Monadic second-order logic for graphs.

Algorithmic and language theoretical applications

Bruno Courcelle

Université Bordeaux 1, LaBRI, and Institut Universitaire de France

Reference : Graph structure and monadic second-order logic,
book to be published by Cambridge University Press, readable on :
http://www.labri.fr/perso/courcell/ActSci.html

History : Confluence of 4 independent research directions, now intimately related :

1. Fixed-Parameter Tractable algorithms for parameters reflecting hierarchical structurings : tree-width, clique-width. This research started with case studies for series-parallel graphs, cographs, partial k-trees.

2. Extension to graphs of the main concepts of Formal Language Theory : grammars, recognizability, transductions, decidability questions

3. Excluded minors and related notions of forbidden configurations (matroid minors, « vertex-minors »).

4. Decidability of Monadic Second-Order logic on classes of finite graphs.

Two ways of considering graphs

1) A graph (finite, up to isomorphism) is an *algebraic object*,

an element of an algebra of graphs

(Similar to words, elements of monoids)

2) A graph is a *logical structure* ;

graph properties can be expressed by logical formulas

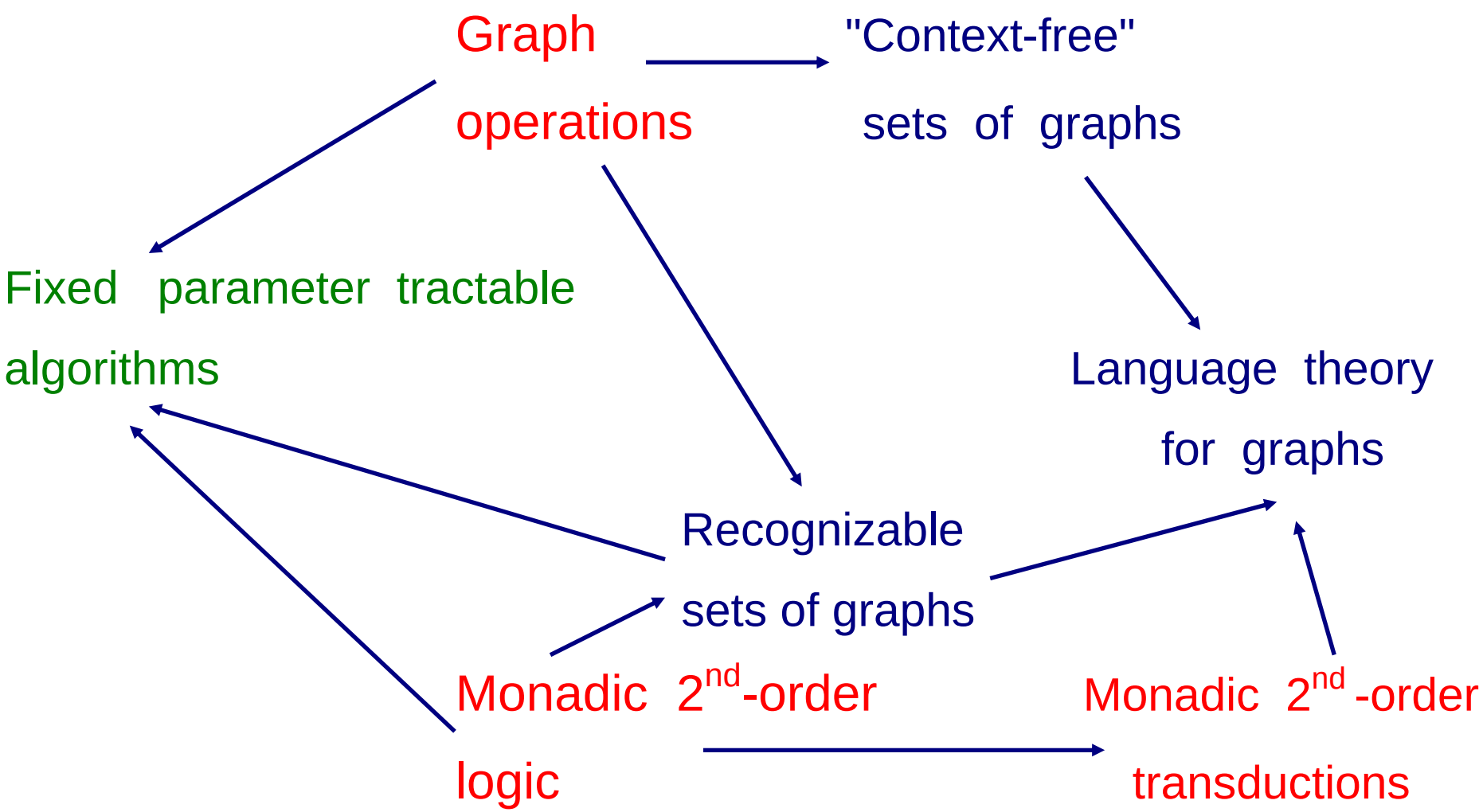
(FO = first-order, MS = monadic second-order, SO = second-order)

Consequences:

a) *Language Theory* concepts extend to graphs

b) *Algorithmic meta-theorems*

An overview chart



Key concepts of Language Theory and their extensions

Languages	*Graphs*
Algebraic structure : monoid $(X^*,\cdot,\varepsilon)$	Algebras based on graph operations : $\oplus$, $\otimes$, // quantifier-free definable operations Algebras : **HR**, **VR**
Context-free languages : Equational subsets of $(X^*,\cdot,\varepsilon)$	Equational sets of the algebras **HR, VR**
Regular languages : Finite automata $\equiv$ Finite congruences $\equiv$ Regular expressions $\equiv$	Recognizable sets of the algebras **HR, VR** defined by finite congruences
$\equiv$ Monadic Second-order definable sets of words or terms	$\cup$ Monadic Second-order definable sets of graphs
Rational and other types of transductions	Monadic Second-order transductions

Summary

1. Context-free sets defined by equation systems
2. Two graph algebras; tree-width and clique-width.
3. Recognizability : an algebraic notion.
4. Monadic second-order sentences define recognizable sets.
5. Fixed-parameter tractable algorithms : constructions of automata
6. Monadic second-order transductions.
7. Robustness results : preservation of classes under direct and inverse monadic second-order transductions. Short proofs in graph theory. (black= graph theory)
8. Logic and graph structure theory : Comparing encoding powers of graph classes *via* monadic second-order transductions
9. Graph classes on which monadic second-order logic is decidable
10. Open questions

1. Equational sets (generalization of context-free languages)

Equation systems = Context-Free (Graph) Grammars
in an algebraic setting

In the case of words, the set of context-free rules

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

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 (→ in graph 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 on graphs,
a, b	denote basic objects (graphs up to isomorphism).

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

The general algebraic setting

F : a finite set of operation symbols with (fixed) arities, called a signature

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

$P(\mathbf{M})$ its power-set algebra with domain $P(M)$ and operations extended to sets : $f_{P(\mathbf{M})}(A,B) = \{ f_{\mathbf{M}}(a,b) \ / \ a \in A, b \in B \}$.

