Courcelle's Theorem

Niclas Gey

Seminar Logic, Complexity, Games:
Algorithmic Meta-Theorems and Parameterized Complexity

Mathematical Foundations of Computer Science
RWTH-Aachen University
Germany

14 January 2022

1 Introduction

There exist many NP-complete graph problems, like 3-Colorability, which under normal circumstances cannot be computed efficiently. Most of those NP-complete problems are not always equally hard, their difficulty depends on the properties of graph they are solved on. If we now restrict these properties of the input graph, we can find ways to efficiently compute NP-complete problems on specific classes of graphs. One commonly used graph property is tree width, which is a parameter that describes how similar a given graph is to a tree. By restricting our inputs to graphs of bounded tree width we can not only solve specific problems but whole classes of problems particularly all problems expressible in monadic second order logic. These types of algorithms that define a way to compute whole range of problems are called algorithmic meta-theorems. This paper constructs an easy to understand proof of Courcelle's theorem and the proofs leading up to it focusing only on the class of *undirected graphs.* This proof is based on the approaches by Flum and Grohe in [4] and Kreutzer in [5].

1.1 Tree Width

The tree width of an undirected graph is a integer value, which informally tells us how similar a graph is to a tree. Formally it is the width of a graphs tree decomposition.

Definition 1.1. [2] A tree decomposition of a graph $\mathcal{G} = (V, E)$ is a tuple $\mathcal{D} = (\mathcal{T}, (B_t)_{t \in T})$ in which $\mathcal{T} = (T, F)$ is a undirected tree and for each vertex $t \in T$ the *bag* B_t is a subset of V. Every tree decomposition fulfils the following conditions:

1. The union of all bags is V, $\bigcup\limits_{t \in T} B_t = V$.
2. If a $v \in V$ is contained in B_i and B_j, then it is contained in every B_k on the path form i to j.
3. For every edge $(v, w) \in E$ there exists a bag B_t with $v, w \in B_t$.

The *width* of a tree decomposition is defined, by the size of the largest bag:

$$width(\mathcal{D}) := max\{|B_t| - 1 \mid t \in T\}$$

The *tree width* $tw(\mathcal{G})$ of a graph $\mathcal{G}$ is the smallest width of all its tree decompositions. We subtract one from the width of a decomposition to assure that trees have a tree width of one.

For the rest of this paper we focus on *ordered tree decompositions* [4, Chap. 11], $\mathcal{D} = (\mathcal{T}, (\bar{b}^t)_{t \in T})$. These are a slight variation of ordinary tree decompositions. We convert $\mathcal{T}$ to be a directed graph with the edges starting in the root and ending in the leaves of the tree. Furthermore, we change the representation of bags from subsets B_t to $k+1$-tuples $\bar{b}^t$ where k is the width of the decomposition. In case a bag contains less than k elements we duplicate the last element until the bag is 'full'.

When given an upper bound k and a graph $\mathcal{G}$ it is possible to effectively compute a tree decomposition of $\mathcal{G}$ of at most width k if it exists, as proven in Bodlaender's Theorem [1].

1.2 Courcelle's Theorem

Theorem 1.2. [3] *Every problem definable in monadic second order logic (MSO) can be decided in linear time on graphs when parametrized by the length of the MSO-sentence and the tree width of the graph.*

We will be referring to the tree width of our graph as k and to the length of the MSO-sentence as l. Thus the problem can be written as: k-l-MC(GRAPH, MSO)

What are simple graph properties we can express in MSO?
We can easily express 3-Colorability, a NP-complete problem:

$$\begin{aligned}\exists F_1 \exists F_2 \exists F_3 \big(&\forall v(F_1 v \lor F_2 v \lor F_3 v) \land \\ &\forall u \forall w \big((F_1 u \land F_1 w) \to \neg Euw\big) \land \\ &\forall u \forall w \big((F_2 u \land F_2 w) \to \neg Euw\big) \land \\ &\forall u \forall w \big((F_3 u \land F_3 w) \to \neg Euw\big)\big)\end{aligned}$$

Besides 3-Colorability we can also express problems like VertexCover, DominatingSet or many other common NP-complete graph problems. All of those problems can be computed in linear time on graphs of bounded tree width.

