Recognizable sets of graphs : algebraic and logical aspects

Bruno Courcelle

LaBRI & *Université de Bordeaux* (ex -1 !)

References:

B.C. and J. Engelfriet : *Graph structure and monadic second-order logic*, Cambridge University Press, 2012.

B.C. and I. Durand: *Automata for the verification of monadic second-order graph properties*. J. Applied Logic 10(4): 368-409 (2012)

Summary

1. Recognizable & equational sets :

an algebraic setting for language theory.

2. A graph algebra with infinitely many operations & clique-width.

3. Monadic second-order logic, recognizability & fixed-parameter tractable model-checking.

4. Constructing automata.

Two ways of considering graphs

A graph is a *logical structure* ;

graph properties can be expressed by logical formulas

(FO = first-order, MSO = monadic second-order)

A graph is an *algebraic object*,

an element of an algebra of graphs

(similar to words as elements of monoids)

Graph algebras (finite graphs, up to isomorphism, as algebraic objects):

Equational sets → context-free sets of graphs, without needing to define graph rewritings

Recognizable sets → defined without automata (there is no "good" notion of graph automaton)

Algebraic understanding of combinatorial notions: tree-width, path-width, rank-width.

Clique-width: a width measure defined algebraically.

The two graph algebras that fit to MSO logic yield:

Hyperedge-replacement *gragras*, *tree-width*, FPT MSO_2 model-checking.

Vertex-replacement *gragras*, *clique-width*, FPT MSO-model-checking.

Below: the algebra **G** that yields *clique-width*.

The automata for model-checking MSO formulas on graphs of bounded clique-width are easier to build than those for graphs of bounded tree-width.

And : all results about model-checking of MSO2 formulas (with edge set quantifications) for graphs of bounded *tree-width* follow from the ones for graphs of bounded clique-width : we replace a graph G by its incidence graph Inc(G), we use : $cwd(Inc(G)) \leq twd(G) + 3$ (T. Bouvier, LaBRI) and we note that MSO2 over G reduces to MSO over Inc(G).

1. Recognizable & equational sets :

an algebraic setting for Language Theory.

Equational sets (generalizing context-free languages)

Equation systems = Context-Free (Graph) Grammars

For words, the set of context-free rules

$$X \to a\,X\,Y\ ; \quad X \to b\ ; \quad Y \to c\,Y\,Y\,X\ ; \quad Y \to a$$

is equivalent to the system of two equations:

$$X = a\,X\,Y \cup \{ b \}$$

$$Y = c\,Y\,Y\,X \cup \{ a \}$$

where X is the language generated by X (idem for Y and Y).

In arbitrary algebras we consider equation systems like:

$$X = f(\, k(\, X \,),\, Y \,) \cup \{ b \}$$

$$Y = f(\, Y \,,\, f(\, g(Y \,),\, m(\, X \,))) \cup \{ a \}$$

where :

f	is a binary operation,
g, k, m	are unary operations,
a, b	denote basic objects.

An *equational set* is a component of the least solution of such an equation system. This notion is well-defined in any algebra.

The general algebraic setting

F : a finite or *countably infinite* set of operation symbols with fixed arities, called a signature.

M = < M, $(f_{\mathbf{M}})_{f \in F}$ > : an F-algebra.

All signatures and algebras are finite or countable and effectively given (that is : *encoded by finite words in a computable way*).

Examples : (*words, trees and graphs will be finite*).

Free monoid of words,

free algebra of terms,

algebra of terms with associative and/or commutative operations,

monoid of traces (words with partial commutations).

Recognizable sets : finite signatures

$\mathbf{M} = \langle M, (f_{\mathbf{M}})_{f \in F} \rangle$: an F-algebra where F is *finite* .

Definition : $L \subseteq M$ is (**M**-)*recognizable* if it is a union of equivalence classes of a finite congruence $\approx$ on **M**.

Congruence = equivalence relation such that :

$m \approx m'$ and $p \approx p' \Rightarrow f_{\mathbf{M}}(m,p) \approx f_{\mathbf{M}}(m',p')$.

Finite means that $\approx$ has finitely many classes ($M / \approx$ is finite).

Equivalently, $L = h^{-1}(D)$ for a homomorphism $h : \mathbf{M} \to \mathbf{A}$, where **A** is a *finite* F-algebra and $D \subseteq A$.

REC(**M**) = the recognizable subsets of **M**. This notion is relative to **M** (not only to the underlying set M). Well-defined in every algebra, without any automaton.

Property	EQ (finite sig.)	REC (finite sig.)
closure for $\cup$	yes	yes
closure for $\cap$, -	no	yes
$L_{eq} \cap K_{rec} \in$ EQ		yes
closure for hom, operations of F	yes	no
closure for hom^{-1}	no	yes
comparison		$\subseteq$ EQ (algebra generated by F)
membership (input given by a term)	undecidable	decidable
emptyness	decidable	decidable

For graphs : The two robust (in a precise logical and algebraic sense) graph algebras have *countably infinite* signatures.

No problem for equational sets because each equation system uses a finite signature, hence they are equational in a finitely generated subalgebra.

Recognizability requires an adapted definition. Below in abstract form; the application to graphs will come soon.

Recognizable sets : infinite signatures

$\mathbf{M} = \langle M, (f_{\mathbf{M}})_{f \in F} \rangle$: an F-algebra where F is *countably infinite*.

F, M, the functions $f_{\mathbf{M}}$ are *effectively given* (encoded by integers in a computable way).

Each $m \in M$ has a type $\pi(m)$ in a countable, effectively given set.

π is a homomorphism :

$$\pi(f_{\mathbf{M}}(m,m')) = {}^{\pi}f(\pi(m), \pi(m')) \text{ for some function } {}^{\pi}f \text{ on types.}$$

For defining the recognizability of a set L, we require that $\approx$ is

type preserving : $m \approx m' \Rightarrow \pi(m) = \pi(m')$,

locally finite : it has finitely many classes of each type,

and L is a union of classes (possibly of different types).

