Some language theoretical concepts extended to finite graphs

Bruno Courcelle

Université Bordeaux 1, LaBRI, and Institut Universitaire de France

References : Graph structure and monadic second-order logic, with J. Engelfriet Cambridge University Press, May 2012, readable on :
http://www.labri.fr/perso/courcell/ActSci.html

Conference presentations from this page.

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

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

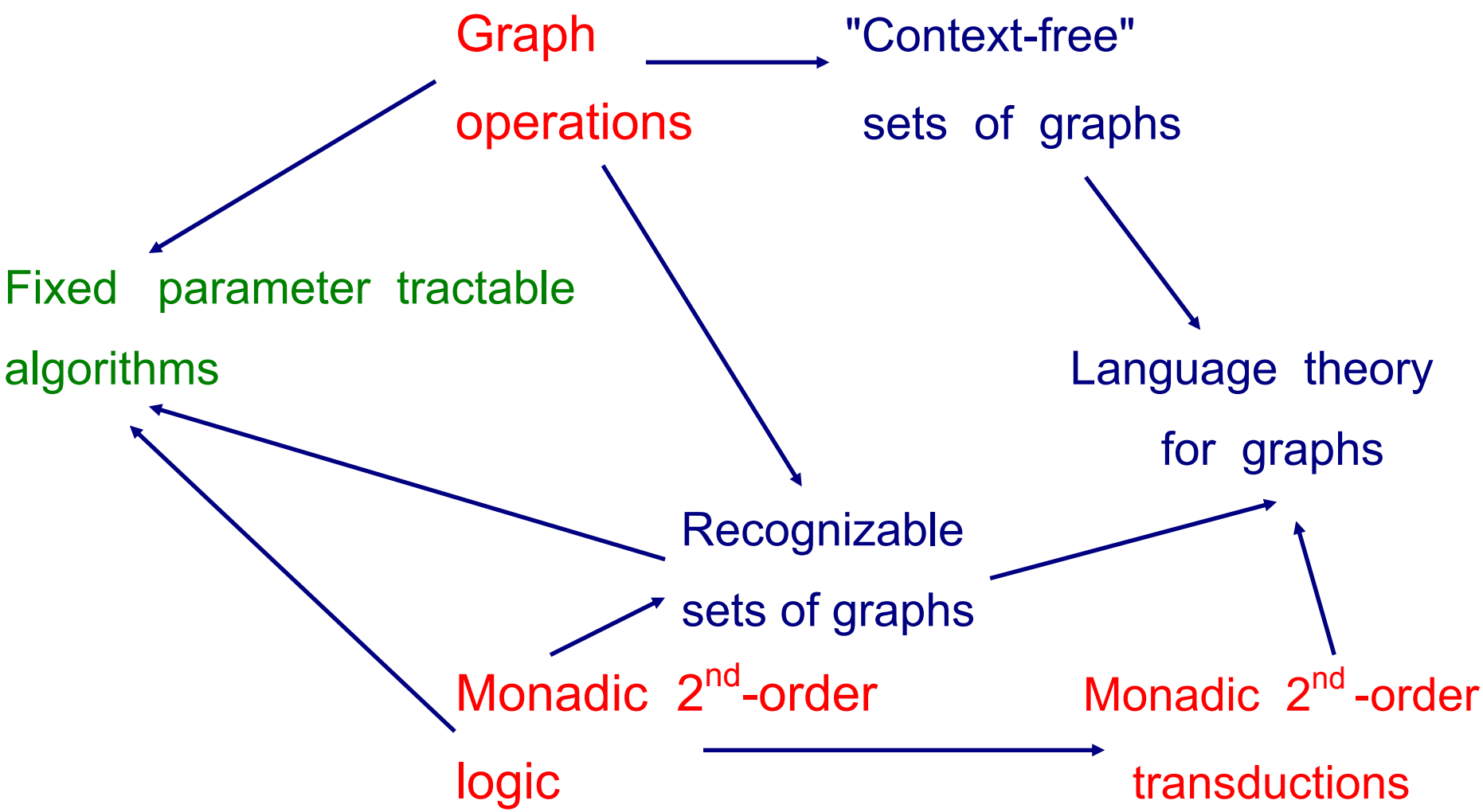
(FO = first-order, MS = monadic second-order, SO = second-order)

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

Context-free sets defined by equation systems

Two graph algebras; tree-width and clique-width

Recognizability : an algebraic notion

Monadic second-order logic

The Recognizability Theorem

Monadic second-order transductions.

Robustness results : preservation of classes under direct and inverse monadic second-order transductions.

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

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

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).

The pair of languages generated by **X** and **Y** is the least solution of the system of two equations. (Ginsburg & Rice, 1962)

In an arbitrary F-algebra $\mathbf{M} = \langle M, (f_{\mathbf{M}})_{f \in F} \rangle$ (F is a set of operations with arity),

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 a system.

This is well-defined in any algebra **M** and will be in *graph algebras*.

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

3) If **M** is "effectively given" and the components of the least solution of a system are finite sets, these sets can be computed by straightforward iteration.

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

5) Extensions of "Parikh's Theorem" (counting the vertices of generated graphs).

2. The graph algebras HR and VR

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

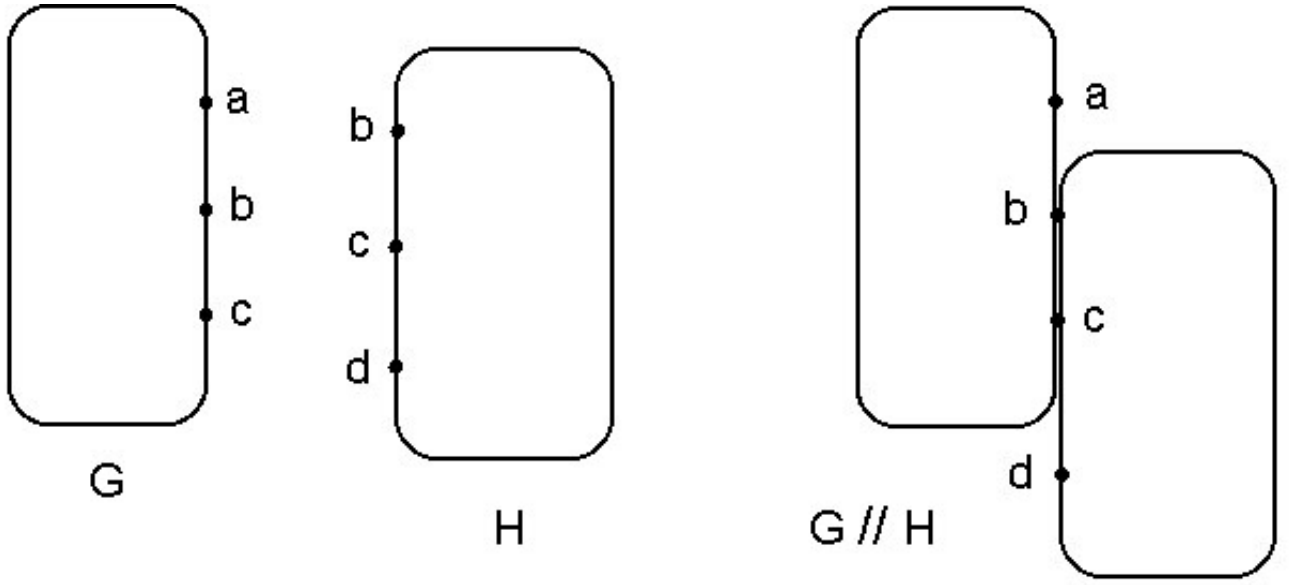
HR operations : Origin: **H**yperedge **R**eplacement *hypergraph grammars*
associated graph complexity measure : tree-width

