Descriptive complexity of list H-coloring problems in logspace: a refined dichotomy

Victor Dalmau*, László Egri†, Pavol Hell‡, Benoît Larose§ and Arash Rafiey¶
*Universitat Pompeu Fabra, Spain
Email: victor.dalmau@upf.edu
†Concordia University, Canada
Email: laszlo.egri@mail.mcgill.ca
‡Simon Fraser University, Canada
Email: pavol@sfu.ca
§Concordia University, Canada
Email: benoit.larose@concordia.ca
¶Indiana State University
Email: arash.rafiey@indstate.edu

***Abstract*—The Dichotomy Conjecture for constraint satisfaction problems (CSPs) states that every CSP is in P or is NP-complete (Feder-Vardi, 1993). It has been verified for conservative problems (also known as list homomorphism problems) by A. Bulatov (2003). Egri et al. (SODA 2014) augmented this result by showing that for digraph templates H, every conservative CSP, denoted LHOM(H), is solvable in logspace or is hard for NL. A conjecture of Larose and Tesson from 2007 forecasts that when LHOM(H) is in logspace, then in fact, it falls in a small subclass of logspace, the set of problems expressible in symmetric Datalog. The present work verifies the conjecture for LHOM(H) (and, indeed, for the wider class of conservative CSPs with binary constraints), and by so doing sharpens the aforementioned dichotomy. A combinatorial characterization of symmetric Datalog provides the language in which the algorithmic ideas of the paper, quite different from the ones in Egri et al., are formalized.**

I. Introduction

A. Fixed template CSPs

In the mid 1990's, in a pair of seminal papers, P. Jeavons [1], and T. Feder and M. Vardi [2], [3] laid the foundation for a program that has oriented much of the theoretical work on the complexity of constraint satisfaction problems (CSP) to this day. Inspired by the works cited above, a series of precise conjectures relating the computational complexity of CSPs to their expressibility in various logics and to the closure properties of constraints under operations was proposed [4], [5] [6]. By importing powerful techniques from universal algebra to the field of CSP, these works shed new light on computational and descriptive complexity issues, and in general, led to a deeper understanding of fixed-template CSPs. Since several natural problems complete for standard complexity classes such as P, NL and L are easily phrased as CSPs, the above results are of general interest in the theory of computation.

A constraint satisfaction problem consists of a finite set of variables, a finite set of constraints on these, and the problem is to determine whether there exists an assignment of values to the variables that satisfies all constraints. Although the general problem is NP-complete, restricting the available constraints may lead to a tractable problem; thus one can parametrise the CSP by a constraint language (also called template), and ask for which languages the CSP is tractable, or solvable with space restrictions, expressible in various logics, and so on. An often convenient way of reformulating this problem is through relational structures and homomorphisms: given a fixed structure $\mathbf{H}$ (essentially equivalent to a constraint language), $\mathrm{CSP}(\mathbf{H})$ is the problem of deciding, given a structure $\mathbf{A}$ of the same signature as $\mathbf{H}$, whether there exists a homomorphism from $\mathbf{A}$ to $\mathbf{H}$ or not. Equivalently, one may view $\mathrm{CSP}(\mathbf{H})$ as the problem of determining if a given primitive positive formula is satisfiable in $\mathbf{H}$. Jeavons [1] was the first to notice that, in Schaefer's generalised satisfiability result classifying all CSPs on 2-element templates as tractable or NP-complete [7], the template associated to tractable CSPs had a "nice" (in some precise technical sense) underlying algebraic structure. Building on this, Bulatov, Jeavons and Krokhin [4] extended Feder and Vardi's dichotomy conjecture by proposing a precise algebraic criterion to separate the tractable and hard cases: the existence of operations preserving the constraints and obeying certain "nice" identities should guarantee tractability.

B. Expressibility of fixed-template CSPs in various logics

Descriptive complexity aims to link the computational complexity of a problem with its definability in various logics. For example, it is known that any first-order definable class of structures belongs to the complexity class AC^0 (see [8]). Therefore showing that a CSP is definable with an FO formula also establishes that the CSP is in AC^0. Defining CSPs using certain logics is not merely a tool to show membership in various complexity classes, but it also deepens our understanding of these problems. For example, studying the class of CSPs definable in FO has led to many elegant characterisations of this class (see [9], [10], [11], [12]), one of them being that these are precisely the CSPs which have *finite duality*.

Datalog is a logic programming language introduced in the 70s. To this day, it has had countless applications in computer science, e.g. in data integration, networking, cloud computing, etc. Roughly speaking, a Datalog program takes as input finitely many finitary relations over a fixed universe (so-called EDBs), and defines new relations (IDBs) over the same universe using primitive positive rules and closure under fixed points. (One may view a Datalog program as defining a Boolean query if some chosen 0-ary IDB is empty or not.) For any Datalog program, it can be decided in polynomial time whether it accepts or rejects a given input structure. In their pioneering work ([2], [3]), Feder and Vardi showed that not all tractable CSPs are expressible in Datalog (e.g. systems of linear equations on a finite field), and therefore asked the question which CSPs are. Already implicit in the same paper, there is a conjecture relating properties of the template $\mathbf{H}$ and expressibility of the CSP in Datalog (see [13] for details). Following Bulatov, Jeavons and Krokhin's, and Feder and Vardi's lead, Larose and Zádori [5] conjectured a precise algebraic criterion characterizing CSP's expressible in Datalog. Confirming the deep connection between the expressibility of the CSP in Datalog and the structure of its underlying algebra, the conjecture was proved by Barto and Kozik in 2009 [14]. In fact, if this criterion is not met, then by a result of Atserias, Bulatov and Dawar [15], the CSP is not even definable in infinitary logics with counting.[1]

Two fragments of Datalog are particularly important when studying the complexity of CSPs: by restricting the presence of auxiliary relations (IDBs) to a single occurrence in the body of each rule, one obtains linear Datalog; this restriction allows us to evaluate such a program in non-deterministic logspace. If one further requires that the program be closed under (purely syntactic) *symmetry of rules*, one obtains so-called symmetric Datalog [17]. A consequence of this symmetry is that these programs can be evaluated in deterministic logspace.[2] Larose and Tesson [6] conjectured precise identities, similar in flavour to those describing Datalog, that capture CSPs definable in linear (resp. symmetric) Datalog; if these conjectures hold they would present a complete picture of the Datalog hierarchy for CSPs. Furthermore they conjectured that all CSPs solvable in NL (L) are expressible in linear (resp. symmetric) Datalog (under standard complexity theoretic assumptions, such as $P \neq NL$). Larose and Tesson confirmed all the above conjectures in the 2-element case, and various other special cases have also been verified [19], [20]. It should be noted that a result of Bulin et al [21] reduces all the above conjectures to the case where the template of the CSP is a digraph. [3]

[1]It turns out that the class of CSPs definable in Datalog is quite robust, and coincides with the class of CSPs solvable by poly-size monotone circuits [13], and also those admitting a robust scheme of approximation [16].

[2]Using Reingold's breakthrough result that undirected reachability is in logspace [18].

[3]A. Kazda has recently announced a proof of the symmetric Datalog conjecture under the hypothesis that the CSP is solvable in linear Datalog, effectively reducing the symmetric Datalog conjecture to the linear one.

C. List-Homomorphism Problems

The list-homomorphism problem is a natural restriction of the CSP, where each variable is constrained to take its values in a prescribed list of values; these problems have been widely studied in the context of graphs as *list* $\mathbf{H}$*-colouring problems* (see for instance [22], [23], [24] and also [25], [26]). Given a target structure $\mathbf{H}$, the list-homomorphism problem for $\mathbf{H}$ takes as input a structure $\mathbf{A}$ together with a list $L(a) \subseteq H$ for each $a \in A$; one must decide if there exists a homomorphism f from $\mathbf{A}$ to $\mathbf{H}$ that respects every list, i.e. such that $f(a) \in L(a)$ for all $a \in A$. These so-called conservative CSPs have attracted a great deal of attention as a good testing ground for the various conjectures: the algebraic dichotomy conjecture is known to hold here [27] but, with the exception of graphs [20], the refined conjectures of [6] are still open. When the underlying structure is a digraph, partial results are known: Hell and Rafiey [24] showed that the tractable cases of the list-homomorphism problem on digraph targets are all solvable in Datalog, a result generalised by Kazda to all structures with at most binary relations [28].

D. Summary of results and organisation of the paper

Datalog, linear Datalog, and symmetric Datalog have been extensively studied (see, e.g. [29], [30], [33]) and each can be described in terms of dualities (bounded tree-, path-, and symmetric pathwidth duality, respectively). Furthermore, elegant and useful pebble game descriptions (see [31]) are available for Datalog and linear Datalog. Unfortunately, no such pebble game is available for symmetric Datalog. In the present paper, we introduce a simple and natural combinatorial characterisation of symmetric Datalog that serves as the language in which our proofs are formalized. We work with the notion of k-canonical symmetric Datalog program (see section V); it is known that if $\mathrm{CSP}(\mathbf{H})$ is decided by some symmetric Datalog program, then it is in fact decided by the k-canonical symmetric Datalog program for some k. Roughly speaking, a k*-walk* in a structure $\mathbf{A}$ is a finite sequence of at most k-element substructures ("bags") of $\mathbf{A}$. Imagine zigzagging back and forth along such a walk from its first bag to the last, obtaining a new walk; we call such a walk a zigzag-expansion. A *realisation* of such a zigzag-expansion Z is a sequence of homomorphisms, one from each bag of Z to $\mathbf{H}$, such that homomorphisms for consecutive bags coincide on the intersection of the bags. We argue that the k-canonical symmetric Datalog program derives the goal predicate on $\mathbf{A}$ if and only if there exists a k-walk on $\mathbf{A}$ such that no zigzag expansion of this walk is realisable in $\mathbf{H}$ (Lemma 2).

We believe that the simplicity and flexibility of this characterisation makes it a powerful tool to show that a given CSP is definable in symmetric Datalog. To illustrate this we proceed to apply our result to prove that the symmetric Datalog conjecture holds for list homomorphism problems on templates that have at most binary relations (Theorem 1). Since membership of a list homomorphism problem in symmetric Datalog is a much stronger statement than membership of that list homomorphism problem in logspace, our results improve

and sharpen the dichotomy proved in [32], stating that the list homomorphism for digraph templates is either in L or hard for NL. Furthermore, our results generalize [32] from digraphs to relational structures that have at most binary relations. As a consequence, we obtain equivalent characterisations of these CSPs in logical, algebraic and graph-theoretic terms (for digraphs).

Here is the overall structure of the paper: sections II and III contain preliminary results and terminology, and section IV presents the basics on Datalog and its fragments. Section V is devoted to our combinatorial characterisation of CSPs describable in symmetric Datalog, and section VI applies our main result to list homomorphism problems on templates with at most binary relations. To avoid disrupting the flow of our presentation, some technical considerations have been relagated to an appendix (section VII).

