On Constraint Satisfaction Problems below P*

László Egri

School of Computer Science, McGill University
Montreal, Canada
laszlo.egri@mail.mcgill.ca

Abstract

Symmetric Datalog, a fragment of the logic programming language *Datalog*, is conjectured to capture all constraint satisfaction problems (CSP) in logarithmic space [10]. Therefore developing tools that help us understand whether or not a CSP can be defined in symmetric Datalog is an important task. A simple, well-known fact is that for any CSP, a fixed set of structures $\mathcal{O}$ (an *obstruction set*) can be defined such that a CSP instance I is a yes-instance iff no structure in $\mathcal{O}$ *maps homomorphically* to I. A CSP having X*-duality* means that the set $\mathcal{O}$ can be chosen to have property X. It is widely known that a CSP is definable in Datalog and *linear* Datalog iff that CSP has *bounded treewidth* [12] and *bounded pathwidth* duality [6], respectively. In the case of symmetric Datalog, Bulatov, Krokhin and Larose ask for such a duality in [4]. We provide two such dualities, and we give applications. In particular, we give a short and simple new proof of the main result of [8] that "Maltsev + Datalog ⇒ symmetric Datalog".

In the second part of the paper, we provide some evidence for the conjecture that every CSP in nondeterministic logarithmic space (NL) is definable in the Datalog fragment linear Datalog [6]. We recall that every problem in NL can be defined by a linear Datalog program with negation and access to an order over the domain of its input (linDat($\mathfrak{suc}$,¬)) [6, 13, 15], or by a poly-size family of *nondeterministic branching programs* [20]. We consider the following restrictions of the previous models: *read-once* linDat($\mathfrak{suc}$) (1-linDat($\mathfrak{suc}$)), and *monotone read-once* nondeterministic branching programs (mnBP1). Although restricted, these models can still define NL-complete problems such as directed *st*-Connectivity, *and also nontrivial problems in* NL *which are not definable in linear Datalog.* We show that any CSP definable by a 1-linDat($\mathfrak{suc}$) program or by a poly-size family of mnBP1s can also be defined by a linear Datalog program. It also follows that a wide class of CSPs–CSPs which do not have bounded pathwidth duality (e.g. the P-complete Horn-3Sat problem)–cannot be defined by any 1-linDat($\mathfrak{suc}$) program or by any poly-size family of mnBP1s.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Constraint satisfaction problems, complexity classes, Datalog fragments

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.203

1 Introduction

Constraint satisfaction problems (CSP) constitute a unifying framework to study various computational problems arising naturally in various branches of computer science, including artificial intelligence, graph homomorphisms, and database theory. Loosely speaking, an instance of a CSP consists of a list of variables and a set of constraints, each specified by an ordered tuple of variables and a constraint relation over some specified domain. The goal is

* Research supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). We thank Benoit Larose and Pascal Tesson for useful discussions and comments. We also thank the anonymous referees for their in-depth reviews.

Computer Science Logic 2011 (CSL'11).
Editor: Marc Bezem; pp. 203–217

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

then to determine whether variables can be assigned domain values such that all constraints are simultaneously satisfied.

Recent efforts have been directed at classifying the complexity of the so-called *nonuniform* CSP. For a fixed finite set of finite relations Γ, CSP(Γ) denotes the nonuniform CSP corresponding to Γ. The difference between an instance of CSP(Γ) and an instance of the general CSP is that constraints in an instance of CSP(Γ) take the form $(x_{i_1}, \ldots, x_{i_k}) \in R$ for some $R \in \Gamma$. Examples of nonuniform CSPs include k-Sat, Horn-3Sat, Graph H-Coloring, and many others.

For a relational structure $\mathbf{B}$, the homomorphism problem HOM($\mathbf{B}$) takes a structure $\mathbf{A}$ as input, and the task is to determine if there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. For instance, consider structures that contain a single symmetric binary relation, i.e. graphs. A homomorphism from a graph $\mathbf{G}$ to a graph $\mathbf{H}$ is a mapping from $V_{\mathbf{G}}$ to $V_{\mathbf{H}}$ such that any edge of $\mathbf{G}$ is mapped to an edge of $\mathbf{H}$. If $\mathbf{H}$ is a graph with a single edge then HOM($\mathbf{H}$) is the set of graphs which are two-colorable. There is a well-known and straightforward correspondence between the CSP and the homomorphism problem. For this reason, from now on we work only with the homomorphism problem instead of the CSP. Nevertheless, we call HOM($\mathbf{B}$) a CSP and we also write CSP($\mathbf{B}$) instead of HOM($\mathbf{B}$), as it is often done in the literature.

