On the power of k-consistency

Albert Atserias Andrei Bulatov Victor Dalmau

January 29, 2007

Abstract

The k-consistency algorithm for constraint-satisfaction problems proceeds, roughly, by finding all partial solutions on at most k variables and iteratively deleting those that cannot be extended to a partial solution by one more variable. It is known that if the core of the structure encoding the scopes of the constraints has treewidth at most k, then the k-consistency algorithm is always correct. We prove the exact converse to this: if the core of the structure encoding the scopes of the constraints does not have treewidth at most k, then the k-consistency algorithm is not always correct. This characterizes the exact power of the k-consistency algorithm in structural terms.

1 Introduction

Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures of the same type. For concreteness, we can think of $\mathbf{A}$ and $\mathbf{B}$ as directed graphs, each consisting of a set of vertices and a binary relation on the vertices. A homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is a map from the domain of $\mathbf{A}$ to the domain of $\mathbf{B}$ that preserves all the relations. Homomorphisms play a prominent role in combinatorics, logic, and algebra, and also in computer science. Consider for example the constraint-satisfaction problem, where we are given a set of variables that range over a domain of values, and a set of constraints between tuples of variables and tuples of values. The goal is to find an assignment of values to the variables in such a way that all given constraints are fulfilled. It was observed by Feder and Vardi [9] that this problem can be phrased as a homomorphism question between a relational structure $\mathbf{A}$ encoding the set of constraint variables (scopes), and a relational structure $\mathbf{B}$ encoding the set of valid assignment to those variables.

The k-consistency algorithm is a well-known heuristic algorithm to decide the existence of a homomorphism between two structures, or equivalently, to solve constraint-satisfaction problems. In order to simplify the exposition, let us focus again on finite directed graphs $\mathbf{A} = (A, E^{\mathbf{A}})$ and $\mathbf{B} = (B, E^{\mathbf{B}})$ and let us fix $k = 1$. The 1-consistency algorithm is commonly referred to as the *arc-consistency algorithm*. This algorithm proceeds in rounds by iteratively reducing the set of possible places $L(a) \subseteq B$ where a vertex $a \in A$ may be mapped. Initially, every $a \in A$ can be mapped to any $b \in B$, so we start with $L(a) = B$. At each round, if there exists an arc $(a, a') \in E^{\mathbf{A}}$ and a $b \in L(a)$ for which no $b' \in L(a')$ exists such that $(b, b') \in E^{\mathbf{B}}$, we remove a' from $L(a)$. Similarly, if there exists a $b' \in L(a')$ for which no $b \in L(a)$ exists such that $(b, b') \in E^{\mathbf{B}}$, we remove b' from $L(a')$. This process is repeated until there are no more changes in the $L(a)$'s. If at termination $L(a)$ is empty for some $a \in A$, we can guarantee that there exists no homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Otherwise, we say that the instance $\mathbf{A}, \mathbf{B}$ *passes the arc-consistency test*. In this case we know that if there exists a homomorphism $h : \mathbf{A} \to \mathbf{B}$, we must have $h(a) \in L(a)$ for every $a \in A$. Henceforth, the arc-consistency algorithm can be used in order to narrow the possible space of solutions, and indeed, many of the practical CSP solvers use some form of consistency in order to prune the search tree. Furthermore, most of the known tractable subcases of the CSP are solvable by testing some sort of consistency.

The k-consistency test for general k is the natural generalization of this algorithm to k-tuples. The main goal of this paper is to study the power of the k-consistency test as a tool to decide the existence of a solution by itself. More precisely, we are interested in characterizing under which circunstances we can guarantee that every instance passing the k-consistency test has a solution.

Note, first, that the consistency test runs in time polynomial in $|A| \cdot |B|$, which is polynomial in the size of the input. Therefore, since the general homomorphism problem is NP-complete, we cannot expect it to be correct on every instance. Interestingly, though, it is known that the algorithm is correct when the underlying graph of $\mathbf{A}$ is acyclic [10]. This gives a large class of inputs where the algorithm can be used to find homomorphisms in polynomial time. It was later observed that it suffices if the *core* of $\mathbf{A}$ is acyclic [6], where the core of a relational structure $\mathbf{A}$ is the smallest substructure that has homomorphisms from and to $\mathbf{A}$. It is known that such a substructure exists and is unique up to isomorphism [13]. This widens the class of instances where the algorithm works correctly even further. But is that all?

Main result The main result of this paper is the complete answer to the question above. In fact, our result answers the corresponding question for the k-consistency test. In this context, the role of graph acyclicity is played by the concept of treewidth, which is a measure to calibrate the similarity of a graph with a tree.