1.3 Overview of the Proof

The proof of Courcelle's Theorem consists of a series of reductions. We start by using *tree language* and *nondeterministic tree automaton* (NTA) to prove that l-MC(TREE$_{lob}$, MSO), the model checking for MSO on *labeled ordered binary trees*, is fixed parameter tractable when parametrized by the length of the MSO-sentence. Next we expand this algorithm to work on the class of *labeled ordered unranked trees.*

The main part of the proof is the following reduction,

$$k\text{-}l\text{-MC(GRAPH, MSO)} \leqslant l\text{-MC(TREE}_{lo}\text{, MSO)}.$$

This reduction is achieved with a MSO-Interpretation [5, pp. 8-9]. We firstly define a coding, which maps $\mathcal{G}$ and $\mathcal{D}$, an *ordered tree decomposition* of $\mathcal{G}$ to $\mathcal{T}(\mathcal{G}, \mathcal{D})$, a labeled tree, without loosing any information about $\mathcal{G}$. In the next step we translate the MSO-sentence φ over the graph $\mathcal{G}$ into a MSO-sentence φ^* over the labeled tree $\mathcal{T}(\mathcal{G}, \mathcal{D})$, with $\mathcal{G} \models \varphi$, if and only if $\mathcal{T} \models \varphi^*$.

$$\boxed{k\text{-}l\text{-MC(GRAPH, MSO)}} \quad \leqslant \quad \boxed{l\text{-MC(TREE}_{lo}\text{, MSO)}} \quad \leqslant \quad \boxed{l\text{-MC(TREE}_{lob}\text{ , MSO)}}$$

Fig. 1. Overview of the proof

2 Model-Checking on Trees

We start by defining the kind of tree we work with in this section. First we look at *labeled ordered binary trees* (TREE$_{lob}$). These are binary trees, in which we differentiate between the first and second child. Each of those childes is referred to by its own relation $E_1 vw$ and $E_2 vu$. Here v is the parent node, w the first and u the second child.

Theorem 2.1. [4, Chap. 10] *A tree language is regular if and only if it can be expressed in MSO.*

Proof Sketch. Every regular tree language can be decided by a nondeterministic tree automaton (NTA). An NTA is like an NFA but it checks *labeled ordered binary trees* instead of *strings*. The transition relations of an NFA consist of a starting-state, an ending-state and a symbol from the alphabet. In an NTAs transition relation, we add a second starting-state to check trees from the bottom up.

"$\Rightarrow$" For the first part of the proof we simulate an NTA with an MSO-sentence. This is done by using one existentially quantified relation for each state in the NTA. These relations specify for which input node the NTA is in which state. The following conditions are also checked: The NTA can only be in one state at a time. It starts in an initial and ends in an accepting state and it only uses existing transitions.

"$\Leftarrow$" For the backward direction we firstly translate our MSO-sentence φ into an equivalent sentence φ^*. We start by replacing all variables in φ with new unary relations in φ^*. Next we define new atomic formulas, which when using unary relations instead of variables are equivalent to the atomic formulas of MSO. For example, one of these new atomic formulas is $subset(X, Z) = \forall z(Xz \to Yz)$, which replaces the atomic formula Zx. Now we inductively construct φ^* by replacing all atomic formulas in φ with the new ones. Our translated formula φ^* now only contains free relation symbols but no free variables. This makes the construction of the matching automata easier by encoding the quantifiers into a larger alphabet.

To create a equivalent automaton to φ^* we first construct an automaton, equivalent to every new atomic formula. This is achieved by encoding all free relations symbols into a larger alphabet. Let $X_1, \ldots, X_k$ be the free relations of φ^*. The new alphabet is $\Sigma' = \Sigma \times \{0,1\}^k$, where each $(a, b_1, \ldots, b_k) \in \Sigma'$ encodes $a \in \Sigma$ and $b_i = \begin{cases} 1, & \text{if } a \in X_i \\ 0, & \text{else} \end{cases}$. Using this alphabet we construct an automaton, equivalent to every new atomic formula and inductively combine those automata to be equivalent to φ^*.

