Monadic second-order logic for graphs.

Algorithmic and language theoretical applications

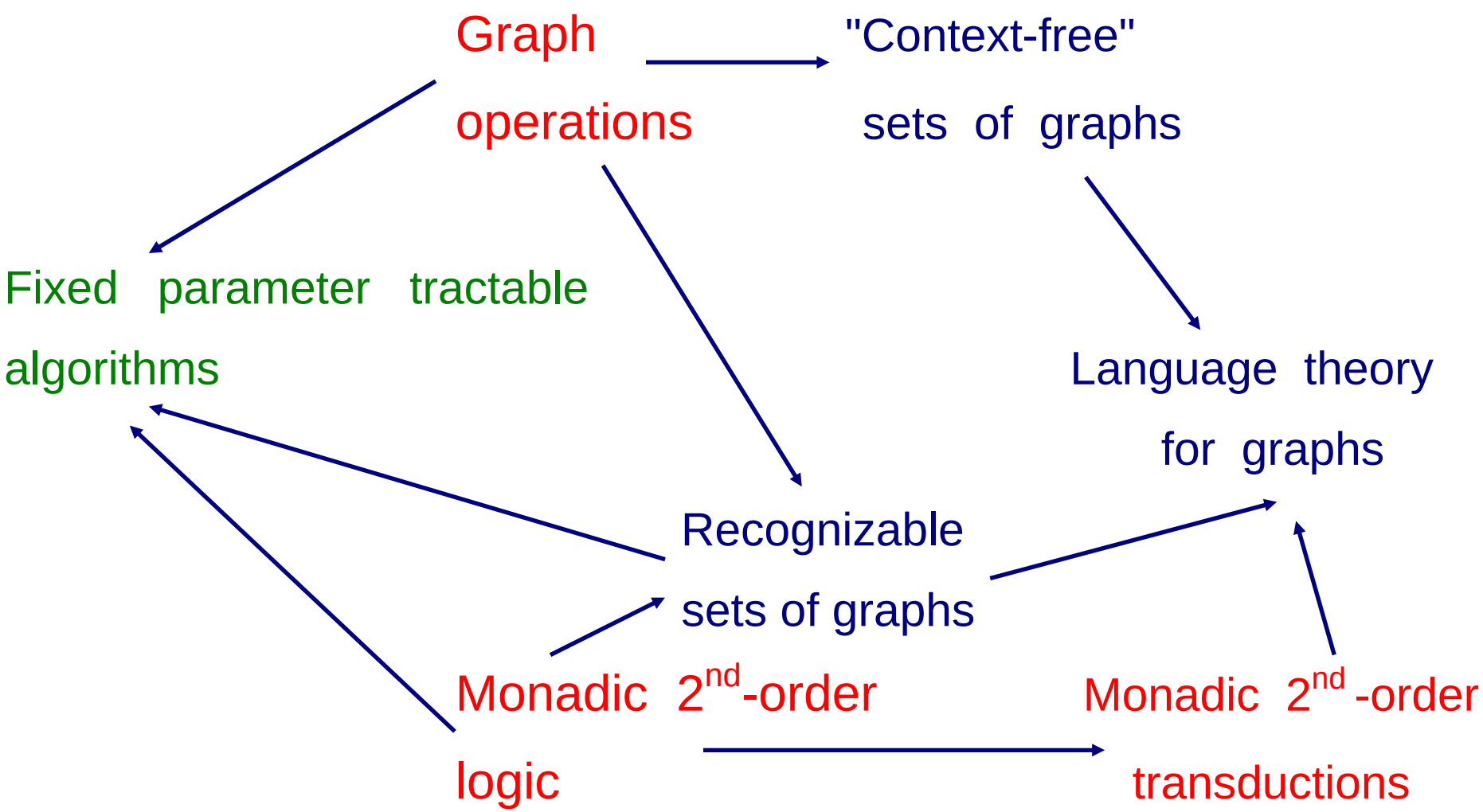
Part 2

Bruno Courcelle

Université Bordeaux 1, LaBRI, and Institut Universitaire de France

Reference : Graph structure and monadic second-order logic,
book to be published by Cambridge University Press, readable on :
http://www.labri.fr/perso/courcell/ActSci.html

An overview chart



Summary

1. Context-free sets defined by equation systems
2. Two graph algebras. Tree-width and clique-width
3. Recognizability : an algebraic notion
4. Monadic second-order sentences define recognizable sets.

