A Note on Restricted Forms of LGG

Ondřej Kuželka[1] and Jan Ramon[2]

[1] School of Computer Science & Informatics, Cardiff University, UK
KuzelkaO@cardiff.ac.uk
[2] Department of Computer Science, KU Leuven, Belgium
jan.ramon@cs.kuleuven.be

Abstract. We study existence of a restricted least general generalization (LGG) with the property that LGGs of clauses from a pre-fixed set belong to this set. We show that there is no such LGG even in simple sets of clauses such as bounded-size clauses or treewidth-1 clauses.

1 Introduction

In this paper we study restricted forms of *least general generalization* (LGG) [6]. One such restricted form of LGG called *bounded LGG* was introduced in [4]. The main difference between ordinary LGG of some clauses $A_1, A_2, \ldots, A_k$ and their bounded LGG w.r.t. a set $\mathcal{X}$ is that the latter type of LGG does not have to be the least general of all generalizations of these clauses, it merely suffices if it is less general than any other generalization from $\mathcal{X}$. In [3], it has been shown that bounded LGG can be used for hypothesis learning without having to resort to using exponential-time algorithms for θ-subsumption if we allow the algorithm to potentially miss some hypotheses not from the set $\mathcal{X}$ (e.g. some high-treewidth hypotheses). One property of bounded LGG was, however, still asking for a further study. When computing bounded LGG w.r.t. a set $\mathcal{X}$, it may often be the case that the resulting clause will not be from the set $\mathcal{X}$. For instance, when $\mathcal{X}$ consists of clauses of treewidth bounded by k, it may be the case that a bounded LGG of some clauses w.r.t. this set will have treewidth higher than k even if the clauses to be generalized are all from the set $\mathcal{X}$ as well. The question was whether there could be another type of restricted LGG which would not have this property, at least for some reasonable sets $\mathcal{X}$. We study this question in this paper and answer it negatively even for simple sets $\mathcal{X}$.

2 Preliminaries

In this section we first briefly review some basic concepts and fix the notations used in this paper. We start with some standard definitions from graph theory and logic.

Graphs. A *directed graph* is a pair (V, E), where V is a finite set of *vertices* and $E \subseteq V \times V$ is a set of *edges*. Two vertices are said to be *adjacent* (in G) if they are connected by an edge (of the graph G). A *labeled directed graph* is a triple (V, E, λ), where (V, E) is a directed graph and $\lambda : V \cup E \to \Sigma$ is a function assigning a label from an alphabet Σ to every element of $V \cup E$. We will denote the set of vertices, the set of edges, and the labeling function of a graph G by $V(G)$, $E(G)$, and λ_G, respectively. We define $|G| = |V(G)| + |E(G)|$ and call this the size of G. A graph H is *homomorphic* to a graph G, denoted $H \preccurlyeq G$, if there exists a mapping $\varphi : V(H) \to V(G)$ such that if $(u, v) \in E(H)$ then $(\varphi(u), \varphi(v)) \in E(G)$. Two graphs H and G are homomorphically equivalent if $H \preccurlyeq G$ and $G \preccurlyeq H$. A graph G is a *core graph* if there is no strictly smaller graph homomorphically equivalent to it. It follows easily that all cores of a graph G must be homomorphically equivalent and, in fact, must also be isomorphic.

Tree decomposition, treewidth. The notion of treewidth was reintroduced in [7]. It proved to be a useful parameter of graphs in algorithmic graph theory. A *tree-decomposition* of a graph G, denoted $TD(G)$, is a pair $(T, \mathcal{X})$, where T is a rooted unordered tree and $\mathcal{X} = (X_z)_{z \in V(T)}$ is a family of subsets of $V(G)$ satisfying

(i) $\cup_{z \in V(T)} X_z = V(G)$,
(ii) for every $\{u, v\} \in E(G)$, there is a $z \in V(T)$ such that $u, v \in X_z$, and
(iii) $X_{z_1} \cap X_{z_3} \subseteq X_{z_2}$ for every $z_1, z_2, z_3 \in V(T)$ such that z_2 is on the simple path connecting z_1 with z_3 in T.

The set X_z associated with a node z of T is called the *bag* of z. The treewidth of $TD(G)$ is $\max_{z \in V(T)} |X_z| - 1$, and the *treewidth* of G, denoted $\mathrm{tw}(G)$, is the minimum treewidth over all tree-decompositions of G. By graphs of bounded treewidth we mean graphs of treewidth at most k, where k is some constant. For example, all trees have treewidth 1, cycles have treewidth 2, rectangular $n \times n$ grid-graphs have treewidth n. Any graph with treewidth 1 is a forest, possibly with loops.