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)

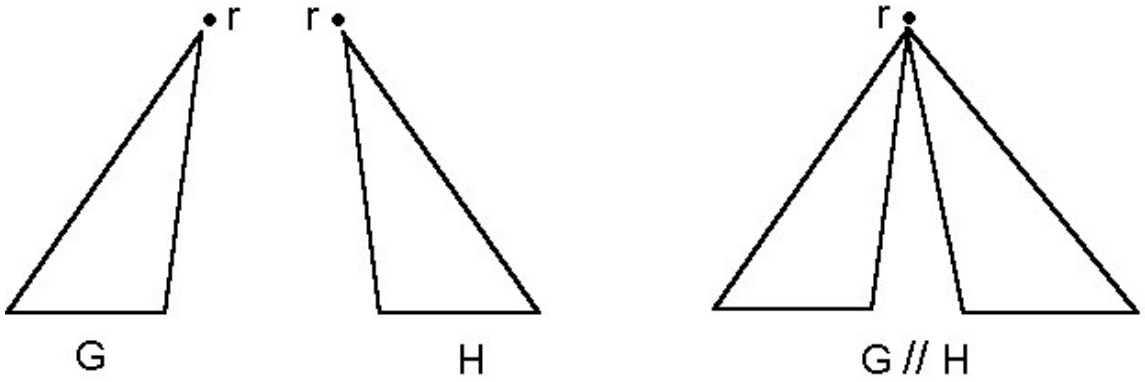
Nullary operations denote *basic graphs* : edge graphs, isolated vertices.

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).

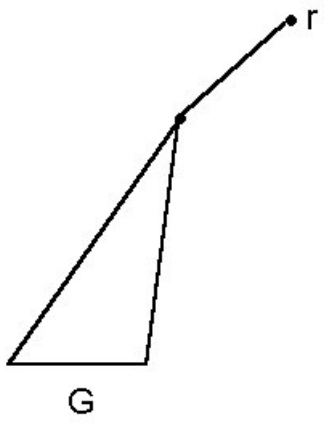
Fusion of two trees at their roots :



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 : $T = T\ //\ T \cup \text{extension}(T) \cup r$

Example : (Directed) series-parallel graphs

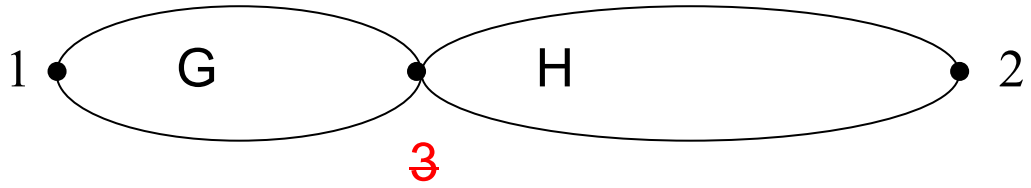
defined as directed graphs with sources 1 and 2,

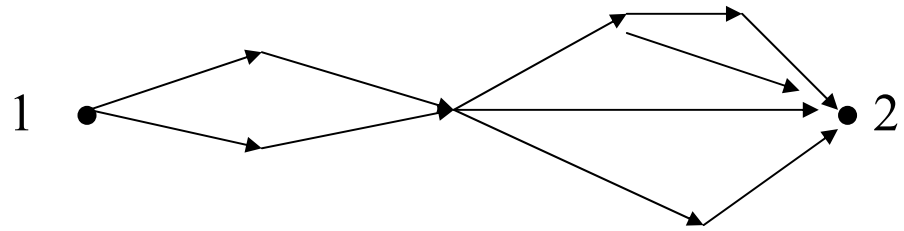
generated from e = 1 ⟶ 2 by the operations // (parallel-composition)

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

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

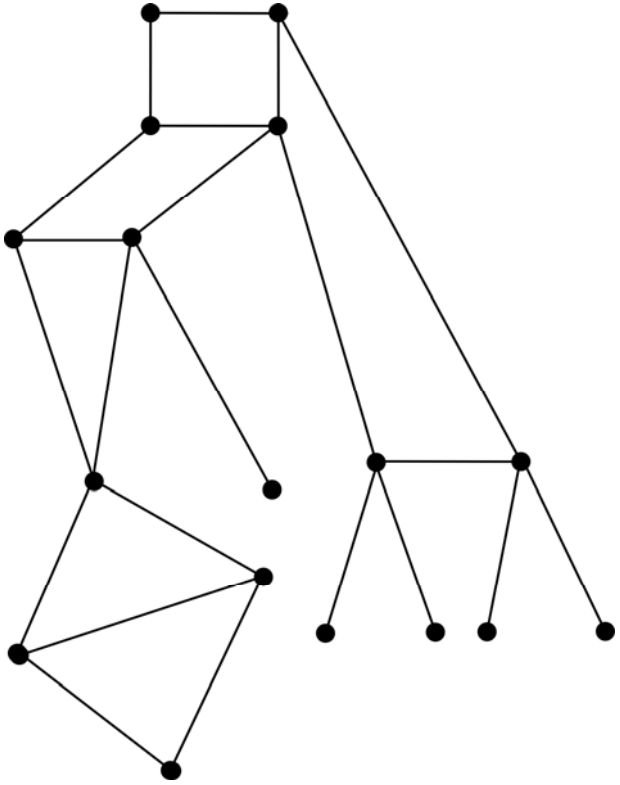
Example :