Treewidth was introduced in the deep work on graph minors by Robertson and Seymour, and has played an important role in algorithmic graph theory since then. For constraint-satisfaction problems, treewidth was identified as useful by Freuder [11], and later revisited by several others [9, 19, 14, 6]. Freuder observed that if the treewidth of $\mathbf{A}$ is at most k, then the k-consistency algorithm is always correct. As with the acyclic case, it was later observed that it suffices that the treewidth of the core of $\mathbf{A}$ be bounded by k. Thus, it was proved in [6] that if the treewidth of the core of $\mathbf{A}$ is at most k, then the k-consistency algorithm is always correct. Our main result is an exact converse to this: if the treewidth of the core of $\mathbf{A}$ is more than k, then the k-consistency algorithm is not always correct. Note that since treewidth 1 agrees with acyclicity, our main result implies, in particular, that if the core of $\mathbf{A}$ is not acyclic, then the arc-consistency test is not always correct.

Related work The notion of k-consistency has proven to be very robust and, besides being one of the central concepts in theory of constraint-satisfaction problems, has also emerged independently in areas as diverse as finite model theory [18], graph theory [16], and proof complexity [1].

The limits of the k-consistency algorithm as a method for finding homomorphisms had been studied before to some extent. First, the k-consistency algorithm runs in polynomial time for every fixed k, so it cannot be always correct unless P = NP. Concrete examples were the algorithm is not correct can be easily found. For example, let $\mathbf{A}$ be a complete graph on $k+2$ vertices, and let $\mathbf{B}$ be a complete graph on $k+1$ vertices. It is not hard to see that there is no homomorphism from $\mathbf{A}$ to $\mathbf{B}$ yet this instance passes the k-consistency test.

Second, Feder and Vardi [9] proved that there exist fixed finite structures $\mathbf{B}$ for which it is possible to determine the existence of a homomorphism $\mathbf{A} \to \mathbf{B}$ in polynomial time, yet the k-consistency algorithm fails for every fixed k. In fact, the structure $\mathbf{B}$ is very explicit and corresponds to the constraint-satisfaction problem of solving systems of linear equations over the two-element field.

Third, Grohe [12] proved the following very general result. Let $\mathcal{F}$ be a class of structures and consider the restricted homomorphism problem when $\mathbf{A}$ is taken from $\mathcal{F}$ and $\mathbf{B}$ is an arbitrary structure. For which $\mathcal{F}$'s is this problem solvable in polynomial time? We know already from [6] that if the class of cores of $\mathcal{F}$ has bounded treewidth, then the problem is solvable in polynomial time. Assuming a conjecture in parameterized complexity theory, Grohe proved the converse to this result: if the problem is solvable in polynomial time, then the class of cores of structures in $\mathcal{F}$ has bounded treewidth. In particular, this implies that for every $k > 1$, there exists some k' such that if the treewidth of the core of a structure $\mathbf{A}$ is at least

k', then the k-consistency algorithm is not always correct. In his proof, the k' is an exponential function of k given by an application of the Excluded Grid Theorem (EGT) of Robertson and Seymour. Instead, our result shows that $k' = k + 1$ with the additional important feature that our proof does not depend on the EGT or on any conjecture in parameterized complexity theory.

2 Preliminaries

Graphs, structures, and treewidth A *vocabulary* is a finite set of relation symbols or predicates. Every relation symbol in a vocabulary has an *arity* associated to it. For a vocabulary σ, a *relational structure* $\mathbf{A}$ of type σ is a pair consisting of a set A, called the *universe* of $\mathbf{A}$, and a sequence of relations $R^{\mathbf{A}}$, one for each relation symbol R from σ, such that the arity of $R^{\mathbf{A}}$ is equal to that of R. For example, a graph is a structure with a single binary relation that is symmetric and irreflexive. All structures in this paper are assumed to be *finite*, i.e. having a finite universe.

A structure $\mathbf{B}$ is called an *induced substructure* of a structure $\mathbf{A}$ of type σ, if the universe B of $\mathbf{B}$ is a subset of the universe A of $\mathbf{A}$, and for any $R \in \sigma$, $R^{\mathbf{B}} = R^{\mathbf{A}} \cap B^r$, where r is the arity of R.

The *Gaifman graph* of a relational structure $\mathbf{A} = (A; R_1, \ldots, R_n)$ is the graph with vertex set A and such that (a, b) is an edge if and only if $a \neq b$, and a and b belong to the same tuple from one of the relations $R_1, \ldots, R_n$. Note that loops are never included in the Gaifman graph.

