Dichotomy for Digraph Homomorphism Problems

Arash Rafiey * Tomás Feder† Jeff Kinne ‡

Abstract

We consider the problem of finding a homomorphism from an input digraph G to a fixed digraph H. We show that if H admits a weak-near-unanimity polymorphism ϕ then deciding whether G admits a homomorphism to H (HOM(H)) is polynomial time solvable. This confirms the conjecture of Bulatov, Jeavons, and Krokhin [BJK05], in the form postulated by Maroti and McKenzie [MM08], and consequently implies the validity of the celebrated dichotomy conjecture due to Feder and Vardi [FV93]. We transform the problem into an instance of the list homomorphism problem where initially all the lists are full (contain all the vertices of H). Then we use the polymorphism ϕ as a guide to reduce the lists to singleton lists, which yields a homomorphism if one exists.

1 Introduction

We have prepared two video presentations for this result and we encourage the reader to watch the videos first. We ask the community for help in checking the correctness of our algorithm. We welcome any comments and suggestions.
https://youtu.be/PBX51Qv5wtw and https://youtu.be/vD68km24Rh0

Our Result in Context For a digraph G, let $V(G)$ denote the vertex set of G and let $A(G)$ denote the arcs (aka edges) of G. An arc (u, v) is often written as simply uv to shorten expressions.

*Indiana State University, IN, USA arash.rafiey@indstate.edu and Simon Fraser University, BC, Canada, arashr@sfu.ca

†268 Waverley Street, Palo Alto, CA 94301, United States, tomas@theory.stanford.edu

‡Indiana State University, IN, USA, jkinne@cs.indstate.edu

A *homomorphism* of a digraph G to a digraph H is a mapping g of the vertex set of G to the vertex set of H so that for every arc uv of G the image $g(u)g(v)$ is an arc of H. A natural decision problem is whether for given graphs G and H there is a homomorphism of G to H. If we view undirected graphs as digraphs in which each edge is replaced by the two opposite directed arcs, we may apply the definition to graphs as well. An easy reduction from the k-coloring problem shows that this decision problem is NP-hard: a graph G admits a 3-coloring if and only if there is a homomorphism from G to K_3, the complete graph on 3 vertices. As a homomorphism is easily verified if the mapping is given, the homomorphism problem is contained in NP and is thus NP-complete.

The following version of the problem has attracted much recent attention. For a fixed digraph H the problem $HOM(H)$ asks if a given input digraph G admits a homomorphism to H. Note that while the above reduction shows $HOM(K_3)$ is NP-complete, $HOM(H)$ can be easy (in P) for some graphs H: for instance if H contains a vertex with a self-loop, then every graph G admits a homomorphism to H. Less trivially, for $H = K_2$ (or more generally, for any bipartite graph H), there is a homomorphism from G to K_2 if and only if G is bipartite. A very natural goal is to identify precisely for which digraphs H the problem $HOM(H)$ is easy. In the special case of undirected graphs (when for each arc uv of H there is also an arc vu of H) the classification has turned out to be this: if H contains a vertex with a self-loop or is bipartite, then $HOM(H)$ is in P, otherwise it is NP-complete [HN90] (see [B05, S10] for shorter proofs). This classification result implies a *dichotomy* of possibilities for the problems $HOM(H)$ when H is an undirected graph, each problem being NP-complete or in P. However, the dichotomy of $HOM(H)$ remained open for general digraphs H. It was observed by Feder and Vardi [FV93] that this problem is equivalent to the dichotomy of a much larger class of problems in NP, in which H is a fixed finite relational structure. These problems can be viewed as *constraint satisfaction problems* with a fixed template H [FV93], written as $CSP(H)$. A constraint satisfaction problem $CSP(H)$ consists of (a) a relational structure H that specifies a set V of variables that each come from some domain D and (b) a set C of constraints giving restrictions on the values allowed on the variables.

The question is whether all constraints can be simultaneously satisfied. 3SAT is a prototypical instance of CSP, where each variable takes values of *true* or *false* (a domain size of two) and the clauses are the constraints. Digraph homomorphism problems can also easily be converted into CSPs: the variables V are the vertices of G, each must be assigned a vertex in H (meaning a domain size of $|V(H)|$), and the constraints encode that each arc of G must be mapped to an arc in H.

Feder and Vardi argued in [FV93] that in a well defined sense the class of problems $CSP(H)$ would be the largest subclass of NP in which a dichotomy holds. A fundamental result of Ladner [L75] asserts that if $P \neq NP$ then there exist NP-intermediate problems (problems neither in P nor NP-complete), which implies that there is no such dichotomy theorem for the class of *all* NP problems. Non-trivial and natural subclasses which do have dichotomy theorems are of great interest. Feder and Vardi made the following *Dichotomy Conjecture*: every problem $CSP(H)$ is NP-complete or is in P. This problem has animated much research in theoretical computer science. For instance the conjecture has been verified when H is a conservative relational structure [B11], or a digraph with all in-degrees and all-out-degrees at least one [BKN09].

Numerous special cases of this conjecture have been verified [ABISV09, B06, BH90, BHM88, CVK10, D00, F01, F06, FMS04, LZ03, S78].

It should be remarked that constraint satisfaction problems encompass many well known computational problems, in scheduling, planning, database, artificial intelligence, and constitute an important area of applications, in addition to their interest in theoretical computer science [CKS01, D92, V00, K92].

While the paper of Feder and Vardi [FV93] did identify some likely candidates for the boundary between easy and hard CSP-s, it was the development of algebraic techniques by Jeavons [J98] that lead to the first proposed classification [BJK05]. The algebraic approach depends on the observation that the complexity of $CSP(H)$ only depends on certain symmetries of H, the so-called *polymorphisms* of H. For a digraph H a polymorphism ϕ of arity k on H is a homomorphism from H^k to H. Here H^k is a digraph with vertex set $\{(a_1, a_2, \ldots, a_k) | a_1, a_2, \ldots, a_k \in V(H)\}$ and arc set $\{(a_1, a_2, \ldots, a_k)(b_1, b_2, \ldots, b_k) \mid a_i b_i \in A(H) \text{ for all } 1 \leq i \leq k\}$. For a polymorphism ϕ, $\phi(a_1, a_2, \ldots, a_k)\phi(b_1, b_2, \ldots, b_k)$ is an arc of H whenever $(a_1, a_2, \ldots, a_k)(b_1, b_2, \ldots, b_k)$ is an arc of H^k.

Over time, one concrete classification has emerged as the likely candidate for the dichotomy. It is expressible in many equivalent ways, including the first one proposed in [BJK05]. There were thus a number of equivalent conditions on H that were postulated to describe which problems $CSP(H)$ are in P. For each, it was shown that if the condition is not satisfied then the problem $CSP(H)$ is NP-complete (see also the survey [HN08]). One such condition is the existence of a weak near-unanimity polymorphism (Maroti and McKenzie [MM08]). A polymorphism ϕ of H of arity k is a k *near unanimity function* (k-nuf) on H, if $\phi(a, a, \ldots, a) = a$ for every $a \in V(H)$, and $\phi(a, a, \ldots, a, b) = \phi(a, a, \ldots, b, a) = \cdots = \phi(b, a, \ldots, a) = a$ for every $a, b \in V(H)$. If we only have $\phi(a, a, \ldots, a) = a$ for every $a \in V(H)$ and $\phi(a, a, \ldots, a, b) = \phi(a, a, \ldots, b, a) = \cdots = \phi(b, a, \ldots, a)$ [not necessarily a] for every $a, b \in V(H)$, then ϕ is a *weak k-near unanimity function* (weak k-nuf).

Given the NP-completeness proofs that are known, the proof of the Dichotomy Conjecture reduces to the claim that a relational structure H which admits a weak near-unanimity polymorphism has a polynomial time algorithm for $CSP(H)$. As mentioned earlier, Feder and Vardi have shown that is suffices to prove this for $HOM(H)$ when H is a digraph. This is the main result of our paper.

Note that the real difficulty in the proof of the *graph* dichotomy theorem in [HN90] lies in proving the NP-completeness. By contrast, in the *digraph* dichotomy theorem proved here it is the polynomial-time algorithm that has proven more difficult.

While the main approach in attacking the conjecture has mostly been to use the highly developed techniques from logic and algebra, and to obtain an algebraic proof, we go in the opposite direction and develop a combinatorial algorithm.

Our main claim is the following.

Theorem 1.1 (Main Claim) *Let H be a digraph that admits a weak near-unanimity function. Then $HOM(H)$ is in P.*

Together with the NP-completeness result of [MM08], this settles the CSP Conjecture in the affirmative.

Our Methods, Very High Level View We start with a general digraph H and a weak k-nuf ϕ of H. We turn the problem $HOM(H)$ into a related problem of seeking a homomorphism with lists of allowed images. The *list homomorphism problem* for a fixed digraph H, denoted $LHOM(H)$, has as input a digraph G, and for each vertex x of G an associated list (set) of vertices $L(x) \subseteq V(H)$, and asks whether there is a homomorphism g of G to H such that for each $x \in V(G)$, the image $g(x)$ is in $L(x)$. Such a homomorphism is called a *list homomorphism* of G to H with respect to the lists L. List homomorphism problems are known to have nice dichotomies. For instance when H is a reflexive undirected graph (each vertex has a loop), the problem $LHOM(H)$ is polynomial when H is an interval graph and is NP-complete otherwise [FH98]. Similar list homomorphism dichotomies were proved for general graphs [FHH03, FHH07], and more recently also for digraphs [HR11]. In fact, motivated by the results in [FH98, FHH03], Bulatov [B11] proved that the list version of constraint satisfaction problems has a dichotomy for general relational systems.

It is not difficult to see that there are digraphs H such that $HOM(H)$ is polynomial while $LHOM(H)$ is NP-complete. For instance, the reflexive four-cycle H has loops and so $HOM(H)$ is trivial, while $LHOM(H)$ is NP-complete since H is not an interval graph. However, we transform the problem $HOM(H)$ into a restricted version of $LHOM(H)$ in which the lists satisfy an additional property related to the weak k-nuf ϕ.

One of the common ingredients in CSP algorithms is the use of consistency checks to reduce the set of possible values for each variable (see, for example the algorithm outlined in [HN04] for $CSP(H)$ when H admits a near unanimity function). Our algorithm includes such a consistency check as a first step. We begin by performing a pair consistency check of the list of vertices in the input digraph G. For each pair (x, y) of $V(G) \times V(G)$ we consider a list of possible pairs (a, b), $a \in L(x)$ (the list in H associated with $x \in G$) and $b \in L(y)$. Note that if xy is an arc of G and ab is not an arc of H then we remove (a, b) from the list of (x, y). Moreover, if $(a, b) \in L(x, y)$ and there exists z such that there is no c for which $(a, c) \in L(x, z)$ and $(c, b) \in L(z, y)$ then we remove (a, b) from the list of (x, y). We continue this process until no list can be modified. If there are empty lists then clearly there is no list homomorphism.

After performing pair consistency checks (and repeating the consistency checks throughout the algorithm), the main structure of the algorithm is to perform *pairwise elimination*, which focuses on two vertices a, b of H that occur together in some list $L(x), x \in V(G)$, and finds a way to eliminate a or b from $L(x)$ without changing a feasible problem into an unfeasible one. In other words if there was a list homomorphism with respect to the old lists L, there will still be one with respect to the updated lists L. This process continues until either a list becomes empty, certifying that there is no homomorphism with respect to L (and hence no homomorphism at all), or until all lists become singletons, specifying a concrete homomorphism of G to H. This technique, due to the lead author, has been successfully used in several other papers [HR11, HR12, EHLR14]. In this paper, the choice of which a or b is eliminated, and how, is governed by the given weak near-unanimity polymorphism ϕ. In fact, we define a family of mappings $f_x, x \in V(G)$ which are each polymorphisms derived from ϕ and use these polymorphisms as a guide. The heart of the algorithm is a delicate procedure for updating the lists $L(x)$ and polymorphisms f_x in such a way that (i) feasibility is maintained, and (ii) the polymorphisms f_x remain polymorphisms (which is key to maintaining feasibility).