Equation systems of the general form :

$S = < X_1 = p_1, \ldots, X_n = p_n >$

$X_1,\ldots,X_n$ are unknowns (ranging over sets)

$p_1,\ldots,p_n$ are polynomials for example :

$$f(k(X_1),X_2) \cup f(X_2, f(g(X_3), X_1))) \cup c$$

Its solutions are the fixed-points of the (recursive) equation :

$\mathbf{X} = S_{P(M)}(\mathbf{X})$ (1) where $\mathbf{X} = (X_1,\ldots,X_n)$

$S_{P(\mathbf{M})}(\mathbf{X}) : = (p_1{}_{P(\mathbf{M})}(\mathbf{X}) ,\ldots, p_n{}_{P(\mathbf{M})}(\mathbf{X}))$

The set $P(M)^n$ ordered by component-wise inclusion is ω-complete, the mapping $S_{P(\mathbf{M})}$ is monotone and ω-continous, hence Equation (1) has a least solution defined by iteration :

$\mu\mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X}) = \bigcup_{i \geq 0} S_{P(\mathbf{M})}(\mathbf{X})^i(\varnothing, \ldots,\varnothing)$ (increasing sequence)

An *equational set* of **M** is a component of $\mu\mathbf{X}. S_{P(\mathbf{M})}(\mathbf{X})$

for some equation system S

Classical examples

Algebra	Equational sets
$\langle A^*, \cdot, \varepsilon, a,b,\ldots,d \rangle$	Context-free languages
$\langle A^*, \varepsilon, (\lambda u \in A^*.ua)_{a \in A} \rangle$	Regular languages
$\mathbf{T}(F)$, terms over F, (*initial F-algebra*)	Regular sets of terms
$\langle \mathbf{N}^k, +, (0,\ldots,0), \ldots (0,\ldots,1,0,\ldots,0) \ldots \rangle$	Semi-linear sets =

finite unions of sets $\{ u + n_1.v_1 + \ldots + n_p.v_p \mid n_1,\ldots,n_p \in \mathbf{N} \}$

for $u, v_1, \ldots, v_p \in \mathbf{N}^k$

Properties of context-free languages valid at the algebraic level

1) If K and L are equational sets of M, so are $K \cup L$ and $f_{P(\mathbf{M})}(K,L)$.

2) The emptiness of an equational set is decidable

Proof : A system S can be solved in $P(\mathbf{T}(F))$ where $\mathbf{T}(F)$ is the F-algebra of terms over F.

"Transfer" of least fixed-points by homomorphisms :

If $h : \mathbf{M'} \rightarrow \mathbf{M}$ then $h(\mu\mathbf{X}.S_{P(\mathbf{M'})}(\mathbf{X})) = \mu\mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X})$

Hence, $\mu\mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X}) = val_{\mathbf{M}}(\mu\, \mathbf{X}.S_{P(\mathbf{T}(F))}(\mathbf{X}))$ ($val_{\mathbf{M}}$ = value mapping : $\mathbf{T}(F) \rightarrow \mathbf{M}$)

Each component of $\mu\mathbf{X}.S_{P(\mathbf{T}(F))}(\mathbf{X})$ is a context-free language (terms are words written in Polish prefix notation). Emptiness can be checked.

3) If **M** is "effectively given" and the components of $\mu\mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X})$ are *all* finite sets, $\mu\mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X})$ can be computed (by straightforward iteration and by stopping as soon as $S_{P(\mathbf{M})}(\mathbf{X})^{i}(\varnothing,\ldots) = S_{P(\mathbf{M})}(\mathbf{X})^{i+1}(\varnothing,\ldots)$).

4) Finiteness test (with some natural "size" conditions).

5) For every context-free language L over k letters : a,…,d, the set of k-tuples $(|u|_a, \ldots., |u|_d)$ in $\mathbf{N}^k$ for all u in L is semi-linear (using transfer theorem for least fixed-points; "Parikh's Theorem").

Here : each function f has a *weight* w(f) in $\mathbf{N}^k$, the weight w(t) of a term t is the sum of weights of its symbols ; if L is equational w(L) is semi-linear.

2. The graph algebras HR and VR

We define two graph algebras → Equational sets of graphs, two generalizations of context-free languages.

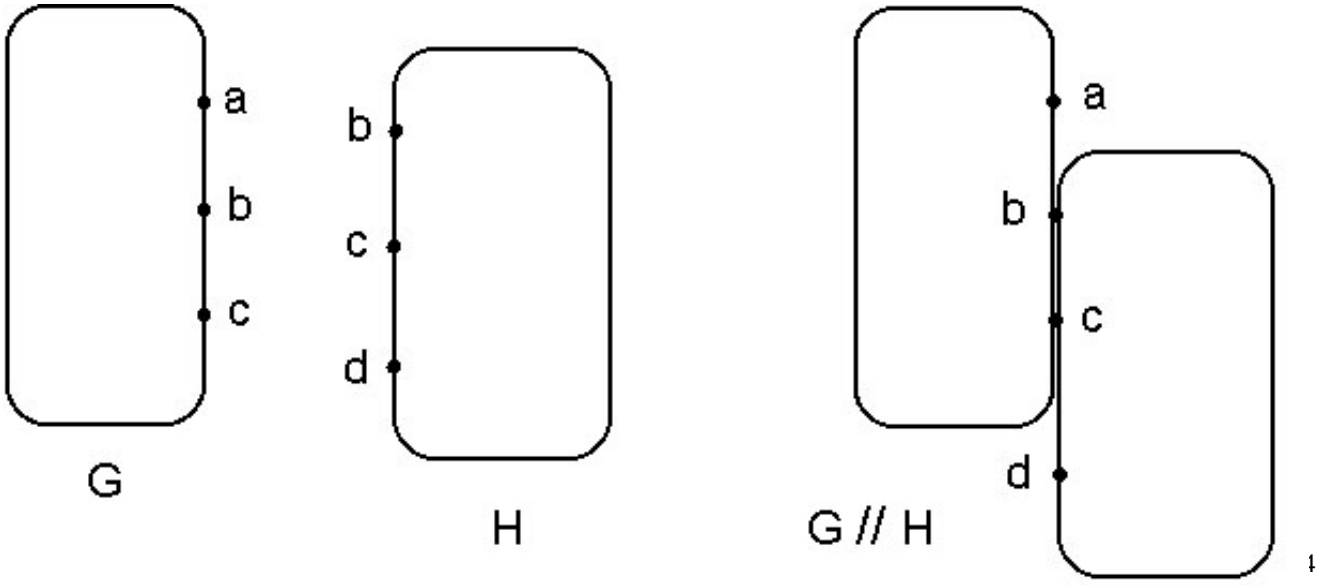
HR operations : Origin: **H**yperedge **R**eplacement *hypergraph grammars*
associated graph complexity measure : tree-width

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

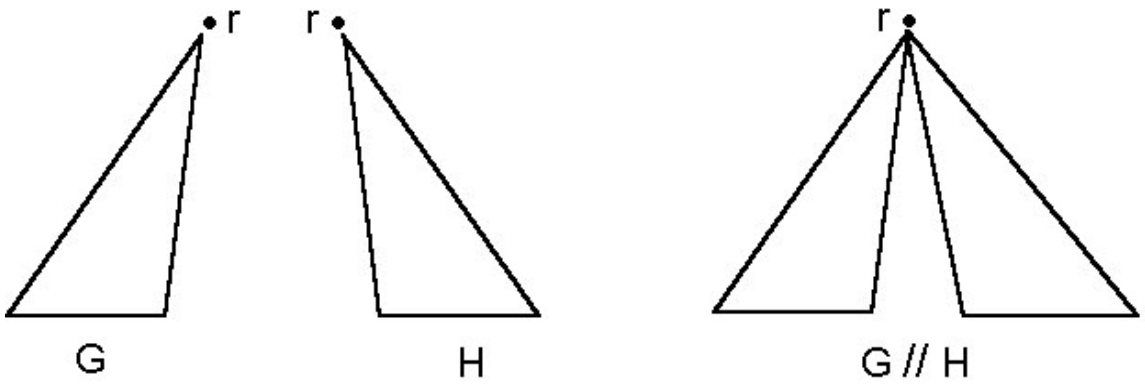
Nullary operations denote *basic graphs* : edge graphs, isolated vertices.

