LIST-HOMOMORPHISM PROBLEMS ON GRAPHS
AND ARC CONSISTENCY

BENOÎT LAROSE AND ADRIEN LEMAÎTRE

Dedicated to Ivo G. Rosenberg, from mathematical son and grandson

Abstract. We characterise the graphs (which may contain loops) whose list-
homomorphism problem is solvable by arc consistency, or equivalently, that
admit conservative totally symmetric idempotent operations of all arities. We
prove that for every bipartite graph $G$, its list-homomorphism problem is
tractable if and only if $G$ admits a monochromatic conservative semilattice
operation; in particular, its list-homomorphism problem can easily be solved
by a combination of two-colouring and arc-consistency. We also present some
results in this direction for the retraction problem on graphs.

1. Introduction

In recent years, the use of universal algebraic methods has become prevalent in
the study of the complexity of fixed-target constraint satisfaction problems (CSPs)
(see for instance [7, 8]). Such CSPs can be most easily viewed as homomorphism
problems: a target structure $H$ is fixed, and one must decide if a given input struc-
ture $G$ admits a homomorphism to $H$. Although the general CSP is NP-complete,
the tractability of the fixed-target CSP depends on the nature of the target struc-
ture; great strides have recently been made to better understand the structures
underlying tractable CSPs, motivated in great part by the dichotomy conjecture
that states that every CSP is either NP-complete or in P [17]. Recent progress has
shown that the algebra of polymorphisms of the target structure controls the com-
plexity of the CSP, and that in all known cases, the stronger the identities satisfied
by this algebra, the lower the complexity of the decision problem turns out to be.
Precise, general conjectures have been proposed [6, 24] which have been verified in
various special cases [3, 4, 5].

In their seminal paper [17], Feder and Vardi give a series of reductions of the
general problem to various specific kinds of structures. In particular, they show
that every fixed target CSP is polynomially equivalent to a retraction problem on
a bipartite graph, and to a retraction problem on a reflexive graph. It is thus
of interest to understand the complexity of graph retraction problems. Similarly,
list-homomorphism problems have attracted a great deal of attention: although
Bulatov has settled the dichotomy question there [5], it is still of great interest to

Date: May 1, 2013.

2000 Mathematics Subject Classification. Primary 08A99; Secondary 05C75.

Key words and phrases. List-homomorphism problems, retraction problems, arc-consistency,
totally symmetric operations, symmetric Datalog, graphs.

The first author’s research is supported by grants from FQRNT and NSERC.
understand the refined complexity of these CSPs. The above mentioned conjectures are still wide open in this case, although they have been completely settled recently in the case of list-homomorphism problems for graphs [13]. In this setting, Feder and Hell [14] asked which of these CSPs are solvable by arc-consistency methods. It follows from results in [17] and [10] that this boils down to a purely algebraic question, namely on the existence of totally symmetric operations of all arities, see section 2.4 below. Since no commutative, idempotent operation can preserve a loopless edge, bipartite graphs don’t admit such operations, and their associated CSP cannot have tree duality. On the other hand one may consider a partial form of tree duality by restricting inputs to be bipartite graphs themselves. Viewed differently, this allows one to solve the associated CSP for a bipartite graph by using a simple combination of 2-colouring (to check if the input is bipartite) and then arc-consistency. We introduce the notion of monochromatic polymorphism to characterise graphs with this property (Theorem 3.2). In the list-homomorphism case we show that all tractable bipartite cases are solvable by arc-consistency on bipartite instances (Theorem 3.2 and Corollary 4.3). These ideas allow us to settle Feder and Hell’s question (Theorem 4.8): a graph \( H \) has its list-homomorphism problem solvable by arc-consistency if and only if it is a bi-arc graph that contains no loopless edge. The analogous problem for the retraction problem for graphs seems more involved, and we present some partial results in this direction. It is known that if a reflexive graph admits a near-unanimity (NU) operation then its retraction problem is actually first-order definable, and in particular is solvable by arc-consistency (see Theorem 5.1 below). In the irreflexive case, the tractable retraction problems have underlying graphs that are bipartite and hence we consider only monochromatic TSI polymorphisms (see section 2.4). We prove that if a bipartite graph admits a near-unanimity polymorphism, then it admits monochromatic TSI polymorphisms of all arities, and its retraction problem is solvable in logspace via the query language symmetric Datalog (Theorem 5.5); we also provide an example of a graph with retraction problem solvable in Nondeterministic logspace (via linear Datalog), but does not admit monochromatic TSI polymorphisms (and hence no NU polymorphisms either.)

We are greatly indebted to Victor Dalmau, Laci Egri, Pavol Hell, Ross Willard and the anonymous referees for their helpful comments. Certain results of this paper have been presented in a preliminary conference version [21].

2. Preliminaries

We refer the reader to [8] and [7] for standard definitions and notations for structures and algebra, and [1] and [27] for basic notions of complexity.

2.1. Relational structures and decision problems. A signature is a finite set of relation symbols, each with an associated arity. A \( \sigma \)-structure \( \mathcal{T} \) consists of a set \( T \) (the universe of \( \mathcal{T} \)) and a relation \( R(T) \) on \( T \) of the corresponding arity for each relation symbol \( R \) of \( \sigma \). In this paper, we always denote the universe of a structure by its roman equivalent, e.g. the universe of \( \mathcal{S} \) is \( S \), and so forth.

Let \( \mathcal{G} \) and \( \mathcal{H} \) be \( \sigma \)-structures. A homomorphism from \( \mathcal{G} \) to \( \mathcal{H} \) is a function \( f : G \to H \) such that for each \( n \)-ary relation \( R \) of \( \sigma \) and every tuple \( (v_1, \ldots, v_n) \in \mathcal{G}^n \), \( R(f(v_1), \ldots, f(v_n)) \in \mathcal{H}^n \).

\[ \sum_{i=1}^{n} w_i \]
Let $\mathbb{H}$ be a $\sigma$-structure. Our focus will be on decision problems of the following form:

- **CSP($\mathbb{H}$)**
  
  **Input:** a $\sigma$-structure $\mathbb{G}$;
  
  **Question:** is there a homomorphism from $\mathbb{G}$ to $\mathbb{H}$?

Equivalently, $\text{CSP}(\mathbb{H})$ can be viewed as the set of all $\sigma$-structures $\mathbb{G}$ such that $\mathbb{G} \to \mathbb{H}$. We shall use $\neg\text{CSP}(\mathbb{H})$ to denote the complement problem, i.e. the set of structures that do not admit a homomorphism to $\mathbb{H}$.

In this paper we consider almost exclusively list-homomorphism problems and so-called retraction problems. Informally, an input to the list-homomorphism problem for the $\sigma$-structure $\mathbb{H}$ consists of a $\sigma$-structure $\mathbb{G}$, together with lists $L_v$ for each vertex $v \in \mathbb{G}$. One must decide whether there exists a homomorphism $f$ from $\mathbb{G}$ to $\mathbb{H}$ such that $f(v) \in L_v$ for all $v \in \mathbb{G}$. The retraction problem, also known as one-or-all list-homomorphism problem, is similar except the lists $L_v$ can only be all of $\mathbb{H}$ or a one-element set. Equivalently, an input to the retraction problem can be seen as a $\sigma$-structure $\mathbb{G}$ with certain vertices coloured by a pre-assigned value from $\mathbb{H}$.\footnote{The problem gets its name from the fact that it is equivalent (in fact under positive first-order reductions) to the following decision problem: given a structure $\mathbb{G}$ containing a copy of $\mathbb{H}$, decide if $\mathbb{G}$ retracts to $\mathbb{H}$.}

We now define formally these problems.

**Definition 2.1.** Let $\mathbb{H}$ be a $\sigma$-structure. For each non-empty subset $A \subseteq \mathbb{H}$, let $S_A$ be a unary relation symbol; when $A = \{h\}$ we denote $S_A$ simply by $S_h$. Let $\tau = \sigma \cup \{S_h : h \in \mathbb{H}\}$, and let $\mathbb{H}^\tau$ denote the $\tau$-structure obtained from $\mathbb{H}$ by adding all relations $S_h(\mathbb{H}^\tau) = \{h\}$. The problem $\text{CSP}(\mathbb{H}^\tau)$ is called the retraction problem for $\mathbb{H}$. Let $\tau' = \sigma \cup \{S_A : A \subseteq \mathbb{H}\}$, and let $\mathbb{H}^\tau_{\tau'}$ denote the $\tau'$-structure obtained from $\mathbb{H}$ by adding all relations $S_A(\mathbb{H}^\tau_{\tau'}) = A$. The problem $\text{CSP}(\mathbb{H}^\tau_{\tau'})$ is called the list-homomorphism problem for $\mathbb{H}$.

