SEMILATTICE POLYMORPHISMS AND CHORDAL GRAPHS

PAVOL HELL AND MARK SIGGERS

ABSTRACT. We investigate the class of reflexive graphs that admit semilattice polymorphisms, and in doing so, give an algebraic characterisation of chordal graphs. In particular, we show that a graph $G$ is chordal if and only if it has a semilattice polymorphism such that $G$ is a subgraph of the comparability graph of the semilattice.

Further, we find a new characterisation of the leafage of a chordal graph in terms of the width of the semilattice polymorphisms it admits.

Finally, we introduce obstructions to various types of semilattice polymorphisms, and in doing so, show that the class of reflexive graphs admitting semilattice polymorphisms is not a variety.

These are, to our knowledge, the first structural results on graphs with semilattice polymorphisms, beyond the conservative realm.

1. Introduction

In this paper, we take a first look at characterising the class of reflexive graphs admitting semilattice polymorphisms.

For a fixed graph (or any other relational structure) $G$ one can define several different constraint satisfaction problems (CSPs). For example, one can consider the homomorphism problem on $G$, in which one must decide if a given graph $H$ admits a homomorphism to $G$, or the retraction problem, in which one must decide if a given supergraph $H$ of $G$ retracts to $G$. One of the central open problems in the area is the Dichotomy Conjecture \cite{8}, claiming that for each fixed structure $G$, the corresponding homomorphism (or retraction) problem is either polynomial-time solvable, or NP-complete. It has been shown that the complexity of each of these problems is intimately related to the existence of certain polymorphisms, or relation preserving operations, on the structure $G$. For example, it is known that if a structure $G$ admits totally symmetric idempotent (TSI) polymorphisms of all arities, then both its homomorphism problem and its retraction problem are polynomial-time solvable \cite{8, 6}. Structures admitting TSI polymorphisms of all arities are important as they were shown in \cite{8} to be exactly the structures that have tree duality. There are two main sources of TSI on a reflexive graph: near-unanimity (NU) polymorphisms \cite{22} and semilattice (SL) polymorphisms \cite{6}. While much has been done to understand the nature of NU polymorphisms (see, for example, \cite{1}, \cite{3}, \cite{9}, \cite{10}, \cite{17}, \cite{20}), SL polymorphisms are not well understood.

Examples of structures admitting semilattice polymorphisms in the literature are sporadic and hard to find. They seem to have been introduced in \cite{16} as ACI

\footnote{Partially supported by NSERC (Canada), and by ERCCZ LL 1201 Cores (Czech Republic). Supported by the Korea NRF Basic Science Research Program (2012-06890000) funded by the Korean government (MEST).}

\footnote{This is a preprint version of the paper of the same name appearing (2014) in European Journal of Combinatorics .}
operations (standing for associative, commutative, and idempotent). From [8] it is known that a core structure $G$ admits TSI polymorphisms of all arities if and only if it is the retract of a certain universal graph $U_{TSI}(G)$, which was observed, in [2] for example, to itself admit an SL polymorphism. In [6] it was observed that structures admitting SL polymorphism also admit TSI polymorphisms of all arities. From this it follows (by possibly adding singleton unary relations to make the structure a core), that any structure admits all TSI polymorphisms if and only if it is the retract of a structure that admits an SL polymorphism. In [5], lattice polymorphisms, pairs of SL polymorphisms defined by the meet and join operations of a lattice ordering, were found to be useful in the characterisation of structures having specific types of tree dualities.

In this paper we focus on SL polymorphisms. We consider reflexive graphs, as this is an intuitive setting, and was shown in [8] to encode all the complexity of CSP: the Dichotomy Conjecture holds if and only if it holds for the retraction problem on reflexive graphs.

When studying structures admitting various polymorphisms, one usually considers whether or not the class is a variety, that is, whether or not it is closed under taking products and retracts. It is well known (see [20]), for example, that the classes of reflexive graphs admitting TSI or NU polymorphisms, are varieties. We show that the class of reflexive graphs admitting SL polymorphisms is closed under products, but is not closed under taking retracts. The fact that the class is not a variety contributes to the difficulty in characterising it.

One of the few concrete things known about the class of reflexive graphs which admit semilattice polymorphisms is from [7]. It follows from this paper that the class of reflexive graphs which admit conservative SL polymorphisms, a special type of SL polymorphism, are exactly the interval graphs. This strong combinatorial structure is due to the fact that having a conservative SL polymorphism is closed under taking induced subgraphs. Although this property does not hold when the assumption of conservativity is removed, we found that, surprisingly, there are weaker restrictions under which the graphs admitting SL polymorphisms retain some degree of combinatorial content.

Our main result is a new characterisation of chordal graphs in terms of SL polymorphisms, which sets the class of graphs admitting SL polymorphisms as an extension of the class of chordal graphs, and in fact, an extension of the class of products of chordal graphs. Given an SL polymorphism of a graph $G$ there are two very natural graphs defined on the vertices of $G$, the comparability graph of the semilattice, and its subgraph, the Hasse diagram of the semilattice. It is natural to consider the relation of a graph $G$ to these two auxiliary graphs. For an SL polymorphism of graph $G$, $G$ need not contain or be contained in either the comparability graph or the Hasse diagram. However, we show that a graph $G$ is chordal if and only if it admits an SL polymorphism $\phi$ such that $G$ is a subgraph of the comparability graph of $\phi$. Furthermore, we show that in this case the Hasse diagram of $\phi$ can be assumed to be both (1) a tree, and, if $G$ is connected, (2) a subgraph of $G$. An SL polymorphism $\phi$ of $G$ is called non-crossing if $G$ is a subgraph of the comparability graph of $\phi$, and is called a tree SL polymorphism if its Hasse diagram is a tree.