Property	EQ	REC finite sign.	REC infinite sign.
closure $\cup$	yes	yes	yes
closure $\cap$, -	no	yes	yes
$L_{eq} \cap K_{rec} \in$ EQ		yes	yes
hom, ops of F	yes	no	no
hom^{-1}	no	yes	yes
comparison		$\subseteq$ EQ	incomparable with EQ
membership (given by a term)	undecidable	decidable	decidable
emptyness	decidable	decidable	undecidable

2. A graph algebra & clique-width.

Cographs : Undirected graphs generated by:

$\oplus$ (*disjoint union*) and $\otimes$ (*complete join*) from

a : vertex without edges (*up to isomorphism*);

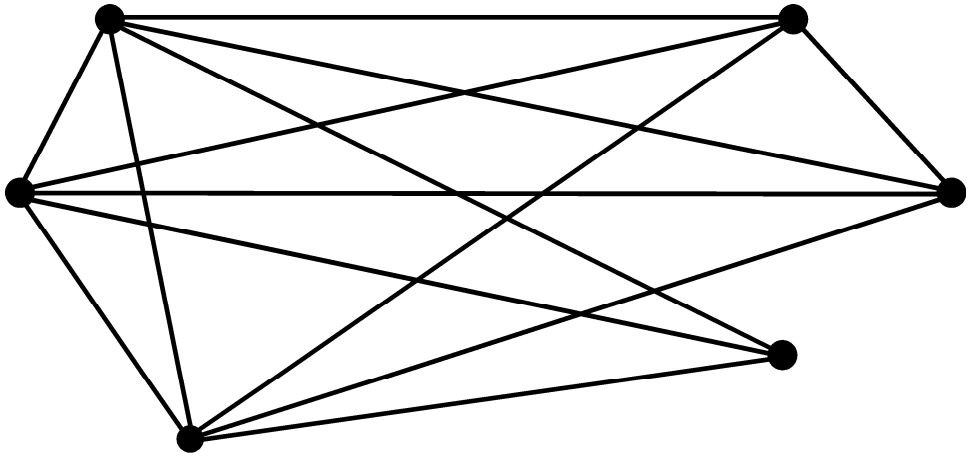
$\otimes$ is defined by : $G \otimes H = G \oplus H$ plus all

undirected edges between G and H,

Cographs are defined by the equation : $C = C \oplus C \cup C \otimes C \cup \{ \mathbf{a} \}$

Example :

$(\mathbf{a} \otimes \mathbf{a} \otimes \mathbf{a}) \otimes ((\mathbf{a} \otimes \mathbf{a}) \oplus \mathbf{a})$



The graph algebra **G** (that yields *clique-width*)

Graphs are *simple*, directed or not.

We use labels: *a*, *b*, *c*, ..., *d*.

Each vertex has one label ; the labelling defines a partition of the vertex set.

A vertex labelled by *a* is an *a-port.*

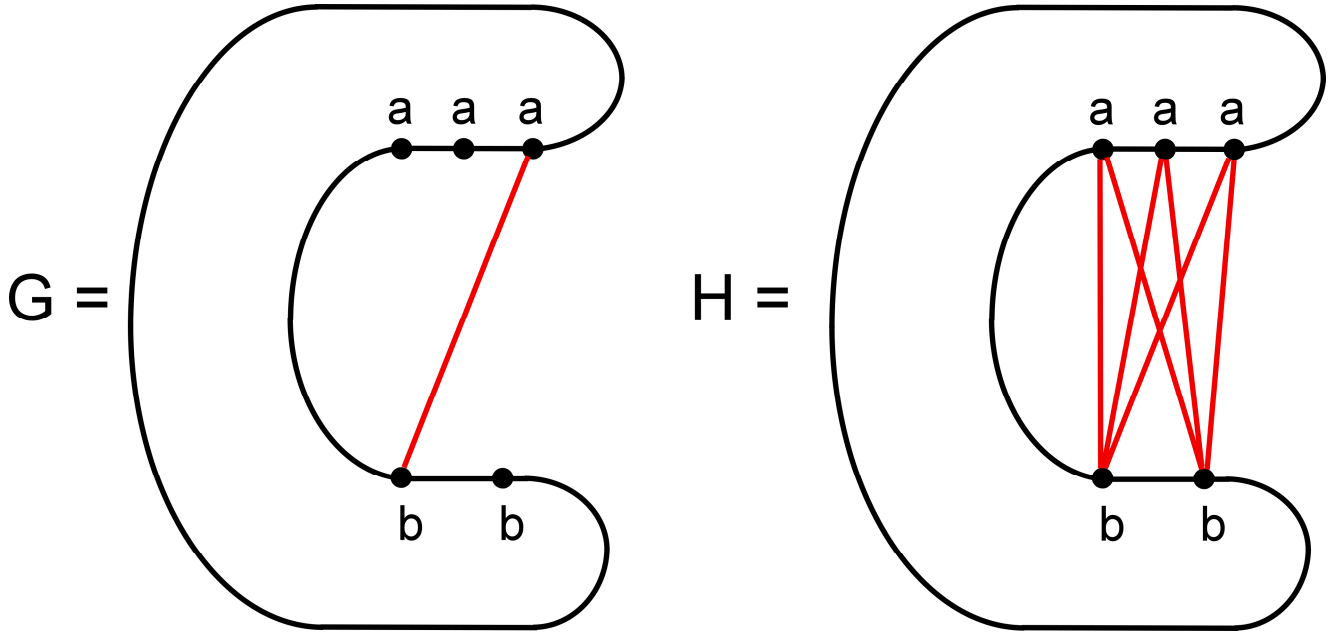
One binary operation : *disjoint union* : $\oplus$

If G and H are not disjoint, we replace H by an isomorphic disjoint copy to define $G \oplus H$. Hence $G \oplus H$ is well-defined *up to isomorphism*.

No such problem in a “decomposition approach”.

Unary operations: Edge-addition

Addition of undirected edges: $Add_{a,b}(G)$ is G augmented with edges *between* every *a*-port and every *b*-port.



H = $Add_{a,b}(G)$; only 5 edges added

The number of added edges depends on the argument graph.

Addition of directed edges: $\overrightarrow{Add}_{a,b}(G)$ is G augmented with edges *from* every *a*-port to every *b*-port.

