From Width-Based Model Checking to Width-Based Automated Theorem Proving

Mateus de Oliveira Oliveira[1,2], **Farhad Vadiee**[2]

[1] Department of Computer and System Sciences, Stockholm University, Stockholm, Sweden
[2] Department of Informatics, University of Bergen, Bergen, Norway
oliveira@dsv.su.se, farhad.vadiee@uib.no

Abstract

In the field of parameterized complexity theory, the study of graph width measures has been intimately connected with the development of width-based model checking algorithms for combinatorial properties on graphs. In this work, we introduce a general framework to convert a large class of width-based model-checking algorithms into algorithms that can be used to test the validity of graph-theoretic conjectures on classes of graphs of bounded width. Our framework is modular and can be applied with respect to several well-studied width measures for graphs, including treewidth and cliquewidth. As a quantitative application of our framework, we prove analytically that for several long-standing graph-theoretic conjectures, there exists an algorithm that takes a number k as input and correctly determines in time double-exponential in $k^{O(1)}$ whether the conjecture is valid on all graphs of treewidth at most k. These upper bounds, which may be regarded as upper-bounds on the size of proofs/disproofs for these conjectures on the class of graphs of treewidth at most k, improve significantly on theoretical upper bounds obtained using previously available techniques.

1 Introduction

1.1 Motivation

When mathematicians are not able to solve a conjecture about a given class of mathematical objects, it is natural to try to test the validity of the conjecture on a smaller, or better behaved class of objects. In the realm of graph theory, a common approach is to try analyze the conjecture on restricted classes of graphs, defined by fixing some structural parameter. In this work, we push forward this approach from a computational perspective by focusing on parameters derived from graph width measures. Prominent examples of such parameters are the *treewidth* of a graph, which intuitively quantifies how much a graph is similar to a tree (Robertson and Seymour 1984; Bertele and Brioschi 1973; Halin 1976) and the cliquewidth of a graph, which intuitively quantifies how much a graph is similar to a clique (Courcelle and Olariu 2000). More specifically, we are concerned with the following problem:

Problem 1 (Width-Based ATP)**.** *Given a graph property $\mathbb{P}$ and a positive integer k, is it the case that every graph of width at most k belongs to $\mathbb{P}$?*

Problem 1 provides a width-based approach to the field of automated theorem proving (ATP). For instance, consider Tutte's celebrated 5-flow conjecture (Tutte 1954), which states that every bridgeless graph has a nowhere-zero 5-flow. Let $\texttt{HasBridge}$ be the set of all graphs that have a bridge, and $\texttt{NZFlow}(5)$ be the set of all graphs that admit a nowhere-zero 5-flow. Then, proving Tutte's 5-flow conjecture is equivalent to showing that every graph belongs to the graph property $\texttt{HasBridge} \vee \texttt{NZFlow}(5)$. Since Tutte's conjecture has been unresolved for many decades, one possible approach for gaining understanding about the conjecture is to try to determine, for gradually increasing values of k, whether every graph of width at most k, with respect to some fixed width-measure, belongs to $\texttt{HasBridge} \vee \texttt{NZFlow}(5)$. What makes this kind of question interesting from a proof theoretic point of view is that several important classes of graphs have small width with respect to some width measure. For instance, trees and forests have treewidth at most 1, series-parallel graphs and outerplanar graphs have treewidth at most 2, k-outerplanar graphs have treewidth at most $3k-1$, co-graphs have cliquewidth at most 2, any distance hereditary graph has cliquewidth 3, etc (Biedl 2015; Bodlaender 1998; Brandstädt, Le, and Spinrad 1999; Bodlaender 1986; Bodlaender and Koster 2008; Kammer 2007). Therefore, proving the validity of a given conjecture on classes of graphs of small width corresponds to proving the conjecture on interesting classes of graphs.

1.2 Our Results

In this work, we introduce a general and modular framework that allows one to convert width-based dynamic programming algorithms for the model checking of graph properties into algorithms that can be used to address Problem 1. More specifically, our main contributions are threefold.

1. We start by defining the notions of a *treelike decomposition class* (Definition 2) and of a *treelike width-measure* (Definition 3). These two notions can be used to express several classic, well studied width measures for graphs, such as treewidth (Bodlaender 1997), pathwidth (Korach and Solel 1993), carving width (Thilikos, Serna,

and Bodlaender 2000), cutwidth (Chung and Seymour 1989; Thilikos, Serna, and Bodlaender 2005), bandwidth (Chung and Seymour 1989), cliquewidth (Courcelle and Olariu 2000), etc, and some more recent measures such as ODD-width (Andrade de Melo and Oliveira Oliveira 2019).

2. Subsequently, we introduce the notion of a *treelike dynamic programming core* (Definition 7), a formalism for the specification of dynamic programming algorithms operating on treelike decompositions. Our formalism combines two points of view for dynamic programming: the *symbolic* point of view (Courcelle and Durand 2016) that allows one to use symbolic tree automata to reason about classes of graphs of bounded width, and the *combinatorial* point of view (Baste et al. 2022) that provides a general methodology for the specification of state-of-the-art combinatorial algorithms for model-checking graph properties on classes of bounded width. Treelike DP-cores satisfying certain *coherency* and *finiteness* properties are suitable for the study of the computational complexity of the problem of determining whether certain graph-theoretic conjectures are valid on the class of graphs of width at most k (for several suitable notions of width). Additionally, our formalism allows one to establish upper bounds on the size of an hypothetical counterexample of width at most k in terms of the computational complexity of algorithms for model-checking the graph properties arising in the statement of the conjecture.
3. Our main result (Theorem 20) states that if a graph property $\mathbb{P}$ is a combination (see Section 5.5) of graph properties $\mathbb{P}_1, \ldots, \mathbb{P}_\ell$ that can be decided by coherent and finite DP-cores $\mathsf{D}_1, \mathsf{D}_2, \ldots, \mathsf{D}_\ell$, then the process of determining whether every graph of width at most k belongs to $\mathbb{P}$ can be decided roughly[1] in time

$$2^{O(\beta(k)\cdot\mu(k))} \leq 2^{2^{O(\beta(k))}},$$

where $\mu(k)$ and $\beta(k)$ are respectively the maximum multiplicity and the maximum bitlength of a DP-core from the list $\mathsf{D}_1, \ldots, \mathsf{D}_\ell$ (see Section 5.2). Additionally, if a counterexample of width at most k exists, then there is a term of height at most $2^{O(\beta(k)\cdot\mu(k))}$ representing such a counterexample.

We illustrate our approach by specializing on graphs of bounded treewidth. Mores specifically, we show that several long-standing conjectures in graph theory can be tested on the class of graphs of treewidth at most k in time double exponential in $k^{O(1)}$. Examples of such conjectures include Hadwiger conjecture (Hadwiger 1943), Tutte's flow conjectures (Tutte 1954) and Barnette's conjecture (Tutte 1969) (Section 6). Our upper bounds improve significantly on upper bounds obtained using previous techniques. For instance, we show that Hadwiger's conjecture for c colors can be verified on the class of graphs of treewidth at most k in time $2^{2^{O(k \log k + c^2)}}$, while the best known upper bound was of the form $p^{p^{p^p}}$ for $p = (k+1)^{c-1}$.

[1]The precise statement of Theorem 20 involves other parameters that are negligible in typical applications.

