Special tree-width and the verification of monadic second-order graph properties with edge quantifications

Reference : B.C. : On the model-checking of monadic second-order formulas with edge set quantifications. *Discrete Applied Maths*, 2012

Edge quantifications

MSO formulas

MSO_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

Tree-width, path-width and clique-width

For G directed or undirected :

$$\mathrm{cwd}(G) < 2^{\,2.\mathrm{twd}(G) + 1}$$

No polynomial bound : $\mathrm{cwd}(G) \leq \mathit{poly}(\mathrm{twd}(G))$

In both cases : $\mathrm{cwd}(G) \leq \mathrm{pwd}(G) + 2.$ Why ??

pwd = path-width = tree-width with paths instead of trees

FPT model-checking algorithms

For MSO properties, the parameter is clique-width.

The case of MSO_2 formulas reduces to that of MSO ones:

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

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

2) every MSO_2 property of G is an MSO property of Inc(G).

To avoid the $2^{O(k)}$ blow-up, one could build fly-automata running on terms representing tree-decompositions.

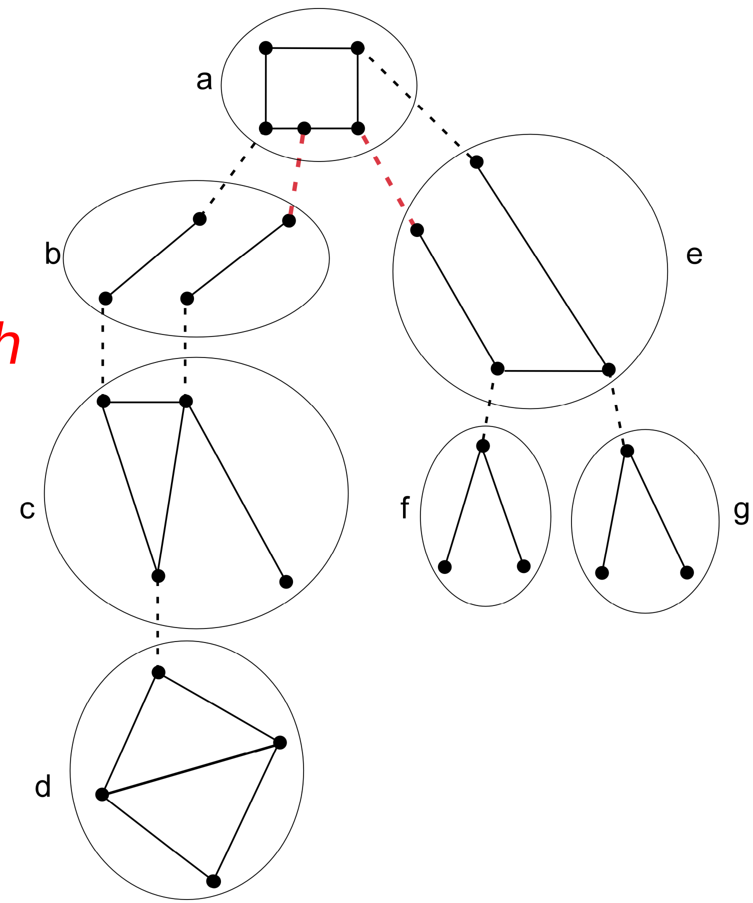
Problem: Because of // (parallel composition) a vertex may correspond to several positions in the term. This yields also an exponential blow-up in automata sizes. How to avoid // ?

Special tree-width

Definition: *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 exponential blow-up).

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

Properties of special tree-width (sptwd)

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

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

whereas cwd(G) $\leq 2^{2.\mathrm{twd(G)} + 1}$ (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*.

Terms that characterize special tree-width;

Definition: *Special terms*

They use the graph operations that define clique-width for *graphs with multiple edges* (Key point : no “vertex fusion” is needed)

1) The set 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 $\leq$ 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. Similar definitions for directed graphs.