II. Basic definitions

Let A be a finite set. A k-ary *tuple* $(a_1, \ldots, a_k)$ over A is any element of A^k. We shall usually use boldface letters to denote tuples of any length. If $\mathbf{a} = (a_1, \ldots, a_k)$ is a tuple and f is a mapping whose domain contains $\{a_1, \ldots, a_k\}$ we write $f(\mathbf{a})$ to indicate the tuple, $(f(a_1), \ldots, f(a_k))$, obtained by applying f to $\mathbf{a}$ component-wise. A k-ary *relation*, R, on A is a collection of k-ary tuples over A or, alternatively, a subset of A^k. For every $i_1, \ldots, i_n \in \{1, \ldots, k\}$, we denote by $\mathrm{pr}_{i_1,\ldots,i_n} R$ the relation

$$\{(a_{i_1}, \ldots, a_{i_n}) \mid (a_1, \ldots, a_k) \in R\}$$

If R is a collection of tuples of not necessarily the same size, we shall write $\mathrm{extremes}(R)$ to denote the relation

$$\{(a_1, a_k) \mid (a_1, \ldots, a_k) \in R\}$$

A *(relational) signature* τ is a collection of relational symbols (also called predicates), in which every symbol has an associated arity. A *structure* $\mathbf{A}$ with signature τ (also called *τ-structure*) consists of a set A called the *universe* of $\mathbf{A}$, and for each symbol $R \in \tau$, of arity, say, k, a k-ary relation $R^{\mathbf{A}}$ on A, called the *interpretation* of R in $\mathbf{A}$. We shall use the same boldfaced and slanted capital letters to denote a structure and its universe, respectively.

If B is a subset of A, then the *substructure of* $\mathbf{A}$ *induced by* B, denoted $\mathbf{A}_{|B}$, is the structure over signature τ with universe B such that for every predicate R in τ with arity, say, k, $R^{\mathbf{B}} = R^{\mathbf{A}} \cap B^k$.

Let $\mathbf{A}, \mathbf{B}$ be relational structures of the same signature with universes A and B, respectively. A mapping $f : A \to B$ is said to be a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ if for any symbol R from τ, and every $\mathbf{a} \in R^{\mathbf{A}}$, $f(\mathbf{a}) \in R^{\mathbf{B}}$. If, furthermore, f is bijective and f^{-1} is a homomorphism from $\mathbf{B}$ to $\mathbf{A}$, then we say that f is an *isomorphism*. A *partial homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is any homomorphism from an induced substructure of $\mathbf{A}$ to $\mathbf{B}$.

We say that a relation $R \subseteq A^k$ is *pp-definable* from $\mathbf{A}$ if there exists a (primitive positive) formula

$$\psi(x_1, \ldots, x_k) \equiv \exists y_1, \ldots, y_\ell \; \varphi(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$$

where φ is a conjunction of formulas with relations in the signature of $\mathbf{A}$, such that for every $(a_1, \ldots, a_k) \in A^k$

$$(a_1, \ldots, a_k) \in R \text{ if and only if } \psi(a_1, \ldots, a_k) \text{ holds in } \mathbf{A}$$

Let $\mathbf{H}$ be a structure. The *constraint satisfaction problem* with template $\mathbf{H}$, denoted $\mathrm{CSP}(\mathbf{H})$, is the problem of deciding, given a structure $\mathbf{A}$ of the same signature than $\mathbf{H}$, whether there exists a homomorphism from $\mathbf{A}$ to $\mathbf{H}$. We will also call a homomorphism from $\mathbf{A}$ to $\mathbf{H}$ a *satisfying assignment* or a *solution*. In what follows $\mathbf{H}$ will always denote the template of a constraint satisfaction problem.

III. Walks

Let $\mathbf{H}$ be a structure and let $\mathbf{A}$ be an instance of $\mathrm{CSP}(\mathbf{H})$. A *walk* ω on $\mathbf{A}$ is a sequence $\mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_n}, \mathbf{a_n}$ of tuples on A satisfying the following condition: for every $1 \leq i \leq n$, $\mathbf{b_i}$ contains all the elements in $\mathbf{a_{i-1}}$ and $\mathbf{a_i}$ (assume that $\mathbf{a_0}$ is the tuple of length 0). A *subwalk* of ω is any walk of the form $\mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_i}, \mathbf{a_i}$ with $i \leq n$. The length of a walk is the number of its tuples divided by 2 and its width is the maximum arity of its tuples. Note that we allow tuples of length 0.

A *realization* (in $\mathbf{H}$) of a walk $\mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_n}, \mathbf{a_n}$ is a walk $\mathbf{b'_1}, \mathbf{a'_1}, \ldots, \mathbf{b'_n}, \mathbf{a'_n}$ on $\mathbf{H}$ such that for every $1 \leq i \leq n$ there exists a partial homomorphism h_i from $\mathbf{A}$ to $\mathbf{H}$ whose domain contains all elements in $\mathbf{b_i}$ such that $\mathbf{b'_i} = h_i(\mathbf{b_i})$, $\mathbf{a'_i} = h_i(\mathbf{a_i})$, and $\mathbf{a'_{i-1}} = h_i(\mathbf{a_{i-1}})$.

A *zigzag-expansion* of a walk $\mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_n}, \mathbf{a_n}$ is any walk of the form

$$\mathbf{b_{i_1}}, \mathbf{a_{i_1}}, \ldots, \mathbf{b_{i_m}}, \mathbf{a_{i_m}}$$

where:

1) $i_1, \ldots, i_m \in \{1, \ldots, n\}$,
2) $i_1 = 1$,
3) $i_m = n$, and
4) $i_{j+1} \in \{i_j + 1, i_j - 1\}$ for every $j = 1, \ldots, m-1$.

A *zigzag-realization* of a walk ω is any realization of any zigzag-expansion of ω.

We shall denote by $\mathrm{real}_\omega^{\mathbf{A}}$ (resp. $\mathrm{zz\text{-}real}_\omega^{\mathbf{A}}$) the set of all realizations (resp. zigzag-realizations) of ω. We might drop the superscript $\mathbf{A}$ if it is clear from the context.

If R is a collection of realizations (for example $\mathrm{real}_\omega^{\mathbf{A}}$ or $\mathrm{zz\text{-}real}_\omega^{\mathbf{A}}$) then we shall use $\mathrm{last}(R)$ to the denote the collection of all tuples that appear at the end of a realization in R.

A walk $\omega = \mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_n}, \mathbf{a_n}$ is *simple* if there exists $a_1, \ldots, a_n$ such that for every $1 \leq i \leq n$, $\mathbf{a_i} = (a_i)$ and $\mathbf{b_i} = (a_{i-1}, a_i)$ (with the exception of $\mathbf{b_1}$ which is set to (a_1)). In this case we shall use $a_1, \ldots, a_n$ to denote the walk ω. Similarly, we shall denote any realization of ω by a sequence of n elements of H. Note that in this way, a simple walk $a_1, \ldots, a_n$ can be regarded as a n-ary tuple on A and real_ω can be regarded as a subset of H^n.

Let $\omega = a_1, \ldots, a_n$ and $\omega' = b_1, \ldots, b_m$ be simple walks. We shall use ω^{-1} to express the walk $a_n, \ldots, a_1$ obtained by reversing the direction of ω. Furthemore, if $a_n = b_1$ then we shall use $\omega \cdot \omega'$ to express the walk $a_1, \ldots, a_n, b_2, \ldots, b_m$.

IV. Datalog Programs

A *Datalog Program* (for template $\mathbf{H}$) is a finite set of rules of the form:

$$T_0 \leftarrow T_1, T_2, \ldots, T_n$$

where each T_i is an atomic formula. Then, T_0 is called the *head* of the rule and $T_1, T_2, \cdots, T_n$ is called the *body* of the rule. There are two kinds of relational predicates occurring in a program: predicates that occur at least once in the head of a rule are called *intensional database predicates* (IDBs) and are not part of the signature of $\mathbf{H}$. The other predicates are called *extensional predicates* (EDBs) and are part of the signature of $\mathbf{H}$. One special IDB, which is 0-ary, is the *goal* predicate of the program.

Let r be the rule $T_0 \leftarrow T_1, T_2, \ldots, T_n$. Rule r is *linear* if at most one atomic formula in its body has and IDB. Furthermore, if r has no IDB in its body we say that is r is *non-recursive*. The *symmetric complement* of a linear rule r is the rule defined in the following way:

1) If r has an IDB in the body, say in T_1, then the symmetric complement is $T_1 \leftarrow T_0, T_2 \ldots, T_n$.
2) If r has no IDB in the body, then the symmetric complement is again r.

A Datalog program is *linear* if so are all its rules. Furthermore, it is *symmetric* if the symmetric complement of any of its rules also belongs to the program.

The semantics of Datalog programs are usually defined in terms of fix-point operators. For convenience we shall use an equivalent definition based on the inductive notion of *derivation*.

Let P be a Datalog program for $\mathbf{H}$ and let $\mathbf{A}$ be a structure with the same signature as $\mathbf{H}$. A *fact* (for $\mathbf{A}$ and P) is any atomic formula of the form $R(\mathbf{b})$ where R is an IDB and $\mathbf{b}$ is a tuple of variables in A with the same arity as R. We say that Datalog program P *derives* a fact, $R(\mathbf{b})$, (on $\mathbf{A}$) if there exists a rule $R(\mathbf{x}) \leftarrow R_1(\mathbf{x_1}), \ldots, R_n(\mathbf{x_n})$ whose head has predicate R and a mapping s of its variables to the universe of $\mathbf{A}$ such that $s(\mathbf{x}) = \mathbf{b}$ and, for every $1 \leq i \leq n$, $R_i(s(\mathbf{x_i}))$ holds in $\mathbf{A}$ or is derived by P.

Lemma 1: Let $k \geq 1$, let $\mathbf{A}$ and $\mathbf{B}$ be instances of $\mathrm{CSP}(\mathbf{H})$ and let f be any homomorphism from $\mathbf{A}$ to $\mathbf{B}$. Then P derives $R(f(a_1), \ldots, f(a_r))$ on input $\mathbf{B}$ for every fact $R(a_1, \ldots, a_r)$ derived on input $\mathbf{A}$.

Proof. Follows directly from the definitions. ∎

Note that if P is linear, then we can associate to every fact $R(\mathbf{b})$ derived by P a finite sequence $r_1, \ldots, r_m$ of (possibly repeated) rules, along with, for every rule r_i a mapping s_i of its variables to elements in A in the following way: r_m and s_m are, respectively, the rule and mapping used in the derivation of $R(\mathbf{b})$. If rule r_m is non-recursive then $m = 1$ and we are done. Otherwise, the derivation of $R(\mathbf{b})$ uses a fact, that must have been derived previously. Then, set r_{m-1} and s_{m-1} to be the rule and mapping used to derive it and iterate in the natural way.

We associate to every Datalog rule $T_0 \leftarrow T_1, \ldots, T_n$ the first order formula $\forall \mathbf{y}\ (T_0 \leftarrow T_1 \wedge \cdots \wedge T_n)$ where $\mathbf{y}$ is a tuple with the variables occurring in the rule. Indeed, we might sometimes slightly abuse notation and move freely between a rule and its associated formula.

We shall denote by $\mathbf{H}^*$ the result of expanding $\mathbf{H}$ with all relations on H. Formally, if τ is the signature of $\mathbf{H}$, then the signature, τ^* of τ contains in addition to all predicates in τ a predicate P_R for each relation R on H. Then $\mathbf{H}^*$ is defined to be the structure with the same universe as $\mathbf{H}$ such that every predicate in τ is interpreted as in $\mathbf{H}$ and every predicate P_R in $\tau^* \setminus \tau$ is intepreted as R.