### 2.2. Graphs and digraphs.

We assume the reader is familiar with the standard terminology of graphs and digraphs (vertex, edge, etc.) and will not define them here. A relational structure $\mathbb{G}$ with a single binary relation $E$ is called a digraph. If $E$ is symmetric, i.e. if $(x,y) \in E$ implies $(y,x) \in E$ then we say $\mathbb{G}$ is a graph. If $(x,x) \in E$ for all $x \in \mathbb{G}$ we say the digraph is reflexive, and if $(x,x) \notin E$ for all $x \in \mathbb{G}$ we say $\mathbb{G}$ is irreflexive. If the vertex set $G$ of a digraph $\mathbb{G}$ can be partitioned in two sets $D$ and $U$ such that $E \subseteq (D \times U) \cup (U \times D)$ we say $\mathbb{G}$ is bipartite. A sequence of not necessarily distinct vertices $v_1, \ldots, v_n$ of $\mathbb{G}$ such that $(v_i, v_{i+1}) \in E$ or $(v_{i+1}, v_i) \in E$ for every $i \in \{1, \ldots, n-1\}$ is called a walk from $v_1$ to $v_n$ in $\mathbb{G}$. If for each pair $u, v$ there exists a walk in $\mathbb{G}$ from $u$ to $v$, $\mathbb{G}$ is called connected.

### 2.3. Algebra.

The product of two $\sigma$-structures $\mathbb{S}$ and $\mathbb{T}$, denoted $\mathbb{S} \times \mathbb{T}$, is the $\sigma$-structure with universe $S \times T$ such that for every $n$-ary $R \in \sigma$, $R(S \times T) = \{(u_1,v_1), \ldots, (u_n,v_n) \mid (u_1, \ldots, u_n) \in R(S), (v_1, \ldots, v_n) \in R(T)\}$.

The product of $\mathbb{T}$ with itself $n$ times is denoted $\mathbb{T}^n$.

Let $\mathbb{H}$ be a relational structure. A $k$-ary operation $f$ on $H$ is a polymorphism of $\mathbb{H}$ if $f$ is a homomorphism from $\mathbb{H}^k$ to $\mathbb{H}$; we also say that $\mathbb{H}$ admits
We refer the reader to [10], [17] and Let for every CSP the structure (17), (10) Theorem 2.2 X all pair (X, Y) H we shall require two of them here. Given a digraph preserving H we say the problem is solvable by arc-consistency. CSP homomorphism from G H does not admit a list-preserving homomorphism to G clearly if the output structure has an empty list then the original structure from non-consistent lists until the lists stabilise. This is a poly-time algorithm, and G is one from CSP (decides correctly indeed a homomorphism; if there always is we say that the arc-consistency check hand there is no guarantee in general that if transforming any given structure (such that (h, h’) is an edge of H. The structure G is consistent if for every edge (g, g’) of G, the pair (Lg, Lg) is consistent. The arc-consistency check algorithm transforms any given structure G into a consistent structure G’ with the property that there exists a list-preserving homomorphism from G to H if and only if there is one from G’ to H; the algorithm simply recursively removes offending vertices from non-consistent lists until the lists stabilise. This is a poly-time algorithm, and clearly if the output structure has an empty list then the original structure G does not admit a list-preserving homomorphism to H: in fact, if f is a list-preserving homomorphism from G to H then f(g’) ∈ Lg for every g’ ∈ G’. On the other hand there is no guarantee in general that if G’ has non-empty lists then there is indeed a homomorphism; if there always is we say that the arc-consistency check decides correctly. One may obviously apply the algorithm to inputs of the problem CSP(H2), but also to CSP(Hk) by setting Lg = H for every uncoloured g ∈ G. If the arc-consistency check decides correctly CSP(Hk) (CSP(Hk)) on all inputs we say the problem is solvable by arc-consistency. Various equivalent conditions on a structure H are known to characterise this property (see [7], Theorem 24), we shall require two of them here. Given a digraph H, consider the digraph P(H) whose vertices are the non-empty subsets of H; there is an edge from X to Y if the pair (X, Y) is consistent. We say a homomorphism f from P(H) to H is constant-preserving if f({c}) = c for all c ∈ H, and we say it is conservative if f(X) ∈ X for all X ⊆ H.

Theorem 2.2 ([17],[10]). Let H be a digraph. Then the following conditions are equivalent:

(1) CSP(Hk) (resp. CSP(Hk)) is solvable by arc-consistency;
(2) the structure P(H) admits a constant-preserving (resp. conservative) homomorphism to H;
(3) for every k ≥ 2, H admits a k-ary (resp. conservative) TSI polymorphism.
3. Monochromatic TSI Polymorphisms and Arc-Consistency

If an undirected graph $H$ admits a TSI operation $f$ and contains at least one edge $(0, 1)$ then $f(0, \ldots, 0, 1)$ and $f(1, \ldots, 1, 0)$ are adjacent and equal, so $H$ contains a loop. In particular by Theorem 2.2 irreflexive graphs cannot have associated CSPs that are solvable by arc-consistency. However, if we consider partially-defined polymorphisms it is possible to adapt Theorem 2.2 to bipartite graphs (Theorem 3.2 below). The proof is a simple adaptation of the standard arguments.

**Definition 3.1.** Let $H$ be a bipartite graph with colour classes $D$ and $U$.

1. A set $X$ of vertices is called monochromatic if $X \subseteq D$ or $X \subseteq U$.
2. A map $f : D^n \cup U^n \to H$ is called a monochromatic polymorphism of $H$ (on $D$ and $U$) if it preserves the edge relation of $H$ whenever it is defined, i.e. if $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_k\}$ are monochromatic and if $(x_i, y_i)$ is an edge of $H$ for every $i$ then $f(x_1, \ldots, x_k)$ is adjacent to $f(y_1, \ldots, y_k)$.
3. Let $f$ be a monochromatic polymorphism. We say $f$ is a monochromatic semilattice (or NU, TSI, etc.) operation if $f$ satisfies the corresponding identities where defined.

It is easy to see that if $f$ is an idempotent monochromatic polymorphism, then it further satisfies $f(U^k) \subseteq U$ and $f(D^k) \subseteq D$, provided $H$ is connected: indeed, if $(u_1, \ldots, u_k) \in U^k$ then there exists a path of even length in the subgraph of $H^k$ induced by $U^k \cup D^k$ from $(u_1, \ldots, u_k)$ to $(u_1, \ldots, u_1)$ and hence there is a path of even length from $f(u_1, \ldots, u_k)$ to $f(u_1, \ldots, u_1) = u_1$ in $H$.

We say that the problem CSP($\mathcal{S}$) is solvable by arc-consistency on bipartite instances to mean that the arc-consistency check correctly decides on bipartite instances. The constants of the graph $\mathcal{P}(H)$ are the one-element subsets of $H$.

The equivalence of (1) and (4) below can be shown to hold for arbitrary bipartite graphs: it is not hard to see that each property holds in a graph if and only if it holds for each of its connected components.

**Theorem 3.2.** Let $H$ be a connected bipartite graph. Then the following are equivalent:

1. $\text{CSP}(H^C)$ (resp. $\text{CSP}(H^L)$) is solvable by arc-consistency on bipartite instances;
2. there is a constant-preserving (resp. conservative) partial homomorphism from $\mathcal{P}(H)$ to $H$ defined on monochromatic sets;
3. there is a constant-preserving (resp. conservative) homomorphism from the connected component of constants in $\mathcal{P}(H)$ to $H$;
4. for every $k \geq 2$, $H$ admits a monochromatic $k$-ary (resp. conservative) TSI polymorphism.

**Proof.** (2) $\Leftrightarrow$ (3): it suffices to prove that the connected component of the constants consists of the monochromatic sets. It is easy to see that if a set belongs to the component of constants it is monochromatic. For the converse, we show that every monochromatic set is in the component of constants by induction on cardinality and on the smallest non-zero distance between vertices in the set. Let $X$ be a monochromatic set and suppose that for all monochromatic sets with smaller cardinality the result is proved. If $X$ is a constant we are done, so otherwise pick two vertices $x, y \in X$ which are at minimal distance from each other in $H$; then we can find a monochromatic set $X'$ adjacent to $X$ with vertices $x', y'$ at distance
strictly shorter than the distance between $x$ and $y$: take the neighbours of $x$ and $y$ on a shortest path between them. If $x'$ and $y'$ are distinct then by induction on nonzero distance $X'$ is in the component of constants; otherwise, we have that $|X'| < |X|$ so by induction on the size we are done.

(1) $\Rightarrow$ (2): consider the instance $G$ which is the subgraph of $P(H)$ induced by the monochromatic sets, with lists $L_X = X$ for every $X \in V(G)$: obviously this is a bipartite instance. It is easy to see that if we run the arc-consistency check on this instance, the lists will remain the same (and hence non-empty). Thus there exists a homomorphism from the instance to $\mathbb{H}$ which is clearly constant-preserving (and conservative if the homomorphism preserves lists).