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

Proposition: (1) G has *special tree-width* $\leq$ k $\Leftrightarrow$ it is defined by a *special term* using $\leq$ k + 2 labels (including the particular label $\perp$)

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

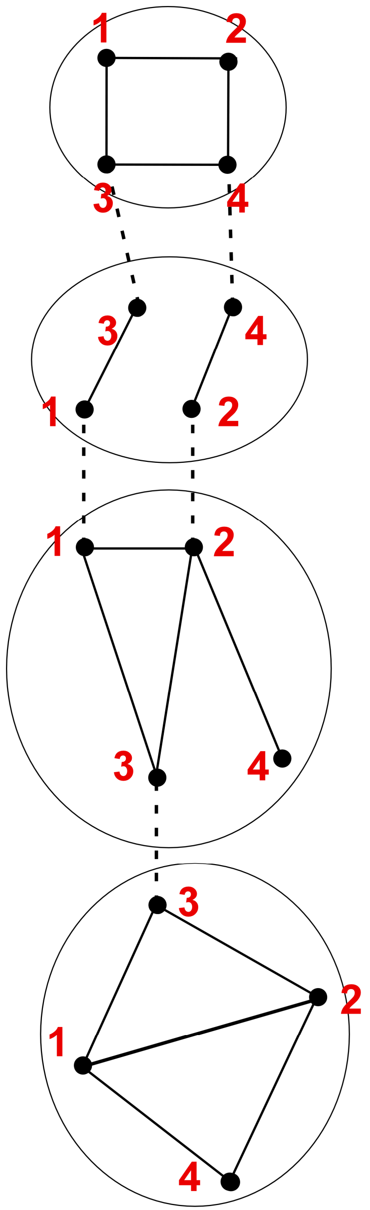
We will compare:

path-width and clique-width,

tree-width and clique-width,

special tree-width and clique-width

Comparing path-width and clique-width :