For every $k \geq 1$, the canonical linear symmetric k-ary program (for $\mathbf{H}$) is the program P defined as follows:

Program P contains an EDB for every predicate in τ and an IDB for every predicate of arity at most k in $\tau^* \setminus \tau$. We shall abuse notation and, when no confusion is possible, use the same symbol R to denote both the predicate and its interpretation in $\mathbf{H}^*$. Note that there are two 0-ary relations on H: the empty set, denoted as `false` (which will be the goal predicate) and the relation containing a tuple of length 0, which we will denote as `true`.

Program P contains all linear rules r with at most k variables such that both r and its symmetric complement (or rather, such that both the formula associated to r and to its symmetric complement) are logically valid in $\mathbf{H}^*$. Note that there are finitely many different rules with at most k variables modulo renaming of variables and removing repeated atomic formulas in the body.

We shall use 'k-program' as a shortand for 'canonical symmetric Datalog k-ary program for $\mathbf{H}$'. We say that an instance $\mathbf{A}$ passes the k-test if the k-program does not derive `false` on input $\mathbf{A}$.

Note that it follows from the definition that for every $k \geq 0$, the k-program always contains the rule

$$\texttt{true} \leftarrow$$

whose head is `true` and body is empty. Furthermore, it will be convenient several times to assume that every derivation starts with this rule. We can assume this WLOG because every application of a non-recursive rule $T_0 \leftarrow T_1, T_2, \ldots, T_n$ can be simulated by applying first the rule '`true` $\leftarrow$', followed by rule $T_0 \leftarrow \texttt{true}, T_1, T_2, \ldots, T_n$ (which must be also present in the k-program).

Also, note that it follows directly from the definitions and Lemma 1 that if the k-program derives the goal predicate on some instance $\mathbf{A}$, then $\mathbf{A}$ must be unsatisfiable.

We say that $\neg\,\mathrm{CSP}(\mathbf{H})$ *is definable in symmetric Datalog* if there is a symmetric Datalog program P such that for every instance $\mathbf{A}$, of $\mathrm{CSP}(\mathbf{H})$, $\mathbf{A}$ is unsatisfiable if and only if P derives the goal predicate on input $\mathbf{A}$. It is easy to see (see [19]) that if $\neg\,\mathrm{CSP}(\mathbf{H})$ is definable by a symmetric Datalog program with at most k variables per rule then it is definable by the k-program.

V. A COMBINATORIAL CHARACTERIZATION OF SYMMETRIC DATALOG

The following lemma gives a combinatorial perspective on the canonical symmetric Datalog program. It is similar in spirit to the characterization in terms of obstructions given in [33] although the formalization given here, in terms of realizations, is more suitable for our proofs.

Lemma 2: Let $\mathbf{H}$ be a structure and let $k \geq 1$. Then, for every instance $\mathbf{A}$ of $\mathrm{CSP}(\mathbf{H})$ the following two sentences are equivalent:

1) $\mathbf{A}$ fails the k-test.
2) there exists some walk ω on $\mathbf{A}$ of width k such that $\text{zz-real}^{\mathbf{A}}_{\omega} = \emptyset$.

Although Lemma 2 could easily be obtained from [33] we include a complete proof for the sake of completeness and, more importantly, because an auxiliary lemma used in its proof (mainly Lemma 3) will be used intensively later.

Lemma 2 follows by combining the following two lemmas.

Lemma 3: Let $\mathbf{A}$ be an instance of $\mathrm{CSP}(\mathbf{H})$ and let $\omega = \mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_n}, \mathbf{a_n}$ be a walk of width at most k on $\mathbf{A}$. Then, on instance $\mathbf{A}$, the k-program derives $R(\mathbf{a_n})$ where $R = \mathrm{last}(\text{zz-real}_{\omega})$.

Proof. For every $1 \leq i \leq n$ consider the relation R_i on H defined as

$$R_i = \bigcup_{\omega'} \mathrm{last}(\mathrm{real}_{\omega'})$$

where the union ranges over all subwalks $\omega' = \mathbf{b_{j_1}}, \mathbf{a_{j_1}}, \ldots, \mathbf{b_{j_m}}, \mathbf{a_{j_m}}$ with $j_m = i$ of any zigzag-expansion of ω.

We shall show that, for every $1 \leq i \leq n$, $R_i(\mathbf{a_i})$ is derived by the k-program. Then, the lemma follows from the fact that $R_n = \mathrm{last}(\text{zz-real}_{\omega})$. We prove the claim by induction on i:

(Case $i = 1$) We can assume WLOG that both $\mathbf{b_1}$ and $\mathbf{a_1}$ are a 0-ary tuple. It follows that $R_1 = \texttt{true}$ and the claim follows since the rule '$\texttt{true} \leftarrow$' belongs to the k-program.

(Case $i > 1$) We can assume WLOG that $\mathbf{b_i}$ does not contain repeated elements. Let $\mathbf{b_i} = (b_1, b_2, \ldots, b_u)$, let $\mathbf{a_i} = (b_{k_1}, b_{k_2}, \ldots, b_{k_v})$, and let $\mathbf{a_{i-1}} = (b_{l_1}, b_{l_2}, \ldots, b_{l_w})$. Consider rule r with variables $y_1, y_2, \ldots, y_u$ given by

$$\forall y_1, \ldots, y_u \; R_i(y_{k_1}, y_{k_2}, \ldots, y_{k_v}) \leftarrow F(y_1, y_2, \ldots, y_u) \wedge R_{i-1}(y_{l_1}, y_{l_2}, \ldots, y_{l_w})$$

where $F(y_1, y_2, \ldots, y_u)$ is a conjunction that contains, for every predicate R and every $t_1, \ldots, t_s \in \{1, \ldots, u\}$ such that $(b_{t_1}, b_{t_2}, \ldots, b_{t_s}) \in R^{\mathbf{A}}$, the atomic formula $R(y_{t_1}, y_{t_2}, \ldots, y_{t_s})$.

Rule r has at most k variables. We shall prove that r is logically valid on $\mathbf{H}^*$. Let s be any assignment of its variables to H satisfying the body of r, that is, such that

$$\mathbf{H}^* \models F(s(y_1), s(y_2), \ldots, s(y_u)) \wedge R_{i-1}(s(y_{l_1}), s(y_{l_2}), \ldots, s(y_{l_w}))$$

It follows from the definition of $F(y_1, y_2, \ldots, y_u)$ that $b_p \mapsto s(y_p), 1 \leq p \leq u$ is a partial homomorphism from $\mathbf{A}$ to $\mathbf{H}$. Furthermore, since $(s(y_{l_1}), s(y_{l_2}), \ldots, s(y_{l_w})) \in R_{i-1}$, there is a subwalk ω' of a zigzag expansion of ω ending at $\mathbf{a_{i-1}}$ such that $(s(y_{l_1}), s(y_{l_2}), \ldots, s(y_{l_w})) \in \mathrm{last}(\mathrm{real}_{\omega'})$. It follows that $(s(y_{k_1}), s(y_{k_2}), \ldots, s(y_{k_v}))$ belongs to $\mathrm{last}(\mathrm{real}_{(\omega', \mathbf{b_i}, \mathbf{a_i})})$ where $(\omega', \mathbf{b_i}, \mathbf{a_i})$ denotes the new walk obtained by adding $\mathbf{b_i}, \mathbf{a_i}$ at the end of ω'. It follows from the definition of R_i that $R_i(s(y_{k_1}), s(y_{k_2}), \ldots, s(y_{k_v}))$ holds in $\mathbf{H}^*$.

One can prove analogously that the symmetric complement of r is also logically valid on $\mathbf{H}^*$. Thus, rule r belongs to the k-program. Now, observe that it follows from the definition of F that $F(\mathbf{b_i})$ holds in $\mathbf{A}$. Furthermore, by induction, $R_{i-1}(\mathbf{a_{i-1}})$ is derived by the k-program. It follows that, using rule r, the k-program can derive $R_i(\mathbf{a_i})$. ∎

Lemma 4: Let $k \geq 1$ and let $\mathbf{A}$ be an instance of $\mathrm{CSP}(\mathbf{H})$ that fails the k-test. Then there exists some walk ω of width k such that $\text{zz-real}^{\mathbf{A}}_{\omega} = \emptyset$.

Proof. By definition, since $\mathbf{A}$ fails the k-test, the k-program derives `false` on instance $\mathbf{A}$. The sequence of derivations that yields to `false` is given by a sequence of (possibly repeated) rules $r_1, \ldots, r_n$ along with, for every rule r_i, $1 \leq i \leq n$, a mapping s_i of its variables to elements in A. For every rule r_i, let $\mathbf{y_i}$ be a tuple with its variables, let $R_i(\mathbf{x_i})$ be its head, and define $\mathbf{b_i} = s_i(\mathbf{y_i})$ and $\mathbf{a_i} = s_i(\mathbf{x_i})$.

Let $\omega = \mathbf{b_1}, \mathbf{a_1}, \mathbf{b_2}, \mathbf{a_2}, \ldots, \mathbf{b_n}, \mathbf{a_n}$. We shall prove that $\text{zz-real}^{\mathbf{A}}_{\omega} = \emptyset$.

Let ω' be a zigzag-expansion of ω, let ω'' be any subwalk of ω' and let $\mathbf{a_i}$ be the last element in ω''. We shall show by induction on the length, j, of ω'' that $\mathrm{last}(\mathrm{real}_{\omega''}) \subseteq R_i$. Since $R_n = \texttt{false}$, it follows that $\mathrm{real}_{\omega'} = \emptyset$.

(Case $j = 1$) This follows directly by assuming WLOG that r_1 is the rule '$\texttt{true} \leftarrow$'.

(Case $j > 1$) We know that the last three tuples of ω'' are (in order) $\mathbf{a_{i-1}}, \mathbf{b_i}, \mathbf{a_i}$ or $\mathbf{a_{i+1}}, \mathbf{b_{i+1}}, \mathbf{a_i}$ for some $i \in \{1, \ldots, n\}$. Consider the first case and let ω''' be the subwalk of ω'' obtained removing $\mathbf{b_i}, \mathbf{a_i}$ at the end of ω''. Let $\mathbf{b_i} = (b_1, b_2, \ldots, b_u)$, let $\mathbf{a_i} = (b_{k_1}, b_{k_2}, \ldots, b_{k_v})$ and let $\mathbf{a_{i-1}} = (b_{l_1}, b_{l_2}, \ldots, b_{l_w})$. It follows directly from the definition that $\mathrm{last}(\mathrm{real}_{\omega''})$ is the relation:

$$\{h(\mathbf{a_i}) \mid h \text{ is an homomorphism from } \mathbf{A}_{|\{b_1, \ldots, b_u\}} \text{ to } \mathbf{H} \text{ such that } h(\mathbf{a_{i-1}}) \in \mathrm{real}_{\omega'''}\}$$

Let $\mathbf{y_i} = (y_1, \ldots, y_u)$ and write rule r_i (or rather its associated first-order formula) as

$$\forall y_1, \ldots, y_u \; R_i(y_{k_1}, \ldots, y_{k_v}) \leftarrow R_{i-1}(y_{l_1}, \ldots, y_{l_w}) \wedge S(y_1, \ldots, y_u)$$

where R_{i-1} is the IDB occurring in the body and $S(y_1, \ldots, y_u)$ is the conjunction of all other atomic predicates occurring in the body.