Corollary 2.2. *The model checking for labeled ordered binary trees parametrized by l, the length of the sentence φ, l-MC(TREE$_{lob}$, MSO) is fixed-parameter tractable and can be decided in linear time.*

Proof. We have just proven, that for each φ we can compute a *nondeterministic tree automaton*, which decides if a given tree satisfies φ. This NTA can be converted into a DTA (deterministic tree automaton), that decides the problem for a given tree in linear time. This conversion results in a exponential blow-up relative to l, the length of the sentence φ. □

To extend the model checking algorithm to work on all labeled trees we define a way to convert a *labeled ordered unranked tree* (TREE$_{lo}$) with its MSO-sentence φ into a *labeled ordered binary tree* (TREE$_{lob}$) with the accordingly translated MSO-sentence φ^*. A labeled ordered unranked tree can be viewed as a tree with arbitrarily many children, where E is the usual edge relation and every child is connected to the previous child by a binary relation N.

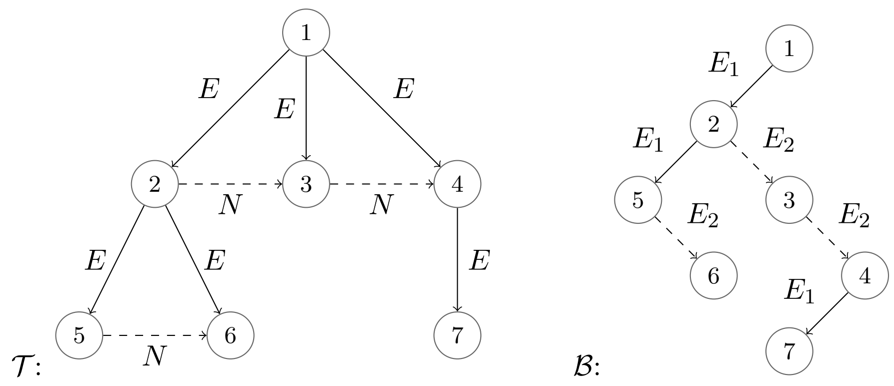

Fig. 2. Conversion from TREE$_{lo}$ to TREE$_{lob}$

Let $\mathcal{T}$ be a labeled ordered unranked tree and $\mathcal{B}$ its translated labeled ordered binary tree. To translate the MSO-sentence φ we define a new sentence φ^* with $\mathcal{T} \models \varphi$ if and only if $\mathcal{B} \models \varphi^*$. We first replace all instances of Nvw in φ with E_2vw in φ^*. To replace Evw in φ we need to define a new formula $edge(v, w)$.

$$edge(v,w) = \exists u\Big(E_1vu \land \forall U\Big(\big(Uu \land \forall x\forall y\big((E_2xy \land Ux) \to Uy\big)\big) \to Uw\Big)\Big)$$

We define this formula by checking if either w is the first child of v or if w is part of a series of "next children", which starts with a first child of v.

3 Model-Checking on Graphs of Bounded Tree Width

We have now completed the first reduction of the proof as seen in the overview in Fig. 1. In this chapter we complete the proof by constructing a MSO-Interpretation to reduce k-l-MC(GRAPH, MSO) to l-MC(TREE$_{lo}$, MSO).
This can be divided into two parts. First we generate a coding for $\mathcal{G}$ as a *labeled tree* Then we translate the MSO-sentence φ to work on that coding.