2 Preliminaries

We let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{N}_+$ denote the set of positive natural numbers. We let $[0] \doteq \emptyset$, and for each $n \in \mathbb{N}_+$, we define $[n] \doteq \{1, ..., n\}$. Given a set S, the set of finite subsets of S is denoted by $\mathcal{P}_{\text{fin}}(S)$.

In this work, a *graph* is a triple $G = (V, E, \rho)$ where $V \subseteq \mathbb{N}$ is a *finite* set of *vertices*, $E \subseteq \mathbb{N}$ is a finite set *edges*, and $\rho \subseteq E \times V$ is an *incidence relation*. For each edge $e \in E$, we let $\text{endpts}(e) = \{v \in V \ : \ (e, v) \in \rho\}$ be the set of vertices incident with e. We assume $|\text{endpts}(e)| = 2$ for all $e \in E$. In what follows, we may write V_G, E_G and ρ_G to denote the sets V, E and ρ respectively. We let $|G| = |V_G| + |E_G|$ be the *size* of G. We let GRAPHS denote the set of all graphs. For us, the *empty graph* is the graph $(\emptyset, \emptyset, \emptyset)$ with no vertices, no edges, and no incidence pairs.

An *isomorphism* from a graph G to a graph H is a pair $\phi = (\phi_1, \phi_2)$ where $\phi_1 : V_G \to V_H$ is a bijection from the vertices of G to the vertices of H and $\phi_2 : E_G \to E_H$ is a bijection from the edges of G to the edges of H with the property that for each vertex $v \in V_G$ and each edge $e \in E_G$, $(v, e) \in \rho_G$ if and only if $(\phi_1(v), \phi_2(e)) \in \rho_H$. If such a bijection exists, we say that G and H are *isomorphic*, and denote this fact by $G \sim H$.

A *graph property* is any subset $\mathbb{P} \subseteq$ GRAPHS closed under isomorphisms. That is to say, for each two isomorphic graphs G and H in GRAPHS, $G \in \mathbb{P}$ if and only if $H \in \mathbb{P}$. Note that the sets $\emptyset$ and GRAPHS are graph properties. Given a set S of graphs, the *isomorphism closure* of S is defined as the set $\mathsf{ISO}(S) = \{G \in \text{GRAPHS} \ : \ \exists H \in S, G \sim H\}$.

Given a graph property $\mathbb{P}$, a $\mathbb{P}$-*invariant* is a function $\mathcal{I} : \mathbb{P} \to S$, for some set S, that is invariant under graph isomorphisms. More precisely, $\mathcal{I}(G) = \mathcal{I}(H)$ for each two isomorphic graphs G and H in $\mathbb{P}$. If $\mathbb{P} =$ GRAPHS, we may say that $\mathcal{I}$ is simply a *graph invariant*. For instance, chromatic number, clique number, dominating number, etc., as well as width measures such as treewidth and cliquewidth, are all graph invariants. In this work, the set S will be typically $\mathbb{N}$, when considering width measures, or $\{0,1\}^*$ when considering other invariants encoded in binary. In order to avoid confusion, we may use the letter $\mathcal{M}$ to denote invariants corresponding to width measures, and the letter $\mathcal{I}$ to denote general invariants.

A *ranked alphabet* is a finite non-empty set Σ together with function $\mathfrak{r} : \Sigma \to \mathbb{N}$ that specifies the arity of each symbol in Σ. The arity of r is the maximum arity of a symbol in Σ. A term over Σ is a pair $\tau = (T, \lambda)$ where T is a rooted tree, where the children of each node are totally ordered, and $\lambda : \mathsf{Nodes}(T) \to \Sigma$ is a function that labels each node p in $\mathsf{Nodes}(T)$ with a symbol from Σ of arity $|\text{Children}(p)|$, i.e., the number of children of p. In particular, leaf nodes are labeled with symbols of arity 0. We may write $\mathsf{Nodes}(\tau)$ to refer to $\mathsf{Nodes}(T)$. We write $|\tau|$ to denote $|\mathsf{Nodes}(T)|$. The height of τ is defined as the height of T. We denote by $\text{Terms}(\Sigma)$ the set of all terms over Σ. If $\tau_1 = (T_1, \lambda_1), ..., \tau_r = (T_r, \lambda_r)$ are terms in $\text{Terms}(\Sigma)$, and $a \in \Sigma$ is a symbol of arity r, then we let $a(\tau_1, ..., \tau_r)$ denote the term $\tau = (T, \lambda)$ where $\mathsf{Nodes}(T) = \{u\} \cup \mathsf{Nodes}(T_1) \cup \cdots \cup \mathsf{Nodes}(T_r)$ for some fresh node u, $\mathsf{root}(T) = u$, $\lambda(u) = a$, and $\lambda|_{\mathsf{Nodes}(T_j)} = \lambda_j$ for each

$j \in [r]$. A tree automaton is a tuple $\mathcal{A} = (\Sigma, Q, F, \Delta)$ where Σ is a ranked alphabet, Q is a finite set of states, F is a final set of states, and Δ is a set of transitions (i.e. rewriting rules) of the form $a(q_1, \ldots, q_r) \to q$, where a is a symbol of arity r, and $q_1, \ldots, q_r, q$ are states in Q. A term τ is accepted by $\mathcal{A}$ if it can be rewritten into a final state in F by transitions in Δ. The language of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of all terms accepted by $\mathcal{A}$. We refer to (Comon et al. 2008) for basic concepts on tree automata theory.

3 Treelike Width Measures

In this section, we introduce the notion of a *treelike width measure*. Subsequently, we show that prominent width measures such as treewidth and cliquewidth fulfil the conditions of our definition. We start by introducing the notion of a *treelike decomposition class*.

Definition 2. *Let $r \in \mathbb{N}$. A treelike decomposition-class of arity r is a sequence $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$, where for each $k \in \mathbb{N}$, Σ_k is a ranked alphabet of arity at most r, L_k is a regular tree language over Σ_k, and $\mathcal{G}_k : \mathsf{L}_k \to$ GRAPHS is a function that assigns a graph $\mathcal{G}_k(\tau)$ to each $\tau \in \mathsf{L}_k$. Additionally, we require that for each $k \in \mathbb{N}$, $\Sigma_k \subseteq \Sigma_{k+1}$, $\mathsf{L}_k \subseteq \mathsf{L}_{k+1}$, and $\mathcal{G}_{k+1}|_{\mathsf{L}_k} = \mathcal{G}_k$.*

Terms in the set $\mathsf{L}(\mathsf{C}) = \bigcup_{k\in\mathbb{N}} \mathsf{L}_k$ are called C-*decompositions*. For each such a term τ, we may write simply $\mathcal{G}(\tau)$ to denote $\mathcal{G}_k(\tau)$. The C-*width* of C-decomposition τ, denoted by $w_{\mathsf{C}}(\tau)$, is the minimum k such that $\tau \in \mathsf{L}_k$. The C-*width* of a graph G, denoted by $w_{\mathsf{C}}(G)$, is the minimum C-width of a C-decomposition τ with $\mathcal{G}(\tau) \simeq G$. We let $w_{\mathsf{C}}(G) = \infty$ if no such minimum k exists.

For each $k \in \mathbb{N}$, we may write $\mathsf{C}_k = (\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)$ to denote the k-th triple in C. The *graph property defined by* C_k is the set $\mathbb{G}[\mathsf{C}_k] = \mathsf{ISO}(\{\mathcal{G}(\tau) : \tau \in \mathsf{L}_k\})$. Note that every graph in $\mathbb{G}[\mathsf{C}_k]$ has C-width at most k, and that $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{C}_{k+1}]$. We let $\mathbb{G}[\mathsf{C}] = \bigcup_{k\in\mathbb{N}} \mathbb{G}[\mathsf{C}_k]$ be the graph property defined by C. We note that the C-width of any graph in $\mathbb{G}[\mathsf{C}]$ is finite.