Let h be any homomorphism from $\mathbf{A}_{|\{b_1, \ldots, b_u\}}$ to $\mathbf{H}$ such that $h(\mathbf{a_{i-1}}) \in \mathrm{real}_{\omega'''}$. Since the atomic formulas in $S(y_1, \ldots, y_u)$ only contain EDBs it follows that $S(h(\mathbf{b_i}))$

holds in $\mathbf{H}$ (and hence in $\mathbf{H}^*$). Furthermore, since $h(\mathbf{a_{i-1}}) \in \text{real}_{\omega'''}$ it follows from induction that $h(\mathbf{a_{i-1}}) \in R_{i-1}$. Then,

$$\mathbf{H}^* \models R_{i-1}(h(\mathbf{a_{i-1}})) \wedge S(h(\mathbf{b_i}))$$

Since r_i is logically valid on $\mathbf{H}^*$ it follows that

$$\mathbf{H}^* \models R_i(h(\mathbf{a_i}))$$

or, equivalently, that $h(\mathbf{a_i}) \in R_i$ and we are done.

Assume now that the last three tuples of ω'' are $\mathbf{a_{i+1}}, \mathbf{b_{i+1}}, \mathbf{a_i}$. As before, let ω''' be the subwalk of ω'' obtained removing $\mathbf{b_{i+1}}, \mathbf{a_i}$ at the end of ω'. Let $\mathbf{b_{i+1}} = (b_1, \ldots, b_u)$, let $\mathbf{a_{i+1}} = (b_{k_1}, \ldots, b_{k_v})$, and let $\mathbf{a_i} = (b_{l_1}, \ldots, b_{l_w})$. Let $\mathbf{y_{i+1}} = (y_1, \ldots, y_u)$ and write rule r_{i+1} as

$$\forall y_1, \ldots, y_u \; R_{i+1}(y_{k_1}, \ldots, y_{k_v}) \leftarrow R_i(y_{l_1}, \ldots, y_{l_w}) \wedge S(y_1, \ldots, y_u)$$

where R_i is the IDB occurring in the body and $S(y_1, \ldots, y_u)$ is the conjunction of all other atomic predicates occurring in the body. The proof goes as in the previous case. We only need to use the fact that the k-program also contains the symmetric complement of r_{i+1}, i.e., the rule

$$\forall y_1, \ldots, y_u \; R_i(y_{l_1}, \ldots, y_{l_w}) \leftarrow R_{i+1}(y_{k_1}, \ldots, y_{k_v}) \wedge S(y_1, \ldots, y_u)$$

■

VI. Application: Conservative CSPs with binary relations

Let $\mathbf{H}$ be a structure and let τ be its signature. Structure $\mathbf{H}$ is said to be *conservative* if it contains, among its relations, all subsets of its universe, that is, if $\mathbf{H}$ is of the form $(H, R_1^{\mathbf{H}}, \ldots, R_m^{\mathbf{H}}, U_1^{\mathbf{H}}, \ldots, U_n^{\mathbf{H}})$ where $\{U_1^{\mathbf{H}}, \ldots, U_n^{\mathbf{H}}\} = 2^H$.

An *order* is any binary relation of the form $\{(a,a), (a,b), (b,b)\}$ where a and b are different elements.

A sequence $f_1, \ldots, f_r$ of ternary operations is called a *Hagemann-Mitschke chain* (HM chain) of lenght r if it satisfies the identifies:

- $x = f_1(x,y,y)$
- $f_i(x,x,y) = f_{i+1}(x,y,y)$ for all $i = 1, \ldots, r-1$
- $f_r(x,x,y) = y$

An operation $f : H^j \to H$ is a *polymorphism* of a relation $R \subseteq H^k$ if for every $\mathbf{a_1}, \ldots, \mathbf{a_j} \in R$, the tuple $f(\mathbf{a_1}, \ldots, \mathbf{a_j})$ obtained by applying f component-wise to $\mathbf{a_1}, \ldots, \mathbf{a_j}$ belongs to R. Then, f is said to be a polymorphism of $\mathbf{H}$ if it is a polymorphism of each one of its relations. We say that $\mathbf{H}$ admits an HM-chain $f_1, \ldots, f_r$ if each f_i is a polymorphism of $\mathbf{H}$. Larose and Tesson [6] proved that if $\neg\,\text{CSP}(\mathbf{H})$ is definable by symmetric Datalog then $\mathbf{H}$ admits a HM chain of polymorphisms and conjectured that the converse also holds. In this section we verify this conjecture for conservative structures with relations of arity at most 2.

Theorem 1: Let $\mathbf{H}$ be a conservative structure with relations of arity at most 2. Then, the following are equivalent:

1) It is not possible to pp-define an order from $\mathbf{H}$.
2) $\mathbf{H}$ admits an HM-chain.
3) $\neg\,\text{CSP}(\mathbf{H})$ is definable in symmetric Datalog.

If one of these conditions holds then $\text{CSP}(\mathbf{H})$ is solvable in logspace, otherwise it is hard for nondeterministic logspace.

Thorought the section $\mathbf{H}$ is a conservative structure. Before embarking on the proof Theorem 1 we need first to introduce a number of concepts and technical lemmas. This is done in the next section.

A. Some technical preliminaires

A *list* L (for conservative template $\mathbf{H}$) is any mapping with range 2^H. We say that L is the list of an instance $\mathbf{A}$ of $\text{CSP}(\mathbf{H})$ if $A = \text{dom}(L)$ and, for every $a \in A$, $L(a)$ is precisely $\bigcap_{i \in I} U_i^{\mathbf{H}}$ where $I = \{i \mid 1 \leq i \leq n \text{ and } a \in U_i^{\mathbf{A}}\}$. It follows directly from the definition of homomorphism that if $\mathbf{A}$ and $\mathbf{A}'$ are instances of $\text{CSP}(\mathbf{H})$ with the same universe, binary relations, and associated list, then both are satisfiable or none. Similarly, it follows from the definition of the canonical program that if the k-program derives some fact in one of the instances then also derives it in the other. In consequence, in order to prove Theorem 1 we can assume, if necessary, that in an instance $\mathbf{A}$ of $\text{CSP}(\mathbf{H})$, every $a \in A$ participates in exactly one of the relations $U_1^{\mathbf{A}}, \ldots, U_n^{\mathbf{A}}$. In this case, one can unambigously express instance $\mathbf{A}$ as a pair $(\mathbf{G}, L)$ where $\mathbf{G} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ is a structure that contains the binary relations of $\mathbf{A}$ and L is the list of $\mathbf{A}$.

For every $k \geq 1$ we shall denote by $\text{list}_k^{(\mathbf{G},L)}$ the list that maps every element a of G to the set $X_1 \cap \cdots \cap X_r$ where $X_1(a), \ldots, X_r(a)$ are all unary facts derived by the k-program on input $(\mathbf{G}, L)$ in which a occurs. To simplify notation we shall write L_k instead of $\text{list}_k^{\mathbf{G},L}$ whenever $\mathbf{G}$ is clear from the context.

Let L be a list (for $\mathbf{H}$) and let (a,b) be an ordered pair of different elements from H. We say that (a,b) is a *good* pair of L (for $\mathbf{H}$) if the following two conditions are satisfied:

1) $\{a,b\} \in L(x)$ for some element $x \in \text{dom}(L)$
2) For every element $y \in \text{dom}(L)$, every $a', b' \in L(y)$, and every binary relation R pp-definable from $\mathbf{H}$: if $(a,a'), (b,b'), (b,a') \in R$ then $(a,b') \in R$.

Lemma 5: Assume that no order can be pp-defined from $\mathbf{H}$ and let L be a list for $\mathbf{H}$ such that $|L(x)| > 1$ for some $x \in \text{dom}(L)$. Then L has a good pair.

Proof. Construct a sequence $(a_1, b_1), R_1, (a_2, b_2), R_2, \ldots$ in the following way. Let a_1, b_1 be any two different elements appearing in $L(y_1)$ for some $y_1 \in \text{dom}(L)$. If (a_1, b_1) satisfy condition (2) of the definition of good pair then stop. Otherwise, we set $a_2, b_2 \in H$ and $R_1 \subseteq H^2$ in such a way that $a_2, b_2 \in L(y_2)$ for some $y_2 \in \text{dom}(L)$ and R_1 is a binary relation pp-definable from $\mathbf{H}$ that contains (a_1, a_2), (b_1, b_2), and (b_1, a_2) but not (a_1, b_2). We can assume, intersecting R_1 with $\{a_1, b_1\} \times \{a_2, b_2\}$ if necessary, that R_1 is exactly $\{(a_1, a_2), (b_1, b_2), (b_1, a_2)\}$. We claim that the iteration of this procedure must eventually come to a halt. It then follows that the last pair (a_n, b_n) in the sequence is a good pair for L.

Let us proof our claim by contradiction. Indeed, if the sequence $(a_1, b_1), R_1, (a_2, b_2), R_2, \ldots$ is infinite then we could find $i \neq j$ with $(a_i, b_i) = (a_j, b_j)$. Then, the relation pp-defined with the formula $\exists x_{i+1}, \ldots, x_{j-1} R_i(y, x_{i+1}) \wedge \cdots \wedge R_j(x_{j-1}, z)$ is an order, a contradiction. ∎

Let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$), let $\omega = (v_1, \ldots, v_n)$ be a simple walk on $\mathbf{A}$ and let $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ be realizations of ω. We say that $\mathbf{a}$ and $\mathbf{b}$ avoid each other if $\{(a_i, b_{i+1}), (b_i, a_{i+1})\} \cap \mathrm{pr}_{i,i+1} \mathrm{real}_\omega = \emptyset$ for every $1 \leq i < n$.

Lemma 6: Assume that no order can be pp-defined from $\mathbf{H}$. Let $(\mathbf{G}, L)$ be an instance of CSP($\mathbf{H}$), let (a, b) be a good pair of L, let ω be a simple walk in $(\mathbf{G}, L)$, and let $\mathbf{a}, \mathbf{b} \in \mathrm{real}_\omega$ be realizations of ω that avoid each other such that $\mathbf{a}$ starts and ends at a and $\mathbf{b}$ starts and ends at b. Then there is no realization $\mathbf{c} \in \mathrm{real}_\omega$ that starts at a and ends at b.

Proof. Let $\omega = (v_1, \ldots, v_n)$, $\mathbf{a} = (a_1, \ldots, a_n)$, and $\mathbf{b} = (b_1, \ldots, b_n)$. Assume, towards a contradiction that there exists some $\mathbf{c} = (c_1, \ldots, c_n)$ falsifying the lemma.

Let i be smallest such that $\{(b_i, c_{i+1}), (c_i, b_{i+1})\} \cap \mathrm{pr}_{i,i+1} \mathrm{real}_\omega \neq \emptyset$ (such i always exists because $c_n = b_n = b$).