Terms over these operations *define* (or *denote*) graphs (with or without sources)

Example : Trees

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

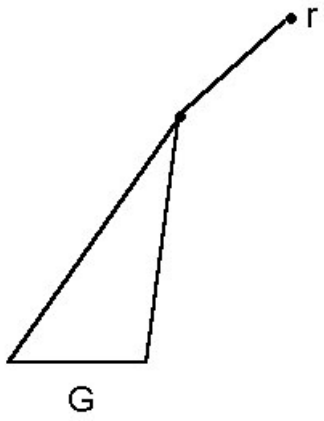
Fusion of two trees at their roots :



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

$e = r \bullet\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\bullet\ n$

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



Trees are defined by : $T = T\ //\ T \cup extension(T) \cup r$

Example : (Directed) series-parallel graphs

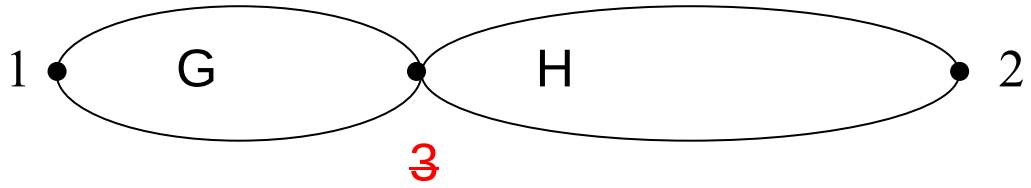
defined as directed graphs with sources 1 and 2,

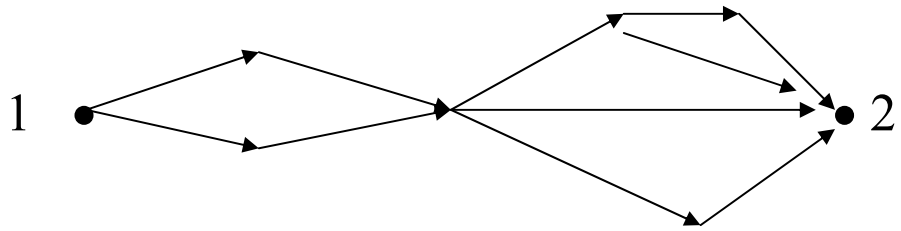
generated from e = 1 ⟶ 2 by the operations // (parallel-composition)

and the *series-composition* defined from the basic operations by :

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

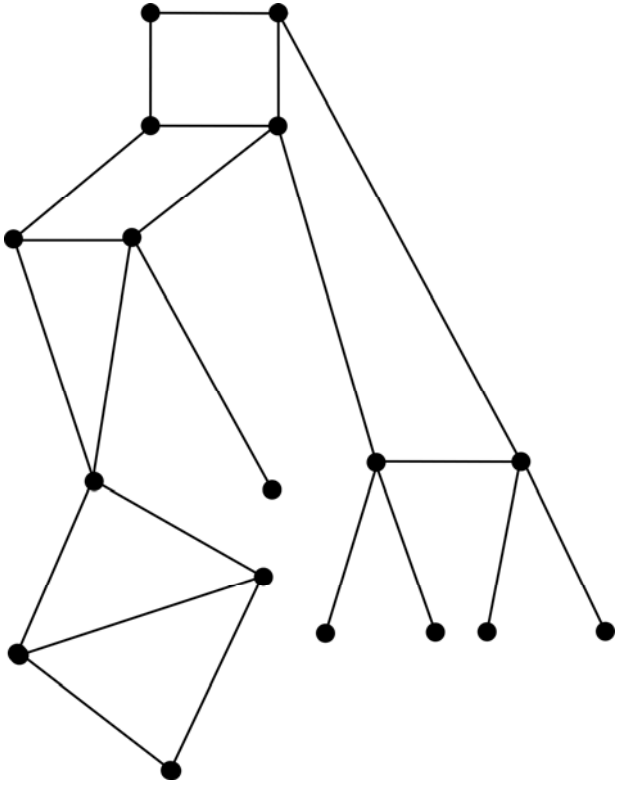
Example :



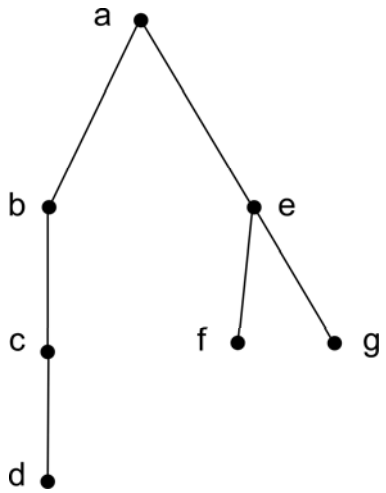


Their defining equation is : $S = S \,//\, S \cup S \bullet S \cup e$

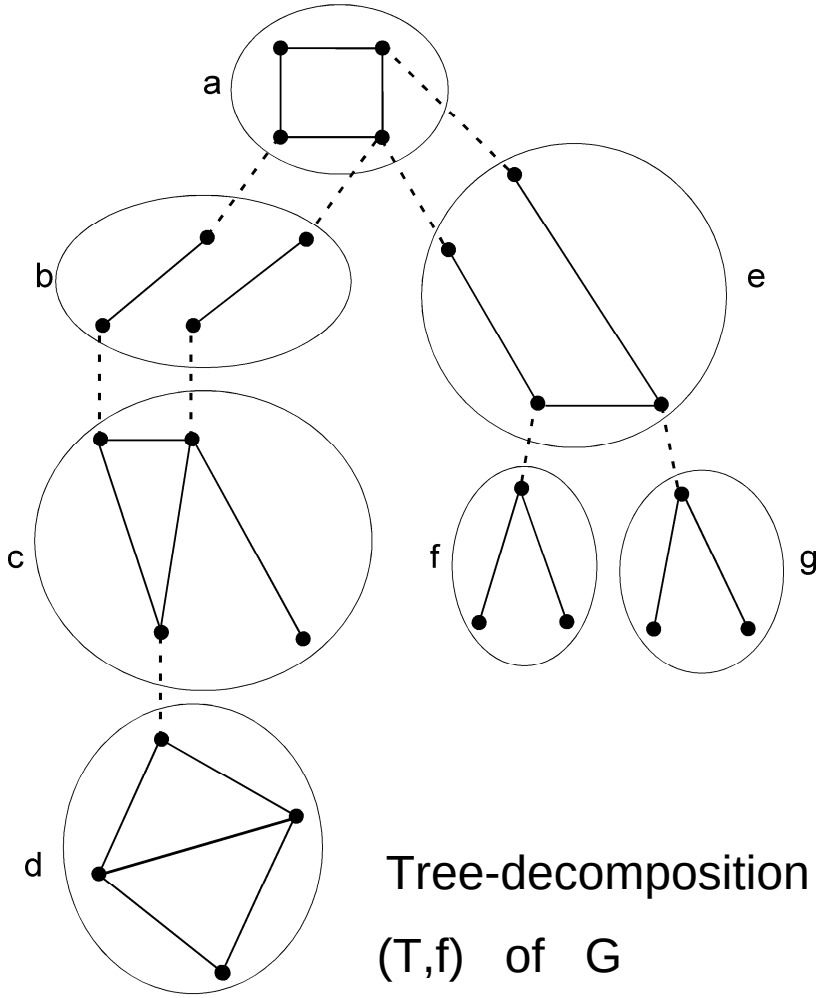
Relation to tree-decompositions and tree-width



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

if and only if it can be constructed from edges by using

the operations $//$, $Ren_{a \leftrightarrow b}$ and $Forget_a$ with $\leq$ k+1 labels a,b,….

