Monadic second-order queries on graphs: Vertex labelling for efficient evaluation and linear delay enumeration

Bruno Courcelle

Université Bordeaux 1, LaBRI

This is part of a global research project on

Graph Decomposition and Logic

Four independent research directions associating Monadic Second-order logic *and graph decompositions are now intimately related :*

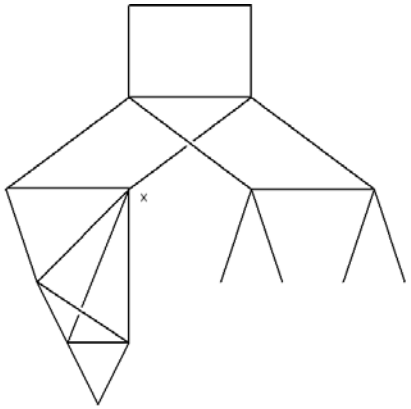
1. Polynomial algorithms for NP-complete and other hard problems on particular classes of graphs, and especially hierarchically structured ones : series-parallel graphs, cographs, partial k-trees, graphs or hypergraphs of tree-width < k, graphs of clique-width < k. Related algorithmic questions
2. Excluded minors and related notions of forbidden configurations (matroid minors, vertex-minors).
3. Decidability of Monadic Second-Order logic on classes of finite graphs, and on infinite graphs.
4. Extension to graphs and hypergraphs of the main concepts of Formal Language Theory : grammars, recognizability, transductions, decidability questions.

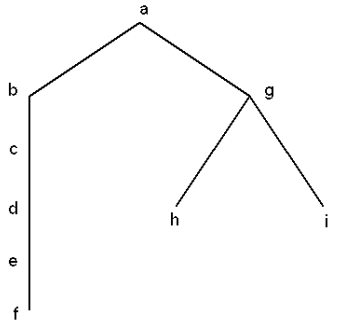
Summary

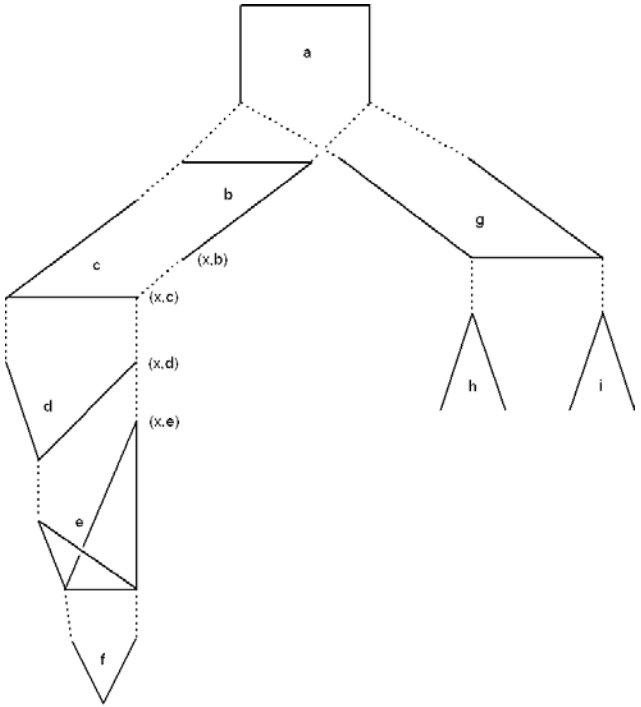
1. Tree-width, clique-width, graph algebras
2. Monadic second-order logic, fixed-parameter tractable algorithms
3. Labelling for efficient querying : Networks with obstacles (Courcelle-Twigg, 2007)
4. Labelling for efficient querying : A general method (Courcelle-Vanicat, 2003).
5. Linear delay enumeration (Courcelle, 2006, to appear, 2008 ?).

The graph algebra HR and tree-width

Tree-width : Tree-decomposition of width k : k+1 = maximum size of a box







Tree-width : twd(G) = minimum width of a tree-decomposition

* Trees have tree-width 1,

* K_n has tree-width n-1

* the n x n grid has tree-width = n

Planar graphs and cliques have unbounded tree-width.

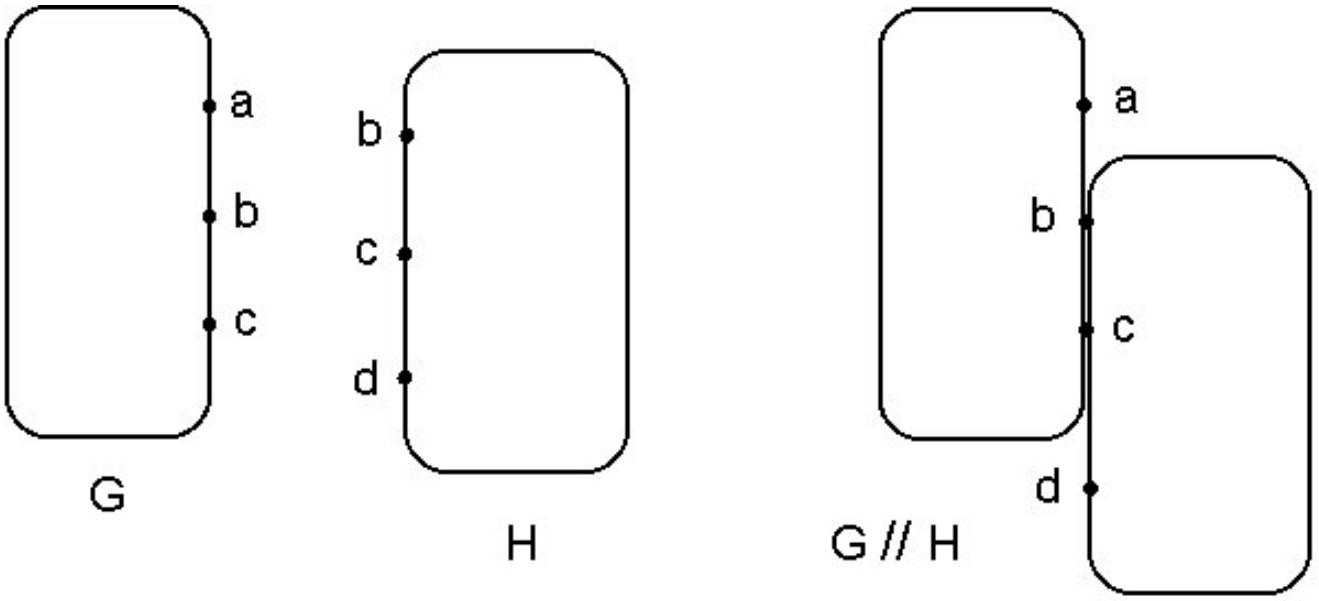
Outerplanar graphs have tree-width at most 2.

A characterization in terms of HR operations : (**H**yperedge **R**eplacement hypergraph grammars ; associated complexity measure : tree-width)

Graphs have distinguished vertices called *sources,* pointed to by labels from a set of size k : {*a, b, c, ..., h*}.

Binary operation(s) : *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 :

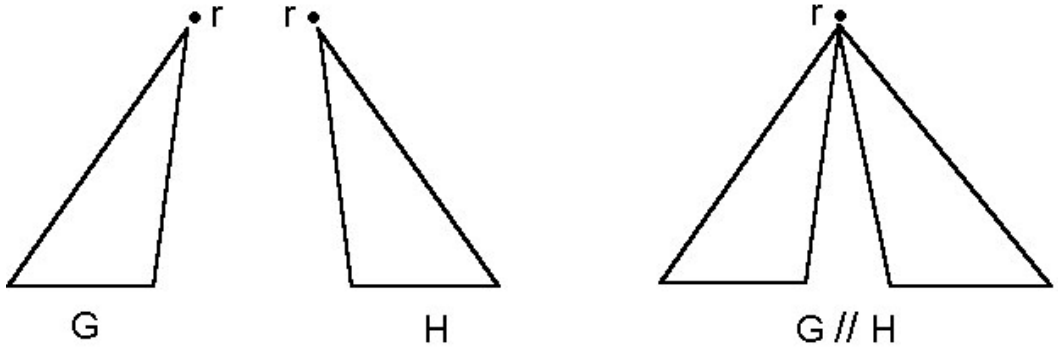
$Ren_{a,b}$(G) exchanges source names *a* and *b*

(replaces *a* by *b* if *b* is not the name of a source)

Nullary operations denote *basic graphs* : the connected graphs with at most one edge. *For dealing with hypergraphs one takes more nullary symbols for denoting hyperedges.*

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 // , $Ren_{a,b}$ and $Forget_a$.

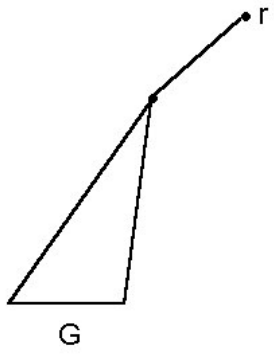
Example : Trees are of tree-width 1, constructed with two source labels, *r* (root) and *n* (new root): Fusion of two trees at their roots :



Extension of a tree by parallel composition with a new edge, forgetting the old root, making the "new root" as current root :

E = *r* ●——● *n*

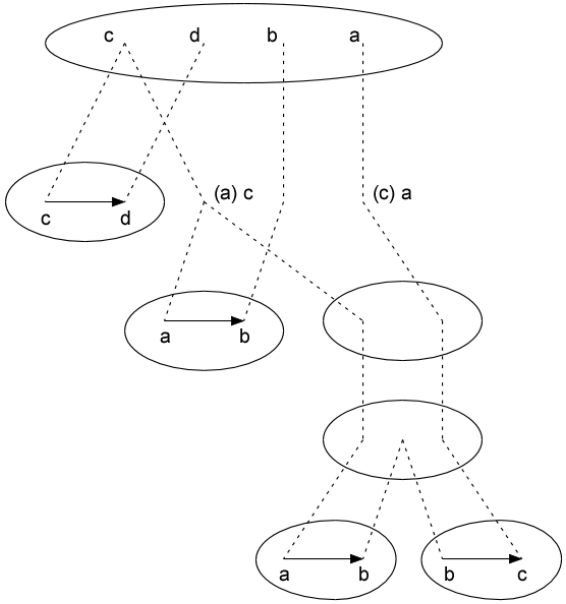
$Ren_{n,r}(Forget_{r}(G // E))$



From an algebraic expression to a tree-decomposition

Example : cd // $Ren_{a,c}$ (ab // $Forget_b$(ab // bc))

Constant ab denotes a directed edge from a to b.