Clauses. A first-order-logic clause is a universally quantified disjunction of first-order-logic literals. For convenience, we do not write the universal quantifiers explicitly. We treat clauses as disjunctions of literals and as sets of literals interchangeably. We will sometimes use a slightly abused notation $a(x, y) \subseteq a(w, x) \vee a(x, y)$ to denote that a set of literals of one clause is a subset of literals of another clause. To denote the number of literals in a clause A, we use the set notation $|A|$. The set of variables in a clause A is written as $vars(A)$ and the set of all terms as $terms(A)$. Terms can be variables or constants.

θ-subsumption and θ-reduction. Let A and B be clauses. The clause A θ-subsumes B, if and only if there is a substitution θ such that $A\theta \subseteq B$. If $A \preceq_\theta B$ and $B \preceq_\theta A$, we call A and B θ-equivalent (written $A \approx_\theta B$). A clause C is θ-reducible if there exists a clause C' such that $C' \approx_\theta C$ and $|C'| < |C|$. A clause with minimal number of literals θ-equivalent to a clause C is its θ-reduction.

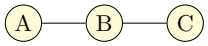
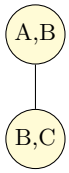
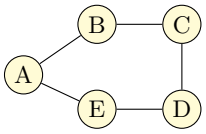
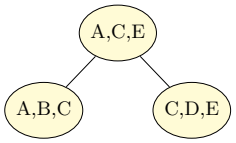
Clause	Gaifman graph	Tree decomposition
$\leftarrow \mathsf{atm}(A, \mathsf{h}) \wedge$ $\mathsf{bond}(A, B, 1) \wedge \mathsf{atm}(B, \mathsf{c}) \wedge$ $\mathsf{bond}(B, C, 2) \wedge \mathsf{atm}(C, \mathsf{o})$		
$\leftarrow \mathsf{bond}(A, B, 1) \wedge$ $\mathsf{bond}(B, C, 1) \wedge \mathsf{bond}(C, D, 1) \wedge$ $\mathsf{bond}(D, E, 1) \wedge \mathsf{bond}(E, A, 1)$		

Table 1. An illustration of Gaifman graphs and tree-decompositions of clauses.

θ-subsumption corresponds to homomorphism and θ-reduction corresponds to core of a graph.

Tree decomposition and treewidth of a clause. The Gaifman (or primal) graph of a clause A is the graph with one vertex for each variable $v \in vars(A)$ and an edge for every pair of variables $u, v \in vars(A)$, $u \neq v$ such that u and v appear in a literal $l \in A$. Gaifman graphs can be used to define treewidth of clauses. The treewidth of a clause is equal to the treewidth of its Gaifman graph. An illustration of Gaifman graphs of two exemplar clauses and their tree-decompositions is shown in Table 1. Note that tree decompositions are not unique. θ-subsumption and θ-reduction can be computed in polynomial time for clauses with bounded-treewidth θ-reductions, which makes treewidth a parameter of interest for ILP.

Least general generalization (LGG) [6]. A clause C is said to be a least general generalization (LGG) of clauses A and B (denoted by $C = \mathsf{LGG}(A, B)$ or by $C \in \mathsf{LGG}(A, B)$[3]) if and only if $C \preceq_\theta A$, $C \preceq_\theta B$ and for every clause D such that $D \preceq_\theta A$ and $D \preceq_\theta B$ it holds $D \preceq_\theta C$. An LGG of two clauses C, D can be computed in time $\mathcal{O}(|C| \cdot |D|)$. LGG can be used as an operator in the process of searching for hypotheses [1,5]. A problem of approaches based on LGG is that the size of an LGG of a set of examples can grow exponentially in the number of examples. In order to keep the LGGs reasonably small, θ-reduction is typically applied on the result of each LGG iteration [1]. An alternative to LGG capable of exploiting tractability of restricted hypothesis classes, called *bounded LGG*, was introduced in [4]. We discuss bounded LGG in the next section w.r.t. its relationship to *LGG in a set*.

[3] We will use the notation with $\in$ when we want to stress that LGG is only unique up to θ-equivalence.

3 Bounded LGG and LGG in a Set

The concept of bounded LGG was introduced in [4] in order to exploit existence of hypothesis classes with tractable θ-subsumption for learning based on LGG.