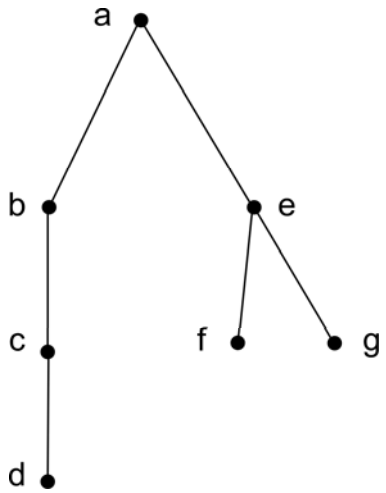


Their defining equation is : $S = S \,//\, S \cup S \bullet S \cup e$

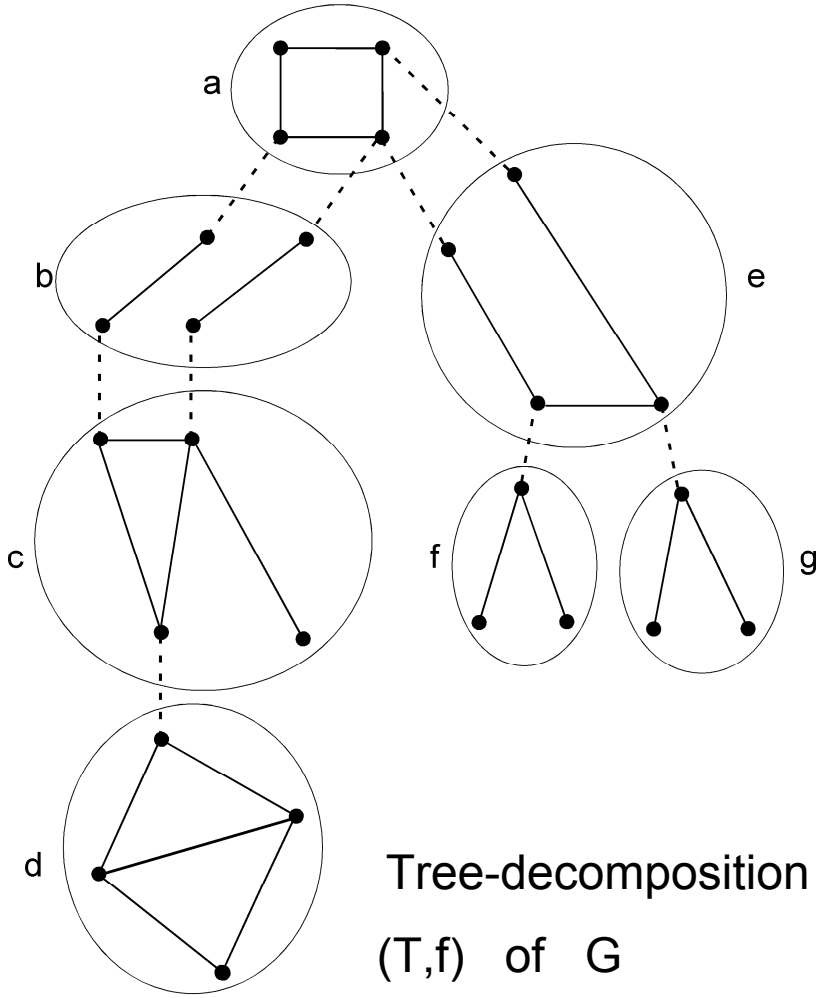
Relation to tree-decompositions and tree-width



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.

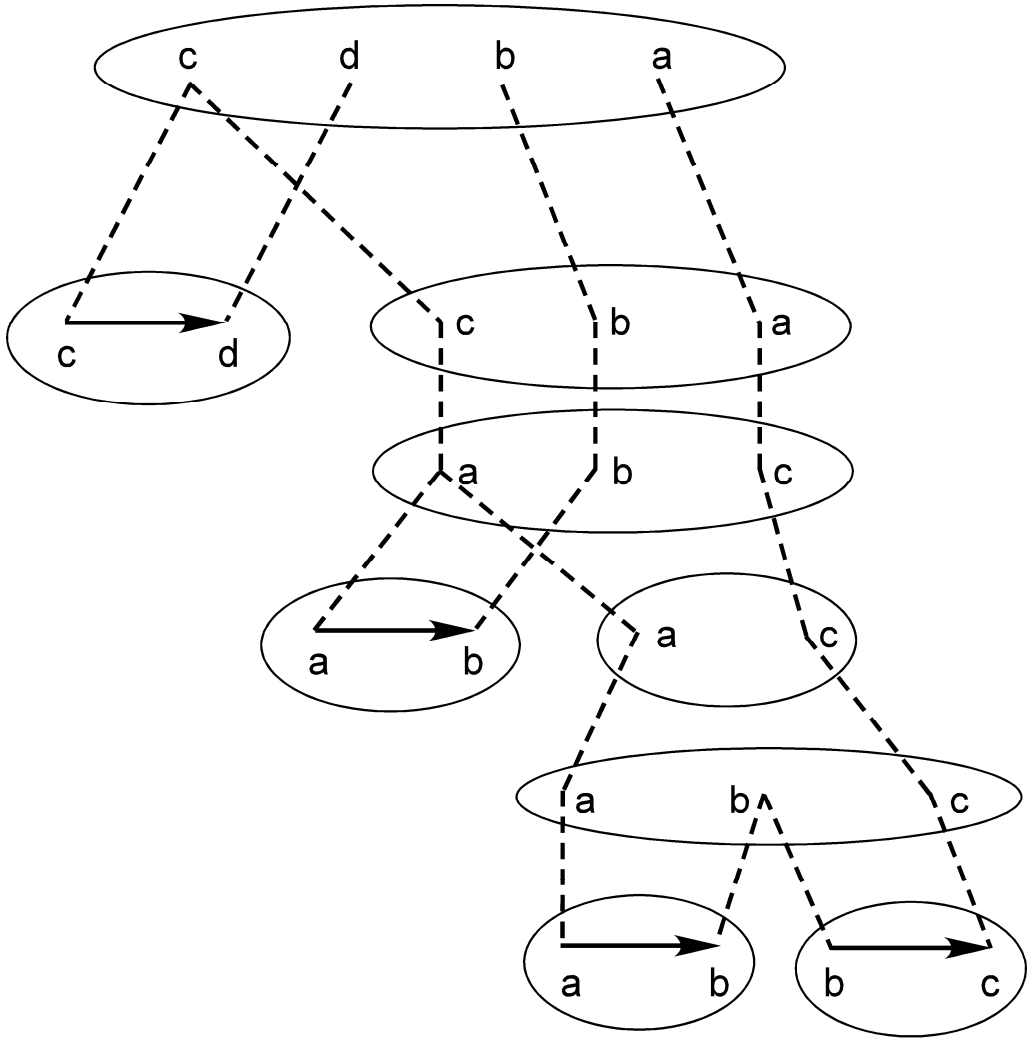
- The set of graphs of tree-width at most k is equational for each k.

- 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)

