Special tree-width and the verification of MS graph properties with edge quantifications

Bruno Courcelle

Institut Universitaire de France & Université Bordeaux 1, LaBRI

References : B. C. : Graph structure and monadic second-order logic,
book to be published by *Cambridge University Press*,

B. C. : On the model-checking of monadic second-order formulas with edge set quantifications. Discrete Applied Maths, to appear

Readable on: http://www.labri.fr/perso/courcell/ActSci.html

From the talk by Irène Durand:

Fixed-parameter tractable model-checking algorithms

for monadic second-order (MS) sentences on graphs

with respect to clique-width.

Introduction of *fly-automata*.

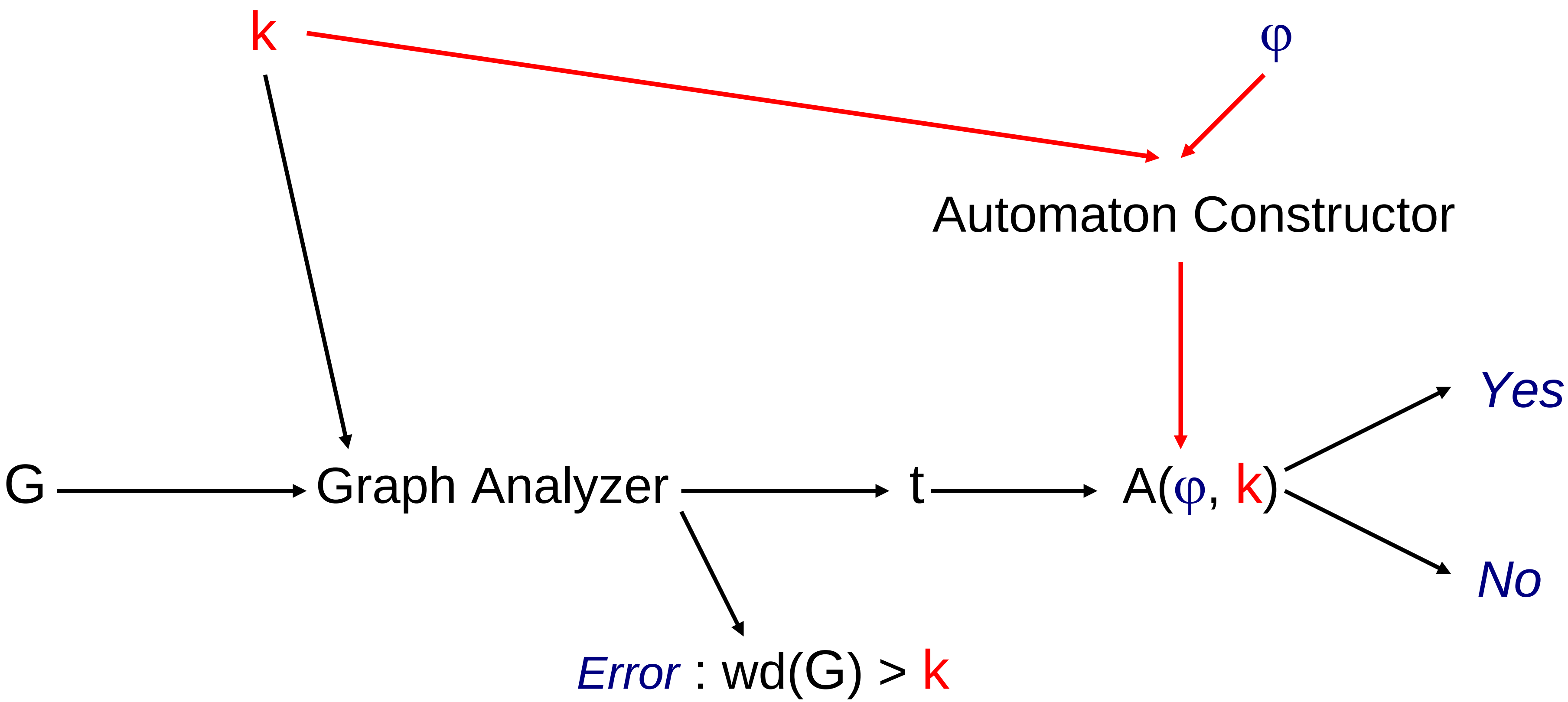
This talk: Extension to MS sentences using edge quantifications

with respect to tree-width.

The class of properties is larger but the parameter is weaker.

A variant of tree-width: *special tree-width*, automata easier to construct.

The general scheme



Steps ⟶ done "once for all", independent of G

A(φ, k) : a *fly*-automaton on terms (wd=tree-width, clique-width, rank-width).

Model-checking problems (for graphs)

MS formulas

MS_2 formulas

using edge quantifications

$G = (V_G , edg_G(.,.).)$

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

For G undirected : $inc_G(e,v) \Leftrightarrow$

v is a vertex (in V_G) of edge e (in E_G).

FPT for clique-width

FPT for tree-width

Comparisons :

(1) The existence of a perfect matching or a Hamiltonian circuit is expressible by an MS_2 formula, but *not by an* MS formula

(2) By Kreutzer, Makowsky et al. MS_2 model-checking *needs* restriction to bounded tree-width unless P=NP, ETH, Exptime=NExptime *etc*…

(3) *Case of* MS_2 *reduces to Case of* MS

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

hence clique-width $\leq 2^{O(k)}$

MS_2 property of G = MS property of Inc(G)

Difficulties for a practical use :

(1) The *sizes* of the automata A(φ, k) : we use *fly-automata* for formulas with few alternations of quantifiers.

