



Monadic second-order logic for graphs.
Algorithmic and language theoretical applications

Part 1

Bruno Courcelle

Université Bordeaux 1, LaBRI, and Institut Universitaire de France

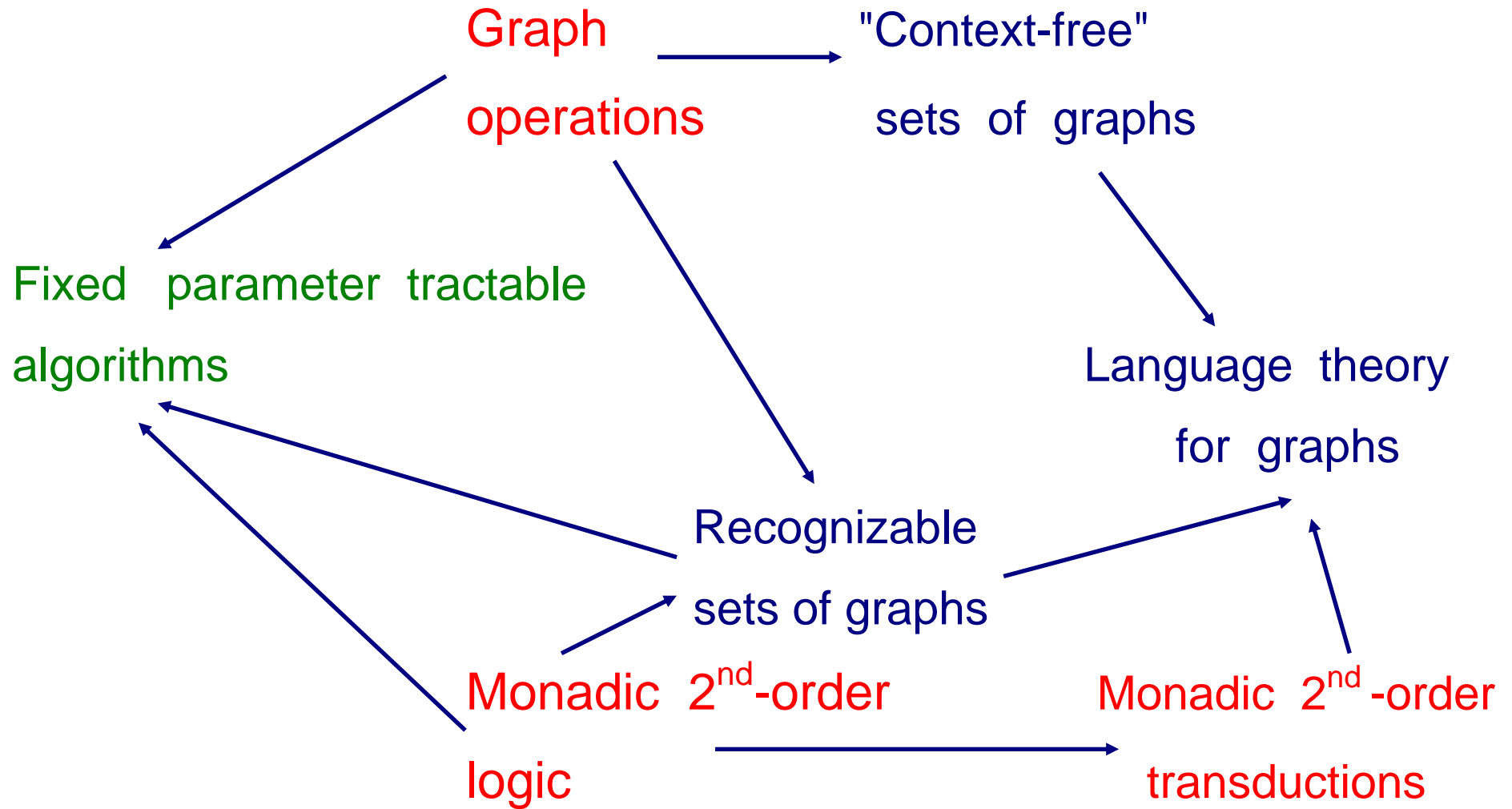


Reference : Graph structure and monadic second-order logic,
book to be published by Cambridge University Press, readable on :
<http://www.labri.fr/perso/courcell/ActSci.html>

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.

An overview chart



Key concepts of Language Theory and their extensions

| <i>Languages</i> | <i>Graphs</i> |
|--|---|
| Algebraic structure : monoid $(X^*, *, \varepsilon)$ | Algebras based on graph operations : $\oplus, \otimes, //$ quantifier-free definable operations Algebras : HR, VR |
| Context-free languages : Equational subsets of $(X^*, *, \varepsilon)$ | 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 (blue = Language Theor. issues)
2. Two graph algebras. Tree-width and clique-width. (red = new notions)
3. Recognizability : an algebraic notion.
4. Monadic second-order sentences define recognizable sets.
5. Fixed-parameter tractable algorithms (green = algorithms)
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* : Graph classes on which monadic second-order logic is decidable
9. *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

$$X \rightarrow aXY; \quad X \rightarrow b; \quad Y \rightarrow cYYX; \quad Y \rightarrow a$$

is equivalent to the system of two equations:

$$X = aXY \cup \{b\}$$

$$Y = cYYX \cup \{a\}$$

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

In arbitrary algebras (in graph algebras) we consider equation systems like:

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

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

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**

$\mathbf{M} = \langle M, (f_{\mathbf{M}})_{f \in F} \rangle$: 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 system of the general form :

$S = \langle X_1 = p_1, \dots, X_n = p_n \rangle$

X_1, \dots, X_n are unknowns (ranging over sets)

p_1, \dots, 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 :

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

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

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

$$\mu \mathbf{X}. S_{P(\mathbf{M})}(\mathbf{X}) = \bigcup_{i \geq 0} S_{P(\mathbf{M})}^i(\emptyset, \dots, \emptyset) \quad (\text{increasing sequence})$$

An **equational set of \mathbf{M}** is a component of $\mu \mathbf{X}. S_{P(\mathbf{M})}(\mathbf{X})$ for some equation system S . **Equat(\mathbf{M})** = the equational sets of \mathbf{M}

Classical examples

Algebra

$\langle A^*, \cdot, \varepsilon, a, b, \dots, d \rangle$

$\langle A^*, \varepsilon, (\lambda u \in A^*. ua)_{a \in A} \rangle$

$T(F)$, terms over F , (initial F -algebra)

$\langle \mathbf{N}^k, +, (0, \dots, 0), \dots (0, \dots, 1, 0, \dots, 0) \dots \rangle$

Equational sets

Context-free languages

Regular languages

Regular sets of terms

Semi-linear sets =

finite unions of sets $\{ \mathbf{u} + n_1 \cdot \mathbf{v}_1 + \dots + n_p \cdot \mathbf{v}_p \mid n_1, \dots, n_p \in \mathbf{N} \}$

for $\mathbf{u}, \mathbf{v}_1, \dots, \mathbf{v}_p \in \mathbf{N}^k$

Properties of context-free languages valid at the algebraic level

- 1) If K and L are equational sets of \mathbf{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-point by homomorphisms :

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

Hence $\mu \mathbf{X}.S_{P(\mathbf{M})}(\mathbf{X}) = \text{val}_{\mathbf{M}}(\mu \mathbf{X}.S_{P(\mathbf{T}(F))}(\mathbf{X}))$ ($\text{val}_{\mathbf{M}}$ = 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 \mathbf{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 stop as soon as $S_{P(\mathbf{M})}(\mathbf{X})^i(\emptyset, \dots) = S_{P(\mathbf{M})}(\mathbf{X})^{i+1}(\emptyset, \dots)$).

4) Finiteness test (with some natural “size” conditions).

5) For every context-free language L over k letters : a, \dots, d , the set of k -tuples $(|u|_a, \dots, |u|_d)$ 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.

Application : The set of terms with all constants at same depth is *not* equational.

2. The graph algebras **HR** and **VR**

We define two graph algebras \rightarrow

Equational sets of graphs, two generalizations of context-free languages.

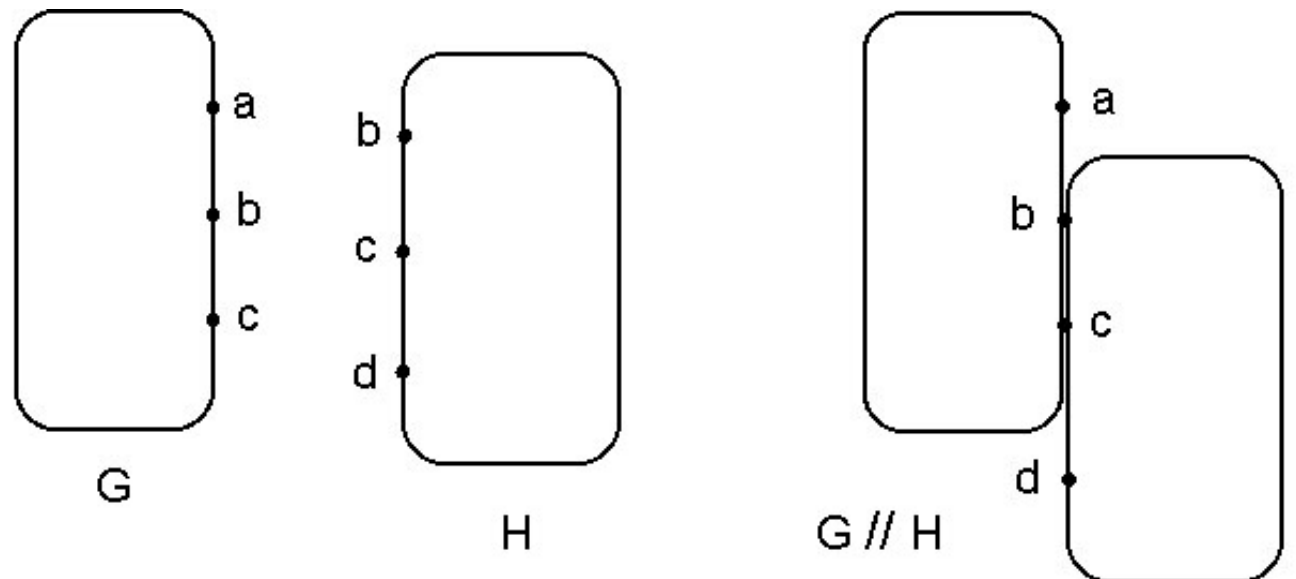
HR operations : Origin: **H**yperedge **R**eplacement hypergraph grammars
Associated complexity measure : **tree-width**

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

Binary operation(s) : **Parallel composition**

$G // H$ is the disjoint union of G and H and sources with same label are **fused**.