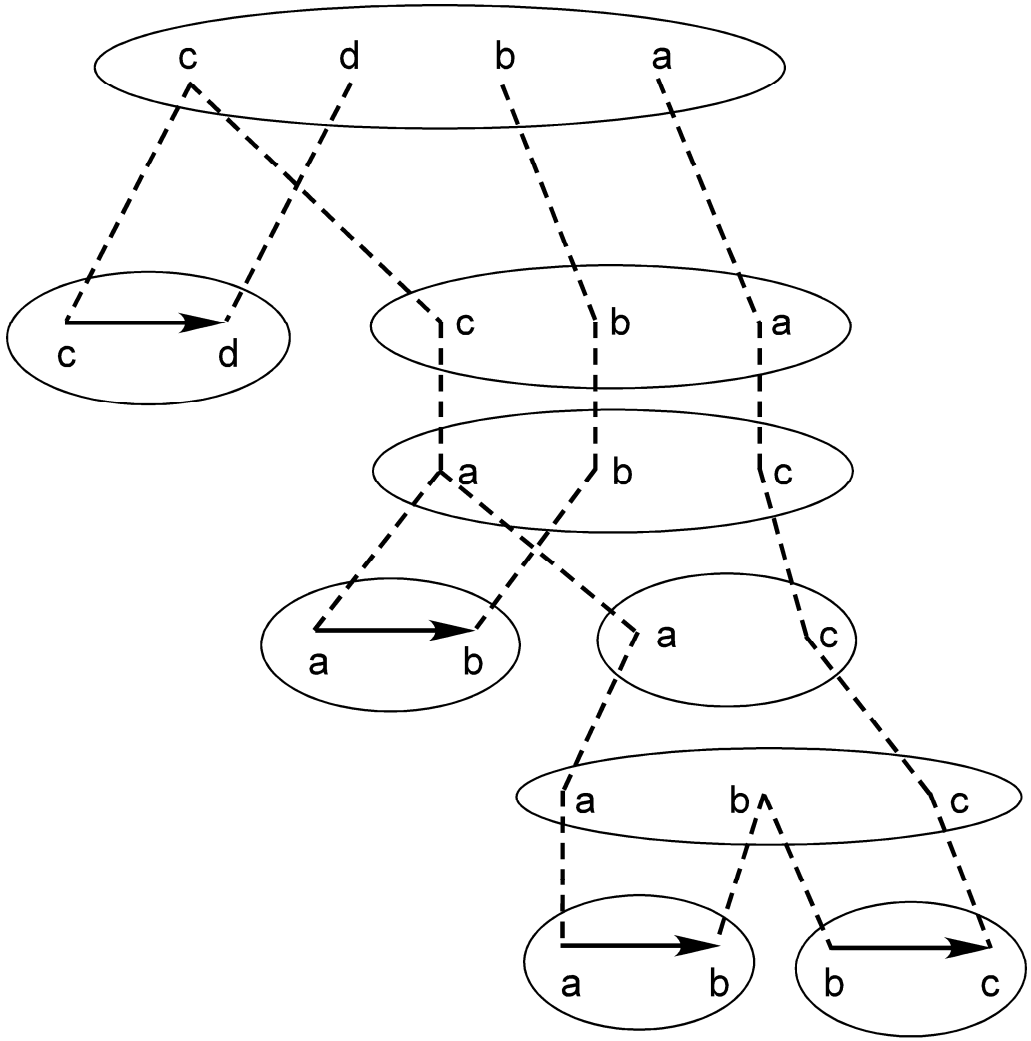
Consequences :

- Representation of tree-decompositions by terms.
- Algebraic characterization of tree-width.
- The set of graphs of tree-width at most k is equational for each k.
- Every **HR** equational set of graphs has bounded tree-width

(an upper bound is easy to obtain from a system S : just count the number of source labels used in S).

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.

Negative facts about **HR**-equational sets

- The set of all finite graphs is not **HR**-equational.

- Neither is the set of all square grids (planar graphs of degree 4)

- Parsing is NP-complete for certain fixed equation systems (graphs of cyclic bandwidth < 3)

But finding a tree-decomposition of width $\leq k$ (if it exists) can be done in "linear" time ($O(2^p.n)$ where n = number of vertices and $p = 32.k^2$)

Examples of **HR**-equational sets:

- Every context-free language but also the language $\{a^n b^n c^n \mid n > 0\}$.

- Outerplanar graphs (having a planar embedding with all vertices on the infinite (external) face) and Halin graphs (planar, made of a tree with a cycle linking all leaves).

The VR graph algebra

Origin : **V**ertex **R**eplacement *graph grammars*.

associated complexity measure: clique-width.

Graphs are *simple*, directed or not

(the definitions can be extended to graphs with multiple edges)

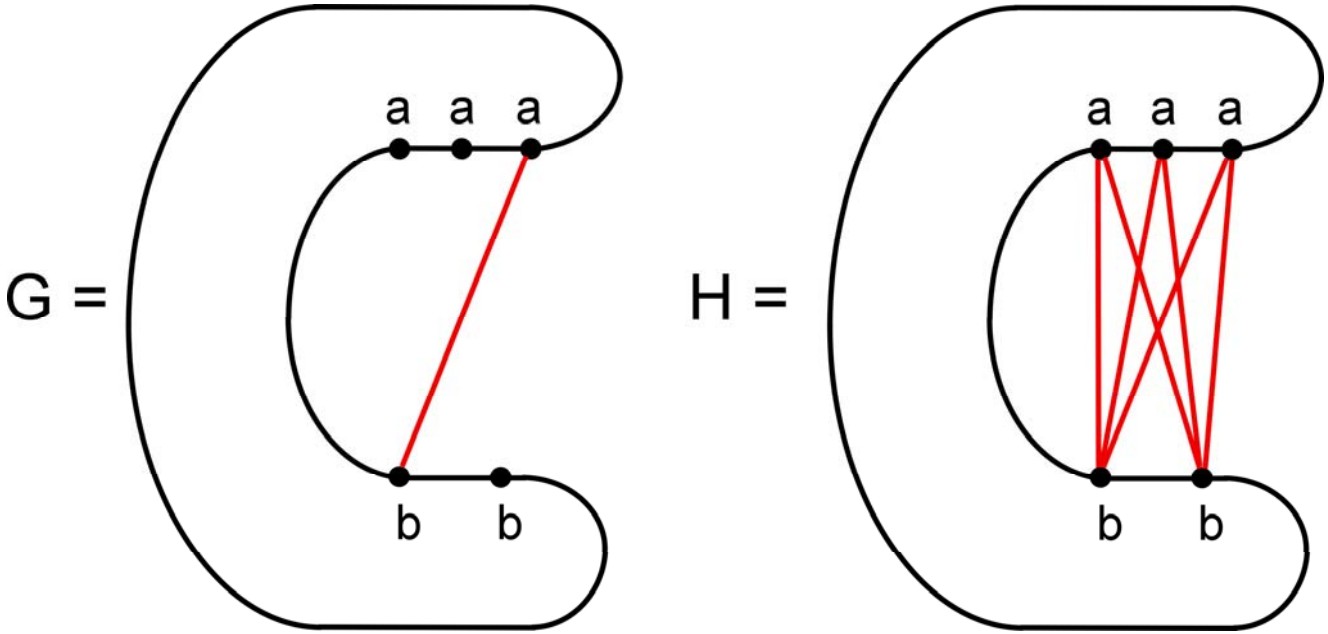
We use labels : *a , b , c, ..., d*.