(2) Clique-width $\leq$ $2^{O(k)}$ → automata of large sizes.

(2') Other method : construct automata for the graph operations that characterize tree-width → difficulty due to // that fuses vertices.

Same difficulty in (2) and (2') : the responsible is //.

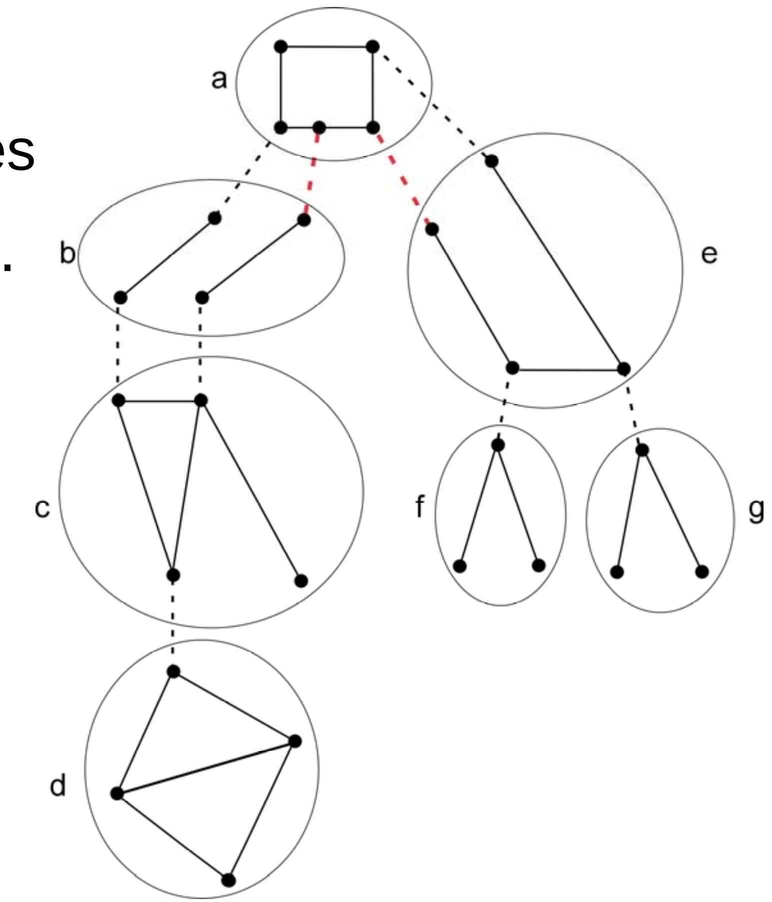
Fact : if G has *path-width* k, then Inc(G) has clique-width $\leq$ k+2, and not $2^{O(k)}$. For these graphs // is not needed.

Special tree-width

Definition : Based on *special tree-decompositions* (T,f) where :

(a) T is a rooted tree,

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



Motivations : (1) Comparison with clique-width.

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

Properties of special tree-width

twd = tree-width ; pwd = path-width ; sptwd = special tree-width ;

cwd = clique-width.

1) $twd(G) \leq sptwd(G) \leq pwd(G)$

2) $cwd(G) \leq sptwd(G) + 2$ (For G simple).

whereas $cwd(\mathrm{G}) \leq 2^{2.twd(\mathrm{G}) + 1}$ (exponential is not avoidable)

3) $sptwd(G) \leq 20\,(twd(G)+1).\,MaxDegree(G)$

(for a set of graphs of bounded degree, bounded special tree-width is *equivalent* to bounded tree-width).

4) Trees have special tree-width 1 (= tree-width) but

graphs of tree-width 2 have *unbounded special tree-width*.

5) The class of graphs of special tree-width $\leq$ k is closed under:

- reversals of edge directions,
- taking *topological minors* (subgraphs and smoothing vertices)

  but *not under taking minors*.

Graphs of tree-width 2 have *unbounded special tree-width*.

Proof sketch: If $G \otimes *$ (= G augmented with a universal vertex $*$) has special tree-width k, then it has path-width $\leq$ k.

Let G be any tree : $G \otimes *$ has tree-width 2.

If $G \otimes *$ has special tree-width $\leq$ k, then G has path-width $\leq$ k.

But trees have *unbounded path-width*, hence graphs of tree-width 2 have unbounded special tree-width.

Terms that characterize special tree-width

and the construction of automata for verifying MS_2 properties.

They use the graph operations that define clique-width *extended to graphs with multiple edges*. (Key point : no "vertex fusion" is needed)

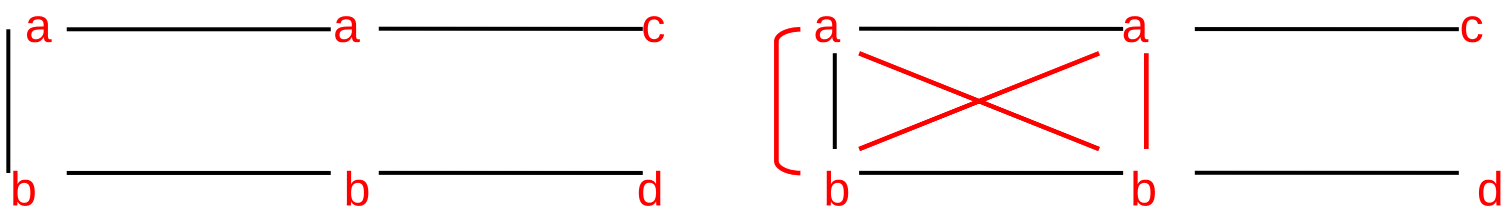
Graphs have vertex labels: *a,b,c ,...* ; each vertex has a single label.