(If G and H are not disjoint, one first makes a copy of H disjoint from G).



Unary operations :

Forget some source label

$\text{Forget}_a(G)$ is G without a -source: the source is no longer distinguished ;
(it is made "internal").

Source renaming :

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

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

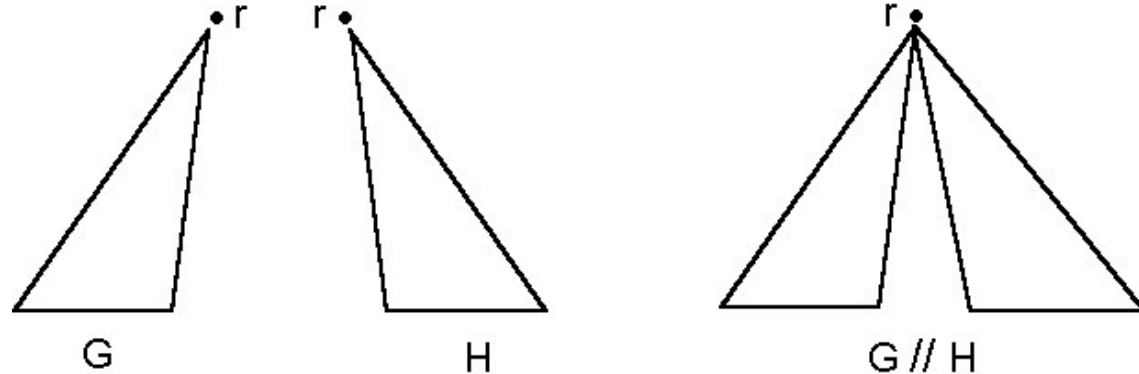
Terms over these operations *define* (or *denote*) graphs (with or without sources)

Example : Trees

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

Fusion of two trees

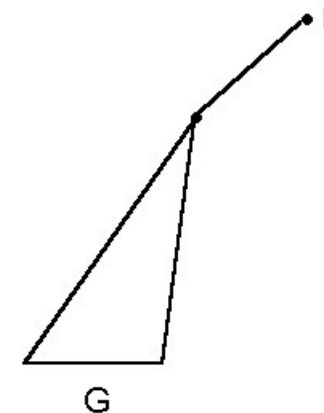
at their roots :



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

$$e = r \bullet \text{---} \bullet n$$

$$\text{Ren}_n \longleftrightarrow r (\text{Forget}_r(G // e))$$



Trees are defined by equation : $T = T // T \cup \text{extension}(T) \cup r$

Example : Series-parallel graphs

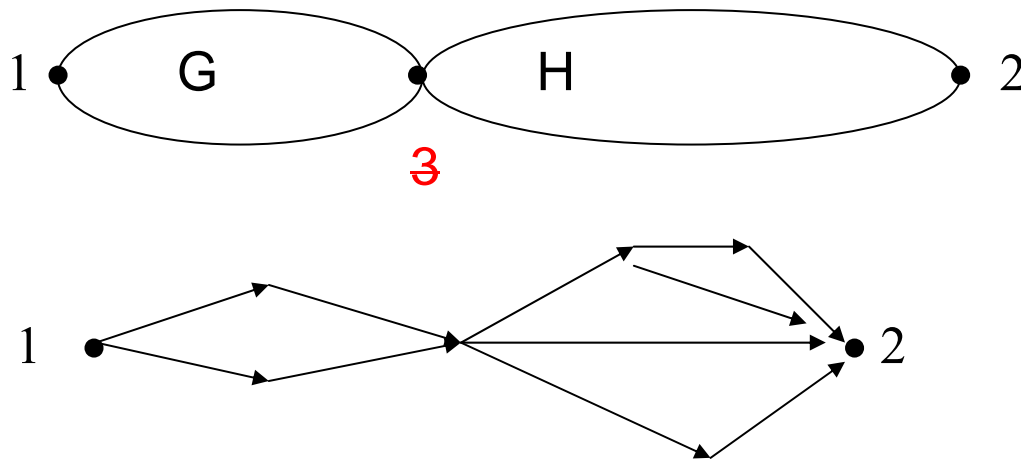
defined as directed graphs with sources 1 and 2,

generated from $e = 1 \longrightarrow 2$ by the operations $//$ (parallel-composition)

