |Y|$ or { $|X| = |Y|$ and $X \leq_{lex} Y$ }

The 3-pass algorithm is applicable to these computations.

*Example :* Checking that a graph has a unique 3-coloring.

1st method : expressing this in MSO : possible but cumbersome.

2nd method : computing the total number of 3-colorings: we want result 6 (assume the graph is not 2-colorable) : OK but lengthy.

3rd method : "optimized" counting that reports a Failure if a *useful* intermediate result more than 6 is found.

This is applicable to $\exists! \underline{X}.P(\underline{X})$ for every MSO property P.

# Digression : Data structures for Sat $\underline{X}$.P($\underline{X}$)

1. A p-dimensional Boolean matrix of size $2^{n.p}$
2. A list of p-tuples of sets, of global size $\leq$ n.p. $|$Sat $\underline{X}$.P($\underline{X}$)$|$
3. Factorized such list: for p = 2 , the set of tuples

   $(X, Y1, \ldots, Yn_X)$ , such that $P(X,Yi)$ , $n_X > 0$.

4. Term $T$ (or dag) over operations $\underline{\cup}$ , $\underline{*}$ and nullaries of the form

   $\{ (A1, \ldots, Ap) \}$ such that $|Ai| \leq 1$,

   of size $\leq |t| . \mathrm{Max}\{ |Q(u)| \ / \ u$ position of t $\}$, cf. page 22.

Question : For which f is f( value of $T$ ) efficiently computable ?

# Different types of evaluations

(1) Numerical values from Sat $\underline{X}$.P($\underline{X}$), as seen above

(2) Extracting tuples from Sat $\underline{X}$.P($\underline{X}$) :

*any* one,

or the *first one* w.r.t. some linear order,

or the set of minimal ones w.r.t. to some partial order.

(3) Parametrized evaluations :

*Examples* : (3.1) Given i, what is the number of sets X of cardinality at most i that satisfy P(X) ?

(3.2) Given j, what is the number of sets X such that j is the number of sets Y that satisfy P(X,Y) ? (Weird question !)

(3.3) Given (i,j), what is the number of sets X such that j is the number of sets Y such that some Z of cardinality $\leq$ i satisfies P(X,Y,Z) ?

We may wish the results for given i,j or the table of values for all i,j.

In (3.1) the "table" is computable from MSp X.P(X). For fixed i, one can adapt the computation of MSp X.P(X) (limit info to sets of card. $\leq$ i).

Cases (3.2) and (3.3) are more difficult.

*Difficulty* for (3.2) :

For $R \subseteq A \times B$ let

$f_R(j)$ := Number of $a \in A$ s.t. $j$ = number of $b \in B$ s.t. $(a,b) \in R$.

If $R = R1 \cup R2$, then $f_R$ cannot be determined from $f_{R1}$ and $f_{R2}$

*Method* : a 2-level construction.

For describing it, we consider in a more concrete way the computations of Sp $\underline{X}.P(\underline{X})$ and MSp $\underline{X}.P(\underline{X})$.

Let $f(G,X) \in D$ for $X \subseteq V_G$ be defined by a deterministic FA over $F^{(1)}$ of the form : $A = < F^{(1)}, Q, \delta, Out >$. We fix a term $t_*X$ that defines G,X.

The state of $A$ at position u is $q(u,X/u)$ and :

$f(G,X) = Out(q(root,X))$, undefined if $q(root,X)$ is not accepting.

Let $f_{Set}(G)$: = { f(G,X) / all X } and

$f_{MSet}(G)$: = **[[** f(G,X) / all X **]]**, the multiset of values f(G,X)

(where we count how many X give each value).

$f_{Set}(G)$ is computed by det(pr(*A*)), the *determinization* of pr(*A*). Its state at u is $q_{Set}(u)$ := { q(u,X/ u) / all X } and $f_{Set}(G)$ = Out($q_{Set}$(*root*)).

