FO-Eigenschaften von Strukturen mit lokaler Baumzerlegung

Gunnar Lüders

January 14, 2022

1 Introduction

There are many examples of problems that are in general considered hard but when restricted to structures with a specific property are efficiently computable. For example 3-Colorability, a NP-complete problem, can be solved in linear time, when restricted to graphs with bounded tree-width [1].
A more general example would be Courcelle's theorem stating that properties definable through monadic second-order logic can be decided in linear time on graphs of three-width at most w[2], that is, the running time is restricted depending on how similar the input graph is to a tree.
Following that Seese gave a theorem expanding this concept to first-order logic, stating that for every $l \geq 1$ and for every first-order definable property of structures there is a linear time algorithm that decides whether a given structure of valence at most l has this property [7].
An other example is that a local neighborhood of a vertex in a planar graph has tree-width bounded by a number only depending on the radius of this neighborhood. This allows us to compute in linear time a family of subgraphs of bounded tree-width such that a suitably big neighborhood of every vertex is completely contained in one of these subgraphs. We define classes of graphs with this property as being *locally tree-decomposable*. An exact definition is given in section 3.6. This leads to our central theorem:

Theorem 1.1. *Let C be a class of relational structures that is locally tree-decomposable and φ a property definable in first-order logic. Then there is a linear time algorithm deciding whether a given structure $\mathcal{A} \in C$ has property φ.*

A few examples of of first-order logic definable properties are k-dominating-set, $\mathcal{H}$-subgraph-isomorphism and $(\mathcal{H}, \mathcal{K})$-extension.
But as mentioned in Theorem 1.1., our result goes further and expands to arbitrary relational structures. A few examples of problems on other relational structures are k-set-cover or (k, d)-circuit-satisfiability. A very practical and naturally occurring problem is the evaluation of (Boolean) database queries formulated in relational calculus on relational databases. Following theorem 1.1. these can be evaluated in linear time if the underlying database graph is planar.
A very closely related theorem is the following, which applies to the even more general context of classes of structures of bounded local tree-width:

Theorem 1.2. *Let C be a class of relational structures of bounded local tree-width and φ a first-order definable property. Then for every $k \geq 1$ there is an algorithm deciding whether a given structure $\mathcal{A} \in C$ has property φ in time $O(n^{1+(1/k)})$*

For the sake of simplicity we restrict our considerations in the following to graphs but keep in mind, that these apply to arbitrary relational structures as well.

2 The General Idea

At its core we are solving a model checking problem. We are given a graph and a fixed first-order formula φ describing a specific property and are interested in whether the structure has this property. The basis for our algorithm is Gaifman's Theorem.

Theorem 3.3.1 (Gaifman [5]). *Every first-order sentence is equivalent to a Boolean combination of sentences of the form*

$$\exists x_1 ... \exists x_k (\bigwedge_{1 \leq i < j \leq k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \leq i \leq k} \psi(x_i))$$

for suitable $r, k \geq 1$ and an r-local $\psi(x)$

Being a r-local formula means, whether $\varphi(x)$ holds at x only depends on the r-neighborhood of x. The r-neighborhood is simply defined as the set of nodes, that are reachable within r steps: $N_r^{\mathcal{A}}(a) := \{b \in A | d^{\mathcal{A}}(a,b) \leq r\}$ where $d^{\mathcal{A}}(a,b)$ is the length of the shortest path in $\mathcal{A}$ connecting a and b. This means that for finite relational structures a first-order formula can really just make statements about local properties. This is useful to us, because it allows us to rather then checking a property for a whole structure, in our case a graph, we can check for properties on restricted parts of the graph and still make a statement about the whole structure. This also explains, why our formula φ must be fixed. Depending on how the Gaifman-combination looks like we have to work with a different r moving forward. Secondary every r-local formula is combined with a distance condition stating, that in the set of nodes that satisfy the r-local formula there must be k nodes that have a distance of $d^{\mathcal{A}}(a,b) \geq 2r$, meaning that there r-neighborhoods do not overlap. For more details see [5]. Now recall Courcelles's Theorem:

Theorem 3.3.2 (Courcelle [2]. *Let $w \geq 1$. Then for every sentence φ of monadic second-order logic there is a linear time algorithm that decides whether a given structure $\mathcal{A}$ of tree-width at most w satisfies φ.*

Notice that the theorem applies for first-order logic as well. So it is possible to decide in linear time if a structure of bounded tree-width satisfies a formula. Now combining the realizations of Gaifman's Theorem and Courcelle's Theorem we can see that if we can guaranty, that the r-local formulas are checked on parts of the graph that have some decomposition of bounded tree-width, we can indeed model check these parts in linear time, allowing us to make a statement about the whole structure. This leads to an outline of an algorithm proving Theorem 1.2. In section 4 we will see, that we can easily derive an algorithm for Theorem 1.1 from this one:

(1) We can assume without loss of generality that the formula has the form described in Gaifman's theorem.