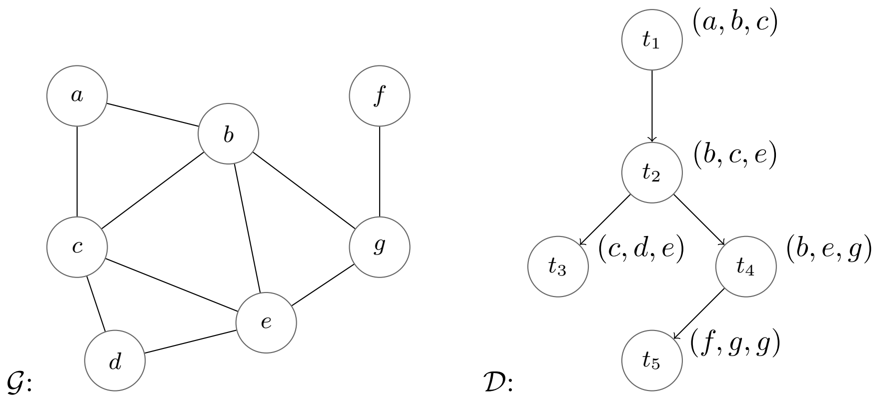

Fig. 3. Graph $\mathcal{G}$ with its *ordered tree decomposition* $\mathcal{D}$

3.1 Generating a coding for $\mathcal{G}$

To encode $\mathcal{G}$ we define $\mathcal{D}(\mathcal{T}, (\bar{b}^t)_{t\in T})$ to be an *ordered tree decomposition* of $\mathcal{G}$ with $k := tw(\mathcal{G}) = width(\mathcal{D})$. This can be computed efficiently for a fixed k, recall Bodlaender's Theorem. We also define $\lambda(t)$ to be the labeling of $t \in T$ consisting of three parts, where

$$\lambda(t) := \big(\lambda_1(t), \lambda_2(t), \lambda_3(t)\big).$$

The first part encodes all edges in the original graph between vertices from the same bag. We let

$$\lambda_1(t) := \big\{(i,j) \in [k+1]^2 \mid \text{ for all } i \leqslant j \text{ with } (b_i^t, b_j^t) \in E\big\}.$$

Duplicate vertices in the same bag are denoted in the second part by

$$\lambda_2(t) := \big\{(i,j) \in [k+1]^2 \mid b_i^t = b_j^t\big\}.$$

The last part of the encoding,

$$\lambda_3(t) := \begin{cases} \emptyset, & \text{if } t \text{ is root of } \mathcal{T} \\ \big\{(i,j) \in [k+1]^2 \mid b_i^s = b_j^t\big\}, & \text{for the parent } s \text{ of } t \end{cases}$$

marks the same vertices appearing in consecutive bags.
This in linear time computable encoding $\mathcal{T}(\mathcal{G}, \mathcal{D})$ carries all the information to fully reconstruct the initial Graph $\mathcal{G}$.

Example 3.1. Consider the Graph $\mathcal{G}$ and its ordered tree decomposition $\mathcal{D}(\mathcal{T}, (\bar{b}^t)_{t \in T})$ in Fig. 3. We will now generate the labeling for $t_2 \in T$.

All of the vertices in the second bag are connected in $\mathcal{G}$, thus $\lambda_1(t_2) = \{(1,2),(1,3),(2,3)\}$. The second part of the labeling only consists of the tuples with the same number on each side, due to t_2 not having additional duplicated vertices. Comparing t_2 to its parent node t_1 we see that the vertices b and c are contained in both. Therefore $\lambda_3(t_2) = \{(2,1),(3,2)\}$. Now combining all three parts of the labeling we end up with

$$\lambda(t_2) = \big(\{(1,2),(1,3),(2,3)\},\{(1,1),(2,2),(3,3)\},\{(2,1),(3,2)\}\big).$$

3.2 Translating the MSO-sentence φ

Now that we have defined a way to map $\mathcal{G}$, our graph of bounded tree width, onto a ordered labeled tree $\mathcal{T}(\mathcal{G},\mathcal{D})$ we proceed by translating φ. To complete the MSO-Interpretation we translate φ into a new MSO-sentence φ^* with $\mathcal{G} \models \varphi$ if and only if $\mathcal{T} \models \varphi^*$. The translation of φ can be divided into four steps. First we define a way to represent subsets of V in T. In the next step we give five conditions, which define the existence of subsets and elements in this new representation of subsets. These conditions get translated into MSO-formulas and are used to replace the quantifier in φ^*. At last we inductively translate our MSO-sentence by using those formulas.