$f_{MSet}(G)$ is computed by c-det(pr(*A*)), the *counting-determinization* of pr(*A*). Its state at u is $q_{MSet}(u)$ := **[[**q(u,X/u) / all X/u **]]**, a multiset of states, equivalently { (q,i) / i is the number of sets X/ u s.t. q= q(u,X/u) }.

Then $f_{MSet}(G)$ = Out($q_{MSet}$(*root*)). The image of a multiset "counts occurrences".

Rk : det(*B*) has states in $P_f(Q)$ and c-det(*B*) in $M_f(Q)$ = [ Q → **N** ]$_f$

*Applications* : Let $A$ be a deterministic FA over $F^{(p)}$ that decides $P(\underline{X})$

# $\underline{X}.P(\underline{X})$ is computed by c-det($pr_{All}(A)$):

# $\underline{X}.P(\underline{X})$ (G) = the *size* of $q_{MSet}(root)$ restricted to

accepting states := $\Sigma$ {i / (q,i) $\in$ $q_{MSet}(root)$, q accepting }

Sp $\underline{X}.P(\underline{X})$ : From $A$ we build $B$ that reaches state $q_B(u,\underline{X}/u)$ =

$(q_A(u,\underline{X}/u), |X_1/u|,...,|X_p/u|)$ and $C$ = det($pr_{All}(B)$). Then

Sp $\underline{X}.P(\underline{X})$ (G) = $f_{Set}(G)$ where f(q,w) := w for q accepting.

MSp $\underline{X}.P(\underline{X})$ (G) = $f_{MSet}(G)$ from $A,B,C$ as above.

*Example (3.2)* : Given j, what is $f_G(j)$ := the number of sets X such that j is the number of sets Y that satisfy P(X,Y) ? (For fixed graph G).

Let *A* be a deterministic FA over $F^{(2)}$ that decides P(X,Y); states in Q.

Let *B* = c-det($pr_Y(A)$) (we neglect Y). Its states are in $[\, Q \to \mathbf{N} \,]_f$ : functions $\sigma$ such that $\sigma(q) \neq 0$ for finitely many q (effectively codable).

Let *C* = c-det(pr(*B*)) (we neglect X). Its states are in $[\, [\, Q \to \mathbf{N} \,]_f \to \mathbf{N} \,]_f$ : functions $\theta$ such that $\theta(\sigma) \neq 0$ for finitely many $\sigma$.

Then $f_G(j) = \Sigma \{\, \theta(\sigma) \,/\, \sigma$ is "j-accepting for *B*" $\}$, $\theta = q_C(\mathit{root})$,

$\sigma$ is "j-accepting" $\Leftrightarrow$ j = $\Sigma \{\, \sigma(q) \,/\, q$ accepting for *A* $\}$.

*Optimizations* : (1) By 2 preliminary passes, one determines, at each position u the set Q(u) of states of *A* that are useful (in some accepting run of *A* for some X,Y). All states in Q(*root*) are accepting. One gets *A'*, deterministic but not complete.

The state of *B'*:= c-det($pr_Y$(*A'*)) at u is in [ Q(u) → **N** ]$_f$ . The state of *C'*:= c-det(pr(*B'*)) at u is in [ [ Q(u) → **N** ]$_f$ → **N** ]$_f$, accepting as for *C*.

(2) For computing $f_G(k)$ for a particular value k, we eliminate in the runs of *B'* the values $\sigma(q) > k$ because they cannot be "k-accepting". This optimization is combined with the previous one.

*Question* : Is that better than computing Sat(X,Y).P(X,Y) and extracting from it the function $f_G$ ?

*Examples* : n = number of vertices.

P(X,Y) = *True* : $f_G(j)$ = *if* $j = 2^n$ *then* $2^n$ *else* 0

P(X,Y) = *False* : $f_G(j)$ = *if* $j = 0$ *then* $2^n$ *else* 0

Similar computations by automata (as we will see), but the sets Sat(X,Y).P(X,Y) have cardinalities $2^{2n}$ or 0.