(2) Compute r-neighborhood for every node on the input graph.

(3) Check for every r-local ψ if there exists a node that with its r-neighborhood satisfies ψ.

(4) Check if all fulfilling nodes are at least $2r$ steps apart.

3 Neighborhood/Tree-Cover And Tree-Decomposable

A way to compute the r-neighborhoods that we need is to calculate a *neighborhood cover*. In fact, at times, this approach delivers more than we actually need. In these cases a *tree cover* suffices. But before we can look at the details recall a few mandatory facts about trees and especially tree-width:

Definition 3.1. A $tree - decompostition$ of a τ-structure $\mathcal{A}$ is a pair $(\mathcal{T}, (B_t)_{t \in T}$, where $\mathcal{T}$ is a tree and $(B_t)_{t \in T}$ a family of subsets of A (called the *blocks* of the decomposition) such that

(1) For every $a \in A$, the set $\{t \in T \mid a \in B_t\}$ is non-empty and connected in $\mathcal{T}$ (that is, induces a subtree).

(2) For every $R \in \tau$ and all $\overline{a} \in R^{\mathcal{A}}$ there is a $t \in T$ such that $\overline{a} \in B_t$.

Now we define the $width$ of a tree-decomposition as $\max\{|B_t| \mid t \in T\} - 1$ and the $tree - width$ $\mathrm{tw}(\mathcal{A})$ of $\mathcal{A}$ as the minimal width of a tree-decomposition of $\mathcal{A}$.

A more intuitive way to think about tree-width is to look at it as an indicator for how similar a graph is to a tree. For example trees as well as forests have tree-width of 1. The higher the tree-width the less the graph resembles a tree.

Definition 3.2.

(1) The *local tree-width* of a structure $\mathcal{A}$ is the function $\mathrm{ltw}^{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathrm{ltw}^{\mathcal{A}}(r) := \max\{\mathrm{tw}(\langle N_r^{\mathcal{A}}(a)\rangle) \mid a \in A\}.$$

(2) A class C of structures has *bounded local tree-width* if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{ltw}^{\mathcal{A}}(r) \leq f(r)$ for all $\mathcal{A} \in \mathrm{C}$, $r \in \mathbb{N}$.

When considering subgraphs one can of course look at the tree-width of those. The local tree-width is defined as the maximum of specific subgraphs, namely all r-neighbourhood-induced subgraphs $\langle N_r^{\mathcal{A}}(a)\rangle$. If, when raising the neighborhood radius r, the local tree-width does not grow faster than a function $f(r)$ then the graph is defined to be of *bounded* local tree-width. With these facts in mind we first define *neighborhood covers*:

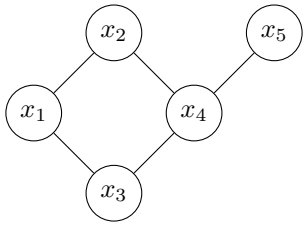
Definition 3.3. Let $r, s \geq 0$. An (r, s)-*neighborhood cover* of a structure $\mathcal{A}$ is a family $\mathcal{N}$ of subsets of A with the follwoing properties:

(1) For every $a \in A$ there exists a $N \in \mathcal{N}$ such that $N_r^{\mathcal{A}}(a) \subseteq N$.

(2) For every $N \in \mathcal{N}$ there exists an $a \in A$ such that $N \subseteq N_s^{\mathcal{A}}(a)$.

Remember, that r-neighborhood is the set of nodes, that are reachable within r steps. A cover usually means it contains either all edges or all vertices of a graph. An (r, s)-neighborhood cover is a collection of subsets that *covers* the whole graph in the sense that for every node their respected r-neighborhood is contained in at least one of these subsets. Additionally each of these subsets must not contain more than some nodes s-neighborhood. One can think of the subsets of being "in between" the r- and s-neighborhoods of the nodes.

Example 3.3.1:



In this example one possible $(1, 2)$-neighborhood cover is $\mathcal{N} = \{\{x_1, x_2, x_3, x_4\}, \{x_5, x_4, x_2, x_3\}\}$.

Definition 3.4. Let $r, w \geq 0$. An (r, w)-tree cover of a structure $\mathcal{A}$ is a family $\mathcal{T}$ of subsets of A with the following properties:

(1) For every $a \in A$ there exists a $T \in \mathcal{T}$ such that $N_r^{\mathcal{A}}(a) \subseteq T$.

(2) For every $T \in \mathcal{T}$ we have $\mathrm{tw}(\langle T\rangle^{\mathcal{A}}) \leq w$.