Definition 1 (Bounded LGG). *Let $\mathcal{X}$ be a set of clauses. A clause B is said to be a bounded LGG of clauses A_1, A_2, ..., A_n w.r.t. the set $\mathcal{X}$ (denoted by $B \in \mathsf{LGG}_{\mathcal{X}}(A_1, A_2, \ldots, A_n)$) if and only if $B \preceq_\theta A_i$ for all $i \in \{1, ..., n\}$ and if for every other clause $C \in \mathcal{X}$ such that $C \preceq_\theta A_i$ for all $i \in \{1, ..., n\}$, it holds $C \preceq_\theta B$.*

Note that neither the clauses to be generalized, nor the resulting bounded LGG w.r.t. $\mathcal{X}$ are required to belong to the set $\mathcal{X}$. In fact, there are cases where we can show easily that there is no bounded LGG belonging to the set $\mathcal{X}$. The following example from [4] is one such case.

Example 1. Let $\mathcal{X} = \{C_1, C_2, \ldots\}$ be a set of clauses of the following form:

$$\begin{aligned} C_1 &= e(A_1, A_2) \\ C_2 &= e(A_1, A_2) \vee e(A_2, A_3) \\ C_3 &= e(A_1, A_2) \vee e(A_2, A_3) \vee e(A_3, A_4) \\ &\ldots \end{aligned}$$

Let us also have the following two clauses:

$$\begin{aligned} A &= e(A, B) \vee e(B, A) \\ B &= e(A, B) \vee e(B, C) \vee e(C, A) \end{aligned}$$

We would like to find a clause from $\mathcal{X}$ which would be their LGG but this is impossible for the following reason. Any clause from $\mathcal{X}$ θ-subsumes both A and B but none of them is least general because for any $C_i \in \mathcal{X}$ we have $C_{i+1} \not\preceq_\theta C_i$, $C_{i+1} \preceq_\theta A$ and $C_{i+1} \preceq_\theta B$. On the other hand, bounded LGG, as actually defined, always exists which follows trivially from the fact that the conventional LGG as computed by Plotkin's algorithm [6] is also a bounded LGG. Nevertheless, it does not belong to the set $\mathcal{X}$.

Notice that the clauses A and B in the above example do not belong to the set $\mathcal{X}$. In fact, one can verify easily that for all the clauses from the set $\mathcal{X}$ from the above example, there is always an LGG belonging to the set $\mathcal{X}$. Thus, one might conjecture that, in general, if we restrict the clauses of interest, which we may want to generalize, to be from the set $\mathcal{X}$ then we will always be able to find a bounded LGG from the set $\mathcal{X}$[4]. This motivates the definition of the following, arguably quite natural, type of LGG which is studied in this paper.

[4] Actually, the main negative results presented in this paper show that this is not the case in the majority of interesting cases.

Definition 2 (LGG in a set $\mathcal{X}$). *Let $\mathcal{X}$ be a set of clauses. A clause $B \in \mathcal{X}$ is said to be an LGG of clauses A_1, A_2, ..., $A_n \in \mathcal{X}$ in the set $\mathcal{X}$ (denoted by $B \in \mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}(A_1, A_2, \ldots, A_n)$) if and only if $B \preceq_\theta A_i$ for all $i \in \{1, ..., n\}$ and if for every other clause $C \in \mathcal{X}$ such that $C \preceq_\theta A_i$ for all $i \in \{1, ..., n\}$, it holds $C \preceq_\theta B$.*

The next remark summarizes the most basic relationships between the three types of LGGs.

Remark 1. The following holds:

1. $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}(A_1, A_2, \ldots, A_n) \subseteq \mathsf{LGG}_{\mathcal{X}}(A_1, A_2, \ldots, A_n)$,
2. $\mathsf{LGG}(A_1, A_2, \ldots, A_n) \subseteq \mathsf{LGG}_{\mathcal{X}}(A_1, A_2, \ldots, A_n)$,
3. In general, $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}(A_1, A_2, \ldots, A_n) \neq \mathsf{LGG}(A_1, A_2, \ldots, A_n)$.