Encoding subsets

We encode subsets of V as $(k+1)$-tuples of subsets of T, where $k+1$ is the size of the bags in $\mathcal{D}$. We denote which bag contains a element from the subset at which position. For every $W \subseteq V$ and $i \in [k+1]$ we let

$$U_i(W) := \{t \in T \mid b_i^t \in W\} \text{ and}$$

$$\bar{U}(W) := \big(U_1(W),\ldots,U_{k+1}(W)\big).$$

We also write $U_i(v) := U_i(\{v\})$ and $\bar{U}(v) := \bar{U}(\{v\})$.

Example 3.2. Recall $\mathcal{G}$ and $\mathcal{D}$ from Fig. 3 again. The encoding of $\{c,e\} \subseteq V$ is $\bar{U}(\{c,e\}) = (\{t_3\},\{t_2,t_4\},\{t_1,t_2,t_3\})$.

Defining five conditions for existence

Lemma 3.3. *Let $\bar{U} = (U_1,\ldots,U_{k+1})$ be a tuple of subsets from T. Then there exists a $W \subseteq V$ with $\bar{U} = \bar{U}(W)$ if and only if the following conditions are satisfied:*

(1) $(i,j) \in \lambda_2(t) \implies (t \in U_i \Leftrightarrow t \in U_j)$.

(2) *s is a parent of t and* $(i,j) \in \lambda_3(t) \implies (t \in U_i \Leftrightarrow s \in U_j)$.

Furthermore, $\bar{U} = \bar{U}(v)$ for an element $v \in V$ iff additionally to (1) and (2) the following statements are satisfied.

(3) $t \in U_i$ *and* $t \in U_j \implies (i,j) \in \lambda_2(t)$.

(4) *s is a parent of t,* $t \in U_i$ *and* $s \in U_j \implies (i,j) \in \lambda_3(t)$

(5) $\bigcup_{i \in [k+1]} U_i$ *is nonempty and connected.*

Proof. Let $\bar{U} = (U_1, \dots, U_{k+1})$ be a tuple of subsets from T and $W \subseteq V$ with $\bar{U} = \bar{U}(W)$. Then they satisfy (1) and (2) per definition of λ.

Know let $\bar{U}$ be a tuple of subsets of T satisfying (1) and (2) and $W \subseteq V$. Suppose $\bar{U} \neq \bar{U}(W)$. Then there exists $s, t \in T$ and $i, j \in [k+1]$ such that $b_i^s = b_j^t$, $s \in \bar{U}$ and $t \notin \bar{U}$. Due to all bags containing the same vertex being connected, there remain two possible cases, which are $s = t$ or either s is a parent of t or the other way around. The first case would imply $(i, j) \in \lambda_2(t)$, contradicting (1). The second case contradicts (2), by implying $(i, j) \in \lambda_3(t)$.

For the second part of the proof the forward direction is again trivial.
Let $\bar{U}$ be a tuple of subsets of T satisfying (1) to (5) and $W \subseteq V$. Because of (5) $|W| > 0$. Suppose $|W| > 1$. Then there exists $s, t \in T$ and $i, j \in [k+1]$ such that $b_i^s \neq b_j^t$, $s \in U_i$ and $t \in U_j$. Due to $\bigcup_{i \in [k+1]} U_i$ being connected (5), we can find i, j, s, t such that $s = t$, or either s is a parent of t or the other way around. However this would contradict (3) or (4). $\square$

Translating the existential conditions

In this section we define five new MSO-formulas, which each express one of the five conditions from Lemma 3.3. Our goal is to combine these five formulas into the formulas *set* and *element*, which for all tuples $\bar{U} = (U_1, \dots, U_{k+1})$ of subsets of T,

$$\mathcal{T}(\mathcal{G}, \mathcal{D}) \models \mathit{set}(U_1, \dots, U_{k+1}) \iff \text{There is a } W \subseteq V \text{ with } \bar{U} = \bar{U}(S),$$
$$\mathcal{T}(\mathcal{G}, \mathcal{D}) \models \mathit{element}(U_1, \dots, U_{k+1}) \iff \text{There is a } v \in V \text{ with } \bar{U} = \bar{U}(v).$$

set and *element* can later be used to encode the existential quantifier for unary relations and vertices in the translated MSO-sentence φ^*.