(2) $\Rightarrow$ (1): let $D$ and $U$ denote the colour classes of $\mathbb{H}$. Let us take a bipartite instance $\mathcal{G}$ (with appropriate lists), run the arc-consistency check on it and suppose we get non-empty lists. We fix a connected component of $\mathcal{G}$, and choose an arbitrary vertex $g$ of it, and pick an arbitrary colour $c$ in its list, without loss of generality suppose it is in the colour class $D$ of $H$: now we remove any colour from $U$ in $g$’s list, and run the arc-consistency check on the new structure: it is clear that we will get non-empty lists again (in fact the algorithm removes from the list $L_x$ precisely the vertices from $D$ or $U$, depending on the parity of the distance from $x$ to $g$). If $F$ denotes the monochromatic homomorphism from $P(\mathbb{H})$ to $\mathbb{H}$, we define a homomorphism $f$ from $\mathcal{G}$ to $\mathbb{H}$ by $f(g) = F(L_g)$, where $L_g$ is the list associated to $g$ output by the second run of the arc-consistency check. Clearly our solution preserves lists if $F$ is conservative.

(2) $\Rightarrow$ (4): let $F$ be a constant-preserving homomorphism from the set of monochromatic sets in $P(\mathbb{H})$ to $\mathbb{H}$, and define for every $k \geq 2$ an operation $f$ by $f(x_1, \ldots, x_k) = F(\{x_1, \ldots, x_k\})$. It is immediate that $f$ is idempotent and totally symmetric, and that it is conservative if $F$ is, and it is easy to verify that it is edge-preserving.

(4) $\Rightarrow$ (2): let $f$ be a monochromatic TSI operation of arity $k = 2|H|$. Let $X$ be a monochromatic subset of $H$. Let $F(X) = f(\overline{x})$ where $\overline{x}$ is any tuple whose set of entries is $X$; since $f$ is totally symmetric its value is the same on any such tuple. It is immediate that $F$ is constant-preserving and that it is conservative if $f$ is. Now let $(X, Y)$ be a consistent pair of subsets of $H$, where $X = \{x_1, \ldots, x_s\}$ and $Y = \{y_1, \ldots, y_t\}$; notice that that $s, t \leq k$. Then for every $x_j \in X$ there exists some $y_n \in Y$ adjacent to $x_j$, and for every $y_n \in Y$ there exists some $x_i \in X$ adjacent to it. Hence

$$F(X) = f(x_1, \ldots, x_s, x_{i_1}, \ldots, x_{i_t}, \ldots, x_i)$$

is adjacent to

$$F(Y) = f(y_1, \ldots, y_s, y_{i_1}, \ldots, y_{i_t}, \ldots, y_i)$$

so $F$ is edge-preserving.

Notice that when the equivalent conditions of the previous result hold, one has a very simple algorithm for the CSPs: on an arbitrary input $\mathcal{G}$, first run the 2-colouring algorithm (which can be done in logspace), and then apply the arc-consistency check. One may also interpret the results in terms of dualities or obstructions (see [7]): the CSPs admit a duality that consists of trees and odd cycles (i.e. it suffices to consider obstructions that are trees and odd cycles.)
4. List-Homomorphism Problems

In this section we consider conservative monochromatic polymorphisms on bipartite graphs, which will allow us to refine some results from [13] and [15]: there, Feder, Hell and Huang show that the list-homomorphism problem associated to a graph is tractable if the graph is bi-arc and NP-complete otherwise.

Let $K_2$ denote the complete (irreflexive) graph on $\{0,1\}$. The complement of a digraph $G$ is the digraph with set of vertices equal to $G$ and whose set of arcs is $(G \times G) \setminus E(G)$. Let $G$ be a reflexive graph, and let $\{A_v : x \in G\}$ be a family of arcs on a circle such that for every $x, y \in G$ with $x \neq y$, $(x, y) \in E(G)$ if and only if $A_x \cap A_y \neq \emptyset$. Then the family $\{A_0, \ldots, A_n\}$ is called a circular arc representation of $G$ and $G$ is called a circular arc graph. Note that a circular arc graph is necessarily reflexive. The complement of $G \times K_2$ is a circular arc graph if and only if $G$ is a bi-arc graph [15].

**Lemma 4.1.** Let $G$ be a graph. If $G$ is bipartite and the complement of $G$ is a circular arc graph, then $G$ admits a conservative monochromatic semilattice polymorphism.

**Proof.** As $G$ is bipartite, its complement has clique covering number 2. Let $N$ and $S$ be two cliques that partition the vertices of the complement of $G$. By Lemma 2 of [28], there exist points $n$ and $s$ on a circle such that the complement of $G$ can be represented by a family of circular arcs $A_u$, where every circular arc representing a vertex in $N$ contains $n$ and not $s$ and every circular arc representing a vertex in $S$ contains $s$ and not $n$, with $n$ and $s$ two points of the circle. Revolving clockwise around the circle, call $s(x)$ the starting point of the circular arc representing $x$ and $e(x)$ the ending point of the circular arc representing $x$. Because the graph has a finite number of vertices we can safely assume that there are no $A_u, A_v$ such that $s(u) = s(v)$ or $e(u) = e(v)$. For any two vertices $u, v \in N$, we define $u <_N v$ if and only if $e(u) \in A_v$. For any two vertices $u, v \in S$, we define $u <_S v$ if and only if $e(u) \in A_v$. It is easy to verify that $<_N$ and $<_S$ are total orders, as for every $u \in N$, $A_u$ contains $n$ and for every $u \in S$, $A_u$ contains $s$. Let $f : N^2 \cup S^2 \to G$ be defined by:

$$f(x, y) = \begin{cases} \min_{<_N} (x, y) & \text{if } x, y \in N \\ \min_{<_S} (x, y) & \text{if } x, y \in S \end{cases}$$

Clearly $f$ is a monochromatic conservative, semilattice operation, so we only need to check it is edge-preserving where defined. If not, then there exist $a, b \in N$ and $c, d \in S$ such that $(a, c), (b, d) \in E(G)$, $(a, d) \notin E(G)$ and $a <_N b, d <_S c$. As $a <_N b$, then $e(a) \in A_b$. As $d <_S c$, then $e(d) \in A_c$. As $(a, c) \notin E(G)$, then $A_a \cap A_c = \emptyset$. As $(b, d) \in E(G)$, then $A_b \cap A_d = \emptyset$. As $(a, d) \notin E(G)$, then $A_a \cap A_d = \emptyset$. So, $e(a) \in A_d$ or $e(d) \in A_a$. It is immediate that in both cases we get a contradiction and this completes the proof. \hfill $\square$

**Lemma 4.2.** Let $H$ be a graph. Then the following are equivalent:

1. $H$ admits a conservative weak NU polymorphism;
2. $H$ is a bi-arc graph;
3. $H \times K_2$ admits a conservative weak NU polymorphism;
4. $H \times K_2$ admits a conservative monochromatic semilattice polymorphism;
5. $H \times K_2$ admits a conservative weak NU polymorphism of arity 3.
Let \( H \) be a semilattice to a 3-ary one in the obvious way, by defining a morphism on \( H \) before those of every element of \( D \) disjoint copies of \( H \) is edge-preserving. Hence \( H \) is conservative, monochromatic operation, and it is easy to verify that it is edge-preserving. Hence \( g \) acts as the minimum function of a total ordering, when \( H \) is conservative and \( m \) is binary. As \( H \) is a conservative, monochromatic semilattice polymorphism, then by Lemma 4.2 (4) \( H \times K_2 \) is isomorphic to two disjoint copies of \( H \), namely \( (D \times \{0\}) \cup (U \times \{1\}) \) and \( (D \times \{1\}) \cup (U \times \{0\}) \). Let \( D' = D \times \{0, 1\} \) and \( U' = U \times \{0, 1\} \); these are colour classes for \( H \times K_2 \). Define a monochromatic operation \( g \) on \( H \times K_2 \) with respect to \( D' \) and \( U' \) as follows:

\[
g((x, i), (y, j)) = \begin{cases} 
(m(x, y), i) & \text{if } i = j, \\
(x, 0) & \text{if } 0 = i \neq j, x, y \in D, \\
(y, 0) & \text{if } 0 = j \neq i, x, y \in D, \\
(x, 1) & \text{if } 1 = i \neq j, x, y \in U, \\
(y, 1) & \text{if } 1 = j \neq i, x, y \in U.
\end{cases}
\]

Intuitively, the operation \( g \) is built as follows: since it is a conservative semilattice on \( D \) and on \( U \), the operation \( m \) acts as the minimum function of a total ordering, when restricted to the vertices of \( D \) or to the vertices of \( U \). Then \( g \) orders \( D' \) by placing every element of \( D \times \{0\} \) before those from \( D \times \{1\} \), and in \( U' \) places those in \( U \times \{1\} \) before those of \( U \times \{0\} \). It is then clear that \( g \) is a conservative, monochromatic semilattice operation, and it is easy to verify that it is edge-preserving. Hence \( H \times K_2 \) admits a conservative monochromatic semilattice polymorphism, so by Lemma 4.2 (4) \( H \) admits a conservative weak NU polymorphism.

The following definitions are a special case of those found in [19].
Definition 4.4. Let \( G \) be a graph.