The tree-decomposition associated with this term.

VR operations and clique-width (m-clique-width)

(**V**ertex **R**eplacement graph grammars)

Graphs are simple, directed or not.

One uses k labels : *a , b , c, ..., h*.

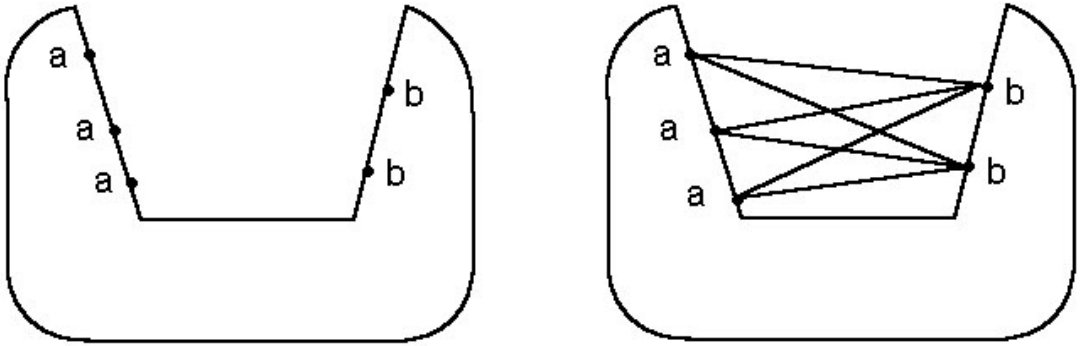
Each vertex has one and only one label ; (none or several)

a label p may label several vertices, called the *p-ports*.

One binary operation: *disjoint union* : $\oplus$

Unary operations: Edge addition denoted by *Add-edg*$_{a,b}$

Add-edg$_{a,b}$(G) is G augmented with (un)directed edges from every *a*-port to every b-port.



G *Add-edg*$_{a,b}$(G)

Vertex relabellings :

$Relab_{a,b}$(G) is G with every vertex labelled by a relabelled into b

Basic graphs are those with a single vertex.

Definition: A graph G has clique-width $\leq k$ $\Leftrightarrow$ it can be constructed from basic graphs by means of k labels and the operations $\oplus$, $Add\text{-}edg_{a,b}$ and $Relab_{a,b}$

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

Proposition : (1) If a set of simple graphs has bounded tree-width, it has bounded clique-width, but not *vice-versa*.

(2) Unlike tree-width, clique-width is sensible to edge directions : Cliques have clique-width 2, tournaments have unbounded clique-width.

(3) (a) Deciding "Clique-width $\leq$ 3" is a polynomial problem. (Habib et al.)

(b) The complexity (polynomial or NP-complete) of "Clique-width = 4" is unknown.

(c) It is NP-complete to decide for given k and G if cwd(G) $\leq$ k. (Fellows et al.)

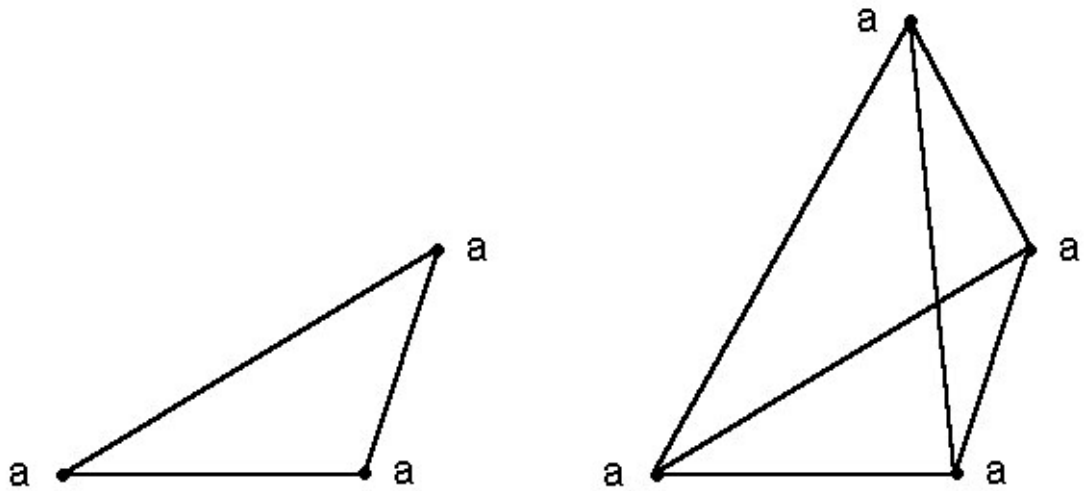
(d) There exists a cubic approximation algorithm that for given k and G answers (correctly) :

either that cwd(G) >k,

or produces a clique-width algebraic term using 2^{24k} labels. (Oum)

This yields Fixed Parameter Tractable algorithms for many hard problems.

Example : Cliques have clique-width 2.



K_n is defined by t_n where t_{n+1} = $Relab_{b,a}(\ Add\text{-}edg_{a,b}(t_n \oplus \mathbf{b}))$

Another example : Cographs

Cographs are generated by $\oplus$ and $\otimes$ defined by :

$G \otimes H$ = $Relab_{b,a}\ (\ Add\text{-}edg_{a,b}\ (G \oplus Relab_{a,b}(H))$

= $G \oplus H$ with "all edges" between G and H.

Algorithmic applications

Fixed parameter tractability results

Theorem (B.C.) : 1) For graphs of clique-width $\leq$ k ,

each monadic second-order property, (ex. 3-colorability),

each monadic second-order optimization function, (ex. distance),

each monadic second-order counting function, (ex. # of paths)

is evaluable :

in linear time on graphs given by a term,

in time $O(n^3)$ otherwise (by S. Oum, 2005).

2) All this is possible in linear time on graphs of tree-width $\leq$ k, for each fixed k (by Bodlaender, 1996)

Monadic Second-Order (MS) Logic

= First-order logic on power-set structures

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

MS properties : transitive closure, properties of paths, connectivity,

planarity (via Kuratowski, uses connectivity), k-colorability.

Examples of formulas for $G = < V_G , edg_G(.,.) >$, undirected

Non connectivity :

$\exists X \, (\exists x \in X \ \& \ \exists y \notin X \ \& \ \forall u,v \, (u \in X \ \& \ edg(u,v) \Rightarrow v \in X) \ )$

2-colorability (i.e. G is bipartite) :

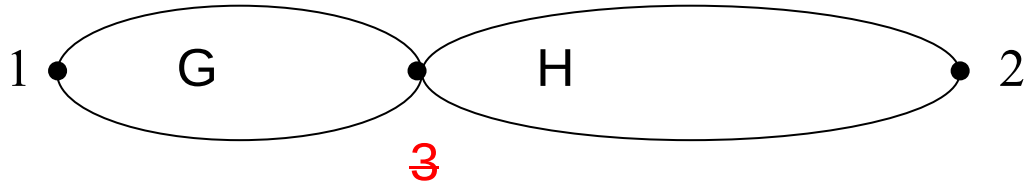
$\exists X \, (\forall u,v \, (u \in X \ \& \ edg(u,v) \Rightarrow v \notin X) \ \& \forall u,v \, (u \notin X \ \& \ edg(u,v) \Rightarrow v \in X) \ )$

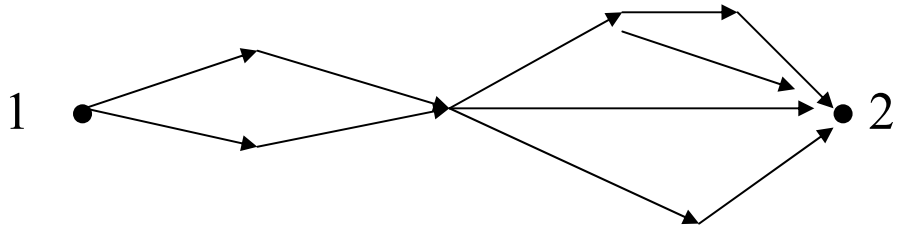
Inductive computations

Example : *Series-parallel graphs*, defined as graphs with sources 1 and 2, generated from e = 1 ⟶ 2 and the operations // (parallel-composition) and *series-composition* defined from other operations by :

$$G \bullet H = Forget_3(Ren_{2,3}(G) \,//\, Ren_{1,3}(H))$$

Example :





Inductive computation : Test of 2-colorability

1) Not all series-parallel graphs are 2-colorable (see K_3)

2) G, H 2-colorable does not imply that G//H is 2-colorable (because $K_3 = P_3 // e$).

3) One can check 2-colorability with 2 auxiliary properties :

Same(G) = G is 2-colorable with sources of the same color,

Diff(G) = G is 2-colorable with sources of different colors

by using rules : Diff(e) = True ; Same(e) = False

$\text{Same}(G//H) \Leftrightarrow \text{Same}(G) \wedge \text{Same}(H)$

$\text{Diff}(G//H) \Leftrightarrow \text{Diff}(G) \wedge \text{Diff}(H)$

$\text{Same}(G \bullet H) \Leftrightarrow (\text{Same}(G) \wedge \text{Same}(H)) \vee (\text{Diff}(G) \wedge \text{Diff}(H))$

$\text{Diff}(G \bullet H) \Leftrightarrow (\text{Same}(G) \wedge \text{Diff}(H)) \vee (\text{Diff}(G) \wedge \text{Same}(H))$

We can compute for every SP-term t, by induction on the structure of t the pair of Boolean values (Same(Val(t)) , Diff(Val(t))).