Graphs are loop-free (to simplify).

Binary operation : *disjoint union* : $\oplus$ (Well-defined up to isomorphism: one takes disjoint copies; G $\oplus$ G is *not* equal to G)

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

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



The directed version of $\overrightarrow{Add_{a,b}}$ adds directed edges *from* every *a*-labelled vertex *to* every *b*-labelled vertex.

Vertex relabellings :

Relab$_{a \longrightarrow b}$(G) is G with every label *a* changed into *b*

Variant : *Relab* $_h$ (G) is G with every label *a* changed into *h(a)* for some function *h* : C $\rightarrow$ C ; C is the *finite* set of labels.

Basic graphs: **a** : one vertex labelled by a, for each a in C;

$\varnothing$: to denote the empty graph (it will be useful)

Definition : A graph G (*not necessarly simple*) has clique-width $\leq$ k

$\Leftrightarrow$ it can be constructed from basic graphs with the operations

$\oplus$, $\overrightarrow{Add}_{a,b}$, $Add_{a,b}$, $Relab_{a \longrightarrow b}$ and constants **a** (and $\varnothing$) with labels *a, b* in a set C of k labels.

Its (exact) clique-width *cwd*(G) is the smallest such k.

Proposition: G has *special tree-width* $\leq$ k if and only if it is defined by a *special term* using $\leq$ k + 2 labels (including the particular label $\perp$).

Definition: *Special terms*

They use the operations for clique-width with the following restrictions:

1) The set C of labels contains $\perp$ to mean “terminated vertex”.

2) Operations $Relab_{a \longrightarrow c}$ and $Add_{a,b}$ only if $a, b \neq \perp$.

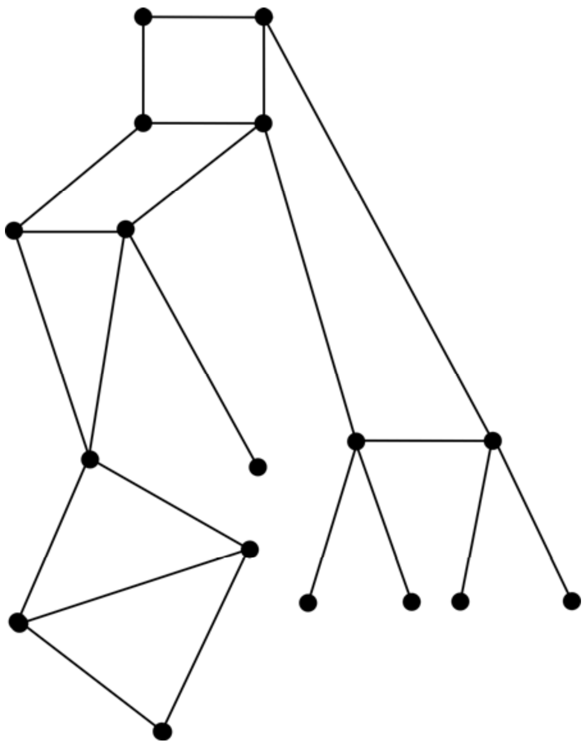
3) Subterms define graphs with ≤ 1 vertex labelled by a if $a \neq \perp$

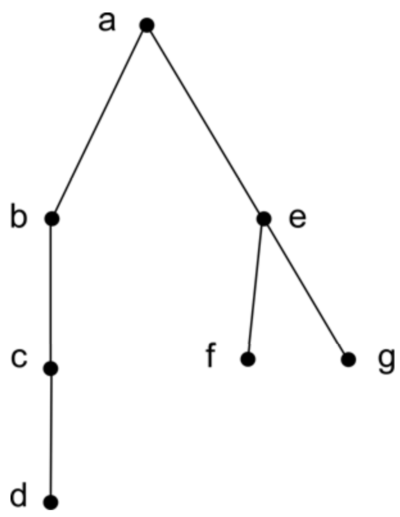
4) $Add_{a,b}(t)$ allowed as subterm only if $G(t)$ has one vertex **x** labelled by a and one vertex **y** labelled by b.

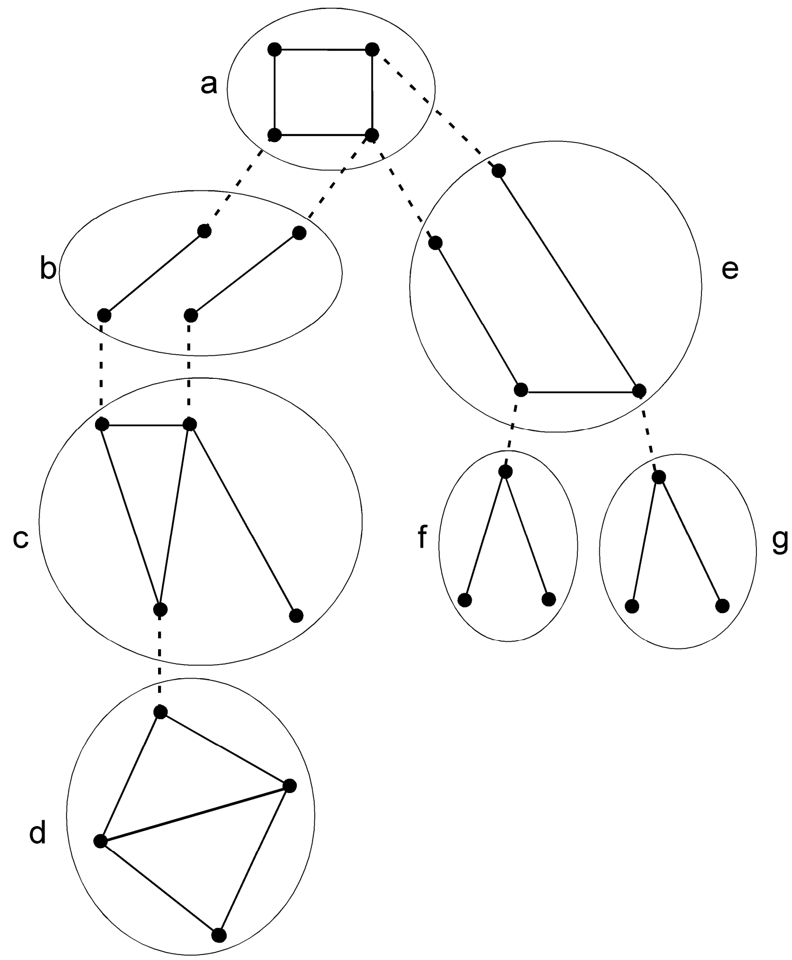
Edges are added “one by one” and are in bijection with the occurrences of the operations $Add_{a,b}$. These operations can define multiple edges.