Translating (1) into the MSO-formula φ_1,

$$\varphi_1(U_1, \dots, U_{k+1}) := \forall v \bigwedge_{i,j \in [k+1]} \Big(\big(\bigvee_{\substack{t \in T \\ (i,j) \in \lambda_2(t)}} v = t \big) \to (U_i v \leftrightarrow U_j v) \Big).$$

Again $k+1$ is the size of the bag and is fixed. The formula itself is a straight forward translation of the condition (1). The conditions (2) to (4) are translated into φ_2 to φ_4 in a similar way.

To translate (5), we first define the formula *connected*(U), which expresses that the set $U \subseteq T$ is connected. We do that by defining that no proper nonempty subset $W \subset U$ can be closed under E in U:

$$\begin{aligned} \mathit{connected}(U) := \forall W \Big(& \overbrace{\forall w(Ww \to Uw) \wedge \exists v(Uv \wedge \neg Wv) \wedge \exists w(Ww)}^{W \text{ is a proper nonempty subset of } T} \\ & \to \underbrace{\neg \forall w \forall u \big((Ww \wedge Uu \wedge (Ewu \vee Euw)) \to Wu \big)}_{W \text{ is not closed under } E} \Big). \end{aligned}$$

By replacing every instance of a atomic formula of the form Ux in *connected*(U) with $\bigvee\limits_{i \in [k+1]} U_i x$ we obtain *connected*$(\bigcup\limits_{i \in [k+1]} U_i)$. Using this formula the rest of the translation in straight forward we obtain

$$\varphi_5(U_1, \dots, U_{k+1}) := \exists v(\bigvee_{i \in [k+1]} U_i v) \wedge \mathit{connected}(\bigcup_{i \in [k+1]} U_i).$$

The conjunction of either the first two formulas or all five results in the replacements for the existential quantifier,

$$\mathit{set}(U_1,\dots,U_{k+1}) := \bigwedge_{i\in[2]} \varphi_i(U_1,\dots,U_{k+1}) \text{ and}$$
$$\mathit{element}(U_1,\dots,U_{k+1}) := \bigwedge_{i\in[5]} \varphi_i(U_1,\dots,U_{k+1}).$$

Translating φ

In this last step we use the formulas *set* and *element* to inductively translate φ. Let $W_1,\dots,W_n \subseteq V$ and $v_1,\dots,v_n \in V$. We want φ^* to be the translated MSO-formula with

$$\mathcal{G} \models \varphi(W_1,\dots,W_n,v_1,\dots,v_n) \text{ iff}$$
$$\mathcal{T}(\mathcal{G},\mathcal{D}) \models \varphi^*(\bar{U}(W_1),\dots,\bar{U}(W_n),\bar{U}(v_1),\dots,\bar{U}(v_n))$$

We will start the induction by translating all atomic formulas. For $\varphi(v,w) = Evw$ we use λ_1 the part of the labeling were we encoded all edges of the original graph $\mathcal{G}$. We define

$$\varphi^*(\bar{V},\bar{W}) := \exists v\Big(\bigvee_{i,j\in[k+1]} (V_i v \wedge W_j v \wedge \bigvee_{\substack{t\in T\\(i,j)\in\lambda_1(t)}} v = t)\Big).$$

The formula first checks if a bag exists in which v and w are contained. Next it looks for a edge between the two vertices by looking if the corresponding labeling in contained in λ_1. The translation of $\varphi(v,w) = (v = w)$ works in a similar way. For the last remaining atomic formula $\varphi(V,w) = Vw$ we express that there exist a bag containing a element of V and w at the same spot. We let

$$\varphi^*(\bar{V},\bar{W}) := \exists v\Big(\bigvee_{i\in[k+1]} (W_i v \wedge V_i v)\Big).$$