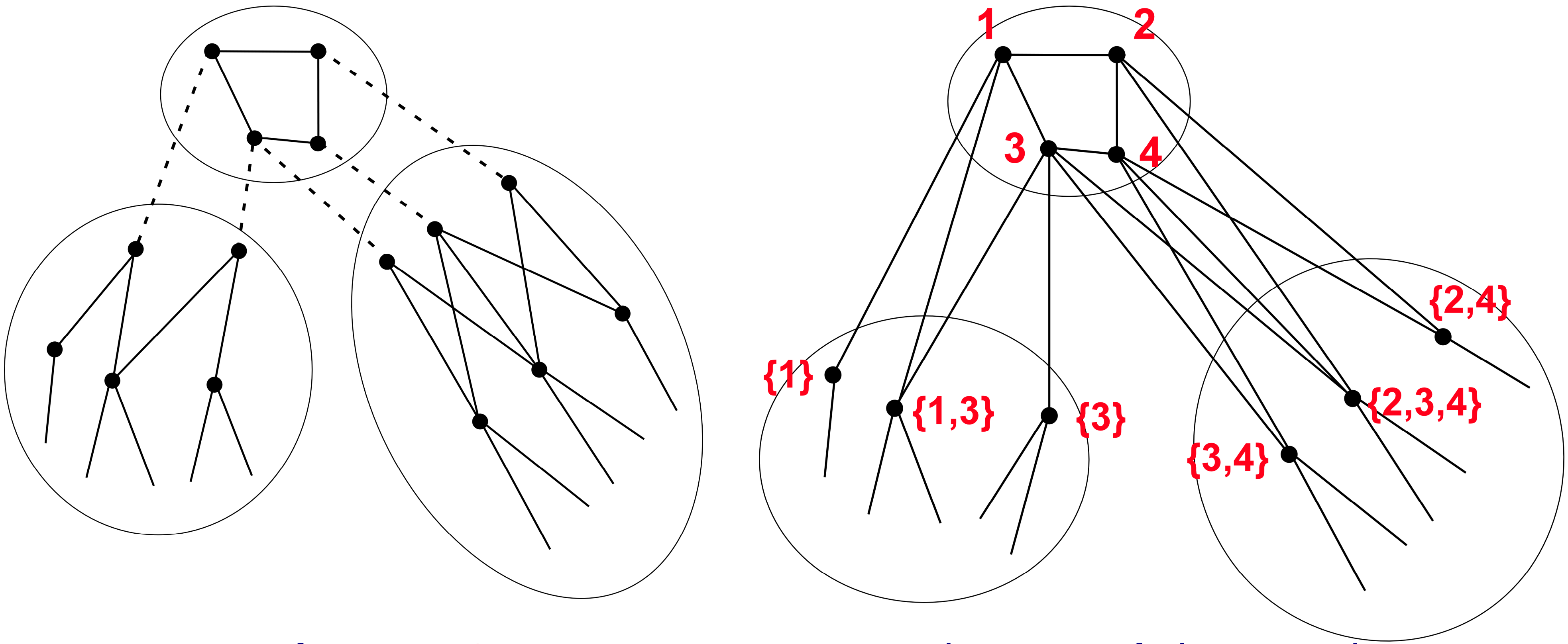


$$cwd(G) \leq pwd(G) + 2$$

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

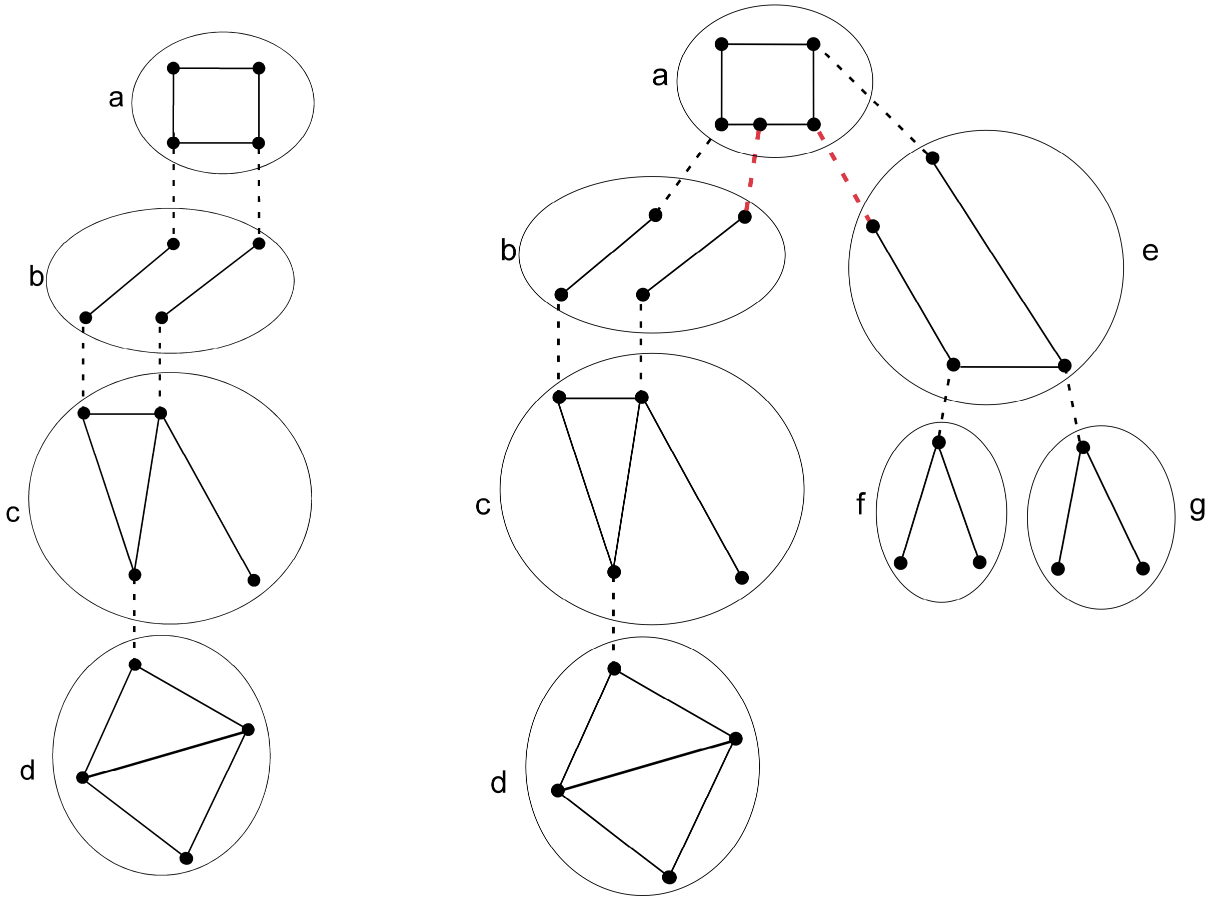


Because of vertex **3**, common to two "son boxes", of the tree-dec, the previous method does not work. (It does not allow fusion of vertices).