Similar definitions for directed graphs.

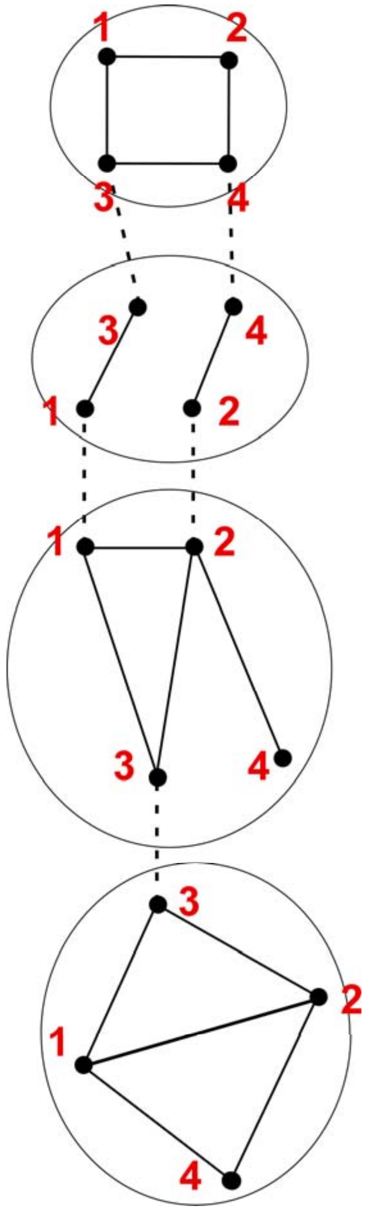
Tree-decompositions (no definition is needed !).







Comparing path-width and clique-width :

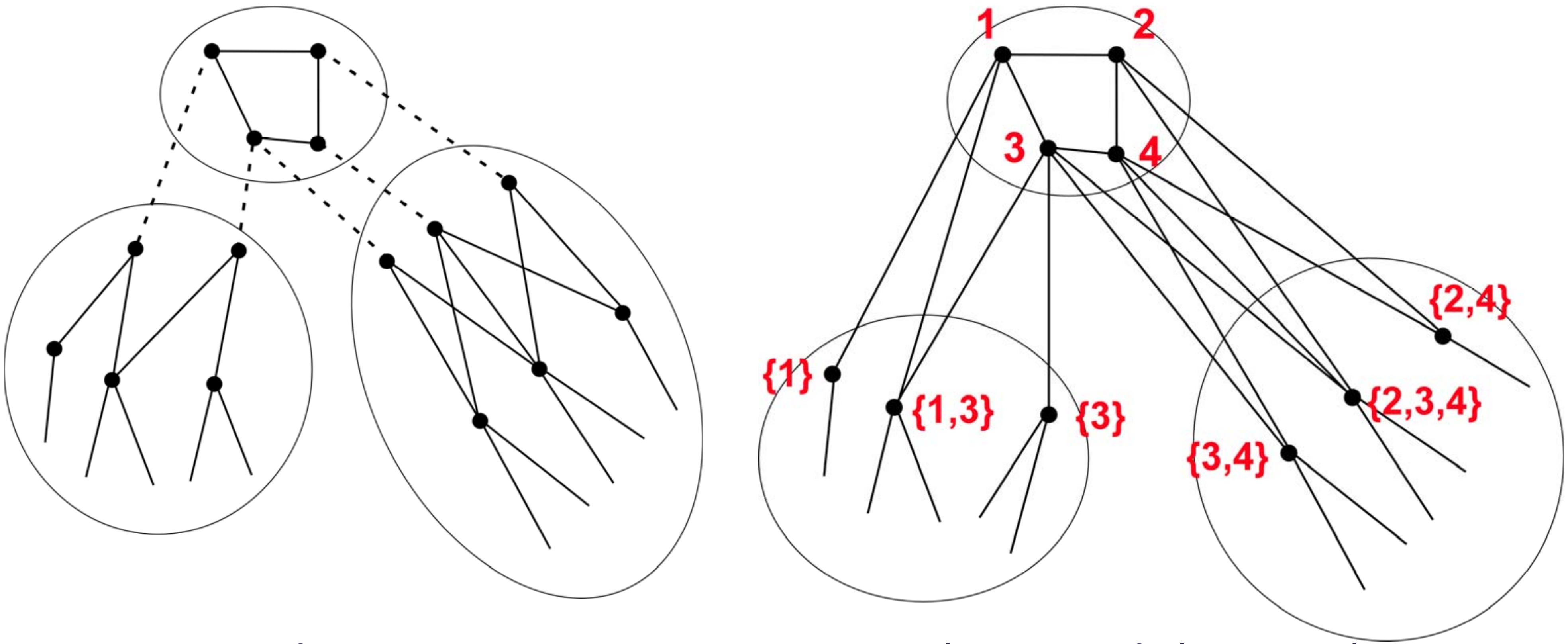


$$cwd(G) \leq pwd(G) + 2$$

Idea : By traversing bottom-up the path decomposition, by using 4 colors + $\perp$, the clique-width operations can add, *one by one*, new vertices (using $\oplus$ **i**) and new edges (using $Add_{a,b}$ or $\overrightarrow{Add}_{a,b}$).

