A multivariate interlace polynomial and its computation for graphs of bounded clique-width

Bruno Courcelle*

Institut Universitaire de France and Bordeaux University, LaBRI

courcell@labri.fr

Submitted: Jul 31, 2007; Accepted: Apr 30, 2008; Published: May 5, 2008

Mathematics Subject Classifications: 05A15, 03C13

Abstract

We define a multivariate polynomial that generalizes in a unified way the two-variable *interlace* polynomial defined by Arratia, Bollobás and Sorkin on the one hand, and a one-variable variant of it defined by Aigner and van der Holst on the other.

We determine a recursive definition for our polynomial that is based on local complementation and pivoting like the recursive definitions of Tutte's polynomial and of its multivariate generalizations are based on edge deletions and contractions. We also show that bounded portions of our polynomial can be evaluated in polynomial time for graphs of bounded clique-width. Our proof uses an expression of the interlace polynomial in monadic second-order logic, and works actually for every polynomial expressed in monadic second-order logic in a similar way.

1 Introduction

There exist a large variety of polynomials associated with graphs, matroids and combinatorial maps. They provide information about *configurations* in these objects. We take here the word "configuration" in a wide sense. Typical examples are colorings, matchings, stable subsets, subgraphs. In many cases, a *value* is associated with the considered configurations : number of colors, cardinality, number of connected components or rank of the adjacency matrix of an associated subgraph. The information captured by a polynomial can be recovered in three ways: either by *evaluating* the polynomial for specific values of the indeterminates, or from *its zeros*, or by *interpreting the coefficients* of its monomials. We will consider the latter way in this article.

*This work has been supported by the GRAAL project of "Agence Nationale pour la Recherche" and by a temporary position of CNRS researcher. Postal address: LaBRI, F-33405 Talence, France

A *multivariate polynomial* is a polynomial with indeterminates depending on the vertices or the edges of the considered graph. Such indeterminates are sometimes called *colors* or *weights* because they make it possible to evaluate the polynomial with distinct values associated with distinct vertices or edges. Several multivariate versions of the dichromatic and Tutte polynomials of a graph have been defined and studied by Traldi in [30], by Zaslavsky in [31], by Bollobás and Riordan in [6] and by Ellis-Monaghan and Traldi who generalize and unify in [16] the previous definitions. Motivated by problems of statistical physics, Sokal studies in [29] a polynomial that will illustrate this informal presentation. The *multivariate Tutte polynomial* of a graph $G = (V, E)$ is defined there as:

$$Z(G) = \sum_{A \subseteq E} u^{k(G[A])} \prod_{e \in A} v_e$$

where $G[A]$ is the subgraph of G with vertex set V and edge set A, and $k(G[A])$ is the number of its connected components. This polynomial belongs to $\mathbb{Z}[u, v_e;\ e \in E]$. An indeterminate v_e is associated with each edge e. The indeterminates commute, the order of enumeration over each set A is irrelevant. We call such an expression an *explicit definition* of $Z(G)$, to be contrasted with its *recursive definition,* formulated as follows ([29], Formula (4.16)) in terms of edge deletions and contractions:

$$\begin{aligned} &Z(G) = u^{|V|} && \text{if } G \text{ has no edge,} \\ &Z(G) = Z(G[E - \{e\}]) + v_e \cdot Z(G/e) && \text{if } e \text{ is any edge,} \end{aligned}$$

where G/e is obtained from G by contracting edge e. From the fact that $Z(G)$ satisfies these equalities, it follows that they form a recursive definition which is *well-defined* in the sense that it yields the same result for every choice of an edge e in the second clause, i.e., for every tree of recursive calls.

There is no general method for constructing a recursive definition from an explicit one or proving that such a definition does not exist. The verification that a recursive definition is well-defined is not easy. This question is considered in depth in [6] and in [16] for multivariate Tutte polynomials. It is not easy either to determine an explicit definition (also called a *closed-form expression*) from a well-defined recursive one. Relating these different types of definitions by means of general tools is an open research direction.

Let us go back to the polynomial $Z(G)$. For two graphs G and G' with sets of edges in bijection, we have $Z(G) = Z(G')$ (where the variables indexed by edges of G and G' that are related by the bijection are considered as identical) if and only if $\mid V(G) \mid = \mid V(G') \mid$ and their cycle matroids are isomorphic (via the same bijection between edges). This observation explains what information about the considered graph is contained in the polynomial $Z(G)$. This polynomial is more general than Tutte's two-variable polynomial $T(G, x, y)$ because (see [29] for details) we have:

$$T(G, x, y) = ((x-1)^{k(G)}(y-1)^{|V|})^{-1} \alpha(Z(G))$$

where α is the substitution:

$$[u := (x-1)(y-1); v_e := y-1 \quad \text{for all} \quad e \in E].$$

Conversely, one can express the polynomial $Z'(G)$ defined as $\beta(Z(G))$ where β replaces every indeterminate v_e by the same indeterminate v in terms of $T(G,x,y)$ in a similar way. Hence, $Z'(G)$ and $T(G)$ are equivalent algebraically and in expressive power and also for the complexity of their computations.

In this article, we define a multivariate polynomial, that generalizes in a unified way the two-variable *interlace* polynomial defined in [3] and denoted by $q(G;x,y)$, its one-variable variant defined in [1] and denoted by $Q(G,x)$, and also, the *independence* polynomial surveyed in [23]. These polynomials have an older history. They are related with a polynomial defined by Martin in his 1977 dissertation for particular graphs, and later generalized to arbitrary graphs by Las Vergnas in [22], under the name of *Martin polynomial*. This polynomial is the generating function of the numbers of partitions of the edge set of a graph in k Eulerian subgraphs. It is defined for directed as well as for undirected graphs. (See Theorem 24 of [2] for the relationships between q and the Martin polynomial). Under the name of *Tutte-Martin polynomial*, Bouchet has extended it to isotropic systems and established relations between it and the polynomials q and Q in [8]. Relationships between interlace and Tutte polynomials are discussed in [1], [8] and [15].

Our multivariate polynomial is given by an explicit definition from which its specializations to the other known polynomials are immediate. We determine for it a recursive definition, somewhat more complicated than the usual ones for Tutte polynomials based on contracting and deleting edges. The known recursive definitions of the polynomials studied in [1,2,3] are derived via the corresponding specializations, with sometimes the necessity of proving auxiliary nontrivial properties.

Two other themes of this article are the evaluation and the computation of the multivariate interlace polynomial. These problems are known to be difficult. Bläser and Hoffmann show in [5] that the two-variable interlace polynomial is #P-hard to evaluate at every algebraic point of $\mathbb{R}^2$ except at those on some exceptional lines for which the complexity is either polynomial or open. On the other hand the multivariate interlace polynomial like other multivariate polynomials is of exponential size, hence cannot be computed in polynomial time. However we obtain efficient algorithms for evaluations and for computations of bounded portions of the interlace polynomials for graphs in classes of *bounded tree-width* and more generally of *bounded clique-width*. The proof uses descriptions of interlace polynomials by formulas of *monadic second-order logic*, and works actually for all polynomials expressible in a similar way.

Let us explain this logical aspect informally. Consider an explicit definition of a graph polynomial:

$$P(G) = \sum\nolimits_{C\in\Gamma(G)} n(C)\cdot v_C u^{f(C)}$$

where C ranges over all configurations of a multiset $\Gamma(G)$, $n(C)$ is the number of occurrences of C in $\Gamma(G)$, v_C is a monomial (like $\prod_{e\in A} v_e$ in the above polynomial $Z(G)$) that describes a configuration C, and $f(C)$ is the value of C. Polynomials of this form have necessarily positive coefficients. Their monomials have evident combinatorial interpretations arising from definitions. We are especially interested in cases where $\Gamma(G)$ and f can be expressed by *monadic second-order formulas*, i.e., formulas with quantifications on sets of objects, say sets of vertices or edges in the case of graphs, because there exist powerful methods for constructing algorithms for problems specified in this logical language and for graphs of bounded tree-width and clique-width. These basic facts, explained in detail in [11, 24, 25] will be reviewed in Section 5.

2 Definitions and basic facts

Graphs are finite, simple, undirected, possibly with loops, with at most one loop for each vertex. A graph is defined as a pair $G = (V_G, M_G)$ of a set of vertices V_G and a symmetric adjacency matrix M_G over GF(2). We omit the subscripts whenever possible without ambiguity. The *rank* $rk(G)$ of $G = (V, M)$ is defined as the rank $rk(M)$ of the matrix M over GF(2); its *corank (or nullity)* is $n(G) := n(M) :=\mid V \mid - rk(M)$. The empty graph $\varnothing$, defined as the graph without vertices (and edges) has rank and corank 0.

The set of *looped vertices* of G , i.e., of vertices i such that $M(i,i) = 1$ is denoted by $Loops(G)$. For a in V, we let $N(G,a)$ be the set of *neighbours* b of a, i.e., of vertices b adjacent to a with $b \neq a$. A looped vertex is *not* a neighbour of itself. For X a set of vertices of G, we denote by $G\nabla X$ the graph obtained by "toggling" the loops in X, i.e., $V_{G\nabla X} := V_G$ and:

$M_{G\nabla X}(i,j) := 1 - M_G(i,j)$ if $i = j \in X$,
$M_{G\nabla X}(i,j) := M_G(i,j)$ otherwise.

If X is a set of vertices, we let $G - X$ denote $G[V - X]$, the induced subgraph of G with set of vertices $V - X$. We write $G = H \oplus K$ if G is the union of disjoint subgraphs H and K. For two graphs G and H we write $H = h(G)$ and we say that they are *isomorphic by* h if h is a bijection of V_G onto V_H and $M_H(h(i), h(j)) = M_G(i,j)$ for all i and j.

Pivoting and local complementation

We first define the operation of *pivoting* on distinct vertices a and b of G. It yields the graph $H = G^{ab}$ defined as follows:

$V_H := V_G$ and
$M_H(i,j) := 1 - M_G(i,j)$ if $\{i,j\} \cap \{a,b\} = \varnothing$ and:
either $i \in N(G,a) - N(G,b)$ and $j \in N(G,b)$,
or $j \in N(G,a) - N(G,b)$ and $i \in N(G,b)$,
or $i \in N(G,b) - N(G,a)$ and $j \in N(G,a)$,
or $j \in N(G,b) - N(G,a)$ and $i \in N(G,a)$;
in all other cases, we let $M_H(i,j) := M_G(i,j)$.

This transformation does not depend on whether a and b are loops or are adjacent. It does not modify any loop. It only toggles edges between neighbours of a and b as specified above. It does not modifies the sets $N(G,a)$ and $N(G,b)$.

Next we define the *local complementation* at a vertex a of G. It yields the graph $H = G^a$ defined as follows:

$V_H := V_G$ and:

$M_H(i,j) := 1 - M_G(i,j)$ if $i, j \in N(G,a)$, including the case $i = j$.

$M_H(i,j) := M_G(i,j)$ otherwise.

Remarks

(1) We do not have $G^{ab} = (G^a)^b$; however, Lemma 1 (4,5,6) establishes relations between these operations.

(2) There is an inconsistency in [3]: the operation of local complementation is defined in Definition 4 in terms of a notion of neighbourhood, denoted by $\Gamma(a)$, such that a loop is a neighbour of itself, and G^a is like G except that the edges and loops in $G[\Gamma(a)]$ are toggled. With this definition, if a is looped it becomes isolated in G^a, and we do not have $(G^a)^a = G$. This definition *does not coincide* with the description given in terms of matrices two lines below, which corresponds to our definition. In proofs, the definition in terms of matrices is used. Actually, all statements in this article concern $G^a - a$ and not G^a alone, and $G^a - a$ is the same with the two possible interpretations of the definition of G^a.

Other notions of local complementation and pivoting exist in the literature. We will also use the following notion of local complementation:

$G * a := (G \nabla N(G,a))^a = G^a \nabla N(G,a)$.

This operation "toggles" the edges of $G[N(G,a)]$ that are not loops. It is used for graphs without loops in the characterization of circle graphs and in the definition of vertex-minors ([7, 13, 28]). Pivoting refers in these articles to the operation transforming G into $((G * a) * b) * a$, which is equal to $((G * b) * a) * b$ when a and b are adjacent. We will not need this notion of pivoting.

We will write $a \sim b$ to express that a and b are adjacent, both without loops. We write $a^\ell \sim b$ to express the same with a looped and b not looped, and $a^\ell \sim b^\ell$ if a and b are adjacent and both looped. The operations of local complementation and pivoting satisfy some properties listed in the following lemma:

Lemma 1: For every graph $G = (V, M)$, for distinct vertices a, b and all sets of vertices X, Y we have:

(1) $(G^a)^a = G$; $G^{ab} = G^{ba}$; $(G^{ab})^{ab} = G$;

(2) $(G \nabla X)^{ab} = G^{ab} \nabla X$; $(G \nabla X)^a = G^a \nabla X$; $(G \nabla X)[Y] = G[Y] \nabla (X \cap Y)$.

(3) $G[X]^{ab} = G^{ab}[X]$; $G[X]^a = G^a[X]$ if a and b are not in X;

(4) if $a \sim b$ or $a^\ell \sim b^\ell$ then $G^{ab} = h(((G^a)^b)^a \nabla a)$ and $(G^{ab})^b = h((G^a)^b \nabla a)$ where h is the permutation of V that exchanges a and b;

(5) if $a^\ell \sim b$ or $a \sim b^\ell$ then $G^{ab} = h(((G^a)^b)^a \nabla b)$ and $(G^{ab})^b = h((G^a)^b \nabla b)$ where h is the permutation of V that exchanges a and b;

(6) in cases (4) and (5) we have:

$G^{ab} - a - b = ((G^a)^b)^a - a - b$ and $(G^{ab})^b - a - b = (G^a)^b - a - b$.

Proof: (1)-(3) are clear from the definitions.

(4) We let $A = N(G, a) - N(G, b)$, $B = N(G, b) - N(G, a), C = N(G, a) \cap N(G, b)$ and $D = V_G - (N(G, a) \cup N(G, b))$.

