Counting Truth Assignments of Formulas of Bounded Tree-Width or Clique-Width

E. Fischer [a,1] J.A. Makowsky [a,2] E.V. Ravve [b,3]

[a] *Department of Computer Science, Technion–Israel Institute of Technology, Haifa, Israel*
e–mail: {eldar,janos}@cs.technion.ac.il

[b] *Tower Semiconductors Ltd., Migdal Haemeq, Israel*

Abstract

We study algorithms for ♯SAT and its generalized version ♯GENSAT, the problem of computing the number of satisfying assignments of a set of propositional clauses Σ. For this purpose we consider the clauses given by their incidence graph, a signed bipartite graph $SI(\Sigma)$, and its derived graphs $I(\Sigma)$ and $P(\Sigma)$.

It is well known, that, given a graph of tree-width k, a k-tree decomposition can be found in polynomial time. Very recently S. Oum and P. Seymour have shown that, given a graph of clique-width k, a $(2^{3k+2}-1)$-parse tree witnessing clique-width can be found in polynomial time.

In this paper we present an algorithm for ♯GENSAT for formulas of bounded tree-width k which runs in time $4^k(n + n^2 \cdot \log_2(n))$, where n is the size of the input. The main ingredient of the algorithm is a splitting formula for the number of satisfying assignments for a set of clauses Σ where the incidence graph $I(\Sigma)$ is a union of two graphs G_1 and G_2 with a shared induced subgraph H of size at most k. We also present analogue improvements for algorithms for formulas of bounded clique-width which are given together with their derivation.

This improves considerably results for ♯SAT, and hence also for SAT, previously obtained by Courcelle, Makowsky and Rotics, [CMR01].

Key words: Parametrized Complexity, Polynomial Hierarchy, Monadic Second Order Logic, Tree-width

[1] partially supported by the VPR-Fund - Dent Charitable Trust - non-military research fund of the Technion - Israel Institute of Technology.
[2] partially supported by a Grant of the Fund for Promotion of Research of the Technion–Israel Institute of Technology.
[3] freely collaborating with J.A. Makowsky

For Peter Hammer
on the occasion of
his 60th birthday

1 Introduction and statement of result

1.1 The problem

We study algorithms for ♯SAT, and ♯GENSAT, the problem of computing the number of satisfying assignments of a set of (generalized) propositional clauses Σ. It was shown by L. Valiant [Val79] that ♯SAT is ♯**P**-complete. ♯GENSAT is the counting problem associated with the generalized satisfiability problem introduced by T.J. Schaefer [Sch78], who proved also a dichotomy theorem, classifying the problems into polynomially solvable cases and **NP**-complete cases, and nothing in between. Instances Σ of GENSAT(S) consist of generalized clauses $r_i(\bar{v}) : i \in \mathbb{N}$, where $\bar{v}$ is a vector of $\rho(i)$ propositional variables. $\rho(i)$ is called the size or arity of r_i. The truth of r_i is given by the truth tables in $S = \{R_i : i \in \mathbb{N}\}$ over the variables of r_i. In [Sch78] S is assumed to be finite. A dichotomy theorem for ♯GENSAT was proven by N. Creignou and M. Hermann in [CH96], where only polynomial time computable and ♯**P**-complete cases occur. For a unified treatment of these results, cf. the book by N. Creignou, S. Khanna and M. Sudhan [CKS01]. For each version of GENSAT, the instances Σ can be expanded into a set of clauses Σ^{exp} of SAT, such that each satisfying assignment z makes Σ true for GENSAT iff z makes Σ^{exp} true for SAT. Note however that in general Σ^{exp} could be exponentially bigger than Σ. We shall introduce the formal framework and examples for GENSAT and ♯GENSAT in Section 2.

We associate with Σ three graphs. The graph $SI(\Sigma)$ is a signed bipartite graph with the variables and clauses of Σ as vertices, indicating whether variables occur positively or negatively in a clause. The graphs $P(\Sigma)$ (the primal graph of Σ) and $I(\Sigma)$ (the incidence graph of Σ) are unsigned graphs. $P(\Sigma)$ has only the variables as its vertices, and edges indicate that two variables occur in a common clause. $I(\Sigma)$ is obtained from $SI(\Sigma)$ by omitting the signs on the edges.

In the first part of the paper (Sections 2-5) we shall study the complexity of $\sharp$GENSAT under the assumption [4] that the tree-width $tw(P(\Sigma))$ of $P(\Sigma)$ or the tree-width $tw(I(\Sigma))$ of $I(\Sigma)$ is bounded by a fixed number $k \in \mathbb{N}$. The exact definitions of these graphs and of tree-width are given in Section 3, where we also discuss examples of formulas of bounded and unbounded tree-width.

Let us note here already the observation of G. Gottlob and R. Pichler, [GP01]:
Proposition 1.1. *For every generalized clause set Σ we have*

$$tw(I(\Sigma)) \leq tw(P(\Sigma)) + 1$$

It was pointed out in Courcelle, Makowsky and Rotics [CMR01] that graph counting problems where the objects to be counted are definable in Monadic Second Order Logic, MSOL [5], are solvable in polynomial time when restricted to graphs of tree-width at most k, for some fixed $k \in \mathbb{N}$. In [CMR01] no estimate of the constants involved is given, but using [Mak04] one can get estimates which depend on the quantifier rank q of the defining formula and an upper bound of $exp_q(c \cdot k) \cdot n^3$ with c small. Here $exp_1(k)$ is the function 2^k and $exp_{m+1}(k) = 2^{exp_m(k)}$.

The method developed in [CMR01] has various applications in the theory of graph polynomials, cf. [Mak01,MM03a,Mak05,Mak04].

To apply the methods of [CMR01] to $\sharp$SAT, one notes that SAT is indeed definable in MSOL over $SI(\Sigma)$. A satisfying assignment can be identified with a subset of vertices V_0 (the variables which are assigned the value `true`), which has the property that every clause contains a literal $v \in V_0$ or it contains a literal $\neg v$ with $v \in V - V_0$. This is easily expressible as a formula $\phi_{sat}(V_0)$ of MSOL of quantifier depth 2. Clearly Σ is satisfiable iff

$$SI(\Sigma) \models \exists V_0 \phi_{sat}(V_0).$$

Furthermore, the function

$$\mathtt{csat}(\Sigma) = \mid \{V_0 \subseteq V : SI(\Sigma), V_0 \models \phi_{sat}(V_0)\} \mid$$

[4] In [CKS01, Chapter 8], other input-restricted satisfaction problems are considered, such as PLANAR$-$SAT, where $I(\Sigma)$ is assumed to be planar, or DENSE$-$SAT, where the number c of clauses over n variables is $\Omega(n^m)$, where $m = \max_i\{\rho(i)\}$. These restrictions leave the SAT **NP**-complete, but make $PLANAR - \mathrm{MAX} - \mathrm{SAT}$ and $DENSE - \mathrm{MAX} - \mathrm{SAT}$ approximable with polynomial time approxiamation schemes (PTAS).

[5] This holds also if we replace MSOL by CMSOL, where we also allow all the modular counting quantifiers $C_{m,p}x\phi(x)$ which state that the number of elements satisfying $\phi(x)$ equals m modulo p.

counts the number of satisfying assignments.

So it follows from [CMR01,Mak04] for ♯SAT that

Theorem 1.2. `csat` *can be computed, and hence also* SAT *can be solved, in time* $O(n^3)$ *for sets of clauses* Σ *with* $tw(I(\Sigma)) \leq k$*, where the constants depend on* k *only, but are at least doubly exponential in* k*.*

Also in [CMR01,Mak04] it is shown that a similar theorem holds for `csat` with formulas where $SI(\Sigma)$ is of bounded clique-width, provided the input is given together with a parse tree of the clique-width. In the second part of the paper (Section 6) we discuss this extension in greater detail.

The rather bad estimate of the size of the constants in the algorithms presented in [CMR01,Mak04] is due to the general character of the method. The proof of Theorem 1.2 uses the Feferman-Vaught Theorem for MSOL, and works for arbitrary counting functions given by MSOL-formulas $\phi(V_0)$ as above. The general running time will be a q-fold iterated exponential of k, where q is the quantifier depth of ϕ. M. Grohe and M. Frick, [FG04], have shown that, unless $\mathbf{P} = \mathbf{NP}$, this is unavoidable for the general method.

1.2 Main results for bounded tree-width

Rather than using the general method of [CMR01,Mak04], we present here a method specially tailored for ♯SAT, which reduces the size of the constants to be simply exponential in the tree-width $tw(I(\Sigma))$ of the incidence graph of Σ. We state our results in a model of computation where arithmetic operations of integers have unit cost. Addition cost of n-bit numbers in bits is $O(n)$, and of multiplication roughly $O(n \log_2(n))$, so the results have to be modified correspondingly, if the complexity is to be measured in bits. For details and optimal bounds cf. the classical monograph [AHU74].

Our results are stated for given tree-decompositions of the incidence and primal graphs ($I(\Sigma)$ and $P(\Sigma)$) of Σ. There are algorithms that find a tree decomposition of bounded width, given a graph of treewidth at most some constant k, and run in $O(n^2)$ time with constants simply exponential in k, cf. [Ami02]. Proofs of our results are given in Sections 4 and 5.