$\perp$ is for “terminated vertices”.

For tree-width : $cwd(G) \leq 2^{2.twd(G) + 1}$

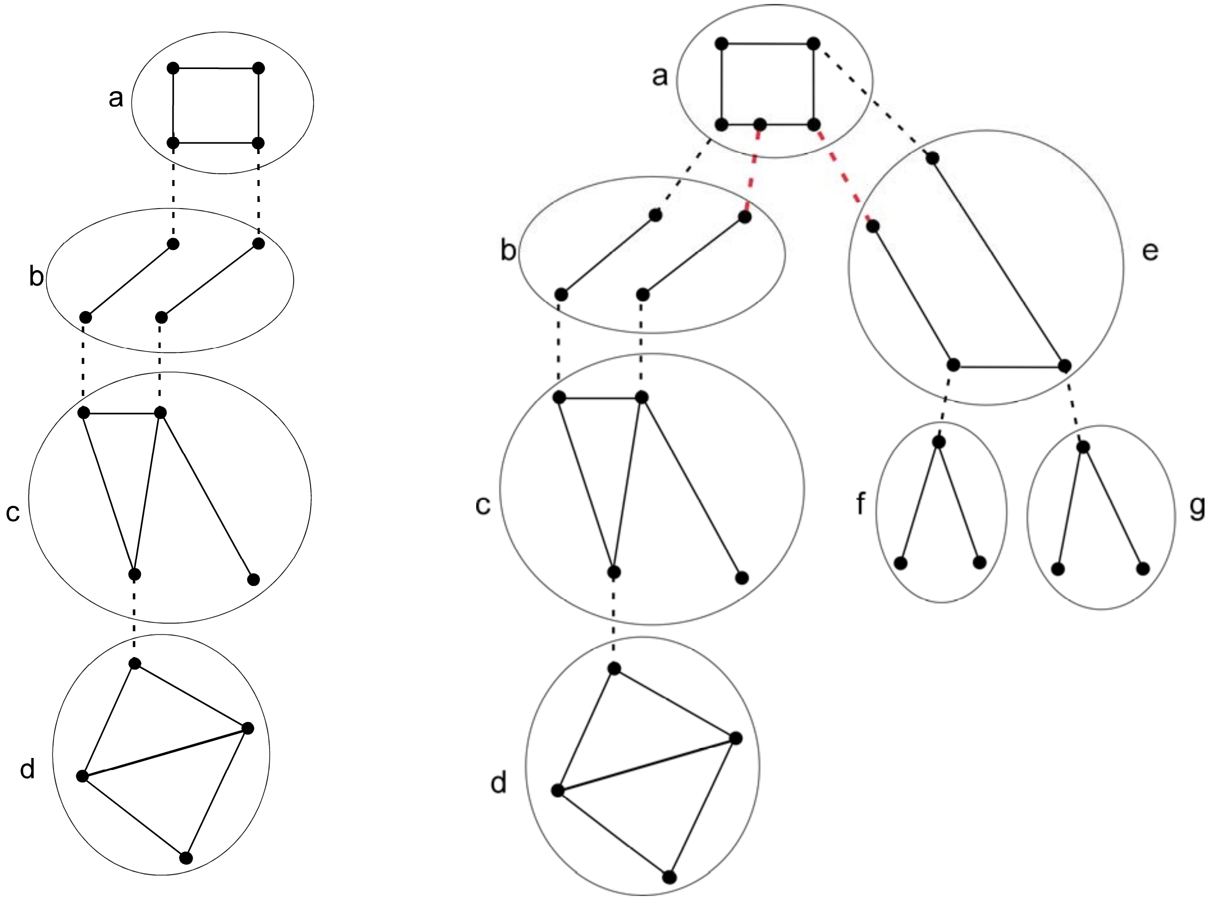


Because of vertex **3**, common to two "son boxes", of the tree-dec, the previous method does not work. (It does not allow fusion of vertices).

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

($2^{2k} – 1$ for directed graphs).

For special tree-width, as for path-width: cwd(G) $\leq$ sptwd(G)+2



The red dotted edges are not incident.

Two “brother” boxes (*b, e*) are disjoint.

This is the characteristic property of *special tree-decompositions*

Why special tree-width is interesting for model-checking of MS_2 properties ?

1) Because cwd(G) $\leq$ sptwd(G)+2. We can use the operations that define clique-width *and* we have a bijection between vertices + edges and the occurrences in special terms of the constants **a** (for vertices) and of the operations *Adda,b* (for edges);

(no incidence graph needed).

2) The automaton for the atomic formula *inc*(X,Y) that means: X consists of one edge incident with the unique element of Y has (only) k+3 states for special tree-width k, *and*

the automaton for *edg*(X,Y) that means : the unique vertex of X is adjacent to the unique element of Y has (only) k^2 +k+3 states.