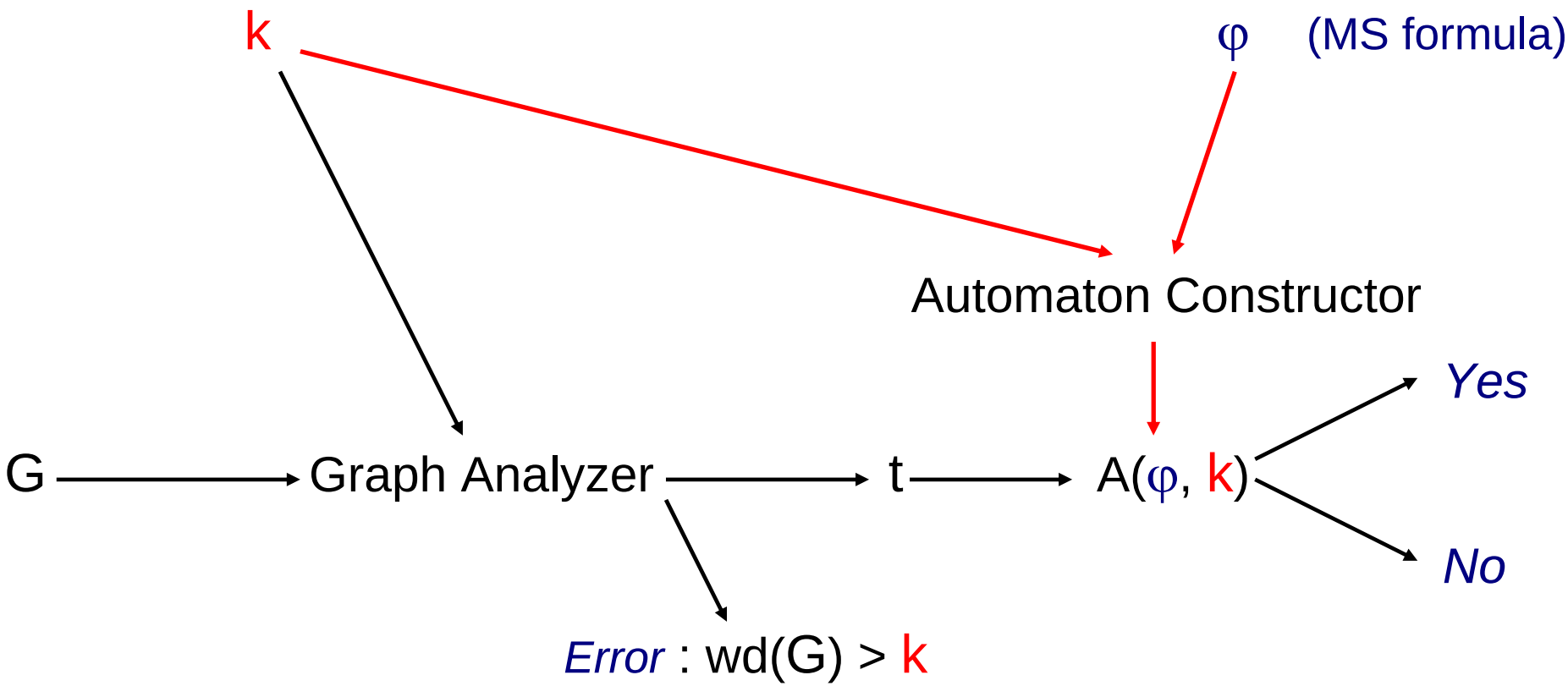
Part 2

5. Fixed-parameter tractable algorithms; constructions of automata.

Part 3

6. Monadic second-order transductions.
7. Robustness results : preservation of classes under direct and inverse monadic second-order transductions. Short proofs in graph theory. (black = graph theory)
8. Logic and graph structure theory : Graph classes on which monadic second-order logic is decidable
9. Some open questions

5. FPT algorithms : Constructions of automata



Steps → done "once for all", independent of G A(φ,k): finite automaton on terms (wd = tree-width or clique-width or equivalent)

Difficulties

1. Parsing : construction of terms (based on tree-decompositions or other graph decompositions).

The *linear-time exact* parsing algorithm by Bodlaender (for tree-width) and the *cubic approximate* parsing algorithm by Hlineny & Oum (for clique-width via *rank-width*) are not implementable.

Bodlaender reports about usable algorithms for (non-random) graphs with 50 vertices and tree-width $\leq$ 35

Specific algorithms : (1) Flow-graphs of structured programs have tree-width at most 6 and tree-decompositions are easy from the parse trees of programs (Thorup).

(2) For certain graph classes of bounded clique-width defined by forbidden induced subgraphs, optimal clique-width terms can be constructed in polynomial time (by using *modular decomposition*).

2. Sizes of automata

The automata $A(\varphi, k)$ may be too large for being practically compiled.

The construction by induction on the structure of φ may need intermediate automata of huge size, even if the *unique minimal deterministic* automaton equivalent to $A(\varphi, k)$ has a manageable number of states.

Examples : Soguet *et al.* using MONA have constructed automata for the following cases ; no success for clique-width 4 :

	Clique-width 2	*Clique-width 3*
MaxDegree≤3	91 states	Space-Out
Connected	11 states	Space-Out
IsConnComp(X)	48 states	Space-Out
Has≤4-VertCov	111 states	1037 states
HasClique ≥ 4	21 states	153 states
2-colorable	11 states	57 states

Other examples of automata too large to be constructed, i.e., “compiled”:

for k = 2 : 4-colorability, 3-acyclic-colorability, NoCycle (i.e., is a forest)

for k = 5 : 3-colorability, clique

for k = 4 : connectedness.

This is not avoidable :

The number of states of $A(\varphi, k)$ is bounded by an h-iterated exponential where h is the number of quantifier alternations of φ.

There is no alternative construction giving an upper bound with a bounded nesting of exponentiations (Meyer & Stockmeyer, Weyer, Frick & Grohe).

What to do ?