A *tree decomposition* of a graph $G = (V; E)$ is a labeled tree T such that

1. every node of T is labeled by a non-empty subset of V,
2. for every edge $(v, w) \in E$, there is a node of T whose label contains $\{v, w\}$,
3. for every $v \in V$, the set of nodes of T, whose labels contain v, is a subtree of T.

The *width* of a tree decomposition T is the maximum cardinality of a label in T minus 1. The *treewidth* of a graph G is the smallest number k such that G has a tree decomposition of width k. Note that the treewidth of a tree is one. The treewidth of a structure is the treewidth of its Gaifman graph.

Homomorphisms, constraint-satisfaction and cores A *homomorphism* from a structure $\mathbf{A}$ to a structure $\mathbf{B}$ of the same type is a mapping $f: A \to B$ between the universes of $\mathbf{A}$ and $\mathbf{B}$ such that for every r-ary $R \in \sigma$ and every $(a_1, \ldots, a_r) \in R^{\mathbf{A}}$, we have $(f(a_1), \ldots, f(a_r)) \in R^{\mathbf{B}}$. The fact that there is a homomorphism from structure $\mathbf{A}$ to structure $\mathbf{B}$ we denote by $\mathbf{A} \to \mathbf{B}$. If a homomorphism does not exist we write $\mathbf{A} \not\to \mathbf{B}$.

Let $\mathbf{A}$ and $\mathbf{B}$ be two finite relational structures over the same vocabulary σ. We can think of the pair $\mathbf{A}, \mathbf{B}$ as an instance of the constraint satisfaction problem, where the elements of A are the variables of the problem, and the elements of B are the values they may take. A tuple $(x_1, \ldots, x_r) \in R^{\mathbf{A}}$ denotes the constraint that the variables $x_1, \ldots, x_r$ need to take values in B in such a way that the resulting tuple belongs to $R^{\mathbf{B}}$. Therefore, a solution is a mapping $f : A \to B$ that defines a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

If $\mathfrak{A}, \mathfrak{B}$ are classes of finite relational structures of the same type, the *constraint-satisfaction problem* $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ asks, given a pair of structures $\mathbf{A} \in \mathfrak{A}$ and $\mathbf{B} \in \mathfrak{B}$, whether or not there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. If $\mathfrak{A}$ is the class of all finite structures of a certain type, then we write $\mathrm{CSP}(*, \mathfrak{B})$ instead of $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$. Similarly, if $\mathfrak{B}$ is the class of all finite structures, we write $\mathrm{CSP}(\mathfrak{A}, *)$. In addition, if $\mathfrak{B}$ is one-element, say, $\mathfrak{B} = \{\mathbf{B}\}$, then we write $\mathrm{CSP}(*, \mathbf{B})$, and similarly for $\mathrm{CSP}(\mathbf{A}, *)$.

Example 1 (H-COLORING) Let H be a (directed) graph. In the H-COLORING problem we are asked whether there is a homomorphism from a given graph G to H. So, the H-COLORING problem is equivalent to the problem $\mathrm{CSP}(*, H)$.

Example 2 (CLIQUE) In the CLIQUE problem we are asked whether a given graph contains a clique of a given size. It is not hard to see that CLIQUE is equivalent to $\mathrm{CSP}(\mathfrak{K}, *)$, where $\mathfrak{K}$ is the class of all finite complete graphs.

An *endomorphism* h of $\mathbf{A}$ is a homomorphism from a $\mathbf{A}$ to itself. Furthermore, h is said to be an *automorphism* if it is bijective. A structure is a *core* if every endomorphism is an automorphism. A *core* of a relational structure $\mathbf{A}$ is an induced substructure $\mathbf{B}$ such that $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B}$ is a core. All cores of a structure are isomorphic, and therefore we can talk about *the core* $\mathsf{core}(\mathbf{A})$ of a structure $\mathbf{A}$. It is easy to see that a structure $\mathbf{A}$ and its core are *homomorphically equivalent*, meaning that $\mathbf{A} \to \mathsf{core}(\mathbf{A})$ and $\mathsf{core}(\mathbf{A}) \to \mathbf{A}$. This allows one to reduce many homomorphism properties of structures and classes of structures, i.e. the complexity of problems $\mathrm{CSP}(*, \mathfrak{B})$, $\mathrm{CSP}(\mathfrak{A}, *)$, to the properties of their cores. Yet, with respect to computational complexity, a structure and its core are not always freely exchangable. In particular, it has been shown that deciding whether a structure is a core is co-NP-complete [15], which implies that, in general, it is hard to compute the core of a structure.