It was shown in [3] that chordal graphs have NU polymorphisms, so our result implies that graphs admitting non-crossing tree SL polymorphisms also admit NU
polymorphisms. In [3], the authors further showed that a chordal graph of leafage $\ell$ admits an NU polymorphism of arity $\ell + 1$ (refer to Section 3.3 for a definition of leafage). Under assumption (1), we obtain a new, and very natural, characterisation of the leafage of a chordal graph, suggesting that non-crossing tree SL polymorphisms, are the ‘right’ polymorphism to consider in terms of the leafage of chordal graphs.

In Section 2 we give the basic definitions. We also look at basic properties of graphs admitting SL and tree SL polymorphisms. Along with this, we give convenient necessary and/or sufficient conditions for a graph to admit SL polymorphisms of various types.

In Section 3 we prove our main results dealing with chordal graphs.

The class of graphs that admit NU polymorphisms can be recognized in polynomial time [17]. To show that a graph does not belong to this class, there are several tools, including the existence of holes [13], disconnected conservative sets, and nondisconnectability [17]. However, at least until [9], it was not easy to construct a great variety of different graphs with NU polymorphisms. The situation with SL polymorphisms is the reverse. It is easy to come up with examples of graphs with an SL polymorphism, but tends to be quite difficult to show that a graph does not admit any SL polymorphisms. In Section 4, we give necessary conditions for a graph to have an SL polymorphism or a tree SL polymorphism. These conditions allow us to show, for example, that the class of graphs admitting SL polymorphisms is not a variety, and that there are graphs admitting SL polymorphisms but not admitting tree SL polymorphisms.

2. Definitions and Basic Properties

2.1. Semilattices. In this subsection we recall standard definitions related to semilattices. From now on we assume that all graphs and sets are finite. Therefore, some definitions are stated more simply than is usual.

A semilattice order on a set $V$ is a partial order $\leq$ on $V$ such that for every pair $u, v \in V$ there is a unique greatest lower bound $\text{glb}(u, v)$. A semilattice function on a set $V$ is a binary function $\wedge : V \times V \to V : (u, v) \mapsto u \wedge v$, which is idempotent, commutative, and associative. It is well known, and easily verified, that these definitions are equivalent through the following inverse constructions. For a semilattice order $\leq$ on $V$, the function $\wedge : V \times V \to V$ defined by $u \wedge v = \text{glb}(u, v)$ is a semilattice function, and for a semilattice function $\wedge$ on $V$ the partial order $\leq$ defined by

$$u \leq v \iff u \wedge v = u$$

is a semilattice order. A semilattice on $V$ refers to either a semilattice order or a semilattice function on $V$ and we will frequently use the above correspondence implicitly to switch between the two definitions.

Given a partial order $\leq$, we use the variants $<, \geq, >$, and $\neq$ of the symbol $\leq$ with the standard variation of meaning. We write $u \perp v$, and say $u$ and $v$ are comparable if $u \leq v$ or $v \leq u$. If they are incomparable, we write $u \parallel v$. The comparability graph of $\leq$ is the graph on $V$ whose edges are all pairs of distinct comparable elements. The width of an semilattice is the maximum number of pairwise incomparable elements. An element $v$ covers or is a cover of an element $u$ if $v \geq u$ and there is no $x$ such that $v > x > u$. 
For a partial order \( \leq \) on a set \( V \), the acyclic digraph defined on \( V \) by putting an arc from \( v \) to \( u \) if \( v \) covers \( u \) is called the Hasse diagram or cover diagram of \( \leq \). We often draw the Hasse diagram of a partial order as the underlying symmetric graph, with the understanding that the arcs are going downwards. We obtain the comparability graph of \( \leq \) from the transitive closure of the Hasse diagram, by making every arc a symmetric edge.

The Hasse diagram contains all the information of the partial order \( \leq \). Indeed \( u \leq v \) if and only if \( u = v \) or there is directed path in the Hasse diagram from \( v \) to \( u \). That is to say, the partial order is the transitive closure of the cover relation. Checking that an arbitrary acyclic digraph is the Hasse diagram of a semilattice is easily done: one takes the reflexive transitive closure, and then verifies that for every pair of vertices, there is a unique greatest lower bound. This can clearly be done in polynomial time.

2.2. SL polymorphisms on graphs. All graphs from now on will be assumed to be reflexive, that is, have loops at all vertices. We call a graph chordal or interval if the graph we obtain by removing loops is respectively chordal or interval. For vertices \( u \) and \( v \) of a graph, the notation \( u \sim v \) means that \( uv \) is an edge.

A polymorphism on a graph \( G \) is a \( d \)-ary function \( f : V(G)^d \rightarrow V(G) \), on the vertex set \( V(G) \), which satisfies

\[
u_i \sim v_i \text{ for all } i \in \{1, \ldots, d\} \Rightarrow f(u_1, \ldots, u_d) \sim f(v_1, \ldots, v_d)
\]

for all \( u_i, v_i \in V(G) \). An SL polymorphism on a graph \( G \) is a semilattice on the vertices of \( G \) that (viewed as a function \( \wedge \)) is also a polymorphism. (For consistency, we will refer to any semilattice on \( V(G) \) as a semilattice on \( G \).) Thus a semilattice on \( G \) is an SL polymorphism on \( G \) if and only if it satisfies the polymorphism property:

(1) \[ a \sim a', \ b \sim b' \Rightarrow (a \wedge b) \sim (a' \wedge b') \]

Observe that if \( \phi \) is an SL polymorphism of a graph \( G \), then an edge \( uv \) of \( G \) need not consist of comparable elements under \( \phi \). We call an edge \( uv \) of \( G \) such that \( u \parallel v \) a cross-edge of \( G \) with respect to \( \phi \).

2.2.1. Connectedness. The following allows us to restrict our attention to connected graphs.

Lemma 2.1. A graph \( G \) admits an SL polymorphism if and only if each component does.