2 Algorithm

2.1 An introduction to the Algorithm

In this section we introduce the main parts of our algorithm. We encourage the reader to take time to read and internalize as much of this section as possible, while consulting the figures that are referenced.

1. We associate to each vertex $x \in V(G)$ a list $L(x)$, with each $L(x)$ initially $V(H)$. We also consider pair lists $L(x, y)$, where $(a, b) \in L(x, y)$ means that it may be possible for x and y to simultaneously map to a and b for the same homomorphism.

2. We let the polymorphism on H be specialized for each vertex in G. We define a homomorphism $f : G \times H^k \to H$, i.e. $f(x; a_1, a_2, \ldots, a_k)$ for $x \in V(G)$ and $a_1, ..., a_k \in V(H)$. Initially $f(x; a_1, ..., a_k) = \phi(a_1, ..., a_k)$ where ϕ is the weak k-nuf polymorphism given to us. We call f a *weak k-nuf homomorphism from G to H* (see Def 2.1).

3. PreProcessing. We perform standard consistency checks that are done in CSP algorithms to prune the lists $L(x)$ for every x, and pair lists $L(x, y)$ for each x and y.

4. We refine f and work towards building a homomorphism g from G to H. The main loop in the algorithm picks a vertex $x \in V(G)$ and tries to remove one of the vertices $a \in V(H)$ from consideration for being $g(x)$. At any given time, we have lists $L(x) \subseteq H$ for each $x \in V(G)$ that have the vertices in H that are being considered for $g(x)$.

5. For $a, b \in L(x)$, if $f(x; b, b, ..., b, a) = a$ we say this is the *minority case*. The minority case is similar to a case of the homomorphism problem that is already solved, namely the setting where the underlying polymorphism is Maltsev (see subsection 2.4). We thus have our main loop choose x, b, a where $f(x; b, b, ..., b, a) \neq a$ and attempt to remove a from consideration. If $f(x; b, b, ..., b, a) = a$ then we leave it alone.

6. The main procedure, then, is to take an $x \in V(G)$, $a, b \in L(x)$ such that $f(x; b, b, ..., b, a) \neq a$ and remove a from consideration – that is, remove a from $L(x)$. We would like $f(x; ...)$

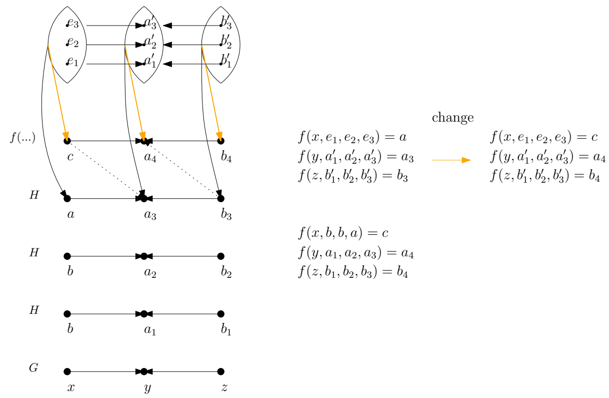

Figure 1: The solid lines are arcs, dotted lines are missing arcs, and no line means either could be true. In the first column, initially $f(x; e_1, e_2, e_3) = a$ but we set $f(x; e_1, e_2, e_3) = c$ where $c = f(x; b, b, a)$. In the second column, initially $f(y; a'_1, a'_2, a'_3) = a_3$ but we change it to $f(y; a'_1, a'_2, a'_3) = a_4$ where $a_4 = f(y; a_1, a_2, a_3)$. The change in the first column is so we can remove a from $L(x)$; the change in the second column is to preserve the homomorphism property (this change is only needed if there is a missing arc from c to a neighbor a_3 of a).

to align with a possible homomorphism $g : G \to H$ (if one exists), so if we remove a from $L(x)$ (meaning we will not have $g(x) = a$) then we would also prefer if it is not the case that $f(x; e_1, e_2, ..., e_k) = a$ for any $e_1, e_2, ..., e_k \in L(x)$. Since we are considering a because $f(x; b, b, ..., b, a) = c \neq a$, we decide to take c as the new value for $f(x; e_1, e_2, ..., e_k)$.

7. This change to set $f(x; e_1, ..., e_k) = c$ could break the homomorphism property of f (if $(y; a'_1, a'_2, a'_3)$ exists with $xy \in A(G)$, $e_i a'_i \in A(H)$ and $cf(y; a'_1, a'_2, a'_3) \notin A(H)$). If so, we update the values of f on these neighbors of x. See Figure 1.

8. DFS of updates to f. We begin by changing f values from a to something else (c) on x, and this forces some changes for f on y's that are neighbors of x, which in turn forces some changes for f on neighbors of y. This results in a DFS through $G \times H^k$. We can imagine a subgraph G_f of $G \times H^k$ where the vertices are the columns in Figure 1 (for $k = 3$) and there is a connection between $(y; a_1, a_2, a_3)$ and $(z; b_1, b_2, b_3)$ if a change to f occurs on the DFS.

9. Cycles in DFS and weak k-nuf property. The DFS of updates to f should be careful of difficulties such as cycles. We leverage the weak k-nuf property to make the DFS of updates consistent. An *almost cycle* is a near cycle in the DFS of updates from some vertex $(x; b, b, .., b, a)$ to $(x; b, ..., b, a, b, ..., b)$ where the only change in the vertex is the coordinate of the a (see Figure 3). The value of f on the two endpoints is the same, due to the weak k-nuf property. This gives the DFS a definite ending point (once the

end of an almost cycle is reached, any neighbors in the DFS from the ending point would also be neighbors of the beginning point).

10. At each step in the DFS, one of the coordinates (e.g., a_3 in the second column of Figure 1) is forced due to a missing arc (the dotted lines in the figures). The remaining coordinates (e.g., a_1 and a_2 in the second column of Figure 1) are more free, and these are chosen to ensure that the DFS of updates to f always moves in the direction of a shortest almost cycle (Figure 3). Different choices for the "more free" coordinates result in different paths through G_f because they use different values for updating f.

11. The hope is that the DFS can complete, and we have fixed any places the homomorphism property for f was broken. If we can fix all the problems, then f would still be a homomorphism after the procedure is done. Further, for the proof of correctness we need to make sure that if we had a correct homomorphism g in mind, then after removing a from $L(x)$ we could change g to remain a homomorphism with respect to the lists L and set $g(x)$ to something other than a.

Outline We have given most of the main ideas that are part of the algorithm and proofs. We give definitions to setup the algorithm in Section 2.2, precisely define the algorithm in Section 2.3, and give the correctness proof in Section 3. We encourage the reader to begin by reading Section 2.3 and consult back to Section 2.2 as needed. We also encourage the reader to look forward to Section 3 to keep in mind the main structure of the proofs.

2.2 Definitions

This section contains definitions that are used in the algorithms in Section 2.3 and the proofs in Section 3.

An *oriented walk (path)* is obtained from a walk (path) by orienting each of its edges. The *net-length* of a walk W, is the number of forward arcs minus the number of backward arcs following W from the beginning to the end. An *oriented cycle* is obtained from a cycle by orienting each of its edges. We say two oriented walk X, Y are congruent if they they follow the same patterns of forward and backward arcs.

Given graphs G and H, let $G \times H^k$ be a graph on the vertices $\{(y; a_1, a_2, \ldots, a_k) | y \in V(G), a_i \in V(H), i \leq i \leq k\}$ with the arcs $(y; a_1, a_2, ..., a_k)(y'; b_1, b_2, ..., b_k)$ where yy' is an arc of G and each $a_i b_i$, $1 \leq i \leq k$, is an arc of H. By convention, we shall further restrict the use of the symbol $G \times H^k$ to the digraph induced on the vertices $\{(y; a_1, a_2, \ldots, a_k) | y \in V(G), a_i \in L(y), 1 \leq i \leq k\}$ where $L(y)$ is the set of vertices in H that are being considered as images of a homomorphism from G to H.

Definition 2.1 (Homomorphism consistent with Lists) *Let G and H be digraphs. For each $x \in V(G)$, let $L(x) \subseteq V(H)$. Let $k > 1$ be a constant integer.*

A function $f : G \times H^k \to H$ is a homomorphism consistent with L *if the following hold.*

- List property : *for every* $x \in V(G)$ *and every* $a_1, a_2, \ldots, a_k \in L(x)$, $f(x; a_1, a_2, \ldots, a_k) \in L(x)$
- Adjacency property: *for every* $x, y \in V(G)$ *and every* $a_1, ..., a_k \in L(x), b_1, ..., b_k \in L(y)$, *if* xy *is an arc of* G *and* $a_i b_i$ *is an arc of* H *for each* $1 \leq i \leq k$ *then* $f(x; a_1, ..., a_k) f(y; b_1, ..., b_k)$ *is an arc of* H.

In addition if f has the following property then we say f has the *weak nuf property*.

- for every $x \in V(G), \{a, b\} \subseteq L(x)$, we have $f(x; a, b, b, ..., b) = f(x; b, a, b, ..., b) = ... = f(x; b, b, b, ...a)$.

We note that this definition is tailored to our purposes and in particular differs from the standard definition of weak k-nuf as follows.

(a) f is based on two digraphs G and H rather than just H (we think of this as starting with a traditional weak k-nuf on H and then allowing it to vary somewhat for each $x \in G$),

(b) We do not require that $f(x; b, b, b, ..., b) = b$ (this is not required in our algorithm, and in fact is more convenient to leave out).

Notation Let $(b^k|_{i \leftarrow a}) = (b, b, ..., a, ..., b)$ be a k-tuple of all b's but with an a in the i^{th} coordinate.

Digraph of Updates to f Let G_f be a digraph where $V(G_f) = V(G \times H^k)$ and with arcs $(y; a_1, a_2, ..., a_k)(y'; b_1, b_2, ..., b_k)$ where $yy' \in A(G)$, $(y'y \in A(G))$ and $a_i b_i \in A(H)$ $(b_i a_i \in A(H))$, $1 \leq i \leq k$, and $f(y; a_1, a_2, \ldots, a_k) b_j$ $(b_j f(y; a_1, a_2, \ldots, a_k))$ is not an arc of H for some $1 \leq j \leq k$. In this case we say $(y; a_1, a_2, ..., a_k)$ avoids $(y'; b_1, b_2, ..., b_k)$ in coordinate j, or simply say the avoidance appears at the j-th coordinate. In Figure 1, $(y; a_1, a_2, a_3)$ is a neighbor of $(z; b_1, b_2, b_3)$ in G_f with avoidance in the third coordinate.

Let B be a walk in H starting at some vertex $a \in V(H)$. We say a path W in G_f *follows* B if whenever the i-th vertex of W, say $w_1 = (y; a_1, a_2, \ldots, a_k)$ avoids the $(i+1)$-th vertex of W, say $w_2 = (z; b_1, b_2, \ldots, b_k)$ at coordinate j $(f(w_1) b_j \notin A(H)$ if $yz \in A(G)$ or $b_j f(w_1) \notin A(H)$ if $zy \in A(G))$ and $a_{j'}$ is the i-th vertex of B and b_j is the $i+1$-th vertex of B then $a_{j'} = a_j$ (see Figure 2). We say W follows walk X in G when X is the walk induced by the first coordinate of the vertices in W.

Let $w = (y; a_1, a_2, \ldots, a_k)$ and $w' = (z; b_1, b_2, \ldots, b_k)$ be two vertices in G_f.