Vertex relabellings :

$Relab_{a \longrightarrow b}(G)$: = G where each *a -port* is made into a *b-port.*

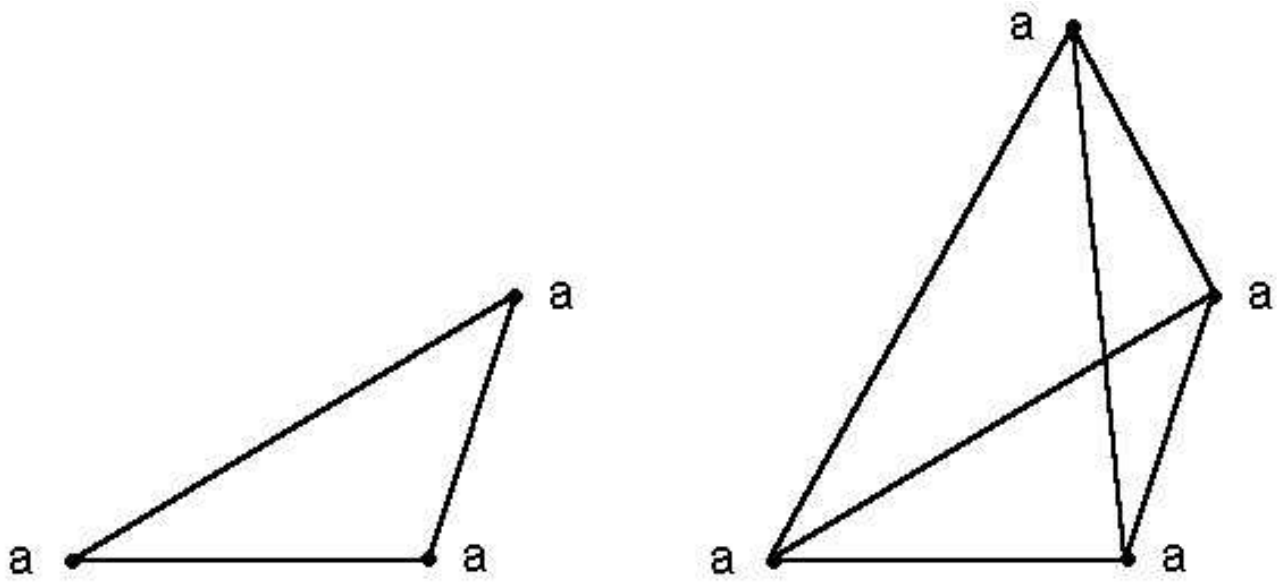
Nullary operations: basic graphs are those with a single vertex.

Every graph with n vertices is the value of a term using $\leq$ n labels.

A graph G has clique-width $\leq k \Leftrightarrow$ it can be constructed from basic graphs with the operations $\oplus$, $Add_{a,b}$ (or $\overrightarrow{Add}_{a,b}$) and $Relab_{a \longrightarrow b}$ by using $\leq k$ labels.

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

Example : Cliques have clique-width 2 (and *unbounded tree-width*)



K_n is defined by t_n where t_{n+1} = $Relab_b \longrightarrow_a (Add_{a,b}(t_n \oplus \mathbf{b}))$

Cliques are defined by the equation :

$$K = Relab_b \longrightarrow_a (Add_{a,b}(K \oplus \mathbf{b})) \cup \mathbf{a}$$

Examples of graphs of bounded clique-width:

An undirected graph is a cograph $\Leftrightarrow$ it has clique-width at most 2.

Distance hereditary graphs have clique-width at most 3.

Cactuses (or *cactii* ?) : biconnected components are cycles.

Examples of graphs of unbounded clique-width:

Planar graphs (even of degree 3),

Interval graphs.

Fact : Unlike tree-width, clique-width is sensible to edge directions :

Cliques have clique-width 2 but tournaments have *unbounded clique-width*.

3. Monadic second-order (MSO) logic & recognizability.

Graphs *as* *logical* *structures* :

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

$edg_G(u,v) \Leftrightarrow$ there is an edge $u \rightarrow v$ (or $u - v$).

parallel edges are not distinguished.

Monadic second-order logic = First-order logic extended with (quantified) set variables denoting sets of vertices.

"A set of edges" is (here) a binary relation over V_G.

Typical MSO properties : k-colorability, transitive closure, properties of paths, connectivity, planarity

Examples : $G = (V_G , edg_G(.,.))$, undirected

Syntax is clear ; shorthands are used (example $X \cap Y = \varnothing$).

(1) *G is 3-colorable :* NP-*complete property.*

$$\exists X,Y \, (X \cap Y = \varnothing \; \wedge$$
$$\forall u,v \, \{ edg(u,v) \Rightarrow$$
$$[(u \in X \Rightarrow v \notin X) \wedge (u \in Y \Rightarrow v \notin Y) \wedge$$
$$(u \notin X \cup Y \Rightarrow v \in X \cup Y) \;]$$
$$\})$$

(2) *G is not connected :*

$$\exists Z \, (\exists x \in Z \; \wedge \; \exists y \notin Z \; \wedge \; (\forall u,v \, (u \in Z \; \wedge \; edg(u,v) \; \Rightarrow \; v \in Z) \;)$$

(3) *Transitive and reflexive closure* : **TC**(R ; x, y) :

$\forall$ X { "X is R-closed" $\wedge$ x $\in$ X $\Rightarrow$ y $\in$ X }

where "X is R-closed" is defined by :

$\forall u,v\ (u \in X \wedge R(u,v) \Rightarrow v \in X)$

The relation R can be defined by a formula. Here :

$\forall x,y\ (x \in Y \wedge y \in Y \Rightarrow$ **TC**("$u \in Y \wedge v \in Y \wedge edg(u,v)$" ; x, y)

Y is free in R. This formula expresses that G[Y] is connected.

(4) *Minors* : G contains a fixed graph H as a minor with $V_H = \{1,\ldots,p\}$:

there exist disjoint sets of vertices $X_1,\ldots, X_p$ in G

such that each $G[X_i]$ is connected and,

whenever i – j in H, there is an edge between X_i and X_j.

(5) *Planarity* is expressible : no minor K_5 or $K_{3,3}$ (Kuratowski - Wagner).

(6) *Has a cycle* (for a graph without loops) :

$\exists x,y,z$ [edg(x,y) $\wedge$ edg(y,z) $\wedge$ “there is a path from x to z avoiding y”]

(7) Is a tree : connected without cycles.

Non-expressible properties

- G is isomorphic to $K_{p,p}$ for some p : checking equal cardinality of two sets “needs” quantification over binary relations in order to find a bijection.

- G has a nontrivial automorphism, or has all vertices of same degree.

- Hamiltonicity needs edge-set quantifications.

Remark : Adding an *equicardinality* set predicate would spoil everything.

Two problems for a class *C* of finite graphs and a logic

Decidability : does a given sentence hold in some (or all) graphs of *C* ?

Model-checking is decidable : what is its time or space complexity ?

Language, class	*Decidability*	*Model-checking*
FO, all graphs	undecidable	polynomial time
MSO, clique-width $\leq$ k	decidable	cubic time
MSO, unbounded cwd.	undecidable	Conjecture : not FPT (*)

(*) A related fact is proved by S. Kreutzer (LICS 2010) for unbounded tree-width and MSO formulas with edge quantifications. (The exact statement is very technical.)

Fixed-Parameter Tractability (FPT) for model-checking

An algorithm is *FPT* if it takes time $f(k).n^c$ for some fixed function f and constant c. The size of the input is n. The value k is a *parameter* of the input :

degree, diameter, tree-width, clique-width, etc.