(1) Focus on MS formulas *without quantifier alternation*, written with "powerful" atomic formulas expressing basic graph properties :

Path(X,Y) : X has exactly 2 vertices linked by a path in G[Y],

NoEdge(X) : G[X] has no edge,

Conn(X) : G[X] is connected,

NoCycle(X) : G[X] is a forest.

(2) Using fly-automata : transitions are *not compiled* but computed as needed

Examples of graph properties

(1) p-colorability

$\exists\, X_1,\ldots,X_p$ (Partition$(X_1,\ldots,X_p)$ $\wedge$ NoEdge(X_1) $\wedge$ ..... $\wedge$ NoEdge(X_p))

(2) p-*acyclic* colorability

$\exists\, X_1,\ldots,X_p$ (Partition$(X_1,\ldots,X_p)$ $\wedge$ NoEdge(X_1) $\wedge$ ..... $\wedge$ NoEdge(X_p) $\wedge$ ...
..... $\wedge$ NoCycle$(X_i \cup X_j)$ $\wedge$ .....) (all $i < j$).

We use set terms : $X_i \cup X_j$, to avoid some quantifications.

(3) minor inclusion : *H* simple, loop-free. *Vertices(H)* = { $v_1,\ldots,v_p$ }

$\exists X_1,\ldots,X_p$ (Disjoint$(X_1,\ldots,X_p)$ $\wedge$ Conn(X_1) $\wedge$ ... $\wedge$ Conn(X_p) $\wedge$ ...
... $\wedge$ Link(X_i , X_j) $\wedge$ ...)

with Link(X_i, X_j) for each edge v_i -- v_j of H ; (there exists an edge between X_i and X_j.)

Remarks on Conn(X) (connectedness of G[X])

and the automata for terms that define graphs of clique-width $\leq$ k.

There are two *non-deterministic* automata for Conn(X) and ¬Conn(X) that have both $2^{O(k.k)}$ states.

There is a *deterministic* automata for Conn(X) (and also ¬Conn(X)) that has $2^{O(2\ \hat{}\ k)}$ states. Its states are the sets of subsets of [k]={1,…,k}

The *minimal deterministic* automaton for Conn(X) has more than $2^{2\ \hat{}\ (k/2)}$ states.

Construction of A(φ, k) for "clique-width" terms

k = the number of vertex labels = the bound on clique-width

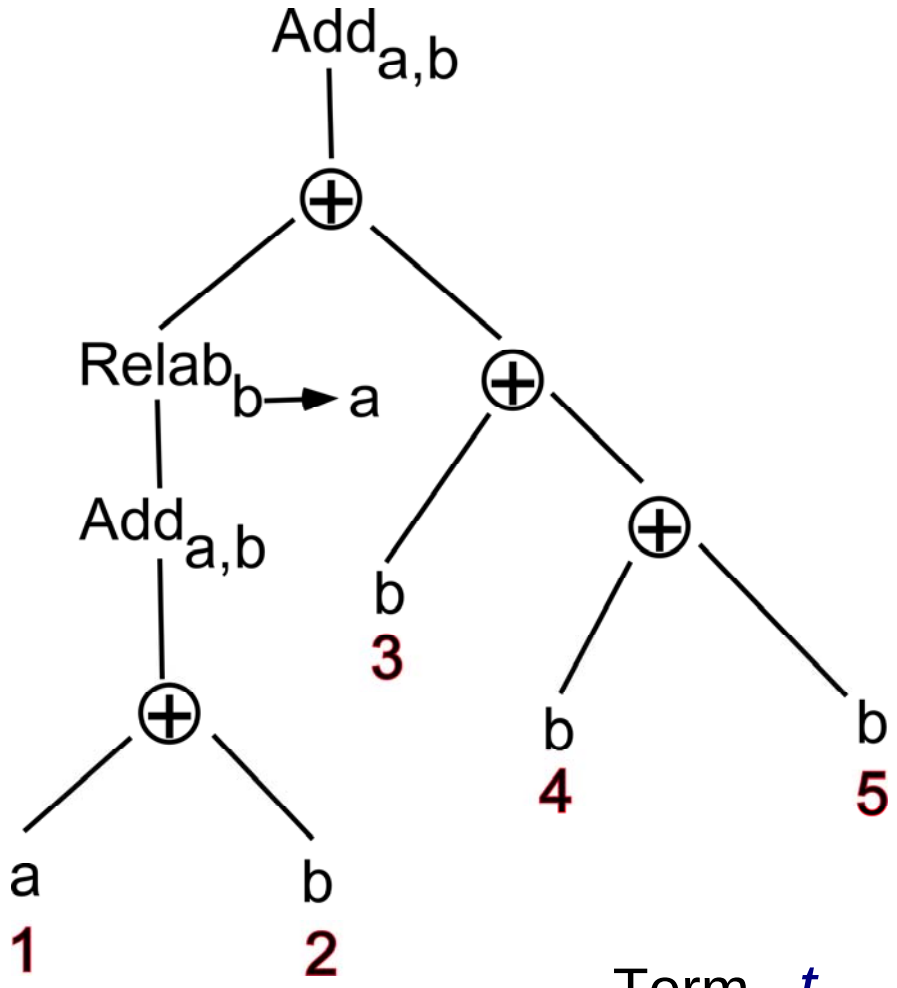
F = the corresponding set of operations and constants :