For the induction step we have to consider quantifiers and boolean operators. The boolean operator are kept the same in the translated formula. Let $\varphi = \psi_1 \vee \psi_2$, then we define $\varphi^* = \psi_1^* \vee \psi_2^*$. The same applies to $\varphi = \neg\psi$.
To translate the quantifiers we use *set* and *element*. For a formula $\varphi = \exists v(\psi)$ and $\varphi = \forall v(\psi)$ we let

$$\varphi^* = \exists V_1 \dots \exists V_{k+1}\big(\mathit{element}(V_1 \dots V_{k+1}) \wedge \psi^*\big),$$
$$\varphi^* = \forall V_1 \dots \forall V_{k+1}\big(\mathit{element}(V_1 \dots V_{k+1}) \rightarrow \psi^*\big).$$

The same formulas can be used for quantifiers over unary relations by replacing *element* with *set*. The translated formula φ^* is computable in linear time for a given k.

4 Conclusion

Combining the results of all sections of the proof we end up with a single algorithm to solve Courcelle's Theorem.

```
COURCELLE($\mathcal{G}$, $\varphi$)
  1. Check if $\varphi$ uses the vocabulary of graphs if not reject.
  2. Compute an ordered tree decomposition $\mathcal{D} = (\mathcal{T}, (\bar{b^t})_{t \in T})$ of
     $\mathcal{G}$ with $width(\mathcal{D}) = tw(\mathcal{G})$.
  3. Compute the labeled tree $\mathcal{T}(\mathcal{G}, \mathcal{D})$.
  4. Compute the translated formula $\varphi^*$.
  5. Check if $\mathcal{T}(\mathcal{G}, \mathcal{D}) \models \varphi^*$.
```

Fig. 4. Courcelle Algorithm

Taking a look at the running time of COURCELLE($\mathcal{G}$, φ). Let $l = |\varphi|$, $k = tw(\mathcal{G})$ and $n = |V|$. Step 1 is obviously decidable in time $O(l)$. As proven in Bodlaender's Theorem [1] the second step can be computed in time $O(2^{p(k)} \cdot n)$ for a suitable polynomial p. The construction of $\mathcal{T}(\mathcal{G}, \mathcal{D})$ in step 3 can be computed in $O(f(k) \cdot n)$ for some suitable function f. Due to the size of $\mathcal{G}$ obiously not having an impact on the computation of φ^*, the fourth step is computable in time $O(g(l, k))$ for some suitable function g. Lastly the model-checking on labeled trees can be computed in linear time as proven in section 2. Thus we end up with a linear runtime for COURCELLE($\mathcal{G}$, φ) when parametrized by l and k.

In this paper Courcelle's Theorem was only proven for the class of *undirected graphs*. It can be generalized to the class of *arbitrary structures* by computing the tree decomposition of a structures *gaifmangraph*. This proof works otherwise identical and can be looked up in [5, pp. 20-21].

References

[1] Hans L. Bodlaender. "A linear-time algorithm for finding tree-decompositions of small treewidth". In: *SIAM Journal on computing* 25.6 (1996), pp. 1305–1317. DOI: `https://doi.org/10.1145/167088.167161`.

[2] Hans L. Bodlaender. "A tourist guide through treewidth". In: 92 (1992), pp. 2–3.

[3] Bruno Courcelle. "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs". In: *Information and Computation* 85.1 (1990), pp. 12–75. DOI: `https://doi.org/10.1016/0890-5401(90)90043-H`.

[4] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag Berlin Heidelberg, 2006. DOI: `https://doi.org/10.1007/3-540-29953-X`.

[5] Stephan Kreutzer. "Algorithmic Meta-Theorems". In: *CoRR* abs/0902.3616 (2009). DOI: `https://doi.org/10.1007/978-3-540-79723-4_3`.