- We say a directed path P in G_f from w to w' is an (i, j)-path if there exists a walk in H from a_i to b_j that P follows.
- $G_f(w, i)$ is the set of vertices $w' \in G_f$ such that there exists an (i, j)-path (for some $1 \leq j \leq k$) from w to w'.

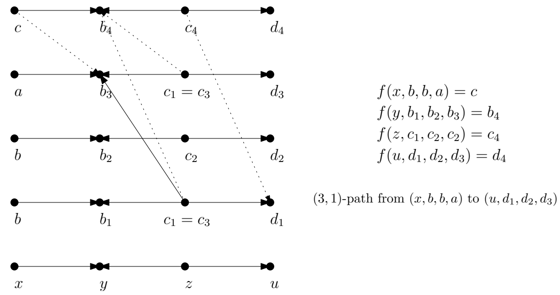

Figure 2: An illustration of the graph G_f, with each column representing a vertex in G_f. The dotted lines are missing arcs (solid lines are arcs, dotted lines are missing arc, and no line means either could be true). $(x; b, b, a)$ is connected to $(y; a_1, a_2, a_3)$ with avoidance at coordinate 3, which in turn is connected to $(z; b_1, b_2, b_3)$ with avoidance at coordinates 1, etc. The figure depicts a $(3, 1)$ path P in G_f – where the avoidance began on the 3rd coordinate and ended on the first coordinate, proceeding by the walk a, a_3, b_1, d_1 in H.

For a fixed vertex $x \in V(G)$ and fixed vertex $b \in V(H)$, let $G_{x,b}$, $1 \leq i \leq k$ be a digraph of pairs (i', a') where $f(x; b^k|_{i' \leftarrow a'}) \neq a'$ and there is an arc from (i', a') to (i'', a'') whenever there exists an (i', i'')-path in G_f from $w' = (x; b^k|_{i' \leftarrow a'})$ to $w'' = (x; b^k|_{i'' \leftarrow a''})$.

Let $w = (x; b^k|_{i \leftarrow a})$ and $w_1 = (x; b^k|_{i' \leftarrow a'})$. Suppose $f(w) \neq a$ and $f(w_1) \neq a'$.

- We say an oriented path W from w to w_1 (in $G \times H^k$) is a *quasi-cycle* when W is an (i, i')-path from w to w' in G_f (see Figure 3).
- If $a' = a$ then we say w is an *almost-cycle*. An almost-cycle also requires that for any $\lambda = (x; c_1, c_2, \ldots, c_k)$ on W, $\lambda = (x; b^k|_{\ell \leftarrow a''})$
- If arc $e \in P$ then we say e is *covered* by the quasi-cycle (almost-cycle) W.

Consider the digraph $G_{x,b}$. Let $S = G_{x,b}(i, a)$ be a strong component of $G_{x,b}$ containing (i, a). We say S is a *sink component* if there is no arc from an element in S to any other vertex in $G_{x,b}$ outside S. We also say S is trivial if it has only one element.

Let $w = (x; b^k|a \leftarrow i)$ and suppose $G_{x,b}(i, a)$ is a sink component in $G_{x,b}$. Let $G^r_f(w, i)$ (r for restricted) be the induced sub-digraph of $G_f(w, i)$ containing vertices $w' = (y; a_1, a_2, \ldots, a_k)$ reachable from w via an (i, j)-path such that for every $\ell \neq j$, $(b, a_\ell) \in L(x, y)$. Some of the vertices in $G^r_f(w, i)$ lie on almost cycles and some do not. We say an arc $w'w'' \in G^r_f(w, i)$ is *nice* if it lies on a shortest almost-cycle.

Remark: If G is an oriented tree (contains no oriented cycle) then it is easy to find a homomorphism from G to H. The complication arises when there are oriented cycles in G. This is the main motivation of defining almost-cycles.

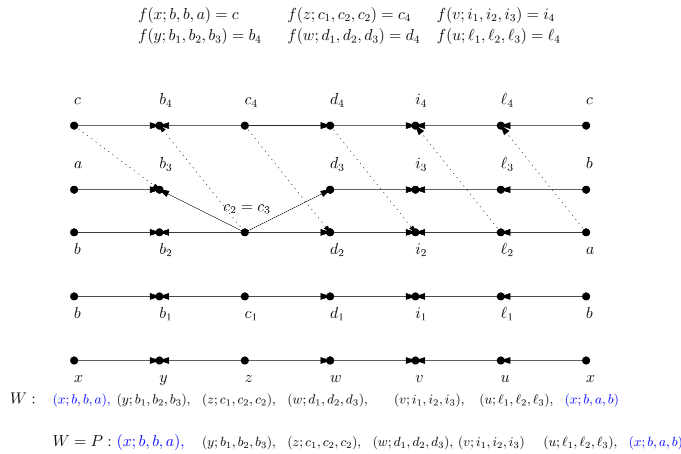

Figure 3: An almost-cycle in G_f. Each column is a vertex in G_f, and dotted lines are missing arcs.

2.3 Main Procedure

In this subsection we present the main algorithm. The main algorithm is Algorithm 1. Subroutines that are used by the main algorithm are Algorithms 3, 2, and a known result discussed in Section 2.4. Algorithm 3 (PreProcessing) simply updates the lists L of vertices in G based on local edge constraints, and also updates the pair lists of vertices in G (pair consistency); standard textbook CSP algorithms for the homomorphism problem would repeatedly invoke the PreProcessing routine, and then make a decision (often greedy, or trying all possible choices that remain).

Algorithm 2 (RemoveNotMinority) is the key subroutine of the main algorithm. It starts with $w = (x; b^k|_{\ell \leftarrow a})$ where $f(w) = c \neq a$ and then it starts modifying f by setting $f(x; e_1, e_2, \ldots, e_k) = f(w)$ for every k-tuple $e_1, e_2, \ldots, e_k \in L(x)$ with $f(x; e_1, e_2, \ldots, e_k) = a$. Now in order to have a homomorphism from $G \times H^k$ to H consistent with L it performs a depth first search (in the sub-digraph $G^r_f(w, \ell)$ of $G \times H^k$) to modify f as necessary. After the execution of Algorithm 2 we remove a from $L(x)$ in Algorithm 1.

After the main loop in Algorithm 1, we end up with a so-called Maltsev or minority instance of the problem – in which we have a homomorphism f consistent with L such that for every $y \in V(G)$ and every $c, d \in L(y)$ we have $f(y; c^k|_{i \leftarrow d}) = d$. We argue in the next subsection that such instances can be solved by using the known algorithm of [BD06]. The Maltsev/minority instances can also be solved in a manner similar to our other arguments.

Observation 2.2 *Let $w' = (y; a_1, a_2, \ldots, a_k)$ be a vertex that lies on an (ℓ, i)-path in $G^r_f(w, \ell)$. Suppose $yz(zy) \in A(H)$. Let $b' \in L(z)$ be an arbitrary out-neighbor (in-neighbor) of a_i, i.e. $a_i b' \in A(H)$. If $f(w')b' \notin A(H)$ ($b' f(w') \notin A(H)$) then there exists $w'' = (z; b_1, b_2, \ldots, b_j, \ldots, b_k)$ such that $w'w''$ is an arc of $G^r_f(w, \ell)$ and w' avoids w'' at coordinate j. Here $b' = b_j$.*

Algorithm 1 The main algorithm for solving the digraph homomorphism problem.

function DIGRAPHHOM(G, H, ϕ) ▷ G and H digraphs, ϕ a weak k-nuf on H
  for all $x \in G$, let $L(x) = V(H)$
  for all $x \in G$ and $a_1, ..., a_k \in V(H)$, let $f(x; a_1, ..., a_k) = \phi(a_1, ..., a_k)$
  PREPROCESSING(G, H, L)
  while $\exists x \in V(G), a', b \in L(x), a' \neq b$ s.t. $f(x; b^k|_{\ell' \leftarrow a'}) \neq a'$ **do**
    Let $a \in L(x)$ such that $G_{x,b}(\ell, a)$ is a sink.
    REMOVENOTMINORITY$(x; b^k|_{\ell \leftarrow a})$
    Remove a from $L(x)$
    PREPROCESSING(G, H, L)
  Note: now, for all $x \in V(G)$ and $a, b \in L(x)$ we know $f(x; b^k|_{i \leftarrow a}) = a$
  REMOVEMINORITY(G, H, L, f)
  Note: now, for all $x \in V(G)$, $|L(x)| = 1$
  if assigning x to $L(x)$ for each x is a homomorphism **then** return true
  else return false

This is because $w' \in G^r_f(w, \ell)$ and hence we have $(b, a_r) \in L(x, y)$, $1 \leq r \neq i \leq k$. Therefore there exist $b_1, b_2, \ldots, b_k \in L(z)$ such that $a_r b_r \in A(H)$, and $(b, b_r) \in L(x, z)$, $1 \leq r \neq j \leq k$ (because of the PreProcessing). Here if $i \neq j$ then $a_i = a_j$ according to the definition of "follow" (see also Figure 2). We let $w'' = (z; b_1, b_2, \ldots, b_k)$ and hence $w'w'' \in G^r_f(w, \ell)$ and w' avoids w'' at coordinate j.

Let $w' = (y; a_1, a_2, \ldots, a_k)$ be a vertex which lies on an (ℓ, i)-path in $G^r_f(w, \ell)$. Suppose yz (zy) is an arc in G. Let $N(w', z, i)$ denote the set of pairs (w'', j) obtained as follows.

$\forall\ b_j \in L(z)$ s.t. $a_i b_j \in A(H)$ $(b_j a_i \in A(H))$ and $f(w')b_j \notin A(H)$ $(b_j f(w') \notin A(H))$

a) Let $w'' = (z; b_1, \ldots, b_j, \ldots, b_k)$ s.t. $w'w''$ is an arc of $G^r_f(w, \ell)$ and w' avoids w'' at coordinate j.

b) If there is a choice then $w'w''$ is a nice arc.

Short description of Algorithm 2: At the current step of the NM-DFS we are at the vertex $w' = (y; a_1, a_2, \ldots, a_k)$ with coordinate i. We look at each neighbor of y, say z. In the list of z we look for all the neighbors b_j of $a_i \in L(y)$ for which the image of $w_1 = f(z; ...)$ should be changed from $b_j \in L(z)$ or $d \in L(z)$ (new value of $f(w_1)$) to an out-neighbor of $f(w')$. Thus we get different arcs $w'w''$ (all having arc yz (zy) of G). Every such w'' is added into a new P-List (the list of the vertices that DFS should follow next). Note that for each b_j only one element w'' is considered.

Remarks In Algorithm 2 the vertices $(x; b^k|i \leftarrow a)$ and $(x; b^k|\ell \leftarrow a)$ of G_f are considered as the same vertex. We also need to note that by Observation 2.2 it is enough to modify f inside $G^r_f(w, \ell)$.

Algorithm 2 Remove a from $L(x)$, updating f so it remains a homomorphism of $G \times H^k$ to H consistent with L.