1. Let \( P = x_0, \ldots, x_n \) and \( Q = y_0, \ldots, y_n \) be two walks in \( G \). We say that \( P \) avoids \( Q \) if for all \( i \in \{0, \ldots, n-1\} \), \( (x_i, y_{i+1}) \notin E(G) \).
2. A pair of distinct vertices \( u, v \) in \( G \) is invertible if
   - there exist walks \( P \) from \( u \) to \( v \) and \( Q \) from \( v \) to \( u \) such that \( P \) avoids \( Q \), and
   - there exist walks \( P' \) from \( v \) to \( u \) and \( Q' \) from \( u \) to \( v \) such that \( P' \) avoids \( Q' \).

Observe that if \( u_0, \ldots, u_n \) avoids \( v_0, \ldots, v_n \) and \( u_n, \ldots, u_m \) avoids \( v_n, \ldots, v_m \), then \( u_0, \ldots, u_n, \ldots, u_m \) avoids \( v_0, \ldots, v_n, \ldots, v_m \).

Lemma 4.5. Let \( G \) be a graph such that the neighbourhood of every loopless vertex is a clique of loops. If \( G \) doesn't admit a conservative binary weak NU polymorphism, then there exist walks \( P = x_0, \ldots, x_n \) from \( u \) to \( v \), \( Q = y_0, \ldots, y_n \) from \( v \) to \( u \), \( P' = x_1, \ldots, x_m \) from \( v \) to \( u \) and \( Q' = y_1, \ldots, y_m \) from \( u \) to \( v \) such that:
   - \( P \) avoids \( Q \);
   - \( P' \) avoids \( Q' \);
   - \( n \) and \( m \) are even.

Proof. If \( G \) doesn't admit a conservative binary weak NU polymorphism then by Lemma 5.1 of [19] there exists an invertible pair \((u, v)\) in \( G \), that is, there exist avoiding walks \( P, Q, P' \) and \( Q' \) satisfying the conclusion of the lemma, except it remains to prove that we can find such walks of even length. The trick is to prove that we can always increase the length by inserting avoiding walks of length 3. We show how to do this for \( P \) and \( Q \), the proof for \( P' \) and \( Q' \) is identical. Notice that since \( u \) and \( v \) are distinct by definition, we have that \( 0 < n < m \).

Suppose first that there exists \( i \in \{0, \ldots, n\} \) such that \( x_i \) is a loop but \( y_i \) isn't. Then \((x_i, y_i) \notin E(G)\) since \((y_i, y_{i+1}) \in E(G)\), \((x_i, y_{i+1}) \notin E(G)\) and the neighbourhood of every loopless vertex in \( G \) is a clique of loops. Thus \( x_i, x_i, x_i \) avoids \( y_i, y_{i+1}, y_{i+1}, y_i \). If \( n \) is odd, we insert these walks in \( P \) and \( Q \) respectively to obtain avoiding walks of even length.

Similarly, if there exists \( i \in \{0, \ldots, n\} \) such that \( y_i \) is a loop but \( x_i \) isn't then \((x_i, y_i) \notin E(G)\), and thus \( x_i, x_{i-1}, x_{i-1}, x_i \) avoids \( y_i, y_i, y_i, y_i \) (if \( i = 0 \), we use \( x_{m-1} \) instead of \( x_{i-1} \)).

So we may now suppose that there exist no \( i \in \{0, \ldots, n\} \) such that exactly one of \( x_i \) and \( y_i \) is a loop. Suppose first there exist \( i \in \{0, \ldots, n\} \) such that \( x_i, y_i, x_{i+1}, y_{i+1} \) are loops; then \( x_i, x_i, x_{i-1}, x_i \) avoids \( y_i, y_{i+1}, y_{i+1}, y_i \) (again, if \( i = 0 \), we use \( x_{m-1} \) instead of \( x_{i-1} \)). Otherwise, since \( 0 < n < m \), there exists some \( i \) such that \( x_i \) and \( y_i \) have no loops. Since neighbours of non-loops are loops, and since we cannot have consecutive loops on the walk, loops and non-loops must alternate and thus \( n \) and \( m \) must be even.

We say that an edge \((u, v)\) of a graph \( G \) is loopless if neither \( u \) nor \( v \) is a loop.

Lemma 4.6. Let \( G \) be a graph with no loopless edge. Then the following are equivalent:

1. \( G \) admits a conservative binary weak NU polymorphism;
2. \( G \times K_2 \) admits a conservative binary monochromatic weak NU polymorphism.
Proof. Let $D$ and $U$ be the colour classes of $G \times K_2$, $D = G \times \{0\}$ and $U = G \times \{1\}$. Suppose that $G$ admits a binary, conservative, commutative polymorphism $f$. Let $g : D^2 \cup U^2 \rightarrow H \times K_2$ be the function defined by:

$$g((a, x), (b, x)) = (f(a, b), x).$$

It is immediate that $g$ is a binary, conservative, commutative, monochromatic polymorphism of $G \times K_2$.

Conversely, suppose that $G$ doesn’t admit a binary, conservative, commutative polymorphism. Suppose first that some loopless vertex has a neighbourhood which is not a clique of loops. Then there exist vertices $u, v, w \in G$ such that $u$ and $w$ are loops but $v$ isn’t, $u$ and $w$ are neighbours of $v$ but not adjacent to each other. Then consider the following walks in $G \times K_2$:

$$\begin{align*}
(u, 0), & \quad (v, 1), \quad (w, 0), \quad (v, 1), \quad (u, 0), \quad (v, 1), \quad (u, 0) \\
(v, 0), & \quad (w, 1), \quad (v, 0), \quad (u, 1), \quad (u, 0), \quad (v, 1), \quad (w, 1), \quad (v, 0)
\end{align*}$$

The first (top) walk avoids the second (bottom) walk. Let $f$ be a conservative, binary monochromatic polymorphism of $G \times K_2$. If $f((u, 0), (v, 0)) = (u, 0)$, then applying $f$ on the pairs in the walks from left to right we see that $f((v, 1), (w, 1)) = (v, 1)$; moving along this way we get after 3 more steps that $f((v, 0), (u, 0)) = (v, 0)$ and hence $f$ cannot be commutative. Similarly, if $f((v, 0), (u, 0)) = (v, 0)$, starting on the middle pair of the walks and moving right, we see that we end up with $f((u, 0), (v, 0)) = (u, 0)$. Hence $G \times K_2$ admits no binary monochromatic commutative conservative polymorphism.

We may now assume that the hypotheses of Lemma 4.5 are satisfied. Then there exist $u, v \in G$ and walks $P = x_0, \ldots, x_n$ from $u$ to $v$, $Q = y_0, \ldots, y_m$ from $v$ to $u$, $P' = x_n, \ldots, x_m$ from $v$ to $u$ and $Q'$ from $u$ to $v$ such that $P$ avoids $Q$, $P'$ avoids $Q'$ and $n$ and $m$ are even. Thus there exist walks $R = (x_0, c_0), \ldots, (x_n, c_n)$ from $(u, 0)$ to $(v, 0)$, $S = (y_0, c_0), \ldots, (y_m, c_m)$ from $(v, 0)$ to $(u, 0)$, $R' = (x_n, c_n), \ldots, (x_m, c_m)$ from $(v, 0)$ to $(u, 0)$ and $S' = (y_m, 0), \ldots, (y_0, 0)$ from $(u, 0)$ to $(v, 0)$ such that $c_i = 0$ if $i$ is even, $c_i = 1$ if $i$ is odd, $R$ avoids $S$ and $R'$ avoids $S'$. An argument similar to the previous case shows that $H \times K_2$ doesn’t admit any binary, conservative, commutative monochromatic polymorphism. \hfill $\square$

Lemma 4.7. Let $H$ be a graph. Then the following are equivalent:

1. $H$ admits a conservative binary weak NU polymorphism;
2. $H$ admits conservative TSI polymorphisms of every arity $k \geq 2$.

Proof. Let $\phi$ be a binary, conservative, commutative polymorphism of $H$. Since $\phi$ is a weak NU operation, by Lemma 4.2 the graph $H \times K_2$ admits a conservative monochromatic semilattice polymorphism $m$. We may extend this binary semilattice to a $k$-ary one in the obvious way, by defining

$$m(u_1, \ldots, u_k) = m(\cdots(m(u_1, u_2), \ldots), u_k).$$

We define a $k$-ary operation $f$ on $H$ as follows: for all $x_i \in H$, define

$$f(x_1, \ldots, x_k) = \phi(\pi_H(m((x_1, 0), \ldots, (x_k, 0)))), \pi_H(m((x_1, 1), \ldots, (x_k, 1))))$$

where $\pi_H : H \times K_2 \rightarrow H$ is simply the projection on the $H$ factor.