Assume first that $(c_i, b_{i+1}) \in \mathrm{pr}_{i,i+1} \mathrm{real}_\omega$. Consider the simple walk $\omega' = (v_1, \ldots, v_i, v_{i+1}, v_i, \ldots, v_1)$ and let R be the relation

$$\mathrm{real}_{\omega'} \cap \{a_1, b_1, c_1\} \times \cdots \times \{a_i, b_i, c_i\} \times \{a_{i+1}, b_{i+1}\} \times \{a_i, b_i\} \times \cdots \times \{a_1, b_1\}$$

It follows that R contains $(a_1, \ldots, a_i, a_{i+1}, a_i, \ldots, a_1)$, $(b_1, \ldots, b_i, b_{i+1}, b_i, \ldots, b_1)$ and $(c_1, \ldots, c_i, b_{i+1}, b_i, \ldots, b_1)$, and hence $\{(a, a), (a, b), (b, b)\} \subseteq \mathrm{extremes}(R)$. It follows from the fact that $\mathbf{a}$ and $\mathbf{b}$ avoid each other and the minimality of i that $(b, a) \notin \mathrm{extremes}(R)$. Since $\mathrm{extremes}(R)$ is pp-definable from $\mathbf{H}$, we obtain a contradiction.

Otherwise, $(b_i, c_{i+1}) \in \mathrm{pr}_{i,i+1} \mathrm{real}_\omega$ and $(c_i, b_{i+1}) \notin \mathrm{pr}_{i,i+1} \mathrm{real}_\omega$. Now, let ω' be $(v_1, \ldots, v_i, v_{i+1})$ and let R be the relation

$$\mathrm{real}_{\omega'} \cap \{b_1, c_1\} \times \cdots \times \{b_i, c_i\} \times \{b_{i+1}, c_{i+1}\}$$

We obtain that $\mathrm{extremes}(R) = \{(a, c_{i+1}), (b, c_{i+1}), (b, b_{i+1})\}$ contradicting that (a, b) is a good pair. ∎

The following two lemmas follow from Lemma 3.

Lemma 7: Let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$) that passes the 2-test, and let L be its list. If $|L(x)| = 1$ for every element $x \in G$ then $\mathbf{A}$ has a solution.

Proof. We show that the mapping h sending every element $a \in A$ to the only element in $L(a)$ is a solution. Indeed, let R be any predicate, let $\mathbf{a} \in R^{\mathbf{A}}$, and let $\omega = \mathbf{b_1}, \mathbf{a_1}$ be the walk with $\mathbf{a_1} = \mathbf{b_1} = \mathbf{a}$. We know from Lemma 3 that $\mathrm{last}(\text{zz-real}_\omega) \neq \emptyset$ (Note that we are using the fact that all relations in $\mathbf{H}$ have arity at most 2). Since $|L(a)| = 1$ for every $a \in A$, it follows that $h(\mathbf{a})$ is the only possible tuple in $\mathrm{last}(\text{zz-real}_\omega)$. Consequently, $h(\mathbf{a}) \in R^{\mathbf{H}}$. ∎

Lemma 8: Let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$), let $\omega = (x, \ldots, y)$ be a simple walk. Then, the 3-program can derive a fact $R(x, y)$ where $R \subseteq \mathrm{extremes}(\text{zz-real}_\omega)$.

Proof. Consider the walk ω' of width 3 obtained by adding x to every tuple of ω. It follows from Lemma 3 that the 3-program can derive $R(x, y)$ where $R = \mathrm{last}(\text{zz-real}_{\omega'})$. It follows easily from the definition of zigzag-realization and the choice of ω' that $R \subseteq \mathrm{extremes}(\text{zz-real}_\omega)$. ∎

Let S and R be relations on H of arity s and r respectively. We denote by $S \times R$ the cartesian product of S and R. Also, we define $S \to R$ to be the $(s+r)$-ary relation $\{(h_1, \ldots, h_{s+r}) \in H^{s+r} \mid (h_1, \ldots, h_s) \notin S \vee (h_{s+1}, \ldots, h_{s+r}) \in R\}$.

Lemma 9: Let r be a rule of the j-program with head $R_0(\mathbf{x_0})$ and body $R_1(\mathbf{x_1}) \wedge \cdots \wedge R_n(\mathbf{x_n})$ where R_1 is an IDB, and let $S(\mathbf{z})$ be an atomic formula where S is an IDB of arity k and $\mathbf{z}$ is a tuple of variables. Then the following holds:

1) The rule obtained from r in the following way:
   a) Replace the head by $(S \times R_0)(\mathbf{z}, \mathbf{x_0})$
   b) Replace $R_1(\mathbf{x_1})$ by $(S \times R_1)(\mathbf{z}, \mathbf{x_1})$
   c) Remove $S(\mathbf{z})$ from the body, if present

   belongs to the $(j + k)$-program.
2) The rule obtained from r in the following way:
   a) Replace the head by $(S \to R_0)(\mathbf{z}, \mathbf{x_0})$
   b) Replace $R_1(\mathbf{x_1})$ by $(S \to R_1)(\mathbf{z}, \mathbf{x_1})$
   c) Remove $S(\mathbf{z})$ from the body, if present

   belongs to the $(j + k)$-program.

Proof. Let r' be the rule generated in (1) or (2). It is only necessary to show that both r' and its symmetric rule are logically valid on $\mathbf{H}^*$. This follows directly from the semantics of first-order logic. For the sake of completeness we enclose a proof in the appendix. ∎

Lemma 10: Let $j, k \geq 0$, let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$), and let $R(\mathbf{a})$ be a fact derived by the j-program on $\mathbf{A}$. Then, for every IDB S of arity k, every $\mathbf{b} \in A^k$, and every $\ell \geq j+k$, the ℓ-program derives $(S \times R)(\mathbf{b}, \mathbf{a})$ iff it derives $S(\mathbf{b})$.

Proof. Consider the sequence $r_1, \ldots, r_n$ of (possibly repeated) rules that are used in the derivation of $R(\mathbf{a})$ by the j-program. We can assume WLOG that r_1 is the rule '$\texttt{true} \leftarrow$'. Let $\mathbf{x}$ be a tuple of k different fresh variables and consider the new sequence $r'_2, \ldots, r'_n$ of rules where r'_i is obtained from r_i and $S(\mathbf{x})$ as in Lemma 9(1). Observe that every rule from $r'_2, \ldots, r'_n$ has at most $j + k$ variables.

Assume that $S(\mathbf{b})$ is derived by the ℓ-program. Then we can apply rules $r'_2, \ldots, r'_n$ (mimicking the derivation of $R(\mathbf{a})$) to derive $(S \times R)(\mathbf{b}, \mathbf{a})$ (here and in the rest of the proof we are using that $(S \times \texttt{true})$ is equivalent to S). Conversely, if the ℓ-program derives $(S \times R)(\mathbf{b}, \mathbf{a})$ then we can apply rules $r'_n, \ldots, r'_2$ (mimicking again the derivation of $R(\mathbf{a})$ but this time in reverse order) to obtain a derivation of $S(\mathbf{b})$. ∎

Lemma 11: Let $j \geq 1$, let $(\mathbf{G}, L)$ be any instance of CSP($\mathbf{H}$), let $a \in G$ be any of its elements and let R be $L_j(a)$. Then the $(j + 1)$-program on instance $(\mathbf{G}, L)$ derives $R(a)$.

Proof. Let $X_1(a), \ldots, X_m(a)$ be the set of all unary facts derived by the j-program for element a. It follows from Lemma 10, that the $(j+1)$-program can derive $(X_1 \times X_2)(a, a)$ and hence $(X_1 \cap X_2)(a)$ since, as it is readily seen, the rule

$(X_1 \cap X_2)(x) \leftarrow (X_1 \times X_2)(x,x)$ belongs to the $(j+1)$-program. Iterative application of this argument shows that the $(j+1)$-program can derive $(X_1 \cap \cdots \cap X_m)(a)$. Since, by definition, $R = X_1 \cap \cdots \cap X_m$, we are done. ∎

Lemma 12: For every $j, k \geq 0$ and every instance $(\mathbf{G}, L)$ of CSP($\mathbf{H}$) the following holds: every fact $R(\mathbf{b})$ derived by the k-program on $(\mathbf{G}, L_j)$ is also derived by the $(1+j+2k)$-program on $(\mathbf{G}, L)$.

Proof. Let $\ell = 1 + j + 2k$. We shall prove the lemma by induction on the number of rule applications that the k-program needs to derive $R(\mathbf{b})$ on $(\mathbf{G}, L_j)$.

(Base case) $R(\mathbf{b})$ is derived by a non-recursive rule. We can assume WLOG that the rule is '$\texttt{true} \leftarrow$' and the claim follows.

(Inductive case) $R(\mathbf{b})$ is derived by a recursive rule r with head $R(\mathbf{x_0})$ and body $R_1(\mathbf{x_1}) \wedge \cdots \wedge R_n(\mathbf{x_n})$ where R_1 is the IDB occurring in the body.

Let s be the instantiation used in the derivation of $R(\mathbf{b})$ and, for every $1 \leq i \leq n$, let $\mathbf{a_i} = s(\mathbf{x_i})$. We can assume, reordering terms if necessary, that there exists some $m \in \{1, \ldots, n\}$ such that for every $i \in \{2, \ldots, n\}$, $R_i(\mathbf{a_i})$ holds in $(\mathbf{G}, L)$ if and only if $i > m$. We can also assume that $m - 1 \leq k$. Indeed, for every $i \in \{2, \ldots, m\}$, since $(\mathbf{G}, L_j)$ and $(\mathbf{G}, L)$ only differ on their list it follows that R_i is unary and that R_i is precisely the list of the only element in $\mathbf{a_i}$. Hence, if the body of r does not contain repeated atomic predicates then we might have at most one such atomic predicate per variable in r.

By iterative application of Lemma 9(1) the rule r' with head $(R \times R_2 \times \cdots \times R_m)(\mathbf{x_0}, \mathbf{x_2}, \cdots, \mathbf{x_m})$ and body $(R_1 \times R_2 \times \cdots \times R_m)(\mathbf{x_1}, \ldots, \mathbf{x_m}) \wedge R_{m+1}(\mathbf{x_{m+1}}) \wedge \cdots \wedge R_n(\mathbf{x_n})$ belongs to the $2k$-program (and hence to the ℓ-program).

In what follows all the derivations are on instance $(\mathbf{G}, L)$. By inductive hypothesis, $R_1(\mathbf{a_1})$ is derived by the ℓ-program. Furthermore, by Lemma 11, $R_2(\mathbf{a_2})$ is derived by the $(j+1)$-program. It follows from Lemma 10 that the ℓ-program derives $(R_1 \times R_2)(\mathbf{a_1}, \mathbf{a_2})$. Then, by iterative application of the same argument it follows that the ℓ-program derives $(R_1 \times R_2 \times \cdots \times R_m)(\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m})$. Since, $R_{m+1}(\mathbf{a_{m+1}}), \ldots, R_n(\mathbf{a_n})$ hold in $(\mathbf{G}, L)$ it follows that, using rule r', the ℓ-program can derive $(R \times R_2 \times \cdots \times R_m)(\mathbf{b}, \mathbf{a_2}, \ldots, \mathbf{a_m})$.