1: **function** RemoveNotMinority($w = \langle x; b^k|_{\ell \leftarrow a}, \ell \rangle$)
2: **Input:** x, a, b such that $f(x; b^k|_{\ell \leftarrow a}) = c \neq a$ and $G_{x,b}(\ell, a)$ is a sink
3: **if** $\exists\, w_1 = (x; a'_1, a'_2, \ldots, a'_k)$ with $f(w_1) = a$ **then**
4: NM-DFS($w = \langle x; b^k|_{\ell \leftarrow a} \rangle, \ell$)

5: **function** NM-DFS($w' = \langle y; a_1, a_2, ..., a_k \rangle$, i) ▷ Change $f(y; \cdot)$ for those that $= a_i$
6: **if** w' already visited at coordinate i in NM-DFS **then return**
7: **for all** $z \in V(G) | yz \in A(G) (zy \in A(G))$ **do**
8: Construct $N(w', z, i)$.
9: **for all** $(w'', j) \in N(w', z, i)$ **do**
10: Let $d \in L(z)$ s.t. $f(z; ...)$ was changed from b_j to d. ▷ $d = b_j$ if no change
11: **if** $f(w')d \notin A(H)$ $(df(w') \notin A(H))$ **then**
12: **for all** $w_2 = (z; b'_1, b'_2, \ldots, b'_k)$ if $f(w_2)$ was changed from b_j to d **do**
13: set $f(w_2) = b_j$
14: **if** $\exists\, w_1 = (z; b'_1, b'_2, \ldots, b'_k)$ such that $f(w_1) = b_j$ **then**
15: Add (w'', j) into a new P-List. ▷ if something has b_j as its value
16: **for all** $(w'', j) \in$ P-List, and $\forall\, w_1 = (z; b'_1, b'_2, ..., b'_k)$ s.t $f(w_1) = b_j$ **do**
17: Set $f(w_1) = f(w'')$. ▷ change $f(z; ...)$ from b_j to $f(w'')$
18: **for all** $(w'', j) \in$ P-List **do**
19: NM-DFS(w'', j)

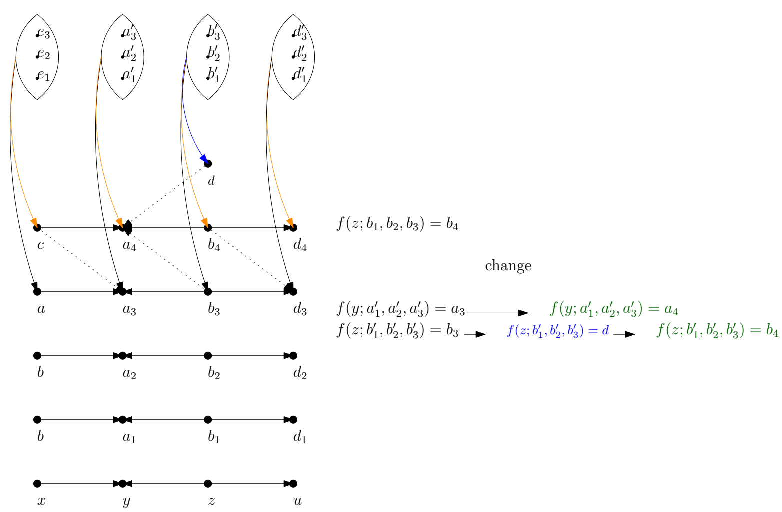

Figure 4: Line 10 from Algorithm 2 where $f(z; b'_1, b'_2, b'_3)$ changed from b_3 to d at an earlier point in the NM-DFS, and so now is changed to b_4 along with other points that map to b_3.

The NM-DFS is used for two purposes. One is to modify f such that at the end f is a homomorphism consistent with lists L and for any k-tuple $e_1, e_2, \ldots, e_k$, $f(x; e_1, e_2, \ldots, e_k) \neq a$. The NM-DFS is also used to show that if there exists a homomorphism $g : G \to H$ consistent with lists L with $g(x) = a$ then there exists a homomorphism $h : G \to H$ consistent with lists L and with $h(x) \neq a$.

Let us partition the $G^r_f(w, \ell)$ into two parts. Let $G^a_f(w, \ell)$ ('a' for almost-cycle) be the subdigraph of $G^r_f(w, \ell)$ containing vertices w' such that w' lies on an almost cycle containing w. Let $G^o_f(w, \ell)$ ('o' is for other) be the sub-digraph of $G^r_f(w, \ell)$ containing those vertices that do not lie on an almost-cycle. We note that we need to modify f inside $G^r_f(w, \ell)$ by Observation 2.2.

Consider an oriented cycle X in G containing vertex x and suppose there exists a homomorphism $g : G \to H$ with $g(x) = a$. Let $g(X)$ denote the image of X under g (an oriented cycle in H containing a and congruent with X.

The goal is to search in $L(X)$ (list of the vertices in X, for all such X) to find an oriented path in $L(X)$ congruent to X containing c where $f(x; b^k|\ell \leftarrow a) = c$. Now a path R from (w, ℓ) to $w' = (x; b^k|j \leftarrow a)$ in $G^a_f(w, \ell)$ which follows $g(X)$ is a shortest almost-cycle (observe that in X each arc is repeated once). We have considered such path R in the NM-DFS. We note that walking through $G^o_f(w, \ell)$ as far as g is concerned could be arbitrary because we do not reach $(x; b^k|j \leftarrow a)$ from w in $G^r_f(w, \ell)$.

Algorithm 3 Update lists of x, y based on edge constraints and pair constraint

```
function PREPROCESSING(G, H, L)
    Input: digraphs G, H, lists L(x) ⊆ V(H) for each x
    ARCCONSISTENCY(G, H, L); PAIRCONSISTENCY(G, H, L)

    function ARCCONSISTENCY(G, H, L)
        update=True
        while update do
            if ∃xy(yx) ∈ A(G), a ∈ L(x) s.t. ∄b ∈ L(y) with ab(ba) ∈ A(H) then
                remove a from L(x) and set update=True.
            else update=False.
        if there is an empty list then return no homomprphism

    function PAIRCONSISTENCY(G, H, L)
        for every (x, y) ∈ V(G) × V(G) set L(x, y) = {(a, b)|a ∈ L(x), y ∈ L(y)}.
        update=True
        while update do
            if ∃x, y, z s.t. ∄c ∈ L(z) s.t. (a, c) ∈ L(x, z)&(c, b) ∈ L(z, y) then
                remove (a, b) from L(x, y) and set update=True.
            else update=False.
        if there is an empty list then return no homomprphism
```

2.4 Minority Algorithm (RemoveMinority)

In this section we show that once the minority case has been reached in our main algorithm, we can reduce to an already solved setting for homomorphism testing – namely that of the Maltsev case. We note that this section is independent of the rest of the algorithm.

Note that at this point for every $a, b \in L(x)$ we have $f(x; b^k|_{i \leftarrow a}) = a$ and in particular when $a = b$ we have $f(x; a, a, \ldots, a) = a$ (idempotent property). This is because when a is in $L(x)$ then it means the RemoveNotMinority procedure did not consider a and in fact did not change the value of $f(x; ...)$ from a to something else. Note that for the argument below we just need the idempotent property for those vertices that are in $L(x)$, $x \in V(G)$.

A ternary polymorphism h' is called Maltsev if for all $a \neq b$, $h'(a, b, b) = h'(b, b, a) = a$. Note that the value of $h'(b, a, b)$ is unspecified by this definition.

Let G and H be as input to Algorithm 1, and suppose line 10 of the algorithm has been reached. We define a homomorphism $h : G \times H^3 \to H$ consistent with the lists L by setting $h(x; a, b, c) = f(x; a, b, b, \ldots, b, c)$ for $a, b, c \in L(x)$. Note that because f has the minority property for all $x \in G$, $a, b \in L(x)$, h is a Maltsev homomorphism consistent with the lists L.

Note that for every $a, b \in L(x)$ we have $f(x; b^k | a \leftarrow a) = a$ and in particular if $a = b$ we have $f(x; a, a, \ldots, a) = a$ (idempotent property). Since a is in $L(x)$, this means the RemoveNotMinority procedure did not considered a and in fact did not change the value of $f(x; ...)$ from a to something else. Note that for the argument below we just need the idempotent property for those vertices that are in $L(x)$, $x \in V(G)$.

Let G' be the structure obtained from G by making each arc a different binary relation. In other words, G' has vertices $V(G)$ and $|E(G)|$ binary relations $R_e, e \in E(G)$, where $R_e = \{xy\}$ if e is the arc $e = xy$.

Let H' be the structure where $V(H')$ is the disjoint union of $L(x), x \in V(G)$, and there are also $|E(G)|$ binary relations $S_e, e \in E(G)$, where S_e is the set of all ordered pairs ab with $ab \in E(H), a \in L(x), b \in L(y)$, where $e = xy$. Note that $|V(H')| \geq |V(G')|$ if each $L(x)$ is non-empty. This may seem unusual for the homomorphism setting, but is certainly allowed.

Now note that there is an L-homomorphism of G to H (i.e., a list homomorphism consistent with lists L) if and only if there is a homomorphism of G' to H'. Homomorphisms of such structures are mappings $f : V(G') \to V(H')$ such that $xy \in R_e$ implies $f(x)f(y) \in S_e$ for all $e \in E(G)$.

Finally, note that the structure H' has a Maltsev polymorphism h' of the ordinary kind. Indeed, let h_x be our Maltsev polymorphisms defined on $L(x)$ by setting $h_x(a, b, c) = h(x; a, b, c)$. We let $h'(a, b, c) = h(x; a, b, c)$ if a, b, c are from the same $L(x)$, and for a, b, c not from the same $L(x)$ define $h'(a, b, c) = a$ unless $a = b$, in which case define it as $h'(a, b, c) = c$. The definition ensures that h is Maltsev. To check it is a polymorphism, note that $aa', bb', cc' \in S_e$ is only possible if $a, b, c \in L(x), a', b', c' \in L(y)$, where $e = xy$. For those, we have the polymorphism property by assumption.

Now we have a structure with a Maltsev polymorphism, so the Bulatov-Dalmau [BD06] algorithm applies and solves the homomorphism problem. Note that Corollary 4.2 of the Bulatov-Dalmau paper explicitly mentions that it is polynomial in both the sizes of G and H.

Therefore we have the following theorem.

Theorem 2.3 *Suppose $h : G \times H^k \to H$ is a minority homomorphism consistent with lists L on G. Then the existence of an L-homomorphism of G to H can be decided in polynomial time.*

We note that it is also possible to give a direct algorithm for the minority case that is similar to how we handle the "not minority" case.

3 Proofs

3.1 Arc-Consistency and List Update

We first show that the standard properties of consistency checking remain true in our setting – namely, that if the PreProcessing algorithms succeed then f remains a homomorphism consistent with the lists L if it was before the pre-processing.

Lemma 3.1 *If f is a homomorphism of $G \times H^k \to H$ consistent with L then f is a homomorphism consistent with L after running the pre-processing.*

Proof: We need to show that if $a_1, a_2, \ldots, a_k$ are in $L(y)$ after the pre-processing then $f(y; a_1, a_2, \ldots, a_k) \in L(y)$ after the pre-processing. By definition vertex a is in $L(y)$ after the pre-processing because for every oriented path Y (of some length m) in G from y to a fixed vertex $z \in V(G)$ there is a vertex $a \in L(z)$ and there exists a walk B in H from a to a' and congruent with Y that lies on $L(Y)$. $L(Y)$ denote the vertices that are in the list of the vertices of Y.

Let $a'_1, a'_2, a'_3, \ldots, a'_k \in L(z)$. Let A_i, $1 \leq i \leq k$ be a walk from a_i to a'_i in $L(Y)$ and congruent to Y. Let $A_i = a_i, a^i_1, a^2_i, \ldots, a^m_i, a'_i$ and let $Y = y, y_1, y_2, \ldots, y_m, z$.

Since f is a homomorphism consistent with L before the pre-processing, $f(y; a_1, a_2, \ldots, a_k)$, $f(y_1; a^1_1, a^1_2, \ldots, a^1_k), \ldots,$ $f(y_i; a^i_1, a^i_2, \ldots, a^i_k), \ldots, f(y_m; a^m_1, a^m_2, \ldots, a^m_k), f(z; a'_1, a'_2, \ldots, a'_k)$ is a walk congruent with Y. This would imply that there is a walk from $f(y; a_1, a_2, \ldots, a_k)$ to $f(z; a'_1, a'_2, \ldots, a'_k)$ congruent with Y in $L(Y)$. and hence $f(y; a_1, a_2, \ldots, a_k) \in L(y)$. Note that here z and Y. $\diamond$

By a similar argument as in the proof of Lemma 3.1 we have the following lemma.