There are several important differences between LGG in a set $\mathcal{X}$ ($\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$) and bounded LGG w.r.t. a set $\mathcal{X}$ ($\mathsf{LGG}_{\mathcal{X}}$). Most importantly, bounded LGG w.r.t. a set $\mathcal{X}$ is not required to belong to the set $\mathcal{X}$ which is the property guaranteeing that it always exists. Since LGG in a set $\mathcal{X}$ must belong to $\mathcal{X}$, it may be the case that it does not exist. Arguably for the sets $\mathcal{X}$ in which $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$ exists, it would be preferable over $\mathsf{LGG}_{\mathcal{X}}$, especially for the sets $\mathcal{X}$ for which tractable θ-subsumption algorithms exist (e.g. bounded-size or bounded-treewidth clauses). That is one of the reasons why, in this paper, we are interested in the question of existence of $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$ in several such sets of clauses. Here, we note that if $A_1, A_2, \ldots, A_n \in \mathcal{X}$ and $\mathsf{LGG}(A_1, A_2, \ldots, A_n) \notin \mathcal{X}$ then this does not yet mean that $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}(A_1, A_2, \ldots, A_n)$ does not exist[5].

From a graph-theoretic point of view, the question about existence of an $LGG^{\mathsf{in}}_{\mathcal{X}}$ corresponds to the question whether graphs, or more precisely homomorphism equivalence classes of graphs, from a set $\mathcal{X}$ form a lattice when partially ordered by homomorphism.

The results of Horváth and Turán [2] imply that an $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$ operator exists in the class of forests of rooted directed trees (although not using this terminology). The results presented in this paper actually show that, a bit surprisingly, the results of Horváth and Turán cannot be extended much (for instance they cannot be generalized to the class of treewidth-1 graphs).

4 No LGGs in Sets of Bounded-Size Clauses

In this section, to start with a simpler problem before we tackle the question of existence of an LGG operator in the set of bounded-treewidth clauses, we consider the question of existence of LGG in the set of clauses consisting of at most k atoms. We show that there is no $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$ operator in the sets of clauses consisting of at most k atoms where k is an integer greater or equal to 4.

Theorem 1. *If $n \geq 4$ then there is no LGG operator in the set $\mathcal{X}_n$ of clauses with at most n atoms based on one binary predicate.*

[5] If this was the case then the problem of existence of $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}}$ would be almost trivial.

Proof. Let us consider the following two clauses

$$A = e(A,B) \vee e(A,C) \vee e(C,B) \vee e(B,A)$$
$$B = e(A,B) \vee e(C,A) \vee e(C,B) \vee e(B,A).$$

One can verify by considering all clauses consisting of at most 4 atoms with predicate $e/2$ that there is no LGG of A and B in this set[6] which already establishes the proof for the case $n = 4$. To finish the proof for an arbitrary $n \geq 4$, we can construct A' as

$$A' = A \cup \{aux(c_1), aux(c_2), \ldots, aux(c_{n-4})\}$$

and B' as

$$B' = B \cup \{aux(c_1), aux(c_2), \ldots, aux(c_{n-4})\}.$$

Now, we can notice that, if there was an LGG of A' and B' with at most n atoms, it would have to contain all of the $aux(c_i)$ literals. However, then it is not hard to show that there cannot be any LGG of size at most n (we would need to include the LGG of A and B but it would have to have at most 4 atoms, which is impossible as we have shown).

In order to show that the proposition holds when the arities of atoms are greater than two, we can simply add the same 'dummy' constant arguments to every atom in the clauses which would then yield the desired result automatically. □

The next example shows[7] that there are clauses $A, B \in \mathcal{X}_3$ such that $\mathsf{LGG}(A,B) \cap \mathcal{X}_3 = \emptyset$ and $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}_3}(A,B) \neq \emptyset$.

Example 2. Let us have the following two clauses

$$A = e(A,B) \vee e(B,A)$$
$$B = e(A,B) \vee e(B,C) \vee e(C,A).$$

Their *conventional* θ-reduced LGG is

$$e(A,B) \vee e(B,C) \vee e(C,D) \vee e(D,E) \vee e(E,F) \vee e(F,A),$$

and thus $\mathsf{LGG}(A,B) \cap \mathcal{X}_3 = \emptyset$. However, there exists an $\mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}_3}(A,B)$, for instance,

$$e(A,B) \vee e(B,C) \vee e(C,D) \in \mathsf{LGG}^{\mathsf{in}}_{\mathcal{X}_3}(A,B).$$

[6] We did this automatically using the code available from the first author on request. However, the number of non-θ-equivalent clauses in this case is small enough that it can also be checked manually.

[7] From this, it also follows that in order to show that, in general, there is no LGG in the set $\mathcal{X}_3$, it is not enough to show that θ-reduced Plotkin's LGG of some two clauses from $\mathcal{X}_4$ has more than 4 atoms.