Definition 3 (Treelike Width Measure)**.** *Let $\mathbb{P}$ be a graph property and $\mathcal{M} : \mathbb{P} \to \mathbb{N}$ be a $\mathbb{P}$-invariant. We say that $\mathcal{M}$ is a* treelike width measure *if there is a treelike decomposition-class C such that $\mathbb{P} = \mathbb{G}[\mathsf{C}]$, and for each graph $G \in \mathbb{P}$, $w_{\mathsf{C}}(G) = \mathcal{M}(G)$. In this case, we say that C is a* realization *of $\mathcal{M}$.*

The next theorem states that several well studied width measures for graphs are treelike.

Theorem 4. *The width measures treewidth, pathwidth, carving width, cutwidth, cliquewidth and ODD width(Andrade de Melo and Oliveira Oliveira 2019) are treelike width measures.*

In our results related to width-based automated theorem proving, we will need to take into consideration the time necessary to construct a description of the languages associated with a treelike decomposition class. Let $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ be a treelike decomposition class of arity r. An automation for C is a sequence $\mathcal{A} = \{\mathcal{A}_k\}_{k\in\mathbb{N}}$ of tree automata where for each $k \in \mathbb{N}$, $\mathcal{L}(\mathcal{A}_k) = \mathsf{L}_k$. We say that $\mathcal{A}$ has complexity $f : \mathbb{N} \to \mathbb{N}$ if for each $k \in \mathbb{N}$, $\mathcal{A}_k$ has at most $f(k)$ states, and there is an algorithm $\mathfrak{A}$ that takes a number $k \in \mathbb{N}$ as input, and constructs $\mathcal{A}_k$ in time $k^{O(1)} \cdot f(k)^{O(r)}$.

4 A DP-Friendly Realization of Treewidth

As stated in the proof of Theorem 4, a construction from (Downey and Fellows 2012) shows that treewidth fulfills our definition of a treelike width measure. Several more logically-oriented constructions have been considered in the literature (Bojańczyk and Pilipczuk 2016; Adler, Grohe, and Kreutzer 2008; Courcelle and Engelfriet 2012; Elberfeld 2016; Flum, Frick, and Grohe 2002). In this section, we introduce an alternative realization of treewidth as a treelike width measure. The reason for us to consider this realization is that, at the same time that it allows one to specify graphs of bounded treewidth using terms over a finite alphabet, this realization is closer in spirit to the notion of an edge introducing nice tree decomposition. This allows one to translate dynamic programming algorithms operating on such decompositions to our framework without much difficulty.

Definition 5. *For each $k \in \mathbb{N}$, we let*

$$\Sigma_k = \{\texttt{Leaf},\ \texttt{IntroVertex}\{u\},\ \texttt{ForgetVertex}\{u\},\ \texttt{IntroEdge}\{u, v\},\ \texttt{Join} : u, v \in [k+1], u \neq v\}.$$

where `Leaf` *is a symbol of arity* 0*,* $\texttt{IntroVertex}\{u\}$*,* $\texttt{ForgetVertex}\{u\}$ *and* $\texttt{IntroEdge}\{u, v\}$ *are symbols of arity* 1*, and* `Join` *is a symbol of arity* 2*. We call Σ_k the k-instructive alphabet.*

Intuitively, the elements of Σ_k should be regarded as instructions that can be used to construct graphs inductively. Each such a graph has an associated set $\mathfrak{b} \subseteq [k+1]$ of *active labels*. In the base case, the instruction `Leaf` creates an empty graph with an empty set of active labels. Now, let G be a graph with set of active labels $\mathfrak{b}$. For each $u \in [k+1] \setminus \mathfrak{b}$, the instruction $\texttt{IntroVertex}\{u\}$ adds a new vertex to G, labels this vertex with u, and adds u to $\mathfrak{b}$. For each $u \in \mathfrak{b}$, the instruction $\texttt{ForgetVertex}\{u\}$ erases the label from the current vertex labeled with u, and removes u from $\mathfrak{b}$. The intuition is that the label u is now free and may be used later in the creation of another vertex. For each $u, v \in \mathfrak{b}$, the instruction $\texttt{IntroEdge}\{u, v\}$ introduces a new edge between the current vertex labeled with u and the current vertex labeled with v. We note that multiedges are allowed in our graphs. Finally, if G and G' are two graphs, each having $\mathfrak{b}$ as the set of active labels, then the instruction `Join` creates a new graph by identifying, for each $u \in \mathfrak{b}$, the vertex of G labeled with u with the vertex of G' labeled with u.

A graph constructed according to the process described above can be encoded by a term over the alphabet Σ_k. We let ITD_k the set of all terms over Σ_k that encode the construction of some graph. The terms in ITD_k are called k-instructive tree decompositions.

Lemma 6. *Let $G \in$ GRAPHS and $k \in \mathbb{N}$. Then G has treewidth at most k if and only if there exists a k-instructive tree decomposition τ such that $\mathcal{G}(\tau) \simeq G$.*

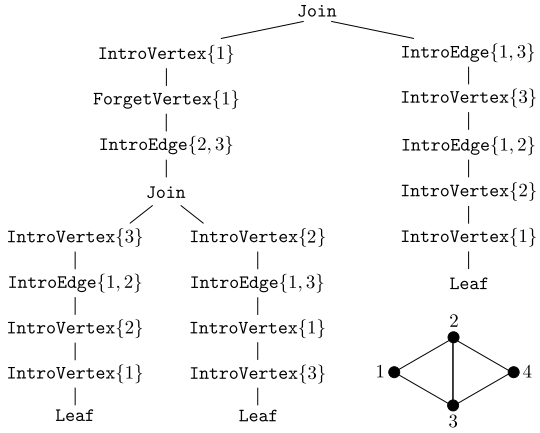

Figure 1: Left: a 2-instructive tree decomposition τ, and the graph $\mathcal{G}(\tau)$ associated with τ. Note that the graph has four vertices even though only elements from $\{1, 2, 3\}$ are used to label the nodes of the tree. Intuitively, once a label has been forgotten, it can be reused to define a new vertex.

5 Treelike Dynamic Programming Cores

In this section, we introduce the notion of a *treelike dynamic-programming core* (treelike DP-core), a formalism intended to capture the behavior of dynamic programming algorithms operating on treelike decompositions. Our formalism generalizes and refines the notion of dynamic programming core introduced in (Baste et al. 2022). There are two crucial differences. First, our framework can be used to define DP-cores for classes of dense graphs, such as graphs of constant cliquewidth, whereas the DP-cores devised in (Baste et al. 2022) are specialized to work on tree decompositions. Second, and most importantly, in our framework, graphs of width k can be represented as terms over ranked alphabets whose size depend only on k. This property makes our framework modular and particularly suitable for applications in the realm of automated theorem proving.