3 The k-Consistency Test

Fix some $k \geq 1$. The k-consistency test is a simple algorithm that, given a pair of structures $\mathbf{A}$ and $\mathbf{B}$, either provides a certificate that there is no homomorphism from $\mathbf{A}$ to $\mathbf{B}$, or narrows the set of elements of $\mathbf{B}$ to which each element of $\mathbf{A}$ may be mapped.

Recall that a solution is a mapping $f : A \to B$ that defines a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. A partial solution, also called a *partial homomorphism*, is a mapping $f : A' \to B$, where $A' \subseteq A$, such that f defines a homomorphism from the substructure of $\mathbf{A}$ with universe A' to the structure $\mathbf{B}$. In other words, f is a function such that for every r-ary relation symbol $R \in \sigma$ and $a_1, \dots, a_r \in A'$, if $(a_1, \dots, a_r) \in R^{\mathbf{A}}$ then $(f(a_1), \dots, f(a_r)) \in R^{\mathbf{B}}$. If f and g are partial solutions we say that g *extends* f, denoted by $f \subseteq g$, if $\mathrm{Dom}(f) \subseteq \mathrm{Dom}(g)$ and $f(a) = g(a)$ for every $a \in \mathrm{Dom}(f)$. If $f \subseteq g$ we also say that f is the *projection* of g to $\mathrm{Dom}(f)$.

Now we can state the k-consistency algorithm.

1. Given structures $\mathbf{A}$ and $\mathbf{B}$;
2. Let H be the collection of all partial solutions f with $|\mathrm{Dom}(f)| \leq k+1$;
3. For every f in H with $|\mathrm{Dom}(f)| \leq k$ and every $a \in A$, if there is no g in H such that $f \subseteq g$ and $a \in \mathrm{Dom}(g)$, remove f and all its extensions from H;
4. Repeat step 3 until H is unchanged;
5. If H is empty reject, else accept.

There are several different but equivalent ways of defining the k-consistency algorithm. Our formulation is inspired by the *existential* $(k+1)$*-pebble game* of Kolaitis and Vardi [19]. The connection between the two concepts is due to Kolaitis and Vardi [19].

Observe that it is possible to run the algorithm in time polynomial in $|A|^{k+1}|B|^{k+1}$ because the size of H in step 2 is bounded by that number, and each iteration removes at least one partial solution. Note that for fixed k, this is time polynomial in the size of the input. However, if k is part of the input, the problem of deciding if the k-consistency test accepts on a given instance is EXP-complete (see [17]).

It is obvious that for any satisfiable instance $\mathbf{A}, \mathbf{B}$ and any $k \geq 1$, the k-consistency test accepts. The converse is not necessarily true. It holds, for example, if k is as large as the cardinality of the universe of $\mathbf{A}$ but it might fail for smaller values of k (a counterexample easy to verify is given by fixing $\mathbf{A} = K_{k+2}$ and

$\mathbf{B} = K_{k+1}$, where K_n is the clique with n vertices). The identification of those cases for which the converse is also true is of great interest as it would allow to use the k-consistency test alone in order to decide the existence of a solution.

Definition 3 *Let $\mathfrak{A}$ and $\mathfrak{B}$ be families of relational structures and let $k \geq 1$. We say that k-consistency solves* $\mathrm{CSP}(\mathfrak{A}, \mathfrak{B})$ *if for every $\mathbf{A} \in \mathfrak{A}$ and $\mathbf{B} \in \mathfrak{B}$ on which the k-consistency test accepts, there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

The vast majority of the CSP literature assumes that either $\mathfrak{A}$ or $\mathfrak{B}$ is the set of all structures. Although some rather limited results have been obtained in the most general case, a serious attack of this problem seems rather challenging and out of reach by the known techniques.

Observe that k-consistency solves $\mathrm{CSP}(*, \mathfrak{B})$ if and only if it solves $\mathrm{CSP}(*, \mathbf{B})$ for every $\mathbf{B} \in \mathfrak{B}$. A similar observation can be made for every $\mathfrak{A}$. Consequently, the two main open problems in this research direction can be formulated in the following way:

Problem 4 **(k-width problem)** *Characterize all structures $\mathbf{A}$ for which k-consistency solves* $\mathrm{CSP}(\mathbf{A}, *)$. *Any such structure is called a k-width structure. We also say that $\mathbf{A}$ has k-width.*

Problem 5 **(width-k problem)** *Characterize all structures $\mathbf{B}$ for which k-consistency solves* $\mathrm{CSP}(*, \mathbf{B})$. *Any such structure is called a width-k structure. We also say that $\mathbf{B}$ has width-k.*