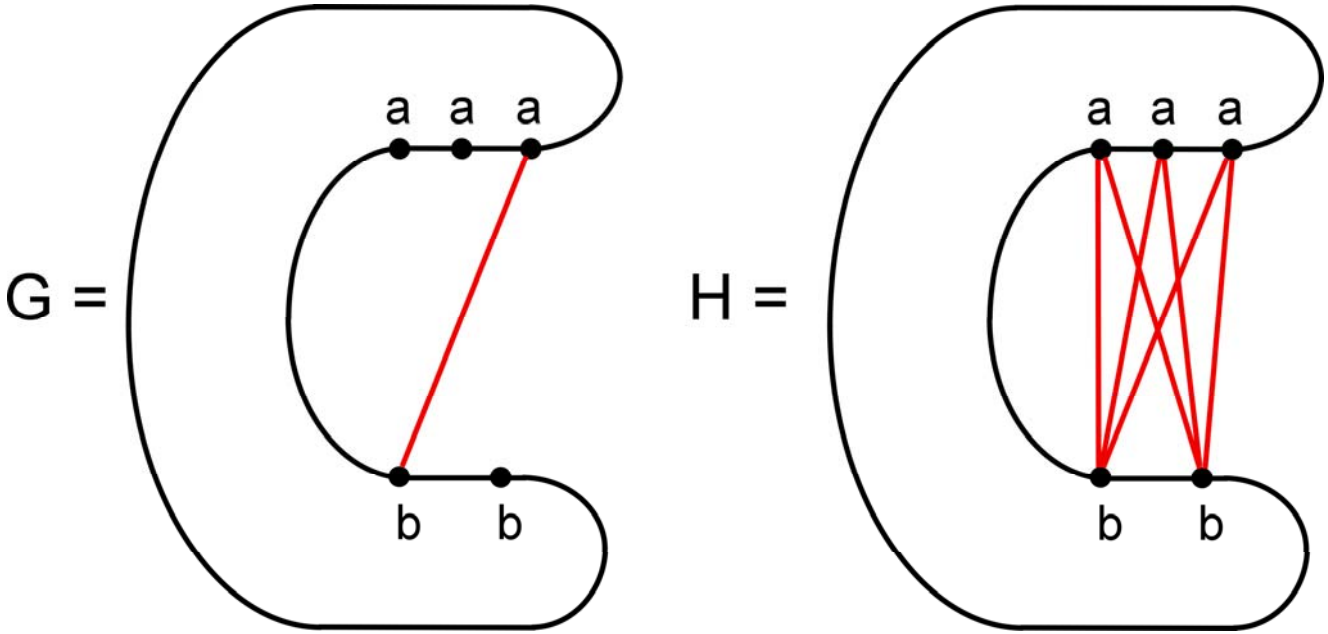
We use labels : *a , b , c, ..., d*.

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 :

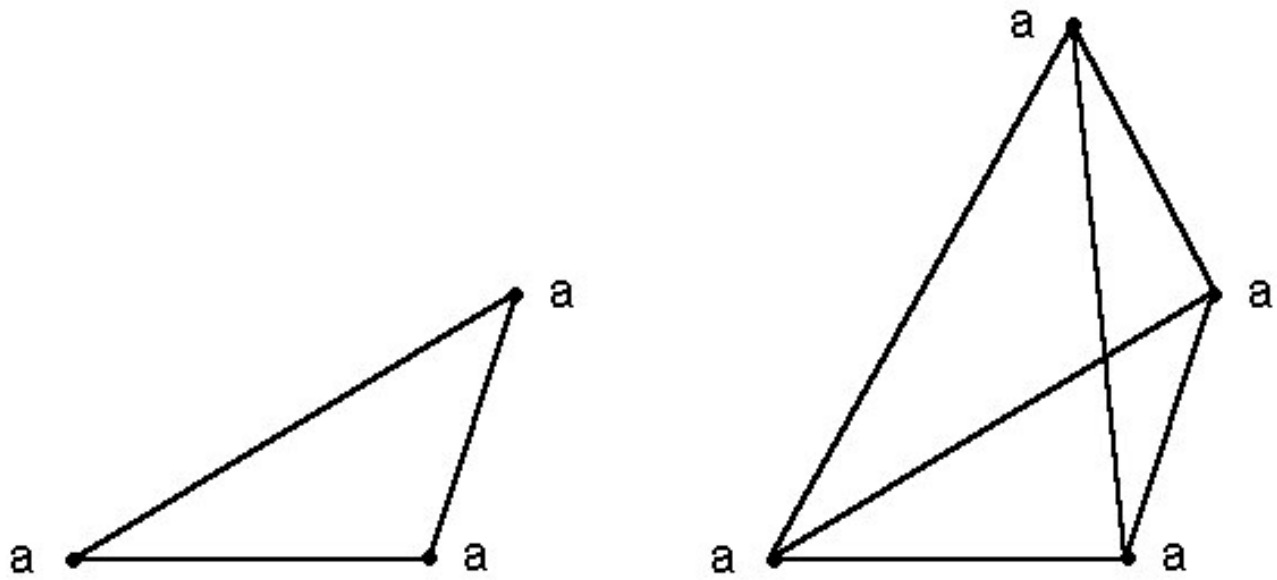
$Relab_{a \longrightarrow b}(G)$ is G with every vertex labelled by a relabelled into b

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

Example 1 : Cliques have clique-width 2.



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

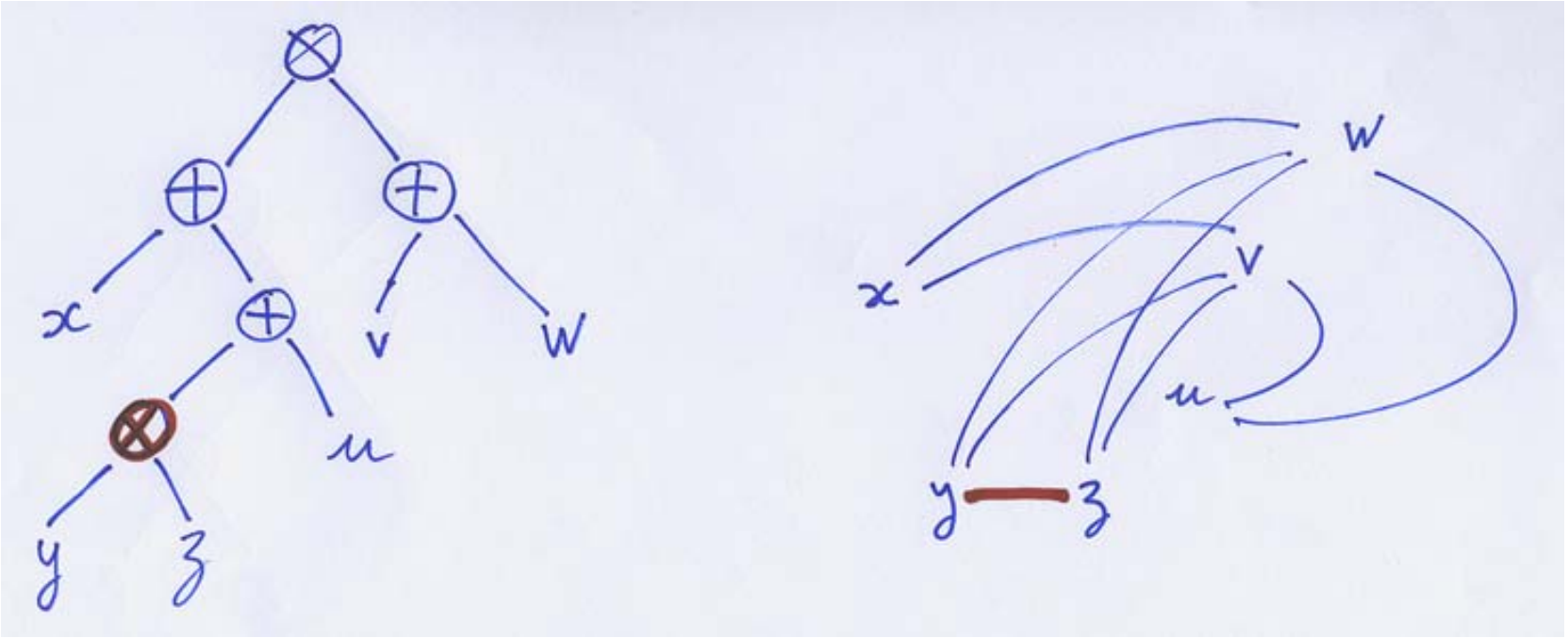
Cliques are defined by the equation :

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

Example 2 : Cographs 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 undirected edges between G and H.