Lemma 3.2 *If f is a homomorphism of $G \times H^k \to H$, consistent with L and $a_1, a_2, \ldots, a_k \in L(x)$, $b_1, b_2, \ldots, b_k \in L(y)$, and $(a_i, b_i) \in L(x, y)$, $1 \leq i \leq k$, after pre-processing then $(f(x; a_1, a_2, \ldots, a_k), f(y; b_1, b_2, \ldots, b_k)) \in L(x, y)$ after the pre-processing.*

3.2 RemoveNotMinority Correctness Proof

The main argument is proving that after each step of the RemoveNotMinority function f still is a homomorphism consistent with the lists and has the weak nuf property (Lemma 3.3). Moreover, after removing a vertex from the list of a vertex of G there still exists a homomorphism from G to H if there was one before removing that vertex from the list (Lemma 3.4).

Lemma 3.3 *If f is a homomorphism of $G \times H^k \to H$, consistent with L with weak nuf property before RemoveNotMinority($w = \langle x; b^k|_{\ell \leftarrow a} \rangle, \ell$) then the modified f remains a homomrphism consistent with L with weak nuf property afterwards. Moreover for every k-tuple $a_1, a_2, \ldots, a_k \in L(x)$, $f(x; a_1, \ldots, a_k) \neq a$.*

Lemma 3.4 *If there is a homomorphism $g : G \to H$ with $g(x) = a$ then there is a homomorphism from G to H after removing a from $L(x)$ according to RemoveNotMinority($\langle x; b^k|_{\ell \leftarrow a} \rangle, \ell$).*

3.2.1 Proof of Lemma 3.3

In order to show that f still is a homomorphism consistent with L with weak nuf property, we need to address items 1,2,3,4 below.

1. The weak nuf property is preserved: $f(y; b_1^k|_{i \leftarrow b_2}) = f(y; b_1^k|_{j \leftarrow b_2})$ for any $y \in G$, $b_1, b_2 \in L(y)$, $1 \leq i, j \leq k$.

2. The RemoveNotMinority function stops.

3. The adjacency property is preserved: for an arbitrary arc $yz \in A(G)$ ($zy \in A(G)$) and for every $a'_1, a'_2, \ldots, a'_k \in L(y)$ and $b'_1, b'_2, \ldots, b'_k \in L(z)$ where $a'_i b'_i \in A(H)$ ($b'_i a'_i \in A(H)$), $1 \leq i \leq k$, we have $f(y; a'_1, a'_2, \ldots, a'_k) f(z; b'_1, b'_2, \ldots, b'_k) \in A(H)$ ($f(z; b'_1, b'_2, \ldots, b'_k) f(y; a'_1, a'_2, \ldots, a'_k) \in A(H)$).

4. We argue that we only need to fix the problems caused by changing $f(x; ...)$ from a to c in $G_f^r(< x; b^k|_{\ell \leftarrow a} >, \ell)$. We show that if $f(x; b^k|_{k \leftarrow a}) = c \neq a$ and $f(x; e_1, e_2, \ldots, e_k) = a$ then $f(x; e_1, e_2, \ldots, e_k)$ is changed to c (in the simple case) or to some other fixed element (which will be used in the next lemma) not equal to a after the execution of RemoveNotMinority($x; b^k|_{\ell \leftarrow a}$).

Proof of 1 : Since we change the value $f(y; a'_1, a'_2, ..., a'_k)$ from a_j to $f(y; a_1, a_2, \ldots, a_k)$ for every k-tuple $a'_1, a'_2, \ldots, a'_k \in L(y)$, we change $f(y; b_1, b_1, \ldots, b_1, b_2) = f(y; b_1, b_1, \ldots, b_2, b_1) = \cdots = f(y; b_2, b_1, \ldots, b_1)$ to the same value. Therefore f still has the weak k-nuf property.

Proof of 2 : Observe that during the NM-DFS the value of $f(y; a'_1, a'_2, \ldots, a'_k)$, $y \neq x$ may change several times. This is because after changing $f(y; a'_1, a'_2, \ldots, a'_k)$ from a_j to some a_{k+1} there might be the case that there is a path A' from a to a_{k+1} in H and some (ℓ, j) path Q in $G_f^r(w, k)$ to some vertex $(y; c_1, c_2, \ldots, c_{j-1}, a_{k+1}, c_{j+1}, \ldots, c_k)$.

However, the value of $f(\theta)$, $\theta = (x; e_1, e_2, \dots, e_k)$ is changed once or the value of $f(\theta)$ may change a number of times while going through a circuit (explained in Case 2 in the proof of (4)) but each time to a new value.

No value is changed back to the one we already changed (the only place that this happens is in Line 13; for remembering the previous image but we immediately change the value to something else in line 16).

Note that the NM-DFS algorithm may end up traversing all the vertices in G_f. There are at most $|G||H^k|$ vertices in G_f.

Proof of 3 : Let $a_{k+1} = f(y; a'_1, a'_2, \dots, a'_k)$ and $b_{k+1} = f(z; b'_1, b'_2, \dots, b'_k)$. If $a_{k+1}b_{k+1}$ is an arc of H then we are done. Otherwise without loss of generality we consider the following two cases.

(a) The original value of $f(y; a'_1, a'_2, \dots, a'_k)$ was a_j and it was changed to a_{k+1} and the value of $f(z; b'_1, b'_2, \dots, b'_k)$ was not changed.

(b) The original value of $f(y; a'_1, a'_2, \dots, a'_k)$ was a_j and it was changed to a_{k+1} and the value of $f(z; b'_1, b'_2, \dots, b'_k)$ was changed after changing the value of $f(y; a'_1, a'_2, \dots, a'_k)$.

We assume these changes are the last changes according to the NM-DFS. In other words, a_{k+1} is the last value of $f(y; a'_1, a'_2, \dots, a'_k)$ and b_{k+1} is the last value of $f(x; b'_1, b'_2, \dots, b'_k)$.

According to the function RemoveNotMinority, there is an (ℓ, j)-path W from $w = (x; b^k|_{\ell \leftarrow a})$ to $w' = (y; a_1, a_2, \dots, a_k)$ in $G^r_f(w, \ell)$.

We also note that the current vertex in the NM-DFS is vertex w' together with coordinate j. Let b_j be the original value of $f(z; b'_1, b'_2, \dots, b'_k)$. We note that $a_j b_j$ is an arc because initially f is a homomorphism consistent with L. Since $a_{k+1}b_j \notin A(H)$, by Observation 2.2 there exists $w'' = (z; b_1, b_2, \dots, b_k)$ such that $w'w'' \in A(G^r_f(w, \ell))$ (for simplicity we assume b_j is in the $(j+1)$-th coordinate of $(z; b_1, b_2, \dots, b_k)$).

Proof of (a) : Here $b_{k+1} = b_j$. Since $w'w'' \in A(G^r_f(w, \ell))$, the NM-DFS should have taken the arc $w'w''$ and should have changed the value of $f(z; b'_1, b'_2, \dots b'_k)$ from b_j to $f(w'')$ ($w'' = (z; b_1, b_2, \dots, b_k)$) which is a contradiction. Note that by Claim 3.5, $f(w'')$ is an out-neighbor of a_{k+1}.

Proof of (b): We assume $f(z; b'_1, b'_2, \dots, b'_k)$ did change after $f(y; a'_1, a'_2, \dots, a'_k)$. Either vertex w'' is immediately after w' (and the value of $f(z; b'_1, b'_2, \dots, b'_k)$ is going to change for the first time) in the NM-DFS and this would mean we should have changed the value of $f(z; b'_1, b'_2, \dots, b'_k)$ as we discussed earlier. Or the value of $b_j = f(z; b'_1, b'_2, \dots, b'_k)$ did change following some path $Z : y, y_1, \dots, y_r, z$ in G together with a path R in $G^r_f(w, \ell)$ starting at w'. By adding arc yz (zy) we close an oriented cycle in G. Suppose the $(j+1)$-th coordinate of the last vertex in R is b_j (again here for simplicity we consider the $j+1$-th coordinate).

Since $Z(yz)$ is a closed walk it may be that the last vertex of R is $w'' = (z; b_1, b_2, \dots, b_j, \dots, b_k)$ and hence by following the modification of f along the path R we set $f(w'') = b_{k+1}$. Since

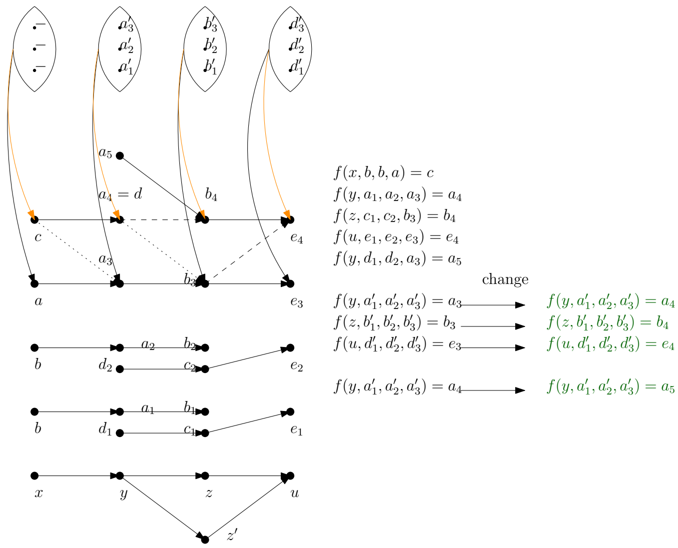

Figure 5: Keep in mind for the proof of 3 (b) in the proof of Lemma 3.3

f is a homomorphism for the vertices in R (by Claim 3.5), $f(w')f(w'')$ is an arc and hence $a_{k+1}b_{k+1}$ must be an arc of H. In other words, the modification is consistent.

Now let us assume that the last vertex of R is not w''. Suppose the last vertex of R is $\omega = (z; c_1, c_2, \ldots, c_{j-1}, b_j, c_{j+1}, \ldots, c_k)$. Now in this case according to line 11 of the NM-DFS (see y as z and z as y in the algorithm, see also Figure 5) we have $d = a_{k+1}$ and db_{k+1} is not an arc of H.

Now first suppose $a_j b_{k+1} \notin A(H)$ (in Figure 5, a_3b_4) then according to the rules of NM-DFS we should have considered the vertex $\omega' = (y; d_1, d_2, \ldots, d_{j-1}, a_j, d_{j+1}, \ldots, d_k)$ where $\omega\omega' \in G^r_f(w,\ell)$ and hence we should have changed the value of $f(y, a'_1, a'_2, \ldots, a'_k)$ from a_{k+1} to an in-neighbor of b_{k+1} (by Claim 3.5 $f(\omega)f(\omega')$ is an arc). This is a contradiction. Second suppose $a_j b_{k+1}$ is an arc (in Figure 5, a_3b_4). In this case the the arc $w'w_2$ where $w' = (y; a_1, a_2, \ldots, a_k)$ and $w_2 = (z; b_1, b_2, \ldots, b_{j-1}, b_{k+1}, b_{j+1}, \ldots, a_k)$ (in Figure 5, $(y; a_1, a_2, a_3)(z; b_1, b_2, b_4)$) is in $G^r_f(w,\ell)$. Therefore $w'w_2$ should have been taken by NM-DFS and we should have changed the value of $f(z, b'_1, b'_2, \ldots, b'_k)$ from b_{k+1}, a contradiction that was the last value (here again by Claim 3.5 $f(w')f(w_2)$ is an arc).