Theorem 1.3. *Given a* k*-tree decomposition of* $I(\Sigma)$*,* `csat`(Σ) *can be computed, and hence also* SAT *can be solved, if restricted to* Σ *with* $I(\Sigma)$ *of tree-width at most* k*, using* $4^k \cdot n$ *arithmetic operations (or in time* $4^k(n+n^2 \cdot \log_2(n))$ *if bit cost is applied).*

When considering ♯GENSAT, Theorem 1.3 can be applied to Σ^{exp}, provided that both the size of Σ^{exp} and the tree-width $tw(I(\Sigma^{exp}))$ are polynomially bounded in the size of Σ, respectively the tree-width of $I(\Sigma)$. For example this is the case, if the size of each clause is bounded by $\log_2(n)$, where n is the size of Σ. However there are instances Σ of size n of GENSAT with $tw(I(\Sigma)) = 1$ and $tw(I(\Sigma^{exp})) = n$. In contrast we have:

Proposition 1.4. *For every instance Σ for* GENSAT(S)

(i) $tw(P(\Sigma^{exp})) = tw(P(\Sigma))$.
(ii) *If the arities $\rho(i)$ of the clauses in S are bounded by m,* $tw(I(\Sigma^{exp})) \leq tw(I(\Sigma)) \cdot m$.

Using Proposition 1.4 gives immediately our main result for ♯GENSAT:

Theorem 1.5. *Given a k_1-tree decomposition of $P(\Sigma)$, a k_2-tree decomposition of $I(\Sigma)$, let $m = \max_i\{\rho(i)\}$ (if it exists), and $k_3 = \max_i\{\rho(i), k_2\}$. Then* ♯GENSAT$(S)$ *can be computed*

(i) *with $4^{k_1} \cdot n^2$ arithmetic operations, provided the size of each clause is bounded by* $\log_2(n)$*;*
(ii) *with $4^{k_1+m} \cdot n$ arithmetic operations, provided the size of each clause is bounded by $m \in \mathbb{N}$;*
(iii) *with $4^{k_3 \cdot m} \cdot n$ arithmetic operations, provided the size of each clause is bounded by $m \in \mathbb{N}$.*

This includes the classical cases ♯NOT-ALL-EQUAL 3SAT and ♯ONE-IN-THREE 3SAT of [GJ79, Problem list A9].

We could also apply Theorem 1.2 to ♯GENSAT, but not all versions of GENSAT are MSOL-definable, for example, HALFSAT, where we require that in each clause at least half of the literals are true. Let HALFSAT$_{f(n)}$ be like HALFSAT but with the size of the clauses bounded by $f(n)$, a function of the input size. If f is the constant function $b \in \mathbb{N}$, we write also HALFSAT$_b$. HALFSAT$_b$ is MSOL-definable by a formula ϕ_b, but the quantifier rank $q = qr(\phi_b) \geq b + 2$. Theorem 1.5 includes also cases not covered by Theorem 1.2:

Corollary 1.6. *For every $f(n)$ the problem* ♯HALFSAT$_{f(n)}$ *restricted to instances Σ with $tw(P(\Sigma)) \leq k$ can be computed with $4^k \cdot 2^{2 \cdot f(n)} \cdot n$ arithmetic operations.*

Finally, using the self-reducibility of SAT, cf. [Pap94, Example 10.3, p. 228], we get also a generating algorithm with polynomial delay in the sense of [JYP88]. These are algorithms which enumerate all instances of a problem where the time elapsing between two such instances is polynomial in the size of the problem. Clearly, this allows an exponential number of instances to be produced.

In our situation we have:

Corollary 1.7. *Under the assumptions of Theorem 1.5,* GENSAT*, restricted to instances Σ with $tw(P(\Sigma)) \leq k$, has a generating algorithm with polynomial* *delay.*

1.3 Main results for bounded clique-width

The notion of clique-width was introduced in [CER93] and studied more systematically in [CE95,Cou92,EvO97,CO00]. In the last ten years, the study of graphs of bounded clique-width became very popular, cf. the work of A. Brandstaedt, B. Courcelle, V.V. Lozin, P. Seymour, J. Spinrad, and their many collaborators.

Clique-width is a more general notion than tree-width and measures somehow how a graph can be built from smaller graphs by remembering only that certain nodes are coloured and the number of colours is fixed. The main difference is the important rôle of the complete bipartite subgraphs. If large bipartite subgraphs are excluded, then bounded clique-width yields bounded tree-width, cf. [Cou03]. Courcelle and Olariu in [CO00] showed that clique-width of graphs of tree-width k, is at most $2^{k+1} + 1$. Therefore, any class of graphs of bounded tree-width, is automatically of bounded clique-width. Moreover, B. Courcelle, J. Engelfriet and G. Rozenberg in [CER93] provided a complicated proof that any given context-free graph grammar based on vertex-replacement (Confluent NCE, or context-free VR grammar) generates graphs of bounded clique-width. Although an upper bound for the clique-width could be derived from their proof, it is not straightforward. In general, finding an explicit bound for the clique-width is a more complicated task than finding a bound for the tree-width. For explicit computations of clique-width, cf. [GR00], [GM03]. In contrast to tree-width, there is also a natural notion of clique-width for directed graphs or signed graphs, which is different from the undirected (unsigned) case. Recall that we denote by $SI(\Sigma)$ the signed version of the incidence graph of Σ where edges are labeled depending whether the variable occurs positively or negatively in a clause.

To get an analogue of Theorem 1.5 one needs a parse tree of the graph with respect to its clique-width. We denote by $der_{SI}(\Sigma)$ or $der_I(\Sigma)$ such a parse tree for the signed, respectively unsigned case. Details are given in Section 6.

By a recent result of S. Oum and P. Seymour, [OS04], described in more detail in Section 6, Theorem 6.2, this can be achieved in the following way, which suffices for our purposes. There is a function f, such that, for given k, there is a polynomial time algorithm that, with input a graph G, either concludes that its clique-width is larger than k, or outputs an $f(k)$-parse tree for G. By a straight inspection of their proof a similar theorem can be proven also for the clique-width of signed graphs where $f(k)$ is replaced by a function $g(k)$ of the same order of growth.

Using the parse tree obtained from the signed version of this theorem, we can now apply our result.

Theorem 1.8. *Given a set of clauses Σ and a signed parse tree $der_{SI}(\Sigma)$ for clique-width of up to k, it is possible to calculate* $\mathrm{csat}(\Sigma)$*, with a number of algebraic operations that is linear in the size of the parse tree $der_{SI}(\Sigma)$, and simply exponential in k.*

This theorem can also be extended to solve $\sharp GENSAT$, but we leave this to the reader. We also believe that a corresponding theorem for unsigned clique-width is true, but we did not work out the details for this paper.

1.4 Significance and applicability of the results

As pointed out by R. Downey and M. Fellows in [DF99] there is a long way to go from establishing that a problem is fixed parameter tractable, FPT, to feasible algorithms. In [CMR01], it was first established that MSOL-definable counting problems are FPT, with constants being multiply exponential with tree-width k as the parameter k. We make the following significant improvements:

- In the case of SAT and $\sharp$SAT with tree-width k as parameter we bring the constants down to being simply exponential in k.
- In the case of SAT and $\sharp$SAT with clique-width k as parameter we also bring the constants down to being simply exponential in $g(k)$. We shall discuss in section 6, how this can be further improved to be simply exponential in k.
- We show many versions of GENSAT and $\sharp$GENSAT to be FPT with the same parameter k and the constants simply exponential in k, even when they are not MSOL-definable.

In industrial applications of hardware and software verification, the problem is often presented in two steps. First a labeled graph G is built for which a property Φ has to be verified. The labeled graph was generated by some graph grammar which takes into account that only a fixed number of labels are used and reflects the modularity of the hardware design or the well-structured character of the software, cf. [Tho98]. As a result of this, cf. [GM03], the graphs are *a priori* of bounded tree-width or clique-width, depending on the particlar grammar only. The tree-decompositions, respectively the parse tree of the clique-width, can be explicitely computed from the parse tree in the graph grammar. In real-life applications of hardware verfication, related methods using tree-width have been successfully implemented, cf. [BKDSZ,WCZK], and the references therein.

In a second step the verification of Φ on G is translated uniformly into an instance of SAT. If the latter translation can be expressed as an MSOL-transduction, it was shown by Courcelle and Engelfriet, cf. [CE95,Cou92,EvO97], that the resulting instance Σ of SAT has an incidence graph, the clique-width of which depends only on the tree-width or clique-width of G and Φ. For a detailed exposition, cf. [CM02]. It remains to be explored in detail, in which concrete situations this can be used.

In applications in Artificial Intelligence very large sets of clauses (rules and facts) have to be tested for satisfiability. But the clauses are often naturally partioned into sets coming from different domains of discourse, where the shared variables are few. E. Amir has explored this in great detail, [Ami01]. In the course of his work he has shown that partitioning sets of clauses in this way is related to the tree-width of the clause graph $P(\Sigma)$. Low tree-width gives good partitions, and partitions with cyclefree overlapping of the variables give also tree-decompositions of low tree-width. To quote from [Ami01, Section 5.2, page 90]:

> *We believe that in domains that deal with engineered physical systems, many of the domain axiomatizations have these structural properties. Indeed, design of engineering artifacts encourages modularization with minimal interconnectivity, see [Ami00,Len95,CSJ+98]. More generally, we believe axiomatizers of large corpora of real-world knowledge tend to try to provide structured representations following some of these principles. Recent experiments with the HPKB knowledge base of SRI and a part of the Cyc knowledge base support this belief. Those experiments are reported in [Ami01, Section 5.8].*

So tree-width and clique-width turn out to be natural concepts in industrial applications of SAT, both in verification of software and hardware, and in automated reasoning.