In G we have $N(G, a) = A \cup C \cup \{b\}$ and $N(G, b) = B \cup C \cup \{a\}$. The first local complementation at a toggles edges and loops in $A, C, \{b\}$ and edges between A and C, b and C, b and A. It follows that $N(G^a, a) = N(G, a)$ and $N(G^a, b) = A \cup B \cup \{a\}$.

The local complementation at b toggles edges and loops in $A, B, \{a\}$ and edges between A and B, a and A, a and B. It follows that $N((G^a)^b, a) = B \cup C \cup \{b\}$ and $N((G^a)^b, b) = N(G^a, b)$.

The second local complementation at a toggles edges and loops in $B, C, \{b\}$ and edges between B and C, b and B, b and C. It follows that $N(((G^a)^b)^a, a) = N((G^a)^b, a) = B \cup C \cup \{b\}$ and $N(((G^a)^b)^a, b) = A \cup C \cup \{a\}$.

These three transformations toggle the edges between A and B, A and C and B and C, exactly as do the pivoting G^{ab}. They toggle twice the edges and loops in A, B, C, which yields no change. They toggle b twice, hence its loop status does not change. The loop status of a changes, and the operation ∇a reestablished the initial loop status of a.

Observe now that $N(((G^a)^b)^a, b) - \{a\} = N(G, a) - \{b\} = A \cup C$ and $N(((G^a)^b)^a, a) - \{b\} = N(G, b) - \{a\} = B \cup C$ and that a and b are both looped or both not looped in G^{ab} and in $((G^a)^b)^a \nabla a$. It follows then from the definition of G^{ab} that $G^{ab} = h(((G^a)^b)^a \nabla a)$ where h is the permutation of V that exchanges a and b.

This proves the first assertion of (4). For the second one we have:
$(G^{ab})^b = \big(h(((G^a)^b)^a \nabla a)\big)^b = h(((G^a)^b)^a \nabla a)^a) = h\big((((G^a)^b)^a)^a \nabla a\big) = h((G^a)^b \nabla a)$.

(5) The edges and loops are toggled in the same way as in case (4). The only difference concerns the loops at a or b. If $a^\ell \sim b$ in G, then $a \sim b$ in $((G^a)^b)^a$ and we have $a \sim b^\ell$ in $((G^a)^b)^a \nabla b$ and thus $h(a) \sim h(b)^\ell$ i.e. $a^\ell \sim b$ in $h(((G^a)^b)^a \nabla b)$ as well as in G^{ab}. Hence $G^{ab} = h(((G^a)^b)^a \nabla b)$.

The proof is similar if $a \sim b^\ell$ in G. The second assertion follows from the first one as in (4).

(6) Clear from (4) and (5) since h is the identity outside of $\{a, b\}$. □

Here is a lemma gathering facts about ranks in graphs.

Lemma 2: For every graph G, for distinct vertices a, b we have:
(1) $rk(G) = 1 + rk(G^a - a)$ if $a \in Loops(G)$;
(2) $rk(G) = 2 + rk(G^{ab} - a - b)$ if $a \sim b$;
(3) $rk(G - a) = rk(G^{ab} - a)$ if $a \sim b$ or if $a^\ell \sim b$.
(4) $rk(G) = 2 + rk((G^a)^b - a - b)$ if $a^\ell \sim b$.

Proof : (1) is proved in Lemma 5 of [3]. The proof is not affected by the inaccuracy observed above.

(2) and (3) are proved in Lemma 2 of [3].

(4) We note that $(G^a)^b - a - b = (G^a - a)^b - b$ and that $(G^a - a)$ has a loop on b. Hence by using (1) twice:

$rk((G^a)^b - a - b) = rk((G^a - a)^b - b) = rk(G^a - a) - 1 = rk(G) - 2$. □

3 The multivariate interlace polynomial

We will give definitions for graph polynomials. They can be easily adapted to other objects like matroids. A *multivariate polynomial* is a polynomial with indeterminates x_a, y_a, z_a,... associated with the vertices or edges a of the considered graph G. We will denote by $\mathbf{X}_G$ the set of such indeterminates for $X = \{x, y, z, \dots\}$. They are the *G-indexed indeterminates*. We denote by U a set $u, v, w, \dots$ of "ordinary" indeterminates not associated with elements of graphs.

By a *polynomial* $P(G)$, we mean a mapping P that associates with a graph G a polynomial in $\mathbb{Z}[U \cup \mathbf{X}_G]$ such that if h is an isomorphism of G onto H, then $P(H)$ is obtained from $P(G)$ by the substitution that replaces x_a by $x_{h(a)}$ for every x_a in $\mathbf{X}_G$.

A *specializing substitution* is a substitution that replaces an indeterminate from a finite set $U = \{u, v, w, \dots\}$ by a polynomial in $\mathbb{Z}[U]$, and a G-indexed indeterminate x_a in $\mathbf{X}_G$, by a polynomial in $\mathbb{Z}[U \cup \{y_a \mid y \in X\}]$, the same for each a. For an example, such a substitution can replace x_a by $y_a(x-1)^2 - 3z_a u + 1$ for every vertex a. If σ is a specializing substitution, then $\sigma \circ P$, defined by $\sigma \circ P(G) = \sigma(P(G))$ for every G is a polynomial in the above sense.

For a set A of vertices we let x_A abbreviate the product (in any order) of the commutative indeterminates x_a, for a in A. If $A = \varnothing$, then $x_A = 1$. If $\mathcal{B}$ is a set of subsets of G, then the polynomial $\sum_{A \in \mathcal{B}} x_A$ describes exactly $\mathcal{B}$. A multiset of sets $\mathcal{B}$ is described by the polynomial $\sum_{A \in \mathcal{B}} n(A) \cdot x_A$ where $n(A)$ is the number of occurrences of A in $\mathcal{B}$.

Definition 3: *The multivariate interlace polynomial.*

For a graph G we define

$$C(G) = \sum\nolimits_{A,B \subseteq V, A \cap B = \varnothing} x_A y_B u^{rk((G \nabla B)[A \cup B])} v^{n((G \nabla B)[A \cup B])}.$$

Hence $C(G) \in \mathbb{Z}[\{u, v\} \cup \mathbf{X}_G\ ]$ where $X = \{x, y\}$. Let us compare it with the existing interlace polynomials. The one-variable interlace polynomial of [2] is only defined recursively. We will denote it by $q_N(G, y)$, as in [3]. It is called the *vertex-nullity interlace polynomial*, and a closed-form expression is determined in [1]:

$$q_N(G, y) = \sum\nolimits_{A \subseteq V} (y-1)^{n(G[A])}.$$

It follows that this polynomial is obtained from $C(G)$ by a substitution. We have $q_N(G, y) = \sigma'(C(G))$ where σ' is the substitution:

$$[u := 1; v := y - 1; x_a := 1, y_a := 0 \text{ for all } a \in V].$$

The interlace polynomial q of [3] is defined by:

$$q(G; x, y) = \sum\nolimits_{A \subseteq V} (x-1)^{rk(G[A])}(y-1)^{n(G[A])}.$$

It is equal to $\sigma(C(G))$ where σ is the substitution:

$$[u := x-1; v := y-1; x_a := 1, y_a := 0 \text{ for all } a \in V].$$

Another polynomial denoted by Q is defined recursively in [1] for graphs without loops, and the following explicit expression is obtained:

$$Q(G,x) = \sum\nolimits_{A,B\subseteq V, A\cap B=\varnothing} (x-2)^{n((G\nabla B)[A\cup B])}$$

Hence, $Q(G,x) = \tau(B(G))$ where τ is the substitution:

$$[u := 1; v := x-2; x_a := y_a := 1 \text{ for all } a \in V].$$

Note that although $Q(G,x)$ is intended to be defined for graphs without loops, its definition is based on the co-ranks of graphs obtained from G by choosing two disjoint subsets of vertices, A and B, by adding loops at the vertices of B and taking the graph of G induced on $A \cup B$. It corresponds to the *global Tutte-Martin polynomial of the isotropic system* presented by G, whereas q_N corresponds to the *restricted Tutte-Martin polynomial* of this isotropic system. These correspondences are established in [1] and [8].

Our motivation for introducing sets B of toggled loops in the definition of $C(G)$ is to obtain a common generalization of $q(G;x,y)$ and $Q(G,x)$ and to handle loops in a homogenous way without making a particular case of graphs without loops.

Let $C_1(G)$ be the polynomial obtained from $C(G)$ by replacing v by 1.

Lemma 4: For every graph G and every set T of vertices:
(1) $C(G) = \theta(C_1(G))$ where $\theta := [u := uv^{-1}; x_a := vx_a; y_a := vy_a$ for all $a \in V]$,
(2) $C(G\nabla T) = \mu(C(G))$ where $\mu := [x_a := y_a, y_a := x_a$ for all $a \in T]$.

Note that we slightly extend the notion of substitution by allowing the substitution of uv^{-1} for u.

Proof: (1) Clear.
(2) We observe that $((G\nabla T)\nabla B)[A\cup B] = (G\nabla(A'\cup B'))[A\cup B]$ where $A' = A\cap T, B' = B - B\cap T$. The result follows. $\square$

We will write: $C = \theta \circ C_1$. The polynomial $C(G)$ can thus be "recovered" from $C_1(G)$. Since every graph G is $G_1\nabla T$ for some T with G_1 without loops, we have $C(G) = \mu(C(G_1))$ where μ is as in Lemma 4. Hence, it is enough to know $C(G)$ for graphs G without loops. However, the recursive definitions to be considered below will introduce graphs with loops in the recursive calls.

Properties of polynomials

The polynomial q defined above satisfies for all graphs G the equality

$$q(G-a) - q(G-a-b) = q(G^{ab}-a) - q(G^{ab}-a-b) \quad \text{if } a \sim b \tag{1}$$

and the polynomial Q satisfies for all graphs G without loops:

$$Q(G * a) = Q(G) \tag{2}$$
$$Q(G^{ab}) = Q(G) \qquad \text{if } a \sim b. \tag{3}$$

Do these equalities hold for $C(G)$? The answer is no for (2) and (3) as a consequence of the next proposition, and also for (1): see below Counter-example 14.

Proposition 5: A graph G and its polynomial $C(G)$ can be reconstructed from $\rho(C(G))$ where $\rho := [v := 1; y_a := 0$ for all $a \in V]$.

Proof: For every set of vertices A, the rank of $G[A]$ is the unique integer n such that $x_A u^n$ is a monomial of $\rho(C(G))$. Now a vertex a has a loop if $rk(G[a]) = 1$, and no loop if $rk(G[a]) = 0$. Hence, we obtain $Loops(G)$ from $\rho(C(G))$. Using this information, we can reconstruct edges as follows.

If a and b are not looped, they are adjacent if and only if $rk(G[\{a,b\}]) = 2$, otherwise $rk(G[\{a,b\}]) = 0$. If one of a, b is looped, they are adjacent if and only if $rk(G[\{a,b\}] = 2$, otherwise $rk(G[\{a,b\}]) = 1$. If both are looped, they are adjacent if and only if $rk(G[\{a,b\}]) = 1$, otherwise $rk(G[\{a,b\}]) = 2$. □

It follows that identities (2) and (3) cannot hold for C and even for $\rho \circ C$.

Remark: This proposition shows that G and thus $C(G)$ can be reconstructed *algorithmically* from $\rho(C(G))$. But $C(G)$ is not definable *algebraically* from $\rho(C(G))$, that is by a substitution.

3.1 Recursive definition

We now determine a *recursive definition* of $C(G)$ (also called a set of *reduction formulas*), from which we can obtain again the recursive definitions given in [3] and in [1].We let a denote the graph with one non-looped vertex a, and a^ℓ denote the similar graph with one looped vertex a.

Lemma 6: For every graph G, for every graph H disjoint from G we have:
(1) $C(\varnothing) = 1$
(2) $C(G \oplus H) = C(G) \cdot C(H)$
(3) $C(a) = 1 + x_a v + y_a u$
(4) $C(a^\ell) = 1 + x_a u + y_a v$.

Proof : Easy verification from the definitions. □

The more complicated task consists in expressing $C(G)$ in the case where a and b are adjacent (this is necessary if no rule of Lemma 6 is applicable). We will distinguish three cases: $a \sim b$, $a^\ell \sim b$, and $a^\ell \sim b^\ell$.

For a graph G and disjoint sets of vertices A and B, we let $m(G, A, B)$ denote the monomial $x_A y_B u^{rk((G\nabla B)[A\cup B])} v^{n((G\nabla B)[A\cup B])}$ so that $C(G)$ is nothing but the sum of these monomials over all pairs A, B (the condition $A \cap B = \varnothing$ will be assumed for each use of the notation $m(G, A, B)$).

For distinct vertices a, b, two disjoint sets A, B can contain a, b or not according to 9 cases. We let $i \in \{0, 1, 2\}$ mean that a vertex is in $V - (A \cup B)$, in A or in B respectively. Let C_{ij} be the sum of monomials $m(G, A, B)$ such that i tells where is a, and j tells where is b. For an example: C_{02} is the sum of monomials $m(G, A, B)$ such that $a \in V - (A\cup B)$, $b \in B$.

Claim 7 : Let G be such that $a \sim b$.
(1) $C_{00} = C(G - a - b)$
(2) $C_{11} = x_a x_b u^2 \cdot C(G^{ab} - a - b)$.
(3) $C_{20} = y_a u \cdot C(G^a - a - b)$; $\quad C_{02} = y_b u \cdot C(G^b - a - b)$;
(4) $C_{12} = x_a y_b u^2 \cdot C((G^b)^a - a - b)$; $\quad C_{21} = x_b y_a u^2 \cdot C((G^a)^b - a - b)$.

