Verification of graph properties using vertex labels

Bruno Courcelle

Université Bordeaux 1, LaBRI and

Institut Universitaire de France

References : Articles with R. Vanicat (2003), with A. Twigg (STACS 2007), with C. Gavoille, M.M. Kanté, A.T. (TGGT, Paris, 2008), and with C.G., M.M.K. (FAW, Changsha, China, 2008)

See : http://www.labri.fr/perso/courcell/ActSci.html

Labelling Schemes for solving First-Order and Monadic Second-Order Queries in Graphs not necessarily of bounded clique-width

Aim : to check properties of vertices and to compute functions like distance from fixed *short* vertex labels.

Short = of length O(log(n)) or $O(\log^2(n))$ bits ; n = number of vertices

Wanted : for fixed class of graphs *C* and function *F* (includes property) two algorithms :

Algo *A* : defines for G in *C* a label J(x) for each x in V = V(G).

Algo *B* : computes *F*(u,v,X,Y) from J(u), J(v), J(X), J(Y)

for vertices u, v, sets of vertices X,Y in some G in *C*

Idea : all necessary information from G is distributed on vertices;

computing *F* need not process the graph (this has been done by *A*).

Results

Adjacency (implicit representation) with labels of size $O(\log(n))$:

Bounded arboricity (includes planar, bounded tree-width)
Bounded clique-width,
Interval graphs (unbounded clique-width and arboricity)

Distance Static : all $\Theta(n)$

trees, bdd tree-width, clique-width $\Theta(\log^2(n))$
planar between $O(n^{1/3})$ and $O(n^{1/2})$
interval graphs $O(\log(n))$

Distance Dynamic : obstacles (forbidden parts, specified in query)

to be computed : $d(u,v,X,F)$, the distance of u and v in $(G-F)\backslash X$

X : forbidden vertices, F : forbidden edges.

Clique-width $\leq$ k : $O(k^2.\log^2(n))$ (CT, stacs07)

Connectivity labelling (dynamic)

Clique-width ≤ k : $O(k^2.\log(n))$ (CT, stacs07)

Planar graphs with obstacles : O(log(n)) (CGKT 08)

General logical approach

Monadic second-order properties (optimization or counting functions) :

Clique-width ≤ k : O(f(k).log(n)) ($O(f(k).\log^2(n))$) (CV 2003)

First-order properties and counting functions :

Classes of graphs "nicely decomposable",
of locally bounded tree-width or clique-width :
O(log(n)) or $O(\log^2(n))$ (CGK 2008)

Tools

Graph structure properties : unions of forests, tree-decompositions,

(*balanced*) clique-width expressions,

decompositions in 3-connected components (for planar graphs)

covering by families of subgraphs of bounded clique-width

with limited overlapping.

Straight-line planar embeddings (De Fraysseix *et al.*, Schnyder)

Monadic second-order formulas on terms translated into finite automata

(Doner, Thatcher-Wright).

First-order formulas decomposed into local and "connected" formulas

(Gaifman, Frick)

Bounded arboricity

G is the union of k edge-disjoint rooted forests $F_1, \ldots, F_k$

f_i is the (partial) father function in forest F_i.

We define label $J(x) = (x, f_1(x), \ldots, f_k(x))$ of size $\leq (k+1).\lceil \log(n) \rceil$

Adjacency check :

u and v are adjacent if and only if :

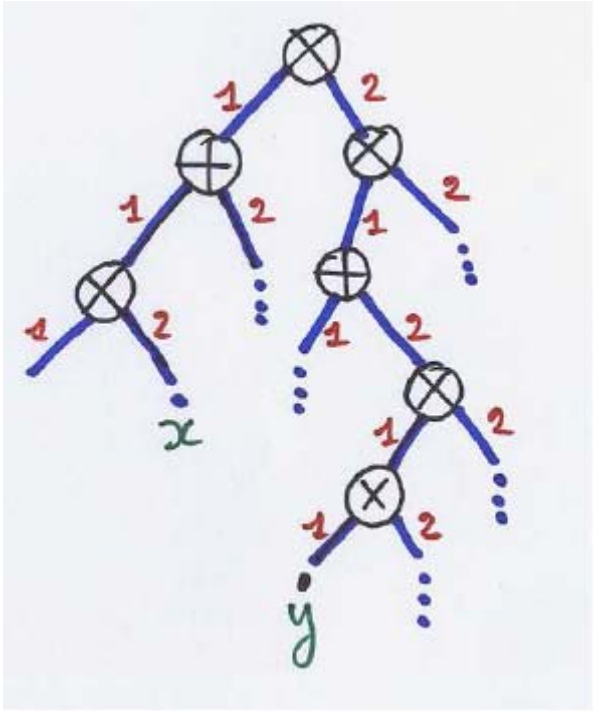
$u = f_i(v)$ or $v = f_i(u)$ for some i.

Cographs

Built from vertices with two binary operations :

$\oplus$: disjoint union

$\otimes$: complete join = disjoint union plus all edges between the two arguments



Adjacency labels are words (branches)

J(x) = $\otimes$ 1 $\oplus$ 1 $\otimes$ 2

J(y) = $\otimes$ 2 $\otimes$ 1 $\oplus$ 2 $\otimes$ 1 $\otimes$ 1

The least common ancestor of x and y is labelled by $\otimes$, hence they are adjacent. The size of J(u) is not O(log(n)). By using other graph operations one makes terms balanced i.e., of height O(log(n)).

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) :

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

Short labels for MS definable queries

(Courcelle and Vanicat, Discrete Applied Maths, 2003)

Theorem : 1) Given k and a monadic second-order graph property $P(X_1,\ldots,X_m)$:

for every graph G defined by a clique-width expression of width k, one can define a label J(x) for each vertex x of G such that, from the labels J(y) for every y in sets of vertices $A_1, \ldots, A_m$, 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 $a = |A_1 \cup \ldots \cup A_m|$

2) For MS optimization functions (like distance) or counting functions (number of tuples of vertices $b_1,\ldots,b_q$ that satisfy $P(b_1,\ldots,b_q,A_1,\ldots,A_m)$ for given $A_1,\ldots,A_m$), we replace log(n) by $\log^2(n)$

Basic result : the case of terms (instead of graphs).

We consider terms t in T(F,C) (F : binary operations, C : constants),

Every MS formula $\varphi(X_1, \ldots, X_m)$ with free variables denoting sets of occurrences of constants in t 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

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

where **t** is t with the $\{0,1\}^m$ labels attached to occurrences u of constants and A_i is the set of occurrences u of some (c,w) in $Cx\{0,1\}^m$ such that w[i] = 1.