Each vertex has one and only one label ; *several* vertices may have same label (whereas a source label designates a unique vertex)

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

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

$Add_{a,b}(G)$ is G augmented with edges *between* every *a*-port and every *b*-port (undirected case) or *from* every *a*-port to every *b*-port (directed case).



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

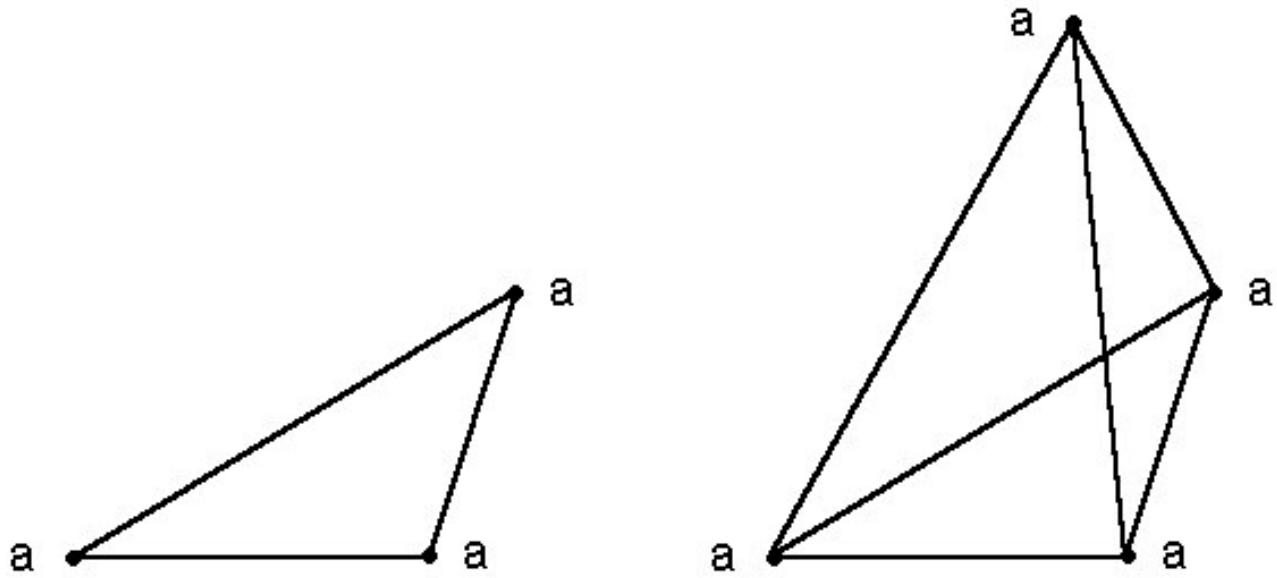
Definition: A graph G has clique-width $\leq k \Leftrightarrow$ it can be constructed from basic graphs with the operations $\oplus$, $Add_{a,b}$ and $Relab_{a \longrightarrow b}$ by using k labels.

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

Clique-width has no combinatorial characterization (like tree-width). It 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 (characterization by excluded *vertex-minors*, *exact cubic* decomposition algorithm).

Example 1 : Cliques have clique-width 2.



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 :

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

Example 2 : Cographs

They are generated by $\oplus$ and $\otimes$ (the *complete join*) defined by :

$G \otimes H = Relab_{b \to a}(Add_{a,b} (G \oplus Relab_{a \to b}(H)))$

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

Cographs are defined by :

$$C = C \oplus C \cup C \otimes C \cup \mathbf{a}$$

Fact : A simple undirected loop-free graph is a cograph if and only if it has clique-width at most 2.

Example 3 : *Distance hereditary graphs* have clique-width at most 3 (and are the graphs of rank-width 1).

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

(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:

Cographs (2), Distance hereditary graphs (3),

Graphs without $\{P_5 , \mathbf{1} \otimes P_4\}$ (5), or $\{\mathbf{1} \oplus P_4 , \mathbf{1} \otimes P_4\}$ (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. (P_n = path with n vertices)

Summary : Two algebras of (finite) graphs **HR** and **VR**

Two notions of "context-free sets" : the equational sets of algebras **HR** and **VR**, (and *below*, two notions of recognizable sets, based on congruences).

1) Comparison of the two classes :

Equat(**HR**) $\subseteq$ Equat(**VR**)

= sets in Equat(**VR**) whose graphs are without some fixed $K_{n,n}$ as subgraph.

2) Why not using a third algebra ?

One could, but Equat(**HR**) and Equat(**VR**) are robust in the following sense :

* logical characterizations independent of the initial definitions,
* stability under certain logically defined transductions,
* generation from trees.

For other algebras, we would loose these properties (proofs below).

3) Properties following from the algebraic setting :

- Closure under union, // , $\oplus$ and the unary operations
- Emptiness and finiteness are decidable (finite sets are computable)
- Semi-linearity Theorem (extends "Parikh's Theorem")
- Derivation trees
- Denotation of the generated graphs by terms,
- Upper bounds to tree-width and clique-width.

4) Properties that do not hold as we could wish :

- The set of all finite (even planar) graphs is neither HR- nor VR-equational.

- Parsing is NP-complete (even for some fixed equation systems)

Exercises

1) Prove that $\{a^n b^n c^n \mid n > 0\}$ and the set of *square words* (ww) are HR-equational.

2) Construct HR equation systems for the outerplanar and the Halin graphs.

3) Construct an HR equation system for the series-parallel graphs having an even number of vertices.

4) Construct a VR equation system for the trees having an number of nodes multiple of 3.

5) Construct a VR equation system for the cographs having an even number of edges.

6) Prove that the non-context-free language $\{a^n \mid n=2^p$ for some $p \geq 0\}$ is HR-equational for some appropriate algebra extending the monoid of words.

7) Complete the proof of the proposition page 19 : transform a tree-decomposition of width k into a term of the HR algebra defining the same graph and using $k+1$ source labels.

8) Prove that the proposition of page 19 holds without the source renaming operations.

3. Recognizable sets : an algebraic definition

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

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

Congruence = equivalence relation such that :

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

Finite means that $M / \approx$ is finite, i.e., $\approx$ has finitely many classes.

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 the algebra **M** (not only to the underlying set M).

Classical examples

Algebra	Recognizable sets
$\langle A^*, \cdot, \varepsilon, a,b,\ldots,d \rangle$	Regular languages (syntactic monoid)
$\langle A^*, \varepsilon, (\lambda u \in A^*.ua)_{a \in A} \rangle$	Regular languages (Myhill-Nerode)
$\mathbf{T}(F)$, terms over F, (initial F-algebra)	Regular sets of terms

On terms, h (cf. page 31) is the run of a *finite deterministic bottom-up automaton*

$\langle \mathbf{N}^k, +, (0,\ldots,0), \ldots (0,\ldots,1,0,\ldots,0) \ldots \rangle$ Finite unions of Cartesian products of k sets $\{ u + n.v \mid n \in \mathbf{N} \}$ for $u,v \in \mathbf{N}$

The algebras **HR** and **VR** have *infinite* signatures

We introduce two notions of type (or *sorts* in a many-sorted framework).

For **HR** : G has type $\tau(G)$ = the set of labels of its sources.