Definition 7 (Treelike DP-Cores)**.** *A treelike dynamic programming core is a sequence of* 6*-tuples* $\mathsf{D} = \{(\Sigma_k, \mathcal{W}_k, \texttt{Final}_k, \Delta_k, \texttt{Clean}_k, \texttt{Inv}_k)\}_{k\in\mathbb{N}}$ *where for each* $k \in \mathbb{N}$*,*

1. Σ_k *is a ranked alphabet;*
2. $\mathcal{W}_k$ *is a decidable subset of* $\{0,1\}^*$*;*
3. $\texttt{Final}_k : \mathcal{W}_k \to \{0,1\}$ *is a function;*
4. Δ_k *is a set containing*
   - *a* finite *set* $\hat{a} \subseteq \mathcal{P}_{\text{fin}}(\mathcal{W}_k)$ *for each symbol* a *of arity* 0*,*
   - *a function* $\hat{a} : \mathcal{W}_k^{\times \mathtt{r}(a)} \to \mathcal{P}_{\text{fin}}(\mathcal{W}_k)$ *for each symbol* a *of arity* $\mathtt{r}(a) \geq 1$*;*
5. $\texttt{Clean}_k : \mathcal{P}_{\text{fin}}(\mathcal{W}_k) \to \mathcal{P}_{\text{fin}}(\mathcal{W}_k)$ *is a function;*
6. $\texttt{Inv}_k : \mathcal{P}_{\text{fin}}(\mathcal{W}_k) \to \{0,1\}^*$ *is a function.*

We let $\mathsf{D}[k] = (\Sigma_k, \mathcal{W}_k, \texttt{Final}_k, \Delta_k, \texttt{Clean}_k, \texttt{Inv}_k)$ denote the k-th tuple of D. We may write $\mathsf{D}[k].\Sigma$ to denote the set Σ_k, $\mathsf{D}[k].\mathcal{W}$ to denote the set $\mathcal{W}_k$, and so on.

Intuitively, for each k, $\mathsf{D}[k]$ is a description of a dynamic programming algorithm that operates on terms from $\text{Terms}(\Sigma_k)$. This algorithm processes such a term τ from the leaves towards the root and assigns a set of local witnesses to each node of τ. The algorithm starts by assigning the set $\mathsf{D}[k].\hat{a}$ to each leaf node labeled with symbol a. Subsequently, the set of local witnesses to be assigned to each internal node p is computed by taking into consideration the label of the node, and the set (sets) of local witnesses assigned to the child (children) of p. The algorithm accepts τ if at the end of the process, the set of local witnesses associated with the root node $\mathsf{root}(\tau)$ has some final local witness, i.e., some local witness $\mathsf{w} \in \mathcal{W}$ such that $\mathsf{D}[k].\texttt{Final}(\mathsf{w}) = 1$.

The function $\mathsf{D}[k].\texttt{Clean}$ is used to remove redundant local witnesses during the processing of a term. The function $\mathsf{D}[k].\texttt{Inv}$ is useful in the context of optimization problems. For instance, given a set S of local witnesses encoding weighted partial solutions to a given problem, $\mathsf{D}[k].\texttt{Inv}(S)$ may return (a binary encoding of) the minimum/maximum weight of a partial solution in the set.

The process described above is formalized in our framework using the notion of *k-th dynamization* of a dynamic core D, which is a function $\Gamma[\mathsf{D}, k]$ that assigns a set $\Gamma[\mathsf{D}, k](\tau)$ of local witnesses to each term $\tau \in \text{Terms}(k)$. Given a symbol a of arity r in the set $\mathsf{D}[k].\Sigma$, and subsets $S_1, \ldots, S_r \subseteq \mathsf{D}[k].\mathcal{W}$, we let $\mathsf{D}[k].\hat{a}(S_1, \ldots, S_r)$ denote the following set:

$$\mathsf{D}[k].\texttt{Clean}\left(\bigcup_{i\in[r],\mathsf{w}_i\in S_i} \mathsf{D}[k].\hat{a}(\mathsf{w}_1, \ldots, \mathsf{w}_r)\right). \quad (1)$$

Using this notation, for each $k \in \mathbb{N}$, the function $\Gamma[\mathsf{D}, k]$ is defined by induction on the structure of τ as follows.

Definition 8 (Dynamization)**.** *Let* D *be a treelike DP-core. For each* $k \in \mathbb{N}$*, the* k*-th dynamization of* D *is the function* $\Gamma[\mathsf{D}, k] : \text{Terms}(\mathsf{D}[k].\Sigma) \to \mathcal{P}_{\text{fin}}(\mathsf{D}[k].\mathcal{W})$ *inductively defined as follows.*

1. *If* $\tau = a$ *for some symbol* $a \in \mathsf{D}[k].\Sigma$ *of arity* 0*, then* $\Gamma[\mathsf{D}, k](\tau) = \mathsf{D}[k].\hat{a}$*.*
2. *If* $\tau = a(\tau_1, \ldots, \tau_r)$ *for some* $a \in \mathsf{D}[k].\Sigma$ *of arity* r*, and some terms* $\tau_1, \ldots, \tau_r$ *in* $\text{Terms}(\mathsf{D}[k].\Sigma)$*, then* $\Gamma[\mathsf{D}, k](\tau) = \mathsf{D}[k].\hat{a}(\Gamma[\mathsf{D}, k](\tau_1), \ldots, \Gamma[\mathsf{D}, k](\tau_r))$*.*

For each $k \in \mathbb{N}$, we say that a term $\tau \in \text{Terms}(\mathsf{D}[k].\Sigma)$ is *accepted* by $\mathsf{D}[k]$ if $\Gamma[\mathsf{D}, k](\tau)$ contains a final local witness, i.e., a local witness w with $\mathsf{D}[k].\texttt{Final}(\mathsf{w}) = 1$. We let $\mathsf{Acc}(\mathsf{D}[k])$ denote the set of all terms accepted by $\mathsf{D}[k]$. We let $\mathsf{Acc}(\mathsf{D}) = \bigcup_{k\in\mathbb{N}} \mathsf{Acc}(\mathsf{D}[k])$. The combination of the notion of a DP-core with the notion of a treelike decomposition class can be used to define graph properties.

Definition 9 (Graph Property of a DP-Core)**.** *Let* C *be a treelike decomposition class, and* D *be a treelike DP-core. For each* $k \in \mathbb{N}$*, the graph property of* $\mathsf{D}[k]$ *is the set*

$$\mathbb{G}[\mathsf{D}[k], \mathsf{C}] = \mathsf{ISO}(\{\mathcal{G}(\tau) \ : \ \tau \in \mathsf{L}_k \cap \mathsf{Acc}(\mathsf{D}[k])\}).$$

The graph property defined by D *is the set*

$$\mathbb{G}[\mathsf{D}, \mathsf{C}] = \bigcup_k \mathbb{G}[\mathsf{D}[k], \mathsf{C}].$$

We note that for each $k \in \mathbb{N}$, $\mathbb{G}[\mathsf{D}[k], \mathsf{C}] \subseteq \mathbb{G}[\mathsf{C}_k]$, and hence, $\mathbb{G}[\mathsf{D}, \mathsf{C}] \subseteq \mathbb{G}[\mathsf{C}]$.

5.1 Coherency

In order to be useful in the context of model-checking and automated theorem proving, DP-cores need to behave coherently with respect to distinct treelike decompositions of the same graph. This intuition is formalized by the following definition.