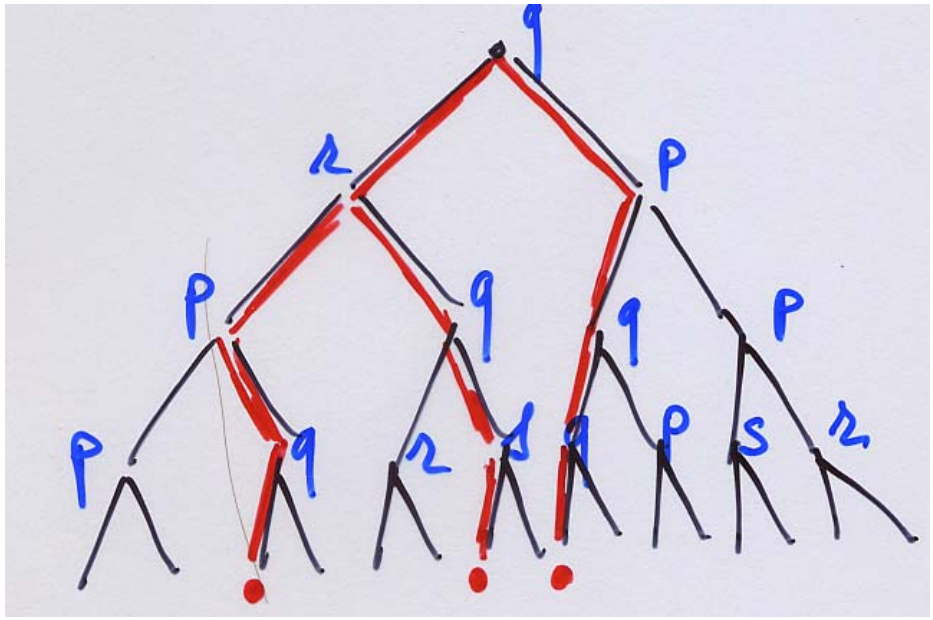
(*Intuition* : u occurrence of (c,0,1) means that u is in X_2 and not in X_1.

The set of terms accepted by *A* encodes terms and the m-tuples of sets of occurrences of constants (leaves of t as a tree) that satisfy φ in these terms.)

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

Assuming *A* constructed and t in T(F,C) be given :

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 states : p,q,r,s, …



If X∪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.

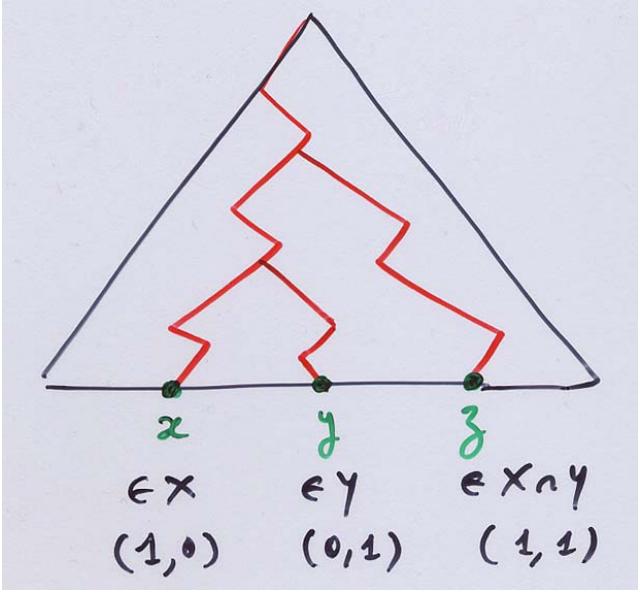
These new states can be obtained from :

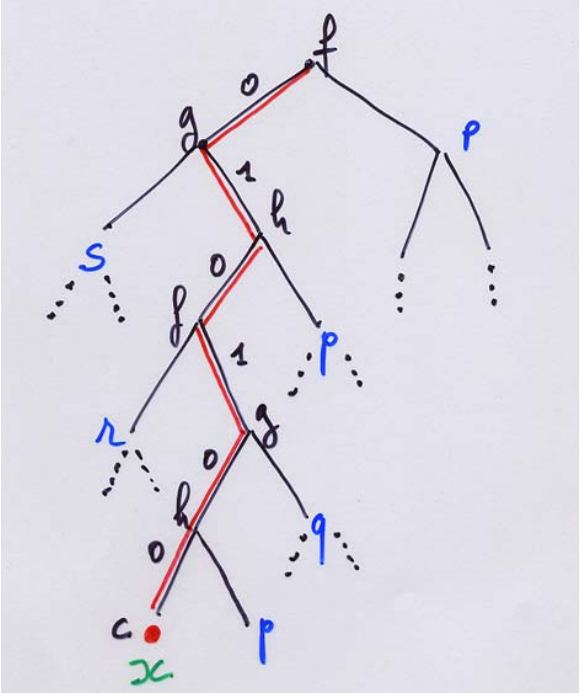
the states on these branches and those at distance 1 of these branches (because the new run is the same on the corresponding subterms).

This information for a branch from **x** can be stored in 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 (= O(log(n)) for a balanced term with n leaves).





The case of graphs :

For graph G defined as val(t) for a term t in T(F,C) (F, C operations defining clique-width), then V(G) = the set of occurrences of constants in t.

Every MS formula $\varphi(X_1, \ldots, X_m)$ can be translated into an equivalent MS formula $\psi(X_1, \ldots, X_m)$ on the term :

$$G \models \varphi(A_1, \ldots, A_m) \quad \text{iff} \quad t \models \psi(A_1, \ldots, A_m).$$

We apply to ψ the previous construction.

Counting functions : at each node w of t, we store numerical information : for each state p, the number of assignments of 0,1's to the leaves below w that yield state p at this node. This table has size #-of-states. $\lceil \log(n) \rceil$

Optimization : similar method, the table indicates the maximum value reachable with state p for some assignments of 0,1's.

Short labels for connectivity check in planar graphs with obstacles.

(Courcelle, Gavoille, Kanté, Twigg 2008)

Question is : are u and v connected in (G-F) \ X ?

Method :

(1) We treat 3-connected planar graphs and their edge subdivisions (new vertices inserted on edges).