The "hard part" of the time complexity depends on some function f (arbitrary in the definition; in practice, it must be "limited"; the algorithm is then usable for small values of k).

Example : 3-colorability is NP-complete, even for planar graphs of degree ≤ 4 (Dailey, 1980). *Degree* is *not* a good parameter, but *tree-width* and *clique-width* are.

MSO logic & recognizability : main results

For a sentence φ, Mod(φ) is the set of finite models of φ.

1. Recognizability theorem : For every MSO sentence φ, Mod(φ) is recognizable in the graph algebra **G**. The *type* $\pi(G)$ of G is the set of its vertex labels.

2. Weak recognizability theorem : For every MSO sentence φ, for every k, Mod(φ) $\cap$ CWD($\leq$ k) is recognizable in **G**[F_k], the subalgebra of **G** generated by the finite signature F_k of operations using labels 1, …, k.

3. The recognizability theorem is *not a consequence* of the weak one. (The set *K* of *kytes*, the nXn grids with a "tail" of length n, is not recognizable but $K \cap CWD(\leq k)$ is finite for each k, hence, recognizable).

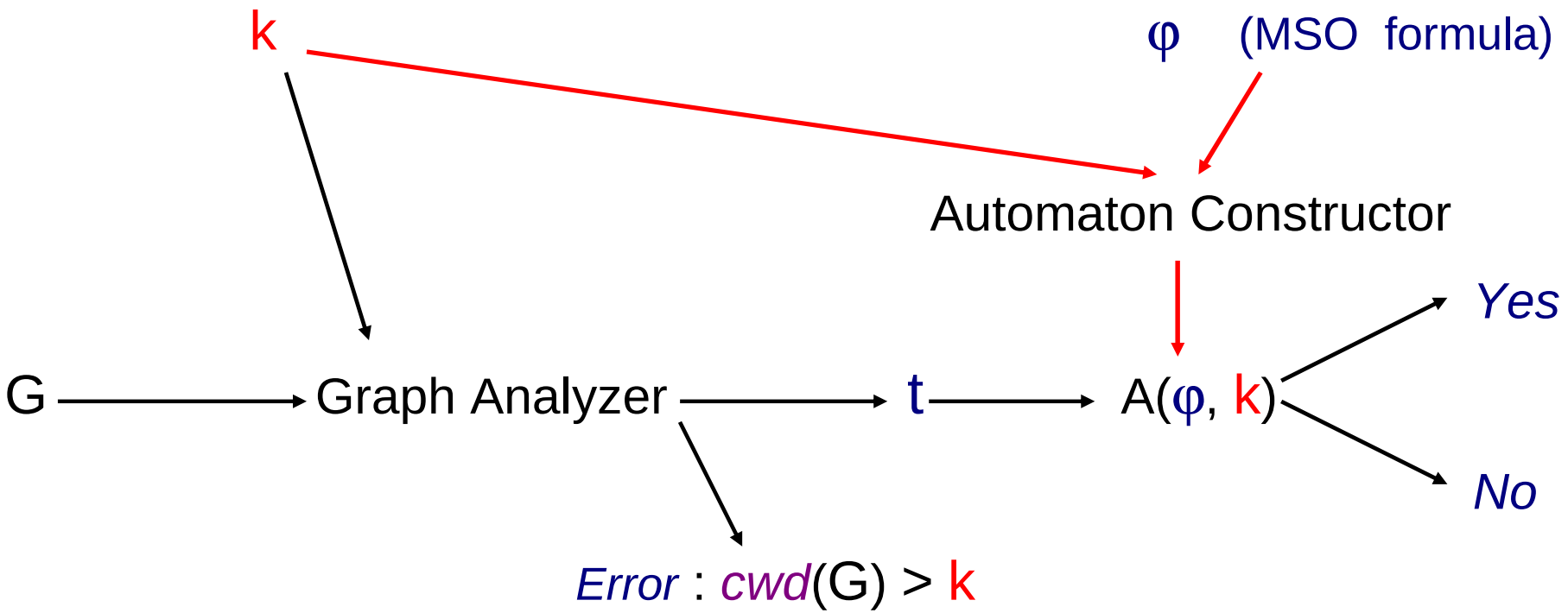
4. Every set of square grids is recognizable. As there are uncountably many such sets, there is no characterization of recognizable sets of graphs by logical formulas or automata.

5. For algorithmic purposes, the weak theorem is enough. Its "Büchi-style" proof is given below, in terms of finite, implementable automata.

6. Proofs of the full theorem:

- either a “Feferman-Vaught-Shelah-style” proof manipulating “theories” (the set of formulas of quantifier-height $\leq$ k valid in a graph) from which a congruence is defined (two graphs have same theory).
- either a “Büchi-style” proof constructing infinite, “fly-automata” (“automates programmés”) that are still implementable.

4. Construction and use of automata



Steps ⟶ are done "once for all", independently of G

A(φ,k): finite automaton on terms t.

Construction of A(φ, k) ("Büchi-style" proof).

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

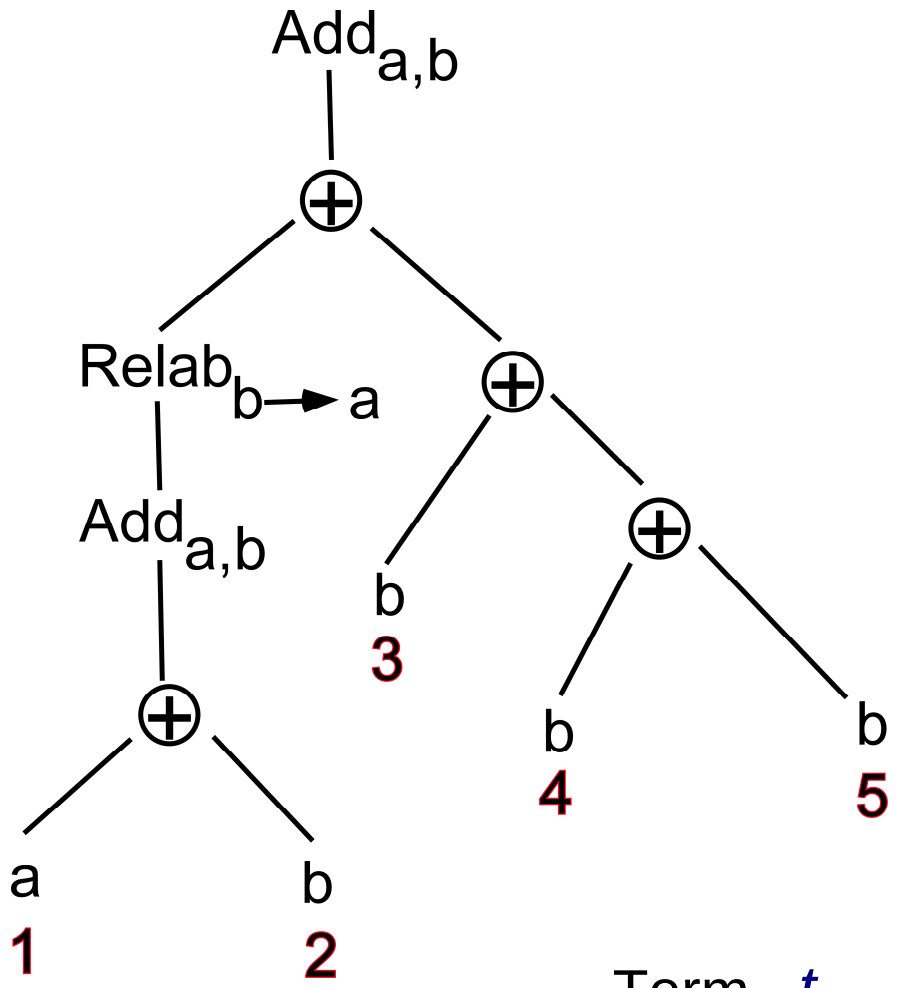
F (= F_k) = the corresponding set of operations and constants :