For some particular cases, having width-k for some $k > 1$ is the only reason for polynomial time decidability of a problem. For example, a celebrated result of Hell and Nesetril [14] asserts that, for a graph H, if H is bipartite then H-COLORING is tractable via the 2-consistency algorithm, and if H is non-bipartite then H-COLORING is NP-complete. Later, Nesetril and Zhu proved, without assuming P $\neq$ NP, that a finite graph H has width-2 if and only if H is bipartite,

A similar statement is not true in the general case of $\mathrm{CSP}(*, \mathfrak{B})$ [9], and even in the case of H-COLORING where H is a digraph [2]: there are constraint-satisfaction problems that are solvable in polynomial time, but not by establishing consistency at any level. An example of this is the problem of checking the consistency of systems of linear equations.

Characterizing those structures having width-k, is a long standing open problem [8], whose solution is only known in a few particular cases of classes of 2- and 3-element structures [21, 4], and of *conservative* structures [5].

4 Main result

This paper addresses and solves completely, in conjunction with [6], the k-width problem. The following sufficient condition for a structure to have k-width was identified in [6]:

Theorem 6 ([6]) *Let $\mathbf{A}$ be a structure and let $k \geq 1$. If $\mathsf{core}(\mathbf{A})$ has treewidth at most k then $\mathbf{A}$ has k-width.*

Here we prove the exact converse.

Theorem 7 *Let $\mathbf{A}$ be a structure and let $k \geq 1$. If $\mathbf{A}$ has k-width then $\mathsf{core}(\mathbf{A})$ has treewidth at most k.*

Before we prove this, it will be convenient to define the existential pebble game and state the connection with the k-consistency test first pointed out by Kolaitis and Vardi [19].

The existential k-pebble game is played between two players, the *Spoiler* and the *Duplicator*, on two relational structures $\mathbf{A}$ and $\mathbf{B}$ in accordance with the following rules: on the first round of the game the

Spoiler places pebbles on some elements $a_1, \dots, a_k$ of $\mathbf{A}$, and the Duplicator responds by placing pebbles on elements $b_1, \dots, b_k$ of $\mathbf{B}$; on every further round the Spoiler removes a pebble and places it on another element of $\mathbf{A}$, the Duplicator responds by moving the corresponding pebble on $\mathbf{B}$. The Spoiler wins if at the end of some round the mapping $a_i \mapsto b_i$, $1 \leq i \leq k$, is not a partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$. The Duplicator wins if he has a *winning strategy*, i.e. a systematic way that allows him to sustain playing "forever".

Although this presentation of the existential k-pebble game is certainly very intuitive, it is customary and generally simpler to work with an equivalent "algebraic" definition of the game. The key notion here is that of winning strategy for the Duplicator.

Definition 8 *A* winning strategy for the Duplicator in the existential k-pebble game between $\mathbf{A}$ and $\mathbf{B}$ *is a nonempty collection $\mathcal{H}$ of partial homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ satisfying the following conditions: (a) (*restriction condition*) if $f \in \mathcal{H}$ and $g \subseteq f$, then $g \in \mathcal{H}$; (b) (*extension condition*) if $f \in \mathcal{F}$, $|\mathrm{Dom}(f)| < k$, and $a \in A$, there is $g \in \mathcal{H}$ such that $f \subseteq g$ and $a \in \mathrm{Dom}(g)$.*

Such a set can be found by starting with the collection of all partial homomorphisms on subsets of at most k elements, and then removing homomorphisms that do not satisfy one of conditions (a) or (b). Note that this is exactly what the algorithm of the $(k-1)$-consistency test does. Now, it is not difficult to see [19] that the k-consistency algorithm constructs the most general, i.e., largest, winning strategy for the Duplicator when it exists and reports unsatisfiable when there is no winning strategy. Now we are ready for the proof of the main result:

Proof of Theorem 7: Let $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_n^{\mathbf{A}})$ be a relational structure, and $G = G(\mathbf{A})$ its Gaifman graph. Let $G = (A, E)$.

Since $\mathbf{A}$ has k-width if and only if $\mathsf{core}(\mathbf{A})$ has k-width, we may assume that $\mathbf{A} = \mathsf{core}(\mathbf{A})$.

E is a symmetric and irreflexive binary relation on A. We denote edges of G by unordered pairs $e = \{a, a'\}$. Let $a_0 \in A$ be a distinguished point of A to be defined later. For every $a \in A$, let d_a denote the degree of a in G, and let $e_1^a, \dots, e_{d_a}^a$ be a fixed enumeration of all the edges that are incident on a.