Proof: (1) Clear from the definitions.
(2) A monomial of C_{11} is of the form:

$$m(G, A, B) = x_A y_B u^{rk((G\nabla B)[A\cup B])} v^{n((G\nabla B)[A\cup B])} \tag{4}$$

with $a, b \in A$ (because of subscript 11). By Lemma 2 (2) we have:

$$rk((G\nabla B)[A \cup B]) = 2 + rk((G\nabla B)[A \cup B]^{ab} - a - b).$$

But $(G\nabla B)[A \cup B]^{ab} - a - b = ((G^{ab} - a - b)\nabla B)[A' \cup B]$ where $A' = A - a - b$ (we use here Lemma 1 (2,3)). Hence:

$$m(G, A, B) = x_a x_b u^2 \cdot m(G^{ab} - a - b, A', B).$$

It follows that:

$$C_{11} = x_a x_b u^2 \cdot C(G^{ab} - a - b)$$

because the set of pairs $A', B \subseteq V - a - b$ such that A' and B are disjoint coincides with the set of pairs $(A - a - b), B$ such that $A, B \subseteq V$, A and B are disjoint and $a, b \in A$.

(3) The proof is similar. A monomial of C_{20} is of the form $m(G, A, B)$ described by Equality (4) with $a \in B$, $b \notin A \cup B$ (because of the subscript 20). By Lemma 2 (1) we have:

$$rk((G\nabla B)[A \cup B]) = 1 + rk((G\nabla B)[A \cup B]^a - a)$$

because a is looped in $(G\nabla B)[A \cup B]$. But:

$$(G\nabla B)[A \cup B]^a - a = (((G\nabla a)^a - a - b)\nabla B')[A \cup B']$$

because $b \notin A\cup B$, and with $B' = B - a$. (By Lemma 1 (2,3)). Clearly, $(G\nabla a)^a - a - b = G^a - a - b$. Hence $m(G, A, B) = y_a u \cdot m(G^a - a - b, A, B')$. It follows that:

$$C_{20} = y_a u \cdot C(G^a - a - b)$$

because the set of pairs $A, B' \subseteq V - a - b$ such that A and B' are disjoint coincides with the set of pairs $A, (B-a)$ such that $A, B \subseteq V$, A and B are disjoint, $a \in B$ and $b \notin A \cup B$. The case of C_{02} is obtained by exchanging a and b.

(4) A monomial of C_{12} is of the form (4) above with $a \in A$, $b \in B$. By Lemma 2 (4) we have:

$$rk((G\nabla B)[A \cup B]) = 2 + rk(((G\nabla B)[A \cup B]^b)^a - a - b)$$

because $b^\ell \sim a$ in $G\nabla B[A \cup B]$. We have:

$$((G\nabla B)[A \cup B]^b)^a - a - b = (((G^b)^a - a - b)\ \nabla B')[A' \cup B']$$

where $A' = A - a, B' = B - b$. Hence:

$$m(G, A, B) = x_a y_b u^2 \cdot m((G^b)^a - a - b, A', B').$$

It follows that:

$$C_{12} = x_a y_b u^2 \cdot C((G^b)^a - a - b)$$

because the set of pairs $A', B' \subseteq V - a - b$ such that A' and B' are disjoint coincides with the set of pairs $(A-a), (B-b)$ such that $A, B \subseteq V$, A and B are disjoint, $a \in A$ and $b \in B$. The case of C_{21} is obtained similarily by exchanging a and b. $\square$

The next claim establishes linear relations between some polynomials C_{ij}.

Claim 8: Let G be such that $a \sim b$.
(1) $C(G - a) = C_{00} + C_{01} + C_{02}$
(2) $C(G - b) = C_{00} + C_{10} + C_{20}$
(3) $y_a u \cdot C(G^a - a) = C_{20} + C_{21} + C_{22}$
(4) $y_b u \cdot C(G^b - b) = C_{02} + C_{12} + C_{22}$

Proof : (1), (2) Clear from the definitions.
(3) From the definitions, $C_{20} + C_{21} + C_{22}$ is the sum of monomials $m(G, A, B)$ such that $a \in B$. We have:

$$rk((G\nabla B)[A \cup B]) = 1 + rk((G\nabla B)[A \cup B]^a - a)$$

by Lemma 2(1). But:
$(G\nabla B)[A \cup B]^a - a = (((G\nabla a)^a - a)\nabla B')[A \cup B']$ (where $B' = B - a$)
$= ((G^a - a)\nabla B')[A \cup B']$.
This gives the result with the usual argument.
(4) Similar to (3) by exchanging a and b. $\square$

If we collect the equalities of Claims 7 and 8 we have 10 definitions or linear equalities for 9 "unknowns". This is enough for obtaining $C(G)$. We get thus:

$$C(G) = (C_{00} + C_{10} + C_{20}) + \{C_{01} + C_{11} + C_{21}\} + (C_{02} + C_{12} + C_{22})$$

$= C(G-b) + \{C_{01} + x_a x_b u^2 \cdot C(G^{ab} - a - b) + x_b y_a u^2 \cdot C((G^a)^b - a - b)\}$
$\quad + y_b u \cdot C(G^b - b).$

Then $C_{01} = C(G-a) - C_{00} - C_{02} = C(G-a) - C(G-a-b) - y_b u \cdot C(G^b - a - b)$.

We obtain by reorganizing and factorizing the expression:

Lemma 9 : Let G be such that $a \sim b$. We have:
$C(G) = x_b u^2 \{x_a \cdot C(G^{ab} - a - b) + y_a \cdot C((G^a)^b - a - b)\}$
$\quad + y_b u\{C(G^b - b) - C(G^b - a - b)\} + C(G-a) + C(G-b) - C(G-a-b).$

Considering C_{22} for which we have two expressions, we get:

Corollary 10: Let G be such that $a \sim b$.
$y_b\{C(G^b - b) - C(G^b - a - b) - x_a u \cdot C((G^a)^b - a - b)\}$
$= y_a\{C(G^a - a) - C(G^a - a - b)) - x_b u \cdot C((G^b)^a - a - b)\}.$

Next we consider the cases where $a^\ell \sim b$ and $a^\ell \sim b^\ell$. Actually, Lemma 4 (2) will shorten the computations.

Lemma 11: (1) Let G be such that $a \sim b^\ell$.
$C(G) = y_b u^2\{x_a \cdot C(G^{ab} - a - b) + y_a \cdot C((G^a)^b - a - b)\}$
$\quad + x_b u\{C(G^b - b) - C(G^b - a - b)\} + C(G-a) + C(G-b) - C(G-a-b).$
(2) Let G be such that $a^\ell \sim b^\ell$.
$C(G) = y_b u^2\{y_a \cdot C(G^{ab} - a - b) + x_a \cdot C((G^a)^b - a - b)\}$
$\quad + x_b u\{C(G^b - b) - C(G^b - a - b)\} + C(G-a) + C(G-b) - C(G-a-b).$

Proof: (1) We have $G = G_1 \nabla b$, $G_1 = G \nabla b$, where in G_1 we have $a \sim b$ so that Lemma 9 is applicable. We get then, letting β be the substitution that exchanges x_b and y_b:

$C(G) = \beta(C(G_1))$
$= y_b u^2\{x_a \cdot C((G\nabla b)^{ab} - a - b) + y_a \cdot C(((G\nabla b)^a)^b - a - b)\}$
$\quad + x_b u\{C((G\nabla b)^b - b) - C((G\nabla b)^b - a - b)\}$
$\quad + \beta(C(G\nabla b - a)) + C(G\nabla b - b) - C(G\nabla b - a - b)$
$= y_b u^2\{x_a \cdot C(G^{ab} - a - b) + y_a \cdot C((G^a)^b - a - b)\}$
$\quad + x_b u\{C(G^b - b) - C(G^b - a - b)\}$
$\quad + C(G-a) + C(G-b) - C(G-a-b),$

For this equality, we use the facts that $(G\nabla b)^{ab} - a - b = G^{ab} - a - b$ and that $C((G\nabla b)^{ab} - a - b)$ has no occurrence of an indeterminate indexed by b, that $((G\nabla b)^a)^b - a - b = (G^a)^b - a - b$ and that $C((G\nabla b^a)^b - a - b)$ has no occurrence of an indeterminate indexed by b. We also use similar remarks concerning $(G\nabla b)^b - b$, $(G\nabla b)^b - a - b$, $G\nabla b - b$, and $G\nabla b - a - b$. Finally, we have $\beta(C(G\nabla b - a)) = C(G-a)$ by Lemma 1 (2) and Lemma 4.

(2) Very similar argument. □

We can now sum up the results of Lemmas 6, 9, 11 into the following proposition, where the three cases are collected into a single one with help of the little trick of introducing "meta-indeterminates" z_c, w_c for each $c \in V$:

$z_c = x_c$ and $w_c = y_c$ if c is not a loop,
$z_c = y_c$ and $w_c = x_c$ if c is a loop.

Proposition 12: For every graph G, for every graph H disjoint from G, every vertex a, we have:

(1) $C(\varnothing) = 1$
(2) $C(G \oplus H) = C(G) \cdot C(H)$
(3) $C(a) = 1 + x_a v + y_a u$
(4) $C(a^\ell) = 1 + x_a u + y_a v$
(5) $C(G) = z_b u^2\{z_a \cdot C(G^{ab} - a - b) + w_a \cdot C((G^a)^b - a - b)\}$
$+ w_b u\{C(G^b - b) - C(G^b - a - b)\}$
$+ C(G - a) + C(G - b) - C(G - a - b).$
if $b \in N(G, a)$.

Proof: Immediate consequence of Lemmas 6,9,11. □

We have an even shorter expression:

Corollary 13: For every graph G and every vertex a, we have:

(1) $C(\varnothing) = 1$
(2) $C(G) = (1 + z_a v + w_a u)C(G - a)$ if $N(G, a) = \varnothing$,
(3) $C(G) = z_b u^2\{z_a \cdot C(G^{ab} - a - b) + w_a \cdot C((G^a)^b - a - b)\}$
$+ w_b u\{C(G^b - b) - C(G^b - a - b)\}$
$+ C(G - a) + C(G - b) - C(G - a - b).$
if $b \in N(G, a)$.

Counter-example 14:

Proposition 8 of [3] states that if $a \sim b$ in G then:

$$q(G - a) - q(G - a - b) = q(G^{ab} - a) - q(G^{ab} - a - b).$$

This is not true for C in place of q. To see this let G be the graph with four vertices a, b, c, d and three edges such that $c \sim a \sim b \sim d$. Note that G^{ab} is G augmented with an edge between c and d. Assume we would have:

$$C(G - a) - C(G - a - b) = C(G^{ab} - a) - C(G^{ab} - a - b). \tag{5}$$

In the left handside, we have a single monomial of the form $y_b y_c x_d u^n$ for some n, and it must be from $C(G - a)$ because b is not in $G - a - b$. This monomial is $y_b y_c x_d u^3$ because $rk(c^\ell \oplus (d \sim b^\ell)) = 3$. In the right handside we have the monomial $y_b y_c x_d u^2$ because $rk(c^\ell \sim d \sim b^\ell) = 2$. Hence we cannot have Equality (5). □

In such a case, we can ask what is the less specialized substitution σ such that the corresponding equality is true for $\sigma \circ C$? Some answers will be given below. We prove actually a more complicated identity.

Proposition 15: If $a \sim b$ in G then:

$$C(G-a)-C(G-a-b)-C(G^{ab}-a)+C(G^{ab}-a-b) = y_b u\{C(G^b-a-b)-C((G^a)^b-a-b)\}.$$

Proof: We use the notation and some facts from Claims 7 and 8:

$$C(G-a)-C(G-a-b) = C_{01}+C_{02} = C_{01}+y_b u \cdot C(G^b-a-b).$$

We let C^{ab}_{01} and C^{ab}_{02} denote the polynomials C_{01} and C_{02} relative to (G^{ab}, a, b) instead of to (G, a, b). Then we have :

$$C(G^{ab}-a)-C(G^{ab}-a-b) = C^{ab}_{01}+C^{ab}_{02} = C^{ab}_{01}+y_b u \cdot C((G^{ab})^b-a-b).$$

We have by Lemma 1: $(G^{ab})^b-a-b = (G^a)^b-a-b$.

On the other hand, C^{ab}_{01} is the sum of monomials:

$$m(G^{ab}, A, B) = x_A y_B u^{rk((G^{ab}\nabla B)[A\cup B])} v^{n((G^{ab}\nabla B)[A\cup B])}$$

for disjoint sets A, B such that $a \notin A\cup B$, $b \in A$. But for such A, B:

$$(G^{ab}\nabla B)[A\cup B] = (G\nabla B)[A\cup B\cup a]^{ab}-a.$$

Hence, using Lemma 1 and Lemma 2 (3):

$$\begin{aligned} &rk((G^{ab}\nabla B)[A\cup B]) = rk((G\nabla B)[A\cup B\cup a]^{ab}-a)\\ &= rk((G\nabla B)[A\cup B\cup a]-a)\\ &= rk((G\nabla B)[A\cup B]).\end{aligned}$$

We have also $n((G^{ab}\nabla B)[A\cup B]) = n((G\nabla B)[A\cup B])$. Hence, $m(G^{ab}, A, B) = m(G, A, B)$ and $C^{ab}_{01} = C_{01}$. Collecting these remarks we get:

$$\begin{aligned} &C(G-a)-C(G-a-b)-C(G^{ab}-a)+C(G^{ab}-a-b)\\ &\quad = C_{02}-C^{ab}_{02}\\ &\quad = y_b u \cdot C(G^b-a-b)-y_b u \cdot C((G^a)^b-a-b). \ \square\end{aligned}$$

We note for later use that Identity (5) holds if either $u = 0$ or $y_b = 0$ for all b.

A polynomial P in $\mathbb{Z}[X]$ is said to be *positive* if the coefficients of its monomials are positive. A mapping P from graphs to polynomials is *positive* if $P(G)$ is positive for every G. It is clear from Definition 3 that C is positive. This not immediate from the recursive definition of Corollary 13 because of two substractions in the right handside of the third clause. However, one can derive from Corollary 13 a stronger statement that is not immediate from Definition 3.

Proposition 16: For every graph G and every vertex a, the polynomials $C(G)$ and $C(G)-C(G-a)$ are positive.