1.5 Methods

The main ingredient of the algorithm is a Feferman-Vaught-type theorem, cf. [Mak04], in form of a splitting formula for the number of satisfying assignments for a set of non-generalized clauses Σ where the incidence graph $I(\Sigma)$ is a union of two graphs G_1 and G_2 with a shared induced subgraph H of size at most k. This is given as Theorem 4.7 in Section 4. Such splitting formulas are well known for graph polynomials for $k = 0, 1$. In the case of H consisting of the empty set or only one vertex, many graph polynomials are multiplicative, e.g., the Tutte polynomial, the matching polynomials and others, cf. [Bol99,Mak04]. In the case of H consisting of two vertices, such a splitting formula was proven by J. Oxley and D. Welsh [OW92] for the Tutte polynomial. For H of arbitrary fixed size k, splitting formulas were established by S. Negami [Neg87], A. Andrzejak [And97], S. Noble [Nob98] and L. Traldi [Tra] for various versions of the Tutte polynomial. In [Mak04] a general existence theorem for such splitting formulas is given. Theorem 1.3 is the result of searching for a splitting formula for the function `csat`.

1.6 Related work

The study of $\sharp$SAT on formulas with $I(\Sigma)$ of bounded tree-width and clique-width was initiated in [CMR01]. SAT on various presentations of the clauses as graphs and restricted to inputs of tree-width at most k was previously studied, among others, by R. Dechter and J. Pearl [DP89] and T. Feder and M. Vardi [FV99]. More recent work was presented by G. Gottlob and R. Pichler [GP01], E. Amir and S. McIlraith [AM05,AM01], M. Alekhnovich and A. Razborov [AR02], and S. Szeider [Sze03].

Most of the previous results are stated for $P(\Sigma)$ having bounded tree-width. In the case of [AR02] branch-width of the clause hypergraph is studied. Here the vertices are the variables, and the hyperedges are the clauses as sets of variables (disregarding negations). Our results are in general much stronger, as we only require that the tree-width of $I(\Sigma)$ or the clique-width of $SI(\Sigma)$ is bounded.

1.7 Outline of the paper

In Section 2 we define the general framework of the satisfiability problems which we consider. In Section 3 we give the necessary background concerning tree-width and clause graphs $P(\Sigma)$ and $I(\Sigma)$. In Section 4 we derive the splitting formula which allows us to count satisfying assignments for H-sums of instances of SAT. This is one of the main new algorithmic ingredients of the paper. In Section 5 we prove the main theorems for the case of bounded tree-width. In Section 6 we give the necessary background concerning clique-width and extend the results to the case of bounded clique-width In Section 7, finally, we draw some conclusions and discuss further research.

2 Generalized satisfiability

We follow closely [Sch78,CH96].

Let $S = \{R_i : i \in \mathbb{N}\}$ be an infinite set of logical relations of rank $\rho(i)$. A logical relation R_i of rank $\rho(i)$ is a subset of $\{0,1\}^{\rho(i)}$. An S-formula Σ is a set of (generalized) clauses of the form $r_i(\bar{v})$ where $\bar{v} = v_{j_1}, \ldots, v_{j_{\rho(i)}}$ are any propositional variables.

The size of an S-formula Σ is the sum of the sizes of its generalized clauses, irrespective of the choice of S. We denote the set of propositional variables by $\mathtt{Var}$ and the set of variables occurring in Σ by $\mathtt{Var}(\Sigma)$.

The S-satisfiability decision problem GENSAT(S) is the problem of deciding whether for a given S-formula Σ there is an assignment $z : \mathtt{Var} \rightarrow \{0, 1\}$ such that for each clause $r_i(\bar{v})$ in Σ, $z(\bar{v}) \in R_i$, i.e., all clauses are simultaneously satisfiable using the semantics given by the S. The S-satisfiability counting problem $\sharp$GENSAT(S) counts the number of satisfying assignments for Σ. If S is not explicitely mentioned we speak of an instance of GENSAT or $\sharp$GENSAT rather than of GENSAT(S) respectively $\sharp$GENSAT(S).

The classical satisfiability problem SAT usually is formulated with literals rather than variables only. When formulating SAT as an instance of GENSAT this amounts to having different r_i's for each distribution of the negation symbols among the literals. If the size of the clauses is bounded by a fixed number then S can be assumed finite.

All instances of GENSAT(S) are in **NP** and all instances of $\sharp$GENSAT(S) are in $\sharp$**P**.

T.J. Schaefer and N. Creignou and M. Hermann, cf. [Sch78,CH96], give a complete classification for which the corresponding instances given by S are **NP**-hard, respectively $\sharp$**P**-hard. They prove a Dichotomy Theorem which states that all the other cases are solvable in polynomial time.

For our purpose it suffices to note that if GENSAT(S) is **NP**-complete, then $\sharp$GENSAT(S) is $\sharp$**P**-complete. In other words, there is an abundance of $\sharp$**P**-complete instances of GENSAT.

3 Tree-width of clause graphs and H-sums

We assume the reader is familiar with some basic graph theory and the notion of a graph minor. A graph H is a minor of a graph G if H can be obtained from G by deleting or contracting edges and deleting vertices. General background on minors and tree-width may be found in [Die96].

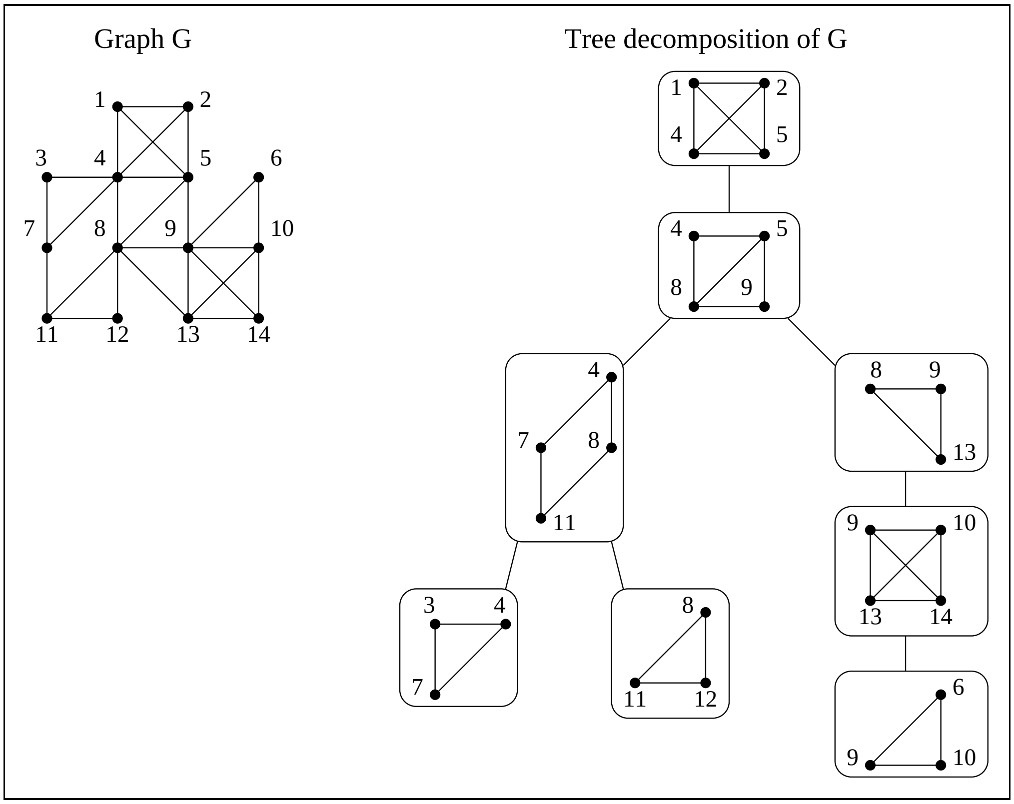

Fig. 1. Example of a 3-tree decomposition of a graph

3.1 Tree-width of graphs

Definition 3.1. *A* k-tree decomposition of G *is given as follows:*

(i) *We have a (not necessarily rooted) tree* $\mathcal{T} = \langle T, f \rangle$, *where* T *is a set and* f *is a function mapping nodes onto their father.*
(ii) *The vertex set* $V(G)$ *of the graph is the union of sets* A_t, *with* $t \in T$ *and* $|A_t| \leq k+1$.
(iii) *For every edge* $e = (x, y) \in E(G)$ *there is a* $t \in T$ *such that* $x, y \in A_t$.
(iv) *For each* $x \in V$ *the set* $T(x) = \{t \in T : x \in A_t\}$ *is a connected subgraph of* $\mathcal{T}$.

If the tree $\mathcal{T}$ *is a path (no branching) we speak of a* k*-path decomposition.*
Remark 3.2. *Under conditions (i)-(iii), (iv) is equivalent to: For every connected subgraph* H *of* G, *the set* $\{t \in T : V(H) \cap A_t \neq \emptyset\}$ *is a connected subtree of* $\mathcal{T}$.

Definition 3.3.

(i) G is of tree-width (resp. path-width) at most k, *if there exists a* k*-tree decomposition (resp.* k*-path decomposition) of* G. *In the literature such*

graphs are sometimes also called partial k-trees.

(ii) The tree-width (or path-width) of a signed (edge coloured) graph is, by definition, the same as its tree width without the colouring.

Given a graph G, finding its tree-width is **NP**-complete, cf. [ACP87], but for fixed k, checking whether G has tree width at most k (and if so, finding a witnessing tree decomposition), can be done in polynomial time, cf. [Bod97]. For the most advanced approximation algorithms to compute the tree-width, cf. [Ami02].

Example 3.4.

(i) The tree-width of a tree is 1.

(ii) The tree-width of C_n*, the cycle with* n *vertices, is* 2.

(iii) The tree-width of K_n*, the complete graph on* n *vertices, is* $n-1$*, and of* $K_{n,n}$*, the complete bipartite graph on twice* n *vertices, is* n.