$\mathbf{a}$, $\varnothing$, $\oplus$, $Add_{a,b}$, $Relab\ a \longrightarrow b$

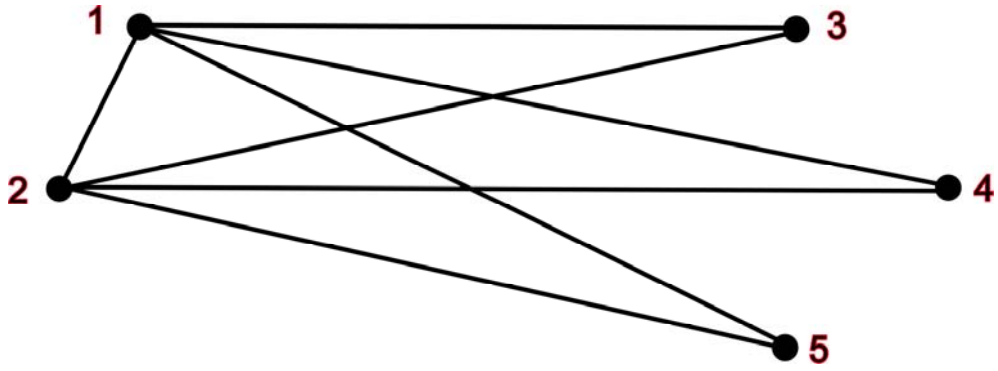
G(t) = the graph defined by a term t in **T**(F).

Its vertices are (in bijection with) the occurrences of the constants in t that are not $\varnothing$

Example



Term *t*



Graph G(*t*)

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 MS formulas (*formulas are written without first-order variables*):

1) we replace in F each constant **a** by the constants $(\mathbf{a}, (w_1,\ldots,w_n))$ where $w_i \in \{0,1\}$: we get $F^{(n)}$

(only constants are modified);

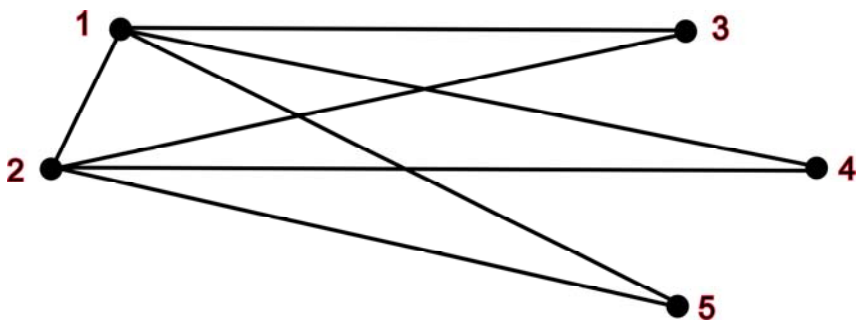
2) a term *s* in $\mathbf{T}(F^{(n)})$ encodes a term *t* in $\mathbf{T}(F)$ and an assignment of sets $V_1,\ldots,V_n$ to the set variables $X_1,\ldots,X_n$:

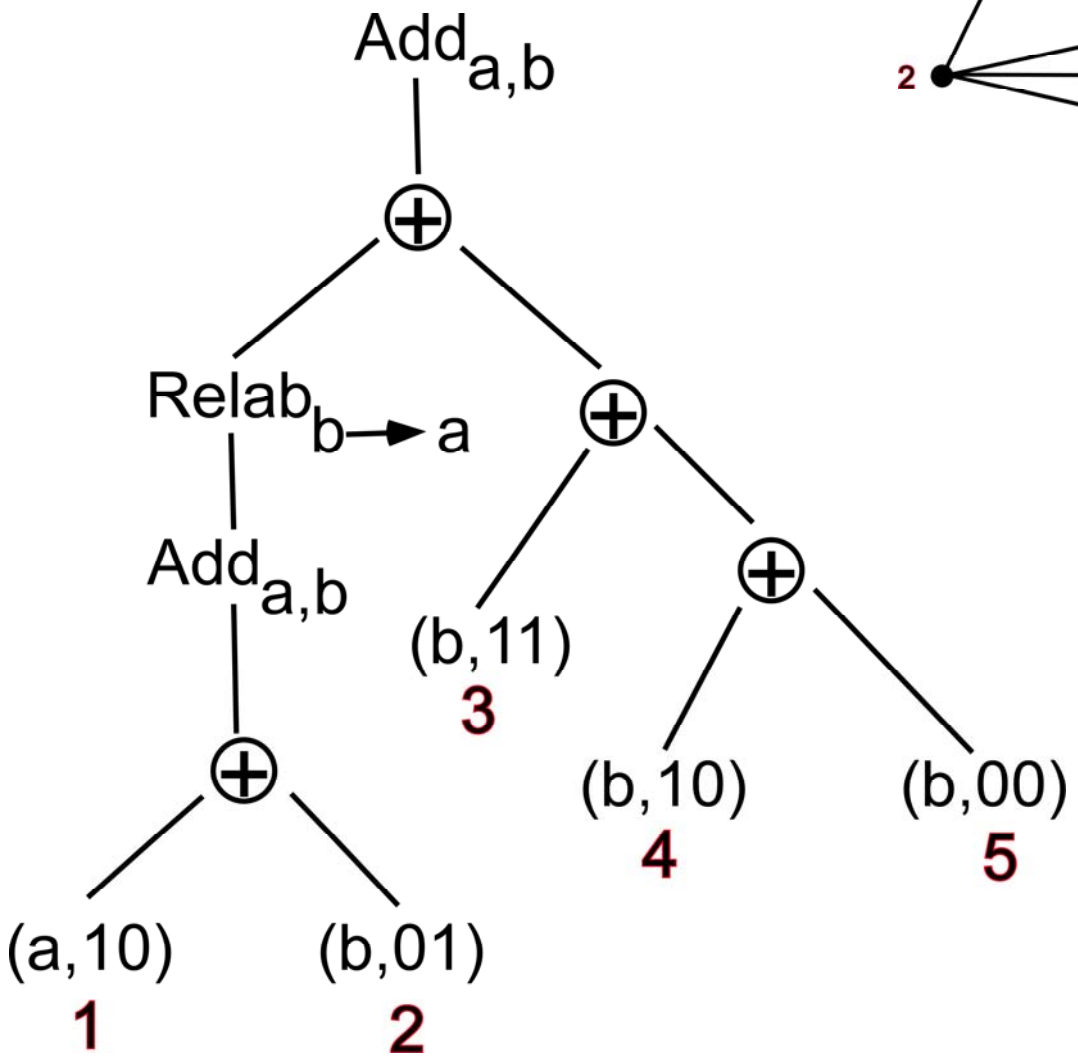
if *u* is an occurrence of $(\mathbf{a}, (w_1,..,w_n))$, then