Proof: By induction on the number of vertices of G, one proves simultaneously these two assertions by using Corollary 13.

In case (2) we have:
$C(G)-C(G-a)=(1+z_a v+w_a u)C(G-a)$
and in case (3) we have
$$\begin{aligned}C(G)-C(G-a) &= z_b u^2\{z_a\cdot C(G^{ab}-a-b)+w_a\cdot C((G^a)^b-a-b)\}\\ &+w_b u\{C(G^b-b)-C(G^b-b-a)\}\\ &+C(G-b)-C(G-b-a),\end{aligned}$$
which gives with the induction hypothesis that $C(G)-C(G-a)$ is positive. So is $C(G)$ since, again by induction, $C(G-a)$ is positive. $\square$

It would remain to give a combinatorial explanation of this fact.

4 Specializations to known polynomials

We have already observed that the polynomials q of [3] and Q of [1] can be obtained by specializing substitutions from $C(G)$. For more clarity with the substitutions of indeterminates we will use u' and v' instead of x and y in these polynomials. So we will study: $q(G;u',v')=\sigma(C(G))$ where σ is the substitution:

$$[u:=u'-1; v:=v'-1; x_a:=1, y_a:=0 \ \text{ for all } \ a\in V],$$

and the polynomial Q, defined for graphs without loops by $Q(G,v')=\tau(C(G))$ where τ is the substitution

$$[u:=1; v:=v'-2; x_a:=y_a:=1 \ \text{ for all } \ a\in V].$$

Both are actually specializations of the following two polynomials. We let:
$C_{y=0}(G):=\sigma_0(C(G))$ where σ_0 is the substitution $[y_a:=0$ for all $a\in V]$,
and
$C_{x=y}(G):=\sigma_=(C(G))$ where $\sigma_=$ is the substitution $[y_a:=x_a$ for all $a\in V]$.

The polynomials $C, C_{x=y}, C_{y=0}$ are by definition positive. The polynomial Q is also positive: this follows from the recursive definition established in [1] that we will reprove in a different way, but this is not obvious from the above definition, because of the term $v'-2$.

4.1 Fixed loops

The polynomial $C_{y=0}(G)$ can be written, for a graph that can have loops:

$$C_{y=0}(G)=\sum_{A\subseteq V} x_A u^{rk(G[A])}v^{n(G[A])}.$$

Configurations are reduced to sets A of vertices, and there is no second component B for toggling loops. Hence loops are "fixed" in the configurations defining the polynomial as they are in G. Clearly $q(G;u',v') = \sigma'(C_{y=0}(G))$ where σ' is the substitution:

$$[u := u'-1; v := v'-1; x_a := 1 \ \text{ for all } \ a \in V].$$

The polynomial q is not positive: if G is reduced to an edge, we have $q(G) = u'^2 - 2u' + 2v'$.

Proposition 17: For every graph G and every vertex a, we have:
(1) $C_{y=0}(\varnothing) = 1$,
(2) $C_{y=0}(G) = (1 + x_a v)C_{y=0}(G - a)$ if $N(G,a) = \varnothing$ and a is not a loop,
(3) $C_{y=0}(G) = x_a u \cdot C_{y=0}(G^a - a) + C_{y=0}(G - a)$ if a is a loop, isolated or not,
(4) $C_{y=0}(G) = x_b x_a u^2 \cdot C_{y=0}(G^{ab} - a - b) +$
$+ C_{y=0}(G - a) + C_{y=0}(G - b) - C_{y=0}(G - a - b)$ if $a \sim b$.

Proof: (1), (2), (4): Immediate from Corollary 13.

(3) If a is isolated, this follows from Corollary 13 (2). Otherwise, using the notation of the proof of Claim 7 we observe that $C_{y=0}(G)$ is the sum of monomials $m(G, A, \varnothing)$; those such that $a \notin A$ yield $C(G - a)$, the others yield $x_a u \cdot C_{y=0}(G^a - a)$ since:

$$rk(G[A]) = rk(G[A]^a - a) + 1 = rk((G^a - a)[A - a]) + 1$$

by Lemma 2(1). This gives the result, however, it is interesting to see what Lemma 11 gives. The two cases where $a^\ell \sim b$ and $a^\ell \sim b^\ell$ yield the same equality.

$$C_{y=0}(G) = x_a u\{C_{y=0}(G^a - a) - C_{y=0}(G^a - a - b)\}$$
$$+ C_{y=0}(G - a) + C_{y=0}(G - b) - C_{y=0}(G - a - b).$$

Hence we have to check that:

$$x_a u \cdot C_{y=0}(G^a - a - b) = C_{y=0}(G - b) - C_{y=0}(G - a - b).$$

This is nothing but Assertion (3) applied to $H = G - b$. Hence (3) can be established by induction on the size of G, with help of Lemma 11, and without repeating the analysis of the monomials $m(G, A, \varnothing)$. $\square$

This proposition yields, with easy transformations, the following recursive definition of q:
(q1) $q(G) = v'^n$ if G consists of n isolated non-looped vertices,
(q2) $q(G) = (u'-1)q(G^a - a) + q(G - a)$ if a is a loop, isolated or not,
(q3) $q(G) = (u'-1)^2 q(G^{ab} - a - b) +$
$+ q(G - a) + q(G - b) - q(G - a - b)$ if $a \sim b$.

However, the recursive definition of q in Proposition 6 of [3] uses rules (q1), (q2) and the following one:

(q3') $q(G) = ((u'-1)^2-1)q(G^{ab}-a-b) + q(G-a) + q(G^{ab}-b) \qquad$ if $a \sim b$
instead of (q3). We will now prove the equivalence of both sets of rules. The following corollary of Proposition 15 generalizes Proposition 8 of [3]:

Corollary 18: If $a \sim b$ in G then:

$$C_{y=0}(G-a) - C_{y=0}(G-a-b) = C_{y=0}(G^{ab}-a) - C_{y=0}(G^{ab}-a-b).$$

Proof: Immediate from Proposition 15 since $y_b = 0$ for all b. $\square$

We get thus the following corollary.

Corollary 19: For every graph G and every vertex a, we have:
(1) $C_{y=0}(\varnothing) = 1$
(2) $C_{y=0}(G) = (1+x_a v)C_{y=0}(G-a) \qquad$ if $N(G,a) = \varnothing$ and a is not a loop,
(3) $C_{y=0}(G) = x_a u \cdot C_{y=0}(G^a - a) + C_{y=0}(G-a) \qquad$ if a is a loop, isolated or not,
(4) $C_{y=0}(G) = (x_b x_a u^2 - 1)C_{y=0}(G^{ab}-a-b) +$
$+C_{y=0}(G-a) + C_{y=0}(G^{ab}-b) \qquad$ if $a \sim b$.

If we apply to these rules the substitution σ' we find the rules of Proposition 6 of [3]. Hence, Corollary 19 lifts at the multivariate level the recursive definition of this article.

4.2 Toggled loops made invisible in the polynomial

We now consider the polynomial $C_{x=y}(G) := \sigma_=(C(G))$ where $\sigma_=$ is the substitution $[y_a := x_a$ for all $a \in V]$. This gives:

$$C_{x=y}(G) = \sum\nolimits_{A,B \subseteq V, A \cap B = \varnothing} x_{A \cup B} u^{rk((G \nabla B)[A \cup B])} v^{n((G \nabla B)[A \cup B])}$$

Note that the factor $x_{A \cup B}$ does not distinguish A and B. The sets B of toggled loops play a role, but they are not visible in monomials like y_B.

This polynomial has two specializations. First the polynomial Q of [1] defined by $Q(G, v') = \tau'(C_{x=y}(G))$ where τ' is the substitution:

$$[u := 1; v := v' - 2; x_a := y_a := 1 \text{ for all } a \in V]$$

so that:

$$Q(G, v') = \sum\nolimits_{A,B \subseteq V, A \cap B = \varnothing} (v'-2)^{n((G \nabla B)[A \cup B])}.$$

Another one is the *independence polynomial* (Levit and Mandrescu [23]), expressible by:

$$I(G, v) = \eta(C_{x=y}(G))$$

where η is the substitution $[u := 0; x_a := 1 \text{ for all } a \in V]$.

Proposition 20: (1) $C_{x=y}(G\nabla T) = C_{x=y}(G)$ for every graph G and set of vertices T.

(2) A graph G without loops and its polynomial $C(G)$ can be uniquely determined from $\rho(C_{x=y}(G))$, where ρ replaces v by 1.

Proof : (1) follows from Lemma 4.

(2) Consider two distinct vertices a and b. By looking at the ranks of the graphs obtained by adding loops to $G[\{a,b\}]$, we see that if $a \sim b$, then we have the monomials $x_a x_b u$ and $3x_a x_b u^2$ in $\rho(C_{x=y}(G))$. Otherwise, we have the monomials $x_a x_b$, $2x_a x_b u$ and $x_a x_b u^2$. □

Corollary 13 yields the following recursive definition:

Proposition 21: For every graph G:

(1) $C_{x=y}(\varnothing) = 1$

(2) $C_{x=y}(G) = (1 + x_a(u+v))C(G-a) \qquad$ if $N(G,a) = \varnothing$,

(3) $C_{x=y}(G) = x_a x_b u^2\{C_{x=y}(G^{ab}-a-b) + C_{x=y}((G^a)^b - a - b)\}$

$\quad +x_b u\{C_{x=y}(G^b - b) - C_{x=y}(G^b - a - b)\}$

$\quad +C_{x=y}(G-a) + C_{x=y}(G-b) - C_{x=y}(G-a-b) \qquad$ if $b \in N(G,a)$.

We wish to compare this definition with the one given in [1] for Q (and for graphs without loops). Proposition 21 yields the following reduction formulas:

(Q1) $Q(\varnothing) = 1$

(Q2) $Q(G) = u' \cdot Q(G-a) \qquad$ if $N(G,a) = \varnothing$,

(Q3) $Q(G) = Q(G^{ab}-a-b) + Q((G^a)^b - a - b)$

$\quad +Q(G^b - b) - Q(G^b - a - b)$

$\quad +Q(G-a) + Q(G-b) - Q(G-a-b) \qquad$ if $b \in N(G,a)$.

However, in the recursive definition of [1], Formula (Q3) is replaced by the following:

(Q3') $Q(G) = Q(G-b) + Q(G*b-b) + Q(G^{ab}-a) \qquad$ if $a \in N(G,b)$,

where $G*b := (G\nabla N(G,b))^b = G^b\nabla N(G,b)$.

We can prove the equivalence of the two recursive definitions. Proposition 15 yields for G such that $a \sim b$:

$Q(G^{ab}-a) =$

$Q(G-a) - Q(G-a-b) + Q(G^{ab}-a-b) - Q(G^b-a-b) + Q((G^a)^b - a - b)$,

so that (Q3) reduces to

$Q(G) = Q(G-b) + Q(G^b - b) + Q(G^{ab}-a)$.

It remains to check that $Q(G*b-b) = Q(G^b - b)$. From the definition of $G*b$ we have:

$Q(G*b-b) = Q(G^b\nabla N(G,b) - b)$

$= Q((G^b - b)\nabla N(G,b))$

$= Q(G^b - b)$

with the help of Proposition 20 (1), as was to be proved. Hence (Q3) is equivalent to (Q3'). Hence, we have reestablished the recursive definition of [1], but *not at the multivariate level* as this was the case for q in Corollary 19. In order to obtain it from that of Proposition 21, we had to take $u = 1$ and $x_a = 1$ for all a.

The advantage of the definition using (Q1), (Q2), (Q3') is that it only deals with loop-free graphs, whereas the definition of Proposition 21, even if used to compute $C_{x=y}(G)$ for G without loops uses the graphs with loops $(G^a)^b$ and G^b. It proves also that Q is positive, which is not obvious from the static definition.

4.3 The independence polynomial.

The *independence polynomial* is defined by

$$I(G,v) = \sum\nolimits_k s_k v^k$$

where s_k is the number of stable sets of cardinality k. (A looped vertex may belong to a stable set). Hence, we have:

$$I(G,v) = \eta(C_{x=y}(G))$$

where η is the substitution $[u := 0; x_a := 1 \text{ for all } a \in V]$.

We let $C_I(G) = \eta'(C(G))$ where η' is the substitution that replaces u by 0. It is a multivariate version of the independence polynomial, that can be defined directly by:

$$C_I(G) = \sum\nolimits_{\psi(A,B)} x_A y_B v^{n((G\nabla B)[A\cup B])}$$

where $\psi(A,B)$ is the set of conditions:

$$A \subseteq V - \mathit{Loops}(G),\ B \subseteq \mathit{Loops}(G), (G\nabla B)[A\cup B] \text{ has no edge},$$

so that $n((G\nabla B)[A\cup B]) =\mid A\cup B\mid$. From Corollary 13, we obtain the recursive definition

(I1) $C_I(\varnothing) = 1$
(I2) $C_I(G) = (1 + x_a v)C_I(G - a)$ if $N(G,a) = \varnothing$, a is not a loop,
(I3) $C_I(G) = (1 + y_a v)C_I(G - a)$ if $N(G,a) = \varnothing$, a is a loop,
(I4) $C_I(G) = C_I(G - a) + C_I(G - b) - C_I(G - a - b)$ if $b \in N(G,a)$.

However we can derive alternative reduction formulas:

Proposition 22: For every graph G:
(I1) $C_I(\varnothing) = 1$
(I5) $C_I(G) = C_I(G - a) + x_a v \cdot C_I(G - a - N(G,a))$, if a is not a loop,
(I6) $C_I(G) = C_I(G - a) + y_a v \cdot C_I(G - a - N(G,a))$, if a is a loop,
(I7) $C_I(G) = C_I(G - e) - x_a x_b v^2 \cdot C_I(G - N(G,a) - N(G,b))$,
if e is the edge linking a and b; (we do not delete a and b in $G - e$).