However : The *parsing* problem is open :

Can one find an $O(n^{g(k)})$ algorithm:

- that reports that the input graph G (with n vertices) has special tree-width more than k or

- that outputs a special tree-decomposition witnessing special tree-width $\leq f(k)$ for G (for a fixed function f hopefully not exponential).

Note: we can use the algorithms producing path-decompositions.

Constructions of automata for "clique-width" terms.

We fix k the number of vertex labels, hence the bound on clique-width.

F = the corresponding set : **a** , $\emptyset$, $\oplus$, $Add_{a,b}$, $Relab_{a \longrightarrow b}$

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

Vertices(G(t)) = the occurrences of constant symbols in t not = $\emptyset$

Terms t are equipped (giving $t_*(V_1,\ldots,V_n)$) (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 and $\forall$).

From F and φ we construct inductively (on φ) a finite (bottom-up) deterministic automaton $A(\varphi(X_1,\ldots,X_n))$ that recognizes :

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

Atomic formulas : $edg(X_1,X_2)$ for directed edges

The automaton $A(edg(X_1,X_2))$ has $N(k) = k^2+k+3$ states.

Vertex labels are from a set C of k labels.

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

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

Meanings of states (at node u in 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)

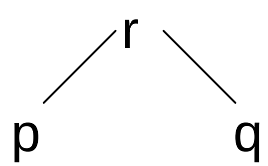
Error : all other cases

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 ⊕:



If $p = 0$ then $r := q$

If $q = 0$ then $r := p$

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

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

Otherwise $r := \text{Error}$

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

$$\begin{array}{c} r \\ | \\ p \end{array}$$

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 $c \neq a$ and $c \neq b$ then $r := bc$

If $p = ca$ where $c \neq a$ and $c \neq b$ then $r := cb$

If $p =$ Error or 0 or Ok or $c(i)$ or cd or dc where $c \neq a$

then $r := p$

Other atomic or basic formulas:

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

Card $_{\mathbf{p,q}}$ (X_1) : cardinality of X_1 is p modulo q,

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

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

Basic and useful graph properties

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	$2^{O(p.p.k.k)}$	$2^{O(k.k)}$	$2^{2^{O(k)}}$

For Connectedness, the minimal automaton has more than $2^{2^{k/2}}$ states.

Automata for the model-checking of MS_2 formulas

We need :

1) Terms to represent graphs, over appropriate operations.

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

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.

2.2 : For special terms : each edge is produced by a unique occurrence of $Add_{a,b}$. This gives what we want for graphs of bounded *special tree-width* (but not for bounded tree-width).

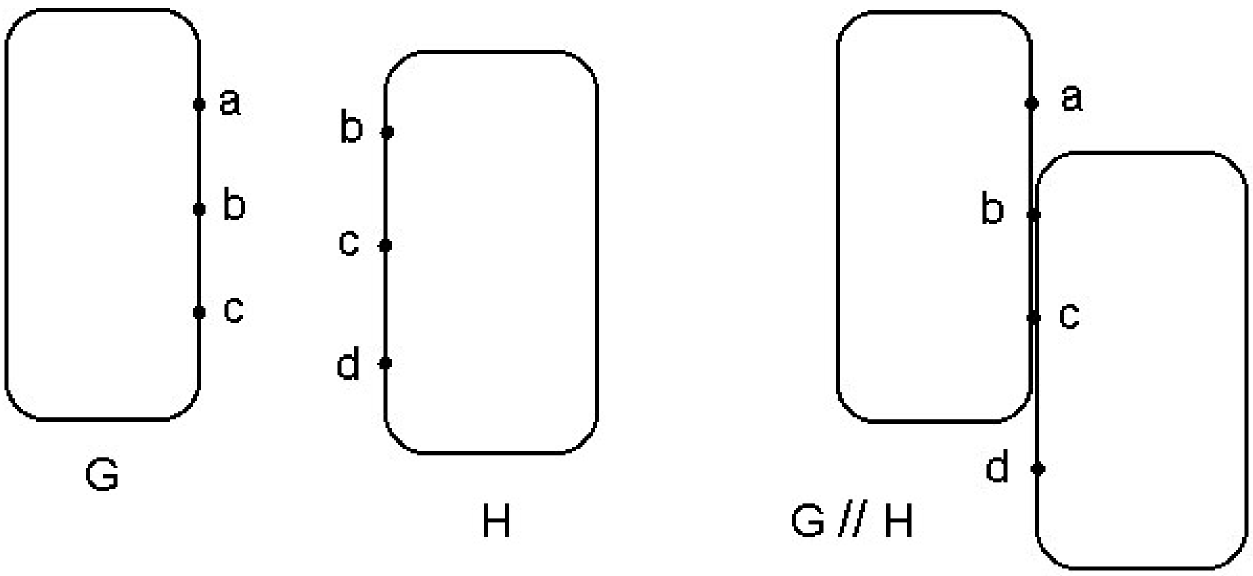
2.3 : Case of terms characterizing tree-width.

Graph operations characterizing tree-width

Graphs have distinguished vertices called *sources,* (or terminals or boundary vertices) pointed to by source labels from a finite set : {*a, b, ..., d*}.

Binary operation : *Parallel composition*

G // H is the disjoint union of G and H and sources with same label are fused. (If G and H are not disjoint, one first makes a copy of H disjoint from G).



Unary operations :

Forget a source label

$Forget_a(G)$ is G without a-source: the source is no longer distinguished ; (it is made "internal").