Now, since the $(j+1)$-program derives $R_m(\mathbf{a_m})$ it follows from Lemma 10 that the ℓ-program derives $(R \times R_2 \times \cdots \times R_{m-1})(\mathbf{b}, \mathbf{a_2}, \ldots, \mathbf{a_{m-1}})$. Iterative application of the same argument yields that the ℓ-program derives $R(\mathbf{b})$.

∎

Corollary 1: For every $j, k \geq 0$ and for every instance $(\mathbf{G}, L)$ of CSP($\mathbf{H}$) the following holds: If $(\mathbf{G}, L)$ passes the $(1+j+2k)$-test then $(\mathbf{G}, L_j)$-passes the k-test.

Lemma 13: Let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$), let S be some EDB, let $\mathbf{a} \in A^j$ where j is the arity of S, and let $\mathbf{A}'$ be the instance obtained, from $\mathbf{A}$ by adding $\mathbf{a}$ to $S^{\mathbf{A}}$. Then, the $(j+k)$-program on input $\mathbf{A}$ derives $(S \to R)(\mathbf{a}, \mathbf{b})$ for every fact $R(\mathbf{b})$ derived by the k-program on $\mathbf{A}'$.

Proof. We prove it by induction on the number of rule applications that the k-program needs derive $R(\mathbf{b})$ on $\mathbf{A}'$.

(Base case) $R(\mathbf{b})$ is derived by a non-recursive rule. We can assume WLOG that the rule is '$\texttt{true} \leftarrow$'. Note that $(S \to \texttt{true})$ is precisely H^j and hence the rule '$(S \to \texttt{true})(\mathbf{x}) \leftarrow$' belongs to the j-program.

(Inductive case) $R(\mathbf{b})$ is derived by a recursive rule r with head $R(\mathbf{y})$ and body $R_1(\mathbf{x_1}) \wedge \cdots \wedge R_n(\mathbf{x_n})$ where R_1 is the IDB occurring in the body. Let s be the instantiation used in the derivation of $R(\mathbf{b})$ and, for every $1 \leq i \leq n$, let $\mathbf{a_i} = s(\mathbf{x_i})$. We can assume that $S(\mathbf{a})$ appears at most once in $R_2(\mathbf{a_2}), \ldots, R_n(\mathbf{a_n})$. Indeed, if $R_i(\mathbf{a_i}) = R_{i'}(\mathbf{a_{i'}}) = S(\mathbf{a})$ then replace r by the new rule obtained by identifying $\mathbf{x_i}$ and $\mathbf{x_{i'}}$ (meaning that we identify the first variable in $\mathbf{x_i}$ with the first variable in $\mathbf{x_{i'}}$ and so on) and removing repeated atomic predicates. Clearly, this new rule still belongs to the k-program (since it is logically valid in $\mathbf{H}^*$) and can be used to derive $R(\mathbf{b})$.

Let $\mathbf{x}$ be a tuple of j variables defined in the following way: If, $S(\mathbf{a}) = R_i(\mathbf{a_i})$ for some $1 \leq i \leq n$ then set $\mathbf{x}$ to be $\mathbf{x_i}$. Otherwise, set $\mathbf{x}$ to be a tuple of fresh variables.

Let r' be the rule obtained from r and $S(\mathbf{x})$ as in Lemma 9(2). Note that r' has at most $j+k$ variables. It follows from inductive hypothesis that $(S \to R)(\mathbf{a}, \mathbf{a_1})$ is derived by the $(j+k)$-program on $\mathbf{A}$. It follows easily that, using r', the $(j+k)$-program can derive $(S \to R)(\mathbf{a}, \mathbf{b})$ on input $\mathbf{A}$. ∎

Corollary 2: Let $\mathbf{A}$ be an instance of CSP($\mathbf{H}$), let S be a unary relation on H, let $\overline{S}$ be its complement (that is $\overline{S} = H \setminus S$), let $a \in A$, and let $\mathbf{A}'$ be the new instance obtained from $\mathbf{A}$ by adding a to $S^{\mathbf{A}}$. If $\mathbf{A}'$ does not pass the k-test then the $(k+1)$-program derives $\overline{S}(a)$ on $\mathbf{A}$.

Proof. The result follows from Lemma 13 by noticing that if $R = \texttt{false}$ then $S \to R$ is precisely $\overline{S}$. ∎

B. Proof of Theorem 1

A CSP describable by a symmetric Datalog program is solvable in logspace by results in [17]; the hardness result is from [6]. The direction (3) ⇒ (2) was shown in [6] whereas the direction (2) ⇒ (1) is well-known and easy to show (we include a proof in the appendix for the sake of completeness). So it only remains to show (1) ⇒ (3).

The *weight* of a list L is defined to be the cardinality of the following set:

$$\{X \subseteq H \mid X \subseteq L(x) \text{ for some } x \in \mathrm{dom}(L) \text{ and } |X| \geq 2\}$$

Let $k : \mathbf{N} \to \mathbf{N}$ be the solution of the recurrence:

1) $k(0) = 2$
2) $k(n) = 12 + 6k(n-1)$, $n > 0$

In order to prove (1) ⇒ (3) we shall show that for every $n \geq 0$, every instance whose list has weight at most n and passes the $k(n)$-test is satisfiable. Since the weight of any list for $\mathbf{H}$ is bounded above by $2^{|H|} - |H| - 1$ it follows that the $k(2^{|H|} - |H| - 1)$-program defines $\neg\,$CSP($\mathbf{H}$). We shall prove the claim by contradiction. Let n be the smallest number falsifying the claim.

Lemma 14: There exists some instance $(\mathbf{G}, L)$ of $\mathrm{CSP}(\mathbf{H})$ where L has weight n, a good pair (a, b) of L, and an element x of G such that:

1) $(\mathbf{G}, L)$ passes the $k(n)$-test,
2) $\{a, b\} \subseteq L_{11+4k(n-1)}(x)$, and
3) the new instance obtained removing b from $L(x)$ does not pass the $k(n)$-test.

Proof. Let $(\mathbf{G}, L)$ be any instance with $\mathrm{weight}(L) \leq n$ that passes the $k(n)$-test and does not have a solution. We can assume WLOG that the list L is minimal in the sense that if for any $x \in G$ we remove any element from $L(x)$, then the resulting instance does not pass the $k(n)$-test. We have $n > 0$ since otherwise it would follow from $k(0) = 2$ and Lemma 7 that $(\mathbf{G}, L)$ has a solution. It follows from Lemma 5 that L has a good pair (a, b). It only remains to show that condition (2) is satisfied. Assume, towards a contradiction that $\{a, b\} \not\subseteq L_{11+4k(n-1)}(x)$ for every element x. It follows that the weight of $L_{11+4k(n-1)}$ is at most $n - 1$. It follows from condition (2) of the definition of $k(n)$ and Corollary 1 that $(\mathbf{G}, L_{11+4k(n-1)})$ passes the $k(n-1)$-test. By the minimality of n, $(\mathbf{G}, L_{11+4k(n-1)})$ (and hence $(\mathbf{G}, L)$) has a solution, which contradicts our assumption. ∎

The next lemma is where Lemma 2 is used.

Lemma 15: There exists some instance $(\mathbf{G}, L)$ of $\mathrm{CSP}(\mathbf{H})$ where L has weight n, a good pair (a, b) of L, and a set $X \subseteq G$ such that:

1) $(\mathbf{G}, L)$ is satisfiable,
2) $\{a, b\} \subseteq L_{11+4k(n-1)}(x)$ for every $x \in X$,
3) there is no solution g of $(\mathbf{G}, L)$ such that $g(x) \neq b$ for every $x \in X$, and
4) $(b, b) \in \mathrm{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G}, L_2)})$ for every simple walk ω whose both extremes belong to X.

Proof. We shall need the following construction. Let $\mathbf{A}$ be any instance of $\mathrm{CSP}(\mathbf{H})$ and let $\omega = \mathbf{b_1}, \mathbf{a_1}, \ldots, \mathbf{b_m}, \mathbf{a_m}$ be any walk on $\mathbf{A}$. We shall define inductively a structure $\mathrm{struct}_{\omega}^{\mathbf{A}}$ and a homomorphism f from the universe of $\mathrm{struct}_{\omega}^{\mathbf{A}}$ to $\mathbf{A}$ in the following way:

For every $i \in \{1, \ldots, m\}$, let $\mathbf{b_i} = (b_1, \ldots, b_{r_i})$, and let $\mathbf{B}_i$ be any isomorphic copy of $\mathbf{B}_{|\{b_1, \ldots, b_{r_i}\}}$. Consider the disjoint union of $\mathbf{B}_i$, $1 \leq i \leq m$ denoted $\biguplus_{1 \leq i \leq m} \mathbf{B}_i$ and the mapping f that sends every element a' in it to the element a in $\mathbf{A}$ of which a' is a copy. The structure $\mathrm{struct}_{\omega}^{\mathbf{A}}$ is the result of applying to $\biguplus_{1 \leq i \leq m} \mathbf{B}_i$ the following 'gluing process': for every $1 \leq i < m$ and every element a in $\mathbf{a_i}$ glue a' and a'' where a' is the only element in $\mathbf{B}_i$ with $f(a') = a$ and a'' is the only element in $\mathbf{B}_{i+1}$ with $f(a'') = a$.

It follows directly from the construction that f is a homomorphism from $\mathrm{struct}_{\omega}^{\mathbf{A}}$ to $\mathbf{A}$ and that $\mathrm{struct}_{\omega}^{\mathbf{A}}$ has a solution if and only if $\mathrm{real}_{\omega}^{\mathbf{A}} \neq \emptyset$.

We are now ready to start the proof of the lemma. Let $(\mathbf{G}, L)$, (a, b), and x satisfying the conditions of Lemma 14. Then, instance $(\mathbf{G}, L')$ fails the $k(n)$-test where L' is the list that sets $L'(x) = L(x) \setminus \{b\}$ and is identical to L for any variable different than x. It follows from Lemma 2 that there exists a walk ω of width $k(n)$ on $(\mathbf{G}, L')$ such that $\text{zz-real}_{\omega}^{(\mathbf{G}, L')} = \emptyset$. Since $(\mathbf{G}, L)$ passes the $k(n)$-test, it follows again from Lemma 2 that there exists a zigzag-expansion ω' of ω such that $\mathrm{real}_{\omega'}^{(\mathbf{G}, L)} \neq \emptyset$. Furthermore, since $\text{zz-real}_{\omega}^{(\mathbf{G}, L')} = \emptyset$ we have that $\mathrm{real}_{\omega'}^{(\mathbf{G}, L')} = \emptyset$.

Let $(\mathbf{T}, I) = \mathrm{struct}_{\omega'}^{(\mathbf{G}, L)}$, let f be the homomorphism from $\mathrm{struct}_{\omega'}^{(\mathbf{G}, L)}$ to $(\mathbf{G}, L)$ as defined at the beginning of the proof, and define X to be $f^{-1}(x)$. We claim that $(\mathbf{T}, I)$, (a, b) and X satisfy the conditions of the lemma.