Proof : We omit the routine verifications, which use formulas (I1), (I2), (I3) and induction on size of graphs. □

Formulas (I5), I(6), (I7) are multivariate versions of reduction formulas given in Proposition 2.1 of the survey [23].

5 Computation of interlace and other monadic second order polynomials

We consider how one can *evaluate* for particular values of indeterminates, or *compute* (symbolically) in polynomial time the polynomials $q(G)$ and $Q(G)$ and the multivariate polynomial $C(G)$ defined and studied in the previous sections. We first recall the obvious fact that the recursive definitions like those of Proposition 12 do not yield polynomial time algorithms because they use a number of calls that is exponential in the number of vertices of the considered graphs. It is also known that the evaluation of many polynomials is difficult in general. For an example, Bläser and Hoffmann have proved that the evaluation of q is #P-hard in most cases [5].

We will describe a method that is based on expressing the static definitions of polynomials in *monadic second-order logic* and that gives polynomial time algorithms for classes of graphs of *bounded clique-width*. This method has been already presented in [11, 24, 25, 27], but we formulate it in a more precise way, and we extend it to multivariate polynomials and their *truncations*. The basic definitions and facts about clique-width and monadic second-order logic will be reviewed in Section 5.1 and 5.2 respectively. We first explain which types of computations we may hope to do in polynomial time for interlace polynomials.

We define the *size* $| P |$ of a polynomial P as the number of its monomials. Since monomials cannot be written in fixed size (with a bounded number of bits), this notion of size is a lower bound, not an accurate measure of the size in an actual implementation. It is clear that the multivariate polynomial $C_{y=0}(G)$ has exponential size in the number of vertices of G, and so have $C_{x=y}(G)$ and $C(G)$. Hence, we cannot hope for computing them in polynomial time.

We define the *quasi-degree* of a monomial as the number of vertices and/or of edges (multivariate Tutte polynomials used indeterminates indexed by edges) that index its indeterminates. If a vertex or an edge indexes r indeterminates, we count it for r. To take a representative example, a monomial of the form $n \cdot x_A y_B u^p v^q$ has quasi-degree $| A | + | B |$ (and not $| A \cup B |$). For every polynomial $P(G)$, we denote by $P(G) \upharpoonright d$ its *d-truncation* defined as the sum of its monomials of quasi-degree at most d.

For each d, the polynomials $C_{y=0}(G) \upharpoonright d$, $C_{x=y}(G) \upharpoonright d$ and $C(G) \upharpoonright d$ have sizes less than n^{2d}, where n is the number of vertices. Hence, asking for their computations in polynomial time has meaning. Since their monomials have integer coefficients bounded by n^{2d}, since they have at most d occurrences of G-indexed indeterminates, and their "ordinary" indeterminates u, v have exponents at most n, we can use the size of a polynomial for discussing the time complexity of the computation of its truncations in such cases.

For a specialization $P(G)$ of $C(G)$ where all G-indexed indeterminates are replaced by constants or by polynomials in ordinary indeterminates $x, y, \ldots$ we have $P(G) = P(G) \upharpoonright 0$. Hence, efficient algorithms for computing d-truncations will yield efficient algorithms for computing the classical (non multivariate) versions of these polynomials, and evaluating them for *arbitrary values* of their indeterminates, but only for particular classes of graphs likes those of bounded tree-width or clique-width. The definition of *clique-width* will be reviewed in the next section.

Theorem 23: For all integers k, d, for each polynomial P among C, $C_{y=0}$, $C_{x=y}$, C_I, its d-truncation can be computed in time $O(\mid V \mid^{3d+O(1)})$ for a graph G of tree-width or clique-width at most k. Polynomials $q(G), Q(G)$ can be computed in times respectively $O(\mid V \mid^7)$ and $O(\mid V \mid^4)$ for graphs of tree-width or clique-width at most k.

This theorem gives for each d a *fixed parameter tractable algorithm* where clique-width (but not d) is the parameter. The theory of fixed parameter tractability is presented in the books [14] and [19]. As a corollary one obtains the result by Ellis-Monaghan and Sarmiento [15] that the polynomial q is computable in polynomial time for *distance-hereditary graphs*, because these graphs have clique-width at most 3, as proved by Golumbic and Rotics [18].

This theorem will be proved by means of expressions of the considered polynomials by formulas of monadic second-order logic. The proof actually applies to all multivariate polynomials expressible in a similar way in monadic second-order logic. Hence we will present in some detail the logical expression of graph polynomials.

5.1 Clique-width

Clique-width is, like tree-width, a graph complexity measure based on hierarchical decompositions of graphs. These decompositions make it possible to build efficient algorithms for hard problems restricted to graphs of bounded clique-width or tree-width (see [9,14,19] for tree-width). In many cases, one obtains *Fixed-Parameter Tractable* algorithms, that is, algorithms with time complexity $f(k).n^c$ where f is a fixed function, c is a fixed constant, k is the clique-width or the tree-width of the considered graph, and n is its number of vertices. Tree-width or clique-width is here the *parameter*.

Let $C = \{1, \ldots, k\}$ to be used as a set of labels. A *k-graph* is a graph G given with a total mapping from its vertices to C, denoted by lab_G. We call $lab_G(x)$ the *label* of a vertex x. Every graph is a k-graph, with all vertices labeled by 1. For expressing its properties by logical formulas we will handle a k-graph as a tuple $(V, M, p_1, \ldots, p_k)$ where the adjacency matrix M is treated as a binary relation ($M(x, y)$ is true if $M(x, y) = 1$, and false if $M(x, y) = 0$) and $p_1, \ldots, p_k$ are unary relations such that $p_j(x)$ is true if and only if $lab_G(x) = j$.

The operations on k-graphs are the following ones:

(i) For each $i \in C$, we define constants $\mathbf{i}$ and $\mathbf{i}^\ell$ for denoting isolated vertices labeled by i, the second one with a loop.

(ii) For $i, j \in C$ with $i \neq j$, we define a unary function $add_{i,j}$ such that:

$$add_{i,j}(V, M, lab) = (V, M', lab)$$

where $M'(x, y) = 1$ if $lab(x) = i$ and $lab(y) = j$ or vice-versa (we want M' to be symmetric), and $M'(x, y) = M(x, y)$ otherwise. This operation adds undirected edges between any vertex labeled by i and any vertex labeled by j, whenever these edges are not already in place.

(iii) We let also $ren_{i\to j}$ be the unary function such that

$$ren_{i\to j}(V, M, lab) = (V, M, lab')$$

where $lab'(x) = j$ if $lab(x) = i$ and $lab'(x) = lab(x)$ otherwise. This mapping relabels by j every vertex labeled by i.

(iv) Finally, we use the binary operation $\oplus$ that makes the union of disjoint copies of its arguments. (Hence $G \oplus G \neq G$ and the number of vertices of $G \oplus G$ is twice that of G.)

A well-formed expression t over these symbols will be called a *k-expression*. Its *value* is a k-graph $G = val(t)$. The set of vertices of $val(t)$ is (or can be defined as) the set of occurrences of the constants (the symbols $\mathbf{i}$ and $\mathbf{i}^\ell$) in t. However, we will also consider that an expression t designates any graph isomorphic to $val(t)$. The context specifies whether we consider concrete graphs or graphs up to isomorphism.

For an example, a path with 5 vertices with a loop at one end is the value of the 3-expression:

$$ren_{2\to 1}(ren_{3\to 1}[add_{1,3}(add_{1,2}(\mathbf{1} \oplus \mathbf{2}) \oplus add_{1,2}(\mathbf{1} \oplus \mathbf{2}^\ell) \oplus \mathbf{3})]).$$

The *clique-width* of a graph G, denoted by *cwd(G),* is the minimal k such that $G = val(t)$ for some k-expression t. It is clear that clique-width does not depend on loops: $cwd(G\nabla T) = cwd(G)$ for every set of vertices T.

A graph with at least one edge has clique-width at least 2. The complete graphs K_n have clique-width 2 for $n \geq 2$. Trees have clique-width at most 3. Planar graphs, and in particular square grids, have unbounded clique-width.

If a class of graphs has bounded tree-width (see [9,14,19]), it has bounded clique-width but not vice-versa. It follows that our results, formulated for graph classes of bounded clique-width also hold for classes of bounded tree-width.

The problem of determining if a graph G has clique-width at most k is NP-complete if k is part of the input (Fellows *et al.* [17]). However, for each k, there is a cubic algorithm that reports that a graph has clique-width $> k$ or produces an $f(k)$-expression for some fixed function f. Several algorithms have been given by Oum in [28] and by Hliněný and Oum in [21]; the best known function f is $f(k) = 2^{k+1} - 1$ (by the result of [21]). The method for construction fixed parameter algorithms based on clique-width and monadic second-order logic is exposed in [11, 24], and will be reviewed in Section 5.4 below.

The following extension of the definition k-expression will be useful. An *ordered k-graph* G is a k-graph equipped with a linear order $\leq_G$ on V. On ordered k-graphs, we will use the variant $\overrightarrow{\oplus}$ of $\oplus$ defined as follows:

(iv) $G \overrightarrow{\oplus} H$ is the disjoint union of G and H with a linear order that extends those of G and H and makes the vertices of G smaller than those of H.

The other operations are defined in the same way. This extension will be used as follows: a graph G being given by a k-expression t, we replace everywhere in t the operation $\oplus$ by $\overrightarrow{\oplus}$. The obtained expression $\overrightarrow{t}$ defines G together with a linear ordering on its vertices.

5.2 Monadic second-order logic and polynomial-time computations

Our proof of Theorem 23 will make an essential use of the expression of the considered polynomials in *monadic second-order logic.* We review the basic definitions and examples. For a systematic exposition, the reader is refered to [9]. In a few words monadic second-order logic is the extension of *first-order logic* using variables denoting sets of objects, hence in the case of graphs, sets of vertices, and in some cases sets of edges (but we will not use this feature in this article).

Formulas are written with special (uppercase) variables denoting subsets of the domains of the considered relational structures in the general case, and sets of vertices in this article. Formulas use atomic formulas of the form $x \in X$ expressing the membership of x in a set X. In order to write more readable formulas, we will also use their negations $x \notin X$. The syntax also allows atomic formulas of the form $Card_{p,q}(X)$ expressing that the set designated by X has cardinality equal to p modulo q, where $0 \leq p < q$ and q is at least 2. We will only need in this article the atomic formula $Card_{0,2}(X)$ also denoted by $Even(X)$. (All interpretation domains are finite, hence these cardinality predicates are well-defined). Rather than a formal syntax, we give several significant examples.

An ordered k-graph is handled as a relational structure $(V, M, \leq, p_1, \ldots, p_k)$. For a k-graph, we simply omit $\leq$. Set variables denote sets of vertices. Here are some examples of graph properties expressed in monadic second-order logic. That a graph G is *3-vertex colorable* (with neighbour vertices of different colors) can be expressed as $G \models \gamma$, read "γ is true in the structure (V, M) representing G" (here $\leq, p_1, \ldots, p_k$ do not matter) where γ is the formula:

$$\begin{aligned}&\exists X_1, X_2, X_3 \cdot [\forall x(x \in X_1 \vee x \in X_2 \vee x \in X_3) \wedge\\ &\qquad \forall x(\neg(x \in X_1 \wedge x \in X_2) \wedge \neg(x \in X_2 \wedge x \in X_3) \wedge \neg(x \in X_1 \wedge x \in X_3))\\ &\quad \wedge \forall u, v(M(u,v) \wedge u \neq v \Longrightarrow \neg(u \in X_1 \wedge v \in X_1) \wedge \neg(u \in X_2 \wedge v \in X_2)\\ &\quad \wedge \neg(u \in X_3 \wedge v \in X_3))].\end{aligned}$$

That $G[B]$ (where $B \subseteq V$) is *not connected* can be expressed by the formula $\delta(X)$, with free variable X:

$$\begin{aligned}&\exists Y \cdot [\exists x \cdot (x \in X \wedge x \in Y) \wedge \exists y \cdot (y \in X \wedge y \notin Y) \wedge\\ &\qquad\qquad \forall x, y \cdot (x \in X \wedge y \in X \wedge M(x,y)\\ &\qquad\qquad\qquad \Longrightarrow \{(x \in Y \wedge y \in Y) \vee (x \notin Y \wedge y \notin Y)\})].\end{aligned}$$

For B a subset of V, $(G, B) \models \delta(X)$, read "δ is true in the structure representing G with B as value of X", if and only if $G[B]$ is not connected.

The formula γ has no free variables, hence it expresses a property of the considered graph. The formula δ with free variable X expresses a property of sets of vertices in the considered graphs, "given as input" to δ as values of X. We now give an example of a property of a pair of sets of vertices, expressed by a formula with two free set variables.

The vertex set V of an ordered graph is linearly ordered, with a strict order denoted by $<$. Hence sets of vertices can be compared lexicographically. For subsets U and W of V we let $U \leq_{lex} W$ if and only if

either $U = W$, or $U \subseteq W$ and every element of $W - U$ is larger than every element of U or $U - W \neq \emptyset$ and the smallest element of $(W - U) \cup (U - W)$ is in U.

This can be expressed by the validity of $\sigma(U, W)$ where $\sigma(X, Y)$ is the formula:

$\forall x \cdot ((x \in X \Longrightarrow x \in Y) \wedge \forall y \cdot [(y \in Y \wedge y \notin X) \Longrightarrow \forall u \cdot (u \in X \Longrightarrow u < y)])$

$\vee(\exists x \cdot (x \in X \wedge x \notin Y) \wedge \forall y \cdot [(y \in Y \wedge y \notin X) \Longrightarrow \exists u \cdot (u \in X \wedge u \notin Y \wedge u < y)])$.

The following lemma will be useful for the monadic second-order expression of interlace polynomials.

Lemma 24 [13]: There exists a monadic second-order formula $\rho(X, Y)$ expressing that, in a graph $G = (V, M)$ we have $Y \subseteq X$ and the row vectors of $M[X, X]$ associated with Y form a basis of the vector space spanned by the row vectors of the matrix $M[X, X]$. Hence, for each set X, all sets Y satisfying $\rho(X, Y)$ have the same cardinality equal to $rk(G[X])$.