It is easy to verify that $f$ is conservative and that it is a TSI operation. It remains to prove that it is a polymorphism. Suppose that $(x_i, y_i)$ is an edge of $H$.
for all \( i = 1, \ldots, k \). Then 
\[
u = m((x_1, 0), \ldots, (x_k, 0))\]
and 
\[
u' = m((x_1, 1), \ldots, (x_k, 1))\]
and hence 
\[
(\pi_H(u), \pi_H(u'))\]
is adjacent to 
\[
(\pi_H(v), \pi_H(v'))\]
then 
\[
f(x_1, \ldots, x_k) = \phi(\pi_H(u), \pi_H(u'))\]
is adjacent to 
\[
\phi(\pi_H(v), \pi_H(v')) = \phi(\pi_H(v'), \pi_H(v)) = f(y_1, \ldots, y_k)\]
and we are done. \(\square\)

**Theorem 4.8.** Let \( H \) be a graph. Then the following are equivalent:

(1) \( \text{CSP}(H^C) \) is solvable by arc-consistency;
(2) \( H \) is a bi-arc graph with no loopless edge;
(3) \( H \) admits a conservative binary weak NU polymorphism.

**Proof.** (1) \( \Rightarrow \) (2): by Theorem 2.2 and Lemma 4.2 \( H \) is a bi-arc graph with a conservative binary, commutative idempotent polymorphism. As remarked at the beginning of section 3, this implies immediately that every edge of \( H \) is incident to at least one loop.

(2) \( \Rightarrow \) (3): If \( H \) is a bi-arc graph, then by Lemma 4.2 \( H \times K_2 \) admits a conservative monochromatic semilattice polymorphism. We can now conclude using Lemma 4.6.

(3) \( \Rightarrow \) (1): immediate by Lemma 4.7 and Theorem 2.2. \(\square\)

5. Retraction Problems

We turn our attention to problems of the form \( \text{CSP}(H^C) \) where \( H \) is a graph. It is known that every CSP is poly-time equivalent to one in this form, even when restricted to the case where \( H \) is reflexive or \( H \) is bipartite [17]. The general question we consider here is for which \( H \) can \( \text{CSP}(H^C) \) be solved by arc-consistency, and if \( H \) is bipartite, when can it be solved by arc-consistency on bipartite instances?

In the reflexive case, it is known that the existence of an NU polymorphism on \( H \) implies that \( \text{CSP}(H^C) \) has finite duality [23], which is known to imply solvability by arc-consistency [22]; in fact Mároti and Zádori have recently proved that the presence of an NU polymorphism (or equivalently of Gumm terms) on a reflexive digraph is sufficient to guarantee solvability by arc-consistency [26]. Reflexive graphs whose retraction problem is solvable by arc-consistency have been studied in [25] but have yet to be characterised. If on the other hand the graph \( H \) is irreflexive, then tractability of its retraction problem ensures the graph is bipartite and hence it cannot be solved by arc-consistency as noted in section 3. However one may then investigate if the problem can be solved by arc-consistency on bipartite instances.

We use recent results from [16] to prove that bipartite graphs admitting an NU operation indeed have this property, and that their retraction problem is solvable in logspace via the query language symmetric Datalog (Theorem 5.5). We do not know if the converse of this statement holds, i.e. is every bipartite graph with
retraction problem solvable in symmetric Datalog an NU graph? We provide an example of a bipartite graph whose retraction problem is solvable in linear Datalog but is not solvable by arc-consistency on bipartite instances (Proposition 5.15). We believe the auxiliary results leading to this are of independent interest.

5.1. Bipartite graphs with NU polymorphisms. The logic programming language Datalog has been at the centre of recent investigations of the tractability of fixed-template CSPs (see [20], [7] for details). We shall only give the briefest of descriptions of it here. Fix a signature \( \tau \). A Datalog program is a finite set of rules of the form \( t_0 : - t_1, \ldots, t_n \) where each \( t_i \) is an atomic formula \( R(x_1, \ldots, x_m) \). Then \( t_0 \) is called the head of the rule, and the sequence \( t_1, \ldots, t_n \) the body of the rule. The predicates occurring in the heads of the rules are not from \( \tau \) and are called IDBs (from “intensional database predicates”), while all other predicates come from \( \tau \) and are called EDBs (from “extensional database predicates”). One of the IDBs is designated as the goal predicate of the program: since the IDBs may occur in the bodies of rules, each Datalog program is a recursive specification of the IDBs, with semantics obtained via least fixed-points of monotone operators, and hence we may view the program as defining the relation corresponding to the goal predicate. If the goal predicate is 0-ary, we’ll use the standard convention of identifying its initial state as \text{false}, meaning it is empty, and we say that a Datalog program accepts a \( \tau \)-structure \( G \) if its goal predicate evaluates to \text{true} on \( G \) (i.e. if it eventually becomes non-empty.)

A rule of a Datalog program is linear if its body contains at most one IDB and is non-recursive if its body contains only EDBs. A linear but recursive rule is of the form \( I_1(\bar{x}) \leftarrow I_2(\bar{y}), E_1(\bar{z}_1), \ldots, E_k(\bar{z}_k) \) where \( I_1, I_2 \) are IDBs and the \( E_i \) are EDBs (note that the variables occurring in \( \bar{x}, \bar{y}, \bar{z}_i \) are not necessarily distinct). For each such linear recursive rule the symmetric of that rule is defined as \( I_2(\bar{y}) \leftarrow I_1(\bar{x}), E_1(\bar{z}_1), \ldots, E_k(\bar{z}_k) \). A Datalog program \( \mathcal{D} \) is non-recursive if all its rules are non-recursive, linear if all its rules are linear and symmetric if it is linear and if the symmetric of each recursive rule of \( \mathcal{D} \) is also a rule of \( \mathcal{D} \).

Let \( \mathbb{H} \) be a structure. We say that \( \neg \text{CSP}(\mathbb{H}) \) is solvable in (linear, symmetric) Datalog if there exists a (linear, symmetric) Datalog program that accepts precisely those structures that do not admit a homomorphism to \( \mathbb{H} \). It is known that if \( \neg \text{CSP}(\mathbb{H}) \) is solvable in (linear, symmetric) Datalog then \( \text{CSP}(\mathbb{H}) \) is solvable in (non-deterministic logspace, deterministic logspace) polynomial time. Precise conjectures have been formulated to characterise algebraically those CSPs solvable in linear and symmetric Datalog [24]. The problem \( \text{CSP}(\mathbb{H}) \) has finite duality, or is first-order definable, if \( \neg \text{CSP}(\mathbb{H}) \) it solvable in non-recursive symmetric Datalog (see [22].)

We’ll say a digraph \( \mathbb{H} \) is strongly bipartite if its vertex set can be partitioned into two sets \( D \) and \( U \) such that every arc \( (x, y) \) of \( \mathbb{H} \) satisfies \( x \in D \) and \( y \in U \). If \( \mathbb{H} \) is a connected bipartite graph with colour classes \( D \) and \( U \), let \( \overline{\mathbb{H}} \) denote the strongly bipartite digraph obtained from \( \mathbb{H} \) by orienting every edge from \( D \) to \( U \), i.e. \( E(\overline{\mathbb{H}}) = E(\mathbb{H}) \cap (D \times U) \). Let \( G \) be any digraph; then we let \( G_u \) denote the underlying undirected graph, i.e. \( E(G_u) \) consists of all pairs \( (x, y) \) such that \( (x, y) \) or \( (y, x) \) is in \( E(\overline{\mathbb{G}}) \).

The next theorem gathers some prior results we shall require.

---

3See footnote in section 5.2.
Theorem 5.1 ([23],[16]). Let $\mathbb{H}$ be a connected digraph.

1. If $\mathbb{H}$ is a bipartite graph, it admits an NU polymorphism if and only if $\overrightarrow{\mathbb{H}}$ does;
2. If $\mathbb{H}$ is a strongly bipartite digraph, it admits an NU polymorphism if and only if $CSP(\mathbb{H}^C)$ has finite duality; if these conditions hold then $CSP(\mathbb{H}^C)$ is solvable by arc-consistency;
3. If $\mathbb{H}$ is a reflexive graph, it admits an NU polymorphism if and only if $CSP(\mathbb{H}^C)$ has finite duality; if these conditions hold then $CSP(\mathbb{H}^C)$ is solvable by arc-consistency.

Lemma 5.2. Let $\mathbb{H}$ be a connected bipartite graph. Then the following are equivalent:

1. $CSP(\overrightarrow{\mathbb{H}}^C)$ is solvable by arc-consistency;
2. $CSP(\mathbb{H}^C)$ is solvable by arc-consistency on bipartite instances.

Proof. (1) $\Rightarrow$ (2): if $\overrightarrow{\mathbb{H}}$ admits TSI polymorphisms of all arities then their restriction to the colour classes of $\mathbb{H}$ are clearly monochromatic TSI polymorphisms for $\mathbb{H}$, and hence we may invoke Theorem 3.2. (2) $\Rightarrow$ (1): By Theorem 3.2 there exists a constant-preserving partial homomorphism $F$ from $P(\mathbb{H})$ to $\mathbb{H}$ defined on monochromatic sets. Notice that in $P(\overrightarrow{\mathbb{H}})$, if $X \rightarrow Y$, then certainly $X$ and $Y$ are monochromatic, and hence $P(\overrightarrow{\mathbb{H}})$ consists of monochromatic sets and isolated vertices. Thus $F$ effectively defines a constant-preserving homomorphism from $P(\overrightarrow{\mathbb{H}})$ to $\overrightarrow{\mathbb{H}}$ (the value is irrelevant on isolated vertices). By Theorem 2.2 $CSP(\overrightarrow{\mathbb{H}}^C)$ is solvable by arc-consistency.