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*

Special tree-width is interesting for model-checking of MSO_2 properties (as we will see) but 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

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

Note: We can use the algorithms that produce path-decompositions

Automata for the model-checking of MSO_2 formulas

We need :

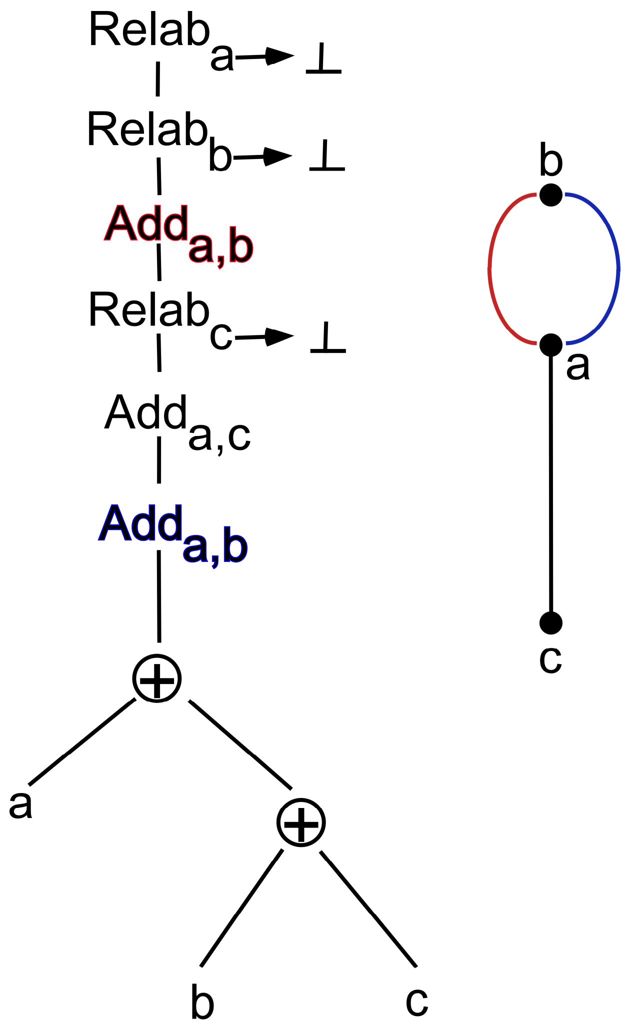
1) Terms to represent graphs, over appropriate operations.

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*.

Using 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

The automata for edg(X,Y) and inc(X,Y) (incidence) have $O(k^2)$ and $O(k)$ states respectively for *sptwd* at most k.

2.3 : Case of terms characterizing tree-width

First idea : make them into "clique-width terms" for the *incidence graph*. But:

clique-width $\leq$ $2^{O(\text{tree-width})}$ → too large automata.

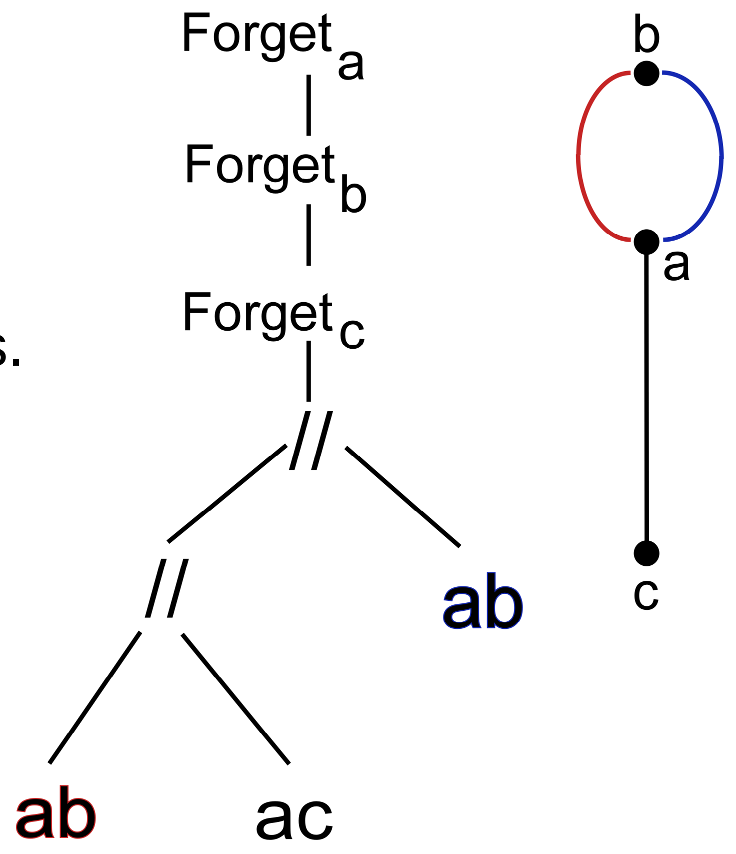
Second idea : handling them "directly", as for "clique-width terms"