Definition 10 (Coherency)**.** *Let* $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ *be a treelike decomposition class, and* D *be a treelike DP-core. We say that* D *is* C*-coherent if for each* $k \in \mathbb{N}$, $\Sigma_k = \mathsf{D}[k].\Sigma$*, and for each* $k, k' \in \mathbb{N}$*, and each* $\tau \in \mathsf{L}_k$ *and* $\tau' \in \mathsf{L}_{k'}$ *with* $\mathcal{G}(\tau) \simeq \mathcal{G}(\tau')$*,*

1. $\tau \in \mathsf{Acc}(\mathsf{D}[k])$ *if and only if* $\tau' \in \mathsf{Acc}(\mathsf{D}[k'])$*, and*
2. $\mathsf{D}.\mathtt{Inv}(\Gamma[\mathsf{D}, k](\tau)) = \mathsf{D}.\mathtt{Inv}(\Gamma[\mathsf{D}, k'](\tau'))$.

Let D be a C-coherent treelike DP-core. Condition 1 of Definition 10 guarantees that if a graph G belongs to $\mathbb{G}[\mathsf{D}, \mathsf{C}]$, then for each $k \in \mathbb{N}$ and each C-decomposition τ of width at most k such that $\mathcal{G}(\tau) \simeq G$, we have that $\tau \in \mathsf{Acc}(\mathsf{D}[k])$. On the other hand, if G does not belong to $\mathbb{G}[\mathsf{D}, \mathsf{C}]$, then no C-decomposition τ with $\mathcal{G}(\tau) \simeq G$ belongs to $\mathsf{Acc}(\mathsf{D})$. This discussion is formalized in the following proposition.

Proposition 11. *Let* $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ *be a treelike decomposition class, and* D *be a* C*-coherent treelike DP-core. Then for each* $k \in \mathbb{N}$*, and each* $\tau \in \mathsf{L}_k$*, we have that* $\mathcal{G}(\tau) \in \mathbb{G}[\mathsf{D}, \mathsf{C}]$ *if and only if* $\tau \in \mathsf{Acc}(\mathsf{D}[k])$.

Coherent DP-cores may be used to define not only graph properties but also graph invariants, as specified in Definition 12.

Definition 12 (Invariant of a DP-Core)**.** *Let* C *be a decomposition class and* D *be a* C*-coherent treelike DP-core. The* $\mathbb{G}[\mathsf{D}, \mathsf{C}]$*-invariant defined by* D *is the function* $\mathcal{I}[\mathsf{D}, \mathsf{C}] : \mathbb{G}[\mathsf{D}, \mathsf{C}] \to \{0, 1\}^*$ *that assigns to each graph* $G \in \mathbb{G}[\mathsf{D}, \mathsf{C}]$*, the string* $\mathsf{D}[w_{\mathsf{C}}(\tau)].\mathtt{Inv}(\Gamma[\mathsf{D}, k](\tau))$ *where* τ *is an arbitrary* C*-decomposition with* $\mathcal{G}(\tau) \simeq G$*.*

We note that Condition 2 of Definition 10 guarantees that

$$\mathsf{D}[w_{\mathsf{C}}(\tau)].\mathtt{Inv}(\Gamma[\mathsf{D}, k](\tau)) = \mathsf{D}[w_{\mathsf{C}}(\tau')].\mathtt{Inv}(\Gamma[\mathsf{D}, k](\tau'))$$

for any two C-decompositions τ and τ' with $\mathcal{G}(\tau) \simeq \mathcal{G}(\tau')$. Therefore, for each graph $G \in \mathbb{G}[\mathsf{D}, \mathsf{C}]$, the value $\mathcal{I}[\mathsf{D}, \mathsf{C}](G)$ is well defined, and invariant under graph isomorphism.

5.2 Complexity Measures

In order to analyze the behavior of treelike DP-cores from a quantitative point of view we define the notions of *bitlength* and *multiplicity* of a DP-core D. We say that a set S of local witnesses is (D, k, n)*-useful* if there is some $\tau \in \mathrm{Terms}(\mathsf{D}[k].\Sigma)$ of size $|\tau|$ at most n such that $\Gamma[\mathsf{D}, k](\tau) = S$. The *bitlength* of D is the function β_{D} that assigns to each pair (k, n) the maximum number of bits in a $\beta_{\mathsf{D}}(k, n)$ in a (D, k, n)-useful witness, while the *multiplicity* μ_{D} of D is the function that assigns to each pair (k, n) the maximum number of elements $\mu_{\mathsf{D}}(k, n)$ in a (D, k, n)-useful set.

An important class of DP-cores is the class of cores where maximum number of bits in a useful local witness corresponding to a term τ is independent of the size of τ. In other words, the number of bits may depend on k but not on $|\tau|$.

Definition 13 (Finite DP-cores)**.** *We say that a treelike DP-core* D *is* finite *if there is a function* $f : \mathbb{N} \to \mathbb{N}$ *such that for each* $n \in \mathbb{N}$*,* $\beta_{\mathsf{D}}(k, n) \leq f(k)$.

If D is a finite DP-core then we may write simply $\beta_{\mathsf{D}}(k)$ and $\mu_{\mathsf{D}}(k)$ to denote the functions $\beta_{\mathsf{D}}(k, n)$, and $\mu_{\mathsf{D}}(k, n)$ respectively.

We say that a DP-core is *internally polynomial*, if there is an algorithm $\mathfrak{A}$ that when given a number $k \in \mathbb{N}$ as input, simulates the functions in $\mathsf{D}[k]$ in time polynomial in k and in the size of the input to these functions. We note that typical dynamic programming algorithms operating on tree-like decompositions give rise to internally polynomial DP-cores. Note that the fact that D is internally polynomial does not imply that one can determine whether a given term τ is accepted by D in time polynomial in $|\tau|$. The complexity of this test is governed by the bitlength and multiplicity of the DP-core in question (see Theorem 14).

5.3 Model Checking and Invariant Computation

Let $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ be a treelike decomposition class, and D be a C-coherent treelike DP-core. Given a C-decomposition of width at most k, we can use the notion of dynamization (Definition 8), to check whether the graph $\mathcal{G}(\tau)$ encoded by τ belongs to the graph property $\mathbb{G}[\mathsf{D}, \mathsf{C}]$ represented by D. The next theorem states that the complexity of this model-checking task is essentially governed by the bitlength and by the multiplicity of D. We note that in typical applications the arity r of a decomposition class is a constant (most often 1 or 2), and the width k is smaller than $\beta_{\mathsf{D}}(k, n)$ for each $n \in N$. Nevertheless, for completeness, we explicitly include the dependence on $k^{O(1)}$ and $r^{O(1)}$ in the calculation of the running time.

Theorem 14 (Model Checking)**.** *Let* $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ *be a treelike decomposition class of arity* r*,* D *be an internally polynomial* C*-coherent treelike DP-Core, and let* τ *be a* C*-decomposition of* C*-width at most* k *and size* $|\tau| = n$*.*

1. *One can determine whether* $\mathcal{G}(\tau) \in \mathbb{G}[\mathsf{D}, \mathsf{C}]$ *in time*
$$T(k, n) = n \cdot k^{O(1)} \cdot r^{O(1)} \cdot \beta_{\mathsf{D}}(k, n)^{O(1)} \cdot \mu_{\mathsf{D}}(k, n)^{r+O(1)}.$$
2. *One can compute the invariant* $\mathcal{I}[\mathsf{D}, \mathsf{C}](\mathcal{G}(\tau))$ *in time*
$$T(k, n) + k^{O(1)} \cdot \beta_{\mathsf{D}}(k, n)^{O(1)} \cdot \mu_{\mathsf{D}}(k, n)^{O(1)}.$$

5.4 Inclusion Test

Let C be a treelike decomposition class, and D be a treelike DP-core. As discussed in the introduction, the problem of determining whether $\mathbb{G}[\mathsf{C}] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ can be regarded as a task in the realm of automated theorem proving. A width-based approach to testing whether this inclusion holds is to test for increasing values of k, whether the inclusion $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ holds. It turns out that if D is C-coherent, then testing whether $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ reduces to testing whether all C-decompositions of width at most k are accepted by $\mathsf{D}[k]$, as stated in Lemma 15 below. We note that this is not necessarily true if D is not C-coherent.