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

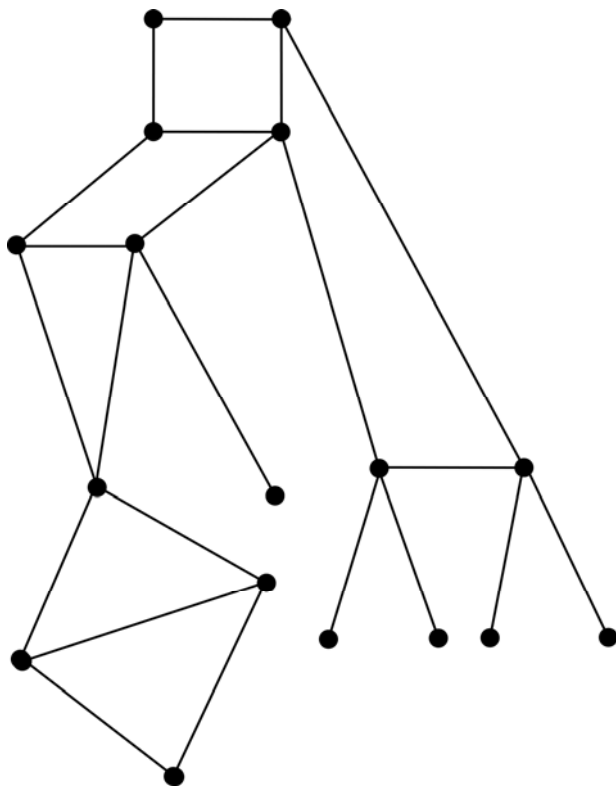
$$G \bullet H = \text{Forget}_3(\text{Ren}_{2 \leftrightarrow 3}(G) // \text{Ren}_{1 \leftrightarrow 3}(H))$$

Example :

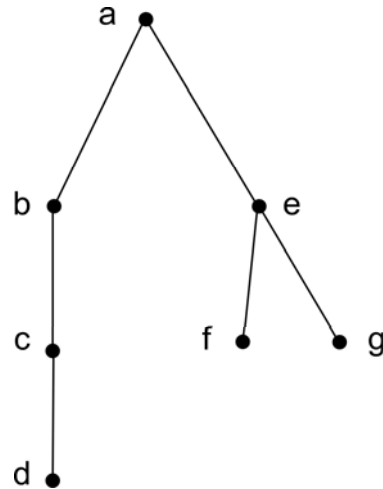


Their defining equation is : $S = S // S \cup S \bullet S \cup e$

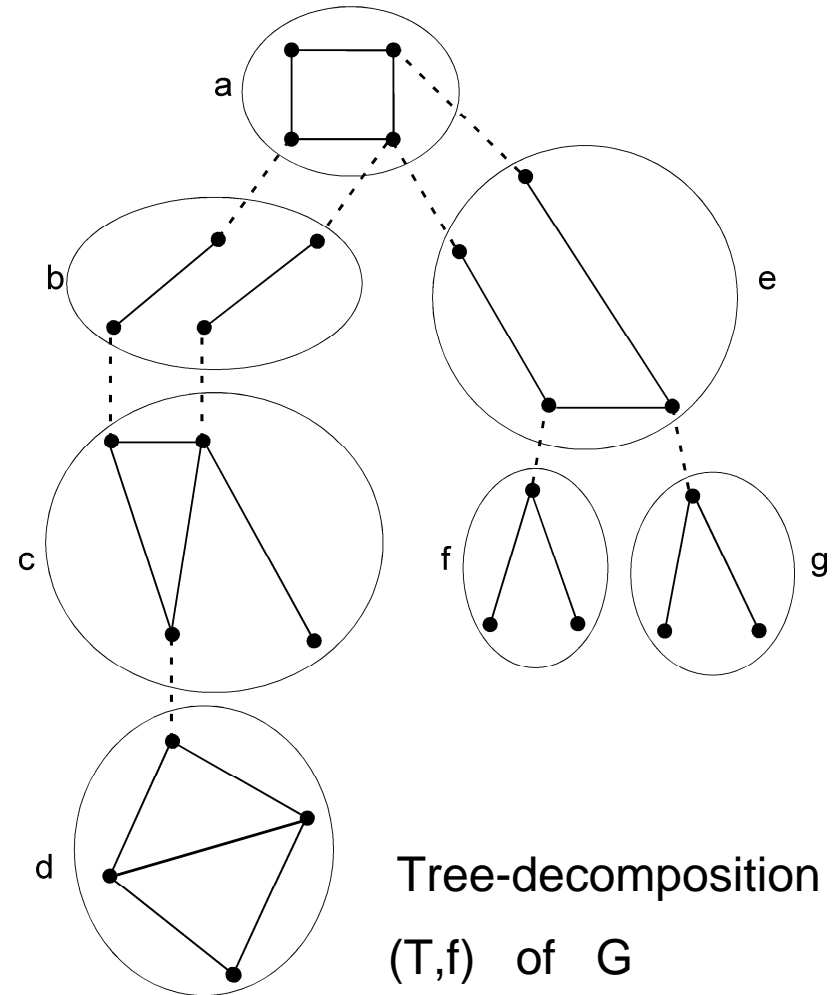
Relation to tree-decompositions and tree-width



Graph G



Tree T



Tree-decomposition
(T,f) of G

Dotted lines - - - link copies of the 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, \dots

Consequences :

Representation of tree-decompositions by terms

Algebraic characterization of tree-width.