Proof. If \( G \) admits an SL polymorphism \( \wedge \), then the restriction of \( \wedge \) to any component is also a semilattice (and so an SL polymorphism on the component). To see this, it is enough to show that for vertices \( u \) and \( v \) in a component \( C \) of \( G \), \( u \wedge v \) is also in \( C \). This is standard: as \( G \) is reflexive one can find a vertex \( w \) and integer \( n \) such that there are paths \( w = x_1 \sim x_2 \sim \ldots \sim x_n = u \) and \( w = y_1 \sim y_2 \sim \ldots \sim y_n = v \) of length \( n \). Applying \( \wedge \) we see that \( w = w \wedge w = x_1 \wedge y_1 \sim x_2 \wedge y_2 \sim \cdots \sim x_n \wedge y_n = u \wedge v \) is a path from \( w \) to \( u \wedge v \). So \( u \wedge v \) is in \( C \).

On the other hand let \( C_1, C_2 \) be components of \( G \) and assume that each admits an SL polymorphism. We show that \( G \) admits an SL polymorphism; the lemma will follow by induction.

Let \( \leq_1 \) and \( \leq_2 \) be the semilattice polymorphisms on \( C_1 \) and \( C_2 \) respectively. Construct the simple sum semilattice ordering \( \leq \) of \( G \), extending \( \leq_1 \) and \( \leq_2 \) by
letting \( m_1 \geq M_2 \), where \( m_1 \) is the minimum element of \( \leq_1 \), and \( M_2 \) is any maximal element of \( \leq_2 \), and then taking the transitive closure.

To see that the resulting semilattice function is a polymorphism let \( aa' \) and \( bb' \) be edges of \( G \). Each of them is in one of the components \( C_1 \) or \( C_2 \). If they are both in the same component \( C_1 \), then \( a \& b = a \& b \sim a' \& b' = a \& b \) where \( \&_i \) is the semilattice function of \( C_i \) defined by the ordering \( \leq_i \). So we may assume that \( a \) and \( a' \) are in \( C_1 \), and that \( b \) and \( b' \) are in \( C_2 \). But then we have \( a \& b = M_2 \& b \sim M_2 \& b' = a' \& b' \), and so \( \& \) is a polymorphism. \( \square \)

2.2.2. The product of graphs admitting SL polymorphisms. It is easy to show that for semilattices \( \& \) and \( \&' \) on \( V \) and \( V' \) respectively, the product function

\[
(u, u') \& (v, v') = (u \& v, u' \& v'),
\]

is a semilattice on \( V \times V' \). It is called the product semilattice. Further, the minimum element of the product semilattice \( \&_\times \) is \((0,0')\) where \( 0 \) and \( 0' \) are the minimum elements of \( \& \) and \( \&' \) respectively.

This product is consistent with the graph product defined on graphs \( G \) and \( G' \) by \( V(G \times G') = V(G) \times V(G') \) and \((u, u') \sim (v, v')\) if \( u \sim v \) and \( u' \sim v' \), in the following sense.

**Lemma 2.2.** Let \( \& \) and \( \&' \) be SL polymorphisms of \( G \) and \( G' \) respectively. Then \( \&_\times = \& \times \&' \) is an SL polymorphism of \( G \times G' \).

**Proof.** Let \((u_a, u'_a) \sim (v_a, v'_a)\) and \((u_b, u'_b) \sim (v_b, v'_b)\).

\[
(u_a, u'_a) \&_\times (u_b, u'_b) = (u_a \& u_b, u'_a \&' u'_b) \sim (v_a \& v_b, v'_a \&' v'_b) = (v_a, v'_a) \&_\times (v_b, v'_b),
\]

where the adjacency is because of the adjacency in the coordinates, which comes from the fact that the two initial semilattices were polymorphisms. \( \square \)

2.2.3. Ordering identities. Given a function on a graph, it is easy to check that the function is a polymorphism, and if it is not, to add edges necessary to make a supergraph under which the function is a polymorphism. One of the big differences between SL polymorphisms, and other polymorphisms such as NU polymorphisms, is that SL polymorphisms have a visual representation, in their Hasse diagram, from which one can easily predict which edges one will have to add. This allows one to construct graphs, with various properties, admitting SL polymorphisms.

Though checking that the polymorphism property holds is computationally easy, it is a bit cumbersome to check by hand on small examples. The following observation helps in checking the polymorphism property by hand; it is particularly useful in predicting which edges must be added to a graph to make a given semilattice into a polymorphism. This will also be useful in proofs.

**Lemma 2.3.** The polymorphism property implies the following properties:

(i) \((u \geq v \geq w, u \sim w) \Rightarrow v \sim w\) \quad (The min property)

(ii) \(u \sim v \Rightarrow u \sim (u \& v)\) \quad (The V-property)

**Proof.** For the first property assume that \( u \geq v \geq w \) and \( u \sim w \). Then taking \( a = u, a' = w \) and \( b = b' = v \), we have \( a \sim a' \) and \( b \sim b' \) so by the polymorphism property \( v = a \& b \sim a' \& b' = w \). For the second property, take \( a = a' = b = u \) and \( b' = v \). We thus have \( a \sim a' \) and \( b \sim b' \), and so \( u = u \& u = a \& b \sim a' \& b' = u \& v \). \( \square \)
We note that the min property and the V property do not imply the polymorphism property. Indeed, they do not even imply the necessary cross-edge \((a \land b) \sim (a' \land b')\) in the most natural case where \(a \parallel a'\) and \(b \parallel b'\), and \(a \land b \parallel a' \land b'\).

The V-property is trivial in the case that \(u \perp v\), but for cross-edges it, along with the min property, adds many edges in the 'Vee' created by \(u, v\) and \(u \land v\). See Figure 1.

2.3. **Tree semilattices.** A semilattice is called a *tree semilattice* if any one of the following equivalent conditions is satisfied. (Their equivalence is easy to see.)

- The Hasse diagram is a tree
- Each element, other than the minimum element, covers a unique element
- \(u \geq v, w \Rightarrow v \perp w\).