The next theorem shows that the result in Theorem 1 is tight.

Theorem 2. *There is an LGG operator in the set $\mathcal{X}_3$ of clauses consisting of at most 3 binary atoms with the same predicate.*

Proof. We used the same program as for Theorem 1 to exhaustively check existence of LGGs in $\mathcal{X}_3$ for all pairs of clauses in $\mathcal{X}_3$. □

Along the same lines, we can show that if we allow more than one binary predicate, the situation becomes even worse.

Theorem 3. *If $n \geq 3$ then there is no LGG operator in the set $\mathcal{X}_n^{(2)}$ of clauses with at most n atoms with two different binary predicates.*

We could see in this section, which was mostly meant to illustrate the general problem of existence of LGGs *in* sets of clauses, that size of the clauses is not a very good measure for defining sets of clauses with an LGG operator. This is a bit unfortunate but not very surprising.

5 No LGGs in Sets of Treewidth-1 Clauses

In this section, we show that there is no LGG in the set of clauses with treewidth 1. This is quite surprising given the positive result of Horváth and Turán [2]. However, there is no disagreement between this positive result and our negative result as the negative result depends on the fact that graphs with loops have treewidth 1 too whereas loops are not allowed in the other setting corresponding to the positive result.

Theorem 4. *There is no LGG operator for the set of clauses with treewidth 1.*

Proof. Let us have two clauses

$$\begin{aligned} A =& \mathit{red}(a1) \vee \mathit{green}(a2) \vee \mathit{yellow}(a3) \vee \mathit{black}(a4) \vee e(a5,a1) \vee a(a5,a2) \vee \\ & \vee e(a5,a6) \vee e(a6,a5) \vee e(a6,a3) \vee e(a6,a4) \vee e(a5,a5) \vee e(a6,a6) \\ B =& \mathit{red}(b1) \vee \mathit{yellow}(b2) \vee \mathit{green}(b3) \vee \mathit{black}(b4) \vee e(b5,b1) \vee a(b5,b2) \vee \\ & \vee e(b5,b6) \vee e(b6,b5) \vee e(b6,b3) \vee e(b6,b4) \vee e(b5,b5) \vee e(b6,b6) \end{aligned}$$

which are depicted in Figure 1 as graphs (the predicate $e/2$ is interpreted as *edge*). The conventional LGG of the clauses A and B is a clause C which represents the graph shown in Figure 2 The clause C is not θ-reducible (i.e. the corresponding labeled graph is a core) and has treewidth greater than 1 as it contains a clique on 4 vertices. As we have already explained this does not guarantee that there is no LGG of A and B in the set of clauses of treewidth 1. We therefore need to prove that there is indeed no such clause of treewidth 1, which we will do by contradiction.

Let us assume that there is a clause D which is an LGG of A and B and which has treewidth 1. Such a clause must correspond to a labeled tree or forest,

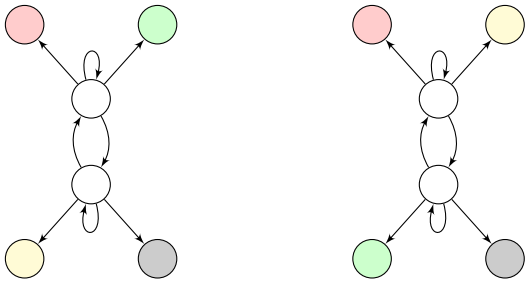


Fig. 1. Two graphs corresponding to clauses A and B from the proof of Theorem 4.

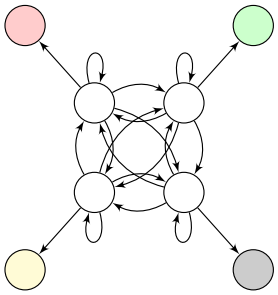


Fig. 2. The graph corresponding to clause $C \in LGG(A, B)$ from the proof of Theorem 4.