$P(X,Y) : \Leftrightarrow X = Y$ : $f_G(j)$ = *if* $j = 1$ *then* $2^n$ *else* 0 (here $2^n$ satisf. pairs)

$P(X,Y) : \Leftrightarrow X \subseteq Y$ : $f_G(j)$ = *if* $j = 2^{n-q}$ *then* $\binom{q}{n}$ *else* 0 (here $f_G$ has n non-null values)

Cases P(X,Y) = *True* and P(X,Y) = *False*

Automaton *A* over $F^{(2)}$ has one state, call it *Ok*.

Let *B* = c-det($pr_Y(A)$) (we neglect Y). Its state at position *u* is the mapping $\sigma_m$ such that $\sigma_m$ (*Ok*) = $2^m$, wh. m is number of vertices below *u*

Let *C* = c-det(pr(*B*)) (we neglect X). Its state at position *u* maps $\sigma_m$ to $2^m$ and any other map in [ { *Ok* } → **N** ]$_f$ to 0.

Then $f_G(j) = \Sigma \{ \theta(\sigma) / \sigma$ is "j-accepting for *B*" $\}$

$\sigma$ is "j-accepting" $\Leftrightarrow$ j = $\Sigma \{ \sigma(q) / q$ accepting for *A* $\}$.

P=*True* : At the root, $\sigma_n$ is $2^n$ –accepting, which gives $2^n$ iff j = $2^n$

P=*False* : At the root, $\sigma_n$ is 0 –accepting, which gives $2^n$ iff j = 0

In both cases, other values are 0.

Case P(X,Y) : $\Leftrightarrow$ X = Y

Automaton *A* over $F^{(2)}$ has 2 states, *Ok* (accepting) and *Error*.

Let *B* = c-det($pr_Y(A)$) (we neglect Y). Its state at position *u* is the mapping $\sigma_m$ such that $\sigma_m$ (*Ok*) = 1, $\sigma_m$ (*Error*) = $2^m$-1.

Let *C* = c-det(pr(*B*)) (we neglect X). Its state at position *u* maps $\sigma_m$ to $2^m$ and any other map in [ { *Ok, Error* } → **N** ]$_f$ to 0.

Then $f_G(j) = \Sigma$ { $\theta(\sigma)$ / $\sigma$ is "j-accepting for *B*" }

$\sigma$ is "j-accepting" $\Leftrightarrow$ j = $\Sigma$ { $\sigma$(q) / q accepting for *A* }.

At the root, $\sigma_n$ is 1 –accepting, which gives $2^n$ iff j = 1 (and 0 otherwise). Using the optimization of p. 33, we can omit *Error*.

*Example* : P(X,Y) : $\Leftrightarrow$ X $\leq$ Y , the lexicographic order.

The set of vertices is linearly ordered (leaves of an ordered tree, MSO definable) and

X $\leq$ Y is defined by : X = Y , or,

letting u be the smallest element that distinguishes X and Y

then u > X or u $\in$ X and u is not > Y

If X is the *p*–th in the set *P*(V), the number of Y $\geq$ X is $2^n - p + 1$.

Hence, $f_G(2^n - p + 1) = 1$ for $1 \leq p \leq 2^n$ and so, $f_G$ has an exponential number of non-null values.

They are all equal !!

The "matrix" of Sat(X,Y).P(X,Y) is diagonal.

Case of example (3.3) :

Given (i,j), what is the number of sets X such that j is the number of sets Y such that some Z of cardinality $\leq$ i satisfies P(X,Y,Z) ?

# Conclusion

1) By uniform constructions, we get dynamic programming algorithms based on fly-automata, that can be quickly constructed from logical descriptions → *flexibility*.
2) In many cases they are implemented. Tests have been made for colorability and connectedness problems. It is hard to obtain upperbounds to time computations. We do not get better algorithms than the specific ones that have been developed.

3) These constructions are applicable to bounded tree-width and MSO with edge set quantifications by means of incidence graphs.

4) Basic numerical evaluations (cardinality, spectrum, multi-spectrum) have been implemented (by I. Durand). Fly-automata can also handled combinations of these basic evaluations. It remains to see if implementation is feasible.

# Appendix (if anybody wants)

*Examples of MSO definability : 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 $K_{3,3}$).*