They are defined by the equation : $C = C \oplus C \cup C \otimes C \cup *$



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.

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 ?

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.

3) Some properties following from the algebraic setting :

- Closure under operations
- Emptiness and finiteness are decidable
- Derivation trees
- Denotation of the generated graphs by terms,
- Upper bounds to tree-width and clique-width.

4) Others 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)

Applications (to be developped) :

Succint descriptions of infinite sets of finite graphs.

Drawing graphs in relation with the grammatical structure.

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 $\approx$ 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 $D \subseteq A$. (**A** : syntactic monoid for languages)

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

Skipping a difficulty: the algebras **HR** and **VR** have *infinite* signatures

oOo

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

Comparison of the two classes : Rec(**VR**) $\subseteq$ Rec(**HR**)

We have seen : Equat(**HR**) $\subseteq$ Equat(**VR**)

Intuition : **VR** has more powerful operations than **HR**.

However, we have equalities for sets of planar graphs or of graphs of bounded degree, or more generally, of graphs without some fixed $K_{n,n}$ 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.

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

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

Example : 2-colorable series-parallel graphs, see below.

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.

  Hence, 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

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

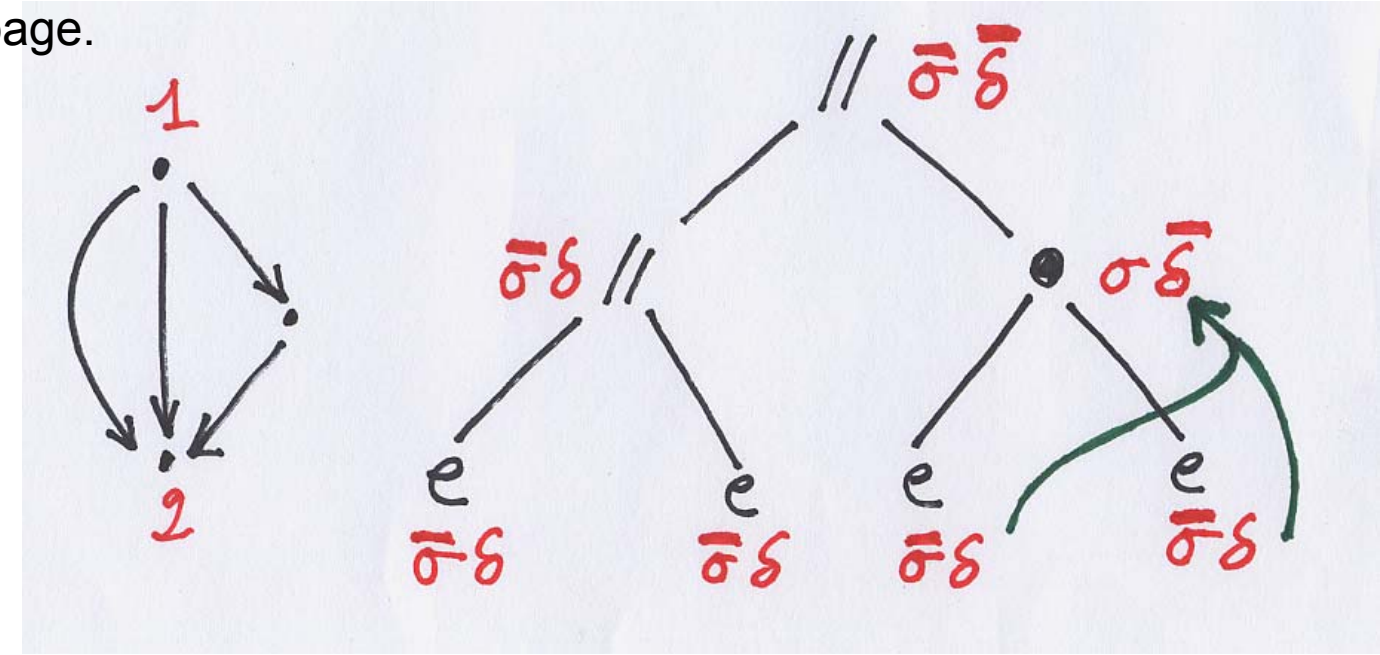
$\text{Same}(G \bullet H) \Leftrightarrow (\text{Same}(G) \wedge \text{Same}(H)) \vee (\text{Diff}(G) \wedge \text{Diff}(H))$
$\text{Diff}(G \bullet H) \Leftrightarrow (\text{Same}(G) \wedge \text{Diff}(H)) \vee (\text{Diff}(G) \wedge \text{Same}(H))$

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 :

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

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

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**

4. Monadic Second-Order (MS) Logic

A logical language that specifies inductive properties and functions:

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

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

G is 3-colorable :

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

Colors: $X \rightarrow 1$, $Y \rightarrow 2$, $V_G - X \cup Y \rightarrow 3$

G (undirected) is not connected :

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

oOo

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

$$\forall \, X \, \{ \text{“X is R-closed”} \, \wedge \, x \in X \, \Rightarrow y \in X \, \}$$

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 :

$$\forall x,y \, (x \in Y \, \wedge \, y \in Y \Rightarrow \, \mathbf{TC}(\text{“}u \in Y \, \wedge v \in Y \wedge edg(u,v)\text{”}, x, y)$$

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

Properties based on checking that two sets have same cardinality or on bijections.

Examples : All vertices of a graph have same degree.

A graph has a nontrivial automorphism.

A word has the form $a^n b^n$: the easiest context-free language that is not regular.

A word w = abbab is represented by the relational structure

< {1,2,3,4,5}, next(.,.) , $P_a(.)$,$P_b(.)$ > where next(1,2), next(2,3), …,

$P_a(1)$, $P_b(2)$, $P_b(3)$, $P_a(4)$, $P_b(5)$.

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

5. The Recognizability Theorem

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

(Doner, Thatcher & Wright, 1968 - 1970).

(2) A set of finite graphs of *clique-width* <k is VR-recognizable

$\Leftrightarrow$ the set of terms that define these graphs is recognizable

$\Leftarrow$ it is MS-definable

(3) A set of finite graphs of *tree-width* <k is HR-recognizable

$\Leftrightarrow$ the set of terms that define these graphs is recognizable

$\Leftarrow$ it is MS_2-ddefinable

Proofs: (2) and (3) : Several proofs can be given. The automata on terms are huge. The best is not to "compile" them but to compute the necessary transitions when needed. ("Fly-automata").

Proof with 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 auxiliary formulas of no larger quantifier-height in each of the two structures.

This is inductivity / recognizability.