(iv) The tree-width of the two dimensional square grid $Grid_{n,n}$ *on* n^2 *vertices is* n.

3.2 Tree-width of clause graphs

The incidence graph $I(\Sigma)$ of an S-formula Σ with variable set $Var(\Sigma)$ is the bipartite simple graph $I(\Sigma) = (\Sigma, Var(\Sigma), E_I)$ where for each generalized clause $C = r_i(\bar{v}) \in \Sigma$ we have that $(v, C) \in E_I$ iff $v \in C$.

The primal graph $P(\Sigma)$ of an S-formula Σ is the simple graph $P(\Sigma) = (Var(\Sigma), E_P)$ where for each $v_1, v_2 \in Var(\Sigma)$ the pair $(v_1, v_2) \in E_P$ iff there is a clause $C \in \Sigma$ where both v_1 and v_2 occur.

Recall that Proposition 1.1 in the introduction stated that for every generalized clause set Σ we have $tw(I(\Sigma)) \leq tw(P(\Sigma)) + 1$.

A natural example of formulas of bounded tree-width can be obtained as follows. Let $V = \{v_0, v_1, \ldots, v_m\}$ be a set of propositional variables and $\Sigma = \{C_0, C_1, \ldots, C_n\}$ a set of clauses over V.

Proposition 3.5. *Assume that there is* $d \in \mathbb{N}$ *such that, if* v_i *or* $\neg v_i$ *occurs in* C_j*, then* $\mid i - j \mid \leq d$*. Then* $I(\Sigma)$ *has path-width at most* d.

This example is related to the cut-width of the hypergraph representing the clauses and has been used successfully in very large real-life applications, cf. [WCZK].

To illustrate the concept of the tree-width of formulas, we look at three classical examples: Horn formulas, Tseitin formulas and pigeon-hole formulas.

Horn clauses are clauses with at most one literal non-negated. Checking satisfiablity of Horn formulas can be done in linear time, [IM87].
Proposition 3.6. *The tree-width of Horn-formulas is unbounded.*

Proof. Take the grid $Grid_{2n,2n}$. It is bipartite, with equal number of variables and clauses. Each clause contains at most four variables. For each clause we choose r_i to be a disjunction where exactly one variable occurs positively. This gives us Σ with $Grid_{2n,2n}$ as its underlying graph. Hence its tree-width is $2n$. □

G. Tseitin showed that the difficulty of proving inconsistency of the Tseitin-formulas $\Sigma(H)$ using regular resolution only depends on the properties of the underlying graph H viewed as an expander graph, cf. [Tse68,Gal77]. The regularity assumption for resolution was later removed by A.Haken. Haken also showed that the formulas PHP_n^{n+1} have long proofs of inconsistency using resolution, cf. [Hak85,CS88]. This is no accident, as it is shown by M. Alekhnovich and A. Razborov, that for sets of inconsitent clauses Σ with $tw(I(\Sigma))$ bounded by k, there are resolution proofs of polynomial length, [AR02]. A. Atserias and V. Dalmau, [AD03], give further interpretations of this phenomenon.

We now compute the tree-width of $I(\Sigma)$ for these examples of sets of (non generalized) clauses which are natural or occur in the literature. When no proofs are given, it is straightforward to verify the statements. The tree-width of $P(\Sigma)$ can be easily estimated using Proposition 1.1.

Tseitin-formulas are formulas obtained as follows: Let $H = (V, E)$ be a graph. Let $\mu : V \to \{0, 1\}$ be a marking of the vertices, with $\sum_{v \in V} \mu(v) = 1 \pmod 2$. We define $\Sigma(H, \mu)$ in the following way: The variables of the formula are represented by the edges in E, whereas the formula is the conjunction of all the clauses $F_v, v \in V$, where

$$F_v = \begin{cases} e_1(v) \oplus \ldots \oplus e_d(v) & \text{if } \mu(v) = 1 \\ \neg(e_1(v) \oplus \ldots \oplus e_d(v)) & \text{if } \mu(v) = 0 \end{cases}$$

and $e_1(v), \ldots, e_d(v)$ are the edges incident with v. It is straightforward to bring this into clausal form, which we denote by $T(H, \mu)$.
Proposition 3.7. *The tree-width of the Tseitin formulas $T(H, \mu)$ is at least as big as the tree-width of H.*

Proof. One can show that H is a minor of $I(T(H,\mu))$. $\square$

The pigeon-hole formulas PHP_n^{n+1} are defined as follows. We have variables $p_{i,j}$, a_i, $b_{i,j,k}$ for $i = 1, \ldots, n+1$ and $k, j = 1, \ldots, n$. $p_{i,j}$ stands for "pigeon i sits in hole j". a_i stands for "pigeon i sits in one of the holes". $b_{i,j,k}$ stands for "pigeon i and j sit both in hole k".

$$\mathrm{PHP}_n^{n+1} = \bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} p_{i,j} \rightarrow \bigvee_{k=1}^{n} \bigvee_{i,j=1, i\neq j}^{n+1} (p_{i,k} \wedge p_{j,k})$$

We use the additional variables to write it in readable clausal form. We write a_i for $\bigvee_{j=1}^{n} p_{i,j}$, and $b_{i,j,k}$ for $(p_{i,k} \wedge p_{j,k})$. This gives:

$$\bigvee_{i=1}^{n+1} a_i \vee \bigvee_{k=1}^{n} \bigvee_{i,j=1, i\neq j}^{n+1} b_{i,j,k}$$

We also add the clauses

$$A_i = a_i \leftrightarrow \bigwedge_{j=1}^{n} \neg p_{i,j}, \text{ and } B_{i,j,k} = b_{i,j,k} \leftrightarrow (p_{i,k} \wedge p_{j,k})$$

Proposition 3.8. *The tree-width of the pigeon-hole formulas* PHP_n^{n+1} *is at least* n.

Proof. The grids $Grid_{n,n}$ are minors of $G_{\mathrm{PHP}_n^{n+1}}$. $\square$

3.4 H-sums of graphs

Given a k-tree decomposition of a graph G with tree $\mathcal{T}$ and sets of vertices $A_t, t \in \mathcal{T}$, we denote by H_t the induced subgraph of G with vertex set A_t. Given the k-tree decomposition and all the induced subgraphs H_t, we can reconstruct the original graph G using successive (almost disjoint) unions. To make this precise we define the H-sum of two graphs.

Given two graphs G_1, G_2 with distinguished induced subgraphs H_1, H_2 which are isomorphic to H with isomorphisms h_1, h_2, the H-sum of G_1 and G_2 is an almost disjoint union of the two graphs where the intersection contains exactly H as induced subgraph (using the isomorphisms h_1 and h_2 to fix it)[6]. In other words:

Definition 3.9.

[6] Strictly speaking we should write $G_1 \oplus_{H,h_1,h_2} G_2$, but we shall drop the isomorphisms when there is no risk of confusion.

(i) *For* $i = 1, 2$ *let* $G_i = \langle V(G_i), E(G_i)\rangle$ *and* $V(G_1) \cap V(G_2) = V(H)$ *and* $E(H) = E(G_1) \cap V(H)^2 = E(G_2) \cap V(H)^2$. *Then* $G = G_1 \oplus_H G_2$ *is given by* $V(G) = V(G_1) \cup V(G_2)$ *and* $E(G) = E(G_1) \cup E(G_2)$.

(ii) H*-sums of edge and vertex coloured graphs are defined similarly.*

In the reconstruction process of G from $\mathcal{T}$ and the H_t's we have to perform a sequence of H-sums where H is always an induced subgraph of the H_t's.

4 A splitting formula for H-sums of clause graphs

In this section all clauses are non-generalized. Let Σ be a set of clauses over a variable set V, and let $W \subseteq V$ and $z : W \to \{0,1\}$ be a partial assignment. We denote by $\Sigma^{(z)}$ the set of clauses obtained from Σ by performing the substitution

$$s(v) = \begin{cases} \textbf{true} & \text{if } z(v) = 1 \\ \textbf{false} & \text{if } z(v) = 0 \end{cases}$$

Similarly, we denote by $\mathtt{csat}_z(\Sigma)$ the number of assignments z' with $z' \mid_W = z$ which make Σ true.

As any k-tree decomposition of $I(\Sigma)$ gives also a k-tree decomposition of $I(\Sigma^{(z)})$, clearly we have

Lemma 4.1.

(i) $\mathtt{csat}(\Sigma^{(z)}) = \mathtt{csat}_z(\Sigma)$.

(ii) $tw(\Sigma^{(z)}) \leq tw(\Sigma)$.

□

The following is a straightforward consequence of our notation.

Lemma 4.2. *With the notation from above we have*

$$\mathtt{csat}(\Sigma) = \sum_{z:W\to\{0,1\}} \mathtt{csat}(\Sigma^{(z)}) = \sum_{z:W\to\{0,1\}} \mathtt{csat}_z(\Sigma)$$

□

We now derive our splitting formula for the function `csat` for H-sums.