Claim 3.5 *At each step of the NM-DFS if $w'w''$ is an arc in $G^r_f(w,\ell)$ then $f(w')f(w'')$ ($f(w'')f(w')$, depending on the orientation) is an arc of H.*

Proof: Suppose $w' = (y; a_1, a_2, \ldots, a_k)$ and $w'' = (z, b_1, b_2, \ldots, b_k)$. Initially $f(w')f(w'')$ is an arc if $w'w'' \in G \times H^k$ or $f(w')f(w'')$ is an arc if $w''w' \in G \times H^k$ because f is a homomorphism consistent with the lists. If $f(w')f(w'')$ is not an arc then it must be the case that $f(w')$

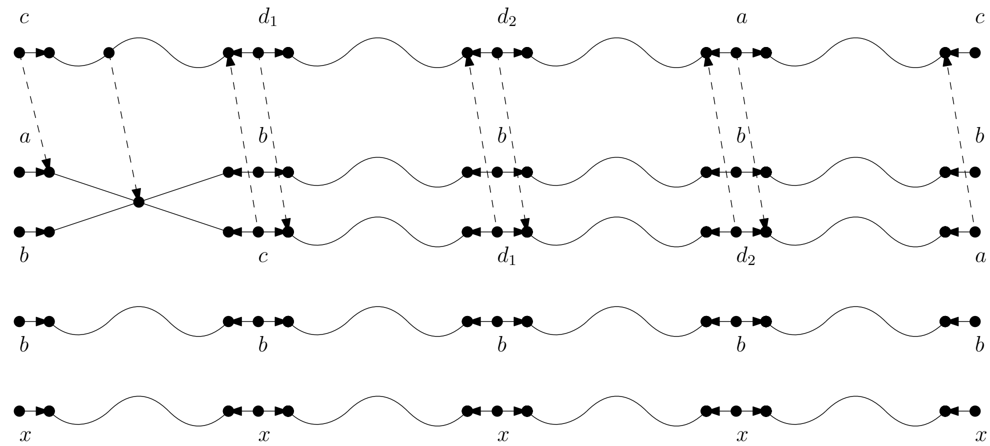

Figure 6: Circuit of length 2

did change and $f(w'')$ did not change. This means there was a (ℓ,ℓ')-path Q_1 from (w,ℓ) to some $\omega=(y,c_1,c_2,\ldots,c_k)$ in $G^r_f(w,\ell)$ and the NM-DFS went along Q_1 and it did change $f(y;a_1,a_2,\ldots,a_k)$ from $c_{\ell'}$ to a different value. Now since $w'w''$ ($w''w'$) is an arc of $G\times H^k$, $f(w'')=d_{\ell'}$ is an out-neighbor (in-neighbor) of $c_{\ell'}$. Note that $f(w')$ is changed to $f(\omega)$ according to the NM-DFS. If $f(\omega)d_{\ell'}\notin A(H)$ then the arc $(y;c_1,c_2,\ldots,c_k)(z;d_1,d_2,\ldots,d_k)$ (for appropriate vertices $d_1,d_2,\ldots,d_k$ in $L(z)$) should have been considered according to the NM-DFS (lines 8,16 dealing with P-List) and $f(w'')$ should have been changed to an out-neighbor of $f(y;c_1,c_2,\ldots,c_k)$.

Proof of 4 : The NM-DFS starts from vertex (w,ℓ) where $w=(x;b^k|_{\ell\leftarrow a})$. The changes to f are forced in the list of vertices y where $w'=(y,a_1,a_2,\ldots,a_k)\in G_f$, i.e. $f(y,\ldots)$ should be changed from a_i to something else. This means there exists an (ℓ,i) path from w to w'. Since we only need to change the image of f in y from a_i, we are free to choose the a_j, $j\neq i$ in $L(y)$ and hence we may assume that NM-DFS is performed in $G^r_f(w,\ell)$. We also note that since (ℓ,a) is a sink component of $G_{x,b}(\ell,a)$, any shortest almost-cycle lies entirely in $G^r_f(w,\ell)$.

Now in the remaining we analyze the behavior of NM-DFS depending on the final value of $f(x;e_1,e_2,\ldots,e_k)$ that is initially a. The final value is used in Lemma 3.4. Let $C=G_{x,b}(\ell,a)$ be the strong component of $G_{x,b}$ containing (ℓ,a) as described before. We say vertex $(i,a)\in C$ is a source if there is no path from (i,a) to (j,c) in C where $f(x;b^k|_{i\leftarrow a})=c$. We say the sink component C is *invertible* if it does not have a source.

Suppose C does not have a source. Now according to the definition of *invertible*, there is a path in C starting at $(i=i_0,a)$ that goes through the vertices $(i_1,c),(i_2,d_1),\ldots,(i_t,d_{t-1}),(i_{t+1},d_t)$ in this order. Thus there exist corresponding vertices $\alpha=(x;b^k|_{i_0\leftarrow a}),\beta=(x;b^k|_{i_1\leftarrow c}),\lambda_j=(x;b^k|_{i_{j+1}\leftarrow d_j})$, $1\leq j\leq t$ and there exists a quasi-cycle Q in G^r_f from α to λ_t that goes through $\beta, \lambda_1, \lambda_2, \ldots, \lambda_t$ in this order and $f(\alpha) = c$, $f(\beta) = d_1$, $f(\lambda_1) = d_2$, $\ldots$, $f(\lambda_{t-1}) = d_t$, $f(\lambda_t) = a$. If $t = 1$ then $d_1 = a$ and $\lambda_{t+1} = \alpha$. We say Q is a *circuit* in $G^r_f(\alpha, i)$ (see Figure 6). Let $Q(t)$ be an almost-cycle obtained from concatenating the path Q and a path Q' from λ_t to $\alpha' = (x; b^k|j \leftarrow a)$ in $G^r_f(\alpha, i)$. Note that path Q' exists because C is a strong component of $G_{x,b}$. Note that here for simplicity we may assume that the last vertex of the circuit would be λ_t with $f(\lambda_t) = a$. It could be the case the d_i plays the role of a.

Observe that if there exists an arc from (ℓ, a) to (i', a'') then this means we do need to modify $f(x; \beta'_1, \beta'_2, \ldots, \beta'_k)$ from a'' to $f(w_1)$ where $w_1 = (x; b^k|_{i' \leftarrow a''})$.

Now we consider two cases depending on whether C is invertible or not.

Case 1. C is not invertible.

First suppose C has only elements (i, a). We note that we do not see any $(x; b^k|_{i' \leftarrow a''})$ for $a'' \neq a$ in the NM-DFS. This means we do not end up changing a value of $f(x; \alpha'_1, \alpha'_2, \ldots, \alpha'_k)$ once we change it to c at the beginning.

So we may assume that C has at least one element of form (i, a'), $a' \neq a$. Since C (a component of $G_{x,b}$) is a non-trivial strong component, there exists a path P' in $G^r_f(w', i')$ to w where $w' = (x; b^k|_{i' \leftarrow a'})$. In this case P' follows a walk in G that may go through x multiple times until it reaches $(x; b^k|_{\ell \leftarrow a})$. This would mean that along P' we are modifying f and in particular we are modifying f in $G^r_f(w', i')$. Note that since (ℓ, a) is a source and C is a sink component, there is no arc from (ℓ, a) to some (j, c) where $f(w) = c$, and hence there is no path in $G^r_f(w, k)$ from w to $(x; b^k|_{j \leftarrow c})$. This means we do not end up changing a value of $f(x; \alpha'_1, \alpha'_2, \ldots, \alpha'_k)$ after changing it from a to c and then changing it from c to something else.

Case 2. C is invertible. We need to show that if there is a path Q_1 from (ℓ, a) to (ℓ, c) (we choose ℓ for simplicity) in C then function NM-DFS would handle this situation and it fixes the value of $f(x; b'_1, b'_2, \ldots, b'_k)$ from a to some element not a. The main purpose of this case would be finding that fixed element.

Suppose such path Q_1 exists in C. Since C is invertible, there exists some path Q_2 from (ℓ, c) to (ℓ, d_t) which goes through the vertices $(\ell, d_1), (\ell, d_2), \ldots, (\ell, d_{t-1})$. Consequently this means there exist vertices : $\alpha = w, \beta = (x; b^k|_{\ell \leftarrow c}), \lambda_i = (x; b^k|_{\ell \leftarrow d_i}), 1 \leq i \leq t$ and there exists a path $Q : \alpha, \ldots, \beta, \ldots, \lambda_1, \ldots, \lambda_2, \ldots, \lambda_t$ in $G^r_f(\alpha, \ell)$ where $f(\alpha) = c$, $f(\beta) = d_1$, $f(\lambda_1) = d_2$, $\ldots$, $f(\lambda_{t-1}) = d_t$, $f(\lambda_t) = a$. For simplicity we assumed that the coordinates $i_1, i_2, \ldots, i_t$ (as in the circuit definition) are all ℓ.

Now according to the definition of a circuit there exists $Q(t)$. According to the function NM-DFS at some point the NM-DFS starts with α and it moves along an almost-cycle $Q(t)$ and changes the value of $f(\lambda_t)$ from a to c. As it moves along $Q(t)$, once it reaches β the NM-DFS algorithm changes the value of $f(\lambda_t)$ from c to d_1 and also changes the value of $f(\alpha)$ from c to d_1, once NM-DFS reaches λ_1 on $Q(t)$, it changes the value of $f(\beta)$ from d_1 to d_2 and the value of $f(\alpha)$ from d_1 to d_2. It continues, until it reaches λ_{t-1}, and at this point the value of all $f(\lambda_i)$, $1 \leq i \leq t$ is set to d_t as well as the value of $f(\alpha), f(\beta)$. Now there does not exists a path from α in $G^r_f(\alpha,\ell)$ to some vertex $\alpha' = (x; b^k|_{j\leftarrow d_t})$ with $f(\alpha') \neq d_t$. This is because of the following observations :

Observation: There is no directed path (ℓ, j)-path R in $G^r_f(\psi,\ell)$, $\psi = (x; c^k|_{\ell\leftarrow d})$ with $f(\psi) \neq d$ to some vertex $\phi = (x; c'^k|_{j\leftarrow d'})$ with $f(\phi) = d'$. To see that suppose such a path R exists. Consider the last arc of R say $e = (y; d_1, d_2, \ldots, d_k)(x; c'^k|_{j\leftarrow d'})$ where we may assume $yx \in A(G)$. Since e is an arc of $G^r_f(\psi,\ell)$ we have $f(y; d_1, d_2, \ldots, d_k)d' \notin A(H)$ and $f(y; d_1, d_2, \ldots, d_k)f(x; c'^k|_{j\leftarrow d'}) \in A(H)$, a contradiction.

Observation: Since C is a source we do not need to consider circuits in other components of $G_{x,b}$.

Now w becomes a source, since $f(w) = d_t$. NM-DFS has changed $f(x; a_1, a_2, \ldots, a_k)$ from a to d_t and it does not change it anymore. NM-DFS starts the usual procedure as if there is no circuit Q. Moreover since C is a sink component there is no path from C to another invertible component.

Note that the NM-DFS goes through the vertices of $\lambda_1, \lambda_2, \ldots, \lambda_{t-1}$ and it continues the usual NM-DFS from λ_{t-1} as it requires to change the value of some $f(u; e'_1, e'_2, \ldots, e'_k)$ where $\lambda_{t-1}(u; e_1, a_2, \ldots, e_k)$ is an arc in $G^r_f(w,\ell)$ if $\lambda_t e_\ell \notin A(H)$ (or $e_\ell \lambda_t \notin A(H)$ if $ux \in A(G)$). Note that we have changed the value of $f(\lambda_{t-1})$ to λ_t. Now $f(u, e'_1, e'_2, \ldots, e'_k)$ is changed and is set to some out-neighbor of λ_t (in-neighbor of λ_t) and NM-DFS continues from there. In the next step it returns back and it starts from λ_{t-2} and so on.