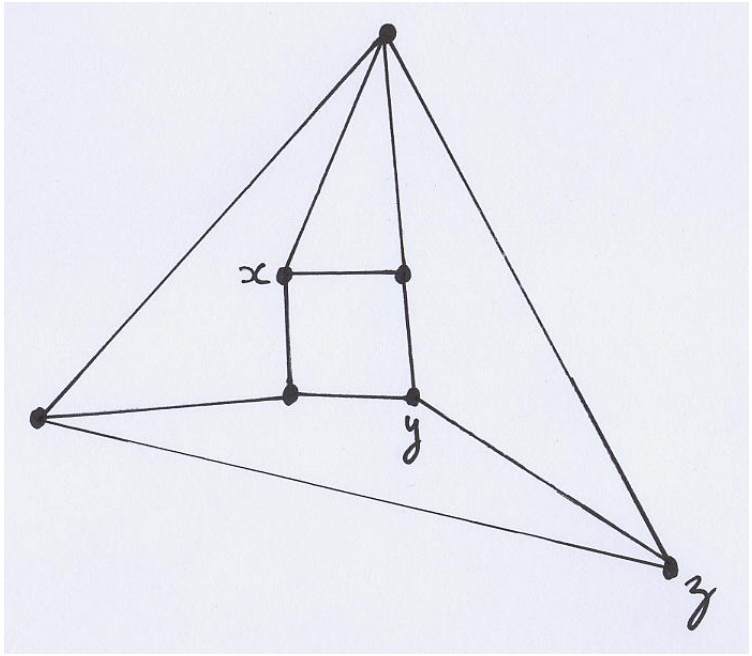
(2) We treat 2-connected planar graphs decomposed in 3-connected blocks.

(3) We treat connected planar graphs decomposed as trees of 2-connected components.

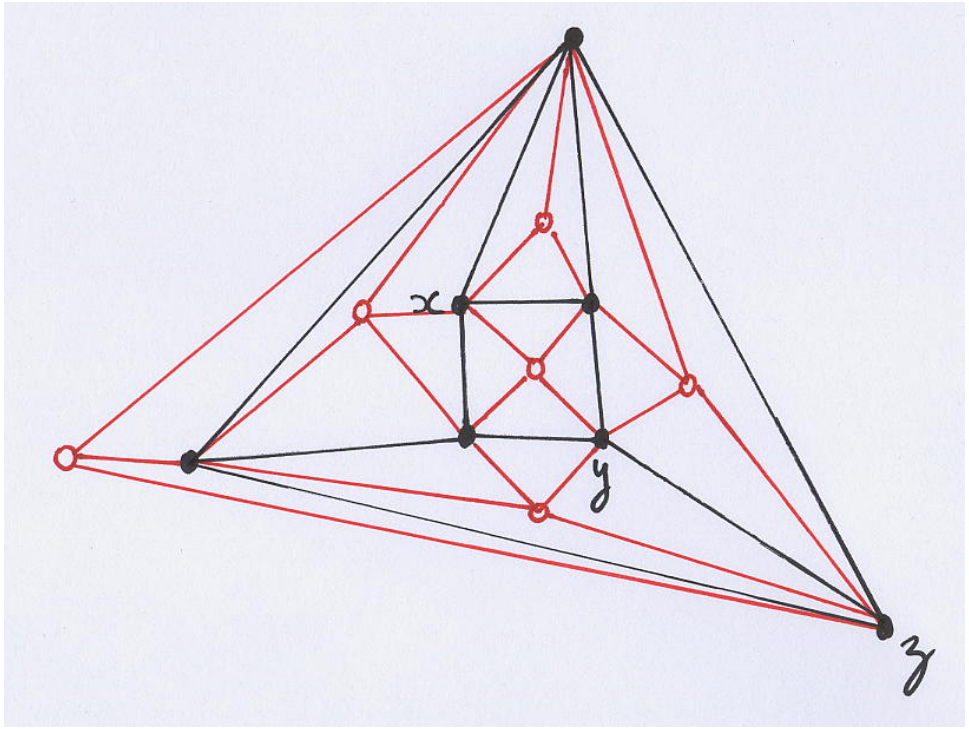
For case (1) we use a geometric method.

For cases (2) and (3) we use a labelling *à la* Courcelle-Vanicat for querying the decomposition trees, combined with labellings of type (1) for 3-connected blocks.