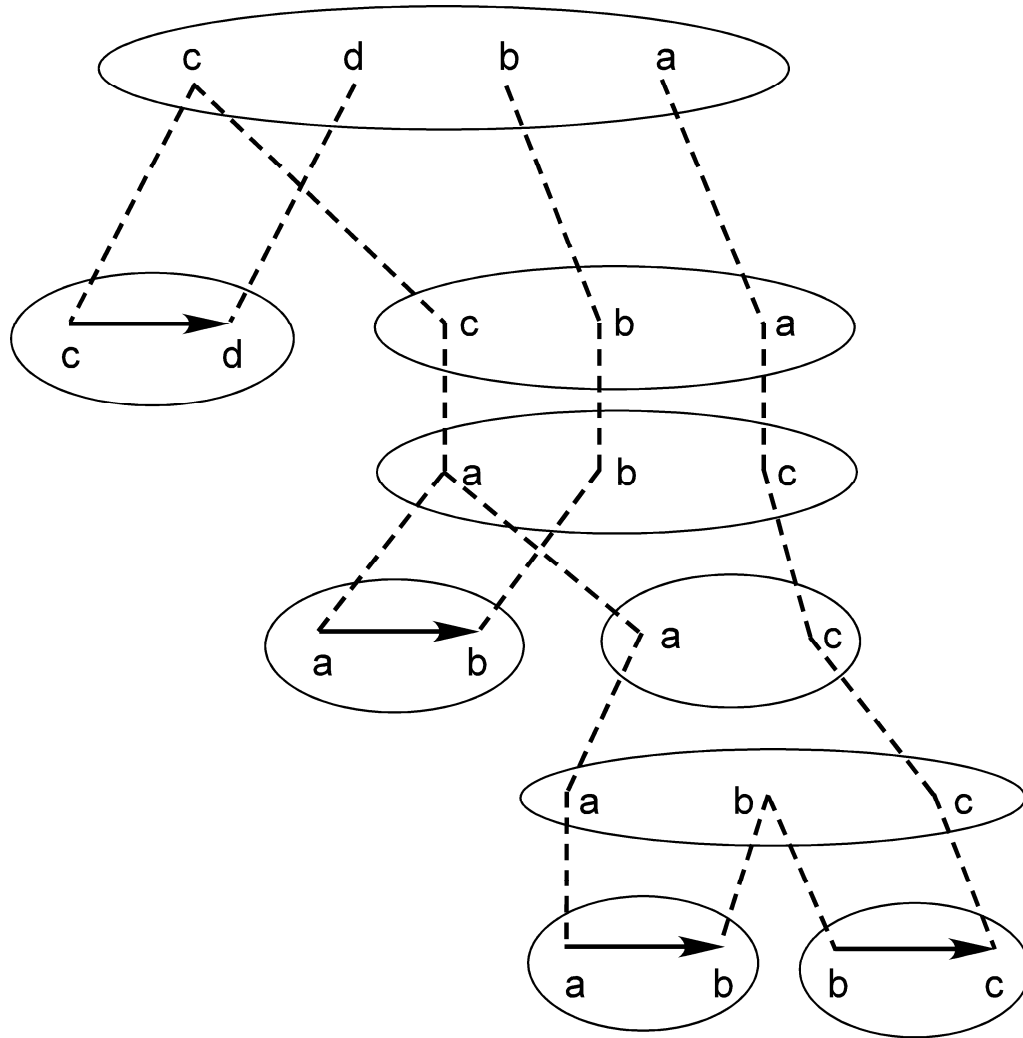
The set of graphs of tree-width at most k is equational for each k .

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 : $cd // \text{Ren}_{a \leftrightarrow c} (ab // \text{Forget}_b (ab // bc))$ (Constant ab denotes an edge from a to b)



The tree-decomposition associated with this term.

Negative facts : what does not hold as we could wish

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

Not even is the set of all square grids (planar graphs of degree 4)

Parsing is NP-complete for certain *fixed* equation systems
(graphs of circular bandwidth ≤ 2)

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

Examples of **HR**-equational equational sets.

Every context-free language. The non-context-free language $\{a^n b^n c^n \mid n > 0\}$.
(A word is a directed path with edges labelled by letters.)

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.