a , ∅ , ⊕ , $Add_{a,b}$, $\overrightarrow{Add}_{a,b}$, $Relab_{a \longrightarrow b}$

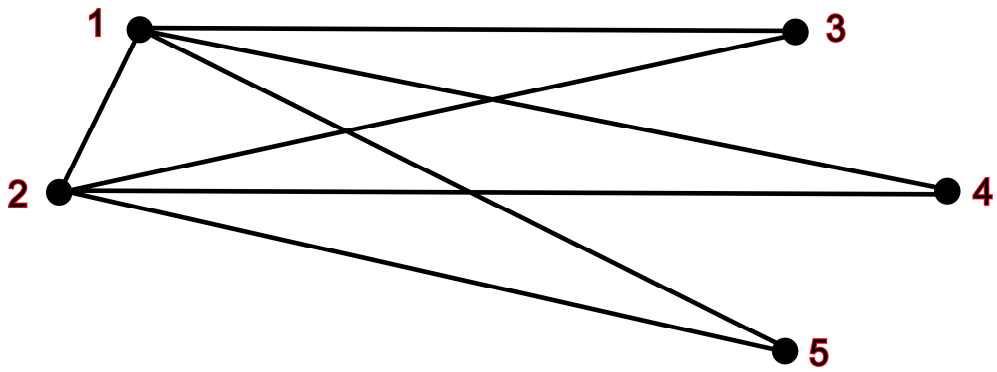
G(*t*) = the graph defined by a term *t* in **T**(F).

Its vertices are (in bijection with) the occurrences of the nullary symbols in *t* that are not ∅.

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

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

(only nullary symbols are modified);

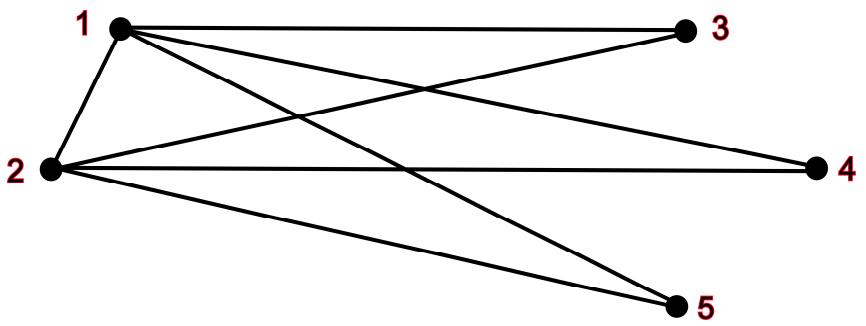
2) a term s in $\mathbf{T}(F^{(n)})$ encodes a term t in $\mathbf{T}(F)$ and an assignment of sets $V_1,\ldots,V_n$ to the set variables $X_1,\ldots,X_n$:

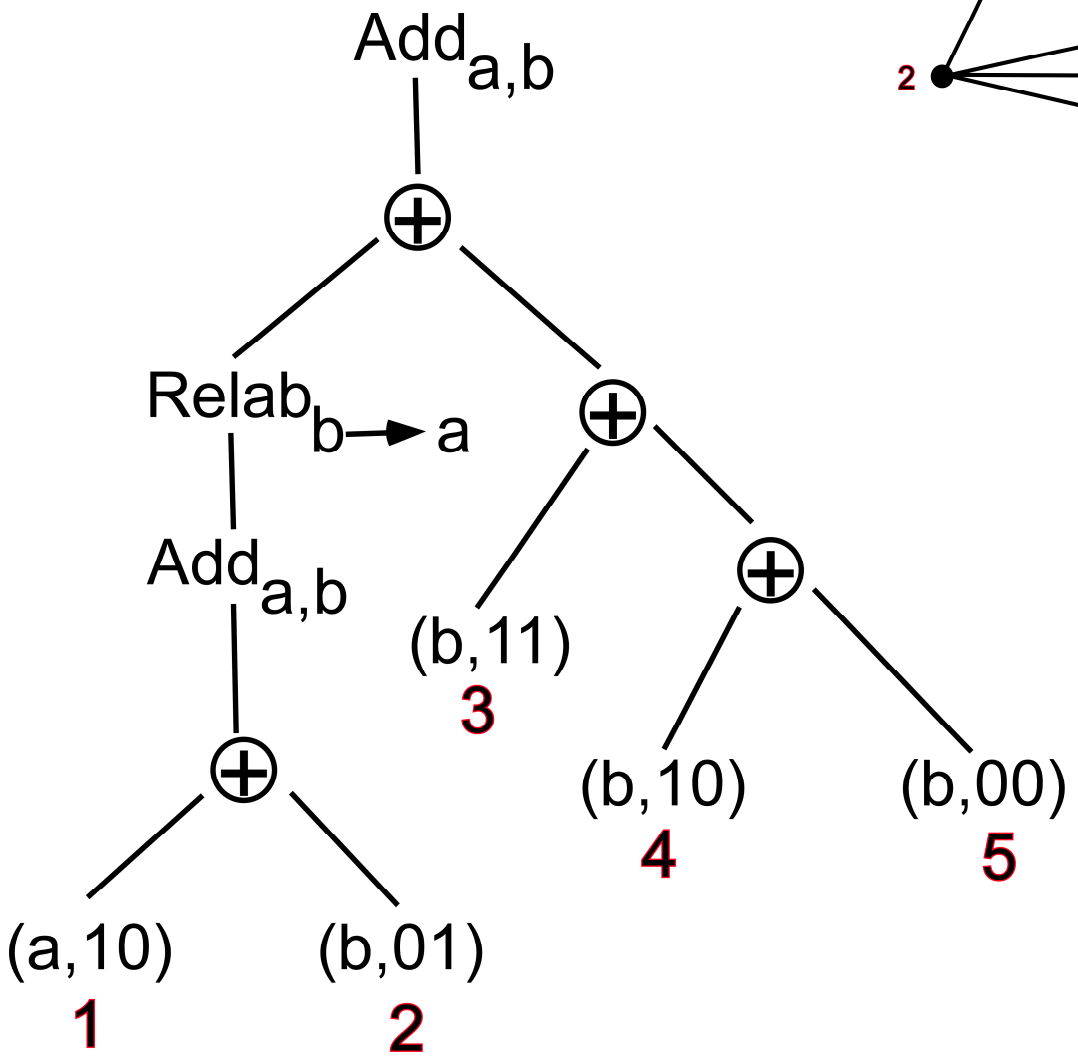
if u is an occurrence of $(\mathbf{a}, (w_1,..,w_n))$, then