The CSP is of course NP-complete, and therefore research has focused on identifying "islands" of tractable CSPs. The well-known CSP dichotomy conjecture of Feder and Vardi [12] states that every CSP is either tractable or NP-complete, and progress towards this conjecture has been steady during the last fifteen years. From a complexity-theoretic perspective, the classification of CSP($\mathbf{B}$) as in P or being NP-complete is rather coarse and therefore somewhat dissatisfactory. Consequently, understanding the fine-grained complexity of CSPs gained considerable attention during the last few years. Ultimately, one would like to know the precise complexity of a CSP lying in P, i.e. to identify a "standard" complexity class for which a given CSP is complete. Towards this, it was established that Schaefer's P – NP dichotomy for Boolean CSPs [19] can indeed be refined: each CSP over the Boolean domain is either definable in first order logic, or complete for one of the classes L, NL, $\oplus$L, P or NP under AC^0-*reductions* [2]. The question whether some form of this fine-grained classification extends to non-Boolean domains is rather natural. The two most important tools to study CSPs whose complexity is below P are *symmetric Datalog* and *linear Datalog*, syntactic restrictions of the database-inspired logic programming language *Datalog*. We say that co-CSP($\mathbf{B}$)–the complement of CSP($\mathbf{B}$)–is definable in (linear, symmetric) Datalog if the set of structures that do not homomorphically map to $\mathbf{B}$ is accepted by a (linear, symmetric) Datalog program.[1]

Symmetric Datalog programs can be evaluated in logarithmic space (L), and in fact, it is conjectured that if co-CSP($\mathbf{B}$) is in L then it can also be defined in symmetric Datalog [10]. There is a considerable amount of evidence supporting this conjecture (see, for example, [10, 9, 8, 16, 5]), and therefore providing tools to show whether co-CSP($\mathbf{B}$) can be defined in symmetric Datalog is an important task. It is well known and easy to see that for any structure $\mathbf{B}$, there is a set of structures $\mathcal{O}$, called an *obstruction set*, such that a structure $\mathbf{A}$ *homomorphically maps to* $\mathbf{B}$ iff there is no structure in $\mathcal{O}$ that homomorphically maps to $\mathbf{A}$. In fact, there are many possible obstruction sets for any structure $\mathbf{B}$. We say that $\mathbf{B}$ has duality X, if $\mathbf{B}$ has an obstruction set which has the special property X. The following two well-known theorems relate definability of co-CSP($\mathbf{B}$) in Datalog and linear Datalog to $\mathbf{B}$

[1] The reason we define co-CSP($\mathbf{B}$) instead of CSP($\mathbf{B}$) in (linear, symmetric) Datalog is a technicality explained in Section 2.5.

having *bounded treewidth* duality and *bounded pathwidth duality*, respectively:

1. co-CSP($\mathbf{B}$) is definable in Datalog iff $\mathbf{B}$ has bounded treewidth duality [12];
2. co-CSP($\mathbf{B}$) is definable in linear Datalog iff $\mathbf{B}$ has bounded pathwidth duality [6].

It was stated as an open problem in [4] to find a duality for symmetric Datalog in the spirit of the previous two theorems. We provide two such dualities: *symmetric bounded pathwidth duality* (SBPD) and *piecewise symmetric bounded pathwidth duality* (PSBPD). We note that SBPD is a special case of PSBPD. For both bounded treewidth and bounded pathwidth duality, the structures in the obstruction sets are restricted to have some special form. For SBPD and PSBPD the situation is a bit more subtle. In addition that we require the obstruction sets to contain structures only of a special form (they must have bounded pathwidth), the obstruction sets must also possess a certain "symmetric closure" property. To the best of our knowledge, this is the first instance of a duality where in addition to the local requirement that each structure must be of a certain form, the set must also satisfy an interesting global requirement.

Using SBPD, we give a short and simple new proof of the main result of [8] that "Maltsev + Datalog $\Rightarrow$ symmetric Datalog". Considering the simplicity of this proof, we suspect that SBPD (or PSBPD) could be a useful tool in an attempt to prove the *symmetric Datalog conjecture* [16], a conjecture that proposes an algebraic characterization of all CSPs lying in L. An equivalent form of this conjecture is that "Datalog + n-permutability $\Rightarrow$ symmetric Datalog" (by combining results from [14],[3] and [17]), where n-permutability is a generalization of Maltsev.