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

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

Example (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 $\varphi(X_1,\ldots,X_n)$ a finite (bottom-up) deterministic automaton $A(\varphi(X_1,\ldots,X_n), k)$ that recognizes:

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

Theorem : For each sentence φ, the automaton $A(\varphi, k)$ accepts in time $f(\varphi, k).\mid t \mid$ the terms t in $\mathbf{T}(F)$ such that $G(t) \mid = \varphi$

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 parameter is clique-width, or, for undirected graphs, the equivalent graph complexity measure *rank-width* defined by Oum & Seymour).

The inductive construction of $A(\varphi, k)$

Atomic formulas : discussed below.

For $\wedge$: product of two automata (deterministic or not)

For $\vee$: union of two automata (or product of two *complete* automata; product preserves determinism)

For *negation* : exchange accepting / non-accepting states

for a complete *deterministic* automaton

Quantifications: Formulas are written without $\forall$

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

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

where pr is the "*projection*" that eliminates the last Boolean.

→ a *non-deterministic* automaton.

oOo

The number of states is an h-iterated exponential,

where h = maximum nesting of negations.

Some tools for constructing automata

Substitutions and inverse images ("cylindrifications").

1) If we know $A(\varphi(X_1, X_2))$, we can get easily $A(\varphi(X_4, X_3))$:

$$L(\varphi(X_4, X_3)) = h^{-1}(L(\varphi(X_1, X_2)))$$ where

h maps $(\mathbf{a}, (w_1, w_2, w_3, w_4))$ to $(\mathbf{a}, (w_4, w_3))$

We take

$$A(\varphi(X_4, X_3)) = h^{-1}(A(\varphi(X_1, X_2)))$$

This construction preserves determinism and the number of states.

2) From $A(\varphi(X_1, X_2))$, we can get $A(\varphi(X_3, \overbrace{X_1 \cup (X_2 \setminus X_4)}^{\textit{Set term}}))$ by h^{-1}

with h mapping $(\mathbf{a}, (w_1, w_2, w_3, w_4))$ to $(\mathbf{a}, (w_3, w_1 \vee (w_2 \wedge \neg w_4)))$

Relativization to subsets by inverse images.

If φ is a closed formula expressing a graph property P, its relativization $\varphi\,[X_1]$ to X_1 expresses that the subgraph induced on X_1 satisfies P. To construct it, we replace recursively

$$\exists\, y.\ \theta \quad \text{by} \quad \exists\, y.\ y \in X_1 \wedge \theta, \ \text{etc…}$$

However, there is an easy transformation of automata :

Let h map (**a** , 0) to $\varnothing$ and (**a** , 1) to **a**.

$$L(\varphi\,[X_1]) = h^{-1}(L(\varphi))$$

Hence:

$$A(\varphi\,[X_1]) := h^{-1}(A(\varphi))$$

The inductive construction (continued) :

Complete *deterministic* automata for atomic formulas and basic graph properties : automaton over $F^{(n)}$ recognizing the set of terms

$$t * (V_1,\ldots,V_n) \quad \text{in} \quad L(\varphi(X_1,\ldots,X_n))$$

Intuition : in all cases, the *state* at node u represents a *finite information* q(u) about the graph G(t / u) and the restriction of $(V_1,\ldots,V_n)$ to the vertices below u (vertices = leaves)

1) if u = f(v,w), we want that q(u) is defined from q(v) and q(w) by a fixed function : → the *transition function* ;

2) whether (G(t), $V_1,\ldots,V_n$) satisfies $\varphi(X_1,\ldots,X_n)$ must be checkable from q(*root*) : → the *accepting states*.

Atomic and basic formulas :

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

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

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

→ Easy constructions with small numbers of states :
respectively 2, 2, 3, q, q+1.