$w_i = 1$ 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:

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

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

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 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 $\forall$

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

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

where pr is the *projection* that eliminates the last Boolean;

→ a *non-deterministic* automaton.

Tools using inverse homomorphisms of automata:

from $A(\varphi(X_1, X_2), k)$, we get $A(\varphi(X_4, X_3), k)$,

from $A(\varphi(X_1, X_2), k)$, we get $A(\varphi (X_3, X_1 \cup (X_2 \setminus X_4)), k)$,

from $A(\varphi, k))$, we get $A(\varphi [X_1] , k)$.

Skip 2 slides

Some tools for constructing automata

Substitutions and inverse images (*cylindrifications*).

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

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

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

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

This construction preserves determinism and the number of states.

2) From $A(\varphi(X_1, X_2), k)$, we get $A(\varphi(X_3, \overbrace{X_1 \cup (X_2 \setminus X_4)}^{\textit{Set term}}), k)$ 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:

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

The inductive construction (continued) :

For atomic formulas and basic graph properties $\varphi(X_1,\ldots,X_n)$, we build *complete deterministic* automata over $F^{(n)}$ for recognizing the set of terms : $t * (V_1,\ldots,V_n)$ in $L(\varphi(X_1,\ldots,X_n))$.

Intuition : in all cases, the *state* reached 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 of automata with few states :

respectively 2, 2, 3, q, q+1 states.

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

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

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

Vertex labels belong to 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 of 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

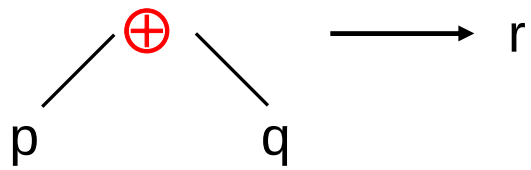
Transition rules

For the constants based on **a** :

(**a**,00) → 0 ; (**a**,10) → a(1) ; (**a**,01) → a(2) ; (**a**,11) → Error

For the binary operation ⊕:

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

We get also the Recognizability Theorem:

$A(\varphi, k)$ is a subautomaton of $A(\varphi, k+1)$. The deterministic automata $A(\varphi, k)$ can be merged into a single *infinite* deterministic automaton $A(\varphi)$ over F (the countable signature of all graph operations).

The state $q(t)$ reached by $A(\varphi)$ on any $t \in T(F)$ belongs to a finite set built from $\pi(G(t))$, the type of the graph $G(t)$ defined by t.

Example of such a set : $\{Ok, Error\} \cup P(\pi(G(t)) \times \pi(G(t)))$.

If t and t' define isomorphic graphs, then $q(t) = q(t')$.

The (global) congruence proving recognizability can be taken:

$G \approx G' \leftrightarrow q(t) = q(t')$ where $G = G(t)$ and $G' = G(t')$

Practical difficulties and remedies.

Parsing : 1. Checking if a graph has clique-width $\leq$ k is NP-complete (with k in the input ; Fellows *et al.*)

2. The *cubic approximate* parsing algorithm (by Oum *et al.*) based on *rank-width* is difficult to implement.

3. Szeider and Heule reduce to SAT the computation of clique-width and get exact values (and the corresponding terms) for graphs with at most 30 vertices and clique-width at most 12.

4. For certain classes of graphs of bounded clique-width defined by forbidden induced subgraphs, optimal clique-width terms can be constructed in polynomial time, by using, in many cases, *modular decomposition*.

5. Heuristics remain to be found.

Sizes of automata :

1. 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 .

2. There is no alternative construction giving a fixed bound on nestings of exponentiations (Meyer & Stockmeyer, Frick & Grohe).

3. 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.

An issue : *Fly-automata* (to be presented by Irène Durand)

States and transitions are not listed in huge tables : they are *specified* (in uniform ways for all k) by "small" programs. These automata can be nondeterministic.

Example of a 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. (type(H) = set of labels of its vertices)

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.

Using fly-automata works for formulas without quantifier alternation but that can use "new" atomic formulas for "basic" properties

Examples : p-*acyclic colorability*

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

(all $i < j$; set terms $X_i \cup X_j$ avoid some quantifications).

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

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

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

Skip : Only for questions.