Condition (1) follows from the definition of $(\mathbf{T}, I)$ and the fact that $\mathrm{real}_{\omega'}^{(\mathbf{G}, L)} \neq \emptyset$. Notice also that for every $u \in T$, $I(u) = L(f(u))$. This implies that $\mathrm{weight}(I) \leq \mathrm{weight}(L) \leq n$ and that (a, b) is a good pair of I. Condition (2) follows from the fact that f is an homomorphism from $(\mathbf{T}, I)$ to $(\mathbf{G}, L)$ and Lemma 1. To show (3), observe that the solutions g of (T, I) such that $g(u) \neq b$ for every $u \in X$ are precisely the solutions of instance (T, I') where I' is the list that acts as I in all elements outside X and is defined to be $I'(u) = I(u) \setminus \{b\}$ for all elements $u \in X$. Notice that (T, I') is (up to isomorphism) precisely $\mathrm{struct}_{\omega'}^{\mathbf{G}, L'}$. Consequently, (3) follows from $\mathrm{real}_{\omega'}^{(\mathbf{G}, L')} = \emptyset$.

Let us show item (4). Let ω'' be any simple walk in $(\mathbf{T}, I_2)$ whose extremes y and z belong to X and assume towards a contradiction that $(b, b) \notin \mathrm{extremes}(\text{zz-real}_{\omega''}^{(\mathbf{T}, I_2)})$. It follows from Lemma 8 and Lemma 12 that the 9-program derives on instance $(\mathbf{T}, I)$ some fact $S(y, z)$ with $(b, b) \notin S$. Consequently, by Lemma 13 the 9-program derives $S(x, x)$ on $(\mathbf{G}, L)$, in contradiction with the fact that $b \in L_{11+4k(n-1)}(x) \subseteq L_9(x)$. ∎

Let $(\mathbf{G}, L)$, (a, b), and X satisfying the conditions of Lemma 15. Let $Y \subseteq X$ be maximal with the property that there exists a solution, say f, of $(\mathbf{G}, L)$ satisfying $f(y) \neq b$ for every $y \in Y$. Let x be any element in $X \setminus Y$ and define Z to be the set containing every node $y \in G$ satisfying the following:

there exists a simple walk ω in $(\mathbf{G}, L_2)$ from x to y such that $(a, b) \notin \mathrm{extremes}(\text{zz-real}_{\omega \cdot \omega^{-1}}^{(\mathbf{G}, L_2)})$.

Lemma 16: There is a solution g of $(\mathbf{G}, L_2)_{|Z}$ such that $g(x) = a$.

Proof. We have $\{a, b\} \subseteq L_{11+4k(n-1)}(x)$ from the choice of x and condition (2) of Lemma 15. It follows from Lemma 12 and $a \in L_{11+4k(n-1)}(x)$ that $H \setminus \{a\}(x)$ has not been derived by the $4+2k(n-1)$-program on input $(\mathbf{G}, L_2)$. Consider now the instance $(\mathbf{G}, J)$ where J is the list that acts as L_2 with the exception of $J(x)$, which is set to $\{a\}$. It follows from Corollary 2 that $(\mathbf{G}, J)$ passes the $3 + 2k(n-1)$-test.

We claim that $J_2(z) \subsetneq L(z)$ for every $z \in Z$. Then, it follows that the list of the instance $(\mathbf{G}, J_2)_{|Z}$ has weight at most $n - 1$. Since, by Corollary 1 this instance passes the $k(n-1)$-test it follows by the minimality of n that it has a solution g, which is also a solution of $(\mathbf{G}, L_2)_{|Z}$. Furthermore,

this solution must necessarily satisfy $g(x) = a$ (since $J(x) = \{a\}$) and we are done.

We shall now prove the pending claim. Let z be any element in Z and let ω be the walk that places z in Z. Then, there exists some $b' \in L(z)$ such that $(b, b') \in \text{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G},L_2)})$. Otherwise, by Lemma 3, the 2-program on instance $(\mathbf{G}, L_2)$ would derive a fact $B(x)$ with $b \notin B$. Then, it would follow by Lemma 12 that $b \notin L_7(z)$, in contradiction with the fact that $b \in L_{11+4k(n-1)}(z)$.

Let $A = \text{last}(\text{zz-real}_{\omega}^{(\mathbf{G},J)})$. We shall prove that $b' \notin A$, which completes the proof, since by Lemma 3, $J_2(z) \subseteq A$. We prove the claim by contradiction. Indeed, if $b' \in A$, then, by definition, $(a, b') \in \text{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G},L_2)})$, which joinly with $(b, b') \in \text{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G},L_2)})$ would imply that $(a, b) \in \text{extremes}(\text{zz-real}_{\omega\cdot\omega^{-1}}^{(\mathbf{G},L_2)})$, contradicting the assumption $z \in Z$.

■

Let $h : G \to H$ be the mapping that acts as f for all vertices outside Z and that acts as g inside Z.

Let R be a predicate and let (x, x') be any tuple in $R^{(\mathbf{G},L)}$.

Lemma 17: $(h(x'), h(x'')) \in R$

Proof. If both endpoints x' and x'' belong to Z or both do not belong to Z then the claim follows directly from the definition of h. Assume then that $x' \in Z$ and $x'' \notin Z$ (the case $x'' \in Z$ and $x' \notin Z$ is analogous).

Let $(b', b'') = (f(x'), f(x''))$. Since f is a solution of $(\mathbf{G}, L)$ it follows that (b', b'') belongs to R. Let $a' = g(x')$. Since $a' \in L_2(x')$ it follows from Lemma 3 that that $(a', a'') \in R$ for some $a'' \in L(x'')$.

To complete the proof we assume, towards a contradiction, that $(a', b'') \notin R$. In what follows all the realizations are for instance $(\mathbf{G}, L)$ unless explicitely stated.

Let $w = v_1, \ldots, v_n$ be the walk that has placed x' in Z and let (ω, x'') be the walk $v_1, \ldots, v_n, v_{n+1}$ with $v_{n+1} = x''$. We define two realizations $\mathbf{a}$ and $\mathbf{b}$ in $\text{real}_{(\omega,x'')}$.

Realization $\mathbf{a}$ is defined to be $(a_1, \ldots, a_{n+1})$ where $a_i = g(v_i)$ if $i \leq n$ and a'' if $i = n + 1$ whereas realization $\mathbf{b}$ is defined to be $(b_1, \ldots, b_{n+1})$ where $b_i = f(v_i)$. Since both f and g are solutions of $(\mathbf{G}, L_2)$ it follows that $\{a_i, b_i\} \subseteq L_2(v_i)$ for all $i \in \{1, \ldots, n\}$.

We shall prove that $\mathbf{a}$ and $\mathbf{b}$ avoid each other by contradiction. Let i be minimum such that (a_i, b_{i+1}) or (b_i, a_{i+1}) belong to $\text{pr}_{i,i+1}\,\text{real}_{(\omega,x'')}$

Consider two cases:

(case $i < n$) Then it would follow that $(a, b) \in \text{extremes}(\text{zz-real}_{\omega\cdot\omega^{-1}}^{(\mathbf{G},L_2)})$, in contradiction with the fact that $x' \in Z$.

(case $i = n$) We have $(a_n, a_{n+1}) = (a', a'')$ and $(b_n, b_{n+1}) = (b', b'')$. We need to show that

$$\{(a', b''), (b', a'')\} \cap \text{pr}_{n,n+1}\,\text{real}_{(\omega,x'')} = \emptyset$$

We have by assumption that $(a', b'') \notin \text{pr}_{n,n+1}\,\text{real}_{(\omega,z)}$. Regarding (b', a''), let R be the relation

$$\text{real}_{(\omega,x'')} \cap \{a_1, b_1\} \times \cdots \times \{a_{n+1}, b_{n+1}\}$$

We have that $\text{extremes}(R)$ contains (a, a'') and (b, b'') and does not contain (a, b''). By the condition (2) of good pair it follows that it does not contain (b, a'') either. This completes the proof that $\mathbf{a}$ and $\mathbf{b}$ avoid each other.

Now, since $x'' \notin Z$, it follows that there is a zigzag-expansion ω' of $v_1, \ldots, v_n, v_{n+1}, v_n, \ldots, v_1$ such that $\text{real}_{\omega'}^{\mathbf{G},L_2}$ (and hence $\text{real}_{\omega'}^{\mathbf{G},L}$) contains a realization $\mathbf{c}' = (a, \ldots, b)$ starting at a and ending at b. Also, we can define in the obvious way from $\mathbf{a}$ and $\mathbf{b}$, two realizations $\mathbf{a}' = (a, \ldots, a)$ and $\mathbf{b}' = (b, \ldots, b)$ in $\text{real}_{\omega'}$ such that $\mathbf{a}'$ and $\mathbf{b}'$ avoid each other. It follows that realizations $\mathbf{a}', \mathbf{b}', \mathbf{c}'$ contradict Lemma 6

■

It follows from the previous lemma that h is a solution of $(\mathbf{G}, L)$. The next lemma shows that $h(y) \neq b$ for every $y \in Y \cup \{x\}$, in contradiction with the maximality of Y.

Lemma 18: $h(y) \neq b$ for every $y \in Y \cup \{x\}$

Proof. We consider two cases:

(case $y \notin Z$) Then $h(y) = f(y) \neq b$.

(case $y \in Z$) Assume, towards a contradiction, that $h(y) = b$. Let ω be the path from x to y that places y in Z. Then $(a, b) \in \text{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G},L_2)})$. This is because the image $g(\omega)$, according to g, of the vertices in ω gives a realization that starts at a and ends with b. Also, from item (4) of Lemma 15 it follows that $(b, b) \in \text{extremes}(\text{zz-real}_{\omega}^{(\mathbf{G},L_2)})$. It follows that $(a, b) \in \text{extremes}(\text{zz-real}_{\omega\cdot\omega^{-1}}^{(\mathbf{G},L_2)})$, contradicting that $y \in Z$.

■

This completes the proof of Theorem 1.

VII. Appendix

A. Proof of Lemma 9

Assume first that r is the rule generated in (1). It is only necessary to show that both r' and its symmetric complement are logically valid on $\mathbf{H}^*$. We enclose here only the proof for r' as the proof of its symmetric complement is analogous. We can assume, reordering if necessary, that if $S(\mathbf{z})$ appears in the body at all then it is precisely $R_n(\mathbf{x_n})$. Hence, r' is

$$(S \times R)(\mathbf{z}, \mathbf{x_0}) \leftarrow (S \times R_1)(\mathbf{z}, \mathbf{x_1}) \wedge R_2(\mathbf{x_2}) \wedge \cdots \wedge R_m(\mathbf{x_m})$$

where $m = n - 1$ or $m = n$ (depending on whether $S(\mathbf{z})$ appears or not in the body). Our goal is to show that

$$\mathbf{H}^* \models (S \times R_0)(s(\mathbf{z}), s(\mathbf{x_0})) \leftarrow (S \times R_1)(s(\mathbf{z}), s(\mathbf{x_1})) \wedge R_2(s(\mathbf{x_2})) \wedge \cdots \wedge R_m(s(\mathbf{x_m}))$$

where s is any mapping from the variables of rule r' to H.