We shall require the following technical lemma (the short proof is due to Laci Egri):

Lemma 5.3. Let $\mathbb{H}$ be a relational structure whose basic relations are one connected binary relation $E$ together with $r \geq 0$ unary relations. If $\neg CSP(\mathbb{H})$ can be solved in symmetric Datalog when restricted to connected instances, then the same program will solve $\neg CSP(\overrightarrow{\mathbb{H}})$ on arbitrary inputs.

Proof. We’ll say a structure $\mathbb{G}$ similar to $\mathbb{H}$ is connected if its underlying digraph is connected. Let $\mathcal{P}$ be a symmetric Datalog program that accepts a connected structure $\mathbb{G}$ if and only if it does not admit a homomorphism to $\mathbb{H}$. We show that in fact $\mathcal{P}$ solves $\neg CSP(\mathbb{H})$. Let $\mathbb{G}$ be an arbitrary structure similar to $\mathbb{H}$ that does not admit a homomorphism to $\mathbb{H}$. Since the signature contains only unary relations other than the underlying digraph, it follows that $\mathbb{G}$ has some connected component $\mathbb{G}_0$ that doesn’t map to $\mathbb{H}$, and hence is accepted by $\mathcal{P}$. It is immediate by definition of Datalog program that this implies $\mathcal{P}$ accepts $\mathbb{G}$ also. Conversely, suppose that $\mathbb{G}$ admits a homomorphism $f$ to $\mathbb{H}$. We show by induction on the number of connected components of $\mathbb{G}$ that $\mathcal{P}$ does not accept it. If $\mathbb{G}$ is connected we are done. Otherwise, let $\mathbb{G}_0$ be a component of $\mathbb{G}$, and let $u \in G_0$ and $v \in G \setminus G_0$ be arbitrary elements. Since $\mathbb{H}$ is connected, there is some path $L$ from $f(u)$ to $f(v)$ in $\mathbb{H}$. Glue to $\mathbb{G}$ a copy of $L$ to connect the vertices $u$ and $v$, and let $\mathbb{G}'$ denote the resulting structure, which has one less connected component than $\mathbb{G}$. Clearly the map $f$ extends to a homomorphism from $\mathbb{G}'$ to $\mathbb{H}$, so by induction hypothesis $\mathcal{P}$ does not accept $\mathbb{G}'$; since $\mathbb{G}$ is a substructure of $\mathbb{G}'$, $\mathcal{P}$ does not accept it either. □
We refer the reader to [24] and [13] for the formal definitions and details on the use of first-order and symmetric Datalog reductions. Roughly speaking, a reduction of CSP(\(\mathbb{H}\)) to CSP(\(\mathbb{K}\)) is first-order if there exist first-order formulas in the signature of \(\mathbb{H}\) that define, for every input structure \(G\) of CSP(\(\mathbb{H}\)), the universe and basic relations of an input structure \(G'\) to CSP(\(\mathbb{K}\)) such that \(G \rightarrow \mathbb{H}\) if and only if \(G' \rightarrow \mathbb{K}\). The reductions presented in Lemmas 5.4, 5.8 and 5.14 below also use parameters and satisfy an extra technical requirement to ensure that they preserve expressibility in symmetric Datalog: a first order reduction is positive if the formula defining the universe is quantifier-free, and the formulas defining the basic relations involve only the existential quantifier, the connectives AND and OR and atomic formulas (see [24] for details). Notice that a non-recursive symmetric Datalog reduction is a particular case of a positive first-order reduction.

**Lemma 5.4.** Let \(\mathbb{H}\) be a connected bipartite graph.

1. There is a symmetric Datalog reduction of CSP(\(\mathbb{H}^C\)) restricted to connected instances to CSP(\(\mathbb{H}^C\));

2. there is a non-recursive symmetric Datalog reduction of CSP(\(\mathbb{H}^C\)) to CSP(\(\mathbb{H}^C\)).

**Proof.** Our proof will describe the reductions rather informally but it is fairly straightforward to complete the details. As usual let \(D\) and \(U\) denote the colour classes of \(\mathbb{H}\). (1): Let \(G\) be a connected input to CSP(\(\mathbb{H}^C\)) the symmetric Datalog program outputs a coloured digraph \(G'\) that maps to \(\mathbb{H}^C\) if and only if \(G\) maps to \(\mathbb{H}^C\). We sketch briefly how the edge relation \(\mathcal{E}\) of \(G'\) can be constructed in symmetric Datalog from the edge relation \(E\) of \(G\) and the unary relations \(a(x), a \in H\). Consider first a binary IDB \(O(x, y)\) which interprets as “there is a path of odd length between \(x\) and \(y\)”, i.e. defined by the rules

\[
\begin{align*}
O(x, y) & : = E(x, y) \\
O(x, z) & : = O(x, y), E(w, y), E(w, z) \\
O(x, y) & : = O(x, z), E(w, y), E(w, z)
\end{align*}
\]

(Here for illustration purposes we assumed \(E\) is a symmetric relation; formally additional rules should appear to account for every possible orientation of the edges but this is straightforward.) First test if \(\mathbb{G}\) is bipartite, i.e. if there is an odd cycle in \(\mathbb{G}\) and if not, put a loop in \(\mathcal{E}\) to guarantee \(G'\) does not map to \(\mathbb{H}^C\):

\[
\begin{align*}
\mathcal{E}(x, x) & : = O(x, x) \\
O(x, x) & : = \mathcal{E}(x, x).
\end{align*}
\]

We can test similarly if vertices of \(\mathbb{G}\) are coloured consistently with \(\mathbb{H}\) (i.e. the parity of distance is preserved): for instance for every \(a, b \in D\), we have the rules

\[
\begin{align*}
\mathcal{E}(x, x) & : = O(x, y), a(x), b(y) \\
O(x, x) & : = \mathcal{E}(x, x), a(x), b(y).
\end{align*}
\]

Notice that in both the above cases the symmetric rule is harmless since \(\mathcal{E}(x, x)\) can only occur once we have detected the input graph does not map to \(\mathbb{H}^C\). So we may now assume that \(\mathbb{G}\) is bipartite, with colour classes \(A\) and \(B\), and that coloured vertices in \(A\) have a colour in \(D\) and coloured vertices in \(B\) have a colour in \(U\). Obviously if we orient the edges of \(\mathbb{G}\) from \(A\) to \(B\), the resulting digraph
will map to $\overline{H}^C$ if and only if $G$ maps to $\overline{H}^C$. For every $a \in D$, we have the rules

$$
\delta'(z, y) : - \quad O(x, y), E(z, y), a(x) \\
O(x, y) : - \quad \delta'(z, y), E(z, y), a(x)
$$

The first rule interprets as follows: if $x$ has colour $a \in D$, then $x \in A$; if there is an odd path from $x$ to $y$, then $y \in B$, so the edge $(z, y)$ should be oriented in that direction. Notice that the symmetric of this rule is harmless, because the input $G$ is connected: indeed, if $\delta'(z, y)$ holds and $x$ has colour $a \in D$, then we have $x \in A$ and $y \in B$ so there is an odd path from $x$ to $y$. Similarly, we orient the edge in the appropriate direction if the edge goes in the other direction:

$$
\delta'(z, y) : - \quad O(x, y), E(y, z), a(x) \\
O(x, y) : - \quad \delta'(z, y), E(y, z), a(x)
$$

The case where the colour $a$ is in $U$ is similar. It should be clear that at the end of the run of the program the IDB $\delta'$ will contain exactly the edges of $G_u$ contained in $A \times B$, thus orienting the edges of $G$ correctly. Notice finally that if $G$ is bipartite and has no coloured vertices, then it maps trivially to $\overline{H}^C$; and that the IDB $\delta'$ will remain empty so $G'$ also maps trivially to $\overline{H}^C$.

(2): Let $G$ be an input to $\text{CSP}(\overline{H}^C)$, let $E$ denote its edge relation, and let $a(x)$ denote the unary colour relations, $a \in H$. Our symmetric Datalog reduction is the following:

$$
\delta'(x, x) : - \quad E(x, y), E(y, z) \\
\delta'(x, x) : - \quad E(y, x), a(x) \quad \text{(one such rule for every } a \in D) \\
\delta'(x, x) : - \quad E(x, y), b(x) \quad \text{(one such rule for every } b \in U) \\
\delta'(x, y) : - \quad E(x, y)
$$

The first rule tests whether the directed path of length 2 maps to $G$; if it does, then obviously $G$ does not map to $\overline{H}^C$. If it does not, then $G$ is a strongly bipartite digraph: let $A$ and $B$ denote its colour classes such that $E(G) \subseteq A \times B$. The next two rules test if some colours are badly placed. Hence we may assume that $G$ is strongly bipartite with colour classes $A$ and $B$ and that coloured vertices in $A$ have a colour in $D$ and coloured vertices in $B$ have a colour in $U$, and all edges of $G$ are in $A \times B$. Hence it follows that $G$ maps to $\overline{H}^C$ if and only if it maps to $\overline{H}^C$, so we simply let $\delta'$ be the edge relation of $G$.