Tree covers function very similar to neighborhood covers. While (1) is exactly the same a tree cover requires that the tree-width of each subset is not greater than some w. Note that a (r, s)-neighborhood cover of a structure $\mathcal{A}$ is a $(r, \mathrm{ltw}^{\mathcal{A}}(s))$-tree cover of $\mathcal{A}$ since local tree-width is defined through neighborhoods. Now due to Peleg [6] and Eppstein [3] we can compute a neighborhood-cover as well as a tree-cover efficiently, making it useful for the use in our algorithm. The corollary given here is an adaptation of the algorithm given by Peleg [6].

Corollary 3.5. *Let $k \leq 1$, τ a vocabulary, and C a class of bounded local tree-width. Then there is an algorithm that, given a structure $\mathcal{A} \in C$, computes an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$ of size $\|\mathcal{N}\| = O(|A|^{1+(1/k)})$ in time $O(|A|^{1+(1/k)})$.*

The idea behind the algorithm is, starting with the full vertex set, the algorithm picks one arbitrary vertex a and computes increasing neighborhoods of a until it reaches the specified size. Then the resulting set is added to the cover and all vertexes, that have there r-neighborhood already covert by this set, are removed from the starting set of all vertexes. Repeat until the starting set is empty.

Lemma 3.6 (Eppstein[3]). *Let $r \geq 0$ and C be a class of graphs that is closed under taking minors and has bounded local tree-width. Let $f : \mathbb{N} \to \mathbb{N}$ be a function bounding the local tree-width of the graphs in C.*

Then there is an algorithm that, given a graph $\mathcal{G} \in C$, computes an $(r, f(2r+1))$-tree cover $\mathcal{T}$ of $\mathcal{G}$ of size $\|\mathcal{T}\| = O(|G|)$ in time $O(|G|)$

Start again by choosing an arbitrary vertex a from the set of all vertexes. Now let $G[i, j]$ be the set containing all vertexes that are reachable in not less than i steps and no more than j steps. If i is 0 or 1 than this set is contained in the j-neighborhood of a and in every other case we can contract the subgraph $\langle G[0, i-1]\rangle$ to a vertex b so that the resulting minor contains the set $G[i, j]$ and is also contained in the $(j - i + 1)$-neighborhood of b. Therefor the tree-width of every set $G[i, j]$ is smaller then some function $f(j - i + 1)$, thus can be used as part of a tree cover. Now we can, By using breadth-first search, compute a tree cover in linear time. We use this consideration to define *locally tree-decomposeable* as a property of graphs where the computation of a tree cover is possible in this way.

Definition 3.7. A class C of graphs is *locally tree-decomposable* if there is a function $g : \mathbb{N} \to \mathbb{N}$ and an algorithm that, given a structure $\mathcal{A} \in C$ and an $r \in \mathbb{N}$, computes an $(r, (g(r))$-tree cover of $\mathcal{A}$ of size $O(|A|)$ in time $O(|A|)$.

4 The Complete Algorithm

At this point we are ready to look at the complete algorithm in pseudo code and discuss some appearing questions. Notice that this algorithm computes a neighborhood cover so it is associated with theorem 1.2. An algorithm for theorem 1.1 can be achieved by simply replacing the first line with a computation of a tree cover:

Algorithm 1 Input: $\mathcal{A} \in C$

1: compute an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$ of size $O(A^{1+(1/k)})$
2: **for all** $N \in \mathcal{N}$ **do**
3: compute $K_N := \{a \in N | N_r^{\mathcal{A}}(a) \subseteq N\}$
4: **end for**
5: **for all** $N \in \mathcal{N}$ **do**
6: compute $P_N := \{a \in K_N | \langle N\rangle^{\mathcal{A}} \models \psi(a)\}$
7: **end for**
8: compute $P := \bigcup_{N \in \mathcal{N}} P_N$
9: **if** there are $a_1, ..., a_m \in P$ such that $d(a_i, a_j) > 2r$ for $1 \leq i < j \leq k$ **then**
10: ACCEPT
11: **else**
12: REJECT
13: **end if**

It should attract attention, that there appear some extra steps we have not considered so far, namely in line 3,8 and 9. In the first line we calculate a neighborhood cover. We now know that

we can compute a neighborhood of that specific size in time $O(|A|^{1+(1/k)})$. Remember that this is a family of subsets so for each node the set containing its r-neighborhood might be a different set. Moving forward we need to know which set holds the neighborhood for each node. Therefore in line 3 we calculate which set corresponds to which node. We introduce the following lemma (Frick, Grohe[4]):

Lemma 4.1. *Let C be a class of τ-structures of bounded local tree-width and $r, w \geq 1$. Then there is an algorithm that solves the following problem in time $O(\|\mathcal{T}\|)$:*