τ has a homomorphic behaviour :

$\tau(G // H) = \tau(G) \cup \tau(G)$; $\tau(Forget_a(G)) = \tau(G) - \{a\}$;

$\tau(Ren_{a \leftrightarrow b}(G)) = \tau(G)[a/b, b/a]$.

For **VR** : the type is $\pi(G)$ = the set of vertex labels having an occurrence.

π has a homomorphic behaviour :

$\tau(G \oplus H) = \tau(G) \cup \tau(H)$; $\tau(Add_{a,b}(G)) = \tau(G)$;

$\tau(Relab_{a \to b}(G)) = \tau(G)[b/a]$.

For defining recognizability of set L, we require that the congruence $\approx$ is *type preserving* (for τ or π according to the case, HR or VR) :

$$G \approx H \implies \tau(G) = \tau(H)$$

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

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

We can also use many-sorted algebras **HR** and **VR** with countably many sorts, and $\tau(G)$ and $\pi(G)$ as respective sorts of a graph G,

(because the type function has a homomorphic behaviour).

Two notions of recognizable sets of graphs, for algebras **HR** and **VR**.

Comparison of the two classes :

Rec(**VR**) $\subseteq$ Rec(**HR**)

= sets in Rec(**HR**) whose graphs are without some fixed $K_{n,n}$ as subgraph. (B.C. & P. Weil).

Recall :

Equat(**HR**) $\subseteq$ Equat(**VR**)

= sets in Equat(**VR**) whose graphs are without some fixed $K_{n,n}$ as subgraph.

Intuition : **VR** has more powerful operations than **HR**, but they make difference only for graphs without some $K_{n,n}$ as subgraph.

Properties of recognizable sets that follow from the algebraic setting :

- Closure under $\cup$, $\cap$ and $-$ (difference),
- under inverse homomorphisms and inverse unary derived operations.

(*Proofs* : clear from the definitions).

- *Filtering Theorem* : The intersection of an equational set and a recognizable one is equational

with *effective constructions*.

(*Proof* : cf. 2-colorability of series-parallel graphs detailed below).

Properties of recognizable sets of graphs that do not follow "algebraically"

Closure under the binary operations of the algebras : $//$, $\oplus$,

under the unary operations fg_a, $ren_{a \leftrightarrow b}$, $relab_{a \to b}$

Remarks: (1) This closure is false for $Add_{a,b}$ but is true if some "harmless" restriction of the use of this operation is made.

(2) Compare with regular languages:

it is *more difficult* to prove their closure under concatenation than under the Boolean operations ;

this is reflected by the sizes of syntactic monoids.

Properties that do not hold as we could wish or expect:

- Emptiness is not decidable (because of infinite signatures).

- Rec and Equat are incomparable (for **HR** and **VR**).

- Every set of square grids is **HR**- and **VR**-recognizable.

- There are uncountably many recognizable sets and *no characterization by finite automata or logical formulas*.

(To be contrasted with the cases of words and terms).

Inductive proofs and computations

Based on equations like the one that defines *Series-Parallel graphs* :

$$S = S \,//\, S \;\cup\; S \bullet S \;\cup\; e$$

Examples : “Proof that all series-parallel graphs are connected”,

“Proof that all series-parallel graphs are planar”,

“Number of directed paths from *Entry* to *Exit* in a given series-parallel graph”.

Sometimes, auxiliary properties and / or functions are necessary.

Recognizability means “finitely many auxiliary properties suffice”

Inductive computation :

Test of 2-colorability for series-parallel graphs

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

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

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 the rules :

Diff(e) = True ; Same(e) = False

Same(G//H) $\Leftrightarrow$ Same(G) $\wedge$ Same(H)
Diff(G//H) $\Leftrightarrow$ Diff(G) $\wedge$ Diff(H)

Same(G•H) $\Leftrightarrow$ (Same(G) $\wedge$ Same (H)) $\vee$ (Diff(G) $\wedge$ Diff(H))
Diff(G•H) $\Leftrightarrow$ (Same(G) $\wedge$ Diff(H)) $\vee$ (Diff(G) $\wedge$ Same(H))

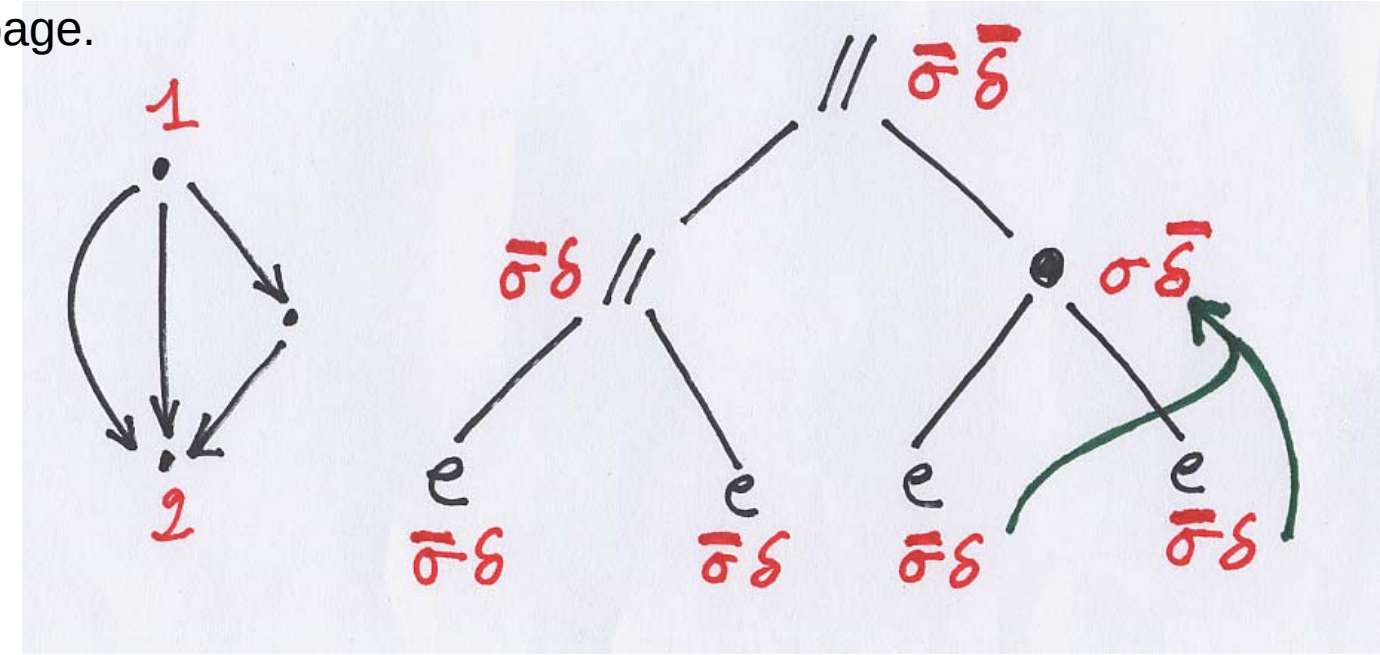
Application 1 : *Linear algorithm*

For every term t, we can compute, by running a finite deterministic bottom-up automaton on 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.

Example : σ at node u means that Same(Val(t /u)) is true, $\overline{\sigma}$ that it is false,

δ that Diff (Val(t /u)) is true, etc… Computation is *done bottom-up* with the rules of previous page.



The graph is not 2-colorable.