4.1 H-sums of incidence graphs $I(\Sigma)$

From now on, let Σ be a set of clauses, such that the corresponding incidence graph $G = I(\Sigma)$ is the H-sum $G_1 \oplus_H G_2$. We denote by Σ_i the set of clauses

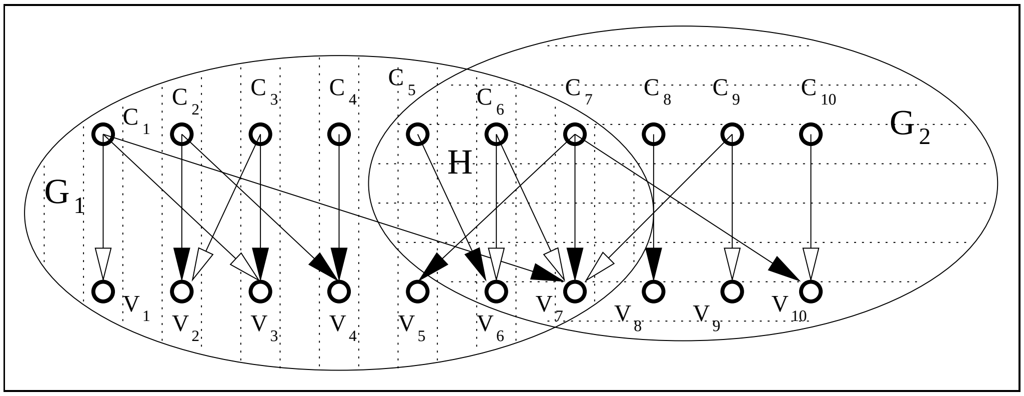

Fig. 2. H-sum of G_1 and G_2 where H contains both clauses and variables

with $I(\Sigma_i) = G_i$, cf. Figure 2.

We distinguish two extreme cases.

4.1.1 H contains only variables

If H contains only variables $W \subseteq V$, we can divide the clauses of Σ into four sets:

(i) $\Sigma_i^{V-W}, i = 1, 2$ which do not contain variables from W and such that the clauses are vertices in G_i, and
(ii) $\Sigma_i^W, i = 1, 2$ which do contain variables from W and such that the clauses are vertices in G_i.

Clearly, $\Sigma_i^{V-W} \cup \Sigma_i^W = \Sigma_i$.

Using Lemma 4.2 we get immediately:
Lemma 4.3. *With the notation from above we have*

$$\texttt{csat}(\Sigma_1 \oplus_W \Sigma_2) = \texttt{csat}(\Sigma) = \sum_{z:W\to\{0,1\}} \texttt{csat}_z(\Sigma^{(z)}) =$$
$$\sum_{z:W\to\{0,1\}} \texttt{csat}_z(\Sigma_1^{(z)}) \cdot \texttt{csat}_z(\Sigma_2^{(z)})$$

□

4.1.2 H contains only clauses

Let $\Delta = \{D_1, \ldots, D_m\}$ be the clauses in H. We write each of those as $D_i^1 \vee D_i^2$ with D_i^j containing only variables from G_j. Again, for $i = 1, 2$, let Σ_i be the set of clauses with $I(\Sigma_i) = G_i$.

Lemma 4.4. *Let $m = 1$. Then*

$$\begin{gathered}\texttt{csat}(\Sigma) = \\ \texttt{csat}(\Sigma_1 - \{D_1^1\}) \cdot \texttt{csat}(\Sigma_2) + \texttt{csat}(\Sigma_1) \cdot \texttt{csat}(\Sigma_2 - \{D_1^2\}) \\ -\texttt{csat}(\Sigma_1) \cdot \texttt{csat}(\Sigma_2)\end{gathered}$$

Proof. Straightforward from the inclusion and exclusion principle. □

The case $m \geq 2$ is based on the inclusion/exclusion principle. We need some notation. Let $[m] = \{1, \ldots, m\}$. For $X \subseteq [m]$ and $i = 1, 2$ denote by $S_i(X) = \{D_j^i : j \in X\}$.

Lemma 4.5. *The tree-width of $\Sigma_i - S_i(X)$ is not bigger than the tree-width of Σ_i.*

□

One now proves by induction, using Lemma 4.4 as the basis:

Lemma 4.6. *With the notation from above and*

$$B_i(X_i) = \texttt{csat}((\Sigma_i - S_i(X_i)))$$

we have

$$\begin{gathered}\texttt{csat}(\Sigma) = \texttt{csat}(\Sigma_1 \oplus_\Delta \Sigma_2) = \\ \sum_{k=1}^{m} (-1)^k \sum_{|X_1 \cap X_2| = k} B_1(X_1) \cdot B_2(X_2)\end{gathered}$$

□

4.1.3 The mixed case

For the mixed case we assume that H is a signed bipartite graph with W as its variable nodes and Δ as its clause nodes.

Theorem 4.7. *With the notation from above, let $\Sigma = \Sigma_1 \oplus_{W,\Delta} \Sigma_2$ and*

$$B_i(X_i)^{(z)} = \texttt{csat}((\Sigma_i - S_i(X_i))^{(z)}).$$

Then

$$\begin{gathered}\texttt{csat}(\Sigma) = \texttt{csat}(\Sigma_1 \oplus_{W,\Delta} \Sigma_2) = \\ \sum_{z:W \to \{0,1\}} \sum_{k=1}^{m} (-1)^k \sum_{|X_1 \cap X_2| = k} B_1(X_1)^{(z)} \cdot B_2(X_2)^{(z)}\end{gathered}$$

Hence, computing Σ needs at most $2^{|W|} \cdot 4^m \leq 4^{tw(I(\Sigma))}$ additions and an equal number of multiplications.

Proof. Apply the inclusion/exclusion principle to $B_i(X_i)^{(z)}$. $\square$

5 Proofs of Theorems 1.3 and 1.5

5.1 Proof of Theorem 1.3

> **Theorem 1.3:** Given a k-tree decomposition of $I(\Sigma)$, $\texttt{csat}(\Sigma)$ can be computed, and hence also SAT can be solved, if restricted to Σ with $I(\Sigma)$ of tree-width at most k, using $4^k \cdot n$ arithmetic operations (or in time $4^k(n + n^2 \cdot \log_2(n))$ if bit cost is applied).

Proof. We use a dynamic programming approach. We start from the leaves. Let n be the number of nodes of G_Σ. Using the k-tree decomposition of G_Σ and the induced subgraphs G_t we know how to reconstruct G_Σ, starting with small graphs (of size at most $k+1$) and then using H-sums where H is of size at most k. In each step where an H-sum is performed we use Theorem 4.7. For this we have to compute $2^{|W|} \cdot 4^{|\Delta|} \leq 4^k$ many times products of $\texttt{csat}((\Sigma_i - S_i(X_i))^{(z)})$, where $(\Sigma_i - S_i(X_i))^{(z)}$ has again tree-width at most k. This uses at most $4^k \cdot n$ additions and multiplications over $\mathbb{Z}$. As the number of assignments is bound by 2^n the bit size of the numbers involved is at most n. Multiplication of n-bit numbers uses no more than $n \cdot \log_2(n)$ bit-operations. Hence we get an algorithm which runs in time $4^k(n + n^2 \cdot \log_2(n))$ on a Turing machine [7]. $\square$

5.2 Proof of Proposition 1.4

> **Proposition 1.4:** For every instance Σ for GENSAT(S)
> (i) $tw(P(\Sigma^{exp})) = tw(P(\Sigma))$.
> (ii) If the arities $\rho(i)$ of the clauses in S are bounded by m, $tw(I(\Sigma^{exp})) \leq tw(I(\Sigma)) \cdot m$.

Proof. For an S-formula Σ we define a set of non generalized clauses Σ^{exp} as follows: Let $R_i \in S$ and $r_i(\bar{v})$ be a corresponding generalized clause. Denote by $\bar{r}_i(\bar{v})$ the formula in conjuctive normal form representing R_i with the appropriate variables. Then

$$\Sigma^{exp} = \{\bar{r}_i(\bar{v}) : r_i(\bar{v}) \in \Sigma\}.$$

[7] A closer computation actually gives $3^{k+1}(n + n^2 \cdot \log_2(n))$

To prove (i), we observe that $P(\Sigma^{exp})$ is the same graph $P(\Sigma)$.

To prove (ii), we show how given a k-tree for $I(\Sigma)$ we can construct an mk-tree for $I(\Sigma^{exp})$. We go over the k-tree of $I(\Sigma)$, and in the first stage in every set A_t of the tree we replace every clause vertex with all its incident variable vertices.

At this stage, the new tree clearly has a bound of mk on its set sizes, and still satisfies the connectivity condition for every variable vertex (the tree does not contain any clause vertex at this stage) It is also easy to see that for every every clause of Σ, and hence of Σ^{exp}, there is a set A_t of the tree containing all of its incident variables (just take any set that in the original tree contained the appropriate clause vertex).

We finish the construction by adding a new leaf for every clause of Σ^{exp} with a set that contains the appropriate clause vertex and all incident variable vertices, connecting this leaf to the appropriate A_t that contains all variable vertices. $\square$

5.3 Proof of Theorem 1.5

Theorem 1.5: Given a k_1-tree decomposition of $P(\Sigma)$, a k_2-tree decomposition of $I(\Sigma)$, let $m = \max_i\{\rho(i)\}$ (if it exists), and $k_3 = \max_i\{\rho(i), k_2\}$. Then $\sharp$GENSAT(S) can be computed

(i) with $4^{k_1} \cdot n^2$ arithmetic operations, provided the size of each clause is bounded by $\log_2(n)$;

(ii) with $4^{k_1+m} \cdot n$ arithmetic operations, provided the size of each clause is bounded by $m \in \mathbb{N}$;

(iii) with $4^{k_3 \cdot m} \cdot n$ arithmetic operations, provided the size of each clause is bounded by $m \in \mathbb{N}$.

Proof. Instead of solving $\sharp$GENSAT(S) with input Σ we reduce it to computing `csat`(Σ^{exp}). According to Proposition 1.4 the reduction does not increase the tree-width of the primal graph. It also increases the tree-width of the incidence graph by at most m, provided that every $\rho(i)$ is bounded by m. Hence we only have to make sure that the size of Σ^{exp} is bounded. But in Σ^{exp} each clause C of Σ with $\rho(i)$ many variables is replaced by at most $2^{\rho(i)}$ many clauses of size at most $\rho(i)$.