Input: Structure $\mathcal{A} \in$ C, (r, w)-tree cover $\mathcal{T}$ of $\mathcal{A}$. *Problem:* Compute $K_T := \{a \in A \mid N_r^{\mathcal{A}}(a) \subseteq T\}$ for all $T \in \mathcal{T}$

Notice that, while the lemma specifically applies to tree covers, it also applies to neighborhood covers because of there close relation mentioned in section 3. For every $T \in \mathcal{T}$ we start with the set $K := T$, then for every node calculate every i-neighborhood from $i = 1$ to $i = r$, removing nodes whose neighbors are not all contained in T. At the end of this calculation we precisely obtain $K = \{a \in T \mid N_r^{\mathcal{A}}(a) \subseteq T\} = K_T$. This can be done in linear time.

In line 6 we apply Courcelles's Theorem. Since we have a set for each node which is holding its r-neighborhood we can check in linear time if this neighborhood satisfies the r-local formula ψ. We do this for each set we calculated in line 3. Line 8 joins all the nodes that satisfy ψ. This obviously can be done in linear time. Lastly it is to check if under the fulfilling nodes there are enough nodes that are a distance of $2r$ apart, as required by Gaifman's Theorem. The following lemma states precisely this problem [4]:

Lemma 4.2. *Let C be a class of structures of bounded local tree-width and $r, m \geq 1$. Then the following problem can be solved in time $O(|A|)$:*

Input: Structure $\mathcal{A} \in$ C, set $P \subseteq A$. *Problem:* Decide if there exists $a_1, ..., a_m \in P$ such that $d^{\mathcal{A}}(a_i, a_j) > r$.

The algorithm proceeds in two phases: Starting with a set of nodes P we count how often we can remove a complete r-neighborhood of an arbitrary node from the set until the set is empty. This gives us a lower bound l of how many nodes are more than r steps apart. Since we need an specified amount of nodes satisfying this condition we may already reach that amount or find none. Then we accept or reject accordingly. If we find more than one but not enough nodes the algorithm proceeds with the next phase. Now we construct a graph $\mathcal{H} := \langle N_{2r}^{\mathcal{A}}(\{a_1, ..., a_l\})\rangle$ of bounded tree-width. Because P is a subset of of $N_r^{\mathcal{A}}(\{a_1, ..., a_l\})$ every path of length at most r in P is also contained in H. We can now check linear time, again by Courcelle's Theorem [2], if there are elements in P that have a distance $> r$ in $\mathcal{H}$ and therefore in $\mathcal{A}$.

5 Running Time

Now we can analyze the overall running time of our algorithm since this ultimately proves our stated Theorems 1.1 and 1.2. We can summarize:

(1) We can assume without loss of generality that the formula has the form described in Gaifman's theorem [5].

(2) With help of Peleg [6] and Eppstein [3] we have seen, that we can compute a sufficient neighborhood cover in time $O(|A|^{1+(1/k)})$ or, if applicable, a tree cover in linear time.

(3) As an intermediate step we can calculate sets of all nodes, whose neighborhoods are covered by a specific set derived from our previous calculated cover in linear time.

(4) From Courcelle [2] we know, that we can model check a formula on a structure of bounded tree-width in linear time.

(5) Lastly we have seen, that we can test if our calculated set of nodes holds k nodes, that are $2r$ steps apart from each other, satisfying the distance condition set by Gaifman's Theorem, in linear time.

In summary we can see that if the input structure is *locally tree-decomposable*, thus calculating a tree cover, the algorithm runs in linear time. When computing a neighborhood cover in time $O(|A|^{1+(1/k)})$, since this is the slowest running time appearing in the algorithm, it runs in time $O(|A|^{1+(1/k)})$. Therefore Theorem 1.1 and 1.2 hold respectively.
While at first glance this result seem to yield promising results for practical applications this is not the case. For example the transformation of a formula into a sentence of the form suggested by Gaifman may blow up the formula by a non elementary factor [6]. Therefore these results are of theoretical significance as they stand.

References

[1] Hans L Bodlaender. Treewidth: Algorithmic techniques and results. In *International Symposium on Mathematical Foundations of Computer Science*, pages 29–36. Springer, 1997.

[2] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 194–242. Elsevier Science Publishers, 1990.

[3] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, Jun 2000.

[4] Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures, 2000.

[5] Haim Gaifman. On local and non-local properties. In J. Stern, editor, *Proceedings of the Herbrand Symposium*, volume 107 of *Studies in Logic and the Foundations of Mathematics*, pages 105–135. Elsevier, 1982.

[6] David Peleg. Distance-dependent distributed directories. *Information and Computation*, 103(2):270–298, 1993.

[7] Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996.