**Theorem 5.5.** Let $H$ be a connected, bipartite graph. If $H$ admits an NU polymorphism then (a) $\neg \text{CSP}(H^C)$ is solvable in symmetric Datalog and (b) $\text{CSP}(H^C)$ is solvable by arc-consistency on bipartite instances.

Notice that we could weaken the statement by replacing NU operation by monochromatic NU: indeed it is easy to verify that a connected, bipartite graph admits one if and only if it admits the other (see for instance the proof of Lemma 4.2 (4) $\Rightarrow$ (5)).

**Proof.** If $H$ admits an NU polymorphism then by Theorem 5.1 (1) $\overline{H}$ admits an NU polymorphism also. It follows by Theorem 5.1 (2) that $\neg \text{CSP}(\overline{H}^C)$, and hence $\neg \text{CSP}(\overline{H}^C)$ has finite duality; in particular it is solvable in symmetric Datalog so by Lemmas 5.3 and 5.4 (1) the first statement is proved. By Theorem 5.1 (2) we also have that $\text{CSP}(\overline{H}^C)$ is solvable by arc-consistency, hence by Lemma 5.2 $\text{CSP}(\overline{H}^C)$ is solvable by arc-consistency on bipartite instances, which proves (b).
5.2. **Two examples.** We do not know if the converse of Theorem 5.5 holds, in fact we do not know of an example of a reflexive graph or bipartite graph with no NU polymorphism whose retraction problem is in symmetric Datalog.\(^4\) On the other hand, we can produce examples of reflexive and bipartite graphs whose retraction problem is in *linear* Datalog, but these graphs admit no NU polymorphisms; in fact they do not even admit binary weak NU operations (Proposition 5.15).

Consider the 9-vertex reflexive graph \(\mathcal{R}\) in Figure 1 (loops are not depicted.) It is known that this graph admits no NU polymorphism, see [23]. Notice furthermore that every idempotent operation on \(\mathcal{R}\) preserves the set \(\{0, 1\}\), as it consists of precisely those vertices adjacent to both \(1'\) and \(u\); similarly, every set \(\{n, n+1\}\) (modulo 4) is invariant under any idempotent polymorphism of \(\mathcal{R}\).

![Figure 1. The reflexive graph \(\mathcal{R}\).](image)

We prove that the problem \(\neg\text{CSP}(\mathcal{R}^C)\) is solvable in linear Datalog, and construct a bipartite graph \(\mathcal{H}\) from \(\mathcal{R}\) in such a way that this property is preserved.

**Definition 5.6.** Let \(\mathcal{K}\) denote the relational structure consisting of the subgraph of \(\mathcal{R}\) induced by \(\{0, 1, 2, 3, u\}\) together with the sets \(\{n, n+1\}\) for \(n = 0, 1, 2, 3\) (modulo 4).

**Lemma 5.7.** The graph \(\mathcal{R}\) admits no binary weak NU polymorphism.

**Proof.** Notice that the vertex set of \(\mathcal{K}\) consists of all neighbours of the vertex \(u\), and hence is preserved by every idempotent operation on \(\mathcal{R}\). Thus it suffices to prove that the structure \(\mathcal{K}\) admits no idempotent, commutative binary operation. Let \(f\) be an edge-preserving idempotent operation on \(\mathcal{K}\) that preserves the sets \(\{n, n+1\}\). Assume without loss of generality that \(f(0, 1) = 0\); this forces \(f(1, 2) = 1\) and thus \(f(2, 3) = 2\). On the other hand \(f(0, 1) = 0\) also forces \(f(3, 2) = 3\), so \(f\) cannot be commutative. \(\Box\)

**Lemma 5.8.** The problems \(\text{CSP}(\mathcal{R}^C)\) and \(\text{CSP}(\mathcal{K}^C)\) are equivalent under positive first-order reductions.

**Proof.** Let \(\mathcal{G}\) be an instance of \(\text{CSP}(\mathcal{R}^C)\). Create an instance \(\mathcal{G}'\) of \(\text{CSP}(\mathcal{K}^C)\) as follows: (i) to every vertex \(x\) adjacent to some vertex coloured by some \(n'\), add the

\(^4\)Added in the editorial process: Ross Willard has recently found examples of such graphs.
list \( \{ n - 1, n \} \); (ii) for every vertex coloured by some \( n' \), change its colour to \( n \); (iii) add a vertex coloured by \( u \) adjacent to every vertex of \( G \). It is easy to describe \( G' \) using a 2-ary, 2-parameter positive FO reduction. We claim that \( G' \to K^C \) if and only if \( G \to R^C \). Indeed, suppose \( f : G \to R^C \). If \( x \) is adjacent to a vertex coloured \( n' \) then \( f(x) \in \{ n', n, n + 1 \} \). Furthermore, if \( g \) is uncoloured and \( f(g) = n' \), we can replace the value by \( n \) (it dominates \( n' \) in \( R^C \)) and we still have a homomorphism. Thus the new function defines a homomorphism from \( G' \to K^C \). Conversely, it is clear that a homomorphism from \( G' \to K^C \) also defines a homomorphism from \( G \to R^C \).

Now let \( G \) be an input to \( CSP(K^C) \). We construct an input \( G' \) to \( CSP(R^C) \) such that \( G \to K^C \) if and only if \( G' \to R^C \): if a vertex \( v \) of \( G \) has the list \( \{ n - 1, n \} \), remove the list and add to \( G \) a neighbour of \( v \) coloured by \( n' \). Secondly, add a vertex coloured \( u \) adjacent to every vertex of \( G \). This construction can be achieved by a 2-ary, 4-parameter positive FO reduction.

**Lemma 5.9.** The structure \( K \) admits a majority polymorphism.

*Proof.* Let \( m(x, y, z) = u \) when \( x, y, z \) are distinct, and majority otherwise. It is immediate that \( m \) is edge-preserving and that it preserves every 2-element set. □

**Corollary 5.10.** The problem \( \neg CSP(R^C) \) is solvable in linear Datalog.

*Proof.* Since the structure \( K \) is invariant under a majority operation it follows from [9] that \( \neg CSP(K^C) \) is solvable in linear Datalog. By Lemma 5.8, we have a positive first-order reduction of \( \neg CSP(K^C) \) to \( \neg CSP(R^C) \) so by Corollary 2.14 of [24] \( \neg CSP(R^C) \) is also solvable in linear Datalog. □

**Proposition 5.11.** The problem \( \neg CSP(R^C) \) is not solvable in symmetric Datalog.

*Proof.* By Lemma 5.8 and results in [12] and [24] (see also Lemmas 15 and 17 of [13]), it suffices to find a primitive, positive definition of the 2-element ordering from the structure \( K \) (this shows that the algebra associated to \( K \) has a 2-element subalgebra of (lattice) type 4). Consider the binary relation defined by \( \theta = \{ (f(x), f(y)) : f : P \to K, f \text{ a homomorphism} \} \) where \( P \) is the structure pictured in Figure 2. A label above a vertex indicates it belongs to this relation. It is easy to verify that \( \theta = \{ (0, 0), (0, 1), (1, 1) \} \).

![Figure 2: Structure \( P \) that pp-defines the ordering from \( K \).](image)

Now we describe the bipartite graph \( B \) we require: we use the functor \( I \) described in [2]. Let \( B = I(\mathbb{R}) \) where \( I(G) \) is defined for any reflexive graph \( G \) as follows: its vertex set is \( D \cup U \) where \( D \) is the set of vertices of \( G \) and \( U \) is the set of maximal cliques of \( G \); \( x \in D \) is adjacent to \( C \in U \) if \( x \in C \). It is easy to see that \( I(G) \) is connected if \( G \) is. In our case, the graph \( B \) has 9 vertices in one colour class.
Let \( I \) be any reflexive graph \( \mathbb{G} \). Recall that an identity of the form \( f(x_1, \ldots, x_k) = f(y_1, \ldots, y_k) \) (where the \( x_i, y_i \) are not necessarily distinct) is called \textit{linear}.