We get the answer for G = Val(t) (the graph that is the *value* of t) regarding 2-colorability.

Important facts :

a) The simultaneous computation of m inductive properties can be implemented by a "tree" automaton working on terms t, with 2^m states. This computation takes time $O(|t|)$.

b) An inductive set of properties can be constructed (at least theoretically) from every monadic-second order formula.

c) This result extends to the computation of values (integers) defined by monadic-second order formulas.

d) Application to graphs depends on two things : *Parsing algorithms* building terms from the given graphs :

1) results by Bodlaender for constructing tree-decompositions (in linear time), whence terms representing them,

2) results by Oum and Seymour for constructing (non-optimal) terms for graphs of clique-width at most k. (Cubic time algorithm)

A language L , i.e., a set of words or terms is Monadic Second-Order (MS) definable if :

$$L = \{ t \;/\; t \models \varphi \}$$ for an MS formula φ

Theorem : A language is MS definable iff it is recognizable (by a deterministic finite automaton).

Two of the results to be presented are based on this theorem.

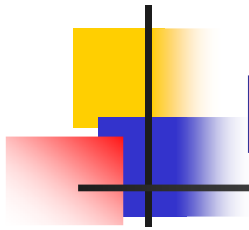
The third one (presented first) is a direct construction, NOT using MS logic and automata.

Compact Forbidden-set Routing

Bruno Courcelle
LaBRI, CNRS
Bordeaux
bruno.courcelle@labri.fr

Andrew Twigg
Cambridge University /
Thomson Research, Paris
andrew.twigg@cl.cam.ac.uk

STACS, Feburary 2007



Background: distance labelling

Distance labelling problem: given graph G, compute labels $J(x)$ for $x \in V$ s.t. given labels $J(x), J(y)$, we can compute the distance $d(x, y)$ in G.

Main results in the area:

- ★ Exact distance labeling for general graphs: $\tilde{\Theta}(n)$ bits [Peleg, Gavoille, ...]
- ★ Stretch-3 scheme for general graphs using $\tilde{O}(n^{1/2})$ bits [Thorup]
- ★ For treewidth k graphs, exact scheme using $\Theta(k \log^2 n)$ bits.
- ★ For graphs excluding a fixed minor, $\tilde{O}(1)$ bits [Thorup, Gavoille, ...]

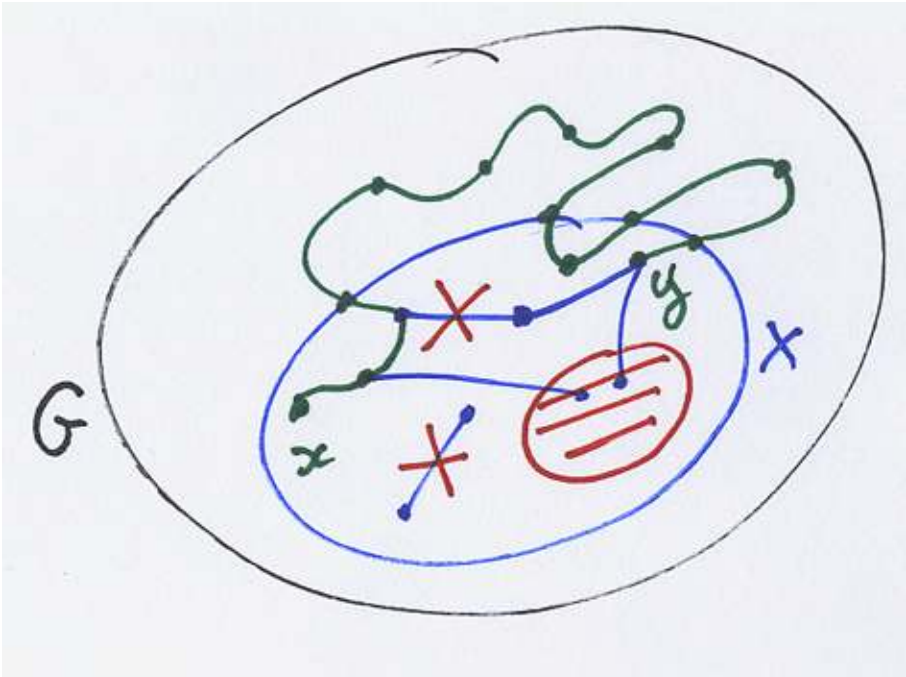
Most of these can be obtained as *compact routing* schemes: given $J(x), J(y)$, determine the *next-hop* on the path $x \to y$.

"Distance labelling with obstacles"

Forbidden-set distance labelling problem: given labels $\{J(x) : x \in X\}$ for $X \subseteq V$, what is the distance with no intermediate nodes in X?
i.e $d_{G\setminus(X\setminus\{x,y\})}(x,y)$ for $x, y \in X$?

Theorem 1 *For graphs of cliquewidth and treewidth $\leq k$, we can use labels of size $O(k^2 \log^2 n)$ bits. Only a factor k larger than pure distance labelling.*

★ WHY?? E.g. Internet routers can specify routing policies; not known if they can be satisfied using small collections of trees.

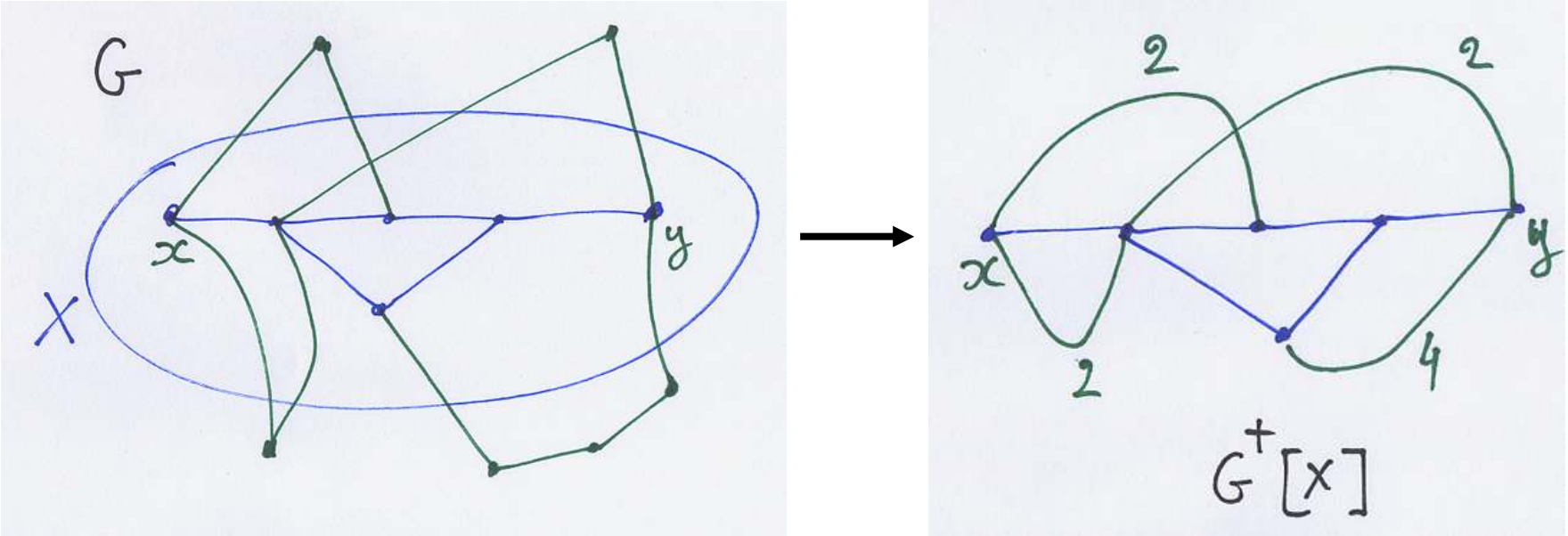


Shortcut graph $G^+[X]$

Given the labels $\{J(x) : x \in X\}$, we can compute the graph

$$G^+[X] = G[X] + \{\text{shortest detours outside } X \text{ between } x, y \in X\}.$$

Then we can answer distance queries with forbidden vertices and edges in X.