3-connected planar graphs

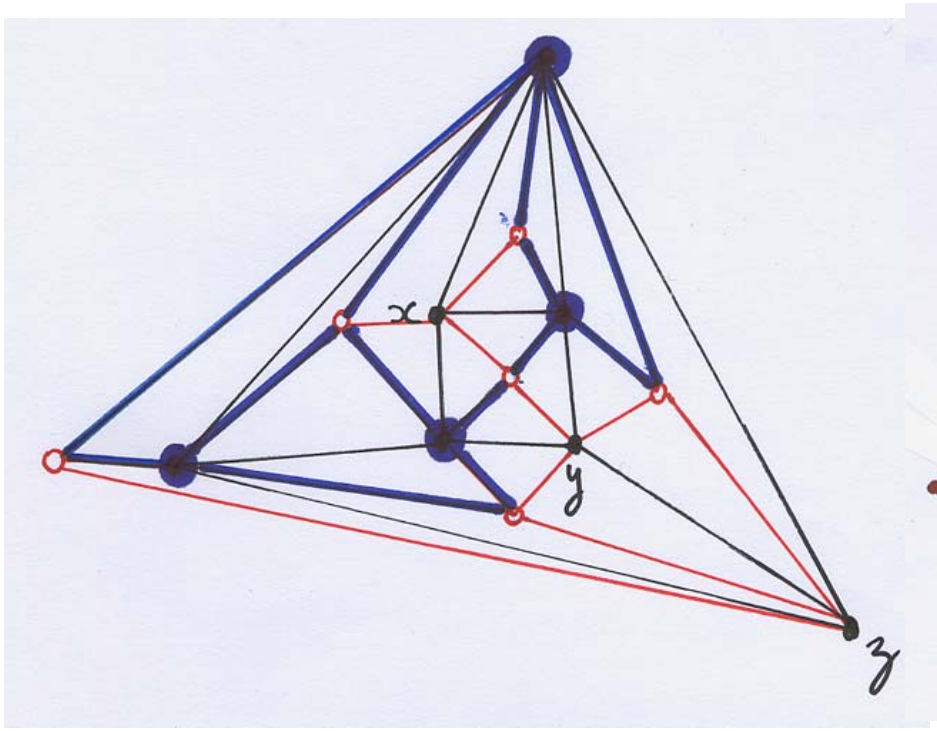


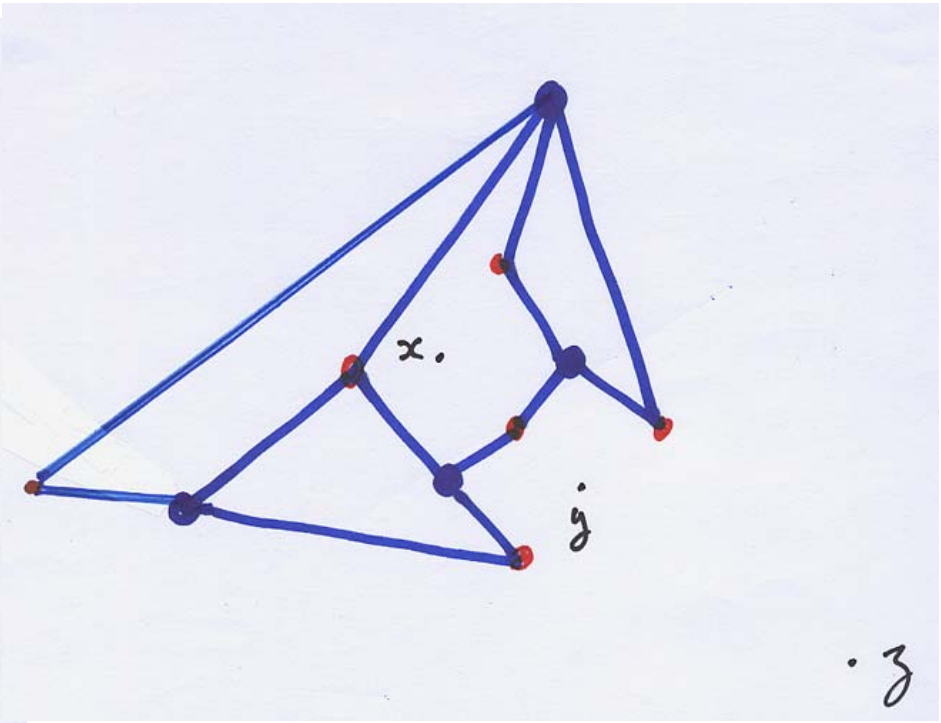
A graph G



Its augmented graph G^+

= G with "face-vertex edges"





X = the set of 4 big blue vertices.

Its *barrier* Bar(X) is the set of **thick blue** edges (consists of all u--f--v where u,v in X and f is a face-vertex)

Deleting X separates x and y but not y and z.

The barrier separates topologically x and y but not y and z

To be done :

1) Construct a straight-line embedding of G^+ with integer coordinates of maximum value O(n). This is possible since G^+ is simple because G is 2-connected. (dF, Sch)

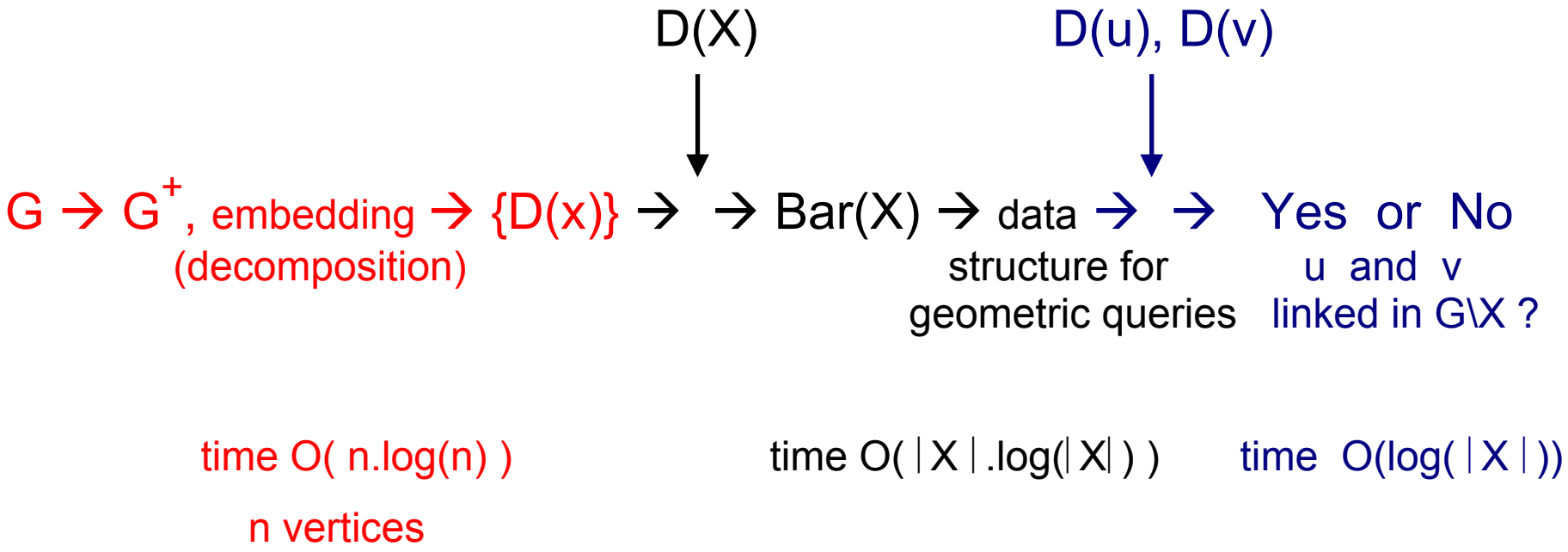
2) From labels of vertices in X, determine the coordinates of the end vertices of the edges in Bar(X).

3) Using computational geometry algorithm, test whether two vertices u, v given by their coordinates are separated in the plane by Bar(X).

For 2) since G^+ is planar, it is the union of 3 forests. One uses an adjacency labelling for G^+, from which, for any two vertices u,v one can obtain the at most two faces f and h to which they are both adjacent, as values of $g_i(u)$ or $g_i(v)$ for some i = 1,…,30 where $g_1,\ldots,g_{30}$ is a finite list of partial functions. (Which i's give the faces incident with u and v depends on tests of the form "$g_j(u) = g_k(v)$?").

Labels : D(x) = (C(x), $C(g_1(x)),\ldots,C(g_{30}(x))$) where C(u) = integer coordinates of u.