The remaining computations for the estimates in (i)-(iv) are left to the reader. $\square$

6 The case of bounded clique-width

6.1 Background on clique-width

The notion of clique-width was introduced in [CER93] and studied more systematically in [CM93,CE95,Cou92,EvO97,CO00,GR00]. In the last ten years, the study of graphs of bounded clique-width became very popular, cf. the work of A. Brandstaedt, B. Courcelle, V.V. Lozin, P. Seymour, J. Spinrad, and their many collaborators. Courcelle and Olariu in [CO00] showed that clique-width of graphs of tree-width k, is at most $2^{k+1}+1$. Therefore, any class of graphs of bounded tree-width is automatically of bounded clique-width. Moreover, B. Courcelle, J. Engelfriet and G. Rozenberg in [CER93] provided a complicated proof that any given context-free graph grammar based on vertex-replacement (Confluent NCE, or context-free VR grammar) generates graphs of bounded clique-width. Although an upper bound for the clique-width could be derived from their proof, it is not straightforward. In general, finding an explicit bound for the clique-width is a more complicated task than finding a bound for the tree-width. For explicit computations of clique-width, cf. [GR00], [GM03].

Courcelle and Olariu in [CO00] study two versions of clique-width, for undirected and for directed graphs. We give here a version for directed or signed graphs where additionally the bipartite character of the graphs is taken into account. We identify a SAT formula Σ with the bipartite graph $SI(\Sigma)$ that has edges 'signed' with '+' and '−' according to which variables appear in a clause and whether they are negated. If we drop the signing of the edges, we just get $I(\Sigma)$.

Definition 6.1. *The set of* SAT formulas of clique width up to k *is defined as the set of formulas that can be obtained by the following operations over such graphs whose vertices are coloured by* $\{1,\dots,k\}$*, starting with singletons (formulas consisting of a single "clause" or "variable" vertex with some colour from* $\{1,\dots,k\}$ *and no edges).*

- *(i) Disjoint union.*
- *(ii) Recolouring: For a vertex-coloured edge-signed bipartite graph* I*, we define* $\rho_{i,j}(I)$ *to be the graph that results by recolouring with* j *all vertices that were previously coloured with* i*.*
- *(iii) Positive edge creation: For a vertex-coloured edge-signed bipartite graph* I*, we define* $\eta^{+}_{i,j}(I)$ *to be the graph that results from connecting all clause-vertices coloured with* i *to all variable-vertices coloured with* j*, with edges signed by '+'. We do not add edges between variable-vertices coloured* i *and clause-vertices coloured* j*, or any other vertices.*
- *(iv) Negative edge creation: Similarly to the above, we define* $\eta^{-}_{i,j}(I)$ *to be the graph resulting from connecting all clause-vertices coloured with* i *to all*

variable-vertices coloured with j, with edges signed by '$-$'.

(v) *In the case of unsigned edges, and without distinguishing clause-vertices and variable-vertices, there is just one operation $\eta_{i,j}$ for each $i \neq j$. This corresponds to the original definition in [CO00].*

(vi) *The clique-width of a (signed, bipartite) graph is the minimum k such that it has clique-width at most k. We denote by $scw(SI(\Sigma))$ the signed bipartite clique-width of $SI(\Sigma)$ and by $cw(I(\Sigma))$ the unsigned clique-width of $I(\Sigma)$.*

A *parse tree der_{SI} for the signed clique-width* of a formula Σ is just the rooted tree whose leaves hold singleton graphs, whose internal vertices are coloured with the operations of the definitions above (so a vertex corresponding to a disjoint union has two children, and vertices corresponding to other operations have one child), and whose root holds the graph $SI(\Sigma)$ (with any vertex colouring). A *parse tree der_I for the clique-width* of a formula Σ is defined similarly for the case of the unsigned graph $I(\Sigma)$.

Every graph G of size n has clique-width $cw(G)$ at most n. The simplest class of graphs of unbounded tree-width but of clique-width at most 2 are the cliques. To see this assume we have two colours red (1) and blue (2). We start with a red singleton and a blue singleton and connect using $\eta_{1,2}$, then we recolour all points red, add a new blue singlton and connect again using $\eta_{1,2}$, and so forth.

Given a graph G and $k \in \mathbb{N}$, determining whether G has clique-width k is in **NP**. A polynomial time algorithm was presented for $k \leq 3$ in [CHL$^+$00]. It remains open whether for some fixed $k \geq 4$ the problem is **NP**-complete. The recogniztion problem for the analogue of clique-width for relational structures, cf. [BC0x], has not been studied so far even for $k = 2$. However, once a parse tree is known the number of satisfying assignments can be efficiently calculated.

However, for our purposes, a recent result of S. Oum and P. Seymour, [OS04], suffices to apply Theorem 1.8. They have shown that testing a graph for clique-width k is fixed parameter tractable, and an approximate parse tree can be be produced in polynomial time in n.

Theorem 6.2 (S. Oum and P. Seymour). *There is a function f, such that, for given k, there is a polynomial time algorithm that, with input a graph G, either concludes that its clique-width is $> k$ or outputs a $f(k)$-parse tree for G. Its running time is $O(n^9 \log n)$ and $f(k) = 2^{3k+2} - 1$.*

By straight inspection of their proof a similar theorem can be proven also for the clique-width of signed graphs.

Theorem 6.3. *There is a function g, such that, for a given k, there is a polynomial time algorithm that, with input a signed graph G, either concludes* *that its signed clique-width is larger that k or outputs a $g(k)$-parse tree for G. Its running time is $O(n^9 \log n)$ and $g(k) = 3^{3k+O(1)} = 2^{O(k)}$.*

Using the parse tree obtained from Theorem 6.3, we can produce $g(k)$-parse trees for signed graphs with clique-width k, which makes our results applicable.

In [MM03b] the following is shown for undirected clique-width, but the same proof gives it also for directed clique-width. To estimate the clique width this is often useful.

Proposition 6.4. *Clique-width is preserved for induced subgraphs. More precisely, if G is a a (undirected, signed, directed) graph, and H is an induced (undirected, signed, directed) subgraph of G, then we have*

$$cw(H) \leq cw(G),$$

respectively

$$scw(H) \leq scw(G).$$

6.2 Clique-width of clause graphs

We noted already that for the unsigned clique-width it is shown in [CO00] that clique-width of graphs of tree-width k, is at most $2^{k+1}+1$. Hence we have

Proposition 6.5. *Let Σ be a set of clauses. Then we have*

(i) $cw(P(\Sigma)) \leq 2^{tw(P(\Sigma))+1}+1$ *and*
(ii) $cw(I(\Sigma)) \leq 2^{tw(I(\Sigma))+1}+1$.

However, a bound on the clique-width of $P(\Sigma)$ gives no computational advantage.

Proposition 6.6. *Let* SAT($cw2$) *be* SAT *restricted to sets of clauses Σ with $cw(P(\Sigma)) = 2$.*

(i) SAT($cw2$) *is* **NP**-*complete.*
(ii) $\sharp SAT(cw2)$ *is* $\sharp$**P**-*complete.*

This follows immediately from:

Lemma 6.7. *For every set of clauses Σ in n variables $v_1, \ldots, v_n$ we define a set of clauses Σ' in $n+1$ variables $v_0, v_1, \ldots, v_n$ by*

$$\Sigma' = \Sigma \cup \{v_0\} \cup \{v_i \vee v_j \vee v_0 : i, j \geq 1, i \neq j\}.$$

For an assignement z for the variables $v_1, \ldots, v_n$ we define the assignment z' for $v_0, v_1, \ldots, v_n$ by setting $z(v_0) = 1$. Then we have

(i) z makes Σ true iff z' makes Σ' true.
(ii) $P(\Sigma')$ is a clique, hence $cw(P(\Sigma')) = 2$.

Next we compare the clique-width of the signed and the unsigned cases:
Proposition 6.8.

$$cw(I(\Sigma)) \leq 2 \cdot scw(SI(\Sigma))$$

Sketch of proof. We take a parse tree der_{SI} for $SI(\Sigma)$. By doubling the number of colours (separating clause-vertices from variable-vertices we get for each colour i two colours i_c and i_v) we can disregard the bipartite character of the graphs. For this we replace each operation $\eta_{i,j}$ by η_{i_c,j_v}. The resulting parse tree is a parse tree for $I(\Sigma)$, where all the operation $\eta_{i,j}$ have different indices i, j. □

Let G be any graph (not necessarily a clause graph of Σ). The incidence graph $I(G) = (V \cup E, F)$ of a graph (V, E) is the bipartite graph with V and E as vertex sets, and $(v, e) \in F$ iff v is a vertex of e. Clique-width and tree-width behave quite differenty, when passing from G to $I(G)$.
Proposition 6.9.

(Folklore) *For every graph, $tw(G) = tw(I(G))$.*
([MR99]) *$cw(K_n) = 2$, but $cw(I(K_n))$ goes to infinity with n.*

A converse inequalty to the one in Proposition 6.8 does not hold.
Proposition 6.10. *For every m there is a set of clauses Σ_m such that*

(i) $I(\Sigma_m) = K_{m,n}$, the complete bipartite graph on m and n elements with $n = \binom{m}{2}$. Hence $cw(I(\Sigma_m)) = 2$;
(ii) The clique width $scw(SI(\Sigma_m))$ is a function of m which tends to infinity with m.