Algebraic representations: Cographs

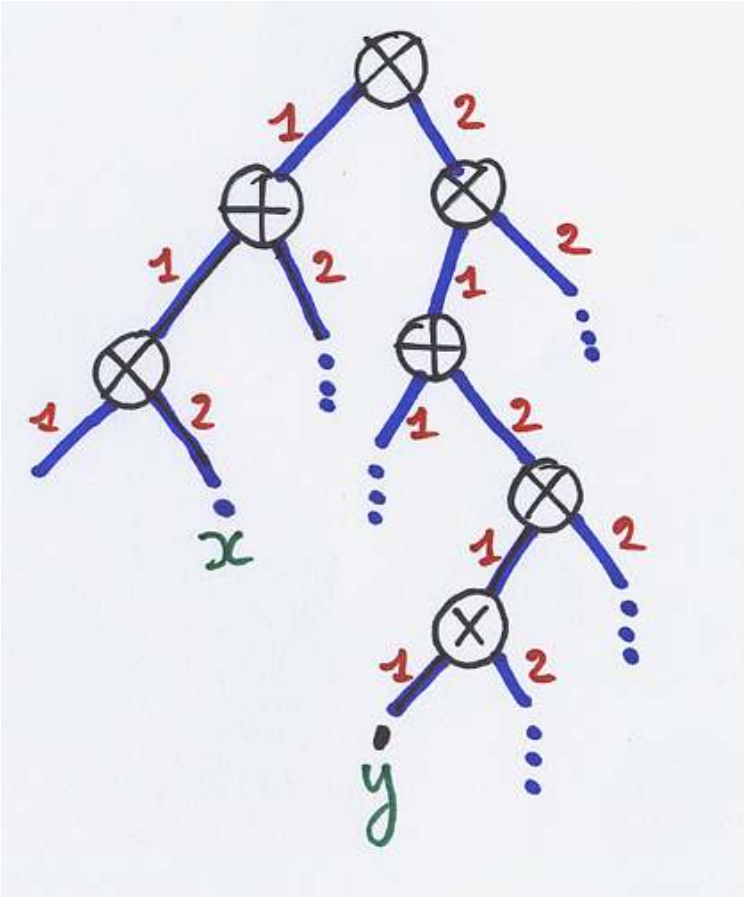
Graphs defined by terms on two binary operations:

- ★ $G \oplus H$: disjoint union of G, H
- ★ $G \otimes H = G \oplus H$ plus all edges between G, H

For example, *adjacency labels*:
$A(x) = \otimes 1 \oplus 1 \otimes 2$
$A(y) = \otimes 2 \otimes 1 \oplus 2 \otimes 1 \otimes 1$

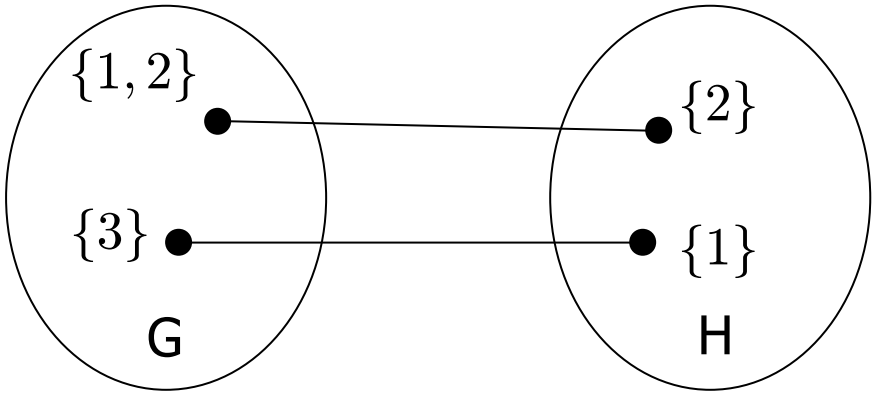
$G[X]$ can be constructed from $\{A(u) : u \in X\}$.

Fact 1 $|A(u)| \leq 2.ht(t)$
More powerful graph operations to be defined next.



Useful class of graphs: mcwd

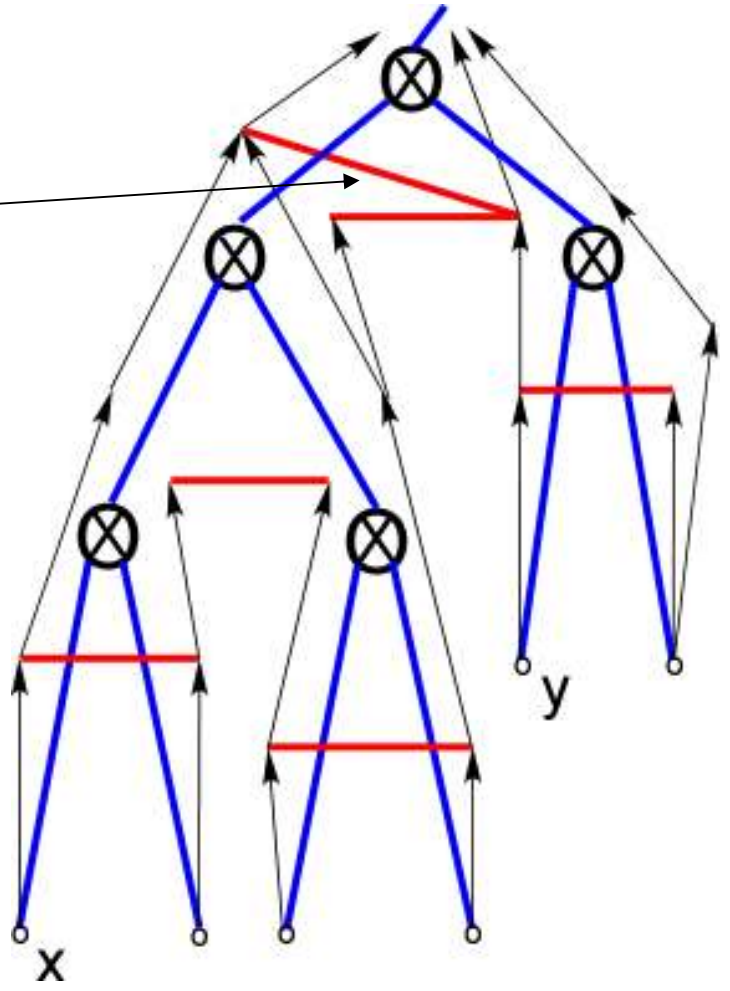
- ★ A variant of clique width; each node can have several colours, drawn from a set L of k colours.
- ★ Graph G can be represented by a term with constants (leaves) in bijection to vertices of G and internal nodes are binary operations $\otimes_{R,g,h}$:



- ★ $R = \{(1,2),(3,1)\}$ is a set of pairs of colours
- ★ No edges are created inside G, H
- ★ g, h are *independent* recolourings of G, H: $g, h : [k] \to \mathcal{P}([k])$

Representation of mcwd terms

Represents two real edges in G, including {x,y}

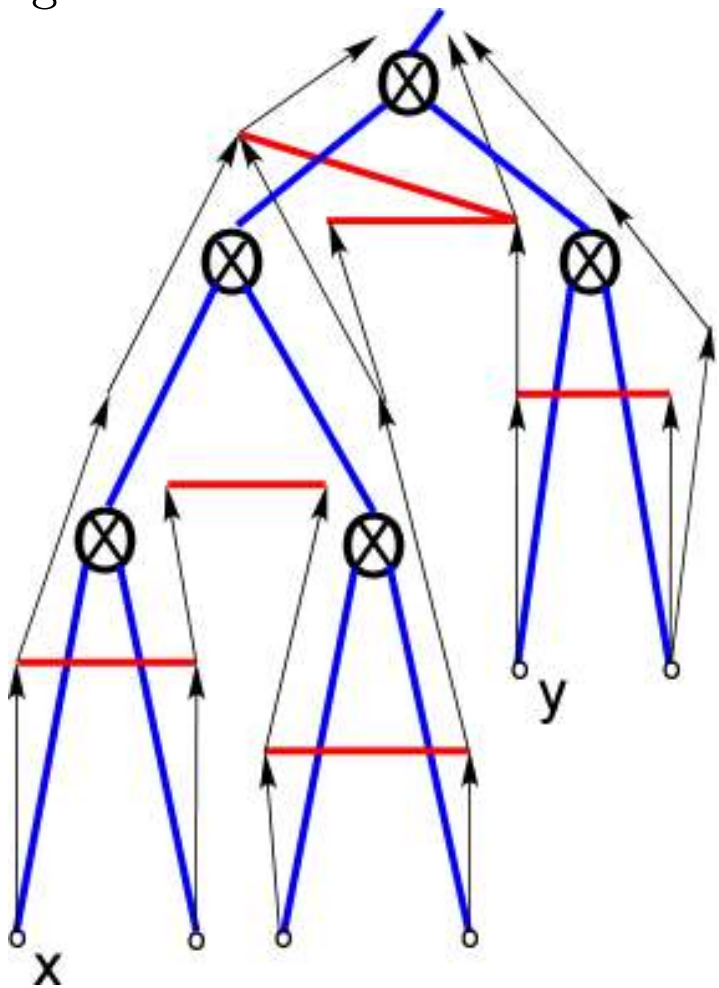


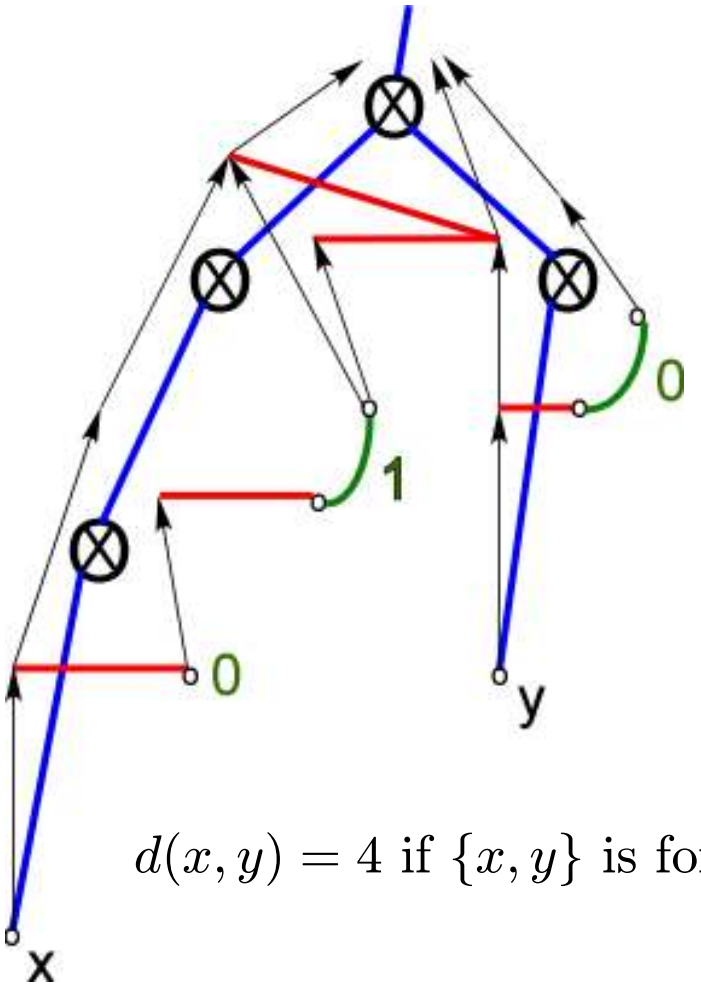
- ★ term t: $\otimes$ and blue edges
- ★ $\uparrow$: relabellings by g, h in $\otimes_{R,g,h}$ and initial labelling of vertices
- ★ red edges: pairs of labels in sets R. Each produces a set of real edges of G.

e.g. Can see that $d(x, y) = 1$
What if we delete edge $\{x, y\}$?