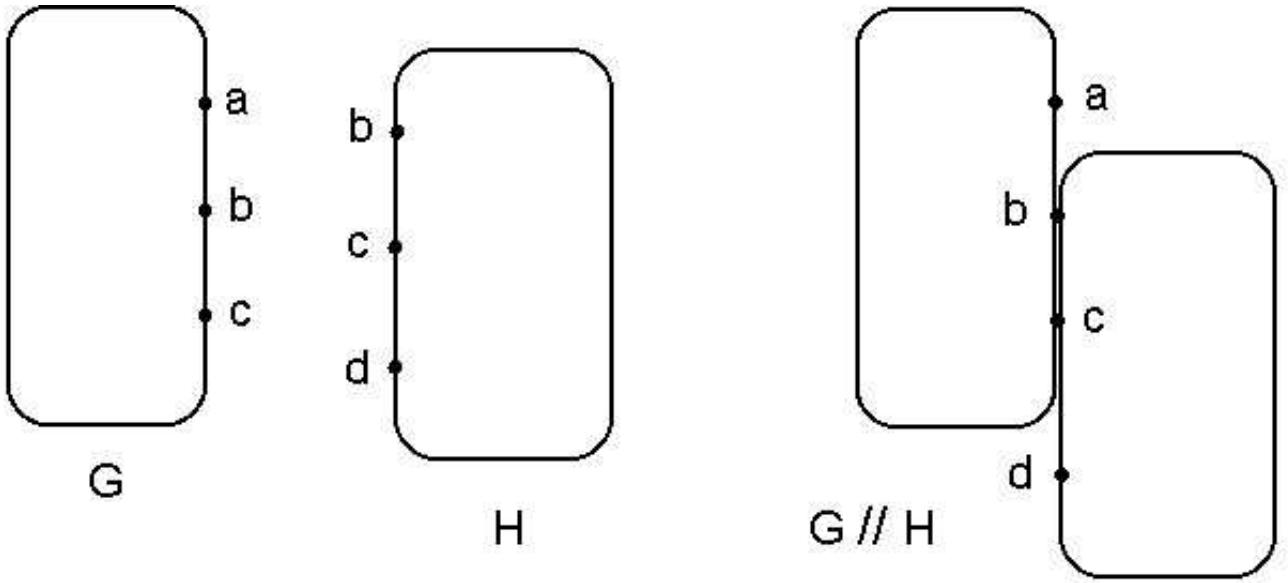
Appendix : Graph operations that characterize tree-width

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

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, we use 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 :

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

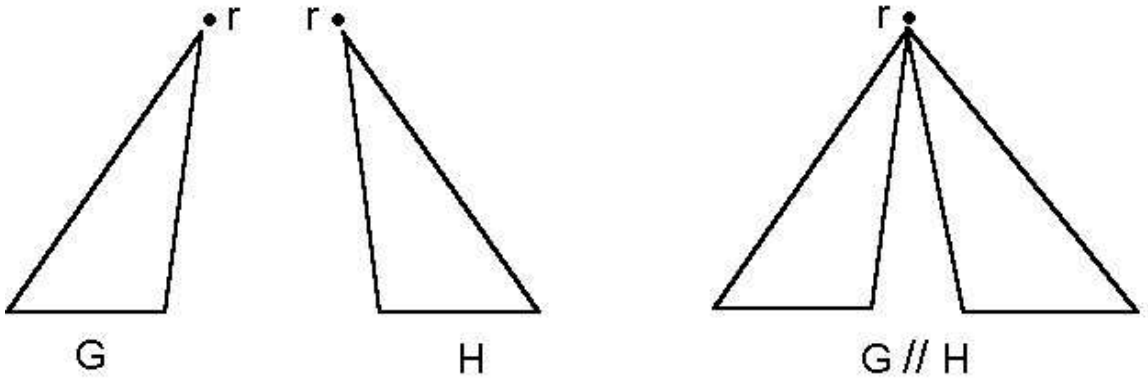
Nullary operations denote *basic graphs* : 1-edge graphs, isolated vertices.

Terms over these operations *define* (or *denote*) graphs (with or without sources). They can have parallel edges.

Example : Trees

Constructed with two source labels, *r* (root) and *n* (new root).

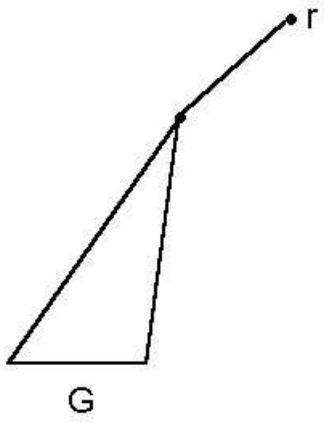
Fusion of two trees at their roots :



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

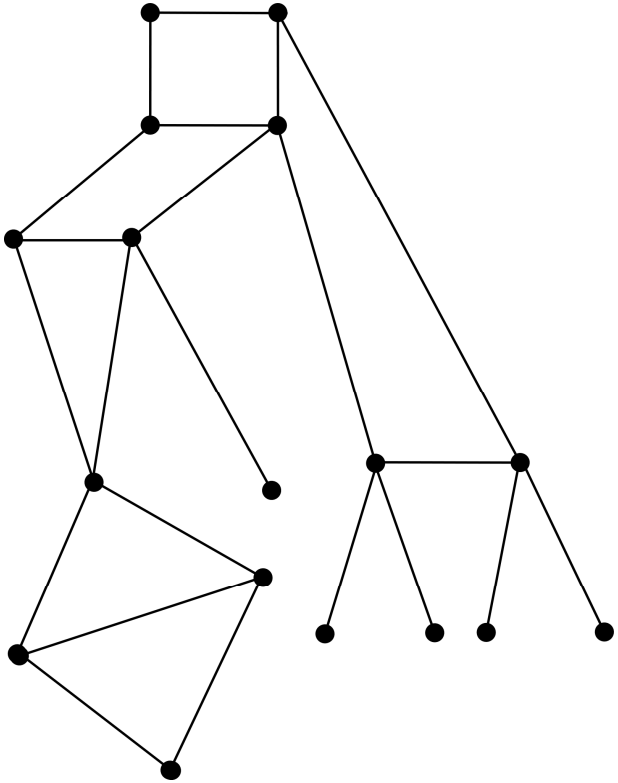
e = ***r*** ●——● ***n***

$Ren_{n \leftrightarrow r}$ ($Forget_r$ (G // e))