Assume then that $(S \times R_1)(s(\mathbf{z}), s(\mathbf{x_1})) \wedge R_2(s(\mathbf{x_2})) \wedge \cdots \wedge R_m(s(\mathbf{x_m}))$ holds in $\mathbf{H}^*$. Since $(S \times R_1)(s(\mathbf{z}), s(\mathbf{x_1}))$ holds in $\mathbf{H}^*$ it follows both $S(s(\mathbf{z}))$ and $R_1(s(\mathbf{x_1}))$ hold in $\mathbf{H}^*$. The latter implies that

$$\mathbf{H}^* \models R_1(s(\mathbf{x_1})) \wedge \cdots \wedge R_m(s(\mathbf{x_m})).$$

It also follows that

$$\mathbf{H}^* \models R_1(s(\mathbf{x_1})) \wedge \cdots \wedge R_n(s(\mathbf{x_n}))$$

This is because either $m = n$ (and there is nothing to prove) or $R_n(\mathbf{x_n}) = S(\mathbf{z})$. Since r is logically valid in $\mathbf{H}^*$ it follows that $R_0(s(\mathbf{x_0}))$ holds in $\mathbf{H}^*$. Since $S(s(\mathbf{z}))$ also holds in $\mathbf{H}^*$, it follows that $\mathbf{H}^* \models (S \times R_0)(s(\mathbf{z}), s(\mathbf{x_0}))$.

Assume now that r' is the rule generated by (2). Again we shall show that r' is logically valid on $\mathbf{H}^*$ and ommit the proof for its symmetric complement. We assume again that if $S(\mathbf{z})$ appears in the body at all, then it is precisely $R_n(\mathbf{x_n})$. Hence, r' is

$$(S \to R_0)(\mathbf{z}, \mathbf{x_0}) \leftarrow (S \to R_1)(\mathbf{z}, \mathbf{x_1}) \wedge R_2(\mathbf{x_2}) \wedge \cdots \wedge R_m(\mathbf{x_m})$$

where $m = n - 1$ or $m = n$. Our goal is to show that

$$\mathbf{H}^* \models (S \to R_0)(s(\mathbf{z}), s(\mathbf{x_0})) \leftarrow (S \to R_1)(s(\mathbf{z}), s(\mathbf{x_1})) \wedge R_2(s(\mathbf{x_2})) \wedge \cdots \wedge R_m(s(\mathbf{x_m}))$$

where s is any mapping from the variables of rule r' to H. Assume then that $(S \to R_1)(s(\mathbf{z}), s(\mathbf{x_1})) \wedge R_2(s(\mathbf{x_2})) \wedge \cdots \wedge R_m(s(\mathbf{x_m}))$ holds in $\mathbf{H}^*$. We can assume that $\mathbf{H}^* \models S(s(\mathbf{z}))$ since otherwise $\mathbf{H}^* \models (S \to R_0)(s(\mathbf{z}), s(\mathbf{x_0}))$ and we are done. It follows that $\mathbf{H}^* \models R_1(s(\mathbf{x_1}))$. Consequently,

$$\mathbf{H}^* \models R_1(s(\mathbf{x_1})) \wedge \cdots \wedge R_m(s(\mathbf{x_m}))$$

It also follows that

$$\mathbf{H}^* \models R_1(s(\mathbf{x_1})) \wedge \cdots \wedge R_n(s(\mathbf{x_n}))$$

This is becuase either $m = n$ (and there is nothing to prove) or $R_n(\mathbf{x_n}) = S(\mathbf{z})$. Since r is logically valid in $\mathbf{H}^*$ it follows that $R_0(s(\mathbf{x_0}))$ (and hence $(S \to R_0)(s(\mathbf{z}), s(\mathbf{x_0}))$) also holds in $\mathbf{H}^*$.

B. Proof of $(\mathbf{2}) \Rightarrow (\mathbf{1})$ *in Theorem 1*

Assume towards a contradiction that an order $\{(a, a), (a, b), (b, b)\}$ is pp-definable from $\mathbf{H}$ and that $\mathbf{H}$ admits an HM chaim $f_1, \ldots, f_r$. For every $1 \leq i \leq r$, let us denote by $\mathbf{a_i}$ be the result of applying f_i to (b, b), (a, b), and (a, a) component-wise. We shall prove by induction that $\mathbf{a_i} = (b, b)$ for every $1 \leq i \leq r$.

First, since $\{(a, a), (a, b), (b, b)\}$ is pp-definable from $\mathbf{H}$ it follows (see for instance [4]) that it also admits $f_1, \ldots, f_r$ as polymorphisms. It follows that $\mathbf{a_i} \in \{(a, a), (a, b), (b, b)\}$ for every $1 \leq i \leq r$. Consequently, to prove our claim it is only necessary to show that the first coordinate of $\mathbf{a_i}$ is b. If $i = 1$, this is done by the first identify of HM whereas if $i > 1$ then it follows from the inductive hypothesis and the second HM identity. Finally, notice that $\mathbf{a_r} = (b, b)$ contradicts the third HM identity.

REFERENCES

[1] P. Jeavons, "On the algebraic structure of combinatorial problems," *Theor. Comput. Sci.*, vol. 200, no. 1-2, pp. 185–204, 1998.

[2] T. Feder and M. Y. Vardi, "Monotone monadic SNP and constraint satisfaction," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, S. R. Kosaraju, D. S. Johnson, and A. Aggarwal, Eds. ACM, 1993, pp. 612–622.

[3] ——, "The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory," *SIAM J. Comput.*, vol. 28, no. 1, pp. 57–104, 1998. [Online]. Available: http://dx.doi.org/10.1137/S0097539794266766

[4] A. A. Bulatov, P. Jeavons, and A. A. Krokhin, "Classifying the complexity of constraints using finite algebras," *SIAM J. Comput.*, vol. 34, no. 3, pp. 720–742, 2005.

[5] B. Larose and L. Zádori, "Bounded width problems and algebras," *Algebra Universalis*, vol. 56, no. 3-4, pp. 439–466, 2007.

[6] B. Larose and P. Tesson, "Universal algebra and hardness results for constraint satisfaction problems," *Theor. Comput. Sci.*, vol. 410, no. 18, pp. 1629–1647, 2009.

[7] T. J. Schaefer, "The complexity of satisfiability problems," in *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, R. J. Lipton, W. A. Burkhard, W. J. Savitch, E. P. Friedman, and A. V. Aho, Eds. ACM, 1978, pp. 216–226.

[8] L. Libkin, *Elements of finite model theory*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2004.

[9] A. Atserias, "On digraph coloring problems and treewidth duality," *Eur. J. Comb.*, vol. 29, no. 4, pp. 796–820, 2008.

[10] J. Nešetřil and C. Tardif, "Duality theorems for finite structures (characterising gaps and good characterisations)," *J. Combin. Theory Ser. B*, vol. 80, no. 1, pp. 80–97, 2000.

[11] ——, "Short answers to exponentially long questions: extremal aspects of homomorphism duality," *SIAM J. Discrete Math.*, vol. 19, no. 4, pp. 914–920 (electronic), 2005.

[12] B. Larose, C. Loten, and C. Tardif, "A characterisation of first-order constraint satisfaction problems," *Logical Methods in Computer Science*, vol. 3, no. 4, 2007.

[13] B. Larose, M. Valeriote, and L. Zádori, "Omitting types, bounded width and the ability to count," *IJAC*, vol. 19, no. 5, pp. 647–668, 2009. [Online]. Available: http://dx.doi.org/10.1142/S021819670900524X

[14] L. Barto and M. Kozik, "Constraint satisfaction problems solvable by local consistency methods," *J. ACM*, vol. 61, no. 1, p. 3, 2014.

[15] A. Atserias, A. A. Bulatov, and A. Dawar, "Affine systems of equations and counting infinitary logic," *Theor. Comput. Sci.*, vol. 410, no. 18, pp. 1666–1683, 2009.

[16] L. Barto and M. Kozik, "Robust satisfiability of constraint satisfaction problems," in *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, H. J. Karloff and T. Pitassi, Eds. ACM, 2012, pp. 931–940.

[17] L. Egri, B. Larose, and P. Tesson, "Symmetric datalog and constraint satisfaction problems in logspace," in *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*. IEEE Computer Society, 2007, pp. 193–202. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4276538

[18] O. Reingold, "Undirected connectivity in log-space," *J. ACM*, vol. 55, no. 4, 2008.

[19] V. Dalmau and B. Larose, "Maltsev + datalog –> symmetric datalog," in *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*. IEEE Computer Society, 2008, pp. 297–306.

[20] L. Egri, A. A. Krokhin, B. Larose, and P. Tesson, "The complexity of the list homomorphism problem for graphs," *Theory Comput. Syst.*, vol. 51, no. 2, pp. 143–178, 2012.

[21] J. Bulin, D. Delic, M. Jackson, and T. Niven, "On the reduction of the CSP dichotomy conjecture to digraphs," in *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, ser. Lecture Notes in Computer Science, C. Schulte, Ed., vol. 8124. Springer, 2013, pp. 184–199.

[22] P. Erdős, A. L. Rubin, and H. Taylor, "Choosability in graphs," in *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing (Humboldt State Univ., Arcata, Calif., 1979)*, ser. Congress. Numer., XXVI. Utilitas Math., Winnipeg, Man., 1980, pp. 125–157.

[23] T. Feder, P. Hell, and J. Huang, "Bi-arc graphs and the complexity of list homomorphisms," *J. Graph Theory*, vol. 42, no. 1, pp. 61–80, 2003.

[24] P. Hell and A. Rafiey, "The dichotomy of list homomorphisms for digraphs," in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, D. Randall, Ed. SIAM, 2011, pp. 1703–1713.

[25] N. Alon and M. Tarsi, "Colorings and orientations of graphs," *Combinatorica*, vol. 12, no. 2, pp. 125–134, 1992.

[26] C. Thomassen, "Every planar graph is 5-choosable," *J. Combin. Theory Ser. B*, vol. 62, no. 1, pp. 180–181, 1994.

[27] A. A. Bulatov, "Complexity of conservative constraint satisfaction problems," *ACM Trans. Comput. Log.*, vol. 12, no. 4, p. 24, 2011.

[28] A. Kazda, "Constraint satisfaction problem and universal algebra," Ph.D. dissertation, Charles University, 2013.

[29] V. Dalmau, "Linear datalog and bounded path duality of relational structures," *Logical Methods in Computer Science*, vol. 1, no. 1, 2005.

[30] V. Dalmau, P. G. Kolaitis, and M. Y. Vardi, "Constraint satisfaction, bounded treewidth, and finite-variable logics," in *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, ser. Lecture Notes in Computer Science, P. V. Hentenryck, Ed., vol. 2470. Springer, 2002, pp. 310–326.

[31] A. A. Bulatov, A. A. Krokhin, and B. Larose, "Dualities for constraint satisfaction problems," in *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*., ser. Lecture Notes in Computer Science, N. Creignou, P. G. Kolaitis, and H. Vollmer, Eds., vol. 5250. Springer, 2008, pp. 93–124.

[32] L. Egri, P. Hell, B. Larose, and A. Rafiey, "Space complexity of list H-colouring: a dichotomy," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, C. Chekuri, Ed. SIAM, 2014, pp. 349–365.

[33] L. Egri, "On constraint satisfaction problems below p," *Journal of Logic and Computation*, 2014.