Forbidden edges

For shortest paths leaving a set X, we replace their corresponding subterms with *shortcut* edges in the graph representation of a term, and stored in labels assigned to vertices of G.

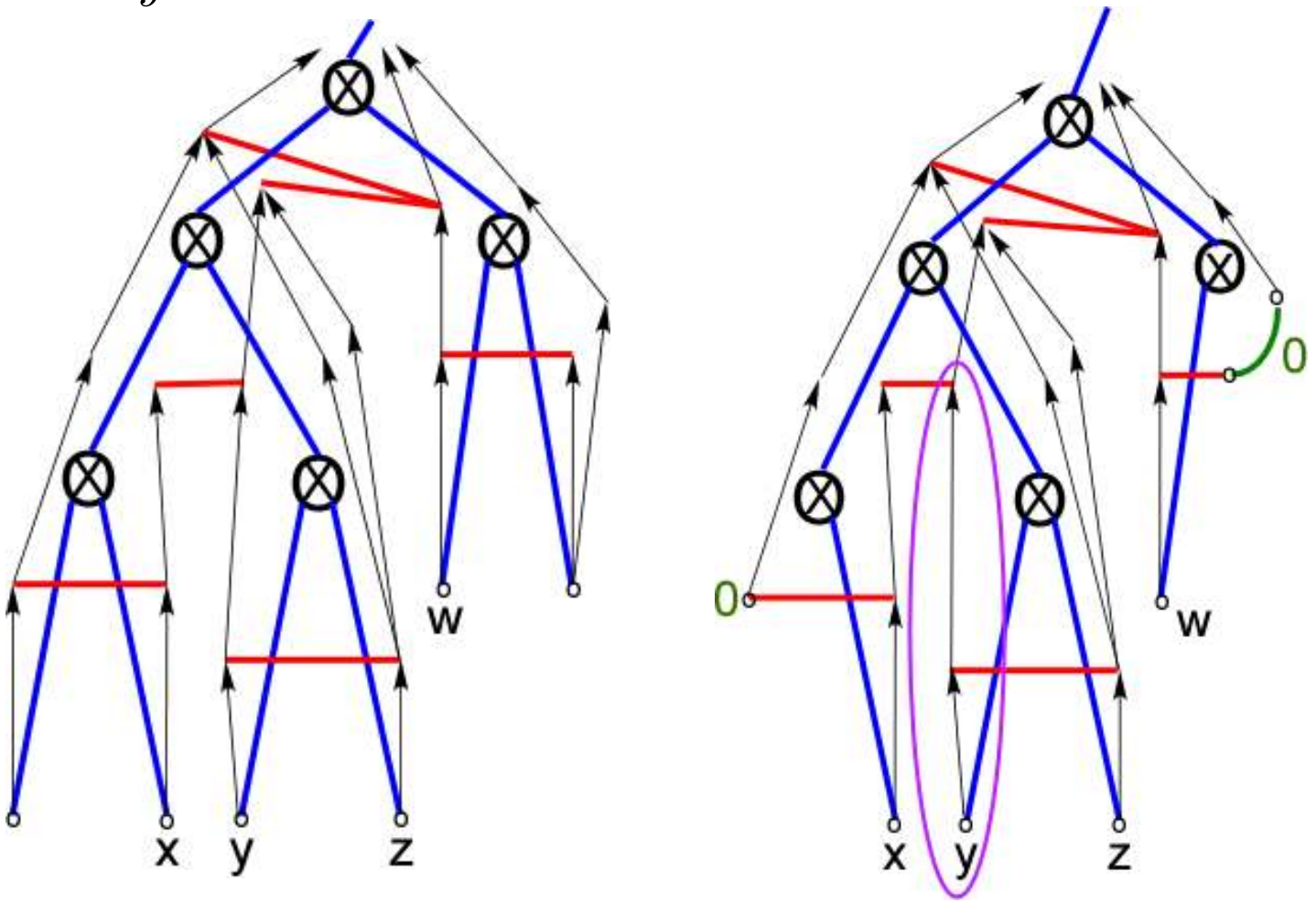


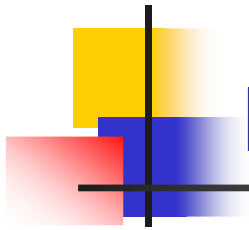


$d(x, y) = 4$ if $\{x, y\}$ is forbidden

Forbidden vertices

We can do a similar thing for deleted vertices. The ellipse touches all edges of G that are adjacent to vertex y of G. E.g. x, y and y, z are adjacent but $d(x, z) = 3$ if y is forbidden.





m-clique width

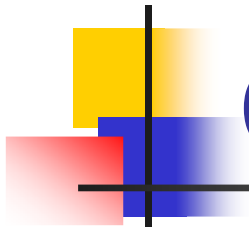
Definition 1 $mcwd(G) = \min\{k : G = val(t) \text{ for a term } t \text{ with colours} \in [k]\}$

Proposition 1

$$\begin{array}{lcl} mcwd(G) & \leq & cwd(G) \leq 2^{mcwd(G)+1} \\ mcwd(G) & \leq & twd(G)+3 \end{array}$$

The adjacency labelling scheme for cographs extends to graphs defined by mcwd terms.

But: Labels for $x \in X$ give $G[X]$ but to construct $G^+[X]$ we need knowledge of paths outside X



Construction of labels

Let G have $mcwd(G) \leq k$. The shortcut edges are represented by a $(k \times k)$ matrix of integers at each occurrence in the term tree. We enrich $A(u)$ into $J(u)$ by inserting at each position corresponding to an occurrence in the term, the associated shortcut matrix.

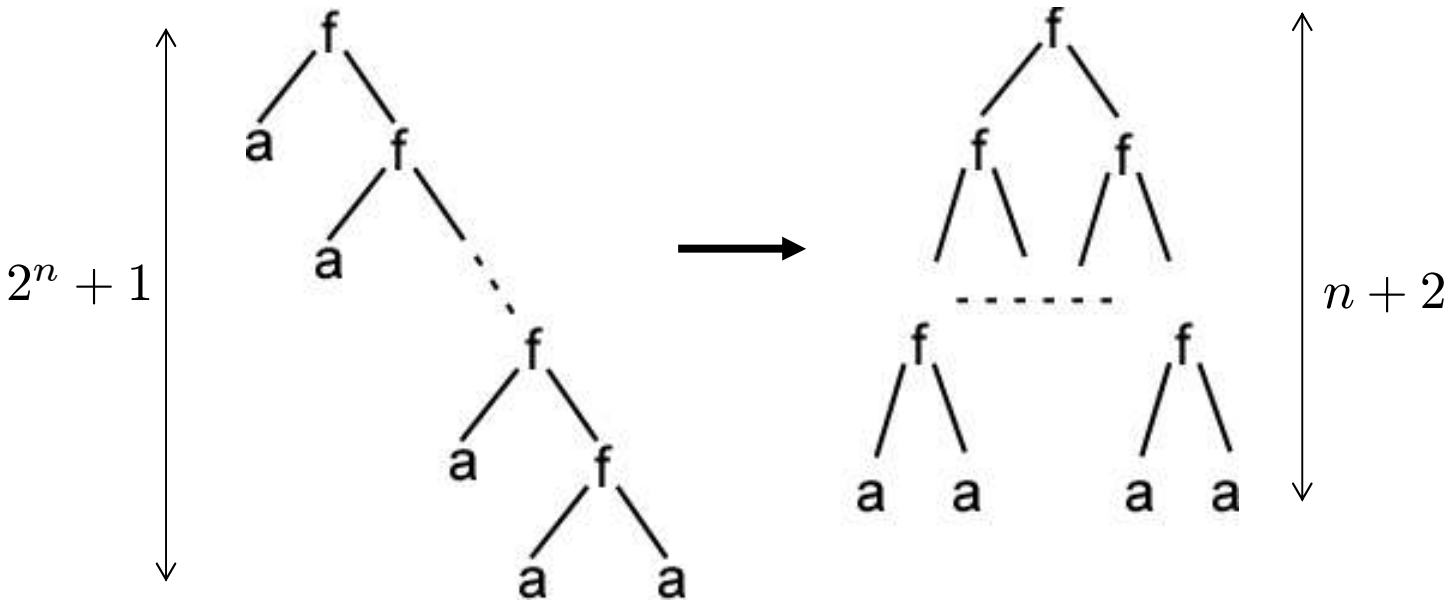
Lemma 1 *We have forbidden-set distance labels $J(x)$ of size $O(k^2 ht(t) \log n)$ bits.*