Source renaming :

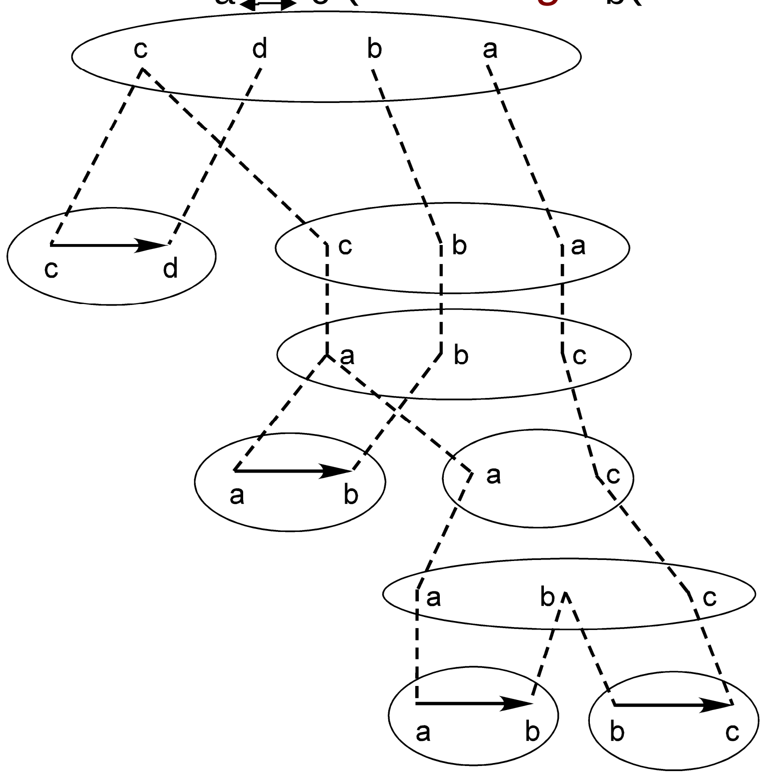
$Ren_{a \longleftrightarrow b}(G)$ exchanges source labels a and b (replaces a by b if b is not the label of a source)

Nullary operations denote *elementary graphs* : the connected graphs with at most one edge.

Proposition: A graph has tree-width $\leq$ k if and only if it can be constructed from basic graphs with $\leq$ k+1 labels by using the operations // , *Ren*$_{a \leftrightarrow b}$ and *Forget*$_a$.

From an algebraic expression to a tree-decomposition

Example : cd // *Ren*$_{a \leftrightarrow c}$ (ab // *Forget*$_b$(ab // bc)) (Constant ab denotes an edge from a to b)



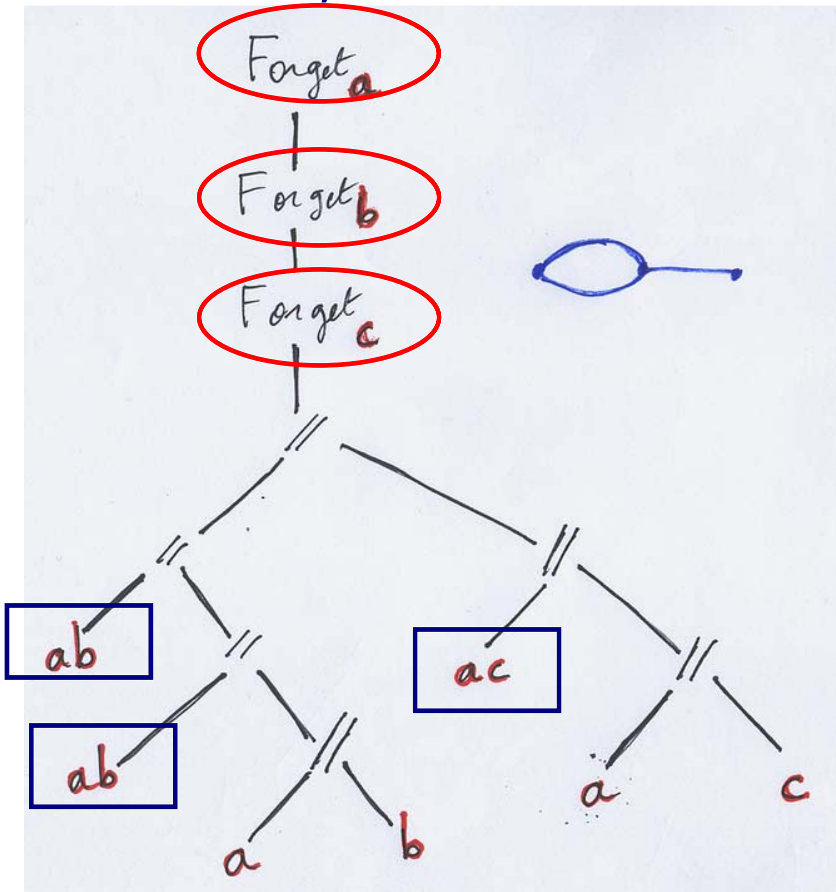
The tree-decomposition associated with this term.

Automata for these terms.