Let $\mathbf{B} = \mathbf{B}(\mathbf{A})$ be the relational structure defined as follows. The set of vertices of $\mathbf{B}$ is the set of all tuples of the form $(a, (b_1, \dots, b_{d_a}))$, where

1. $a \in A$ and $b_1, \dots, b_{d_a} \in \{0, 1\}$,
2. $b_1 + \dots + b_{d_a} \equiv 0 \pmod 2$ if $a \neq a_0$,
3. $b_1 + \dots + b_{d_a} \equiv 1 \pmod 2$ if $a = a_0$.

A tuple $((a^1, (b_1^1, \dots, b_{d_{a^1}}^1)), \dots, (a^n, (b_1^n, \dots, b_{d_{a^n}}^n))$ belong to the (n-ary) relation $R_i^{\mathbf{B}}$ if and only if

1. the tuple $(a^1, \dots, a^n)$ is belongs to $R_i^{\mathbf{A}}$,
2. if ℓ, m, i and j are such that $\{a^\ell, a^m\} = e_i^{a^\ell} = e_j^{a^m}$, then $b_i^\ell = b_j^m$.

Intuitively, each vertex of $\mathbf{B}$ is an assignment of $0/1$-values to the edges of $G(\mathbf{A})$ incident to a given point $a \in A$, and the tuples from relations of $\mathbf{B}$ encode the constraints that the assignments of values to the edges of two adjacent points $a, a' \in A$ must be consistent. Note that the first projection $\pi : B \to A$, defined by $\pi((a, (b_1, \dots, b_{d_a}))) = a$ is a homomorphism from $\mathbf{B}$ to $\mathbf{A}$.

Lemma 9 *If $\mathbf{A}$ is a core, then there is no homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

Proof: Suppose that $\mathbf{A}$ is a core, and suppose for contradiction that $h : A \to B$ is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Let $\pi : B \to A$ be the first projection. Composing, $f = h \circ \pi$ is a homomorphism from $\mathbf{A}$ to $\mathbf{A}$, and since $\mathbf{A}$ is a core, it must be an automorphism. Now, let $g = h \circ f^{-1}$ and note that g is still a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ with the additional property that $g(a) = (a, (b^a_1, \dots, b^a_{d_a}))$ for some $b^a_1, \dots, b^a_d \in \{0, 1\}$ and every $a \in A$.

Now, for every edge $e = \{a, a'\}$ of $G(\mathbf{A})$, define $x_e = b^a_i = b^{a'}_j$, where i and j are such that $e^a_i = e$ and $e^{a'}_j = e$. The equality $b^a_i = b^{a'}_j$ follows from the fact that g is a homomorphism. Now, we have

$$x_{e^a_1} + \cdots + x_{e^a_{d_a}} \equiv 0 \pmod 2$$

for every $a \neq a_0$, and

$$x_{e^a_1} + \cdots + x_{e^a_{d_a}} \equiv 1 \pmod 2$$

for $a = a_0$. Since every edge of $G(\mathbf{A})$ has exactly two end-points, adding up all equations we get

$$2 \sum_e x_e \equiv 1 \pmod 2;$$

a contradiction. $\square$

We need an alternative definition of treewidth. Let $G = (V, E)$ be a graph. We say that two sets $B, C \subseteq V$ *touch* if either they intersect or there is an edge of G with an end-point in B and the other in C. A *bramble* is a collection $B_1, \dots, B_r$ of pairwise touching connected subsets of G. A *cover* of this bramble is a set of points that intersects every B_i. The *order* of a bramble is the minimum size of its covers. Seymour and Thomas [22] proved that a connected graph has treewidth at least k if and only if it has a bramble of order at least $k + 1$.

Lemma 10 *If the treewidth of $\mathbf{A}$ is at least $k + 1$, then the Duplicator wins the existential $k + 1$-pebble game on $\mathbf{A}$ and $\mathbf{B}$.*

Proof: We start with some definitions. For every walk $P = (a_0, a_1 \dots, a_r)$ in $G(\mathbf{A})$ that starts at a_0 and for every edge e of $G(\mathbf{A})$, we define

1. $x^P_e = 1$ if e appears an odd number of times in P,
2. $x^P_e = 0$ if e appears an even number of times in P.

Now we define $h^P(a) = (a, (x^P_{e^a_1}, \dots, x^P_{e^a_{d_a}}))$ for every $a \in A$.