A tree semilattice is well defined constructively by designating a root element, and then for each other element designating one element that it covers from among those already defined. The following identities about tree semilattices are useful, and easily checked.

**Lemma 2.4.** For any tree semilattice on a set \(A\), and any \(a, b, a', b' \in A\), the following are true.

- (i) If \(a \geq a'\) and \(b \geq b'\) and \(a' \parallel b'\) then \(a \land b = a' \land b'\).
- (ii) In particular if \(a \land b \parallel a' \land b'\), then \(a \land b \geq a \land a' = b \land b' \leq a' \land b'\).
- (iii) If \(a \land b \geq a' \land b'\), then either \((a' \land b') = (a' \land a) = (a' \land b)\) or \((b' \land a') = (b' \land a) = (b' \land b)\).

2.4. **Tree SL polymorphisms on graphs.** Recall that an SL polymorphism in which the semilattice is a tree semilattice is called a tree SL polymorphism. As we also discussed above Lemma 2.3, the following lemma is useful in constructing
examples of graphs admitting tree SL polymorphisms, and in proving that given
tree SL functions are polymorphisms.

**Lemma 2.5.** A tree semilattice on a graph $G$ is a tree SL polymorphism if and
only if it satisfies the min property, the $V$ property, and the polymorphism property
for all cross-edges $aa'$ and $bb'$ such that $a \land a' = b \land b'$.

Note that Lemma 2.4 implies that for such cross-edges we have $a \land b \geq a \land a' = b \land b' \leq a' \land b'$.

Proof. The necessity of the conditions is immediate. To prove their sufficiency, let
$G$ be as in the premise of the lemma, and let $aa'$ and $bb'$ be edges. We must show
that $a \land b \sim a' \land b'$.

If $a \land b \parallel a' \land b'$ then $a \parallel a'$ and $b \parallel b'$ because $\leq$ is a tree semilattice, and so $aa'$ and $bb'$ are cross-edges. Further by Lemma 2.4, $a \land a' = b \land b'$, so by the premise of
the lemma, we are done.

If $a \land b = a' \land b'$ then the conclusion is trivial. So we may assume that $a \land b > a' \land b'$. By Lemma 2.4, we may assume that $(a' \land b') = (a \land a')$ and so $a' \land b'$ is adjacent
to $a$ by the $V$-property. As $a \geq a \land b \geq a' \land b'$, the min property then gives us that
$a \land b \sim a' \land b'$. \hfill \Box

### 3. Chordal Graphs, Leafage, and Non-Crossing Tree SL Width

Recall that an SL polymorphism $\leq$ on a graph $G$ is non-crossing if $G$ is a subgraph
of the comparability graph of $\leq$. Equivalently it is non-crossing if $G$ has no cross-
edges with respect to $\leq$. It is embedded if the Hasse diagram of $\leq$ (and so the
minimal symmetric graph containing it) is a subgraph of $G$. It is not too hard to see
that if an SL polymorphism on $G$ is non-crossing and embedded, then it must be
a tree SL polymorphism. It is somewhat surprising though that a connected graph
admitting a non-crossing SL polymorphism must admit an embedded non-crossing
tree SL polymorphism. We show this in this section by showing that a connected
graph admits these polymorphisms if an only if it is chordal.

In the first subsection we define chordal graphs and recall various alternate def-
nitions. In the second subsection we state and prove the equivalences mentioned
above. We finish the section by relating the so-called leafage of a chordal graph to
the width of the non-crossing tree SL polymorphisms that it admits.

#### 3.1. Chordal graphs

Recall that a graph is chordal if all cycles of length at least
four have a chord.

It is well known ([11]) that chordal graphs can be characterized as the graphs for
which there exist a perfect elimination order (PEO). A PEO of a graph is a total
order $\geq$ of its vertices which satisfies the following property:

$$(u \geq v \geq w), (u \sim w), (u \sim v) \Rightarrow v \sim w.$$  

Another well-known characterisation ([11]) of chordal graphs is that a graph $G$
is chordal if and only if it can be represented as the intersection graph of a set of
subtrees of a tree. Such a representation of $G$, called a subtree representation, is
given as an injection $t$ of $V(G)$ into the set $S(T)$ of subtrees of some host tree $T$,
such that $u \sim v$ if and only $t(u)$ and $t(v)$ intersect.
3.2. **Chordal graphs and non-crossing SL polymorphisms.** For non-crossing tree semilattices, all the premises of Lemma 2.5, with the exception of that about the min property, become vacuous. Thus we obtain the following.

**Corollary 3.1.** A non-crossing tree semilattice on a graph is a non-crossing tree SL polymorphism if and only if it satisfies the min property.

It follows from [7] that the graphs that admit non-crossing tree SL polymorphisms in which the semilattice is a chain, are exactly the interval graphs (which are precisely the chordal graphs of leafage 2). The following theorem, can be viewed as an extension of this fact.

**Theorem 3.2.** For a graph $G$ the following are equivalent.

(i) $G$ is chordal.

(ii) $G$ admits a non-crossing SL polymorphism.

(iii) $G$ admits a non-crossing tree SL polymorphism.

Further, if $G$ is connected, the non-crossing tree SL polymorphism can be chosen to be embedded.

The implication (iii) $\Rightarrow$ (ii) of the theorem is immediate. We thus prove (ii) $\Rightarrow$ (i) in Lemma 3.3 and (i) $\Rightarrow$ (iii) for connected graphs in Lemma 3.4. This proof for connected graphs gives the ‘Further’ statement of the theorem. The implication (i) $\Rightarrow$ (iii) for disconnected graphs follows by observing that each component of a chordal graph is chordal, and from Lemma 2.1 by observing in the proof of that lemma, that the simple sum of two tree semilattice orderings is still a tree semilattice ordering.