Lemma 15. *Let $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ be a treelike decomposition class and D be a C-coherent treelike DP-core. Then, for each $k \in \mathbb{N}$, $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ if and only if $\mathsf{L}_k \subseteq \mathsf{Acc}(\mathsf{D}[k])$.*

Lemma 15 implies that if D is coherent, then in order to show that $\mathbb{G}[\mathsf{C}_k] \nsubseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ it is enough to show that there is some C-decomposition τ of width at most k that belongs to L_k but not to $\mathsf{Acc}(\mathsf{D}[k])$. We will reduce this later task to the task of constructing a dynamic programming refutation (Definition 16).

Let C be a decomposition class with automation $\mathcal{A}$, and let D be a C-coherent treelike DP-core. An $(\mathcal{A}, \mathsf{D}, k)$-pair is a pair of the form (q, S) where q is a state of $\mathcal{A}_k$ and $S \subseteq \mathsf{D}[k].\mathcal{W}$. We say that such pair (q, S) is $(\mathcal{A}, \mathsf{D}, k)$-*inconsistent* if q is a final state of $\mathcal{A}_k$, but S has no final local witness for D.

Definition 16 (DP-Refutation)**.** *Let $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ be a decomposition class, $\mathcal{A}$ be an automation for C, D be a C-coherent treelike DP-core, and $k \in \mathbb{N}$. An $(\mathcal{A}, \mathsf{D}, k)$-refutation is a sequence of $(\mathcal{A}, \mathsf{D}, k)$-pairs*

$$R \equiv (q_1, S_1)(q_2, S_2) \dots (q_m, S_m)$$

satisfying the following conditions:

1. *(q_m, S_m) is $(\mathcal{A}, \mathsf{D}, k)$-inconsistent.*
2. *For each $i \in [m]$,*
   (a) *either $(q_i, S_i) = (q, \mathsf{D}[k].\hat{a})$ for some symbol a of arity 0 in Σ_k, and some state q such that $a \to q$ is a transition of $\mathcal{A}_k$, or*
   (b) *$(q_i, S_i) = (q, \mathsf{D}[k].\hat{a}(S_{j_1}, \dots, S_{j_{\mathfrak{r}(a)}}))$, for some $j_1, \dots, j_{\mathfrak{r}(a)} < i$, some symbol $a \in \Sigma_k$ of arity $\mathfrak{r}(a) > 0$, and some state q such that $a(q_{j_1}, \dots, q_{j_{\mathfrak{r}(a)}}) \to q$ is a transition of $\mathcal{A}_k$.*

The following theorem shows that if C is a decomposition class with automation $\mathcal{A}$ and D is a C-coherent treelike DP-core, then showing that $\mathbb{G}[\mathsf{C}] \nsubseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$, is equivalent to showing the existence of some $(\mathcal{A}, \mathsf{D}, k)$-refutation.

Theorem 17. *Let $\mathsf{C} = \{(\Sigma_k, \mathsf{L}_k, \mathcal{G}_k)\}_{k\in\mathbb{N}}$ be a decomposition class with automation $\mathcal{A}$, and D be a C-coherent treelike DP-core. For each $k \in \mathbb{N}$, we have that $\mathbb{G}[\mathsf{C}_k] \nsubseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ if and only if some $(\mathcal{A}, \mathsf{D}, k)$-refutation exists.*

Theorem 17 implies the existence of a simple forward-chaining style algorithm for determining whether $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ when D is a *finite* and C-coherent treelike DP-core.

Theorem 18 (Inclusion Test)**.** *Let C be a treelike decomposition class of complexity $f(k)$ and arity r, and let D be a finite, internally polynomial C-coherent treelike DP core. One can determine whether $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ in time*

$$f(k)^{O(r)} \cdot 2^{O(r\cdot\beta_{\mathsf{D}}(k)\cdot\mu_{\mathsf{D}}(k))} \leq f(k)^{O(r)} \cdot 2^{r\cdot 2^{O(\beta_{\mathsf{D}}(k))}}.$$

5.5 Combinators and Combinations

Given a graph property $\mathbb{P}$, and a graph $G \in$ GRAPHS, we let $\mathbb{P}(G)$ denote the Boolean value *true* if $G \in \mathbb{P}$ and the value *false*, if $G \notin \mathbb{P}$. For each $\ell \in \mathbb{N}$ we call a function of the form

$$\mathcal{C} : \{0,1\}^\ell \times (\{0,1\}^*)^\ell \to \{0,1\}.$$

an ℓ-*combinator*. Given graph properties $\mathbb{P}_1, \dots, \mathbb{P}_\ell$ and graph invariants $\mathcal{I}_1 \dots, \mathcal{I}_\ell$, we let $\hat{\mathcal{C}}(\mathbb{P}_1, \dots, \mathbb{P}_\ell, \mathcal{I}_1, \dots, \mathcal{I}_\ell)$ denote the graph property consisting of all graphs G such that $\mathcal{C}(\mathbb{P}_1(G), \dots, \mathbb{P}_\ell(G), \mathcal{I}_1(G), \dots, \mathcal{I}_\ell(G)) = 1$. We say that $\mathcal{C}$ is *polynomial* if it can be computed in time $O_\ell(|X|^c)$ for some constant c on any given input X.

Intuitively, a combinator is a tool to define graph classes in terms of previously defined graph classes and previously defined graph invariants. It is worth noting that Boolean combinations of graph classes can be straightforwardly defined using combinators. Nevertheless, one can do more than that, since combinators can also be used to establish relations between graph invariants. For instance, using combinators one can define the class of graphs whose *covering number* (the smallest size of a vertex-cover) is equal to the *dominating number* (the smallest size of a dominating set). This is just a illustrative example. Other examples of invariants that can be related using combinators are: *clique number*, *independence number*, *chromatic number*, *diameter*, and many others. Next, we will use combinators as a tool to combine graph properties and graph invariants defined using DP-cores. We let $\delta_{\mathsf{D}}(k, n)$ be the number of (D, k, n)-useful sets. We call δ_{D} the *deterministic state complexity* (d.s.c.) of D.