Application 2 : Equation system for 2-colorable series-parallel graphs

$S_{\sigma,\delta}$ = the set of series-parallel graphs that satisfy Same (σ) and Diff (δ)

$S_{\sigma,\overline{\delta}}$ = the set of those that satisfy Same and *not* Diff , etc …

From the equation : $S = S // S \cup S \bullet S \cup e$, we get the equation system :

$$(a)\quad S_{\sigma,\delta} = S_{\sigma,\delta}//S_{\sigma,\delta} \cup S_{\sigma,\delta}\bullet S_{\sigma,\delta} \cup S_{\sigma,\delta}\bullet S_{\sigma,\bar{\delta}} \cup S_{\sigma,\delta}\bullet S_{\bar{\sigma},\delta} \cup S_{\sigma,\bar{\delta}}\bullet S_{\sigma,\delta} \cup S_{\bar{\sigma},\delta}\bullet S_{\sigma,\delta}$$

$$(b)\quad S_{\bar{\sigma},\delta} = e \cup S_{\bar{\sigma},\delta}//S_{\bar{\sigma},\delta} \cup S_{\sigma,\delta}//S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\delta}//S_{\sigma,\delta} \cup S_{\sigma,\bar{\delta}}\bullet S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\delta}\bullet S_{\sigma,\bar{\delta}}$$

$$(c)\quad S_{\sigma,\bar{\delta}} = S_{\sigma,\delta}//S_{\sigma,\bar{\delta}} \cup S_{\sigma,\bar{\delta}}//S_{\sigma,\delta} \cup S_{\sigma,\bar{\delta}}//S_{\sigma,\bar{\delta}} \cup S_{\sigma,\bar{\delta}}\bullet S_{\sigma,\bar{\delta}} \cup S_{\bar{\sigma},\delta}\bullet S_{\bar{\sigma},\delta}$$

$$\begin{aligned}(d)\quad S_{\bar{\sigma},\bar{\delta}} = {} & S_{\bar{\sigma},\bar{\delta}}//S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\bar{\delta}}//S_{\sigma,\delta} \cup S_{\bar{\sigma},\bar{\delta}}//S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\bar{\delta}}//S_{\sigma,\bar{\delta}} \cup \\ & S_{\sigma,\delta}//S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\delta}//S_{\bar{\sigma},\bar{\delta}} \cup S_{\sigma,\bar{\delta}}//S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\delta}//S_{\sigma,\bar{\delta}} \cup \\ & S_{\sigma,\bar{\delta}}//S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\bar{\delta}}\bullet S_{\sigma,\delta} \cup S_{\bar{\sigma},\bar{\delta}}\bullet S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\delta}\bullet S_{\sigma,\bar{\delta}} \cup \\ & S_{\bar{\sigma},\bar{\delta}}\bullet S_{\bar{\sigma},\bar{\delta}} \cup S_{\sigma,\delta}\bullet S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\delta}\bullet S_{\bar{\sigma},\bar{\delta}} \cup S_{\sigma,\bar{\delta}}\bullet S_{\bar{\sigma},\bar{\delta}}\end{aligned}$$

In equation

$$(a)\quad S_{\sigma,\delta} = S_{\sigma,\delta}//S_{\sigma,\delta} \cup S_{\sigma,\delta} \bullet S_{\sigma,\delta} \cup S_{\sigma,\delta} \bullet S_{\sigma,\bar{\delta}} \cup S_{\sigma,\delta} \bullet S_{\bar{\sigma},\delta} \cup S_{\sigma,\bar{\delta}} \bullet S_{\sigma,\delta} \cup S_{\bar{\sigma},\delta} \bullet S_{\sigma,\delta}$$

$S_{\sigma,\delta}$ is in all terms of the righthand side : it defines (least solution) the empty set. This proves (a small theorem) :

Fact : No series-parallel graph satisfies Same and Diff.

We can simplify the system {(a), (b), (c), (d)} into :

$$(b')\quad S_{\bar{\sigma},\delta} = e \cup S_{\bar{\sigma},\delta}//S_{\bar{\sigma},\delta} \cup S_{\sigma,\bar{\delta}} \bullet S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\delta} \bullet S_{\sigma,\bar{\delta}}$$

$$(c')\quad S_{\sigma,\bar{\delta}} = S_{\sigma,\bar{\delta}}//S_{\sigma,\bar{\delta}} \cup S_{\sigma,\bar{\delta}} \bullet S_{\sigma,\bar{\delta}} \cup S_{\bar{\sigma},\delta} \bullet S_{\bar{\sigma},\delta}$$

$$(d')\quad S_{\bar{\sigma},\bar{\delta}} = S_{\bar{\sigma},\bar{\delta}}//S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\bar{\delta}}//S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\bar{\delta}}//S_{\sigma,\bar{\delta}} \cup S_{\bar{\sigma},\delta}//S_{\sigma,\bar{\delta}} \cup S_{\bar{\sigma},\bar{\delta}} \bullet S_{\bar{\sigma},\delta} \cup S_{\bar{\sigma},\bar{\delta}} \bullet S_{\sigma,\bar{\delta}} \cup S_{\bar{\sigma},\bar{\delta}} \bullet S_{\bar{\sigma},\bar{\delta}} \cup S_{\bar{\sigma},\delta} \bullet S_{\bar{\sigma},\bar{\delta}} \cup S_{\sigma,\bar{\delta}} \bullet S_{\bar{\sigma},\bar{\delta}}$$

By replacing $S_{\bar{\sigma},\bar{\delta}}$ by T_σ, $S_{\sigma,\delta}$ by T_δ, by using commutativity of //, we get the system (for the 2-colorable series-parallel graphs)

$$\begin{cases} T = T_\sigma \cup T_\delta \\ T_\sigma = T_\sigma//T_\sigma \cup T_\sigma \bullet T_\sigma \cup T_\delta \bullet T_\delta \\ T_\delta = e \cup T_\delta//T_\delta \cup T_\sigma \bullet T_\delta \cup T_\delta \bullet T_\sigma \end{cases}$$

Recognizability and inductive sets of properties

Definition : A set P of properties on an F-algebra **M** is F-inductive if, for every $p \in P$ and $f \in F$, there exists a Boolean formula *B* such that :

$$p(f_{\mathbf{M}}(a,b)) = B[\ldots,q(a),\ldots,q'(b),\ldots.] \quad \text{for all } a \text{ and } b \text{ in } \mathbf{M}$$

$$q, q' \in P, \; q(a),\ldots, q(b) \in \{\textit{True, False}\}.$$

Proposition : A subset L of **M** is recognizable if and only if it is the set of elements that satisfy a property belonging to a *finite* inductive set P of properties

Inductive sets formalize the notion of “auxiliary properties” in proofs by induction.

Inductive sets of properties and automata on terms

The simultaneous computation of m inductive properties can be implemented by a finite deterministic bottom-up automaton with 2^m states running on terms *t*.

This computation takes time $O(|t|)$: this fact is the key to fixed-parameter tractable algorithms.

Remark : Membership of an element **m** of **M** in a *recognizable set* L can be tested by such an automaton on *any* term *t* in T(F) defining **m**

(in some term if L is equational, i.e. “context-free”).

4. Monadic Second-Order (MS) Logic

A logical language that specifies inductive properties and functions

= First-order logic on power-set structures

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

MS (expressible) properties : transitive closure, properties of paths, connectivity, planarity (via Kuratowski, uses connectivity), k-colorability.

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

G is 3-colorable :

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

G (undirected) is not connected :

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

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 as in :

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

expressing that G[Y] is connected (note that Y is free in R).