Q: How to replace the height $ht(t)$ by $\log n$?
A: Using balanced terms

We need balanced terms

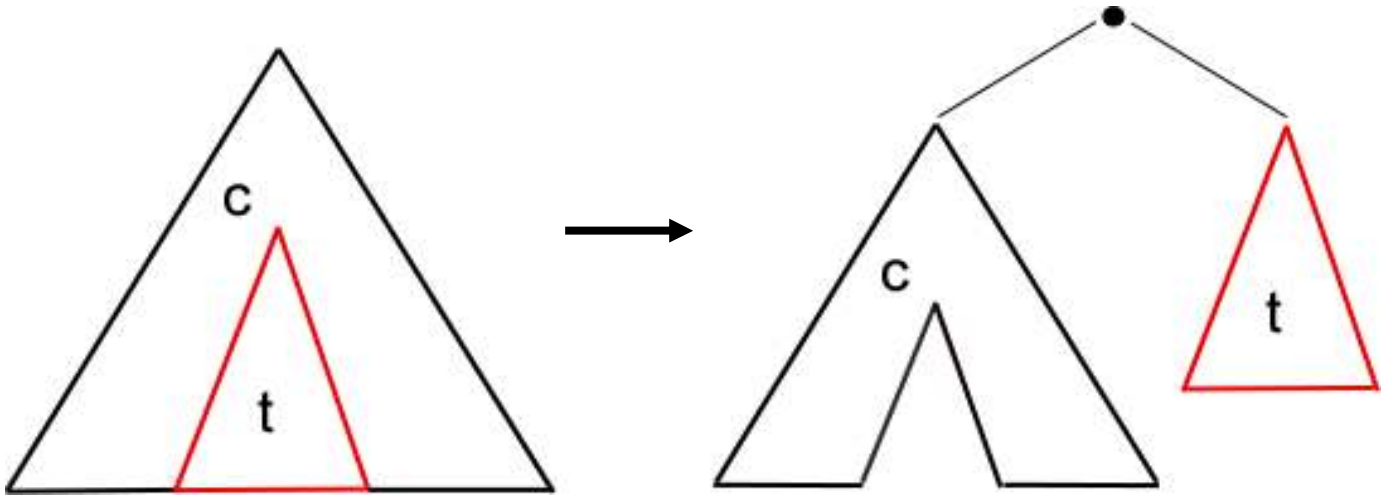
Definition 1 *A term t is a-balanced if $ht(t) \leq a \log |t|$.*

If f is associative then we can make terms balanced by simple reorganization:



Contexts and terms

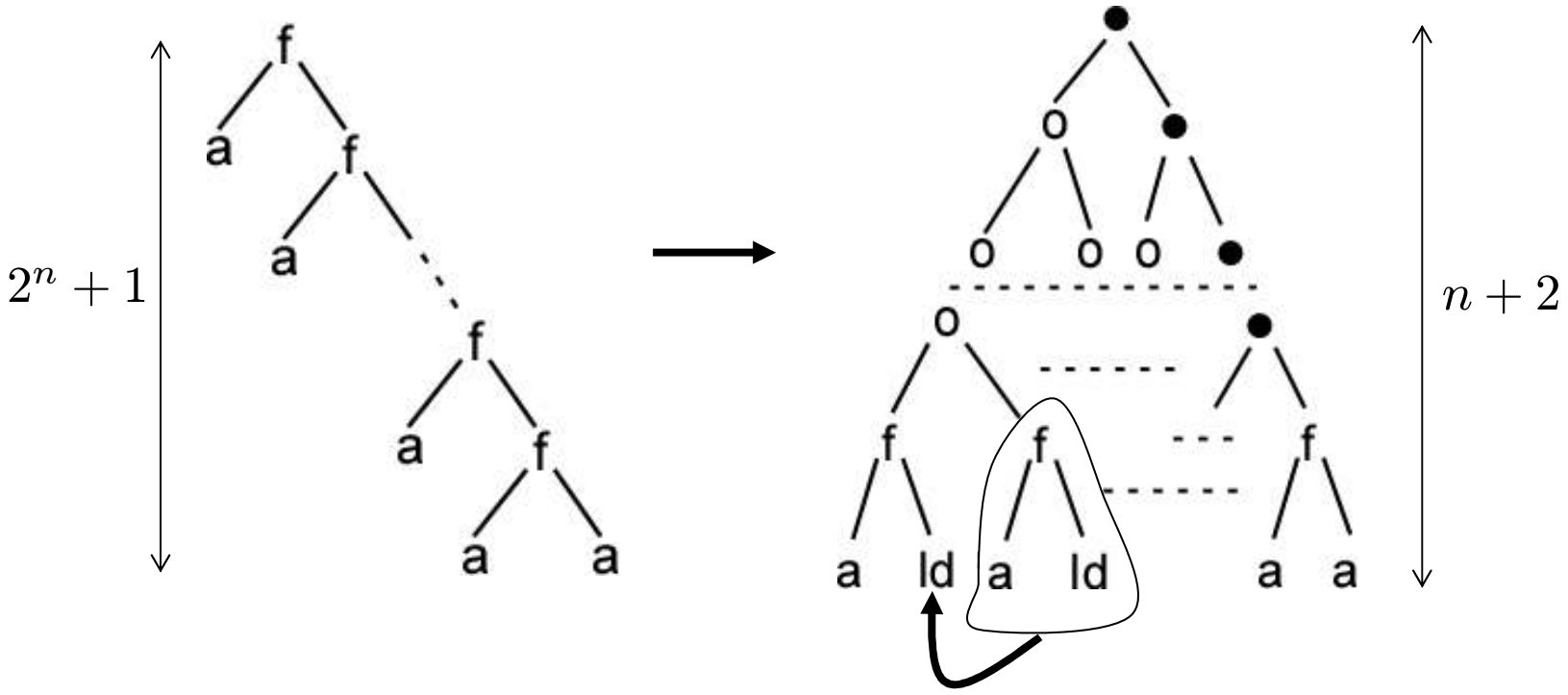
A *context* is a term with a special variable u. For nonassociative f, use an explicit substitution operation $\bullet$ on context c and term t where $c \bullet t = c[t/u]$:



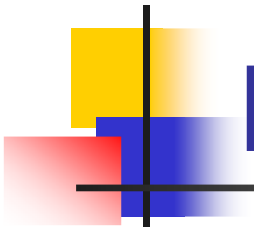
Idea: Choose c, t almost equal in size and recurse

Balanced terms II

Let $\bullet$ be substitution of terms into contexts, $\circ$ be substitution of contexts into contexts and Id the special *identity* context Id with $c \circ \mathsf{Id} = \mathsf{Id} \circ c = c$.
($\circ$ is associative with unit element Id)



Recursively cut terms using $\bullet$ and contexts using $\circ$

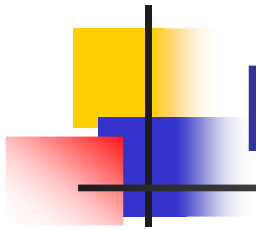


Balanced terms III

Proposition 1 (Courcelle-Vanicat) *Every term in $T(F, C)$ is equivalent to a 3-balanced term in $T(F \cup \{\circ, \bullet\}, C \cup \{Id\})$.*

But this is no longer a cwd term! To get a balanced cwd term has exponential blowup in cwd. For mcwd, we get only a constant blowup:

Lemma 1 *If $mcwd(G) \leq k$ then G can be defined by a 3-balanced mcwd term of width $\leq 2k$*



Balanced terms III

Proposition 1 (Courcelle-Vanicat) *Every term in $T(F, C)$ is equivalent to a 3-balanced term in $T(F \cup \{\circ, \bullet\}, C \cup \{Id\})$.*

But this is no longer a cwd term! To get a balanced cwd term has exponential blowup in cwd. For mcwd, we get only a constant blowup:

Lemma 1 *If $mcwd(G) \leq k$ then G can be defined by a 3-balanced mcwd term of width $\leq 2k$*

Proof. Given mcwd term t of width $\leq k$, use above proposition to obtain a 3-balanced term using $\circ, \bullet$. Interpret $\circ, \bullet$ as mcwd operations on $2k$ labels:

$$c \bullet t_H \equiv G_c \otimes_{R,g,h_c} H$$

Where $t_H \in T(F_k, C_k)$ defines the graph H, G_c is a graph with labels in $[k] \cup \{k+1, \ldots, 2k\}$. Applying this recursively to c and t_H we get a 6-balanced term in $T(F_{2k}, C_{2k})$. The construction is compositional:

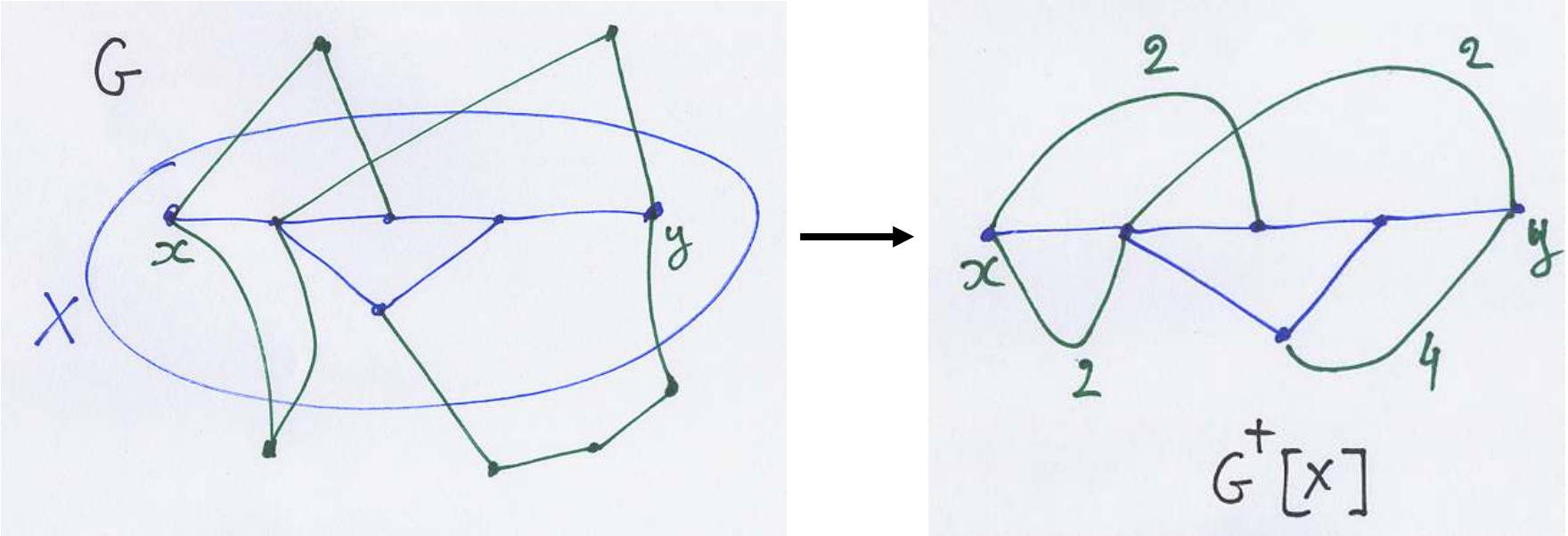
$$G_{c \circ d} = G_c \otimes G_d \quad \text{and} \quad h_{c \circ d} = h_c \circ h_d$$