**Lemma 3.3.** Any graph admitting a non-crossing SL polymorphism is chordal.

**Proof.** Let a graph $G$ have a non-crossing tree SL polymorphism, and assume that $C$ is a cycle in $G$. As edges of $C$ are between comparable vertices, there must be a maximal vertex of $C$, which is greater than both its neighbours in $C$. By the polymorphism property, these vertices are adjacent, so either $C$ has girth 3, or has a chord. Thus $G$ is chordal. $\square$

**Lemma 3.4.** A connected chordal graph has an embedded non-crossing tree SL polymorphism.

**Proof.** Let $\geq$ be a PEO of a connected chordal graph $G$, and complete the following steps to modify $\geq$ to a tree semilattice $\geq'$ of $G$.

(i) Let the $\geq$-minimum vertex be $\geq'$-minimum.

(ii) For any other vertex $v$, let $v$ cover in $\geq'$ the maximum neighbour below it in $\geq$ (such a neighbour exists because $G$ is connected).

(iii) Extend $\geq'$ transitively.

We show that $\geq'$ is an embedded non-crossing tree SL polymorphism of $G$. By construction it is clearly a tree ordering, and since the covers in (ii) are chosen to be edges of $G$, it is clearly embedded. To see that $G$ is non-crossing with respect to $\geq'$ assume that there is some cross-edge $uv$ with $u \geq v$. Further, assume that among all such cross-edges, $u$ was taken to be minimal with respect to $\geq'$. As $u$ does not cover $v$, it covers some neighbour $u'$ such that $u' \geq v$, and $\geq$ is a PEO, we must have that $u' \sim v$. But either $u' \geq' v$, contradicting the fact that $u \not\geq' v$, or $u' \parallel' v$, contradicting the choice of $u$. 
What is left to show is that $\geq'$ satisfies the polymorphism property. By Corollary 3.1, it is enough to verify the min property. Assume that $u \geq' v \geq' w$, and $u \sim w$. We must show that $v \sim w$. As $u \geq' v$ and the covers of $\geq'$ are adjacent vertices, there is a path $u = v_d, v_{d-1}, \ldots, v_1 = v$ in $G$ with $v_i \geq' v_{i-1}$ for each $i$. Starting at $i = d$, we have that

$$v_i \geq' v_{i-1} \geq' w, v_i \sim v_{i-1}, \text{ and } v_i \sim w.$$ 

As $\geq$ is a linear extension of $\geq'$ this holds for $\geq$ as well, and as $\geq$ is a PEO, we obtain $v_{i-1} \sim w$. By induction, we obtain that $v = v_1 \sim w$, as needed. □

This completes the proof of the lemmas and so of Theorem 3.2.

3.3. Leafage and non-crossing tree SL width. In [21], the leafage of a chordal graph $G$ was defined as the minimum number of leaves in the host tree of a subtree representation of $G$. In [3] it was shown that a chordal graph with leafage $k$ admits an NU polymorphism of arity $k + 1$.

It is natural to ask if this leafage is related to the number of leaves in the Hasse diagram of a non-crossing tree SL polymorphism. It turns out it is. Observe that the number of leaves in the Hasse diagram of a tree semilattice is the width of the semilattice if the minimum element is not a leaf, and is one less than the width if the minimum element is a leaf.

Definition 3.5. The non-crossing tree SL width of a chordal graph $G$ is the minimum width over all non-crossing tree SL polymorphisms of $G$.

The following actually reproves most of Theorem 3.2 (all but the ‘Further’ part).

Theorem 3.6. For any chordal graph, the leafage is one more than the non-crossing tree SL width.

Proof. The proof is by two direct constructions: (1) From a tree representation of a chordal graph we obtain a tree SL polymorphism whose Hasse diagram is (up to some small reductions which can only reduce the number of leaves) the host tree, and whose minimum element is a leaf. (2) From a non-crossing tree SL polymorphism we obtain a tree representation whose host tree is (up to some small reductions which can only reduce the number of leaves) the Hasse diagram.

Construction (1) gives that the leafage is at least one more than the non-crossing tree SL width. Construction (2) gives that the leafage is at most one more than the non-crossing tree SL width.

(1) Assume $G$ is a chordal graph with a subtree representation $t : V(G) \to S(T)$ in some host tree $T$. We may assume that every vertex of $T$ is in the image of $t$. Choosing some leaf $r$ of $T$, we obtain a tree semilattice $\leq$ of the vertices of $T$ by orienting the edges towards $r$:

$$u \leq v \text{ if } u \text{ is on the path between } v \text{ and } r \text{ in } T.$$ 

Let $m(v)$ denote the $\leq$-minimal vertex of $t(v)$. By subdividing edges of $T$, we may assume that $m$ is injective, and by contracting edges of $T$, we may assume that it is surjective. Identifying a vertex $v$ of $G$ with its image $m(v)$, the order $\leq$ on $T$ induces a tree semilattice on $G$ whose Hasse diagram is $T$. We call this $\leq$ as well. (Observe that edge divisions and retractions can only reduce the number of leaves of $T$.) We now show that $\leq$ is a non-crossing tree SL polymorphism of $G$, proving that the non-crossing tree SL width is at most the leafage of $G$. 
To see that $G$ has no cross-edges with respect to $\leq$, observe that if $u \sim v$ in $G$, then $t(u)$ and $t(v)$ share a vertex in $T$. So without loss of generality, $m(v)$ is in $t(u)$, and so $m(u) \leq m(v)$.

We have now just to show that $\leq$ is a polymorphism. As it is non-crossing it is enough to show that it satisfies the min property. Assume that $u \geq v \geq w$ and that $u \sim w$, we must show that $v \sim w$. From $u \geq v \geq w$ we have that $m(u) \geq m(v) \geq m(w)$ and from $u \sim w$ that $m(u) \in t(w)$. So $m(v)$, being on the path between $m(u)$ and $m(w)$ in $T$, is also in $t(w)$. Thus $v \sim w$. This completes one direction of the proof.