Overview of the algorithm(s) :

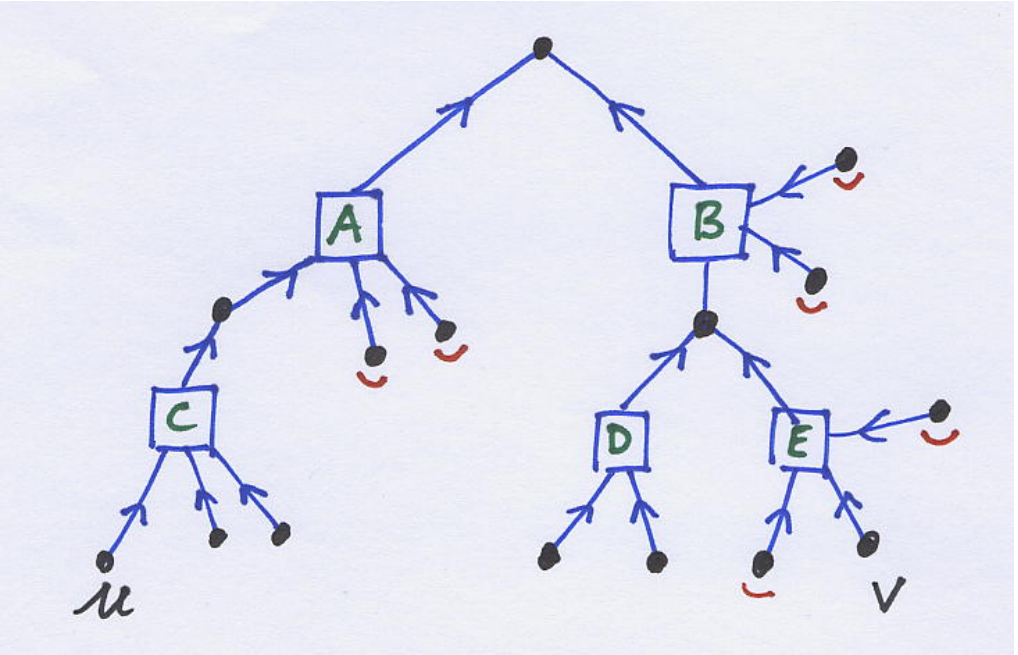


First we process G, then X for fixed G.

Then connectivity queries for various u,v, and fixed X.

General case : Tool 1 : the tree of 2-connected components.

Its nodes are the vertices of G and nodes representing 2-connected components (A,B,C,…). Adjacency = membership of a vertex in a 2-connected component.



Fact : X separates **u** and **v** in the graph G if **u** and **v** are separated by X in the tree or if they are separated in G by X∩B, where B is a problematic biconnected component on the path in the tree between **u** and **v** (like A,B,E, with ≥ 2 vertices in X).

A labelling K of the tree can be built with Courcelle-Vanicat's technique, from which :

one can detect if the first case holds

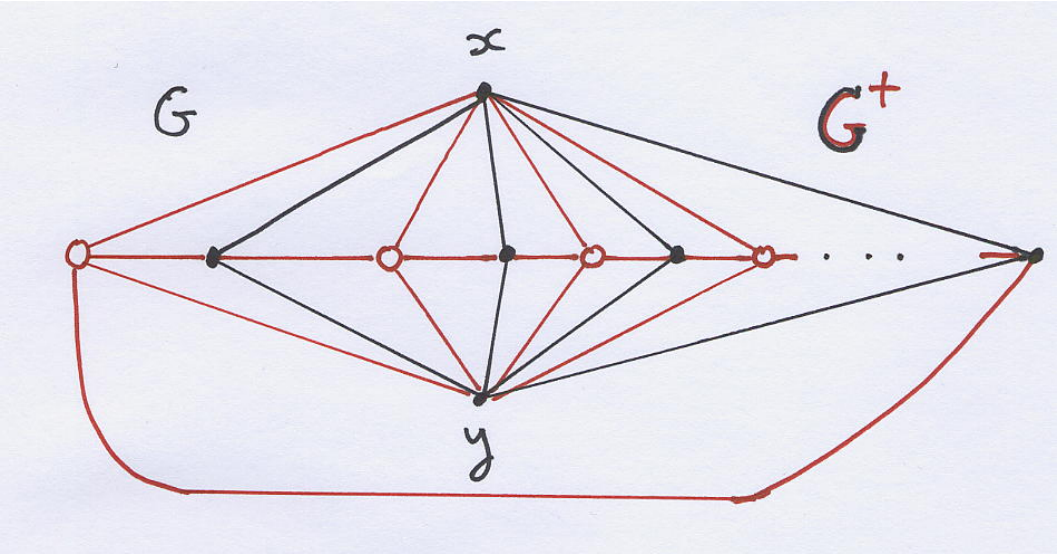
and, if it does not, those problematic 2-connected components B that may separate **u** and **v**.

For each of them, the geometric method (using labelling D) can be used.

The label of x is then (D(x), K(x)). (Omitting technical details).

Tool 2: 2-connected components decomposed in 3-connected ones

Where is the difficulty ?



For such graphs, **x** and **y** are incident with an unbounded number of faces, hence one cannot specify all of them with a fixed number of functions, and one cannot find the coordinates of all edges in Bar({**x,y**}).

Method : we replace the barrier by a *reduced barrier* : if **x** and **y** are incident with at least 3 faces, $f_1,\ldots,f_k$ we put in *RBar*({x,y}) only **x**— f_1 and **y**— f_1.

Problem : The reduced barrier *RBar*(X) will *miss some cases* where the given vertices are separated by the "full" *Bar*(X).

These cases will be detected on the decomposition tree in 3-connected components, by a labelling of this tree with using Courcelle and Vanicat's method and several additional nice tricks.