Proof: We first build a basic formula $\lambda(Z, X)$ expressing that $Z \subseteq X$ and that the row vectors of $M[X, X]$ associated with Z are linearly dependent over GF(2).

Condition $Z \subseteq X$ is expressed by $\forall y \cdot (y \in Z \Longrightarrow y \in X)$. (We will then use $\subseteq$ in formulas, although this relation symbol does not belong to the basic syntax).

The second condition is equivalent to the fact that for each $u \in X$, the number of vertices $z \in Z$ such that $M(z, u) = 1$ is even. This fact is written:

$$\forall u \cdot (u \in X \Longrightarrow \exists W \cdot [Even(W) \wedge \forall z \cdot (z \in W \Longleftrightarrow z \in Z \wedge M(z, u))]).$$

With $\lambda(Z, X)$ one expresses that Y (such that $Y \subseteq X$) forms a basis by:

$$\neg\lambda(Y, X) \wedge \forall Z \cdot (\{Y \subseteq Z \wedge Z \subseteq X \wedge \neg(Z \subseteq Y)\} \Longrightarrow \lambda(Z, X)).$$

We get thus the formula $\rho(X, Y)$. $\square$

We will say that the rank function is *definable by a monadic second-order (MS) formula*. In general we say that a function f associating a nonnegative integer $f(A, B, C)$ with every triple of sets (A, B, C) is *defined by an MS formula* $\psi(X, Y, Z, U)$ if, for every (A, B, C) the number $f(A, B, C)$ is the common cardinality of all sets D such that $(G, A, B, C, D) \models \psi(X, Y, Z, U)$. (We distinguish the variables X, Y, Z, U from the sets A, B, C, D they can denote). The generalization to functions f with k arguments is clear, and the defining formula has then $k + 1$ free variables.

5.3 Multivariate polynomials defined by MS formulas and substitutions

For an MS formula φ with free variables among $X_1, \dots, X_m$, for a graph G, we let:

$$\begin{aligned}\mathbf{sat}(G, \varphi, X_1, \dots, X_m) = &\{(A_1, \dots, A_m) \mid A_1, \dots, A_m \subseteq V, \\ &(G, A_1, \dots, A_m) \models \varphi(X_1, \dots, X_m)\}.\end{aligned}$$

This is the set of all m-tuples of sets of vertices that satisfy φ in G. The condition $(G, A_1, \dots, A_m) \models \varphi(X_1, \dots, X_m)$ will be written in a shorter way as $\varphi(A_1, \dots, A_m)$. We can write the set $\mathbf{sat}(G, \varphi, X_1, \dots, X_m)$ in the form of a multivariate polynomial:

$$P_\varphi(G) = \sum\nolimits_{\varphi(A_1,\dots,A_m)} x^{(1)}_{A_1} \dots x^{(m)}_{A_m}.$$

It is clear that P_φ describes exactly $\mathbf{sat}(G, \varphi, X_1, \dots, X_m)$ and nothing else. Its set of indeterminates is $\mathbf{W}_G$ where $W = \{x^{(1)}, \dots, x^{(m)}\}$. Such a polynomial is called a *basic MS polynomial*, and m is its *order*. Neither the polynomial $Z(G)$ recalled in the introduction nor the interlace polynomial $C(G)$ is a basic MS-polynomial. Before giving general definitions, we show how multivariate polynomials can be expressed as specializations of basic MS-polynomials. To avoid heavy formal definitions, we consider a typical example:

$$P(G) = \sum\nolimits_{\varphi(A,B,C)} x_A y_B u^{f(A,B,C)} \tag{6}$$

where $\varphi(X, Y, Z)$ is an MS formula and f is a function on triples of sets defined by an MS formula $\psi(X, Y, Z, U)$. (This necessitates that for each triple X, Y, Z all sets U satisfying $\psi(X, Y, Z, U)$ have the same cardinality). After usual summation of *similar* monomials (those with same indeterminates with same exponents) the general monomial of $P(G)$ is of the form $c \cdot x_A y_B u^p$ where c is the number of sets C such that $f(A, B, C) = p$. We first observe that $P(G) = \sigma(P'(G))$ where:

$$P'(G) = \sum\nolimits_{\varphi(A,B,C)} x_A y_B z_C u^{f(A,B,C)}$$

where σ replaces each z_c by 1. We are looking for an expression of $P(G)$ as $\mu(\sigma(P_\theta(G)))$ $= \mu \circ \sigma(P_\theta(G))$ where μ replaces each u_d by u in:

$$P_\theta(G) = \sum\nolimits_{\theta(A,B,C,D)} x_A y_B z_C u_D$$

for some formula $\theta(X, Y, Z, U)$. Taking $\theta(X, Y, Z, U)$ to be $\varphi(X, Y, Z) \wedge \psi(X, Y, Z, U)$ would be incorrect in cases where several sets D satisfy $\psi(A, B, C, D)$ for a triple (A, B, C) satisfying φ. We overcome this difficulty in the following way: we let V be linearly ordered in an arbitrary way; we let ψ' be the formula, written with a new binary relation symbol $\leq$ denoting the ordering of V, such that $\psi'(X, Y, Z, U)$ is equivalent to:

$$\psi(X, Y, Z, U) \wedge \forall T \cdot [\psi(X, Y, Z, T) \Longrightarrow \text{“}U \leq_{lex} T\text{”}]$$

where $U \leq_{lex} T$ means that U is less than or equal to T in the lexicographic order derived from the ordering of V. This is expressible by an MS formula as we have seen in Section 5.2. The formula $\psi'(X, Y, Z, U)$ defines the function f by selecting a unique set D such that $f(A, B, C) = \mid D \mid$. This set is unique for each linear order on V by the hypothesis on ψ, but its cardinality does not depend on the chosen linear order. Hence we have the desired expression of P:

$$P(G) = (\mu \circ \sigma)(\sum_{\varphi(A,B,C) \wedge \psi'(A,B,C,D)} x_A y_B z_C u_D).$$

This expression uses an ordering of the given graph, but $P(G)$ does not depend on the chosen order. The substitution $\mu \circ \sigma$ replaces replaces each u_d by u and each z_c by 1. These remarks motivate the following definition:

Definition 25: *Monadic second-order (MS-) polynomials*
An *MS-polynomial* is a polynomial of the form:

$$P(G) = \sum_{\varphi(A_1,\ldots,A_m)} x_{A_1}^{(1)} \ldots x_{A_{m'}}^{(m')} u_1^{f_1(A_1,\ldots,A_m)} \ldots u_p^{f_p(A_1,\ldots,A_m)} \tag{7}$$

where $\varphi(X_1, \ldots, X_m)$ is an MS formula, $m' \leq m$ and $f_1, \ldots, f_p$ are monadic second-order definable functions. An *MS-polynomial* in *normal form* is a polynomial of the form $P = \sigma \circ P_\varphi$ where:

$$P_\varphi(G) := \sum_{\varphi(A_1,\ldots,A_m)} x_{A_1}^{(1)} \ldots x_{A_m}^{(m)} \tag{8}$$

for an MS formula φ and a substitution σ that can replace a variable x_a by 1 or by an ordinary variable say u. From the above observations, it is clear that every MS-polynomial can be written in normal form, for graphs arbitrarily ordered. The expression of P is said to be *order-invariant*: we mean by this that even if the logical expression of P uses a linear order on the set of vertices, in particular for handling monadic second-order definable functions, then for any two linear orders, the defined polynomials are the same.

A *generalized MS-polynomial* is a polynomial defined as $P = \sigma \circ P_\varphi$ where P_φ is as in (8) for an MS-formula φ and a specializing substitution σ. A generalized MS-polynomial may have negative coefficients, and in this case, cannot have a normal form. We say that P is of *order* m if it can be expressed as $P = \sigma \circ P_\varphi$ where φ has m free variables. Hence a polynomial of the form (7) above is of order at most $m + p$. Then for a graph G with n vertices and P defined by (7), $P(G)$ has size at most $2^{m'n}$, degree at most $n(m' + p)$, positive coefficients of value at most 2^{mn}. These bounds will be useful for evaluating the cost of computations of truncations of polynomials.

Question : Is it true that every positive generalized MS-polynomial (i.e., that is positive for every graph) has an expression in normal form ?

Transformations of MS-polynomials.

In order to show the robustness of the definition, we make precise how some specializations can be reflected by transformations of the defining formulas. We review some cases which arise in the present article, by taking a polynomial P of the form (6) with associated formulas φ, ψ. For each case 1 to 4 we denote by P_i the polynomial obtained from P.

Case 1 : $x_a := 0$. We have:
$P_1(G) = \sum_{\varphi'(B,C)} y_B u^{f(\varnothing,B,C)}$
where $\varphi'(Y,Z)$ is defined as $\exists X \cdot [\varphi(X,Y,Z) \wedge \forall z \cdot z \notin X]$ so that $\varphi'(B,C)$ is equivalent to $\varphi(\varnothing,B,C)$. The function $f(\varnothing,Y,Z)$ is defined by $\exists X \cdot [\psi(X,Y,Z,U) \wedge \forall z \cdot z \notin X]$.

Case 2: $u := 0$. We have
$P_2(G) = \sum_{\varphi'(A,B,C)} x_A y_B$
where $\varphi'(X,Y,Z)$ is $\varphi(X,Y,Z) \wedge \exists U \cdot [\psi(X,Y,Z,U) \wedge \forall z \cdot z \notin U]$, which is equivalent to $\varphi(X,Y,Z) \wedge$ "$f(X,Y,Z) = 0$". The assumption that f is defined by ψ implies that if $\psi(X,Y,Z,\varnothing)$ is true, then no nonempty set U satisfies $\psi(X,Y,Z,U)$.

Case 3: $x_a := 1$. We have
$P_3(G) = \sum_{\varphi(A,B,C)} y_B u^{f(A,B,C)}$.
The sets A are, like the sets C, "invisible" in the monomials. However, they play a role. They contribute to the multiplicity of the monomials $y_B u^p$, for $p = f(A,B,C)$. This case has been considered above at the beginning of Section 5.3. No transformation of formulas has to be done in this case.

Case 4: $x_a := y_a$. Here
$P_4(G) = \sum_{\varphi(A,B,C)} x_{A \cup B} u^{f(A,B,C)}$
We assume here, and this is enough for the cases considered in this article, that the condition $\varphi(A,B,C)$ implies that A and B are disjoint. Then to reach the general syntax we write $P_4(G)$ as follows
$P_4(G) = \sum_{\varphi'(D,B,C)} x_D u^{g(D,B,C)}$
where D stands for $A \cup B$, $\varphi'(W,Y,Z)$ is chosen to be equivalent to the condition: $Y \subseteq W \wedge \varphi(W - Y, Y, Z)$ and the function g is defined by:
$g(D,B,C) = f(D - B, B, C)$
whence also by a formula $\psi'(W,Y,Z,U)$ equivalent to:
$\exists X \cdot [\psi(X,Y,Z,U) \wedge X \subseteq W \wedge$ "$Y = W - X$"$]$.

Lemma 26: Each polynomial P among C, $C_{y=0}$, $C_{x=y}$, C_I is a generalized MS-polynomial. It can be expressed as $P = \sigma \circ P_\theta$ for a monadic second-order formula θ expressing properties of ordered graphs, and a specializing substitution σ, so that $P(G) = \sigma \circ P_\theta(G)$ for every graph G, ordered in an arbitrary way. We have $P(G) = \sigma(P_\theta(G))$ for every linear order on G.

Proof: We only consider $C(G)$. The other cases follow by the techniques presented above. We recall the definition:

$$C(G) = \sum_{A,B \subseteq V, A \cap B = \varnothing} x_A y_B u^{rk((G \nabla B)[A \cup B])} v^{n((G \nabla B)[A \cup B])}$$

We let

$$P_\theta(G) = \sum\nolimits_{\theta(A,B,C,D)} x_A y_B u_C v_D$$

where $\theta(A,B,C,D)$ holds if and only if $A \cap B = \varnothing, C \subseteq A \cup B, D = A \cup B - C$, C is the smallest basis of the vector space spanned by the row vectors of the matrix $M_{G\nabla B}[A \cup B, A \cup B]$ (where $M_{G\nabla B}$ is the adjacency matrix of $G\nabla B$). By "smallest" we mean with respect to the lexicographic ordering derived from the ordering of G. It follows that $\mid C \mid = rk((G\nabla B)[A \cup B])$ and $\mid D \mid = n((G\nabla B)[A \cup B])$. By Lemma 24, one can express these conditions by an MS formula θ. Hence, $C(G) = (\sigma \circ P_\theta)(G)$ where σ replace each u_a by u and each v_a by v. $\square$

5.4 The Fefermann-Vaught paradigm applied to monadic second order polynomials

In order to prove Theorem 23, we will show how multivariate polynomials defined by MS formulas can be computed by induction on k-expressions defining graphs of clique-width bounded by a fixed value k (Definitions are in Section 5.1). We will use the *Fefermann-Vaught paradigm*, presented in detail by Makowsky [24] and used in [25] to compute the Tutte polynomial of graphs of bounded tree-width.

We need operations that manipulate sets of q-tuples, in particular, those of the form $\mathbf{sat}(G, \varphi, X_1, \ldots, X_q)$, and, equivalently as we will see, the polynomials $P_\varphi(G)$. For sets R, S and $S' \subseteq \mathcal{P}(V)^q$, we write:

$R = S \uplus S'$ if $R = S \cup S'$ and $S \cap S' = \varnothing$, and

$R = S \boxtimes S'$ if $S \subseteq \mathcal{P}(V_1)^q$, $S' \subseteq \mathcal{P}(V_2)^q$, $V_1 \cap V_2 = \varnothing$, and R is the set of q-tuples $(A_1 \cup B_1, \ldots, A_q \cup B_q)$ such that $(A_1, \ldots, A_q) \in S$ and $(B_1, \ldots, B_q) \in S'$.