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

Atomic formula : edg(X_1,X_2) for directed edges

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

Vertex labels $\in$ a set C of k labels.

k^2+k+3 *states* : 0, Ok, a(1), a(2), ab, Error, for a,b in C , $a \neq b$

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

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

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

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

a(2) : $X_1 = \varnothing$, $X_2 = \{w\}$, 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

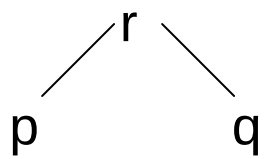
Transition rules

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 $\overrightarrow{Add}_{a,b}$

r

p

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

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

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 or 0 or Ok or c(i) or cd or dc where c ≠ a

then r : = p

Sizes of some deterministic automata :

k = bound on clique-width

Property	Partition $(X_1,\ldots,X_p)$	edg(X,Y)	NoEdge	Connected, NoCycle for degree $\leq p$	Path(X,Y)	Connected, Nocycle
Number of states N(k)	2	k^2+k+3	2^k	$2^{O(p.k.k)}$	$2^{O(k.k)}$	$2^{2^{O(k)}}$

Difficulties with the *sizes* of the automata $A(\varphi, k)$

Examples of automata too large to be constructed, i.e., “compiled”:

for k = 2 : 4-colorability, 3-acyclic-colorability, NoCycle (i.e., is a forest)

for k = 4 : connectedness,

for k = 5 : 3-colorability, clique.

From now on : work in progress

Fly-automata :

States and transitions are not listed in huge tables :

they are *specified* (in uniform ways for all k) by "small" programs.

Example of state for connectedness :

q = { {a}, {a,b}, {b,c,d}, {b,d,f } },

a,b,c,d,f are vertex labels; q is the set of *types* of the connected components of the current graph.

Some transitions :

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

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

Transitions for $\oplus$: union of sets of types.

This method works for formulas with no quantifier alternation, but that use "powerful atomic formulas".

Examples : p-acyclic colorability

$$\exists\, X_1,\ldots,X_p\ (\text{Partition}(X_1,\ldots,X_p) \wedge \text{NoEdge}(X_1) \wedge \ldots\ldots \wedge \text{NoEdge}(X_p) \wedge \ldots$$
$$\ldots\ldots\ldots\ldots \wedge \text{NoCycle}(X_i \cup X_j) \wedge \ldots\ldots\ )$$

Minor inclusion : *H* simple, loop-free. *Vertices(H)* = { $v_1,\ldots,v_p$ }

$$\exists X_1,\ldots,X_p\ (\text{Disjoint}(X_1,\ldots,X_p) \wedge \text{Conn}(X_1) \wedge \ldots \wedge \text{Conn}(X_p) \wedge \ldots$$
$$\ldots \wedge \text{Link}(X_i\ ,\ X_j) \wedge \ldots\ )$$

Existence of "holes" : odd induced cycles (to check *perfectness* ; one checks "anti-holes" on the edge-complement of the given graph).

Some experiments with fly-automata (by Irène Durand, LaBRI)

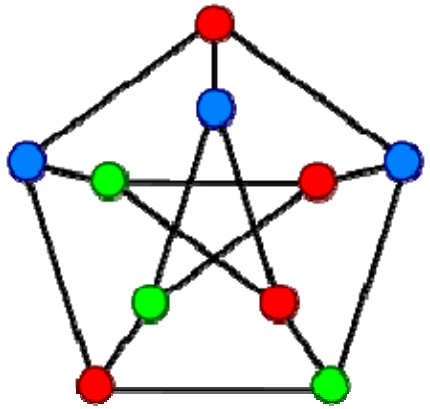
3-colorability of the 6 x 300 grid (of clique-width 8) in less than 2 hours,

4-acyclic-colorability of the Petersen graph (clique-width 7) in 17 minutes.

(3-colorable but not acyclically;

red and **green** vertices

induce a cycle).



Closure properties of fly-automata (can be non-deterministic):

Union (for $\vee$)

Product (for $\wedge$)

Image (for $\exists$)

Inverse image (for substitutions and relativization)

New tool : Annotations

At some positions in the given term, we attached some (finite) contextual information.

Example :

At position u in a term *t*, we attach the set

$ADD_t(u)$ = the set of pairs (a,b) such that some operation

$Add_{c,d}$ *above* u (hence, in its "context") adds edges between the (eventual) vertices *below* u labelled by a and b.

These sets can be computed in *linear time* by means of a top-down traversal of *t*.

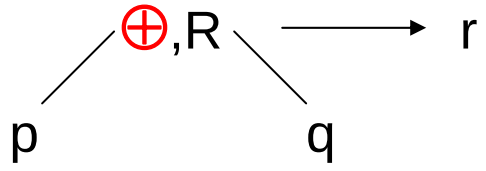
Certain automata on annotated terms may have less states.

Example : *edg*(X_1, X_2) : 2k+3 states instead of k^2 +k +3 (cf. page 21):

0, Ok, a(1), a(2), Error, for a in C.

Transitions for $\oplus$ annotated by R :

(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) $\in$ R $\wedge$ then r := Ok ;

and if (a , b) $\notin$ R $\wedge$ then r := Error ;

if p = b(2), q = a(1) : *idem* ;

otherwise r : = Error.