Claim 1 *If $P = (a_0, a_1, \dots, a_r)$ is a walk in $G(\mathbf{A})$ that starts at a_0 and $a \neq a_r$, then $h^P(a)$ belongs to B.*

Proof: Suppose that $P = (a_0, a_1, \dots, a_r)$ is a walk in $G(\mathbf{A})$ that starts at a_0, and let $a \neq a_r$. We need to check that

$$x^P_{e^a_1} + \cdots + x^P_{e^a_{d_a}} \equiv 0 \pmod 2 \tag{1}$$

if $a \neq a_0$, and

$$x^P_{e^a_1} + \cdots + x^P_{e^a_{d_a}} \equiv 1 \pmod 2 \tag{2}$$

if $a = a_0$. Suppose first that $a \neq a_0$. Let $i_1 < \cdots < i_s$ be the enumeration of all positions i in the walk $P = (a_0, a_1, \dots, a_r)$ with $a_i = a$. Since the walk does not start or end at a, we have $0 < i_1 < \cdots < i_s < r$. It follows immediately that the total number of occurrences of edges in the walk that are incident on a is even (we are using the fact that $G(\mathbf{A})$ has no self-loops so for every $1 \leq j \leq r-1$ there exists l such that $i_j < l < i_{j+1}$). Thus, equation (1) holds. Suppose now that $a = a_0$. Again, let $i_1 < \cdots < i_s$ be the enumeration of all positions i in the walk such that $a_i = a$. Since the walk starts at $a_0 = a$ and does not end at a, we have $0 = i_1 < \cdots < i_s < r$. It follows immediately that the total number of occurrences of edges in the walk that are incident on a is odd. Thus, equation (2) holds. □

Claim 2 *Let $S \subseteq A$, and let $P = (a_0, a_1, \ldots, a_r)$ be a walk in $G(\mathbf{A})$ that starts at a_0 and does not end in a point in S. Then, the restriction $h^P|_S$ of h^P to S is a partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$.*

Proof: The previous claim guarantees that $h^P(a)$ belongs to B for every $a \in S$. We need to check now that for every (n-ary) relation $R_i^{\mathbf{A}}$ and any tuple $(b_1, \ldots, b_n) \in R_i^{\mathbf{A}}$ such that $b_1, \ldots, b_n \in S$, the tuple $(h^P(b_1), \ldots, h^P(b_n))$ is also a tuple from $R_i^{\mathbf{B}}$. But this is obvious because for any j and ℓ if s and t are such that $e_s^{b_j} = e_t^{b_\ell} = \{b_j, b_\ell\}$, then trivially $x^P_{e_s^{b_j}} = x^P_{e_t^{b_\ell}} = x^P_{\{b_j, b_\ell\}}$. □

Now we can define the winning strategy for the Duplicator in the existential $k+1$-pebble game between $\mathbf{A}$ and $\mathbf{B}$. Suppose that the treewidth of $\mathbf{A}$ is at least $k+1$. Let $\{B_1, \ldots, B_r\}$ be a bramble of $\mathbf{A}$ of order at least $k+2$. It is finally the time to define a_0. Let us fix a_0 to be any point of A connected to the bramble. We define a collection of partial homomorphisms $\mathcal{H}$ as follows: for any walk P in $G(\mathbf{A})$ that starts with a_0 and any $S \subseteq A$ such that $|S| \leq k+1$ and the last vertex of P belongs to a bag of the bramble that does not contain any element of S, we include $h^P|_S$ into $\mathcal{H}$. By the claims above, each such $h^P|_S$ is indeed a partial homomorphism. The election of a_0 guarantees that $\mathcal{H}$ is nonempty. Conditions (a),(b) from the definition of a winning strategy can be easily checked. Condition (a) holds trivially. For (b), let h_S^P be any function with $|S| \leq k$ in $\mathcal{H}$. Hence P is a walk that starts at a_0 and ends at a point, say a_r, that sits in a bag B_i that does not contain any point in S. Now let $a \in A$. Let B_j be a bag of the bramble that does not contain any point in $S' = S \cup \{a\}$. Such a bag also exists because $|S'| \leq k+1$ and the bramble cannot be covered with less than $k+2$ points. Since all pairs of bags of the bramble touch, and since bags are connected, there must be a walk Q from a_r to a point in B_j that runs entirely inside B_i except for the last point, which lands in B_j. Now we let P' be the concatenation of P with Q. The walk P' has the properties we need: it starts at a_0 and it ends in a point that belongs to a bag B_j of the bramble that does not contain any point in S'. Thus $h^{P'}|_{S'}$ belongs to $\mathcal{H}$. Finally, since the only edges that P' adds to P are edges that are entirely inside B_i except for the last that may land inside B_j, none of these edges has an end-point in S because both B_i and B_j are S-free. It follows that $x_e^P = x_e^{P'}$ for every edge e with an end-point in S, so $h^{P'}(a) = h^P(a)$ for every $a \in S$. □