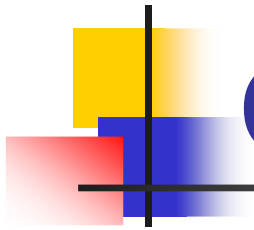
Main result

The balancing lemma allows us to use $ht(t) = O(\log n)$, hence we get

Theorem 1 *For G having mcwd $\leq k$ and n vertices, given $\{J(x) : x \in X\}$ we can compute the shortcut graph $G^+[X]$, where $|J(x)| = O(k^2 \log^2 n)$ bits.*

Since mcwd is more powerful than *cwd* and *twd*, we get labels of size $O(k^2 \log^2 n)$ for graphs having $cwd, twd \leq k$.





Conclusion

- Forbidden-set distance labelling for mcwd graphs
  - "Distance labelling with obstacles"
  - Can obtain a compact routing scheme
- Techniques
  - Use algebraic representation of graphs
  - Better describes *how* the graph is connected
  - Mcwd terms have good balancing properties

Open problems

- Planar graphs??

A generalization to MS definable queries (Courcelle and Vanicat, Discrete Applied Maths, 2003)

Theorem :

1) Given k, given an MS graph property $P(X_1, \ldots, X_m)$:

For every graph G defined by a clique-width (or m-clique-width) expression of width k , on can compute for each vertex x of G a label J(x) such that :

for sets of vertices $A_1, \ldots, A_m$, from the labels J(y) for every y in $A_1, \ldots, A_m$, one can determine if $P(A_1, \ldots, A_m)$ is true.

Size of J(y) : $O(\log(n))$

Preprocessing time : $O(n.\log(n))$

Answer to query : $O(a.\log(n))$ where is the $a = |A_1 \cup \ldots \cup A_m|$

2) For MS optimization functions (distance) replace log(n) by $\log^2(n)$

The basic result :

For graph G defined as val(t) for a term t in T(F,C) (binary operations and constants), in such a way that V(G) = the set of occurrences of constants in t , then every MS formula $\varphi(X_1, \ldots, X_m)$ can be translated into a deterministic finite automaton *A* for the signature $F \cup Cx\{0,1\}^m$ such that *A* accepts a term **t** in $T(F,Cx\{0,1\}^m)$ iff

$$G \models \varphi(A_1, \ldots, A_m)$$

where G is the graph defined by t = **t** without the $\{0,1\}^m$ elements

and A_i is the set of vertices u of G such that u is an occurrence of some (c,w) in $Cx\{0,1\}^m$ such that w[i] = 1.

(*Intuition* : u occurrence of (c,0,1) means that the vertex u is X_2 and not in X_1.

The set of terms accepted by *A* encodes all graphs of, say m-cwd at most k and the m-tuples of sets of vertices that satisfy φ in these graphs.)

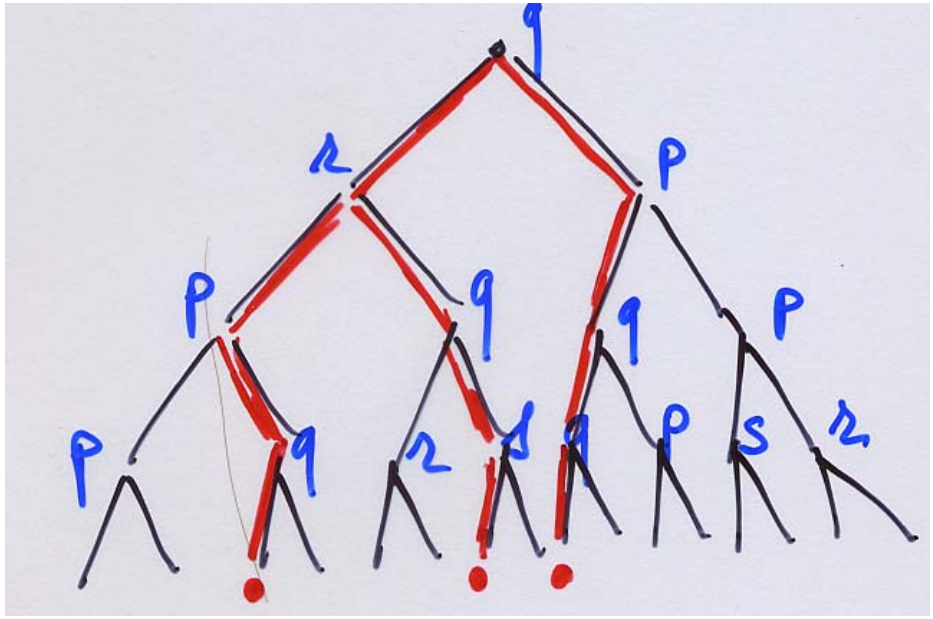
The method for $\varphi(X,Y)$ (m=2).

Assuming *A* constructed and G defined by a balanced term t in T(F,C) (for m-cwd at most k) :

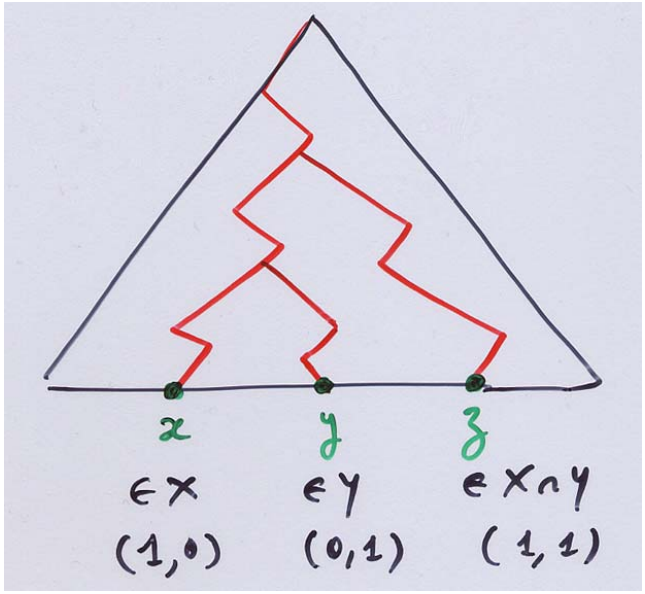
We run *A* on t with each c replaced by (c,0,0) (i.e. for empty sets X,Y),

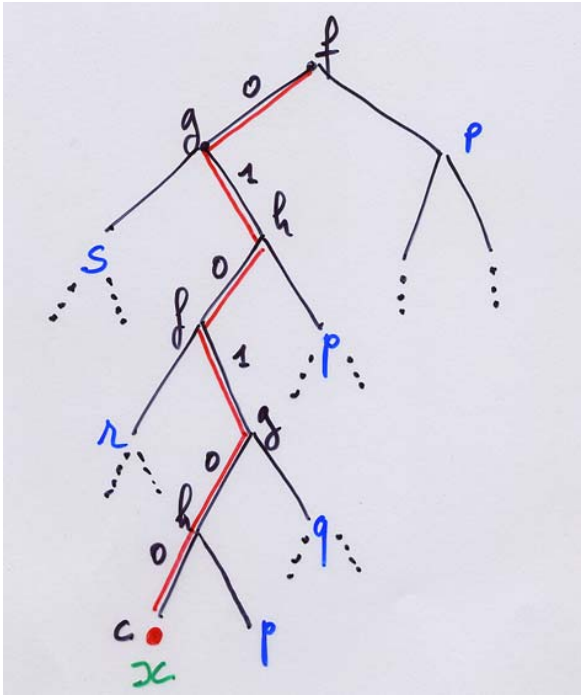
we mark each node with the corresponding "basic" states : p,q,r,s, …

If $X \cup Y$ is not empty, we modify accordingly some leaves. The new run will only modify the states on the branches from these leaves to the root.



These new states can be obtained from the “basic” states on nodes at distance one of these branches (because the new run is the same on the corresponding subterms. This information for a branch from **x** to the root can be stored into a





word J(**x**) like : [f,0,p] [g,1,s] [h,0,p] [f,1,r] [g,0,q] [h,0,p]c of size proportional to the length of the branch (which gives O(log(n)) for balanced terms and n vertices).

Linear delay enumeration (Discrete Applied Maths, to appear)

The problem is now to enumerate (to list) the set :

$$Sat(G,\varphi,(X_1, \ldots, X_m) = \{ (A_1, \ldots, A_m) \ / \ G \models \varphi(A_1, \ldots, A_m)\}$$

in linear delay : the next output is produced in time proportional to its size.

A data structure is built from G of m-cwd k, given by a (balanced) term, and the MS formula φ. It gives also $|Sat(G,\varphi,(X_1, \ldots, X_m)|$.

Background : For φ first-order and G of degree $\leq$ d the set

$Sat(G,\varphi,(x_1, \ldots,x_m)$ can be enumerated with constant delay between two outputs.