Other examples :

For *Clique*(X) meaning that X induces a clique :

$2^k + 2$ states instead of $2^{O(k.k)}$.

For *Connectedness* : same states but they "shrink" quicker :

cf. the rules for $Add_{a,c}$ on page 26.

Model-checking of MS_2 formulas

(1) By Kreutzer, Makowsky *et al.,* FPT model-checking for MS_2 formulas (using edge quantifications) *needs* restriction tree-width as parameter unless P=NP, ETH, Exptime=NExptime *etc*…

(2) The case of MS_2 formulas reduces to that of MS ones:

- G of tree-width $k \geq 2 \rightarrow$ Inc(G) has tree-width k,

  hence, clique-width $\leq 2^{O(k)}$ (exponential blow-up !)

- every MS_2 property of G is an MS property of Inc(G)

(3) For a "direct" construction, we need :

(3.1) Terms to represent graphs, over appropriate operations.

(3.2) A representation of vertices *and edges* by occurrences of operations and constants in these terms.

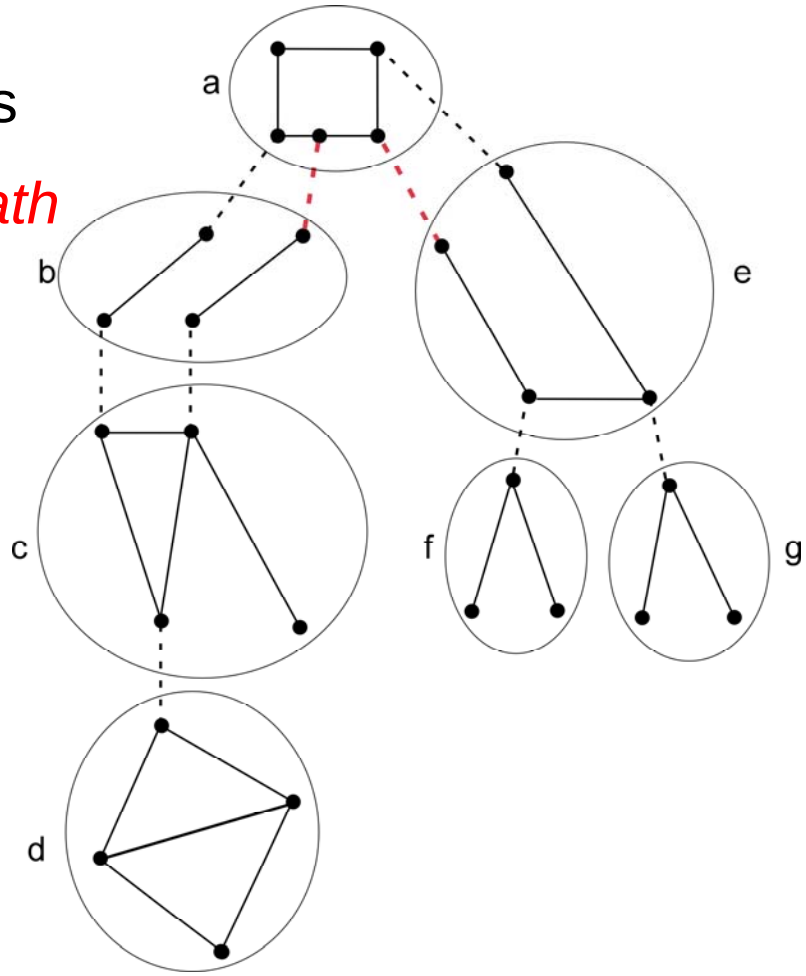
(3.2.1) : For "clique-width" terms : we have *no* good representation of edges because each occurrence of $Add_{a,b}$ may add simultaneously an unbounded number of edges.

(3.2.2) : For "special terms", i.e., "clique-width" terms where each occurrence of $Add_{a,b}$ adds a unique edge, we have such a bijection. This is OK for graphs of bounded *special tree-width* (but not for bounded tree-width). (cf. my lecture to FST-TCS 2010).

Special tree-width is the minimal width of a *special tree-decomposition* (T,f) where :