Sketch of proof. Let the variables be $v_1, \ldots, v_m$. For each $i \neq j \leq m$ let $C_{i,j}$ be the clause containing all the variables, but where exactly v_i and v_j occur negatively. Σ_m is the set of such clauses. Clearly, $I(\Sigma_m) = K_{m,n}$, the complete bipartite graph on m and n elements with $n = \binom{m}{2}$. So (i) is established. To see (ii), assume $der_{SI}(m)$ is a parse tree for (Σ_m). We omit each η^+ in der_{SI} to obtain a parse tree $der_I(m)$. But $der_I(m)$ is a parse tree for $I(K_m)$, which is unbounded by Proposition 6.9. Note that here we use Proposition 6.4. □

6.3 Clique-width of pigeon-hole and Tseitin formulas

We return to the examples of subsection 3.3. First we quote from [GR00]

Proposition 6.11. *The clique-width of the grid graphs $Grid_{n,n}$ is at least n.*

From this, together with Proposition 6.4, the following is not difficult to show.
Proposition 6.12. *The undirected, and hence the directed clique-width of the pigeon-hole formulas and the Tseitin formulas is unbounded.*

6.4 Main result for bounded clique-width

We restate from the introduction

> **Theorem 1.8**: Given a set of clauses Σ and a signed parse tree $der_{SI}(\Sigma)$ for clique-width of up to k, it is possible to calculate $\mathtt{csat}(\Sigma)$, with a number of algebraic operations that is linear in the size of the parse tree $der_{SI}(\Sigma)$, and exponential in k.

Remark 6.13.

(i) *The corresponding theorem for unsigned clique-width seems to be true as well, but the proof may be more involved and we did not check it in detail.*
(ii) *Although bounded tree-width of a class of graphs implies bounded clique-width of the same class, cf. Proposition 6.5, the clique-width grows exponentially. Therefore, Theorem 1.8 does not imply Theorem 1.3, even if the unsigned version of Theorem 1.8 is true.*

The proof is given in Subsection 6.5. We leave it to the reader to formulate and prove the corresponding theorem for GENSAT.

Before we continue, we define some possible transformations of formulas corresponding to vertex-coloured edge-signed bipartite graphs.
Definition 6.14. *Given subsets A, B, C of $\{1, \dots, k\}$ (not necessarily disjoint), and a formula Σ whose signed graph $SI(\Sigma)$ is vertex-coloured with $\{1, \dots, k\}$, we define $\Sigma^{(A,B,C)}$ as the formula resulting from Σ by the following operations:*

(i) *Every clause in Σ whose vertex is coloured with a member of A is removed (but we do nothing with variables whose vertices are coloured with members of A).*
(ii) *For $i \in \{1, \dots, k\}$, denote by X_i the set of variables whose vertices are coloured with i. For every $i \in B$ we add a clause consisting of the disjunction of all the variables in X_i.*
(iii) *For every $i \in C$ we add a clause consisting of the disjunction of all the negations of the variables in X_i.*

Note that in particular $\Sigma = \Sigma^{(\emptyset,\emptyset,\emptyset)}$.

We assume w.l.o.g. that in the parse tree $der_{SI}(\Sigma)$ of Σ, all unions are made between graphs that use disjoint subsets of the colour set in their vertex colouring. The reason for this is that given a parse tree that does not satisfy this condition and uses k vertex colours in all, one can easily construct a parse tree with $2k$ vertex colours for which this additional condition holds.

To prove Theorem 1.8, we calculate for every node v of the parse tree $der_{SI}(\Sigma)$ not only the value $\texttt{csat}(\Sigma_v)$ for the formula Σ_v constructed by the operation of that node, but we also calculate $\texttt{csat}(\Sigma_v^{(A,B,C)})$ for every $A, B, C \subseteq \{1, \dots, k\}$, which are used in the reductions through which we obtain the final $\texttt{csat}(\Sigma)$. We use the following reduction lemmas.

Lemma 6.15. *If the operation in node v is a disjoint union of its children u and w, then*

$$\texttt{csat}(\Sigma_v^{(A,B,C)}) = \texttt{csat}(\Sigma_u^{(A,B,C)}) \cdot \texttt{csat}(\Sigma_w^{(A,B,C)})$$

for every A, B, C.

Proof. We assumed above that in all disjoint unions, the colour sets used by the two children are also disjoint, and under this assumption it is not hard to see that the above holds. □

Lemma 6.16. *If the operation in v is $\rho_{i,j}(\Sigma_w)$ where w is the child of v, for every A, B, C it is possible to calculate $\texttt{csat}(\Sigma_v^{(A,B,C)})$ from the values stored for w using a constant number of operations.*

Proof. If $i \in B$ or $i \in C$ then $\texttt{csat}(\Sigma_v^{(A,B,C)}) = 0$, because Σ_v contains no variables coloured with i and hence $\Sigma_v^{(A,B,C)}$ contains an empty (unsatisfiable) clause. From now on we assume that B and C do not contain i. If $j \in A$ we set $A' = A \cup \{i\}$, and otherwise we set $A' = A \setminus \{i\}$. We now distinguish four cases:

Case 1: B and C do not contain j.
In this case clearly

$$\texttt{csat}(\Sigma_v^{(A,B,C)}) = \texttt{csat}(\Sigma_w^{(A',B,C)}).$$

Case 2: B contains j but C does not.
In this case we use the inclusion-exclusion principle.
We set $B_1 = B \cup \{i\} \setminus \{j\}$, $B_2 = B$, and $B_3 = B \cup \{i\}$, and obtain

$$\texttt{csat}(\Sigma_v^{(A,B,C)}) = \texttt{csat}(\Sigma_w^{(A',B_1,C)}) + \texttt{csat}(\Sigma_w^{(A',B_2,C)}) - \texttt{csat}(\Sigma_w^{(A',B_3,C)}).$$

Case 3: C contains j but B does not.
This is analogous to the previous case. In this case we set $C_1 = C \cup \{i\} \setminus \{j\}$, $C_2 = C$, and $C_3 = C \cup \{i\}$, and obtain

$$\mathtt{csat}(\Sigma_v^{(A,B,C)}) = \mathtt{csat}(\Sigma_w^{(A',B,C_1)}) + \mathtt{csat}(\Sigma_w^{(A',B,C_2)}) - \mathtt{csat}(\Sigma_w^{(A',B,C_3)}).$$

Case 4: Both B and C contain j.
We define $B_1, B_2, B_3, C_1, C_2, C_3$ as above and use again the inclusion-exclusion principle, but this time the resulting formula is somewhat more complex:

$$\begin{gathered}\mathtt{csat}(\Sigma_v^{(A,B,C)}) = \\ \mathtt{csat}(\Sigma_w^{(A',B_1,C_1)}) + \mathtt{csat}(\Sigma_w^{(A',B_1,C_2)}) \\ +\mathtt{csat}(\Sigma_w^{(A',B_2,C_1)}) + \mathtt{csat}(\Sigma_w^{(A',B_2,C_2)}) \\ -\mathtt{csat}(\Sigma_w^{(A',B_3,C_1)}) - \mathtt{csat}(\Sigma_w^{(A',B_3,C_2)}) \\ -\mathtt{csat}(\Sigma_w^{(A',B_1,C_3)}) - \mathtt{csat}(\Sigma_w^{(A',B_2,C_3)}) \\ +\mathtt{csat}(\Sigma_w^{(A',B_3,C_3)}).\end{gathered}$$

□

Lemma 6.17. *If the operation in v is $\eta^+_{i,j}(\Sigma_w)$ where w is the child of v, for every A, B, C it is possible to calculate $\mathtt{csat}(\Sigma_v^{(A,B,C)})$ from the values stored for w using a constant number of operations.*

Proof. If $i \in A$ then clearly

$$\mathtt{csat}(\Sigma_v^{(A,B,C)}) = \mathtt{csat}(\Sigma_w^{(A,B,C)}),$$

and if $j \in B$ then clearly

$$\mathtt{csat}(\Sigma_v^{(A,B,C)}) = \mathtt{csat}(\Sigma_w^{(A\cup\{i\},B,C)}).$$

Otherwise we note that a satisfying assignment for $\Sigma_v^{(A,B,C)}$ is an assignment that satisfies $\Sigma_w^{(A,B,C)}$ or $\Sigma_w^{(A\cup\{i\},B\cup\{j\},C)}$, and we use the inclusion-exclusion principle to obtain

$$\mathtt{csat}(\Sigma_v^{(A,B,C)}) = \mathtt{csat}(\Sigma_w^{(A,B,C)}) + \mathtt{csat}(\Sigma_w^{(A\cup\{i\},B\cup\{j\},C)}) - \mathtt{csat}(\Sigma_w^{(A,B\cup\{j\},C)}).$$

□

Lemma 6.18. *If the operation in v is $\eta^-_{i,j}(\Sigma_w)$ where w is the child of v, for every A, B, C it is possible to calculate $\mathtt{csat}(\Sigma_v^{(A,B,C)})$ from the values stored for w using a constant number of operations.*

Proof. Virtually identical to that of the previous lemma. □

Proof of Theorem 1.8. We start with the leaves and go upward, at every node v calculating $\mathtt{csat}(\Sigma_v^{(A,B,C)})$ for all possible A, B, C. For every node the total number of calculated values is exponential in k, and the number of operations to calculate each of them is constant, and so the number of operations required to reach the root is linear in the size of the tree and exponential in k. $\square$

7 Conclusions and further research

We have presented evidence from the literature that sets of clauses Σ with clause graphs of bounded tree-width or clique-width can be derived from real-world applications. Small tree-width and small clique-width are structural properties of the various clause graphs. Engineering artifacts come with built in modularization with minimal or well structured interconnectivity which imply these structural properties, cf. [Ami01].

We have shown how to solve SAT, GENSAT, ♯SAT and ♯GENSAT efficiently on sets of clauses with incidence graphs of tree-width at most k. Our new algorithm has feasible constants, when k is not too large. It also allows us to solve SAT efficiently, but it remains to be checked whether it is more efficient than the resolution method, applied to formulas of bounded tree-width as presented in [AR02].