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

Each vertex has one and only one label ; *several* vertices may

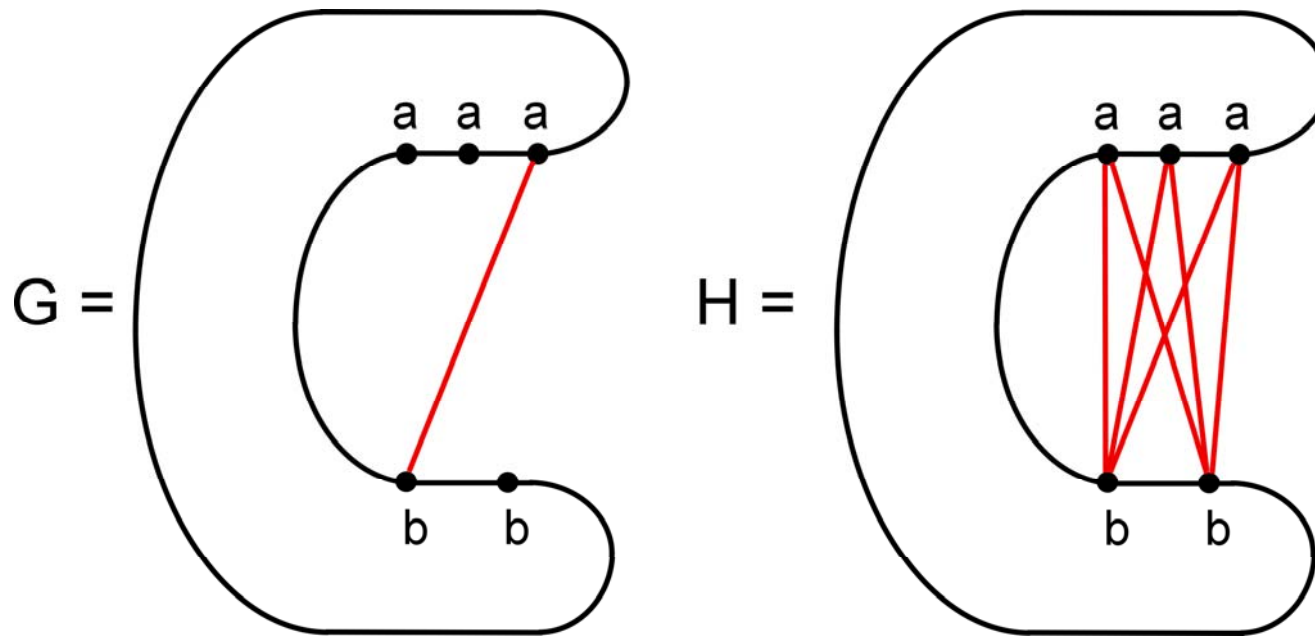
have same label

(a source name designates a unique vertex)

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 edges *between* every a -port and every b -port (undirected case) or *from* every a -port to every b -port (directed case).



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

The number of added edges depends on the argument graph.

Vertex relabellings :

$Relab_a \rightarrow b(G)$ is G with every vertex labelled by a relabelled into b

Basic graphs are those with a single vertex a

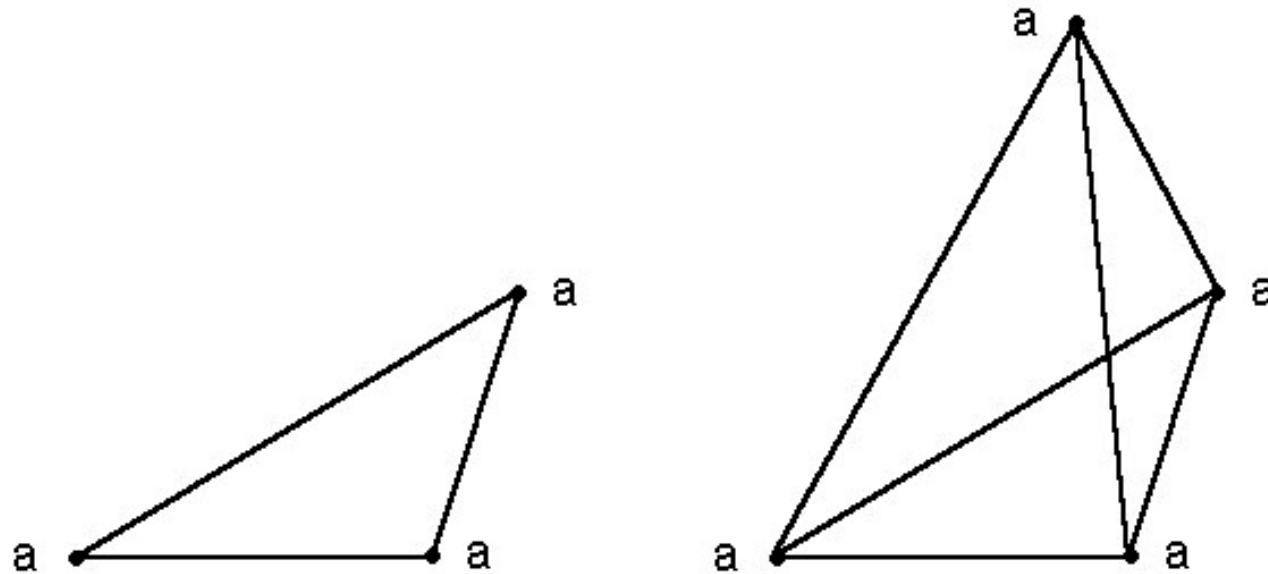
Definition: A graph G has **clique-width** $\leq k \Leftrightarrow$ it can be constructed from basic graphs with the operations \oplus , $Add-edg_{a,b}$ and $Relab_a \rightarrow b$ by using k labels.

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

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

Example 1 : Cliques have clique-width 2.



K_n is defined by t_n where $t_{n+1} = \text{Relab } b \rightarrow a(\text{Add-edg}_{a,b}(t_n \oplus b))$

Cliques are defined by the equation :

$$K = \text{Relab } b \rightarrow a(\text{Add-edg}_{a,b}(K \oplus b)) \cup a$$

Example 2 : Cographs

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

$$\begin{aligned} G \otimes H &= \text{Relab}_{b \rightarrow a}(\text{Add-edges}_{a,b}(G \oplus \text{Relab}_{a \rightarrow b}(H))) \\ &= G \oplus H \text{ with "all possible" undirected edges between } G \text{ and } H. \end{aligned}$$

Hence by the equation :

$$C = C \oplus C \cup C \otimes C \cup a$$

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 those of rank-width 1).

Proposition : (1) Bounded tree-width implies bounded clique-width, 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 two notions of recognizable sets, based on congruences).