(2) For the other direction, let $G$ have a non-crossing tree semilattice $\leq$ with Hasse diagram $T$. We provide a subtree representation of $G$ with $T$ being the host tree.

Observe that $G$ and $T$ are graphs on the same set of vertices. For each vertex $v$ of $G$ let $t(v)$ be the subgraph of $T$ induced by $v$ and all neighbours of $v$ in $G$ that are greater than it in $\leq$. By the min property of $\leq$ each such graph is connected, so is a subtree of $T$. We must show that $u \sim v$ in $G$ if and only if the subtrees $t(u)$ and $t(v)$ intersect.

Assume that $u \sim v$ in $G$. Then because $\leq$ is non-crossing, we have $u \perp v$. We may assume that $u \leq v$. Then $v$ is a vertex in $t(u)$. As $v$ is necessarily in $t(v)$, $t(u)$ and $t(v)$ intersect.

Assume, on the other hand, that $t(u)$ and $t(v)$ intersect. As $T$ is a tree, the minimum (with respect to $\leq$) vertex of one must be contained in the other, so assume that $u$ is in $t(v)$. Then $u \sim v$ in $G$.

\[ \Box \]

**Corollary 3.7.** For any chordal graph of leafage $\ell$, the minimum element of any non-crossing tree SL polymorphism of width $\ell - 1$ must be a leaf.

**Proof.** If not, applying Construction (2) and then Construction (1) from the proof of the theorem would give a non-crossing tree SL polymorphism of width $\ell - 2$. \[ \Box \]

In [3] it was shown that chordal graphs of leafage $k$ admit $(k + 1)$-ary NU polymorphisms. Thus we obtain the following.

**Corollary 3.8.** If a graph admits a non-crossing tree SL polymorphism of width $k - 1$, then it admits a $(k + 1)$-ary NU polymorphism.

Observe that we can define non-crossing SL width and embedded non-crossing tree SL width of chordal graphs analogous to non-crossing tree SL width. It is clear that for any chordal graph

- non-crossing SL width $\leq$ non-crossing tree SL width
- $\leq$ embedded non-crossing tree SL width.

We give examples showing that each of these inequalities may be strict. This tells us that, at least for leafage, the right type of polymorphism to consider is the non-crossing tree SL polymorphism.

**Example 3.9.** A star with $\ell$ leaves has non-crossing tree SL width one, but embedded non-crossing tree SL width $\ell - 1$.

**Example 3.10.** Consider the graph shown for $n = 3$ in the first picture of Figure 2. Dark edges, thick and thin, are the graph edges. This graph is chordal. The thick edges, dark and light, are the covers of a non-crossing semilattice, which one can
quickly show by Lemma 2.5 to be a polymorphism. Thus the graph has non-crossing SL-width at most 3. The second picture is a critical tree obstruction with $2n + 2$ leaves, whose existence implies that it has no NU-polymorphism of arity $2n + 2$ or less (see, for example [9]). By Corollary 3.8 this tells us it has non-crossing tree SL width of at least $2n + 1$.

While graphs admitting non-crossing SL polymorphisms are chordal and so by [3] admit NU polymorphisms, the following example shows that as soon as we allow a single cross-edge, a graph admitting a tree SL polymorphism need not admit an NU polymorphism.

**Example 3.11.** One can quickly check using Lemma 2.5 that the tree semilattice indicated on the graph in the first picture of Figure 3 is a tree SL polymorphism. It clearly has only one cross-edge. On the other hand, the second figure represents a class of arbitrarily large critical tree obstructions (see [9]), showing that this graph admits no NU polymorphism.

### 4. Obstructions to SL and Tree SL Polymorphisms

In this section we give some basic obstructions to SL and tree SL polymorphisms, and use them to give examples of graphs admitting TSI polymorphisms of all arities, but not SL polymorphisms, and graphs admitting SL polymorphisms but not tree SL polymorphisms. The first example shows that the class of reflexive graphs admitting SL polymorphisms is not closed under retractions.

#### 4.1. Obstructions to SL polymorphisms.**

Given a graph $G$, the *clique graph* $Cl(G)$ is the graph whose vertices are inclusion maximal cliques of $G$, and in which vertices are adjacent if the corresponding cliques intersect. If $G$ admits an SL polymorphism $\land$, then the map

$$f : Cl(G) \to G : K \mapsto \bigwedge K,$$
Figure 3. Example 3.11. Left: A graph with a tree SL polymorphism with only one cross-edge (namely 12), but no NU polymorphism. Right: Critical tree obstructions showing the the graph has no NU polymorphism.

where \( \bigwedge V \) denotes \( v_1 \land v_2 \land \cdots \land v_n \) for a set \( V = \{v_1, v_2, \ldots, v_n\} \), is clearly a homomorphism. Indeed, if \( K \sim K' \) in \( \text{Cl}(G) \), then \( K \cap K' \) is non-empty, so contains some \( w_0 \). Where \( K = \{w_0, u_1, \ldots, u_a\} \) and \( K' = \{v_1, \ldots, v_b, w_0\} \), we have that

\[
\bigwedge K = w_0 \land u_1 \land \cdots \land u_a \\
= w_0 \land \cdots \land w_0 \land u_1 \land \cdots \land u_a \\
\sim v_1 \land \cdots \land v_b \land w_0 \land \cdots \land w_0 \\
= v_1 \land \cdots \land v_b \land w_0 \\
= \bigwedge K'
\]

Furthermore, replacing \( K' \) in the above argument with \( \{u_i, u_i\} \) for \( u_i \in K \), we see that \( \bigwedge K \sim u_i \) for all \( u_i \) in \( K \). So \( \bigwedge K \in K \) for all maximal cliques \( K \). What we have observed here is in fact true of any TSI polymorphism, and has likely been observed before. However, in the case of an SL polymorphism \( \land \), we also have the following: \( f(K) = \bigwedge K \leq u \) for any \( u \) in \( K \), (where \( \leq \) is the semilattice ordering associated to \( \land \)). This allows us to show the following.