Closing remark : We run the function NM-DFS$(x; b^k|_{\ell\leftarrow a},\ell)$ starting at coordinate ℓ. Once we change the value of $f(x; a_1, a_2, \ldots, a_k)$ from a to c (or d_t when C is invertible) then potentially we need to modify the value for $f(y; b_1, b_2, \ldots, b_k)$ from an out-neighbor of a, say a' in $L(y)$ to an out-neighbor of c and this procedure is the same when we start with NM-DFS$(x; b^k|_{i\leftarrow a},\, i)$. $\diamond$

3.2.2 Proof of Lemma 3.4

Suppose g is a homomorphism from G to H with $g(x) = a$. We may assume that there exists a $(x; e_1, e_2, \ldots, a_k)$ with $f(x; e_1, e_2, \ldots, a_k) = a$. Otherwise in some previous step we have shown that there exists a homomorphism that does not map x to a if there exists one that maps x to a. Let $w = (x; b^k|\ell \leftarrow a)$ and let $c = f(w)$.

Let G'_f be the induced sub-digraph of $G^r_f(w,\ell)$ with vertices $(y, a_1, a_2, \ldots, a_k)$ such that :

- $(y; a_1, a_2, \ldots, a_k)$ is reachable from w in $G^r_f(w,\ell)$,
- $f(w_1) = a_k$, for $w_1 = (y; a'_1, a'_2, \ldots, a'_k)$ where $f(w_1)$ was initially $g(y)$,
  $\triangleright$ for simplicity we consider the k-th coordinate, depending on the path from w to

$(y; a_1, a_2, \ldots, a_k)$, $f(w_1)$ could be some a_j.

- a_k is the value before the last time $f(w_1)$ is going be to changed.
  $\triangleright$ the value of $f(w_1)$ is going to be changed to $f(y; a_1, a_2, \ldots, a_k)$ according to NM-DFS for the last time.

Let G' be the induced sub-digraph of G with the vertices $y \in V(G)$ where $(y; a_1, a_2, \ldots, a_k) \in G'_f$.

Observation : Consider a path W in G'_f from $(x; b^k|_{\ell \leftarrow a})$ to $(x; b^k|_{j \leftarrow a})$. Now let X be a walk in G from x to x corresponding to W, i.e. W follows X. We note that by replacing a_k with $f(y; a_1, a_2, \ldots, a_k)$ for every $y \in X$ where $(y; a_1, a_2, \ldots, a_k)$ is on path W, we get a mapping from X to H and under this mapping the beginning of X and end of X are mapped to same vertex c. This is because of the property of weak k-nuf. Moreover this mapping is a homomorphism from X to H because f is a homomorphism consistent with L.

Let Y be a path in G' from x to y. Corresponding to Y there is a path in G'_f from $(x; b^k|_{\ell \leftarrow a})$ to some $(y; a_1, a_2, \ldots, a_k)$. Now define $h(y) = f(y; a_1, a_2, \ldots, a_k)$. For every $y \in V(G) \setminus V(G')$ set $h(y) = g^*(y)$ (here $g^*(y)$ is the last value of $f(y; a'_1, a'_2, \ldots, a'_k)$ where initially was $g(y)$). Note that $h(x) = c$.

We need to show that h is a homomorphism from G to H. Let zz' $(z'z)$ be an arc of G where $z \in V(G')$. We only need to consider the case $z \in V(G')$. By definition there exists a path W from w to some $w' = (z; b_1, b_2, \ldots, b_k) \in G'_f$ with $f(w') = b_{k+1}$. Note that if there does not exists such path W, then by definition $z \notin V(G')$.

For simplicity we consider coordinate k, there exists a path B from a to b_k where W follows B.

Since zz' is an arc of G and g is a homomorphism from G to H, $g(z)g(z') \in A(H)$. Moreover, according to the NM-DFS $b_k d_k \in A(H)$ (here d_k is the value in which some vertex $w'_1 = (z', d'_1, d'_2, \ldots, d'_k)$ is going to change from $g(z')$ after visiting the arc zz' by the NM-DFS). Note that d_k is in $L(z')$. This is because either $d_k = g(z')$ and hence it passes the PreProcessing test or it is $f(z', ...)$ and hence by Lemma 3.1 $d_k \in L(z')$. Now if $b_{k+1} d_k$ is an arc of H then we are done. Otherwise according to the NM-DFS there exist vertices $d_1, d_2, \ldots, d_k \in L(z')$ such that $b_i d_i \in A(H)$, $1 \leq i \leq k$ and $w'w''$, $w'' = (z', d_1, d_2, \ldots, d_k)$, is an arc of G'_f. Since this is the last time we change the value of some $f(w'_1)$ from $g(z)$ in the NM-DFS algorithm, we have $w'' \in G'_f$ and hence $z' \in V(G')$. This would imply that $h(z') = f(w'')$. Since f is a homomorphism consistent with L, we have $f(w')f(w'') \in A(H)$ and because we set $h(z) = f(w')$ and $h(z') = f(w'')$ we have $h(z)h(z') \in A(H)$. Note that in the special case when $z' = x$ we have $g(z') = a$ and by the Observation above we have $h(z') = c$. Therefore h is a homomorphism from G to H with $h(x) = c$.

The argument is the same when the strong component $G_{x,b}(\ell, a)$ does not have a source. $\diamond$

4 Additional Algorithms and Proofs (New Minority Algorithm)

This section collects additional statements and proofs mainly related to new Minority algorithm. We develops a direct algorithm to handle the minority case in our main algorithm. Of course, it would be easier to appeal to Bulatov-Dalmau Maltsev result as explained in Subsection 2.4.

Let M^3 be the digraph of triple (y, c, d), $x \in G, c, d \in L(y)$. There is an arc from (y, c, d) to (y', c', d') where $yy' \in A(G)$, $cc', dd' \in A(H)$ and $cd' \notin A(H)$ or $y'y \in A(G)$ and $c'c, d'd \in A(H)$, $d'c \notin A(H)$. Note that M^3 is a graph but we view M^3 as digraph and when we talk about a path in M^3 we mean an oriented path that reflect the direction of the edge xy when $(y, c, d)(y', c', d')$ is an arc of M^3. Let $G_x(a, b)$ be the strong component of M^3 containing (x, a, b) such that for each vertex $(y, c, d) \in G_x(a, b)$, $(a, c) \in L(x, y)$ and $(b, d) \in L(x, y)$. We say a strong component $G_x(a, b)$ of M^3 is *invertible* if both (x, a, b) and (x, b, a) are in $G_x(a, b)$.

Again the idea is similar to the one handling the RemoveNotMinority case. At each step we consider a vertex x of G and two vertices $a, b \in L(x)$ and try to eliminate one of the a, b from $L(x)$. To decide whether remove a or b we construct an instance of the problem say (G', H, L') and solve this instance recursively. Based on the existence of a L'-list homomorphism from G' to H we decide to remove a or b. Depending on $G_x(a, b)$ being invertible or not two different instances for the sub-problem are constructed. At the end we have singleton lists and if there is a homomorphism from G to H with the singleton lists then success otherwise we report there is no homomorphism from G to H. We denote the underline graph of digraph G by $UN(G)$. If $G_x(a, b)$ is invertible we construct a new instance G', H, L' and to construct the lists L' we use the Maltsev property.

Lemma 4.1 *Let X be an oriented path in G and let B, C, D be three walks in $L(X)$ all congruent to X where B is from a to c and C is from b to c and D is from b to D. Then there exists a walk E from a to d in $L(X)$ that is congruent with X.*

Proof: By following B, C, D on the vertices in X and applying the definition of polymorphism h, we conclude that E exists. $\diamond$

Lemma 4.1 implies the following corollary.

Corollary 4.2 *If $(a, c) \in L(x, y)$ and $(b, c), (b, d) \in L(x, y)$ then $(a, d) \in L(x, y)$.*

Lemma 4.3 *The Algorithm 4 runs in polynomial time. Moreover, if there is a homomorphism g from G to H with $g(x) \in \{a, b\}$ then there is a homomorphism from G to H after removing a or b from $L(x)$ according to Maltsev-(G, H, L, h).*

Proof: First consider the case that $G_x(a, b)$ is not invertible. The Algorithm 4 is recursive but at each recursive call the size of the input decreases by at least one. This

Algorithm 4 RemoveMinority – Using Matlsev Operations

function REMOVEMINORITY(G, H, L)
  Input: x, a, b such that $f(x; b^k|_{i \leftarrow a}) = a$
  Define Maltsev consistent homomorphism $h : G \times H^3 \to H$ where $h(x; a, b, c) = f(x; a, b, b, \ldots, b, c)$ for $a, b, c \in L(x)$.
  while $\exists\, x \in V(G)$ with $|L(x)| \geq 2$ **do**
    MATLSEV$(G, H, L\ )$
  if $\exists$ a list homomorphism from G to H **then** return True
  else return False.

function MALTSEV(G, H, L)
  Let $a, b \in L(x)$ be two distinct vertices. Construct $G_x(a, b)$.

  if $G_x(a, b)$ is not invertible **then**
    Set $L'(x) = a$ and $L'(y) = \emptyset$ for every $x \neq y$.
    Set $G'' = \emptyset$, and let G' be an induced sub-digraph of G constructed as below.
    for all $y \in G$ s.t. $\exists$ a path from (x, a, b) to (y, c, d) in $G_x(a, b)$ **do**
      add y to G' and c to $L'(y)$,
      let yy' an arc of G' when $(y, c, d)(y', c', d') \in G_x(a, b)$.

  if $G_x(a, b)$ is invertible **then**
    Set $L'(x) = a$ and $L'(y) = \emptyset$ for every $x \neq y$.
    Let G'' be the set of vertices $y \in V(G)$ where (y, c, d, e) lies on an oriented path P from (x, a, b, b) to (x, b, b, a) in $UN(G \times H^3)$ s.t. no intermediate vertex in P is (x, a', b', c') for $a', b', c' \in L(x)$
    for all arc $(y, c, d, e)(y', c', d', e') \in P$ **do**
      add $h(y, c, d, e)$ to $L'(y)$, $h(y', c', d', e')$ to $L'(y')$, and add arc yy' $(y'y)$ to G''
    Let G' be the induced sub-digraph of G with vertices
    $y \in G \setminus G''$ and arcs yy' where $(y, c, d)(y', c', d') \in G_x(a, b)$ and add c into $L'(y)$ when $(y, c, d) \in G_x(a, b)$

  if Matlsev$(G' \cup G'', H, L')$ **then**
    remove b from $L(x)$
  else remove a from $L(x)$.

is because we do not add b into $L'(x)$ and hence we have an instance (G', L', H) of the problem in which at least one vertex has a smaller size list. We also note that once we make a decision to remove a vertex from a list the decision is not changed. Therefore the overall procedure in this case is polynomial (assuming in each call $G_x(a, b)$ is not invertible). According to Maltsev-(G, H, L, h) if G' does not admit a homomorphism to H then there is no homomorphism from G to H that maps x to a since G' is a sub-digraph of G.

No suppose b is removed according to Maltsev-(G, H, L, h) and there exists a homomorphism g from G to H with $g(x) = b$. Let ψ be the homomorphism from G' to H. Note that $\psi(x) = a$.

Define G_1 be a sub-digraph of G' consists of the vertices y such that $(y, \psi(y), g(y))$ is reachable from (x, a, b) in $G_x(a, b)$.