1) Comparison of the two classes :

$$\mathbf{Equat}(\mathbf{HR}) \subseteq \mathbf{Equat}(\mathbf{VR})$$

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

2) Why not using a third algebra ? One could, but $\mathbf{Equat}(\mathbf{HR})$ and $\mathbf{Equat}(\mathbf{VR})$ 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 lose these properties (proofs below).

3) Properties of equational sets of graphs 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 sometimes NP-complete.

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 outerplanar and Halin graphs.
- 3) Construct an HR equation system for series-parallel graphs having an even number of vertices.
- 4) Construct a VR equation system for trees having a number of nodes multiple of 3.
- 5) Construct a VR equation system for cographs having an even number of edges.
- 6) Prove that the non-context-free language $\{a^n \mid n=2^p \text{ for some } p \geq 0\}$ is HR-equational for some appropriate algebra extending the monoid of words.
- 7) Complete the proof of the algebraic characterization of tree-width : transform a tree-decomposition into a term of the HR algebra defining the same graph.

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 :

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

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

Equivalently, $L = h^{-1}(D)$ for a homomorphism $h : \mathbf{M} \rightarrow \mathbf{A}$, where

\mathbf{A} is a *finite* F-algebra and $D \subseteq A$.

Rec(M) = the recognizable subsets of M . This notion is relative to the algebra \mathbf{M} .

Classical examples

Algebra

$\langle A^*, \cdot, \varepsilon, a, b, \dots, d \rangle$

$\langle A^*, \varepsilon, (\lambda u \in A^*. ua)_{a \in A} \rangle$

$\mathbf{T}(F)$, terms over F , (initial F -algebra)

On terms, h is the run of a *finite deterministic bottom-up automaton*

$\langle \mathbf{N}^k, +, (0, \dots, 0), \dots (0, \dots, 1, 0, \dots, 0) \dots \rangle$

Finite unions of Cartesian products of k sets $\{ \mathbf{u} + n \cdot \mathbf{v} \mid n \in \mathbf{N} \}$ for $\mathbf{u}, \mathbf{v} \in \mathbf{N}^k$

Recognizable sets

Regular languages
(syntactic monoid)

Regular languages
(Myhill-Nerode)

Regular sets of terms

Finite unions of Cartesian

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 :

$$\tau(G//H) = \tau(G) \cup \tau(H) \quad ; \quad \tau(\text{Forget}_a(G)) = \tau(G) - \{a\} \quad ;$$

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

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

π has a homomorphic behaviour :

$$\pi(G \oplus H) = \tau(G) \cup \tau(H) \quad ; \quad \pi(\text{Add-edg}_{a,b}(G)) = \tau(G) \quad ;$$

$$\pi(\text{Relab}_a \longrightarrow b(G)) = \pi(G)[b/a].$$

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**):

$$G \approx H \text{ implies } \tau(G) = \tau(H)$$

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 a recognizable set of graphs, for algebras **HR** and **VR**.

Comparison of the two classes :

$$\mathbf{Rec}(\mathbf{VR}) \subseteq \mathbf{Rec}(\mathbf{HR})$$

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

Recall :

$$\mathbf{Equat}(\mathbf{HR}) \subseteq \mathbf{Equat}(\mathbf{VR})$$

= 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)

(from $h: \mathbf{M} \rightarrow \mathbf{A}$ and $k: \mathbf{M} \rightarrow \mathbf{B}$, make $\text{hom.} : \mathbf{M} \rightarrow \mathbf{A} \times \mathbf{B}$)

Closure under inverse homomorphisms and inverse unary derived operations.

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

(generalizes the intersection of a context-free and a regular language)

With *effective constructions*.

Properties of recognizable sets of graphs that **do not follow** “algebraically”

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

(This closure is even **false** for **add-edg** but is true if some “harmless” restriction of the use of this operation is made.)

(It is **more difficult** to prove the closure under concatenation of regular languages than their closure under Boolean operations ; this is reflected by the sizes of syntactic monoids $\leq n.p.2^{n.p}$ vs. $\leq n.p$).

Properties 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.

Inductive proofs and computations

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

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

“Proof that all series-parallel graphs are connected”

(Connectedness is preserved by $//$ and \bullet , holds for e)

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

Sometimes, **auxiliary properties** or **functions** are necessary.

“Proof that all series-parallel graphs are planar”

($//$ does not preserve planarity ; a stronger property is preserved by $//$ and \bullet , and holds for e)

Recognizability means that “**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 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))