The difficulty : to have a bijection between occurrences in the term and the vertices and edges of the graph. *Two possibilities* :

(1) Vertices are in bijection with the occurrences of *Forget*$_a$. The edges are at the leaves of the syntactic tree, *below* the nodes representing their ends. The automaton for *edg(X,Y)* has $2^{\Theta(k.k)}$ states. (k^2 for sptwd)

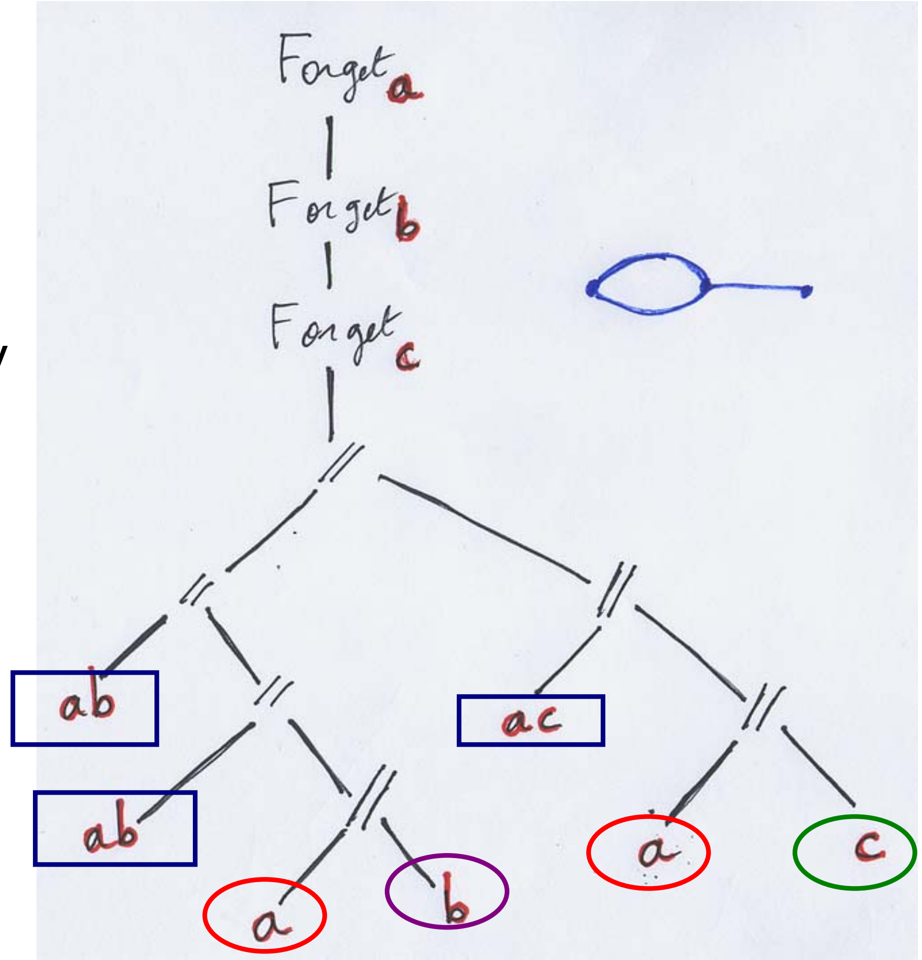
Too bad for a basic property!



(2) Vertices are at the leaves,

the edges are at nodes *close to* those representing their ends.

Because of **//** which fuses some vertices each vertex is represented by several leaves: see



Equality of vertices is then an equivalence relation $\simeq$ on leaves.

Hence:

there exists a set of vertices X such that …

is expressed by:

there exists a set of leaves X, saturated for $\simeq$ such that …

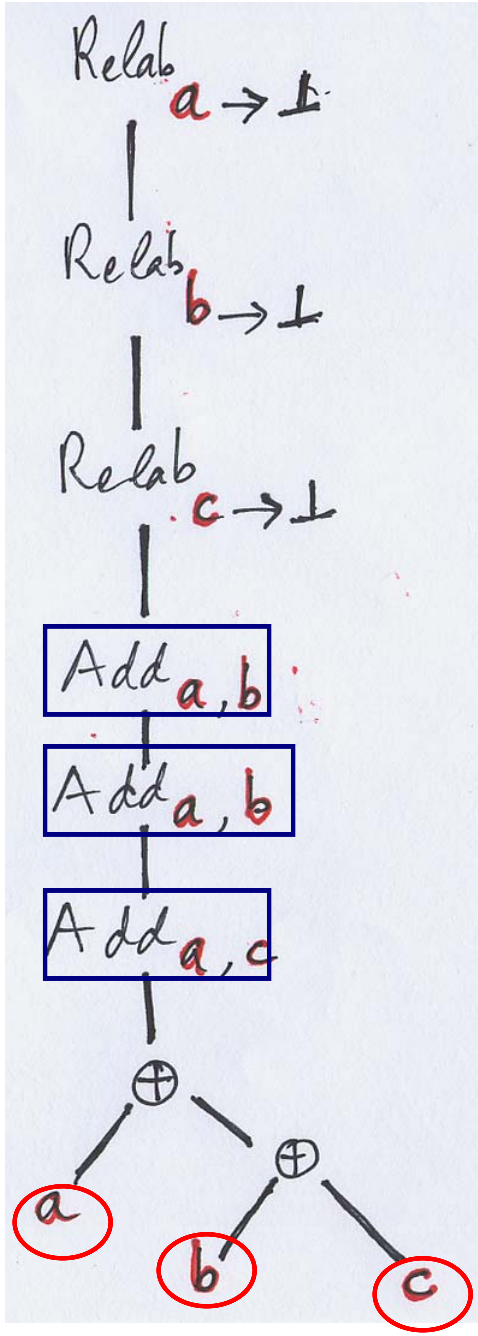
Same exponential blow up. For representing special tree-decompositions, **//** is *not* needed. This drawback disappears, solution (2) is then OK .

The special term for the same graph

Nodes representing the vertices

Nodes representing the edges

The two nodes with *$Add_{a,b}$* represent two different edges. We use the "multigraph interpretation" of "clique-width" terms.



Conclusion

Special tree-width is less powerful than tree-width, but the constructions of automata are simpler.

The parsing problem is open.

In many cases (in particular bounded degree, but certainly others) special tree-width is linearly bounded in tree-width.