One way to gain more insight into the dividing line between CSPs in L and NL is through studying the complexity of CSPs corresponding to oriented paths. The only known thing regarding the complexity of these CSPs is that they are all in NL (by combining results from [11, 7, 6]). To make progress in this direction, it is natural to ask whether there are oriented paths for which the CSP is NL-complete and L-complete. We provide two classes of oriented paths, $\mathcal{C}_1$ and $\mathcal{C}_2$, such that for any $\mathbf{B}_1 \in \mathcal{C}_1$, the corresponding CSP is NL-complete, and for any $\mathbf{B}_2 \in \mathcal{C}_2$, the corresponding CSP is L. In fact, it can be seen with the help of [16] that for most $\mathbf{B}_2 \in \mathcal{C}_2$, CSP($\mathbf{B}_2$) is L-complete. To prove the membership of CSP($\mathbf{B}_2$) in L (for $\mathbf{B}_2 \in \mathcal{C}_2$), we use PSBPD in an essential way. One can hope to build on this work to achieve an L-NL dichotomy for oriented paths.

In the second part of the paper, we investigate CSPs in NL. Based on the observation that any CSP known to be in NL is also known to be definable by a linear Datalog program, Dalmau conjectured that every CSP in NL can be defined by a linear Datalog program [6]. Linear Datalog($\mathfrak{suc}$,$\neg$) (linDat($\mathfrak{suc}$,$\neg$)) denotes the extension of linear Datalog in which we allow negation and access to an order over the domain of the input. It is known that any problem in NL can be defined by a linDat($\mathfrak{suc}$,$\neg$) program [6, 13, 15], and therefore one way to prove the above conjecture would be to show that any CSP that can be defined by a linDat($\mathfrak{suc}$,$\neg$) program can also be defined by a linear Datalog program. We consider a restriction of the conjecture because proving it in its full generality would separate NL from P (using [1]).

Read-once linear Datalog($\mathfrak{suc}$) (1-linDat($\mathfrak{suc}$)) is a subclass of linDat($\mathfrak{suc}$,$\neg$), but a subclass that has interesting computational abilities, and for which we are able to find the chink in the armor. We can easily define some NL-complete problems in 1-linDat($\mathfrak{suc}$), such as the CSP directed *st*-connectivity (*st*-Conn), and also problems that are not *homomorphism-closed*, such as determining if the input graph is a clique on 2^n vertices, $n \geq 1$. Because any problem that can be defined with a linear Datalog program must be homomorphism closed, it follows that 1-linDat($\mathfrak{suc}$) can define nontrivial problems which are in NL but which are not definable

by any linear Datalog program. However, our main result shows that if co-CSP($\mathbf{B}$) can be defined by a 1-linDat($\mathfrak{suc}$) program, then co-CSP($\mathbf{B}$) can also be defined by a linear Datalog program. The crux of our argument applies the general case of the Erdős-Ko-Rado theorem to show that a 1-linDat($\mathfrak{suc}$) program does not have enough "memory" to handle structures of unbounded pathwidth.

Our proof establishing the above result for 1-linDat($\mathfrak{suc}$) programs can be adapted to show a parallel result for a subclass of *nondeterministic branching programs*, which constitute an important and well-studied class of computational models (see the book [20]). More precisely, we show that if co-CSP($\mathbf{B}$) can be defined by a *poly-size family of read-once*[2] *monotone nondeterministic* branching programs (mnBP1(poly)) then co-CSP($\mathbf{B}$) can also be defined by a linear Datalog program.[3]

Finally, our results can be interpreted as lower-bounds on a wide class of CSPs: if $\mathbf{B}$ does not have bounded pathwidth duality, then co-CSP($\mathbf{B}$) cannot be defined with any 1-linDat($\mathfrak{suc}$) program or with any mnBP1(poly). A specific example of such a CSP would be the P-complete Horn-3Sat problem, and more generally, Larose and Tesson showed that any CSP whose associated *variety admits the unary, affine or semilattice types* does not have bounded pathwidth duality (see [16] for details).

2 Preliminaries

2.1 Algebra

A *vocabulary* (or *signature*) is a finite set of relation symbols with associated arities. The arity function is denoted with $\mathrm{ar}(\cdot)$. If $\mathbf{A}$ is a relational structure over a vocabulary τ, then $R^{\mathbf{A}}$ denotes the relation of $\mathbf{A}$ associated with the symbol $R \in \tau$. The lightface equivalent of the name of the structure denotes the universe of the structure, e.g. the universe of $\mathbf{A}$ is A.