5 Further Comments and Remarks

If we put together Theorem 6 and Theorem 7 we obtain that $\mathbf{A}$ has k-width if and only if core($\mathbf{A}$) has treewidth at most k. In turn, it was proved in [6] that for every fixed $k \geq 1$, it is an NP-complete problem to decide if a given structure has a core of treewidth at most k. This implies that it is an NP-complete problem to decide if a given structure has k-width. Before our result, it was not even known whether this problem was decidable. Dually, it is an important open problem whether it is decidable if a given structure has width-k.

The second remark is about an application of our result to preservation theorems in finite model theory. Let us first note that, for a given a core $\mathbf{A}$ of treewidth at least k, the proof of our main result provides a structure $\mathbf{B}$ with the following three properties: $\mathbf{B} \to \mathbf{A}$, $\mathbf{A} \not\to \mathbf{B}$, and the Duplicator wins the existential k-pebble game on $\mathbf{A}$ and $\mathbf{B}$. Using these three properties, it is possible to solve an open problem in [3]. The problem asked whether every sentence of first-order logic that can be written equivalently as an existential-positive infinitary sentence with k variables on finite structures is also equivalent to a existential-positive finite sentence with k variables on finite structures. The construction above, in combination with Rossman's Theorem [20], provides the right tool to establish this preservation theorem. We will provide the details of the proof in the journal version of this paper.

It would be interesting to see if the construction we give has more applications in finite model theory. Interestingly, up to now we have used it to establish both a *negative* result (limits on the k-consistency algorithm), and a *positive* result (a preservation theorem in finite model theory).

References

[1] A. Atserias and V. Dalmau. A Combinatorial Characterization of ResolutionWidth. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 239-247, Aarhus, Denmark, July 2003.

[2] A. Atserias. On digraph coloring problems and treewidth duality. In *Proceedings of the 20th Annual IEEE Simposium on Logic in Computer Science*, pages 106–115, Chicago, USA, June 2005.

[3] A. Atserias, A. Dawar, and P. Kolaitis. On Preservation under Homomorphisms and Unions of Conjunctive Queries. *Journal of the ACM*, 53(2):208–237, 2006.

[4] A. Bulatov. A dichotomy theorem for constraints on a three-element set. *J. of the ACM*, 53(1):66–120, 2006.

[5] A.A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th Annual IEEE Simposium on Logic in Computer Science*, pages 321–330, Ottawa, Canada, June 2003. IEEE Computer Society.

[6] V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite variable logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, CP'02*, Lecture Notes in Computer Science, pages 311–326. Springer-Verlag, 2002.

[7] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, 1992.

[8] T. Feder and M.Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of 25th ACM Symposium on the Theory of Computing (STOC)*, pages 612–622, 1993.

[9] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal of Computing*, 28:57–104, 1998.

[10] E. Freuder. A Sufficient Condition for Backtrack-Free Search. *Jouranl of the ACM*, 29(1):24–32, 1982

[11] E. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence AAAI-90*, pages 4–9, 1990

[12] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 44th Annual Simposium on Foundations of Computer Science*, pages 552–561, Cambridge, Massachusets, USA, October 2003. IEEE Computer Society.

[13] P. Hell and Nešetřil. *Graphs and homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2004.

[14] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Ser.B*, 48:92–110, 1990.

[15] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1-3):117–126, 1992.

[16] P. Hell, J. Nešetřil, and X. Zhu. Duality and polynomial testing of tree homomorphisms. *Trans. of the AMS*, 348(4):1281–1297, 1996.

[17] Ph.G. Kolaitis and J. Panttaja. On the Complexity of Existential Pebble Games. In *Proceeding of the 17th International Workshop on Computer Science Logic*, pages 314–329, 2003.

[18] Ph.G. Kolaitis and M.Y. Vardi. On the expressive power of Datalog: tools and case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995.

[19] Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proceedings of the 17th National (US) Conference on Artificial Intelligence, AAAI'00*, pages 175–181, 2000.

[20] B. Rossman. Existential Positive Types and Preservation under Homomorphisms. In *Proceedings of the 20th IEEE Symposium on Logic in Computer Science*, pages 467-476, 2005.

[21] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226, 1978.

[22] P. Seymour and R. Thomas. Graph searching, and a min-max theorem for treewidth. *Journal of Combinatorial Theory, Series B*, 58:22–23, 1993.