Theorem 19. *Let $\mathcal{C}$ be and ℓ-combinator, C be a treelike decomposition class, and $\mathsf{D}_1, \dots, \mathsf{D}_\ell$ be C-coherent treelike DP-cores. Then, there exists a C-coherent treelike DP-core $\mathsf{D} = \mathsf{D}(\mathcal{C}, \mathsf{D}_1, \dots, \mathsf{D}_\ell)$ satisfying the following properties:*

1. *$\mathbb{G}[\mathsf{D}, \mathsf{C}] = \mathcal{C}(\mathbb{G}[\mathsf{D}_1, \mathsf{C}], \dots, \mathbb{G}[\mathsf{D}_\ell, \mathsf{C}], \mathcal{I}[\mathsf{D}_1, \mathsf{C}], \dots, \mathcal{I}[\mathsf{D}_\ell, \mathsf{C}])$.*
2. *D has bitlength $\beta_{\mathsf{D}}(k, n) = \sum_{i=1}^{\ell} \beta_{\mathsf{D}_i}(k, n) \cdot \mu_{\mathsf{D}_i}(k, n)$.*
3. *D has multiplicity $\mu_{\mathsf{D}}(k, n) = 1$.*
4. *D has d.s.c. $\delta_{\mathsf{D}}(k, n) \leq \prod_{i=1}^{\ell} \delta_{\mathsf{D}_i}(k, n)$.*

We call the DP-core $\mathsf{D} = \mathsf{D}(\mathcal{C}, \mathsf{D}_1, \dots, \mathsf{D}_\ell)$ the $\mathcal{C}$-combination of $\mathsf{D}_1, \dots, \mathsf{D}_\ell$. If the DP-cores $\mathsf{D}_1, \dots, \mathsf{D}_\ell$ are also *finite*, besides being C-coherent, and internally polynomial, then Theorem 19 together with Theorem 18 directly imply the following theorem, which will be used in Section 6 to establish analytic upper bound on the time necessary to verify long-standing conjectures on graphs of bounded treewidth.

Theorem 20 (Inclusion Test for Combinations)**.** *Let C be a treelike decomposition class of arity r; $\mathsf{D}_1, \dots, \mathsf{D}_\ell$ be finite, internally polynomial, C-coherent treelike DP-cores; and $\mathcal{C}$ be a polynomial ℓ-combinator. Let $\mathsf{D} = \mathsf{D}(\mathcal{C}, \mathsf{D}_1, \dots, \mathsf{D}_\ell)$ be the $\mathcal{C}$-combination of $\mathsf{D}_1, \dots, \mathsf{D}_\ell$, $\beta(k) = \max_i \beta_{\mathsf{D}_i}(k)$ and $\mu(k) = \max_i \mu_{\mathsf{D}_i}(k)$. Then, for each $k \in \mathbb{N}$, one can determine whether $\mathbb{G}[\mathsf{C}_k] \subseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$ in time*

$$f(k)^{O(r)} \cdot 2^{O(\ell\cdot r\cdot\beta(k)\cdot\mu(k))} \leq f(k)^{O(r)} \cdot 2^{\ell\cdot r\cdot 2^{O(\beta(k))}}.$$

We note that in typical applications, the parameters r and ℓ are constant, while the growth of the function $f(k)$ is negligible when compared with $2^{O(\beta(k)\cdot\mu(k))}$. Therefore, in these applications, the running time of our algorithm is of the form $2^{O(\beta(k)\cdot\mu(k))} \leq 2^{2^{O(\beta(k))}}$. It is also worth noting that if $\mathbb{G}[\mathsf{C}_k] \nsubseteq \mathbb{G}[\mathsf{D}, \mathsf{C}]$, then there is a term τ of height at most $2^{O(\beta(k)\cdot\mu(k))}$ encoding a graph in $\mathbb{G}[\mathsf{C}_k] \backslash \mathbb{G}[\mathsf{D}, \mathsf{C}]$. Such a term can be constructed by backtracking.

6 Applications of Theorem 20

In this section, we show that Theorem 20 can be used to show that several long-standing graph-theoretic conjectures can be tested in time double exponential in $k^{O(1)}$ on the class of graphs of treewidth at most k. The next theorem enumerates upper bounds on the bitlength and multiplicity of DP-cores deciding several graph properties. These upper bounds are obtained by translating combinatorial dynamic programming algorithms for these properties parameterized by treewidth into internally polynomial, ITD-coherent, finite DP-cores. Here, ITD is the class of instructive tree decompositions introduced in Section 4. This class has complexity 2^k.

Theorem 21. *Let* ITD *be the instructive tree decomposition class defined in Section 4. The properties specified above have* ITD*-coherent DP-cores with complexity parameters (bitlength β, multiplicity μ, state complexity ν, deterministic state complexity δ) as specified in Table 1.*

Property	$\beta(k)$	$\mu(k)$
`Simple`	$O(k^2)$	1
$\texttt{MaxDeg}_{\geq}(c)$	$O(k \cdot \log c)$	1
$\texttt{MinDeg}_{\leq}(c)$	$O(k \cdot \log c)$	1
`Colorable(c)`	$O(k \log c)$	$2^{O(\beta(k))}$
`Conn`	$O(k \log k)$	$2^{O(\beta(k))}$
`VConn(c)`	$O(\log c + k \log k)$	$2^{O(\beta(k))}$
`EConn(c)`	$O(\log c + k \log k)$	$2^{O(\beta(k))}$
`Hamiltonian`	$O(k \log k)$	$2^{O(k)}$
$\texttt{NZFlow}(\mathbb{Z}_m)$	$O(k \log m)$	$2^{O(\beta(k))}$
`Minor(H)`	$O(k \log k + \lvert V_H\rvert + \lvert E_H\rvert)$	$2^{\beta(k)}$

Table 1: Complexity measures for DP-cores deciding several graph properties.

Note that in the case of the DP-core `C-Hamiltonian` the multiplicity $2^{O(k)}$ is smaller than the trivial upper bound of $2^{O(k \cdot \log k)}$ and consequently, the deterministic state complexity $2^{2^{O(k)}}$ is smaller than the trivial upper bound of $2^{2^{O(k \cdot \log k)}}$. We note that the proof of this fact is a consequence of the rank-based approach developed in (Bodlaender et al. 2015). Next, we will show how Theorem 20 together with Theorem 21 can be used to provide double-exponential upper bounds on the time necessary to verify long-standing graph-theoretic conjectures on graphs of treewidth at most k. If such a conjecture is false, then one can establish an upper bound on minimum height of a term representing a counterexample for the conjecture (we refer the reader to the full version of this paper for further details).

Hadwiger's Conjecture. This conjecture states that for each $c \geq 1$, every graph with no K_{c+1}-minor has a c-coloring (Hadwiger 1943). This conjecture, which suggests a far reaching generalization of the 4-colors theorem, is considered to be one of the most important open problems in graph theory. The conjecture has been resolved in the positive for the cases $c < 6$ (Robertson, Seymour, and Thomas 1993), but remains open for each value of $c \geq 6$. By Theorem 21, $\texttt{Colorable}(c)$ has DP-cores of deterministic state complexity $2^{2^{O(k \log c)}}$, while $\texttt{Minor}(K_{c+1})$ has DP-cores of deterministic state complexity $2^{2^{O(k \log k + c^2)}}$. Therefore, by using Theorem 20, we have that the case c of Hadwiger's conjecture can be tested in time $f(c,k) = 2^{2^{O(k \log k + c^2)}}$ on graphs of treewidth at most k.

Using the fact for each fixed $c \in \mathbb{N}$, both the existence of K_{c+1}-minors and the existence of c-colorings are MSO-definable, together with the fact that the MSO theory of graphs of bounded treewidth is definable one can estimate $f(c,k)$ by writing explicitly MSO sentences and then by bounding the running time of the decision algorithm. This estimate is however very large (a tower of exponentials of height 10 in $k + c$ suffices). In (Kawarabayashi and Reed 2009) Karawabayshi have estimated that $f(c,k) \leq p^{p^{p^p}}$, where $p = (k+1)^{(c-1)}$. It is worth noting that our estimate of $2^{2^{O(k \log k + c^2)}}$ obtained by a combination Theorem 20 and Theorem 21 improves significantly on both the estimate obtained using the MSO approach and the estimate provided in (Kawarabayashi and Reed 2009).