(a) T is a rooted tree,

(b) the set of nodes whose boxes contain any vertex is a *directed path*

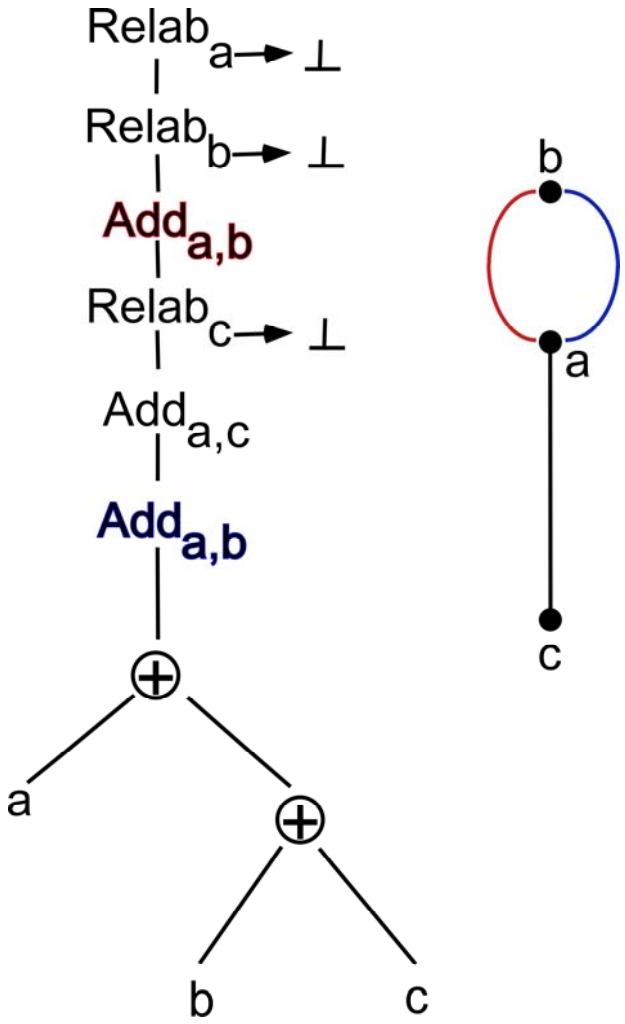


Motivations : (1) Comparison with clique-width (no exp. blow-up)

cwd(G) ≤ sptwd(G)+2

(2) The automata for checking adjacency are exponentially smaller than for bounded tree-width

Special terms :



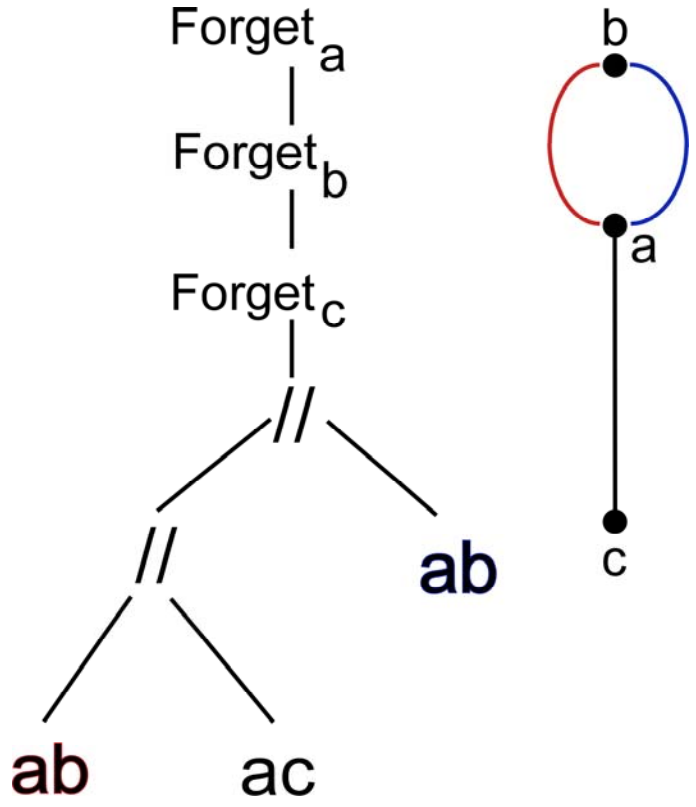
The leaves represent the vertices.

The nodes labelled $Add_{a,b}$ and $Add_{a,c}$ represent the edges ; each occurrence of $Add_{a,b}$ represents one of the two parallel edges

(3.2.3) : Case of terms characterizing tree-width

How to handle them "directly", as for special terms ? The difficulty is to have a bijection between *nodes* in the term and the *vertices and edges* of the graph.

Vertices are in bijection with the occurrences of *Forget* operations. The edges are at the leaves of the tree, *below* the nodes representing their ends.



The automaton for *edg*(X,Y) has $2^{\Theta(k.k)}$ states ($O(k^2)$ for sptwd). Too bad for a basic property !

An improvement using annotations

Undirected graphs of tree-width $\leq$ k-1 are denoted by terms over the operations of the HR algebra : $//$, $Forget_a$ and the constants a, ab for a,b $\in$ [k]={1,…,k}. *Without renamings of labels*.

The vertices are in bijection with the occurrences of the *Forget* operations.

The annotation : at each occurrence u of $Forget_a$ representing a vertex x is attached the set of labels b such that the first occurrence of $Forget_b$ above u represents a vertex adjacent to x.

The automaton for *edg*(X,Y) has $2^{2k}+2$ states (instead of $2^{\Theta(k.k)}$).

Remarks :

incidence : in(X,Y) uses k^2 + 3 states (for undirected graphs) (only k+3 states for directed graphs).

adjacency : edg(X,Y), can be written $\exists Z$ (in(Z,X) $\wedge$ in(Z,Y)) (for undirected graphs) which gives a deterministic automaton with $2^{O(k.k)}$ states.

With this annotation, incidence and adjacency are handled separately on “redundant” representations of graphs by terms.

Conclusion

Using automata for model-checking of MS sentences on graphs of bounded tree-width or clique-width is not hopeless if we use fly-automata, built from (possibly non-deterministic) "small" automata for basic graph properties (and their negations), and for sentences without quantifier alternation (in order to keep flexibility, by allowing variations on the input sentences).

More tests on significant examples are necessary, and also comparison (theory and practice) with other approaches : games, monadic Datalog, specific problems, "Boolean width".

Can one adapt fly-automata to counting and optimization problems?