For each h , the equivalence relation such that :

$G \approx H \Leftrightarrow$ the same sentences of quantifier-height $\leq h$ hold in G and H

is congruence; this proves the recognizability of any set of graphs defined by a sentence of quantifier-height $\leq$ h

Algorithmic consequences of the Recognizability Theorem

MS formulas

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

$O(f(k).n^3)$ for clique-width $\leq k$

finding a **VR**-term defining the graph is possible in cubic time

(Hlineny, Oum & Seymour)

MS_2 formulas

using edge quantifications

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

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

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

$O(g(k).n)$ for tree-width $\leq k$

finding a tree-decomposition (an **HR** -term) is possible in linear time (Bodlaender)

Language Theoretical consequences

One can filter out from HR- or VR-equational sets the graphs which do not satisfy a given MS_2- or MS-property 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.

6. Monadic second-order transductions

Let *C* and *D* be two classes of graphs with labels on edges and vertices, represented by relational structures. (Can be classes of terms or words.) An MS-transduction is a partial function

$\tau : C \times$ "data" $\rightarrow D$ specified by MS formulas.

Basic case : $\tau : C \rightarrow D$; G = τ(H) is defined "inside" H by MS formulas.

Examples : The edge –complement.

edg(x,y) in H is defined as $x \neq y \wedge \neg$ edg(x,y) in G

The transitive closure of a directed graph.

The reversal of a word.

Next case : $G = \tau(H, \text{“data”})$; the “data” is a tuple $X_1, \ldots, X_p$ of subsets of the domain of H ; these sets are called the parameters.

Parameters $X_1, \ldots, X_p$ are constrained to satisfy an MS property.

Examples : $(G, \{u\}) \longmapsto$ the connected component containing u.

$(G,X,Y,Z) \longmapsto$ the *minor* of G having vertex set X, resulting from the contraction of the edges of Y and the deletion of the edges and vertices of Z. (This transduction is $MS_{2,2}$; see below.)

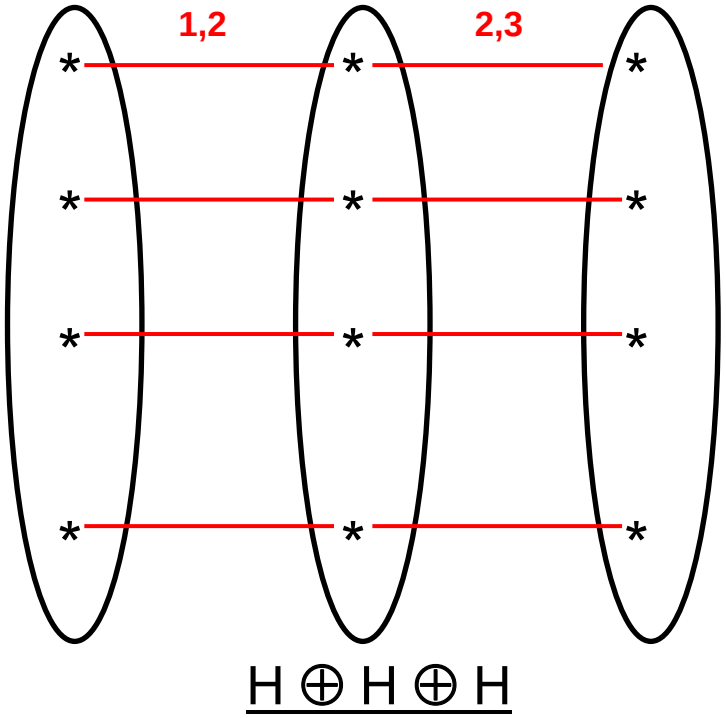
In the second example, the condition is that no two vertices of X should be linked by a path of edges in Y

$\tau(H) :=$ the set of all $G = \tau(H, X_1, \ldots, X_p)$

for all “good” tuples of parameters.

General case : G is defined as above inside

H ⊕ H ⊕ ... ⊕ H : disjoint copies of H with "marked" equalities of copied elements



The fundamental property of MS transductions

If $G \mid\!\longrightarrow \tau(G)$

then $\tau^{\#}(\psi) \longleftarrow\!\mid \psi$

Every MS formula ψ has an effectively computable *backwards translation* $\tau^{\#}(\psi)$, an MS formula such that :

$G \mid\!= \tau^{\#}(\psi)$ if and only if $\tau(G) \mid\!= \psi$

The verification of ψ in the object graph $\tau(G)$ reduces to the verification of $\tau^{\#}(\psi)$ in the given graph G (G contain all the necessary information to describe $\tau(G)$; the MS properties of $\tau(G)$ are expressible *in* G by MS formulas).

Theorem : The composition of two MS-transductions is an MS-transduction.

Example 1 (without parameters) : The *square* mapping δ on words : u $\mapsto$ uu

For u = aac, we have G = $\bullet \to \bullet \to \bullet$ (a a c)

$\underline{G \oplus G}$ $\bullet \to \bullet \to \bullet$ (p_1) $\quad \bullet \to \bullet \to \bullet$ (p_2) (marking edges omitted)

$\delta(G)$ $\bullet \to \bullet \to \bullet \to \bullet \to \bullet \to \bullet$ (a a c a a c)

In $\delta(G)$, we redefine *next* (i.e., $\to$) as follows :

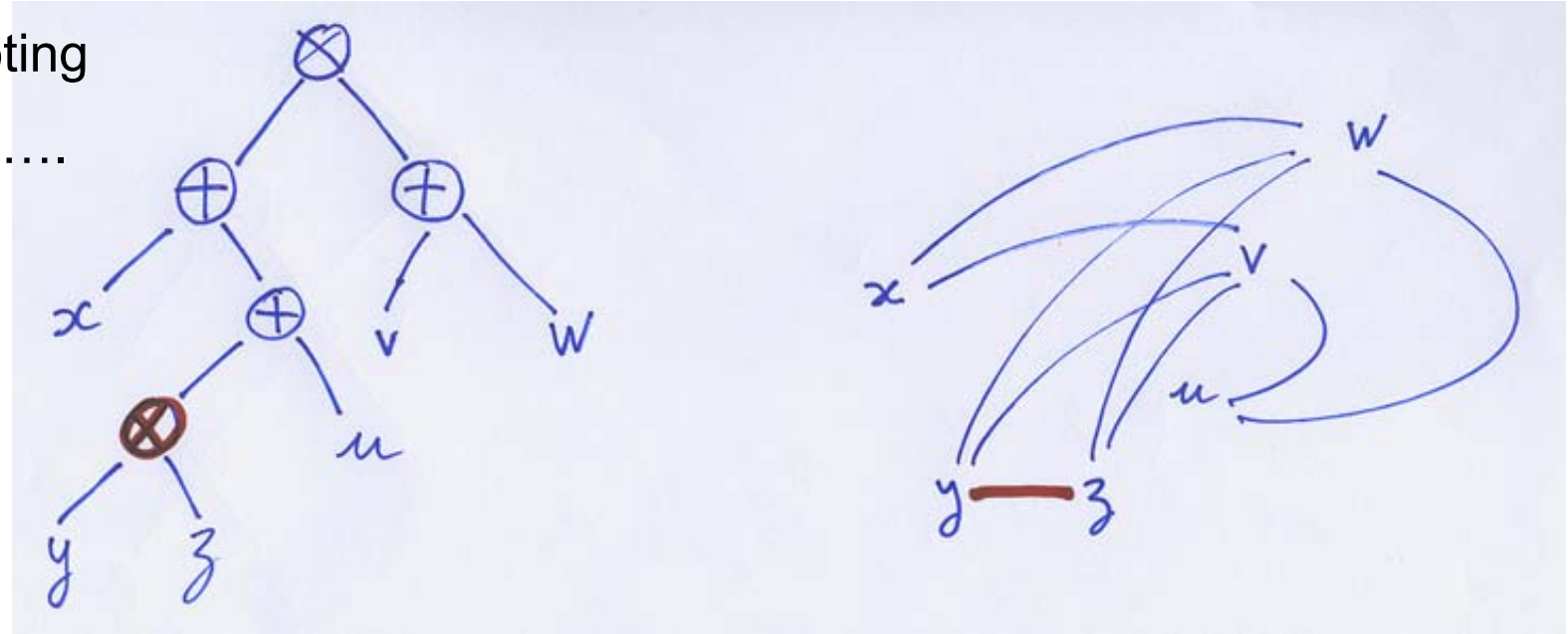
next(x,y) : $\Leftrightarrow$ (p_1 (x) $\wedge$ p_1 (y) $\wedge$ *next*(x,y)) **V** (p_2 (x) $\wedge$ p_2 (y) $\wedge$ *next*(x,y))