Extension 1 : deleting edges and adding new links

X : deleted vertices,

F : deleted edges, handled as degree 2 vertices in a subdivided graph

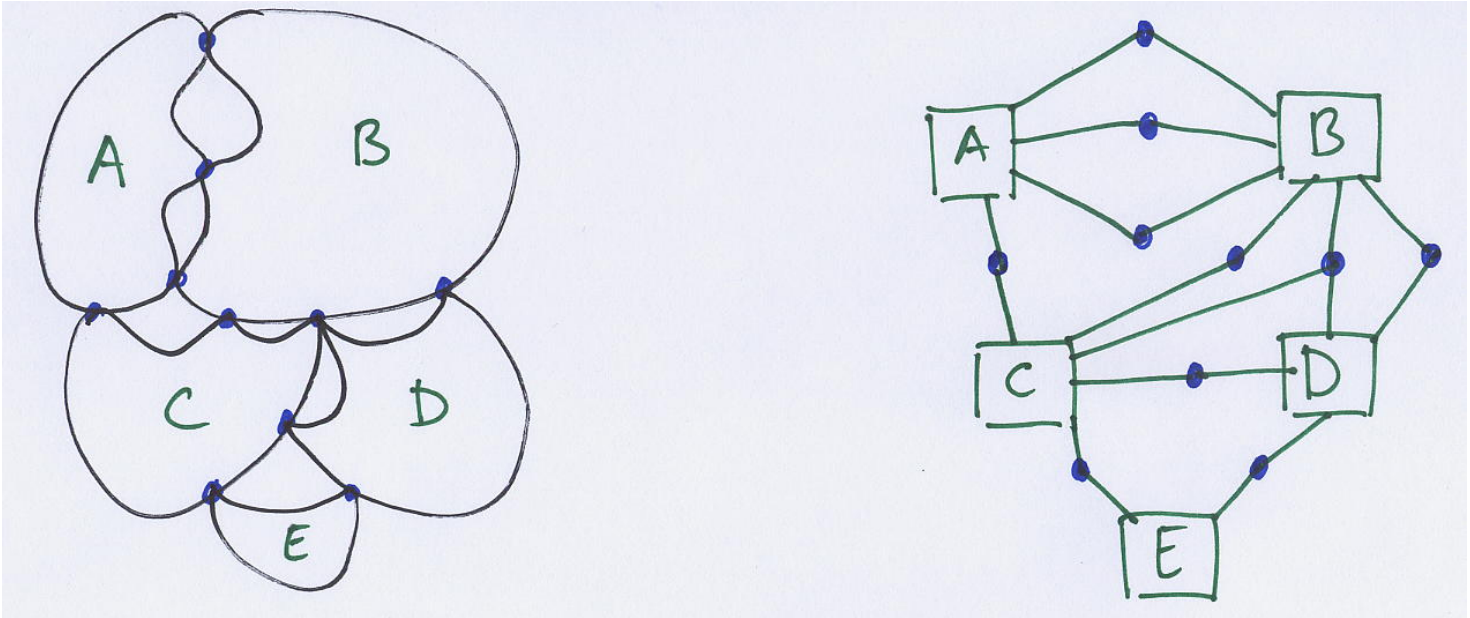
H : new links between pairs of vertices.

Query : Are u and v connected by a path in $((G - F) \setminus X) + H$?

The data structure is built for X and F

Query takes time $O(|X| . |H|^2)$

Extension 2 : Graph covers with limited overlaps ; combining schemes



Connected blocks with

O(log(n)) labelling

Skeleton graph

bipartite, degree $\leq$ d

O(log(n)) labelling

The two labelling schemes can be combined into a single O(log(n)) one.

First-order formulas in graphs of unbounded clique-width

(Courcelle, Gavoille, Kanté, 2008)

First-order formulas with set arguments (*unquantified set variables*).

Motivation (networks with failures, graphs with obstacles) :

A formula $\varphi(x_1,\ldots,x_m,X)$ can express that $G \setminus X \models \psi(x_1,\ldots,x_m)$.

Definitions and notation

$N(G,t, a_1,\ldots,a_m)$ = all vertices at distance $\leq t$ of some a_i.

t-local property with tuple $\underline{B}$ of set arguments (treated as colors):

$P(a_1,\ldots,a_m,\underline{B})$ true in G if and only if

$P(a_1,\ldots,a_m,\underline{B'})$ true in $G[N(G,t,a_1,\ldots,a_m)]$,

where $B'_i = B_i \cap N(G,t,a_1,\ldots,a_m)$

Remark : Distances do not depend on $\underline{B}$ (colors)

t-connected property : t-local and

$P(a_1,\ldots,a_m,\underline{B})$ true in G implies $d(a_i,a_j) \leq 2t+1$

Basic t-local "sentences" with free set variables :

$\exists x_1,\ldots,x_m \, (\bigwedge_i \psi(x_i,\underline{X}) \wedge \bigwedge_{i,j} d(a_i,a_j) > 2t+1)$ where ψ is t-local

Theorem [(1) Gaifman 1982, (2) Frick 2004] :

(1) Every FO formula $\varphi(x_1,\ldots,x_m,\underline{X})$ is effectively equivalent to a Boolean combination of t-local and basic t-local formulas.

(2) Every t-local formula is effectively equivalent to a Boolean combinations of formulas of the form

$$\bigwedge_k \psi_k(\underline{u}_i,\underline{X}) \wedge \bigwedge_{i,j} d(\underline{u}_i,\underline{u}_j) > 2t+1$$

where each ψ_k is t-connected, the $\underline{u}_i$'s are tuples of free variables and $d(\underline{u}_i,\underline{u}_j) > 2t+1$ means $d(v,v') > 2t+1$ for each v in $\underline{u}_i$ and v' in $\underline{u}_j$.

Nicely decomposable classes of graphs C :

For every r, there exists d and integer function g, and one can construct in polynomial time for every graph G in C a family of sets **W**, called a cover, such that :

(1) V(G) is the union of the sets in **W** and furthermore :

for every vertex u, $N(G,r,u) \subseteq U$ for some $U \in \mathbf{W}$.

(2) The intersection graph of the cover has degree $\leq d$,

(3) For all $U_1,\ldots,U_k$ in **W** , $cwd(G[U_1 \cup \ldots \cup U_k]) \leq g(k)$.

Examples : Bounded degree, planar, graphs covered by blocks of bounded clique-width and bounded overlap, excludes a minor, others ???

Labels for local queries $\varphi(x_1,\ldots,x_m,\underline{Y})$

It is enough to consider formulas that are Boolean combinations of :

(a) $d(x_i,x_j) \leq 2t+1$ (and their negations)

(b) $\psi_k[N(G,r, x_i)](x_{i1},\ldots,x_{ip},\underline{Y})$, ψ_k t-connected,

for $r = m(2t+1)$

We build a cover **W** for this r, with parameters (d,g).

It has at most n.d sets.

In each U of **W**, we construct a CV-labelling J_U for checking formulas (a), (b). We let for each vertex x :

$L(x) = (x, \{(U, J_U(x)) / N(G,r, x) \subseteq U\}, \{(U, J_U(x)) / x \in U, N(G,r, x) \not\subseteq U\})$.