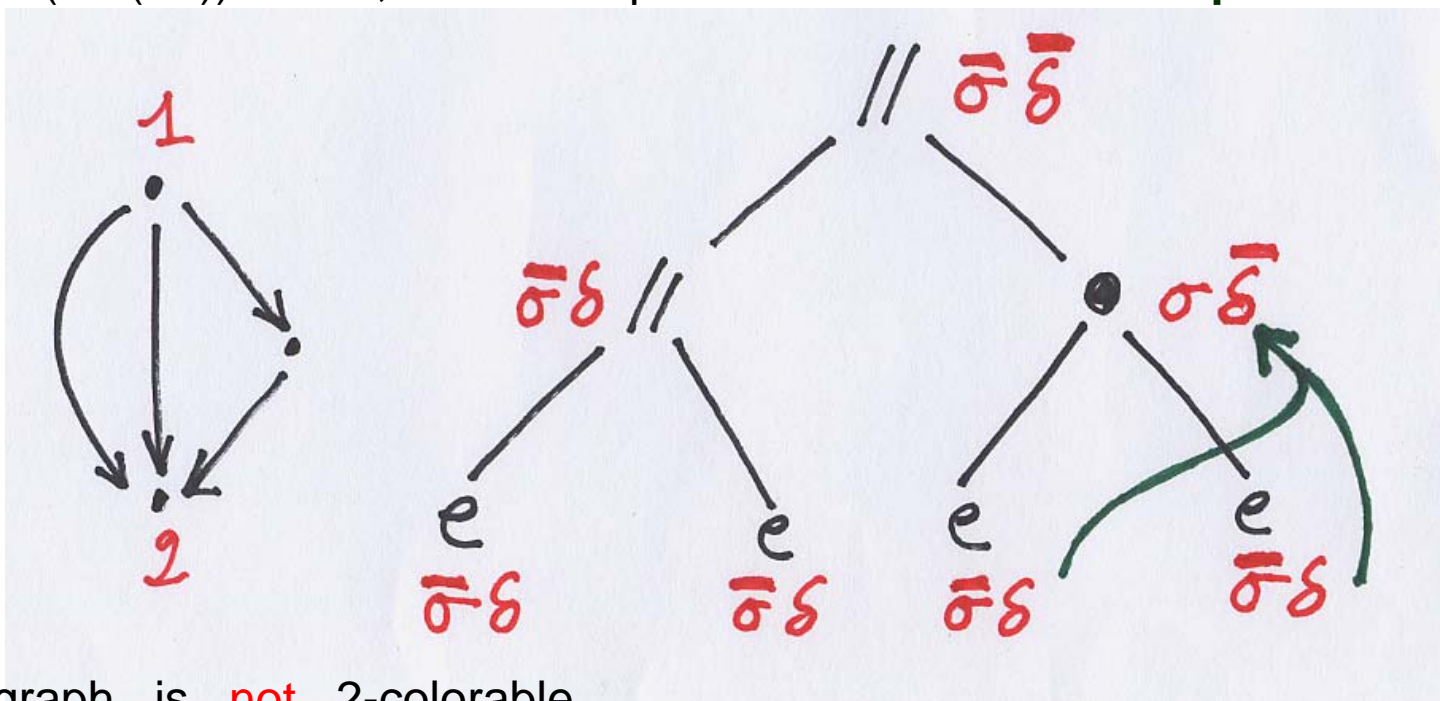
Diff(G•H) \Leftrightarrow (**Same**(G) \wedge **Diff**(H)) \vee (**Diff**(G) \wedge **Same**(H))

Application 1 : Linear algorithm

For every SP-term t , we can compute, by running a finite **deterministic bottom-automaton** on t , the pair of Boolean values ($\text{Same}(\text{Val}(t))$, $\text{Diff}(\text{Val}(t))$).

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

Example : σ at node u means that $\text{Same}(\text{Val}(t/u))$ is true, $\bar{\sigma}$ that it is false, δ that $\text{Diff}(\text{Val}(t/u))$ is true, etc... Computation is **done bottom-up** with the rules :



The graph is **not** 2-colorable.

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

We let $S_{\sigma,\delta}$ be the set of series-parallel graphs that satisfy **Same** (σ) and **Diff** (δ)
 $S_{\sigma,\bar{\delta}}$ be 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 :

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

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

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

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

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. Hence, 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 :

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

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

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

By replacing $S_{\sigma,\bar{\delta}}$ by T_σ , $S_{\bar{\sigma},\delta}$ by T_δ , by using commutativity of // , we get the system

$$\begin{cases} T = T_\sigma \cup T_\delta & \text{(defining 2-colorable series-parallel graphs)} \\ T_\sigma = T_\sigma // T_\sigma \cup T_\sigma \bullet T_\sigma \cup T_\delta \bullet T_\delta \\ T_\delta = e \cup T_\delta // T_\delta \cup T_\sigma \bullet T_\delta \cup T_\delta \bullet T_\sigma \end{cases}$$

Recognizability and inductive sets of properties

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

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

$$q, q' \in P, \quad q(a), \dots, q(b) \in \{True, False\}.$$

Proposition : A subset L of \mathbf{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 inductive proofs.

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 \mathbf{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”).

Next section: An inductive set of properties can be effectively constructed (at least theoretically!) from *every* monadic-second order formula.

Exercises

1) Construct congruences proving that the set of connected graphs is **HR**- and **VR**-recognizable.

2) Prove that the image of a recognizable language under an alphabetical homomorphism **h** (replacement of letter **a** by **h(a)**) is recognizable, by constructing a congruence for the image from one for the given language.

3) Prove the *Filtering Theorem*.

Hint : Let (L_1, \dots, L_n) be the least solution of a system **S** in $P(\mathbf{M})$ and $h : \mathbf{M} \rightarrow \mathbf{A}$ be a homomorphism with **A** finite. Construct a system **S'** with unknowns $x_{i,a}$ for all $i = 1, \dots, n$ and a in **A**, such that the component of the least solution of **S'** corresponding

to $x_{i,a}$ is $L_i \cap h^{-1}(a)$.

4) Prove “inductively” that every series-parallel graph is **3**-colorable.