V (p_1 (x) $\wedge$ p_2 (y) $\wedge$ "x has no successor" $\wedge$ "y has no predecessor")

We also remove the "marker" predicates p_1, p_2.

Example 2 : From a term to a cograph

Terms are written with $\oplus$ (disjoint union), $\otimes$ (complete join) and constants x,y,z, … denoting vertices x,y,z ….



Vertices = {x,y,z,u,v,w } = occurrences of constants in the term.

Two vertices are adjacent if and only if their *least common ancestor* is labelled by $\otimes$ (like y and z , or u and w).

These conditions can be expressed by MS formulas on the labelled tree.

Edge quantification and edge description

There are 2 representations for an input graph and 2 for the output: type 1 : $G = (V_G, edg_G)$ and type 2 : $Inc(G) = (V_G \cup E_G, in_G)$.

Hence 4 types of graph transductions, denoted by :

$MS_{1,1}$ (or MS to simplify), $MS_{1,2,}$ $MS_{2,1}$ and $MS_{2,2}$

$MS_{i,o}$ means i = type of input, o = type of output.

I only use below MS and $MS_{2,2}$

Main Results (will be made more precise) :

(1) MS-transductions preserve bounded clique-width and the (corresponding) class of VR-equational sets.

(2) $MS_{2,2}$-transductions preserve bounded tree-width and the (corresponding) class of HR-equational sets.

Meaning : Robustness of the *two* graph hierarchies based on clique-width and tree-width.

MS - transductions and $MS_{2,2}$ - transductions are incomparable

Why ? For expressing graph properties, MS_2 logic is more powerful than MS_1 logic (the "ordinary" MS logic).

For building graphs with $MS_{2,2}$ - transductions, we have *more* possibilities of using the input graph, but we want *more* for the output : to specify each edge as a copy of some vertex or some edge of the input graph.

Transitive closure is MS (= $MS_{1,1}$) but not $MS_{2,2}$

Edge subdivision is $MS_{2,2}$ but not MS

Proofs : Easy since, if H is transformed into G by an MS-transduction :

$$| D_G | \leq k. | D_H | \quad \text{for fixed } k$$

7. Robustness results : Preservation of widths

For every class of graphs C :

1) C has bounded tree-width $\Leftrightarrow$ $C \subseteq \tau(\text{Trees})$ for some $MS_{2,2}$ – transduction τ (the proof is constructive in both directions)

Corollary: If C has tree-width $\leq$ k and τ is an $MS_{2,2}$ – transduction, then $\tau(C)$ has tree-width $\leq f_\tau(k)$

Hence: $MS_{2,2}$-transductions preserve bounded tree-width.

Very similarly:

2) C has bounded clique-width $\Leftrightarrow$ $C \subseteq \tau(\text{Trees})$ for some MS – transduction τ (the proof is constructive)

Similar corollary: MS-transductions preserve bounded clique-width.

Gives easy proofs (but no good bounds) of facts like :

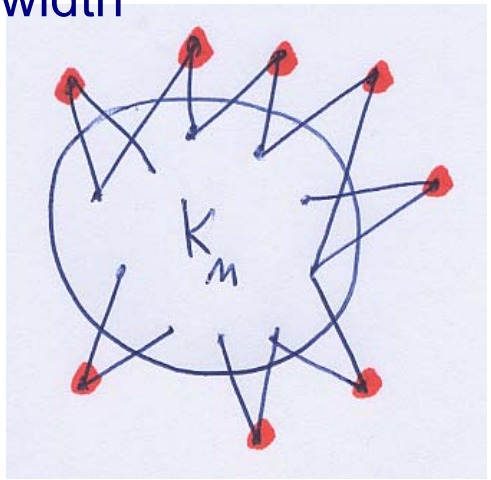
1) If C has bounded tree-width, its line graphs have bounded clique-width.

2) If C (directed graphs) has bounded tree-width or clique-width, the transitive closures of its graphs have bounded clique-width.

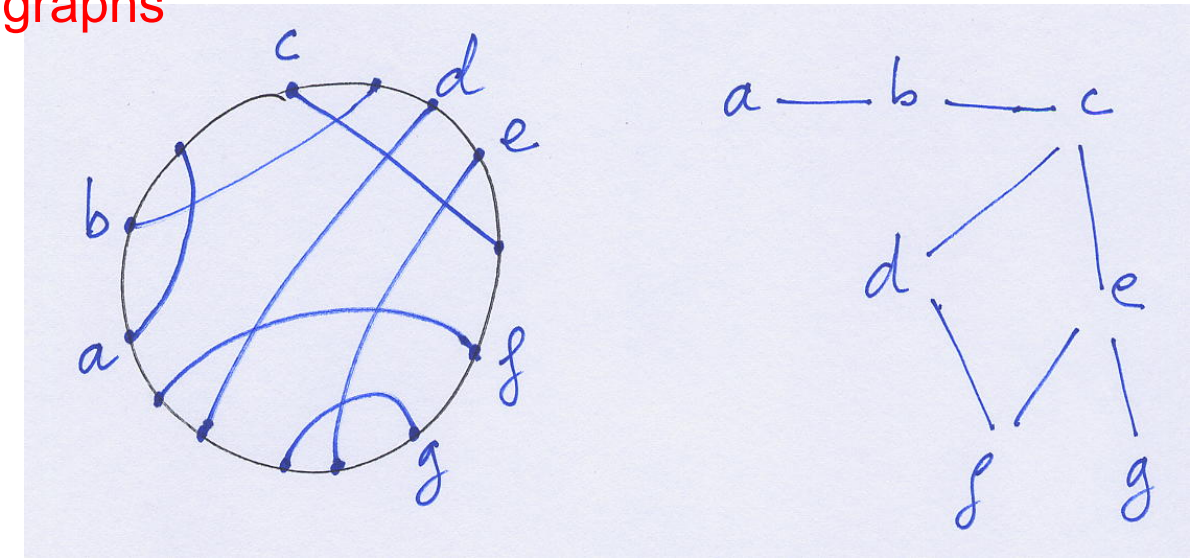
3) If C (directed graphs) has bounded clique-width, the transitive reductions of its graphs have bounded clique-width.