Lemma 4.1. Let \( G \) be a graph and \( K_1 \sim \cdots \sim K_d \sim K_1 \) be an induced cycle \( C \) of cliques in \( \text{Cl}(G) \). If for each \( i = 1, \ldots, d \), each edge from \( K_i \) to \( K_{i+1} \) has a vertex in \( K_i \cap K_{i+1} \), then \( G \) admits no SL polymorphisms.

Proof. Let \( C = K_1 \sim \cdots \sim K_d \sim K_1 \) be such a cycle in \( \text{Cl}(G) \). Then the homomorphism \( f : \text{Cl}(G) \to G \) defined above maps at least one of \( K_1 \) and \( K_2 \) to \( K_1 \cap K_2 \). Assume, without loss of generality, that \( f(K_2) \in K_1 \cap K_2 \). Similarly \( f \) maps at least one of \( K_2 \) and \( K_3 \) to \( K_2 \cap K_3 \), but since \( C \) is induced, and \( f(K_2) \in K_1 \), we must have \( f(K_3) \in K_2 \cap K_3 \). Continuing in this way we see that \( f(K_i) \in K_{i-1} \cap K_i \) modulo \( d \), for all \( i = 1, \ldots, d \). Because \( f(K_i) = \bigwedge K_i \), the fact \( f(K_i) \in K_{i-1} \)
Figure 4. Example 4.2. A dismantlable graph having all TSI but no NU or SL polymorphisms.

implies that $f(K_{i-1}) \leq f(K_i)$. So we have $f(K_{i-1}) \leq f(K_i)$ modulo $d$ for all $i = 1, \ldots d$. But this is impossible, as $\leq$ is acyclic.\qed

**Example 4.2.** The graph in Figure 4 has no SL polymorphisms, as the cycle of five triangles around the outside form an induced cycle of maximal cliques satisfying the premises of Lemma 4.1. This graph was shown in [20] to have TSI polymorphisms of all arities, and be dismantlable, but to have no NU polymorphism.

**Corollary 4.3.** The class $\text{SL}$ is not closed under retractions.

**Proof.** We mentioned in the introduction that a graph $G$ has TSI polymorphisms of all arities if and only if it is the retract of a particular graph $U_{\text{TSI}}(G)$ which admits an SL polymorphism. The graph in Example 4.2 is thus an example of a retract, having no SL polymorphisms, of a graph having an SL polymorphism.\qed

4.2. **Obstructions to tree SL polymorphisms.** We know that graphs admitting tree SL polymorphisms need not be chordal, so may have large induced cycles. However, if a graph admits a tree SL polymorphism, then any induced cycle is ‘close’ to a non-induced cycle in the following sense.

Two $d$-circuits $C = v_1 \sim v_2 \sim \cdots \sim v_d \sim v_1$ and $D = u_1 \sim u_2 \sim \cdots \sim u_d \sim u_1$ in a graph $G$ are $h$-equivalent if $u_i \sim v_i$ and $u_i \sim v_{i+1}$ for all $i = 1, \ldots d$.

A circuit $C$ triangle reduces to a shorter circuit $C'$ if their symmetric difference, as edge-sets, is a 3-circuit. (More explicitly, let $C = v_1, \ldots, v_d$, where $v_i \sim v_{i+1}$ for all $i$ (indices mod $d$), be a circuit in a graph $G$. If there is a chord $v_{i-1} \sim v_{i+1}$ for any $i$, then $C' = v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_d$ is also a circuit, and we say $C$ triangle reduces to $C'$.) A $d$-circuit $C_d$ in $G$ is triangulated if there is a sequence of circuits $C_d, C_{d-1}, \ldots, C_1$, where $C_1$ is a single point, such that for all $i = 2, \ldots, d$, $C_i$ triangle reduces to a $C_{i-1}$. It is clear that a graph is chordal if and only if every circuit is triangulated.

**Lemma 4.4.** If a graph $G$ admits a tree SL polymorphism, then any cycle in $G$ is $h$-equivalent to a triangulated circuit.

**Proof.** Given a cycle $C = v_1 \sim v_2 \sim \cdots \sim v_d$ in a graph $G$ having a tree SL polymorphism $\land$ let $u_i = v_i \land v_{i+1}$ for all $i$. Then $D = u_1 \sim u_2 \sim \cdots \sim u_d$ is also a
Figure 5. The graph $G$ and the subgraph $G'$ with cycle $C$.

circuit, as

$$u_i = v_i \land v_{i+1} \sim v_{i+1} \land v_{i+1} = u_{i+1}$$

and by the $V$-property, $u_i \sim v_i$ and $u_i \sim v_{i+1}$, so $C$ and $D$ are $h$-equivalent.

We show that $D$ is triangulated. Since $u_i = v_i \land v_{i+1}$ and $u_{i+1} = v_{i+1} \land v_{i+2}$, both $u_i$ and $u_{i+1}$ are below $v_{i+1}$. Since $\land$ is a tree semilattice, this means that $u_i \perp u_{i+1}$.

As $D$ is a cycle of comparable pairs, there is some maximal $u_i$; so $u_i \geq u_{i-1}$ and $u_i \geq u_{i+1}$. By the min property we have that $u_{i-1} \sim u_{i+1}$, so $D$ triangle reduces to $D_{d-1} = u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_d$.

Now because $u_{i-1}$ and $u_{i+1}$ were both below $u_i$, they are also comparable. So $D_{d-1}$ is also a cycle of comparable pairs, and can again be triangle reduced. Continuing in this fashion, we triangulate $D$. \qed

**Proposition 4.5.** The product of two paths of length at least 4 does not admit a tree SL polymorphism.

**Proof.** Let $G$ be the product of two paths of length at least 4 and let $C$ be the 16-cycle in $G$ consisting of vertices of distance 2 from the vertex $v_0 = (2,2)$. See Figure 5. For any cycle $D$ that is $h$-equivalent to $C$, the vertices of $D$ must be in the subgraph $G'$ of $G$ consisting of vertices of distance at least 1 (and at most 3) from $v_0$.