Now for every vertex $y \in G_1$ set $\phi(x) = \psi(x)$ and for every $y \in G \setminus G_1$ set $\phi(y) = g(y)$. Now let zz' be an arc of G. If none of the $z, z' \in G_1$ then clearly since g is a homomorphism, $\phi(z)\phi(z') \in A(H)$. If both $z, z' \in G_1$ then again since ψ is a homomorphism, we have $\phi(z)\phi(z') \in A(H)$. Suppose $z \in G_1$ and $z' \notin G_1$. Since $z \in G_1$ there exists a path in $G_x(a, b)$ from (x, a, b) to $(z, \psi(z), g(z))$. Now if $\psi(z)g(z')$ is an arc then we are done. Otherwise $(z', c, g(z')) \in G_x(a, b)$ where $c \in L(z')$ and $\psi(z)c \in A(H)$ (note that since $\psi(z)$ is in $L(z)$ it must have an out-neighbor in $L(z')$). This would mean $z' \in G_1$, a contradiction. Note that when $z' = x$ and $z \in G_1$ then because $(\psi(z), a) \in L(z, x)$ we have $\psi(z)a \in A(H)$.

Second consider the case that $G_x(a, b)$ is invertible. Again as we argued in the previous case the algorithm is recursive but at each recursive call the size of the input decreases and once a decision made (removing a or b) it won't change.

Observe that if there exists a homomorphism g that maps x to a then for every closed walk X from x to x, the image of $g(X)$ is a closed walk from a to a in H. Since there is a walk BB from b to b in $L(X)$ congruent and since $(a, b) \in L(x, x)$, there is a walk BA in $L(X)$ congruent to X. Now by Lemma 4.1 there is a walk AB from a to b in $L(X)$ and congruent with X. These would imply that there exists a path P in $U(G \times H^3)$ from (x, b, b, a) to (x, a, b, b). Now according to the definition of G'' for every walk BB in $L(X)$ from b to b congruent with X we keep a walk AA in $L'(X)$ congruent with X and is obtained by adding $h(y, c_1, c_2, c_3)$ into $L'(y)$ where (y, c_1, c_2, c_3) is in P.

This would mean that in the list of $L'(X)$, corresponding to $\psi(X)$ (ψ is a homomorphism from G to H where $\psi(x) = a$ if there exists one) we have a path from a to a in $L'(X)$. Therefore if there exists no homomorphism from $G' \cup G'' \to H$ that maps x to a then there is no homomorphism from G to H that maps x to a.

Now suppose b is removed according to Maltsev-(G, H, L, h) and there exists a homomorphism g from G to H with $g(x) = b$. Let ψ_1 be the homomorphism from G' to H and ψ_2 be a homomorphism from G'' to H. Note that $\psi_1(x) = \psi_2(x) = a$. First we show that there is no arc e (forward or backward) from a vertex y in G' to a vertex in G''. If this is the case then there is a walk Q from x to y and there is a walk Q' from y to x. Now since yz is an arc then $Qee^{-1}Q'$ is a closed walk from x to x and hence we should have added z into G''.

Note that G'' consists of all the induced oriented cycles including x and also all the paths reaching out of vertices of these cycles except x.

Define G_1 be a sub-digraph of G' consists of vertices y such that $(y, \psi_1(y), g(y))$ is reachable from (x, a, b) in $G_x(a, b)$. Now for every vertex $y \in V(G_1)$ set $\phi_1(y) = \psi(y)$ and for every $y \in V(G) \setminus V(G_1)$ set $\phi_1(y) = g(y)$. Define G_2 be a sub-digraph of G'' consisting of vertices y such that $(y, \psi_2(y), g(y))$ is reachable from (x, a, b) in $G_x(a, b)$.

Now for every vertex $y \in V(G_2)$ set $\phi_2(y) = \psi_2(y)$ and for every $y \in V(G) \setminus V(G_2)$ set $\phi_2(y) = g(y)$. And finally let $\phi(y) = \phi_1(y)$ when $y \in V(G')$ and $\phi(y) = \phi_2(y)$ when $y \in V(G'')$. Now let zz' be an arc of G. We need to verify that $\phi(z)\phi(z')$ is an arc of H.

If none of the $z, z' \in V(G_1 \cup G_2)$ then clearly since g is a homomorphism, $\phi(z)\phi(z') \in A(H)$. If both $z, z' \in V(G_i)$ $(i = 1, 2)$ then again since ψ_i is a homomorphism, we have $\phi(z)\phi(z') \in A(H)$. Suppose $z \in V(G_2)$ and $z' \notin V(G_2)$. Since $z \in V(G_2)$, there exists a closed walk ZZ' that contains z where Z is a path from x to z and Z' is a walk from z to x. Note that $\psi_2(ZZ')$ is a walk from a to a in $L(ZZ')$ and $g(ZZ')$ is a walk from b to b in $L(ZZ')$ and since $(b, a) \in L(x \times x)$ there exists a walk BA in $L(ZZ')$ congruent with ZZ'. These would imply that corresponding to ZZ' there exists a path P from (x, b, b, a) to (x, a, b, b) containing vertex $(z, \psi_2(z), d, g(z))$. Now since $g(z) \in L(z)$ there exists a vertex $c \in L(z')$ such that $g(z)c \in A(H)$ and there exists $e' \in L(z')$ such that $dd' \in A(H)$.

Now we add the two arcs $(z, \psi_2(z), d, g(z))(z', c, d', g(z')), (z', c, d', g(z'))(z, \psi_2(z), d, g(z))$ into P and hence we obtain a path P' from (x, b, b, a) to (x, a, b, b) that goes through vertex $(z', c, d', g(z'))$ and hence $z' \in G''$. Now since $\phi_2(z)g(z') \notin A(H)$, by definition $z' \in V(G_2)$, a contradiction.

As we argue before one can show that ϕ_1 is also a homomorphism from G to H. Therefore ϕ is a homomorphism from G to H with $\phi(x) = a$. $\diamond$

Acknowledgements : We would like to thank Pavol Hell for so many helpful discussions and his enormous support. We would also like to thank Geoffrey Exoo, László Egri, Víctor Dalmau, and Ross Willard for several useful comments.

References

[ABISV09] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: refining schaefer's theorem. *Journal of Computer and System Sciences*, 75(4): 245–254 (2009).

[BHM88] J. Bang-Jensen, P. Hell, G. MacGillivray. The complexity of colouring by semicomplete digraphs. *SIAM J. Discrete Math.*, 1 : 281–298 (1988).

[BH90] J. Bang-Jensen, P. Hell. The effect of two cyles on the complexity of colourings by directed graphs. *Discrete Appl. Math.*, 26 : 1–23 (1990).

[BKN09] L. Barto, Marcin Kozik, and Todd Niven. The CSP Dichotomy Holds for Digraphs with No Sources and No Sinks (A Positive Answer to a Conjecture of Bang-Jensen and Hell). *SIAM J. Comput*, 38(5) : 1782–1802 (2009).

[B02b] A.A. Bulatov. A dichotomy constraint on a three-element set. In *Proceedings of STOC* 649–658 (2002).

[B05] A.Bulatov. H-Coloring dichotomy revisited. *Theoret. Comp. Sci.*, 349 (1) : 31-39 (2005).

[B06] A. Bulatov. A dichotomy theorem for constraints on a three-element set. *Journal of the ACM*, 53(1): 66–120 (2006).

[B11] A.Bulatov. Complexity of conservative constraint satisfaction problems. *Journal of ACM Trans. Comput. Logic*, 12(4) : 24–66 (2011).

[BD06] A. Bulatov and V. Dalmau. A Simple Algorithm for Mal'tsev Constraints. *SIAM J. Comput.*, 36(1): 16–27 (2006).

[BJK05] A.A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM journal on computing*, 34(3): 720-742 (2005).

[CCL13] J.Y. Cai, X. Chen, P. Lu. Graph Homomorphisms with Complex Values: A Dichotomy Theorem. *SIAM J. Comput.*, 42(3): 924–1029 (2013).

[CVK10] C. Carvalho, V.Dalmau, and A.A. Krokhin. CSP duality and trees of bounded pathwidth. *Theor. Comput. Sci.*, 411(34–36): 3188–3208 (2010).

[CKS01] N. Creignou, S. Khanna, and M. Sudan. Complexity Classifications of Boolean Constraint Satisfaction Problems. *SIAM Monographs on Discrete Math. and Applications*, vol. 7 (2001).

[CL14] P. Csikvári and Z. Lin. Graph homomorphisms between trees. *Elec. J. Combin.*, 21 : 4–9 (2014).

[D92] R. Dechter. Containt networks. *Encyclopedia of Artificial Intelligence* 276–285 (1992).

[D00] V. Dalmau. A new tractable class of constraint satisfaction problems. In *Proceedings 6th International Symposium on Artificial Intelligence and Mathematics*, 2000.

[DF03] V. Dalmau, D. Ford. Generalized satisfiability with k occurrences per variable: A study through delta-matroid parity. In *Proceedings of MFCS 2003, Lecture Notes in Computer Science*, 2747 : 358–367 (2003).

[EHLR14] L.Egri, P.Hell, B.Larose, and A.Rafiey. Space complexity of List H-coloring : a dichotomy. In *Proceedings of SODA*, (2014).

[F01] T. Feder. Homomorphisms to oriented cycles and k-partite satisfiability. *SIAM J. Discrete Math.*, 14 : 471–480 (2001).

[F06] T. Feder A dichotomy theorem on fixed points of several nonexpansive mappings. *SIAM J. Discrete Math.*, 20 : 291–301 (2006).

[FHH99] T.Feder, P.Hell, J.Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42 : 61–80 (1999).

[FH98] T. Feder, P. Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory Ser.*, B 72 : 236–250 (1998).

[FHH99] T. Feder, P. Hell, J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19 : 487–505 (1999).

[FHH03] T. Feder, P. Hell, J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42 : 61–80 (2003).

[FHH07] T. Feder, P. Hell, J. Huang. List homomorphisms of graphs with bounded degrees. *Discrete Math.*, 307 : 386–392 (2007).

[FMS04] T. Feder, F. Madelaine, I.A. Stewart. Dichotomies for classes of homomorphism problems involving unary functions. *Theoret. Comput. Sci.*, 314 : 1–43 (2004).

[FV93] T.Feder and M.Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of STOC*, 612–622 (1993).

[FV98] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1): 57–104 (1998).

[HN90] P. Hell and J. Nešetřil. On the complexity of H-colouring. *J. Combin. Theory B*, 48 : 92–110 (1990).

[HN04] P. Hell, J. Nešetřil. *Graphs and Homomorphisms*, Oxford University Press, 2004.

[HN08] P.Hell, J.Nešetřil. Colouring, Constraint Satisfaction, and Complexity. *Computer Science Review* , 2(3): 143–163 (2008).

[HR11] P.Hell and A.Rafiey. The Dichotomy of List Homomorphisms for Digraphs. In *Proceedings of SODA* 1703–1713 (2011).

[HR12] P. Hell and A. Rafiey. The Dichotomy of Minimum Cost Homomorphism Problems for Digraphs. *SIAM J. Discrete Math.*, 26(4): 1597–1608 (2012).

[KSDR96] L.G. Kroon, A. Sen, H. Deng, A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. In *Graph-Theoretic Concepts in Computer Science (Cadenabbia, 1996), Lecture Notes in Computer Science*, 1197 : 279–292 (1997).

[K92] V. Kumar. Algorithms for constraint-satisfaction problems. *AI Magazine*, 13 :32–44 (1992).

[J98] P. Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theor. Comput. Sci.*, 200(1-2): 185-204 (1998).

[L75] R. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of the ACM (JACM)*, 22(1): 155–171 (1975).

[LZ03] B. Larose, L. Zádori. The complexity of the extendibility problem for finite posets. *SIAM J. Discrete Math.*, 17 : 114–121 (2003).

[MM08] M. Maroti, and R. McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59 : 463–489 (2008).

[S78] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of STOC*, 216–226 (1978).

[S10] M.Sigge. A new proof of the H-coloring dichotomy. *SIAM J. Discrete Math.*, 23 (4) : 2204–2210 (2010).

[V00] M.Y. Vardi. Constraint satisfaction and database theory: a tutorial. *Proceedings of the 19th Symposium on Principles of Database Systems (PODS)*, 76–85 (2000).