(Not trivial because clique-width is not monotone for subgraph inclusion).

4) The set of chordal graphs has *unbounded* clique-width because an MS transduction can define arbitrary graphs from chordal graphs, and graphs have *unbounded* clique-width. This transduction deletes each red vertex w and the edge u—v where u, v are adjacent to w.



5) Circle graphs



Chord diagram Δ Circle graph $G(\Delta)$

Theorem: Graphs Δ have bounded tree-width $\Leftrightarrow$ $G(\Delta)$ have bounded clique-width.

1) MS - transduction from $G(\Delta)$ to Δ ;

2) use "*split decomposition*" (Cunningham) and an MS-transduction from *prime* circle graphs to their unique chord diagrams.

Logical characterizations of equational sets

C is HR-equational $\Leftrightarrow$ $C = \tau(Trees)$ for some

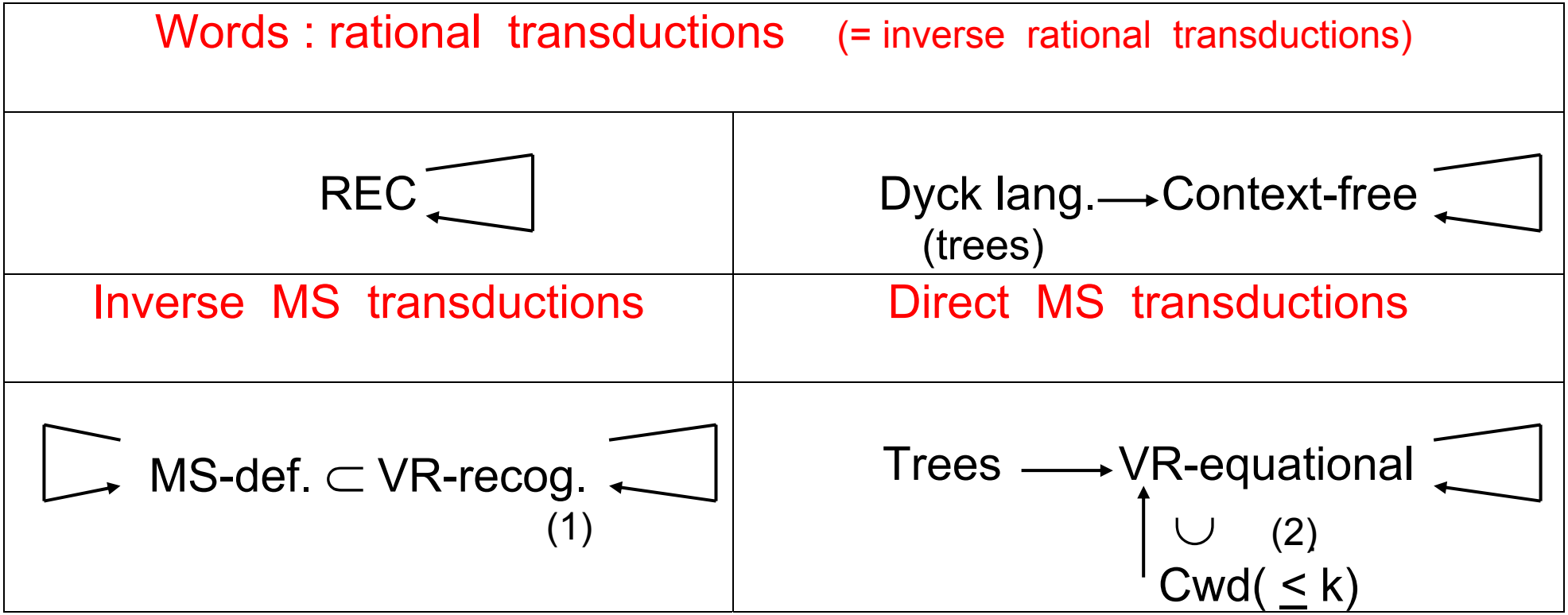
$MS_{2,2}$-transduction τ (for bounded tree-width we have $\subseteq$).

C is VR-equational $\Leftrightarrow$ $C = \tau(Trees)$ for some

MS - transduction τ (for bounded clique-width we have $\subseteq$).

Consequences : Closure of equational sets under the corresponding transductions.

Robustness results for HR- and VR-equational sets

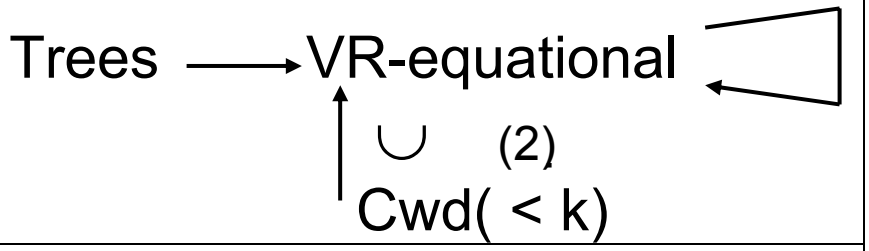
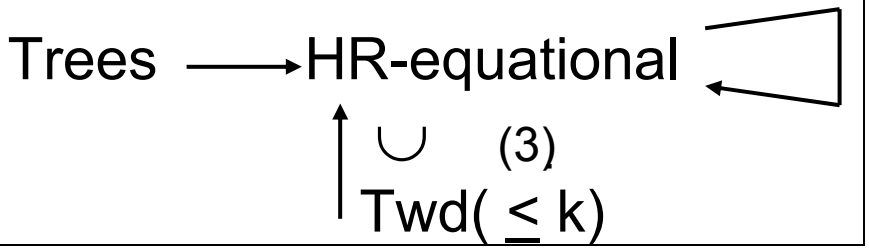


VR-equational $\Rightarrow$ bounded clique-width.

(1) : A. Blumensath - B.C.

(2) : J. Engelfriet.

Robustness results : Preservation and generation (2)

Inverse MS transductions	Direct MS transductions
	
Inverse $MS_{2,2}$ transductions	Direct $MS_{2,2}$ transductions
	

VR-equational $\Rightarrow$ bounded clique-width.

HR-equational $\Rightarrow$ bounded tree-width.

(1) : A. Blumensath - B.C. (2) : J. Engelfriet. (3) : B.C.- J. Engelfriet

8. A few open questions

1. What should be the clique-width of a hypergraph (or a relational structure), so that we have a polynomial time recognition algorithm ?

2. Is it true for a set of relational structures that the decidability of its MS theory implies bounded clique-width ?

3. When is it possible to specify a linear order by MS formulas ? (No in general, yes for connected graphs of degree $\leq k$).

4. Can one define by an MS transduction applied to a graph G of tree-width $\leq k$, a tree-decomposition of G of width $< k$? (Possible for $k \leq 3$).

5. Betweenness and circular order are represented by ternary relations. Is the consistency for betweenness or circular order of a set of triples MS-expressible?