It seems clear from the figure that no such $D$ can be triangulated, and so $G$ omits tree-SL. But to actually prove it we use simplicial homology. All concepts and definitions we use are basic in the field of algebraic topology, and can be found in any text on the subject (see, for example, [12].)

Construct a simplicial complex $X$ from $G'$ by letting $G'$ be the 1-skeleton of $X$, and letting every 3-cycle of $G'$ be a 2-simplex. Consider the homology group $H_1(X) = H_1(X, \mathbb{Z})$.

A cycle $\alpha$ in the graph $G'$ is a 1-cycle of $X$, so represents a class $[\alpha]$ in $H_1(X)$. Now $h$-equivalent circuits $\alpha = v_1 \sim v_2 \sim \cdots \sim v_d$ and $\alpha' = u_1 \sim u_2 \sim \cdots \sim u_d$ are
homologous: they differ by the boundary of the 2-chain

$$\sum_{i=1}^{d} ([v_i, v_{i+1}, u_i] + [u_i, u_{i+1}, v_{i+1}]),$$

If a circuit $\alpha$ triangle reduces to $\alpha'$ then they are homologous because their symmetric difference is a 3-circuit, so the boundary of a 2-simplex of $X$. Thus if $D$ were triangulated, then we would have $[C] = [D] = 0$ in $H_1(X)$.

We show this is not the case. Indeed, letting $G'$ be embedded as a subgraph of the unit distance graph of $\mathbb{R}^2$ defines by linear extension an embedding of the standard geometric realisation of $X$ into $\mathbb{R}^2$. The map taking any point $x$ in $G'$ to the unique point that is distance 1 from $v_0$ on the ray from $v_0$ through $x$, is a continuous mapping of $G'$ to a copy of the 1-dimensional sphere $S^1$. This mapping takes the 1-cycle $C$ to the generator of $\pi_1(S^1) \cong \mathbb{Z}$. So $C$ must be non-trivial in $\pi_1(X)$ and so in its abelianisation $H_1(X)$.

This is what we needed to show, so there is no triangulation of $D$, and by Lemma 4.4 $G$ admits no tree SL polymorphisms. \hfill $\Box$

By 2.2 and the fact that a path has an SL polymorphism, the above proposition gives us the following.

**Corollary 4.6.** There are graphs that admit SL polymorphisms but no tree SL polymorphisms.

The graph $G$ in Figure 5 also has an NU polymorphism, so we also obtain the following.

**Corollary 4.7.** There are graphs that admit NU polymorphisms but no tree SL polymorphisms.

Lemma 4.4 actually says that if $G$ admits a tree SL polymorphism, then the first homology group of the clique complex of $G$ is trivial. It was shown in [18] that this is actually true if $G$ has certain more general polymorphisms, so this is known. But the lemma shows something stronger for tree SL polymorphisms. If $G$ admits a tree SL polymorphism, then any 1-cycle in the clique complex is $h$-equivalent to a 1-cycle that is not only homologically trivial, but triangle reducible. Such ideas are considered further in [4].

5. **Concluding Remarks**

We have presented a first look at the structure of reflexive graphs admitting SL polymorphisms. We looked at SL polymorphisms in general, and also the more restricted tree SL polymorphisms. We found that the class of graphs admitting tree SL polymorphisms is a natural extension of the class of (products of) chordal graphs, but as soon as we went beyond chordal graphs, we found examples of such graphs that did not admit NU polymorphisms.

Tree SL polymorphisms seem more manageable than general SL polymorphisms, in particular, examples are easier to construct by hand. But they are perhaps less ‘algebraic’ than the general class. Though the class of graphs admitting SL polymorphisms is closed under products, this is not the case for the class of graphs admitting tree SL polymorphisms. In this way, we feel that the class of graphs admitting tree SL polymorphisms is more akin to the class of chordal graphs, and a characterization of graphs admitting tree SL polymorphisms might be similar.
in form to characterisations of chordal graphs. In [15], we show that graphs admitting tree SL polymorphisms with at most one cross edge can be recognised in polynomial time. One wonders if the recognition of graphs admitting tree SL polymorphisms with at most $k$ cross edges can also be done in polynomial time, or more interestingly, if recognition of graphs admitting any tree SL polymorphisms can be.

On the other hand, we tend to believe that a characterization of the class of graphs admitting SL polymorphisms might be more akin to the characterizations of the graphs that admit NU polymorphisms. As such, we wonder what the relation of the class of graphs admitting SL polymorphisms is to other varieties of graphs. It is not a variety, but it is contained in the variety of graphs admitting all TSI polymorphisms. The same was shown in [19] for the class of posets admitting SL polymorphisms. The authors of [19] wondered if all posets admitting NU polymorphisms must also admit SL polymorphisms. We wonder if the class of graphs admitting SL polymorphisms contains the variety of graphs admitting NU polymorphisms, or the variety of graphs generated by chordal graphs, or even the variety of absolute retracts, which is generated by paths ([14]). Though we can show that the class of graphs admitting SL polymorphisms is not closed under retraction, we can answer none of these questions.

Finally, we wonder whether the necessary topological condition in Lemma 4.4 is also sufficient for a graph $G$ to admit a tree SL polymorphism; and if not, what additional conditions may be needed.

We would like to thank the anonymous referees whose thoughtful suggestions greatly improved the paper.

**References**


**DEPARTMENT OF COMPUTING SCIENCES, SIMON FRASER UNIVERSITY, BURNABY, BC, CANADA**

**E-mail address:** pavol@sfu.ca

**COLLEGE OF NATURAL SCIENCES, KYUNGPook NATIONAL UNIVERSITY, DAEGU 702-701, SOUTH KOREA**

**E-mail address:** mhsiggers@knu.ac.kr