Tutte's Flow Conjectures. Tutte's 5-flow, 4-flow, and 3-flow conjectures are some of the most well studient and important open problems in graph theory. The 5-flow conjecture states that every bridgeless graph G has a $\mathbb{Z}_5$-flow. This conjecture is true if and only if every 2-edge-connected graph has a $\mathbb{Z}_5$-flow (Tutte 1954). By Theorem 21, both $\texttt{ECon}(2)$ and $\texttt{NZFlow}(\mathbb{Z}_5)$ have coherent DP-cores of deterministic state complexity $2^{2^{O(k \log k)}}$. Since Tutte's 5-flow conjecture can be expressed in terms of a Boolen combination of these properties, we have that this conjecture can be tested on graphs of treewidth at most k in time $2^{2^{O(k \log k)}}$ on graphs of treewidth at most k. The 4-flow conjecture states that every bridgeless graph with no Petersen minor has a nowhere-zero 4-flow (Wang, Zhang, and Zhang 2009). Since this conjecture can be formulated using a Boolean combination of the properties $ECon(2)$, $\texttt{Minor}(P)$ (where P is the Pettersen graph), and $\texttt{NZFlow}(\mathbb{Z}_4)$, we have that this conjecture can be tested in time $2^{2^{O(k \log k)}}$ on graphs of treewidth at most k. Finally, Tutte's 3-Flow conjecture states that every 4-edge connected graph has a nowhere-zero 3-flow (Fan 1993). Similarly to the other cases it can be expressed as a Boolean combination of $\texttt{ECon}(4)$ and $\texttt{NZFlow}(\mathbb{Z}_3)$. Therefore, it can be tested in time $2^{2^{O(k \log k)}}$ on graphs of treewidth at most k.

Barnette's Conjecture. This conjecture states that every 3-connected, 3-regular, bipartite, planar graph is Hamiltonian. Since a graph is bipartite if and only if it is 2-colorable, and since a graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as minors, Barnette's conjecture can be stated as a combination of the cores $\texttt{VCon}(3)$, $\texttt{MaxDeg}_{\geq}(3)$, $\texttt{MinDeg}_{\leq}(3)$, $\texttt{Colorable}(2)$, $\texttt{Minor}(K_5)$ and $\texttt{Minor}(K_{3,3})$. Therefore, by Theorem 20, it can be tested in time $2^{2^{O(k \log k)}}$ on graphs of treewidth at most k.

Acknowledgements

We acknowledge support from the Research Council of Norway (projects 288761 and 326537) and from the Sigma2 network (project NN9535K).

References

Adler, I.; Grohe, M.; and Kreutzer, S. 2008. Computing excluded minors. In *Proc. of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*, 641–650. SIAM.

Andrade de Melo, A.; and Oliveira Oliveira, M. d. 2019. On the width of regular classes of finite structures. In *Proc. of the 27th International Conference on Automated Deduction (CADE 2019)*, 18–34. Springer.

Baste, J.; Fellows, M. R.; Jaffke, L.; Masařík, T.; de Oliveira Oliveira, M.; Philip, G.; and Rosamond, F. A. 2022. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303: 103644.

Bertele, U.; and Brioschi, F. 1973. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2): 137–148.

Biedl, T. C. 2015. On triangulating k-outerplanar graphs. *Discret. Appl. Math.*, 181(1): 275–279.

Bodlaender, H. 1986. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Department of Information and Computing Sciences, Utrecht University.

Bodlaender, H. L. 1997. Treewidth: Algorithmic techniques and results. In *Proc. of the 22nd International Symposium on Mathematical Foundations of Computer Science*, 19–36. Springer.

Bodlaender, H. L. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1-2): 1–45.

Bodlaender, H. L.; Cygan, M.; Kratsch, S.; and Nederlof, J. 2015. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243: 86–111.

Bodlaender, H. L.; and Koster, A. M. 2008. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3): 255–269.

Bojańczyk, M.; and Pilipczuk, M. 2016. Definability equals recognizability for graphs of bounded treewidth. In *Proc. of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016)*, 407–416. ACM.

Brandstädt, A.; Le, V. B.; and Spinrad, J. P. 1999. *Graph classes: a survey*. SIAM.

Chung, F. R.; and Seymour, P. D. 1989. Graphs with small bandwidth and cutwidth. *Discrete Mathematics*, 75(1-3): 113–119.

Comon, H.; Dauchet, M.; Gilleron, R.; Jacquemard, F.; Lugiez, D.; Löding, C.; Tison, S.; and Tommasi, M. 2008. Tree automata techniques and applications. *HAL Inria*, (hal-03367725): 1–262.

Courcelle, B.; and Durand, I. 2016. Computations by fly-automata beyond monadic second-order logic. *Theor. Comput. Sci.*, 619: 32–67.

Courcelle, B.; and Engelfriet, J. 2012. *Graph structure and monadic second-order logic: A language-theoretic approach*, volume 138. Cambridge University Press.

Courcelle, B.; and Olariu, S. 2000. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3): 77–114.

Downey, R. G.; and Fellows, M. R. 2012. *Parameterized complexity*. Springer Science & Business Media.

Elberfeld, M. 2016. Context-Free Graph Properties via Definable Decompositions. In *Proc. of the 25th Conference on Computer Science Logic (CSL 2016)*, volume 62 of *LIPIcs*, 17:1–17:16.

Fan, G. 1993. Tutte's 3-Flow Conjecture and Short Cycle Covers. *Journal of Combinatorial Theory, Series B*, 57(1): 36–43.

Flum, J.; Frick, M.; and Grohe, M. 2002. Query evaluation via tree-decompositions. *Journal of the ACM (JACM)*, 49(6): 716–752.

Hadwiger, H. 1943. Über eine klassifikation der streckenkomplexe. *Vierteljschr. Naturforsch. Ges. Zürich*, 88(2): 133–142.

Halin, R. 1976. S-functions for graphs. *Journal of geometry*, 8(1-2): 171–186.

Kammer, F. 2007. Determining the smallest k such that G is k-outerplanar. In *European Symposium on Algorithms*, 359–370. Springer.

Kawarabayashi, K.-i.; and Reed, B. 2009. Hadwiger's conjecture is decidable. In *Proc. of the 41st Annual ACM Symposium on Theory of Computing*, 445–454.

Korach, E.; and Solel, N. 1993. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1): 97–101.

Robertson, N.; Seymour, P.; and Thomas, R. 1993. Hadwiger's conjecture forK 6-free graphs. *Combinatorica*, 13(3): 279–361.

Robertson, N.; and Seymour, P. D. 1984. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1): 49–64.

Thilikos, D. M.; Serna, M.; and Bodlaender, H. L. 2005. Cutwidth I: A linear time fixed parameter algorithm. *Journal of Algorithms*, 56(1): 1–24.

Thilikos, D. M.; Serna, M. J.; and Bodlaender, H. L. 2000. Constructive linear time algorithms for small cutwidth and carving-width. In *Proc. of the 11th International Symposium on Algorithms and Computation*, 192–203. Springer.

Tutte, W. T. 1954. A contribution to the theory of chromatic polynomials. *Canadian journal of mathematics*, 6: 80–91.

Tutte, W. T. 1969. Recent progress in combinatorics. In *Proceedings of the 3rd Waterloo Conference on Combinatorics*. Academic Press.

Wang, X.; Zhang, C.-Q.; and Zhang, T. 2009. Nowhere-zero 4-flow in almost Petersen-minor free graphs. *Discrete mathematics*, 309(5): 1025–1032.