For each $S \subseteq \mathcal{P}(V)^q$, we let $P(S)$ be the multivariate polynomial:

$$P(S) = \sum\nolimits_{(A_1,\ldots,A_q)\in S} x^{(1)}_{A_1} \ldots x^{(q)}_{A_q}.$$

Hence, $P_\varphi(G) = P(\mathbf{sat}(G, \varphi, X_1, \ldots, X_q))$. The following is clear:

Lemma 27: For $S, S' \subseteq \mathcal{P}(V)^q$, we have $P(S \uplus S') = P(S) + P(S')$ and $P(S \boxtimes S') = P(S) \cdot P(S')$.

We denote by $\mathbf{U}_k$ the (finite) set of unary operations allowed in k-expressions. We denote by $MS(k,q)$ the set of MS formulas written with the basic symbols $=, \neq, \in, \notin$, $Card_{r',r}$ and the relation symbols $M, \leq, p_1, \ldots, p_k$ (hence able to express properties of ordered k-graphs) with free variables in the set $\{X_1, \ldots, X_q\}$.

One could hope to define $\mathbf{sat}(G \overrightarrow{\oplus} H, \varphi, X_1, \ldots, X_q)$ in terms of $\mathbf{sat}(G, \varphi, X_1, \ldots, X_q)$ and $\mathbf{sat}(H, \varphi, X_1, \ldots, X_q)$ but this not possible in general. Auxiliary sets $\mathbf{sat}(G, \theta_i, X_1, \ldots, X_q)$ and $\mathbf{sat}(H, \psi_i, X_1, \ldots, X_q)$ are needed, but only for finitely many auxiliary formulas

θ_i and ψ_i. This is the key of the construction of linear algorithms from monadic second-order formulas. The following theorem discussed in [24] is proved in [9,12] in closely related forms:

Theorem 28: For every k, q, for every formula ξ in $MS(k,q)$, there exists a finite subset Φ of $MS(k,q)$ containing ξ and satisfying the following properties:

(1) For every $\varphi \in \Phi$ for every $op \in \mathbf{U}_k$ there exists a formula $\varphi^{op} \in \Phi$ such that, for every ordered k-graph G:

$$\mathbf{sat}(op(G), \varphi, X_1, \ldots, X_q) = \mathbf{sat}(G, \varphi^{op}, X_1, \ldots, X_q).$$

(2) For every $\varphi \in \Phi$ there exist p and $(\theta_1, \ldots, \theta_p, \psi_1, \ldots, \psi_p) \in \Phi^{2p}$ such that for disjoint ordered k-graphs G and H:

$$\mathbf{sat}(G \overrightarrow{\oplus} H, \varphi, X_1, \ldots, X_q) = \\ \biguplus_{1 \leq i \leq p} \mathbf{sat}(G, \theta_i, X_1, \ldots, X_q) \boxtimes \mathbf{sat}(H, \psi_i, X_1, \ldots, X_q).$$

These statements also hold for (unordered) k-graphs and the operation $\oplus$ instead of $\overrightarrow{\oplus}$.

The existence of fixed parameter tractable algorithms for checking monadic second-order properties and for listing the sets $\mathbf{sat}(G, \varphi, X_1, \ldots, X_q)$ for graphs of bounded clique-width G and monadic second-order formulas φ (see Flum *et al.* [18] for a detailed presentation of this result for graphs of bounded tree-width) is based on this result. Basically, this theorem says that the computations can be done by induction on the structure of k-expressions defining the input graphs. Since the set Φ is finite for fixed k and ξ (although it is very large), the computations can be done in time linear in the sizes of the given k-expressions (with very large constants).

Let Φ be a set of formulas as in Theorem 28. We get a finite family of polynomials $(P_\varphi)_{\varphi \in \Phi}$ that satisfy mutually recursive computations rules. Actually, the recursive rules apply to the family of polynomials $(\sigma \circ P_\varphi)_{\varphi \in \Phi}$ where σ is a specializing substitution.We recall that by a polynomial we mean (ambigously) a mapping P associating with a graph G a polynomial in $\mathbb{Z}[U \cup \mathbf{W}_G]$.

Corollary 29: Let Φ satisfy the properties of Theorem 28. Let σ be a specializing substitution. We have the following computation rules:

(1) For every $\varphi \in \Phi$, for every $op \in \mathbf{U}_k$, for every ordered k-graph G:

$$(\sigma \circ P_\varphi)(op(G)) = (\sigma \circ P_{\varphi^{op}})(G).$$

(2) For every $\varphi \in \Phi$ for every two disjoint ordered k-graphs G and H:

$$(\sigma \circ P_\varphi)(G \overrightarrow{\oplus} H) = \sum_{1 \leq i \leq p} (\sigma \circ P_{\theta_i})(G) \cdot (\sigma \circ P_{\psi_i})(H).$$

where φ^{op} and $(\theta_1, \ldots, \theta_p, \psi_1, \ldots, \psi_p)$ are as in Theorem 28.

Proof: If σ is the identity substitution, then (1) and (2) are direct translations of (1) and (2) of Theorem 28.

Since $\sigma \circ (P + Q) = \sigma \circ P + \sigma \circ Q$ i.e., $(\sigma \circ P + \sigma \circ Q)(G) = \sigma(P(G)) + \sigma(Q(G)) = \sigma((P+Q)(G))$ and similarly, $\sigma \circ (P \cdot Q) = (\sigma \circ P) \cdot (\sigma \circ Q)$, for every substitution σ and every two polynomials P, Q, the equalities extend to the general case as stated. □

Hence, this corollary concerns all multivariate polynomials described in Section 5.3. We will use it for computing their truncations.

5.5 Computing monadic second-order polynomials in polynomial time

We discuss the computation of polynomials written in the form $\sigma \circ P_\xi$ for a basic MS-polynomial P_ξ and a substitution σ. This will also apply to *evaluations* of polynomials where all indeterminates are given some numeric value, either integer, real or in some other field. Such an assignment of values can formalized by the application of a substitution.

Let a graph G be given by a k-expression t. We "order" t into $\overrightarrow{t}$ which defines G with a linear order of its vertices (cf. Section 5.1). This will be useful in cases where we have to select a unique lexicographically minimal set satisfying a condition. It is clear that for each constant, $\mathbf{i}$ or $\mathbf{i}^\ell$, each polynomial $(\sigma \circ P_\varphi)(\mathbf{i}^\ell)$ can be computed by using the definitions. By Corollary 29, we can thus compute for each subterm s of $\overrightarrow{t}$ the family of polynomials $((\sigma \circ P_\varphi)(val(s)))_{\varphi \in \Phi}$. In particular, at the end of the computation, one gets the Φ-tuple $((\sigma \circ P_\varphi)(G))_{\varphi \in \Phi}$. It clearly best to have sets Φ of "small cardinality". A method for restricting such computations to their useful parts is described in [12] or in [24], Definition 4.17.

This computation uses at most $n\cdot \mid \Phi \mid$ times the computation rules of Corollary 29 (2) (n will always be the number of vertices of the considered graph G), because in a term, the number of occurrences of $\overrightarrow{\oplus}$ is $s-1$, where s is the number of occurrences of constants, which is equal to $\mid V \mid = n$. Here, $\mid \Phi \mid$ is constant. The computation time is bounded by $2n \cdot c_G \cdot p_{max} \cdot \mid \Phi \mid$, where p_{max} is the maximum value of p in the rules of Theorem 28 (2), and c_G bounds the cost of the addition and of the multiplication of two polynomials. This bound depends on G.

When do we obtain a polynomial algorithm in n?

We first make precise the way we will count the cost of operations on polynomials. Using *unit cost measure*, we will count for one the cost of each basic operation on numbers : comparison, addition, substraction, multiplication. The cost of evaluating x^m for a positive integer m is thus $O(\log(m))$ if we use iterated squarings: $x, x^2, x^4, \ldots$ and multiplications of intermediate results. In the computation of a truncated polynomial $P \upharpoonright d$, we will consider d as "small" and fixed, like $\mid \Phi \mid$ (and actually much smaller in potential applications.) Hence we will count for one the cost of computing x^m for $m \leq d$.

However, for measuring computations and evaluations of polynomials, we could also use the *real cost measure* and consider that the cost of a comparison, an addition, a substraction and a multiplication of two positive integers x and y is $O(\log(x+y))$. Coefficients of polynomials may be exponential in the sizes of the considered graphs. However, in situations where the absolute values of coefficients and exponents are no larger than $2^{p(n)}$ for fixed polynomials p, a polynomial bound on computation time with respect to unit cost measure remains polynomial with respect to real cost measure. The exponents of the polynomial bounds are just larger.

We will base the following estimations of the cost of computations on straightforward data structures: a polynomial is a uniquely defined list of monomials sorted by increasing order of quasi-degree, where two monomials of same quasi-degree are ordered lexicographically. Each monomial is written in a canonical way by means of a fixed ordering of indeterminates. We deal with monomials with a variable number of indeterminates, however, this number is always bounded by $n\cdot \mid X \mid + \mid U \mid$ where the G-indexed indeterminates are defined from X, and U the set of ordinary indeterminates.

The basic operations on pairs of monomials m, m' are comparison, summation of coefficients if m, m' are *similar* monomials (if they have same indeterminates with same respective degrees), and multiplication. For a monomial m, we denote by $\mathbf{v}(m)$ the number of its indeterminates. The costs of these operations are respectively $O(\mathbf{v}(m)+\mathbf{v}(m')), 1$ and $O(\mathbf{v}(m)+\mathbf{v}(m'))$.The cost of the multiplication of two monomials is not 1 because one must sort the indeterminates according to the chosen order. We denote by $\mathbf{v}(P)$ the number of indeterminates in a polynomial P.

Lemma 30 : For every P, Q, d, if $\mathbf{v}(P), \mathbf{v}(Q) \leq \mathbf{v}_{\max}$ we have:
(1) $(P+Q) \upharpoonright d = P \upharpoonright d + Q \upharpoonright d$ and $(PQ) \upharpoonright d = ((P \upharpoonright d) \cdot (Q \upharpoonright d)) \upharpoonright d$.
(2) Computing $P+Q$ takes time $O(\mathbf{v}_{\max} \cdot (\mid P \mid + \mid Q \mid))$.
(3) Computing PQ takes time: $O(\mathbf{v}_{\max}\cdot \mid P \mid^2 \cdot \mid Q \mid)$ if $\mid P \mid \leq \mid Q \mid$.

Proof: (1) Clear from the definitions.

(2) Note that $\mid P+Q \mid \leq \mid P \mid + \mid Q \mid$. The addition of P and Q is done by interleaving their lists of monomials and by adding the coefficients of similar monomials. This gives the result by the remaks on the cost of operations on monomials.

(3) Let $\mid P \mid \leq \mid Q \mid$ and $\mathbf{v}(P), \mathbf{v}(Q) \leq \mathbf{v}_{\max}$. Note that $\mid PQ \mid \leq \mid P \mid \cdot \mid Q \mid$. We compute PQ by multiplying $\mid P \mid$ times the polynomial Q by a monomial of P, and by performing $\mid P \mid -1$ additions of polynomials of size at most $\mid P \mid \cdot \mid Q \mid$. The time taken is at most:

$$O(\mathbf{v}_{\max}\cdot \mid P \mid \cdot \mid Q \mid + (\mid P \mid -1) \cdot \mathbf{v}_{\max}\cdot \mid P \mid \cdot \mid Q \mid) = O(\mathbf{v}_{\max}\cdot \mid P \mid^2 \cdot \mid Q \mid). \square$$

If in (3) P and Q are positive, one gets the bound $O(\mathbf{v}_{\max}\cdot \mid P \mid \cdot \mid PQ \mid)$ because all intermediate results have size at most $\mid PQ \mid$.

Theorem 31: Let k, d be fixed integers. The d-truncation of a generalized MS-polynomial $P(G)$ can be computed in time $O(n^{6d+O(1)})$ for every graph G of clique-width at most k.

The constants hidden by the O-notation depend on P and k. The particular case of evaluations (for numerical values of indeterminates) will be discussed later. Closely related formulations of this theorem are in [11, 24, 25]. For a polynomial P, we define $\|P\| :=$

$(\mid P \mid, \deg(P), C_{\max}(P))$ where $\deg(P)$ is the degree of P and $C_{\max}(P)$ the maximum absolute value of its coefficients. Triples of integers are ordered componentwise.

Lemma 32: Let P be a polynomial and σ be a substitution such that $\|\sigma(x)\| \leq (\mathbf{s}_{max}, \mathbf{d}_{max}, \mathbf{c}_{max})$ for every indeterminate x. The polynomial $\sigma \circ P$ satisfies:

$$\|\sigma \circ P\| \leq (\mid P \mid \cdot (\mathbf{s}_{max})^{\deg(P)}, \deg(P) \cdot \mathbf{d}_{max}, \mid P \mid \cdot C_{\max}(P) \cdot (\mathbf{s}_{max}\mathbf{c}_{max})^{\deg(P)}).$$

Proof: Easy verification. □

It follows from observations made in Definition 25 that if P_φ has order m, then for every graph G, the polynomial $\sigma(P_\varphi(G))$ has size and coefficients bounded by $2^{O(n)}$ and degree bounded by $n(m'+p) \cdot \mathbf{d}_{max} = O(n)$ ($m' \leq m$ and p are as in that definition). We can thus use the unit cost measure.

Proof of Theorem 31: Let k, d be integers, let P be a polynomial expressed as $\sigma \circ P_\xi$ for an MS formula ξ and a substitution σ. We aim at computing its d-truncation. Let Φ be the corresponding set of formulas as in Theorem 28. We observe that by Lemma 30 (1), Corollary 29 yields:

(1) For every $\varphi \in \Phi$, for every $op \in \mathbf{U}_k$ for every ordered k-graph G:
$\sigma(P_\varphi(op(G))) \upharpoonright d = \sigma(P_{\varphi^{op}}(G)) \upharpoonright d.$

(2) For every $\varphi \in \Phi$ for every disjoint ordered k-graphs G and H:
$\sigma(P_\varphi(G \overrightarrow{\oplus} H)) \upharpoonright d = \sum_{1 \leq i \leq p}((\sigma(P_{\theta_i}(G)) \upharpoonright d) \cdot (\sigma(P_{\psi_i}(H)) \upharpoonright d)) \upharpoonright d.$

Note that we do not have: $\sigma(P(G)) \upharpoonright d = \sigma(P(G) \upharpoonright d)$ in general.