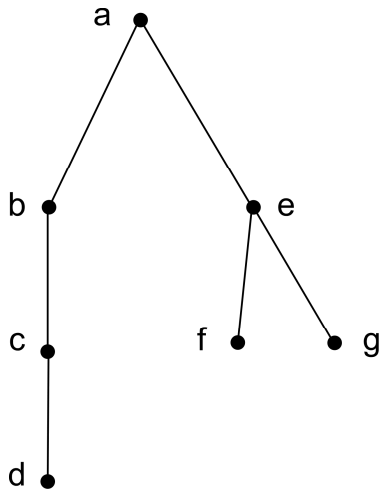


Trees are defined by : T = T // T ∪ extension(T) ∪ ***r***

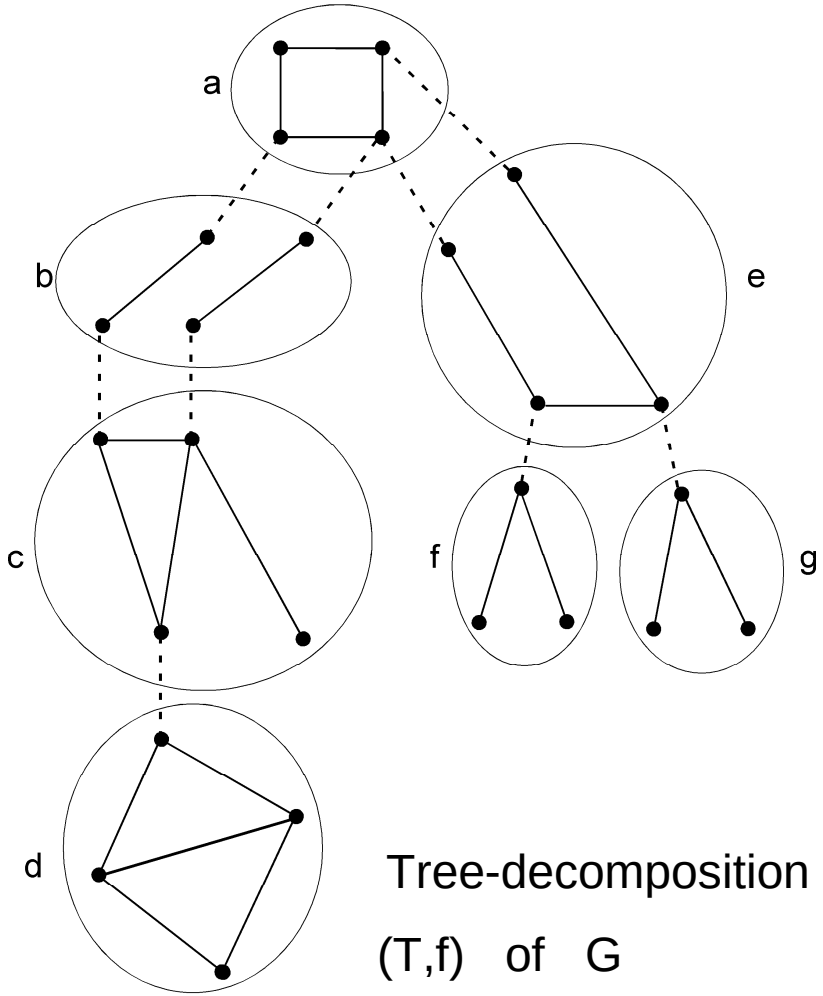
Relation to tree-decompositions and tree-width



Graph G



Tree T



Tree-decomposition (T,f) of G

Dotted lines - - - - link *copies* of a same vertex.

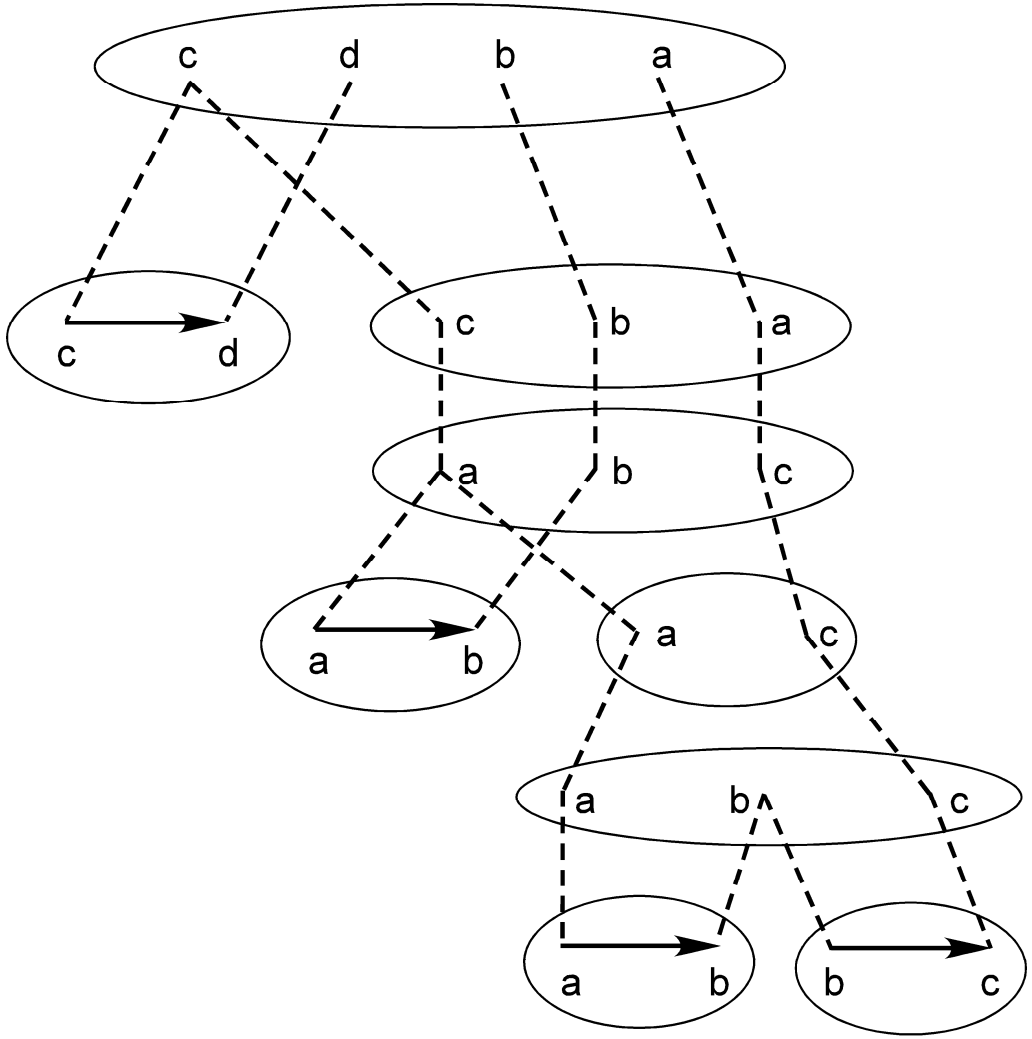
Width = max. size of a box -1. Tree-width = min. width of a tree-dec.

Proposition: A graph has tree-width $\leq k$ $\Leftrightarrow$ it can be constructed from edges by using the operations $//$, $Ren_{a \leftrightarrow b}$ and $Forget_a$ with $\leq k+1$ labels a,b,….

Proposition : Bounded tree-width implies bounded clique-width (cwd(G) $\leq 2^{2twd(G)+1}$ for G directed), but not *conversely*.

From an algebraic expression to a tree-decomposition

Example : cd // $Ren_{a \leftrightarrow c}$ (ab // $Forget_b$(ab // bc)) (ab denotes an edge from a to b)



The tree-decomposition associated with this term.