(A. Durand, E. Grandjean).

Applications (examples from WG 2007)

1) Transversal hypergraphs generation : H=(A,B,E), bipartite graph.

Wanted : the inclusion-minimal subsets X of A that have at least vertex adjacent to each b in B.

No “output-polynomial” algorithm is known. There is one for H of clique-width $\leq$ k.

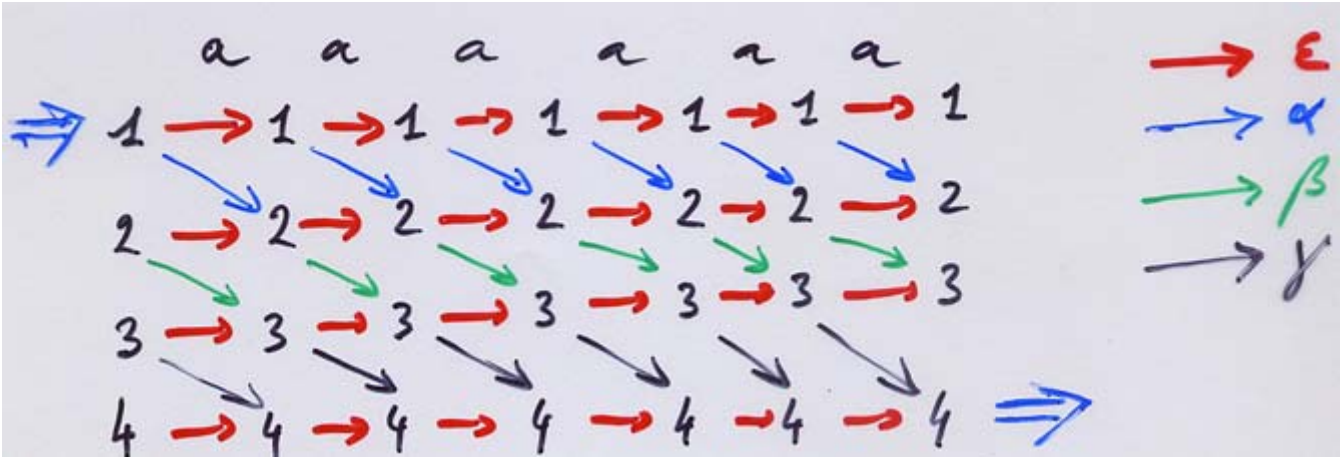
2) Counting and enumerating configurations : e.g. Eulerian orientations or bipolar orientations in planar graphs. This is possible in polynomial time for outerplanar or Halin graphs (twd $\leq$ 3).

The case of graphs reduces to that of terms denoting graphs.

Example concerning words

For a word *w* over letter *a*, find triples of occurrences (x,y,z) such that x<y<z.

For given *w*, one constructs the automaton generating the finite set of all "good" configurations. Here we take w = a^6.

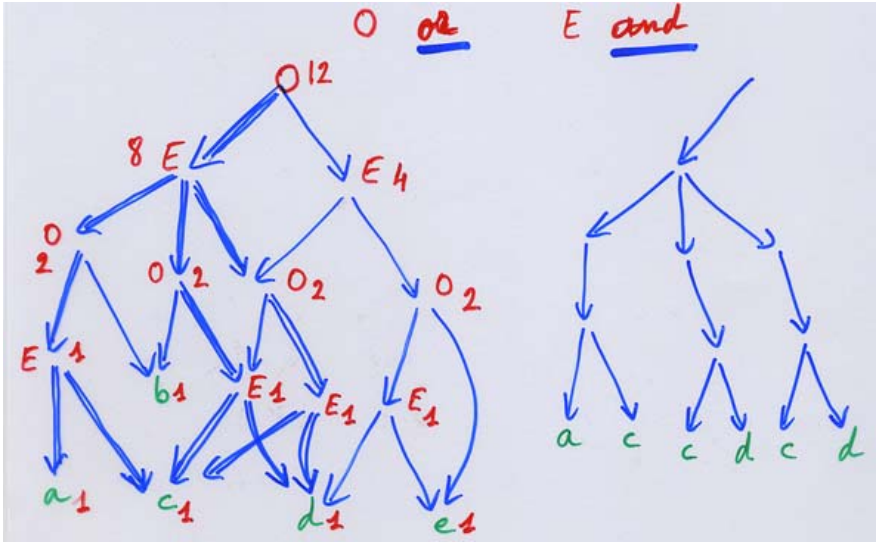


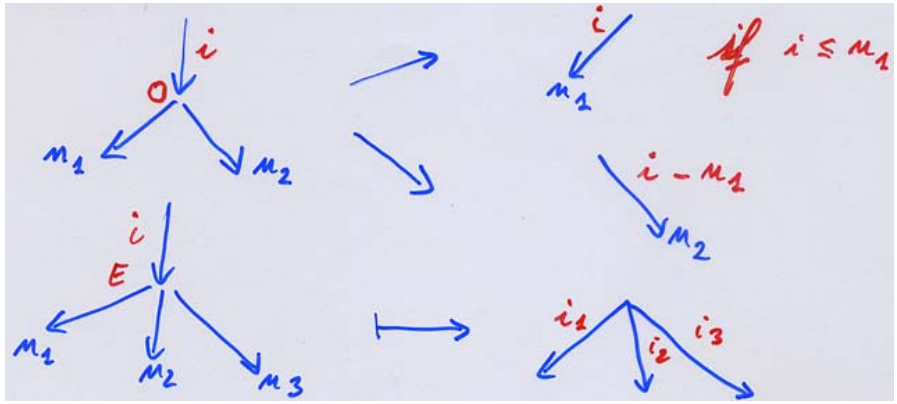
where α = (*a*,1,0,0), β = (*a*,0,1,0), γ =(*a*,0,0,1), ε = (*a*,0,0,0).

We can enumerate the paths from entry to exit with linear delay. But these paths contain a lot of useless information: not *linear delay w.r.t. the answers* to the query.

It suffices to eliminate ε-transitions.

Enumerating the trees embedded in an AND/OR DAG





Value of a node : the number of trees issued from that node. Computable bottom-up.

Determination of the i-th tree issued from a node :

OR-node with sons valued n_1 and n_2 : go left if $i < n_1$ otherwise go right with $i - n_1$

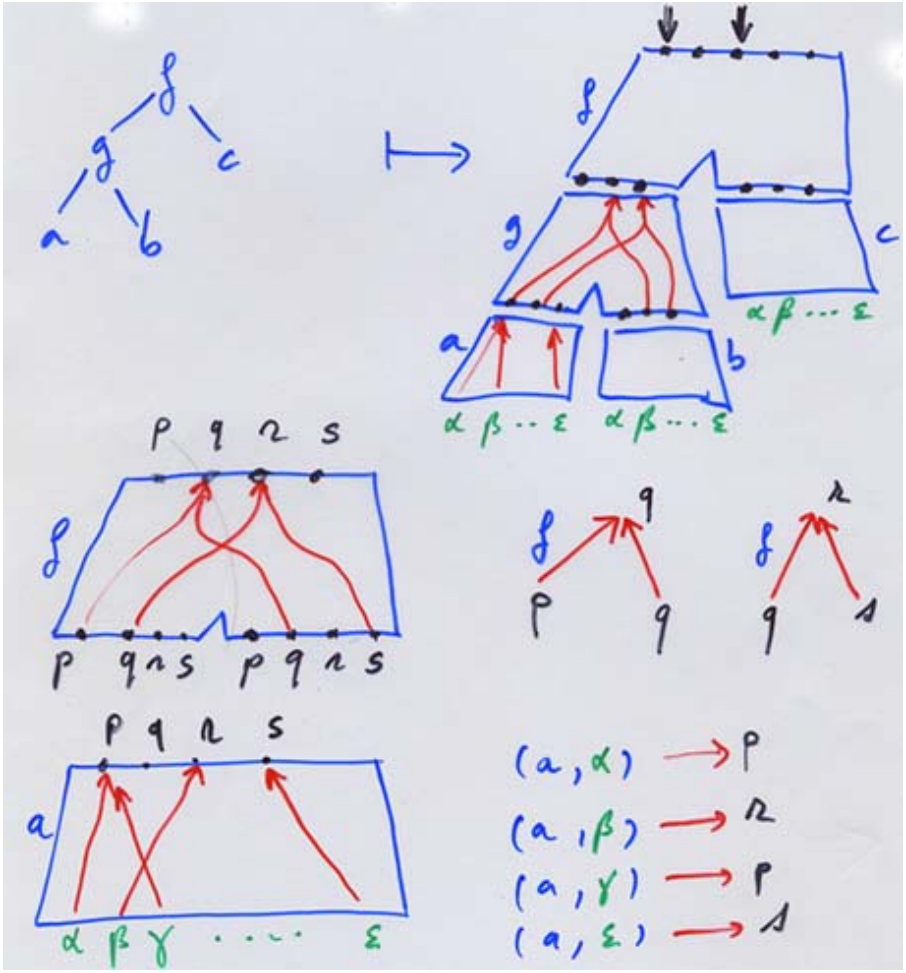
AND-node with sons valued n_1, n_2, n_3 : find i_1, i_2, i_3 such that :

$$i = i_1 + (i_2-1)\, n_1 + (i_3-1)\, n_1 . n_2$$

and go to all sons with respective values i_1, i_2, i_3 .

Gives a *linear delay enumeration* of the trees embedded in an AND/OR DAG.

AND/OR DAG associated with a term and a deterministic automaton on terms in $T(F,Cx\{0,1\}^m)$



It remains to extract the embedded trees starting from the accepting root nodes.

"Dead" subtrees that do not contribute to the result can be avoided with help of a kind of ε-reduction.

One uses the previous result.

This takes time O(n.log(n)) where n = number of vertices = number of occurrences of constants and the term is assumed balanced.