For a graph G given by a k-expression $\overrightarrow{t}$ that defines it together with a linear order on its vertex set, we will use (1) and (2) above to compute for each subterm s of $\overrightarrow{t}$ the family of "truncated polynomials" $((\sigma \circ P_\varphi)(val(s)) \upharpoonright d)_{\varphi \in \Phi}$. At the end of the computation, we get the Φ-tuple $((\sigma \circ P_\varphi)(G) \upharpoonright d)_{\varphi \in \Phi}$ from which we can extract the desired polynomial $(\sigma \circ P_\xi)(G) \upharpoonright d$. As observed above, the time to compute $\sigma(P(G)) \upharpoonright d$ is $2n \cdot c_G \cdot p_{max} \cdot \mid \Phi \mid$, where c_G bounds the costs of adding and multiplying the polynomials occuring in recursion rules (1) and (2). We need only count multiplications which are more costly than additions and in proportional number. Let us assume that all the polynomials $\sigma(P_\varphi(G))$ are in $\mathbb{Z}[\mathbf{X}_G \cup U]$. By Lemma 30 we have, for the d-truncations of all such polynomials P, Q:

(i) $c_G = O(\mathbf{v}_{\max} \cdot \mid P \upharpoonright d \mid^2 \cdot \mid Q \upharpoonright d \mid)$ with

(ii) $\mathbf{v}_{\max} = n \cdot \mid X \mid + \mid U \mid = O(n)$, and

(iii) $\mid P \upharpoonright d \mid = O(n^{2d+|U|})$.

For proving (iii), we note that a monomial of $P \upharpoonright d$ is a product of at most d factors of the form x_a^s and of at most $\mid U \mid$ factors of the form u^s, in both cases for $s \leq \deg(P) \cdot \mathbf{d}_{max}$. There are $n \cdot \mid X \mid \cdot \deg(P) \cdot \mathbf{d}_{\max}$ factors of the form x_a^s and $\deg(P) \cdot \mathbf{d}_{\max}$ factors u^s for each u. Hence:

$$\mid P \upharpoonright d \mid = O((n \cdot \deg(P))^d \cdot \deg(P)^{|U|}) = O(n^{2d+|U|})$$

since $\deg(P) = O(n)$. This gives for $2n \cdot c_G \cdot p_{max} \cdot \mid \Phi \mid$ the bound $O(n^{2+6d+3|U|})$.

We must take into account the cost of building for a graph G of clique-width at most k a clique-width expression. In cubic time, one can construct for graphs of clique-width at most k an $f(k)$-expression, for a fixed function f, by the results of [21, 28]. This suffices for our purposes. The total time is thus $O(n^t)$ where $t = Max\{3, 2+6d+3\cdot \mid U \mid\}$. (This bound applies if $d = \mid U \mid = 0$.) □

We now extend this result to numerical evaluations. Let $P(G)$ be a polynomial in $\mathbb{Z}[\mathbf{X}_G \cup U]$. An *evaluating substitution* ν replaces indeterminates by numerical values. Let ν be such a mapping. One can consider $\nu(P(G))$ as a polynomial reduced to a constant, that is the desired value of $P(G)$ for the values of indeterminates specified by ν. Note that $\nu(P(H))$ is well-defined for every graph H with set of vertices included in the set V of vertices of G. This remark will be useful for the computation of $\nu(P(G))$ by an induction on the structure of G using Corollary 29. The costs of computations are the same for polynomials with numerical values of their indeterminates since we use unit cost measure.

Corollary 33: Let k be an integer. For every generalized MS-polynomial P and every evaluating substitution ν, the corresponding value of $P(G)$ for a graph G of clique-width at most k can be computed in cubic time in the number of vertices of G. It can be computed in linear time if the graph is given by a k-expression.

Proof: Let ν be an evaluating substitution. It associates a number with each u in U and each x_a in $\mathbf{X}_G$, for a given graph G. As in the proof of Theorem 31 we have:

(1) For every $\varphi \in \Phi$, for every $op \in \mathbf{U}_k$, for every ordered k-graph H with $V_H \subseteq V_G$:
$\nu(P_\varphi(op(H))) = \nu(P_{\varphi^{op}}(H))$.

(2) For every $\varphi \in \Phi$ and all disjoint ordered k-graphs H and H' with $V_H, V_{H'} \subseteq V_G$:
$\nu(P_\varphi(H \overrightarrow{\oplus} H')) = \sum_{1 \leq i \leq p} \nu(P_{\theta_i}(H)) \cdot \nu(P_{\psi_i}(H'))$.

Here we compute the family of values $(\nu(P_\varphi(val(s))))_{\varphi \in \Phi}$ as opposed to a family of polynomials. The cost is thus $2n \cdot p_{max} \cdot \mid \Phi \mid = O(n)$ assuming known a k-expression defining G. Otherwise, one must construct an $f(k)$-expression and this can be done in cubic time. □

We now make precise the bounds for the interlace polynomial C and its specializations. Theorem 23 is actually a corollary of Theorem 31.

Proof of Theorem 23:

For each k, for every graph G with n vertices and of tree-width or clique-width at most k, we have the following bounds:

$\|C(G)\| \leq (3^n, 2n, 1)$ and $\mid C(G) \upharpoonright d \mid = O(n^{d+2})$,
$\|C_{y=0}(G)\| \leq (2^n, 2n, 1)$ and $\mid C_{y=0}(G) \upharpoonright d \mid = O(n^{d+1})$,
$\|C_{x=y}(G)\| \leq (n2^n, 2n, 2^n)$ and $\mid C_{x=y}(G) \upharpoonright d \mid = O(n^{d+2})$,
$\|C_I(G)\| \leq (2^n, 2n, 1)$, and $\mid C_I(G) \upharpoonright d \mid = O(n^{d+1})$,
$\|q(G)\| \leq (n^2, 2n, 2^n)$,
$\|Q(G)\| \leq (n+1, n, 3^n)$.

As in the proof of Theorem 29, we need only bound the costs c_G of the multiplications of polynomials. From these evaluations, we get the bounds $O(n^{3d+8})$ for C and $C_{x=y}, O(n^{3d+5})$, for $C_{y=0}$ and C_I, $O(n^7)$ for q and $O(n^4)$ for Q. $\square$

The result by Bläser and Hoffmann [5] indicates that restrictions to bounded tree-width or clique-width (or some alternative restriction) are necessary in Theorem 23 and Corollary 33. The result of [15] showing that the interlace polynomial can be evaluated in polynomial time on distance hereditary graphs is a special case of Corollary 33 because these graphs have clique-width at most 3.

Remark about the size of constants.

Sets Φ in Theorem 28 are very large if they are constructed in a blind manner from monadic second-order formulas: their sizes are towers of exponentials proportional to the quantification depth h of the formulas ξ which specify the considered polynomials. The reason is that the general proof based on the Feferman-Vaught paradigm exposed in [24] takes as set Φ the set of all formulas in some kind of normal form that have quantifier depth at most h. However, if alternatively, families of polynomials $\sigma \circ P_\varphi$ satisfying Corollary 29 are constructed directly, by taking advantage of the meanings of the properties defined by formulas ξ, then one may obtain usable recursive definitions.

The upper bounds of Theorem 23 leave a great space for improvements.

6 Conclusion

We have defined a multivariate interlace polynomial that generalizes the existing interlace polynomials. The multivariate methodology puts in light the *meaning of polynomials*. Classical polynomials are degraded versions of multivariate ones. The multivariate approach is well-adapted to the logical description of polynomials, and the use of monadic second-order logic yields fixed parameter algorithms for evaluating polynomials at particular values of indeterminates or for computing significant portions of them, called *truncations*.

For computing such polynomials in full, one might use linear delay enumeration algorithms and try to obtain monomials one by one, by increasing degrees, with a delay between two outputs linear in the size of the next output. Such algorithms are considered in [10] and [4] for monadic second-order definable problems and graphs of bounded clique-width.

Another research perspective consists in enriching the notion of configuration. In this article, in particular in Section 5, a configuration is an m-tuple of subsets for fixed m. One could try to extend the methodology of Section 5.3 to more complex configurations like partitions of unbounded size or permutations.

Finally, in order to build a zoology as opposed to maintaining a zoo (as pointed out by J. Makowsky in [26]) it is important to relate the various polynomials by means of algebraic reductions (specializations), or logical reductions or transformations of similar kinds.

Acknowledgements: This work is part of a larger project concerning the logical definition and the complexity of computation of graph polynomials conducted with J. Makowsky. I thank S. Oum for the equality of Lemma 2 (4). I thank also the referee for many useful comments.

7 References

[1] M. Aigner, H. van der Holst, Interlace polynomials, Linear algebra and applications **377** (2004) 11-30.

[2] R. Arratia, B. Bollobás, G. Sorkin, The interlace polynomial: a new graph polynomial, Journal of Combinatorial Theory Series B **92** (2004) 199-233.

[3] R. Arratia, B. Bollobás, G. Sorkin, A two-variable interlace polynomial, Combinatorica **24** (2004) 567-584.

[4] G. Bagan, MSO queries on tree decomposable structures are computable with linear delay, *Proceedings of Computer Science Logic 2006*, Lec. Notes Comput. Sci. **4207** (2006) 167-181.

[5] M. Bläser, C. Hoffmann, On the complexity of the interlace polynomial, Proceedings of 25th Annual Symposium on Theoretical Aspects of Computer Science, Dagstuhl Seminar Proceedings **08001**, IBFI, Germany, 2008, pp. 97-108.

[6] B. Bollobás, O. Riordan, A Tutte polynomial for coloured graphs, Combinatorics, Probability and Computing **8** (1999) 45-93.

[7] A. Bouchet, Circle graph obstructions, Journal of Combinatorial Theory Series B **60** (1994) 107-144.

[8] A. Bouchet, Graph polynomials derived from Tutte-Martin polynomials, Discrete Mathematics **302** (2005) 32-38.

[9] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, Chapter 5 of the "Handbook of graph grammars and computing by graph transformations, Vol. 1: Foundations", G. Rozenberg ed., World Scientific, 1997, pp. 313-400.

[10] B. Courcelle, Linear delay enumeration and monadic second-order logic, March 2006, to appear in Discrete Applied Mathematics.

[11] B. Courcelle, J. A. Makowsky, U. Rotics, On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic, Discrete Applied Mathematics **108** (2001) 23-52.

[12] B. Courcelle, M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs. Theor. Comput. Sci. **109** (1993) 49-82.

[13] B. Courcelle, S. Oum, Vertex-minors, monadic second-order logic and a conjecture by Seese, Journal of Combinatorial Theory Series B **97** (2007) 91-126.

[14] R. Downey et M. Fellows, *Parameterized complexity*, Springer-Verlag, 1999

[15] J. Ellis-Monaghan, I. Sarmiento, Distance hereditary graphs and the interlace polynomial, Combinatorics, Probability and Computing **16** (2007) 947-973.

[16] J. Ellis-Monaghan, L. Traldi, Parametrized Tutte polynomials of graphs and matroids, Combinatorics, Probability, and Computing **15** (2006) 835-854.

[17] M. Fellows, F. Rosamond, U. Rotics, S. Szeider, Clique-width minimization is NP-hard, Proceedings of the 38th annual ACM Symposium on Theory of Computing, Seattle, 2006, pp. 354-362.

[18] J. Flum, M. Frick, M. Grohe, Query evaluation via tree-decompositions, J. ACM **49** (2002) 716-752.

[19] J. Flum, M. Grohe, *Parametrized complexity theory*, Springer, 2006.

[20] M. Golumbic, U. Rotics, On the clique-width of some perfect graph classes, Int. J. Found. Comput. Sci. **11** (2000) 423-443.

[21] P. Hliněný, S. Oum, Finding branch-decompositions and rank-decompositions, Proceedings ESA, 2007, Lec. Notes Comput. Sci. **4698** (2007) 163-174.

[22] M. Las Vergnas, Le polynôme de Martin d'un graphe eulérien, Annals of Discrete Mathematics **17** (1983) 397-411.

[23] V. E. Levit, E. Mandrescu, The independence polynomials: a survey, Preprint, october 2005.

[24] J. Makowsky, Algorithmic uses of the Feferman–Vaught theorem, Annals of Pure and Applied Logic **126** (2004) 159–213.

[25] J. Makowsky, Coloured Tutte polynomials and Kauffman brackets for graphs of bounded tree-width, Discrete Applied Mathematics **145** (2005) 276-290.

[26] J. Makowsky, From a zoo to a zoology: Descriptive complexity for graph polynomials, In: A. Beckmann, *et al.* eds., Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, Lec. Notes Comput. Sci. **3988** (2006) 330-341.

[27] J. Makowsky, U. Rotics, I. Averbouch, B. Godlin, Computing graph polynomials on graphs of bounded clique-width, In: Proceedings of WG06, H. Bodlaender *et al.* eds, Lec. Notes Comput. Sci. **4271** (2006) 191-204.

[28] S. Oum, Approximating rank-width and clique-width quickly. Proceedings WG 2005, Lec. Notes Comput. Sci. **3787** (2005) 49-58.

[29] A. Sokal, The multivariate Tutte polynomial (alias Potts model) for graphs and matroids, Surveys in Combinatorics, in Volume **327** of London Math. Soc. Lec. Notes, 2005, pp. 173-226.

[30] L. Traldi, A dichromatic polynomial for weighted graphs and link polynomials, Proc. of American Math. Soc. **106** (1989) 279-286.

[31] T. Zaslavsky, Strong Tutte functions of matroids and graphs, Trans. Amer. Math. Soc. **334** (1992) 317–347.