We have also shown how to use parse trees of signed clique-width efficiently to solve SAT and ♯SAT. This widens the applicability of our results considerably, especially, since S. Oum and P. Seymour have shown that finding a suitable parse tree for (signed) graphs is fixed parameter tractable (in FPT).

Our methods apply also to any other problem which is reducible to SAT by polynomial time Turing reductions where the tree-width or clique-width is bounded. We have shown how to use this for various versions of GENSAT.

It would be interesting to see, for which versions of GENSAT there are splitting formulas similar to the one given in Theorem 4.7. Such splitting formulas are bound to give better constants than the ones one gets by using reductions.

The results of [CMR01,Mak04] give general splitting theorems and polynomial time algorithms for many other counting problems. It remains a challenge to find direct proofs for simpler splitting formulas, say, for counting perfect matchings, hamiltonian cycles or various colourings.

Acknowledgements

We thank A. Magid and Y. Magid, the first author's M.Sc. students, for checking our first proof of Theorem 4.7, A. Glikson for Figure 1, H. Bodlaender and J. Naor for guidence to the literature. We also thank L. Traldi, whose [Tra] he made available to us before publication and which inspired us to write this paper. Finally we thank three anonymous referees for very valuable suggestions.

References

[ACP87] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embedding in a k–tree. *SIAM. J. Algebraic Discrete Methods*, 8:277–284, 1987.

[AD03] A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. In *Proceedings of the 18th IEEE Conference on Computational Complexity (CCC 2003), Aarhus (Denmark)*, pages 239–247, 2003.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[AM01] E. Amir and S. McIlraith. Theorem proving with structured theories. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 624–631, 2001.

[AM05] E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 162(1-2):49–88, 2005.

[Ami00] E. Amir. (de)Composition of situation calculus theories. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'00)*, pages 456–463. AAAI Press/MIT Press, 2000.

[Ami01] E. Amir. *Dividing and Conquering Logic*. PhD thesis, Department of Computer Science, Stanford University, 2001.

[Ami02] E. Amir. Approximation algorithms for treewidth. *submitted*, xx:xx–yy, 2002.

[And97] A. Andrzejak. Splitting formulas for Tutte polynomials. *Journal of Combinatorial Theory, Series B*, 70.2:346–366, 1997.

[AR02] M. Alekhnovich and A. Razborov. Satisfiability, branch-width and Tseitin tautologies. In *Proceedings of the 43th Annual Symposium on the Foundations of Computer Science (FOCS 2002), Vancouver, British Columbia, Canada*, pages 593–603. IEEE Computer Society Press, 2002.

[BKDSZ] P. Bjesse, J. Kukula, R. Damiano, T. Stanion and Y. Zhu. Guiding SAT diagnosis with tree decompositions. In *Sixth International Conference on Theory and Applications of Satisfiability Testing, (SAT'03) S. Margherita Ligure (Italy)*, Lecture Notes in Computer Science, pages 315–329. Springer, 2003.

[BC0x] A. Blumensath and B. Courcelle. Recognizability and hypergraph operations using local information. *Information and Computation*, to appear 200x.

[Bod97] H. Bodlaender. Treewidth: Algorithmic techniques and results. In I. Privara and P. Ruzicka, editors, *Proceedings of the 22th International Symposium on the Mathematical Foundation of Computer Science, MFCS'97*, volume 1295 of *Lecture Notes in Computer Science*, pages 29–36. Springer, 1997.

[Bol99] B. Bollobás. *Modern Graph Theory*. Springer, 1999.

[CE95] B. Courcelle and J. Engelfriet. A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars. *Mathematical Systems Theory*, 28:515–552, 1995.

[CER93] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. System Sci.*, 46:218–270, 1993.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

[CHL+00] D. G. Corneil, M. Habib, J.-M. Lanlignel, B. Reed, and U. Rotics. Polynomial time recognition of clique-width ≤ 3 graphs. In *Proceedings of LATIN'2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 126–134. Springer, 2000.

[CKS01] N. Creignou, S. Khanna, and M. Sudhan. *Complexity classifications of boolean constraint satisfaction roblems*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 2001.

[CM93] B. Courcelle and M. Mosbah. Monadic second–order evaluations on tree–decomposable graphs. *Theoretical Computer Science*, 109:49–82, 1993.

[CM02] B. Courcelle and J.A. Makowsky. Fusion on relational structures and the verification of monadic second order properties. *Mathematical Structures in Computer Science*, 12.2:203–235, 2002.

[CMR01] B. Courcelle, J.A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second order logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[CO00] B. Courcelle and S. Olariu. Upper bounds to the clique–width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000.

[Cou92] B. Courcelle. Monadic second-order logic of graphs VII: Graphs as relational structures. *Theoretical Computer Science*, 101:3–33, 1992.

[Cou03] B. Courcelle. The monadic second order logic of graphs, XIV: Uniformly sparse graphs and edge set quantification. *Theoretical Computer Science*, 299(1):1–36, 2003.

[CS88] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[CSJ+98] P. Cohen, P. Schrag, R. JOnes, E. Pease, A. Lin, A. Starr, B. Gunning, and M. Burke. The DARPA high-performance knowledge base project. *AI Magazine*, 19(4):25–49, 1998.

[DF99] R.G. Downey and M.F Fellows. *Parametrized Complexity*. Springer, 1999.

[Die96] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 1996.

[DP89] R. Dechter and J. Pearl. The clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[EvO97] J. Engelfriet and V. van Oostrom. Logical description of context-free graph-languages. *Journal of Computer and System Sciences*, 55:489–503, 1997.

[FG04] M. Frick and M Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1):3–31, 2004.

[FV99] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999. (Conference version in *STOC'93*, pages 612–622. ACM, 1993.)

[Gal77] Z. Galil. On the complexity of regular resolution and the Davis–Putnam procedure. *Theoretical Computer Science*, 4:23–46, 1977.

[GJ79] M.G. Garey and D.S. Johnson. *Computers and Intractability*. Mathematical Series. W.H. Freeman and Company, 1979.

[GM03] A. Glikson and J.A. Makowsky. NCE graph grammars and clique-width. In H.L. Bodlaender, editor, *Proceedings of the 29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2003), Elspeet (The Netherlands)*, volume 2880 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 2003.

[GP01] G. Gottlob and R. Pichler. Hypergraphs in model checking: Acyclicity and hypertree-width versus clique-width. In *28th International Colloquium on Automata, Languages and Programming, ICALP'01*, volume 2077 of *Lecture Notes in Computer Science*, pages 708–719, 2001.

[GR00] M. C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *Internation Journal of Foundations of Computer Science*, 11:423–443, 2000.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[IM87] A. Itai and J.A. Makowsky. Unification as a complexity measure for logic programming. *Journal of Logic Programming*, 4(2):105–117, 1987.

[JYP88] D.S. Johnson, M. Yannakakis, and C.H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[Len95] D.B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Mak01] J.A. Makowsky. Colored Tutte polynomials and Kauffman brackets on graphs of bounded tree width. In *Proceedings of the 12th Symposium on Discrete Algorithms*, pages 487–495. SIAM, 2001.

[Mak04] J.A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126:1–3, 2004.

[Mak05] J.A. Makowsky. Colored Tutte polynomials and Kauffman brackets on graphs of bounded tree width. *Disc. Appl. Math.*, 145(2):276–290, 2005.

[MM03a] J.A. Makowsky and J.P. Mariño. Farrell polynomials on graphs of bounded treewidth. *Advances in Applied Mathematics*, 30:160–176, 2003.

[MM03b] J.A. Makowsky and J.P. Mariño. Tree-width and the monadic quantifier hierarchy. *Theoretical Computer Science*, 303:157–170, 2003.

[MR99] J.A. Makowsky and U. Rotics. On the cliquewidth of graphs with few P_4's. *International Journal on Foundations of Computer Science*, 10:329–348, 1999.

[Neg87] S. Negami. Polynomial invariants of graphs. *Transaction of the American Mathematical Society*, 299:601–622, 1987.

[Nob98] S.D. Noble. Evaluating the Tutte polynomial for graphs of bounded tree-width. *Combinatorics, Probability and Computing*, 7:307–321, 1998.

[OS04] S. Oum and P. Seymour. Approximating clique-width and branch-width. *submitted,*, 2004.

[OW92] J.G. Oxley and D.J.A. Welsh. Tutte polynomials computable in polynomial time. *Discrete Mathematics*, 109:185–192, 1992.

[Pap94] C. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th Symposium on Theory of Computing (STOC'78), San Diego (California, USA)*, pages 216–226, 1978.

[Sze03] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *Sixth International Conference on Theory and Applications of Satisfiability Testing, (SAT'03) S. Margherita Ligure (Italy)*, Lecture Notes in Computer Science, pages 188–202. Springer, 2003.

[Tho98] M. Thorup. Structured programs have small tree-width and good register allocation. *Information and Computation*, 142:159–181, 1998.

[Tra] L. Traldi. A note on the colored Tutte polynomial of a graph of bounded tree width. Preprint, January 2003.

[Tse68] G.S. Tseitin. On the complexity of derivation in propositional calculus. In A.O. Slisenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II.*, pages 115–125. Translation from Russian, 1968.

[Val79] L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[WCZK] D. Wang, E. Clarke, Y. Zhu and J. Kukula. Using Cutwidth to improve symbolic simulation and boolean satisfiability. To be completed. In *Proceedings of the Sixth IEEE International High-Level Design Validation and Test Workshop (HLDVT'01)*, pages 165–170. IEEE Computer Society, 2003.