How to use $L(a_1),\ldots,L(a_m)$, $L(B_1),\ldots,L(B_q)$?

(1) For each pair a_i,a_j , we determine if $d(a_i,a_j) \leq 2t+1$: this is true iff there exists U in **W** with $N(G,r,a_i) \subseteq U$, $a_j \in U$, and $d(a_i,a_j) < 2t+1$ in $G[U]$.

From $L(a_i),L(a_j)$, we can find possible U and check distance using J_U.

(2) From Frick's decomposition, we can determine those formulas $\psi_k[N(G,r, a_i)](a_{i1},\ldots,a_{ip},\underline{B} \cap N(G,r, a_i))$ to check. Such a formula holds iff there exists U in **W** such that $N(G,r,a_i) \subseteq U$ and $\psi_k[U](a_{i1},\ldots,a_{ip},\underline{B}\cap U)$ holds.

From labels, one can determine the sets $\underline{B} \cap U$ and check the truth.

Remark : We need only $cwd(G[U]) \leq g(1)$ for all U in **W**.

First-order queries with set arguments.

From Gaifman's Theorem, we need check "sentences" (with set arguments) :

$\exists x_1,\ldots,x_m \; (\bigwedge_i \psi(x_i,\underline{X}) \wedge \bigwedge_{i,j} d(a_i,a_j) > 2t+1)$ where ψ is t-local.

We build **W** a (2t+1,d,g) cover

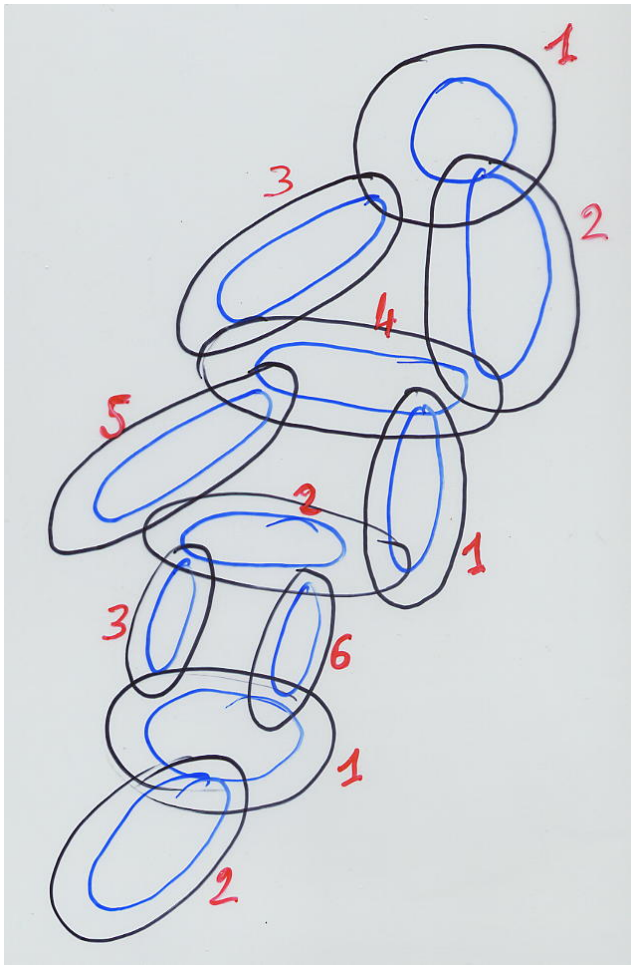
Define : **K**(U) = {a / N(G,2t+1,a) $\subseteq$ U}

γ : a distance-2 coloring of the intersection graph of **W**

using fixed number of colors (depends on d)

W(i,j) = union of blocks of colors i and j

G(i,j) = G[W(i,j)] = disjoint union of unions of pairs of blocks , hence cwd(G(i,j)) $\leq$ g(2).



For each graph G(i,j) we construct a labelling (tool from Courcelle-Vanicat) for checking the formulas :

$$\exists x,y\ (d(x,y) > 2t+1 \wedge \psi(x,\underline{B}) \wedge \psi(y,\underline{B}) \wedge \text{“}\delta(x)= i\text{”} \wedge \text{“}\delta(y)= j\text{”})$$

where $\delta(x)= i$ iff a $\in$ **K**(U) , $\gamma(U) = i$, and i is smallest of this form.

Hence, δ is a coloring of G derived from **W** and γ that we add to the graph before computing the labelling.

A key fact is that d(a,b) > 2t+1 in G iff d(a,b) > 2t+1 in G(i,j), if $\delta(a)= i$ and $\delta(b)= j$.

If m>2 , we use for γ a distance-m labelling of the intersection graph and the bound g(m) instead of g(2) for the cwd($G(i_1,\ldots,i_m)$)'s.

The labelling L(a) of a consists of : a, δ(a), and

- the labels of a relative to the (finitely many) graphs $G(i_1,\ldots,i_m)$, containing it , for the sentences to test,
- the truth values of the sentences to check in the graphs $G(i_1,\ldots,i_m)$, in case the set arguments are all empty.

How to use $L(B_1),\ldots,L(B_q)$?

1. One determines the graphs $G(i_1,\ldots,i_m)$ that do not meet any set $B_1,\ldots,B_q$. For them, one gets the desired truth value, from the label of any element of $B_1,\ldots,B_q$. If one is true we can stop.

3. For all other graphs $G(i_1,\ldots,i_m)$, we compute the restriction of $\underline{B}$ to them, and using the labels, we determine the validity of the considered "sentence" in $G(i_1,\ldots,i_m)$. If one answer is positive, we stop.

Conclusion

Extensions and open questions :

Graphs on nonplanar surfaces

Counting queries

Enumeration (better than counting)

Understanding “nicely decomposable classes”

Larger labels against less conditions on graphs

How hard is it to update the structure if one adds vertices and/or edges ? (deletions are handled by set arguments included in queries).

Reachability in directed planar graphs with obstacles.

Graph operations defining Clique-width

Clique-width has no combinatorial characterization but is defined in terms of few very simple graph operations (giving easy inductive proofs).

Equivalent notion: rank-width (Oum and Seymour) with better structural and algorithmic properties.

Graphs are simple, directed or not.