Application : G contains (fixed) H as a minor
where $V_H = \{1,\ldots,p\}$: there exist disjoint sets of vertices $X_1,\ldots, X_p$ in G such that each $G[X_i]$ is connected and, whenever if i -- j in H, there is an edge between X_i and X_j.

Consequence : planarity is MS-expressible (no minor K_5 or $K_{3,3}$).

Provably non-expressible properties

- G is isomorphic to $K_{p,p}$ for some p (*not fixed*; needs equipotence of two sets, hence quantification over binary relations to find if there is a bijection).

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

- Card(X) is a multiple of p. (But this is possible if the graph is linearly ordered or some linear order is definable by an MS formula).

Definition : Adding these *cardinality set predicates* to MS logic gives Counting monadic second-order logic (or CMS): all good properties of MS logic hold for it.

(Adding an *equicardinality* set predicate to MS would spoil everything.)

Edge set quantifications *increase* the expressive power

Incidence graph of G undirected, **Inc**(G) = ($V_G \cup E_G$, $inc_G(.,.)$)

$inc_G(v,e)$ $\Leftrightarrow$ v is a vertex of edge e.

Monadic second-order formulas written with inc can use quantifications on sets of edges : they define MS_2 –expressible graph properties.

The existence of a perfect matching or a Hamiltonian circuit is MS_2 -expressible but not MS-expressible.

Definition : A set L of finite graphs is MS-definable (MS_2 –definable) if

$L = \{ G \text{ finite } / \; G \models \varphi \}$ ($L = \{ G \text{ finite } / \; \mathbf{Inc}(G) \models \varphi \}$) for a fixed MS sentence (a formula without free variables) φ.

Recognizability Theorem : (1) A language (set of words or *finite terms*) is recognizable (by congruence or automaton) $\Leftrightarrow$ it is MS definable

(2) A set of finite graphs is VR-recognizable $\Leftarrow$ it is CMS-definable

(3) A set of finite graphs is HR-recognizable $\Leftarrow$ it is CMS_2-definable

Proofs: (1) Doner, Thatcher & Wright, (1968 - 1970).

(2) and (3) can be proved together in two ways :

- by using the Feferman-Vaught paradigm
- by constructing an automaton on terms by induction on the structure of the given formula. This method (see Section 5 below) is better for concrete implementation.

The Feferman-Vaught paradigm

Main idea : the validity of an MS formula in the disjoint union of two relational structures can be deduced from those of finitely many auxilliary formulas of no larger quantifier-height in each of the two structures. (A very simple case of the "Composition Method" for infinite combinations of structures by Feferman & Vaught and Shelah.)

This is inductivity / recognizability.

We consider the easiest case, that of **VR**-recognizability

Notation : The result of the *query* defined by formula φ with free variables among $X_1,\ldots,X_n$,i.e., the set of satisfying assignments in G is

$$Sat(G, \varphi, X_1,\ldots,X_n) = \{ (V_1,\ldots,V_n) \ / \ G \models \varphi(V_1,\ldots,V_n) \}$$

Lemma 1 : If *f* is a *quantifier-free* mapping on graphs (edge-addition, vertex relabeling, edge complement) , every φ has a Backwards Translation $f^{\#}(\varphi)$ relative to *f* such that for all G :

$$Sat(f(G), \varphi, X_1,\ldots,X_n) = Sat(G, f^{\#}(\varphi), X_1,\ldots,X_n)$$

where $f^{\#}(\varphi)$ has no larger quantifier-height than φ.

Splitting Theorem : One can construct formulas ψ_i, θ_i , i = 1,…,p, of no larger quantifier-height than φ such that for all disjoint G and H : Sat(G $\oplus$ H, φ, X_1,…,X_n) is the disjoint union of the sets

$$\mathrm{Sat}(G, \psi_i, X_1,\ldots,X_n) \lozenge \mathrm{Sat}(H, \theta_i, X_1,\ldots,X_n), \quad i = 1,\ldots,p,$$

where $\lozenge$ combines "partial answers" as follows :

$$\underline{A} \lozenge \underline{B} = \{ (A_1 \cup B_1,\ldots,A_n \cup B_n) \ / \ (A_1,\ldots,A_n) \in \underline{A} , (B_1,\ldots,B_n) \in \underline{B} \}$$

Proof : Induction on the structure of φ.

Lemma 2 : For each n and h there are finitely many formulas $\varphi(X_1,\ldots,X_n)$ of quantifier-height $\leq$ h, up to a decidable and sound equivalence.

Proof : Routine. But yields large number !

Proof of the Recognizability Theorem :

For each h , the equivalence relation such that :

$G \approx H \Leftrightarrow Sat(G, \psi) = Sat(H, \psi)$ (= $\varnothing$ or ( ) , the empty sequence)

for every sentence ψ of quantifier-height $\leq$ h

is a *type-preserving*, *locally finite* congruence on **VR** that saturates the set of graphs defined by φ, for each φ of quantifier-height $\leq$ h

(The same proof works for **HR**, the algebra of graphs with sources, but one more lemma is necessary to handle the fusion of sources in parallel composition; sources are represented by constants of the logical structures).

Extensions of the proof

1) For *counting* valid assignments, i.e., for computing, for given G and fixed formula φ the cardinality of the set $Sat(G, \varphi, X_1,\ldots,X_n)$, the Splitting Theorem gives (*because of* *disjoint unions*) the recursion :

$$|Sat(G \oplus H, \varphi, X_1,\ldots,X_n)| =$$

$$\sum_{i = 1,\ldots,p} |Sat(G, \psi_i, X_1,\ldots,X_n)| \,.\, |Sat(H, \theta_i, X_1,\ldots,X_n)|$$

2) Similar fact for *optimizing functions*, defined by :

$$MaxSat(G, \varphi, X) = Max \{ |A| \;/\; G \;| = \varphi(A) \}$$

Algorithmic consequences of the Recognizability Theorem

MS formulas

MS_2 formulas

using edge quantifications

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

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

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

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

FPT *cubic* for clique-width

finding a **VR**-term defining the graph is possible in cubic time (Hlineny, Oum & Seymour)

FPT *linear* for tree-width

finding a tree-decomposition is possible in linear time (Bodlaender)

(even in *LogSpace*, Elberfeld *et al.,* FOCS 2010)

Language Theoretical consequences

One can filter out from HR- or VR-equational sets the graphs which do not satisfy given MS_2- or MS-properties and one obtains HR- or VR-equational sets.

Generalizes : the intersection of a context-free language and a regular language is context-free.

Consequences for the decidability of logical theories

The MS_2-theory of the set of graphs of tree-width $\leq$ k is decidable.

(is a given sentence true in all graphs of tree-width $\leq$ k ?)

The MS-theory of the set of graphs of clique-width $\leq$ k is decidable.

Exercises

1) Write an MS-sentence expressing that the considered *simple* graph is a tree.

2) Write an MS-formula with free variables x,y,z expressing that, in a tree (undirected and unrooted), if one takes x as root, then y is an ancestor of z.

3) Write an MS_2-sentence expressing that a graph has a Hamiltonian cycle.

4) A nonempty word over alphabet {a,b} can (also) be considered as a directed path given with unary relations lab_a and lab_b representing the sets of occurrences of letters a and b. Prove that every regular language over {a,b} is MS-definable.

5) A complete bipartite graph $K_{n,m}$ has a Hamiltonian cycle iff n=m. Construct such a graph "over" any word in $\{a,b\}^+$ having at least one a and at least one b. Deduce from 4) that Hamiltonicty is not MS-expressible.

6) Write an MS_2-sentence expressing that a graph has a spanning tree of degree ≤ 3. Show as in 5) that this property is not MS-expressible.