The difficulty is to have a bijection between nodes in the term and the vertices and edges of the graph.

First possibility

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 (compare with $O(k^2)$ for sptwd).

Too bad for a basic property !

Second possibility

Vertices are at the leaves,

the edges are at nodes *close to*

those representing their ends.

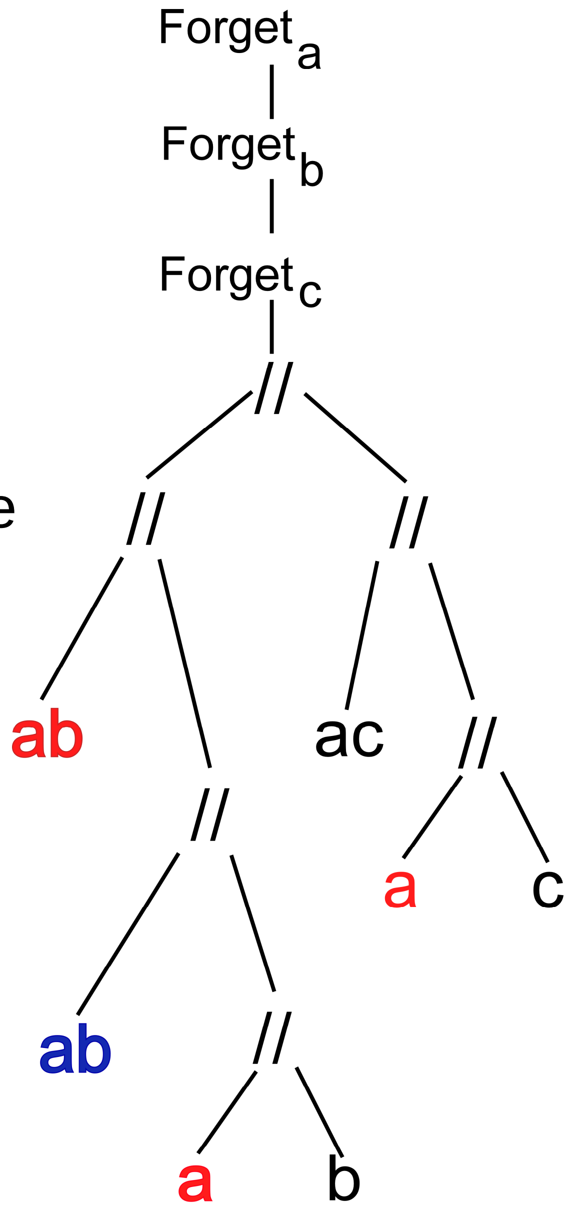
Because of **//** which fuses some

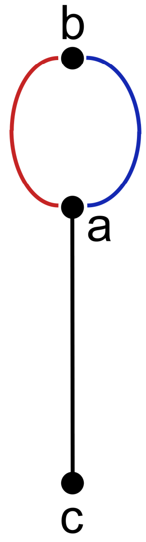
vertices, each vertex is

represented by several leaves.

On the figure, vertex a is

represented by two leaves.





Equality of vertices is 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 as with the second possibility.

The responsible is // (that is not needed for representing special tree-decompositions).

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) special tree-width is linearly bounded in tree-width.