possibly with loops. It follows from the definitions of LGG and LGG in a set that D must also θ-subsume C. Let us define a family of clauses

$$\begin{aligned} E_1 =& \mathit{red}(Y_1) \vee e(X_1, Y_1) \vee e(X_1, X_2) \vee \mathit{green}(Y_2) \vee e(X_2, Y_2) \vee e(X_2, X_3) \vee \\ & \vee \mathit{black}(Y_3) \vee e(X_3, Y_3) \vee e(X_3, X_4) \vee \mathit{yellow}(Y_4) \vee e(X_4, Y_4) \end{aligned}$$

$$\begin{aligned} E_2 =& \mathit{red}(Y_1) \vee e(X_1, Y_1) \vee e(X_1, X_2) \vee \mathit{green}(Y_2) \vee e(X_2, Y_2) \vee e(X_2, X_3) \vee \\ & \vee \mathit{black}(Y_3) \vee e(X_3, Y_3) \vee e(X_3, X_4) \vee \mathit{yellow}(Y_4) \vee e(X_4, Y_4) \vee \\ & \vee e(X_4, X_5) \vee \mathit{red}(Y_5) \vee e(X_5, Y_5) \vee \cdots \vee \mathit{yellow}(X_8) \vee e(X_8, Y_8) \\ & \ldots \\ E_k =& \mathit{red}(Y_1) \vee e(X_1, Y_1) \vee e(X_1, X_2) \vee \cdots \vee \mathit{yellow}(Y_{4k}) \vee e(X_{4k}, Y_{4k}). \\ & \ldots \end{aligned}$$

Clearly, each E_i θ-subsumes C. By the assumption that D is an LGG in the set of clauses of treewidth 1, each E_i should also θ-subsume D (because each E_i has treewidth 1 and θ-subsumes A and B). Let us denote $D_i = E_i\theta$ where $E_i\theta_i \subseteq D$ and θ_i is an arbitrary suitable substitution. Since D θ-subsumes the clause C, no

vertex in the graph corresponding to the clause D can be adjacent to two vertices labeled by different colors, i.e. the clause D cannot contain simultaneously e.g. literals $e(X,Y)$, $e(X,Z)$, $yellow(Y)$ and $red(Z)$. It follows that if $e(X_j, X_{j+1}) \vee e(X_{j+1}, X_{j+2}) \subseteq E_i$ then θ_i cannot map X_j and X_{j+2} on the same term (this follows from the construction of E_i's). For similar reasons, θ_i cannot map X_j and X_{j+1} to the same term. Since D corresponds to a directed tree, possibly with loops or cycles of length 2, and therefore contains no simple cycles of length greater than 2, it follows that no two $X_j \neq X_{j'}$ can be mapped to the same term in D. However, since D is finite, there must be E_k such that $E_k \preceq_\theta A$, $E_k \preceq_\theta B$ but $E_k \not\preceq_\theta D$ which is a contradiction with D being an LGG of A and B in the set of treewidth-1 clauses. It follows that there is no finite LGG of A and B in this set of clauses. □

Note that the above theorem shows the existence of a counterexample only for treewidth-1 clauses and not for treewidth-k clauses in general. Thus, theoretically, it might be the case that there is an LGG operator in the class of clauses of treewidth at most k, where $k > 1$, and a proof would still be needed to disprove such a conjecture for general k. This seems unlikely, though.

6 Conclusions

The problems studied in this paper were motivated by the question whether bounded LGG w.r.t. a set $\mathcal{X}$, introduced in [4], could not be replaced by another type of LGG guaranteeing that the resulting generalized clauses would belong to the set $\mathcal{X}$, at least when generalizing clauses from $\mathcal{X}$. We have shown that such an alternative LGG does not exist already for natural and simple sets $\mathcal{X}$ such as the set of bounded-size clauses and the set of treewidth-1 clauses. Thus, to our best knowledge, bounded LGG remains the only candidate for an LGG capable of exploiting tractability of bounded-treewidth clauses for learning based on LGG.

Acknowledgement. This work was supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks, a Theory-based approach". The first author is supported by a grant from the Leverhulme Trust (RPG-2014-164).

References

1. T. Horváth, G. Paass, F. Reichartz, and S. Wrobel. A logic-based approach to relation extraction from texts. In *ILP*, pages 34–48, 2009.
2. T. Horváth and G. Turán. Learning logic programs with structured background knowledge. *Artif. Intell.*, 128(1-2):31–97, 2001.
3. O. Kuželka. *Fast Construction of Relational Features for Machine Learning.* PhD thesis, CTU in Prague, 2013.
4. O. Kuželka, A. Szabóová, and F. Železný. Bounded least general generalization. In *ILP 2012*, pages 116–129, 2012.
5. S. Muggleton and C. Feng. Efficient induction of logic programs. In *ALT*, pages 368–381, 1990.

6. G. Plotkin. *A note on inductive generalization.* Edinburgh University Press, 1970.
7. N. Robertson and P. D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.