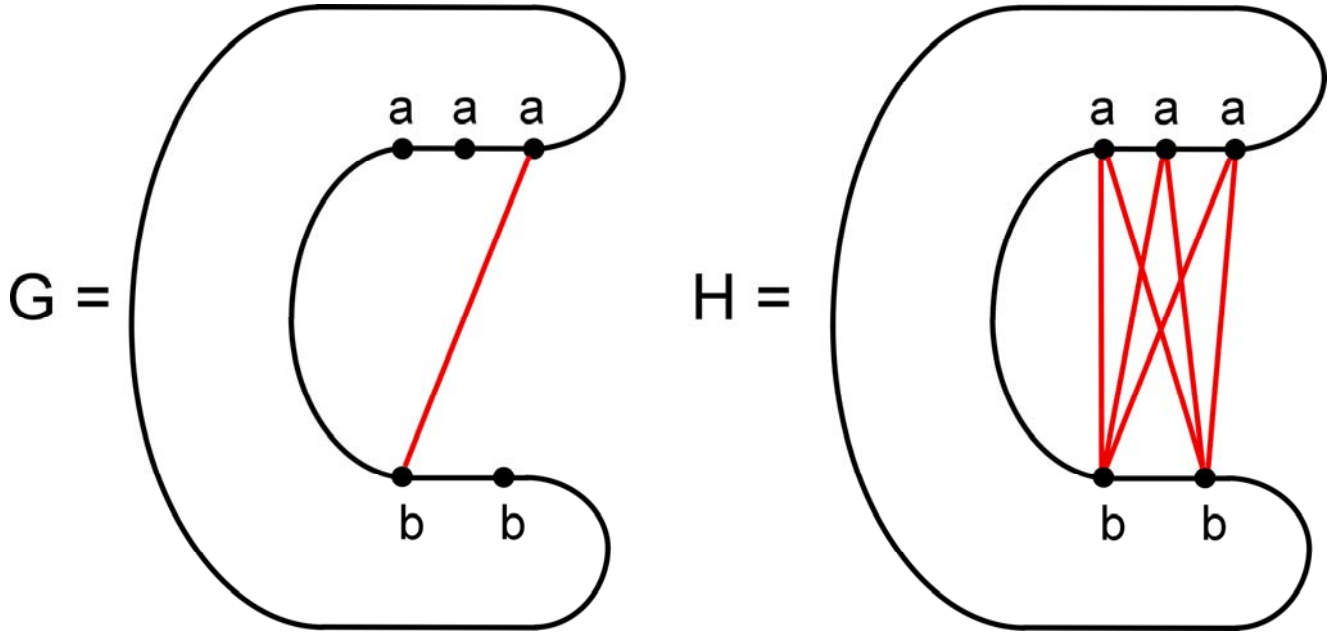
k labels : *a* , *b* , *c*, ..., *h*. Each vertex has one and only one label ;

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

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

Unary operations: Edge addition denoted by $Add\text{-}edg_{a,b}$

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



H = $Add\text{-}edg_{a,b}$(G) ; only 5 new edges added

The number of added edges depends on the argument graph.

Vertex relabellings :

$Relab_{a \longrightarrow b}(G)$ 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 with the operations $\oplus$, $Add\text{-}edg_{a,b}$ and $Relab_{a \longrightarrow b}$ with k labels. Its 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.

Classes of unbounded tree-width and bounded clique-width.

Cliques (2),

Complete bipartite graphs (2),

Distance hereditary graphs (3),

Graphs without P_5 and $\mathbf{1} \otimes P_4$ (5), or $\mathbf{1} \oplus P_4$ and $\mathbf{1} \otimes P_4$ (16) as induced subgraphs. (many similar results for exclusion of induced subgraphs with 4 and 5 vertices).

Classes of unbounded clique-width :

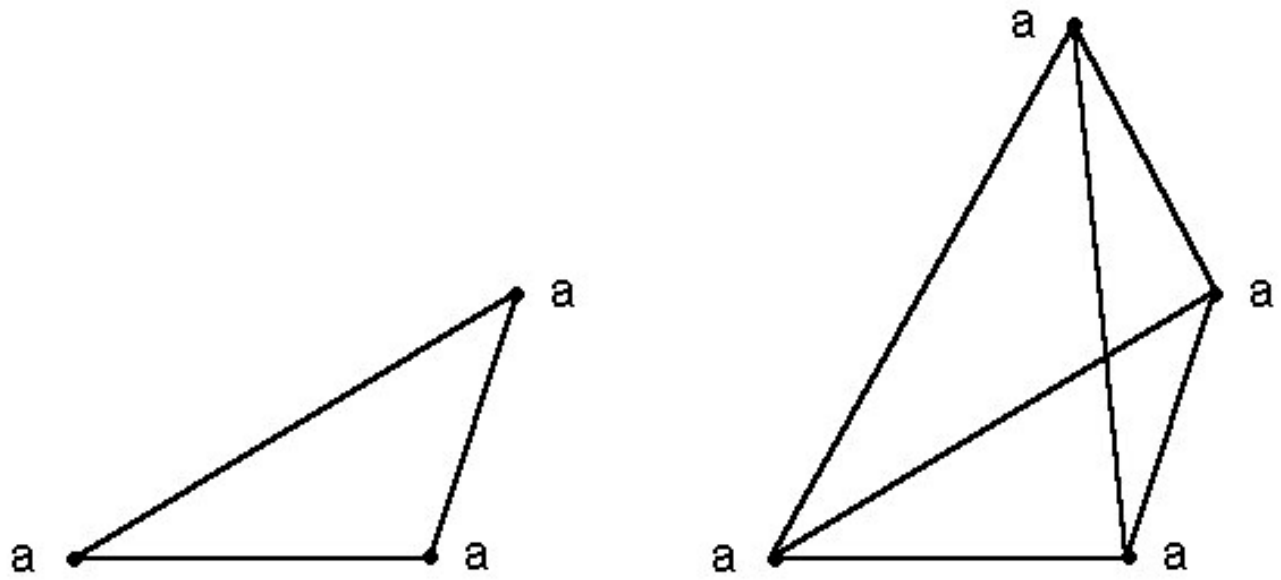
Planar graphs of degree 3,

Tournaments,

Interval graphs,

Graphs without induced P_5.

Example : Cliques have clique-width 2.



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

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

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

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

Proposition : (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 term using 2^{2k+1} labels. (Hlineny and Oum 2007)

This yields Fixed Parameter Cubic algorithms for many hard problems (MS property, (ex. 3-colorability), MS optimization function, (ex. distance), MS counting function, (ex. # of paths).