**Lemma 5.12.** Let \( \mathbb{G} \) be any connected reflexive graph, and let \( f : I(\mathbb{G})^k \to I(\mathbb{G}) \) be a monochromatic idempotent polymorphism. Then the restriction of \( f \) to the vertex set of \( \mathbb{G} \) is a polymorphism of \( \mathbb{G} \) that satisfies every linear identity satisfied by \( f \).

\[ \text{Proof.} \] As mentioned after Definition 3.1, \( f \) preserves the colour classes of \( I(\mathbb{G}) \) hence the operation \( F : \mathbb{G}^k \to \mathbb{G} \) given by \( F(x_1, \ldots, x_k) = f(x_1, \ldots, x_k) \) is well-defined. It is immediate that \( F \) satisfies any linear identity that \( f \) does. Now let \( (x_i, y_i) \) be edges of \( \mathbb{G} \), \( i = 1, \ldots, k \). Then we may find maximal cliques \( C_i \) of \( \mathbb{G} \) such that \( x_i, y_i \in C_i \) for all \( i \); hence \( f(x_1, \ldots, x_k) \) and \( f(y_1, \ldots, y_k) \) are adjacent to the maximal clique \( C = f(C_1, \ldots, C_k) \) in \( I(\mathbb{G}) \), i.e. \( F(x_1, \ldots, x_k) \) and \( F(y_1, \ldots, y_k) \) belong to the clique \( C \) and hence must be adjacent.

\[ \square \]

**Corollary 5.13.** The graph \( \mathbb{B} \) admits no monochromatic binary idempotent, commutative polymorphism.

\[ \text{Proof.} \] Immediate by Lemmas 5.7 and 5.12. \[ \square \]

**Lemma 5.14.** Let \( \mathbb{M} \) be a connected reflexive graph.

\[ \begin{align*}
(1) & \quad \text{The problem } \text{CSP}(\mathbb{M}^C) \text{ reduces to } \text{CSP}(I(\mathbb{M})^C) \text{ under a positive first-order reduction;} \\
(2) & \quad \text{if } \neg\text{CSP}(\mathbb{M}^C) \text{ is solvable in (linear, symmetric) Datalog then } \neg\text{CSP}(I(\mathbb{M})^C) \text{ is solvable in (linear, symmetric) Datalog.}
\end{align*} \]

\[ \text{Proof.} \] (Parts of these reductions are from [16]). Let \( \mathbb{H} = I(\mathbb{M}) \).

(1): Let \( \mathbb{G} \) be an input for \( \text{CSP}(\mathbb{M}^C) \). Construct a bipartite instance \( \mathbb{G'} \) of \( \text{CSP}(\mathbb{H}^C) \) as follows: one colour class \( A \) consists of the vertices of \( \mathbb{G} \) (with appropriate colours if applicable), the other colour class \( B \) consists of the edges of \( \mathbb{G} \); a vertex \( v \) of \( A \) if adjacent to an element \( e \) of \( B \) if \( v \) is incident to \( e \) in \( \mathbb{G} \) (it is straightforward to define this via (for instance) a 4-ary, 2-parameter positive first-order reduction.) We claim that \( \mathbb{G'} \to \mathbb{H}^C \) if and only if \( \mathbb{G} \to \mathbb{M}^C \). Suppose first that there is a colour-preserving homomorphism \( f : \mathbb{G} \to \mathbb{M}^C \). We define a homomorphism \( F \) from \( \mathbb{G'} \) to \( \mathbb{H}^C \) as follows: first \( F(a) = f(a) \) for every \( a \in A \). Then for every edge \( e = (x, y) \) of \( \mathbb{G} \), \( f(x) \) and \( f(y) \) are contained in some maximal clique \( C \), so define \( F(e) = C \) (any clique will do). It is immediate that \( F \) is a homomorphism and colour-preserving. Conversely suppose there is a colour-preserving homomorphism \( F \) from \( \mathbb{G'} \) to \( \mathbb{H}^C \); notice that connected components of \( \mathbb{G} \) correspond to connected components of \( \mathbb{G'} \), and that in both cases components with no coloured elements map trivially to the target. So we may assume that every component of \( \mathbb{G'} \) contains at least one coloured element, ensuring that \( F \) maps the set \( A \) to vertices of \( \mathbb{M} \). Define \( f : \mathbb{G} \to \mathbb{M} \) by \( f(x) = F(x) \) for all \( x \); it is immediate that \( f \) is a colour-preserving homomorphism.

(2) Let \( \mathcal{P} \) denote the problem \( \text{CSP}(\mathbb{H}^C) \) with inputs restricted to bipartite digraphs which are consistently coloured, i.e. if \( x \) and \( y \) are in the same component and are coloured by \( a \) and \( b \) respectively, then the parity of the distance from \( x \) to \( y \) in \( \mathbb{G} \) is the same as that between \( a \) and \( b \) in \( \mathbb{H} \). We first prove that there is
There exists a bipartite graph $S$. Arora, B. Barak. CSP Theorem 3.2, a positive, first-order reduction of $P$ to $CSP(M^C)$. Let $G$ be a bipartite input to $CSP(M^C)$. (Note that since both targets are symmetric, orientation of the edges of $G$ is immaterial). We construct an input $G'$ of $CSP(M^C)$ (via an $|M|$-ary, 2-parameter positive first-order reduction). In short, the vertices of $G'$ are those of $G$ together with $|M|$ new vertices, one for each vertex of $M$ and coloured accordingly; vertices with a colour from $M$ retain this colour, all other colours are removed. Vertices $u$ and $v$ of $G$ are adjacent in $G'$ if they share a common neighbour in $G$; and if a vertex $u \in G$ is adjacent in $G$ to a vertex coloured by the clique $C = \{c_1, \ldots, c_q\}$, we add an edge from $u$ to each of the new vertices coloured by $c_1, \ldots, c_q$ respectively.

We claim that $G \rightarrow \mathbb{H}^C$ if and only if $G' \rightarrow M^C$. Let $A$ and $B$ denote the colour classes of $G$ such that, if $x \in A$ is coloured then its colour is in $M$. Notice that since $G$ is consistently coloured, the subgraph of $G'$ induced by $B$ is not connected to the rest of $G'$ and contains no coloured vertex. Thus this subgraph plays no part in the rest of our argument. Suppose first that $f : G \rightarrow \mathbb{H}^C$; then $f$ defines naturally a map on $G'$. If $x$ and $y$ in $A$ are adjacent in $G'$, it means there is some $b \in B$ adjacent to both in $G$, so $f(x), f(y)$ belong to the clique $f(b)$ and hence are adjacent in $G'$. If $x \in A$ is adjacent (in $G'$) to some new vertex $m$ (correspondingly, and coloured by colour $m \in M$), by construction there exists $b \in B$ coloured by a clique $C$ containing $m$ such that $x$ and $b$ are adjacent in $G$, so $f(x) \in C$ and thus $f(x)$ is adjacent to $m$ as required. Conversely suppose that $F : G' \rightarrow M^C$. Define $f : G \rightarrow \mathbb{H}$ by $f(x) = F(x)$ if $x \in A$; if $b \in B$ is uncoloured, let $x_1, \ldots, x_s$ be its neighbours in $A$. Then by definition of $G'$ $F(x_1), \ldots, F(x_s)$ form a clique in $M$ so let $f(b) = C$ where $C$ is any maximal clique of $M$ that contains all of them. We claim $f$ is edge-preserving. Indeed, if $a \in A$ is adjacent to an uncoloured $b \in B$, then by definition $f(b)$ is a clique containing $f(a)$; if on the other hand $b$ is coloured by some maximal clique $C$, by construction of $G'$ $f(a)$ is adjacent to every $c \in C$. Since $C$ is a maximal clique, $f(a) \in C$ and we're done.

Now suppose that we have a (linear, symmetric) Datalog program for $\neg CSP(M^C)$. By the preceding argument we have a positive first-order reduction of $CSP(I(M)^C)$ restricted to consistently coloured bipartite instances to $\neg CSP(M^C)$, and hence by Corollary 2.14 of [24] we have a (linear, symmetric) Datalog program to determine if a consistently coloured bipartite instance maps to $I(M)^C$. As in the proof of Lemma 5.4, we can test in symmetric Datalog whether an arbitrary input is bipartite and consistently coloured; add these rules to the program to obtain a (linear, symmetric) Datalog program for $CSP(I(M)^C)$.

**Proposition 5.15.** There exists a bipartite graph $\mathbb{B}$ such that $\neg CSP(\mathbb{B}^C)$ is solvable in linear Datalog, but $CSP(\mathbb{B}^C)$ is not solvable by arc-consistency on bipartite instances (and of course $\mathbb{B}$ admits no NU polymorphism).

**Proof.** Let $\mathbb{B} = I(\mathbb{R})$ where $\mathbb{R}$ is the graph in Figure 1. By Lemma 5.14 $\neg CSP(\mathbb{B}^C)$ reduces to $\neg CSP(\mathbb{R}^C)$ in symmetric (and in particular linear) Datalog, and hence by Corollary 5.10 $\neg CSP(\mathbb{B}^C)$ is solvable in linear Datalog. By Corollary 5.13 and Theorem 3.2, $CSP(\mathbb{B}^C)$ is not solvable by arc-consistency on bipartite instances.

**References**

20 B. LAROSE AND A. LEMAÎTRE


LIST-HOMOMORPHISM PROBLEMS ON GRAPHS AND ARC CONSISTENCY

Department of Mathematics and Statistics, Concordia University, 1455 de Maisonneuve West, Montréal, QC, Canada, H3G 1M8
E-mail address: larose@mathstat.concordia.ca
URL: http://cicma.mathstat.concordia.ca/faculty/larose/

Département d’informatique et de recherche opérationnelle, Pavillon André Eisenstadt, 2920 chemin de la tour Montréal, QC, Canada H3T 1J4
E-mail address: lemaistra@iro.umontreal.ca