A *tuple structure* $\tilde{\mathbf{A}}$ over a vocabulary τ is a set of pairs $(R, \mathbf{t})$ where $R \in \tau$ and $\mathbf{t}$ is an $\mathrm{ar}(R)$-tuple. We associate a domain $\tilde{A}$ with a tuple structure: $\tilde{A}$ contains every element that appears in some tuple in $\mathbf{A}$, and possibly some other elements. Clearly, tuple structures are equivalent to relational structures. If $\mathbf{A}$ is a relational structure, we denote the equivalent tuple structure with $\tilde{\mathbf{A}}$, and vice versa. For convenience, we use the two notations interchangeably. We note that *all* structures in this paper are finite.

Let $\mathbf{B}$ be a structure of the same signature as $\mathbf{A}$. A *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a map f from A to B such that $f(R^{\mathbf{A}}) \subseteq R^{\mathbf{B}}$ for each $R \in \tau$. A structure is called a *core* if it has no homomorphism to any of its proper substructures. If there exists a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, we often denote it with $\mathbf{A} \to \mathbf{B}$. If that homomorphism is f, we write $\mathbf{A} \xrightarrow{f} \mathbf{B}$. We denote by CSP($\mathbf{B}$) the class of all τ-structures $\mathbf{A}$ such that $\mathbf{A} \to \mathbf{B}$, and by co-CSP($\mathbf{B}$) the complement of CSP($\mathbf{B}$). If we are given a class of τ-structures $\mathcal{C}$ such that for any $\mathbf{A} \in \mathcal{C}$, and any $\mathbf{B}$ such that $\mathbf{A} \to \mathbf{B}$ it holds that $\mathbf{B} \in \mathcal{C}$, then we say that $\mathcal{C}$ is *homomorphism-closed*. *Isomorphism closure* is defined in a similar way.

An n-ary operation on a set A is a map $f : A^n \to A$. Given an h-ary relation R and an n-ary operation f on the same set A, we say that f *preserves* R or that R is *invariant* under f if the following holds: given any matrix M of size $h \times n$ whose columns are in R,

[2] Our read-once restriction for nondeterministic branching programs is less stringent than the usual definition because we require the programs to be read-once only on certain inputs.

[3] A 1-linDat($\mathfrak{suc}$) can be converted into an mnBP1(poly), so another way to present our results would be to do the proofs in the context of mnBP1s, and then to conclude the parallel result for 1-linDat($\mathfrak{suc}$).

applying f to the rows of M produces an h-tuple in R. A *polymorphism* of a structure $\mathbf{B}$ is an operation f that preserves each relation in $\mathbf{B}$.

▶ **Definition 1** (Maltsev Operation)**.** A ternary operation $f : A^3 \to A$ on a finite set A is called Maltsev if it satisfies the following identities: $f(x, y, y) = f(y, y, x) = x, \forall x, y \in A$.

2.2 Datalog

We provide only an informal introduction to Datalog and its fragments, and the reader can find more details, for example, in [18, 6, 10]. Datalog is a database-inspired query language whose connection with CSP-complexity is now relatively well understood (see e.g. [3]). Let τ be some finite vocabulary. A Datalog program over τ is specified by a finite set of rules of the form $h \leftarrow b_1 \wedge \cdots \wedge b_t$, where h and the b_i are atomic formulas $R(x_1, \ldots, x_k)$. When we specify the variables of an atomic formula, we always list the variables from left to right, or we simply provide a tuple $\mathbf{x}$ of variables whose i-th variable is $\mathbf{x}[i]$. We distinguish two types of relational predicates occurring in a Datalog program: predicates I that occur at least once in the head of a rule (i.e., its left-hand side) are called *intensional database predicates* (*IDBs*) and are not in τ. The predicates which occur only in the body of a rule (its right-hand side) are called *extensional database predicates* (*EDBs*) and must all lie in τ. A rule that contains no IDB in the body is called a *nonrecursive rule*, and a rule that contains at least one IDB in the body is called a *recursive rule*. A Datalog program contains a distinguished IDB of arity 0 which is called the *goal predicate*; a rule whose head IDB is a goal IDB is called a *goal rule*.

Linear Datalog is a syntactic restriction of Datalog in which there is at most one IDB in the body of each rule. The class of linear Datalog programs that contains only rules with at most k variables and IDBs with at most $j \leq k$ variables is denoted with *linear* (j, k)*-Datalog*. We say that the *width* of such a linear Datalog program is (j, k).

Symmetric Datalog is a syntactic restriction of linear Datalog. A linear Datalog program $\mathcal{P}$ is symmetric if for any recursive rule $I(\mathbf{x}) \leftarrow J(\mathbf{y}) \wedge \bar{E}(\mathbf{z})$ of $\mathcal{P}$ (except for goal rules), where $\bar{E}(\mathbf{z})$ is a shorthand for the conjunction of the EDBs of the rule over variables in $\mathbf{z}$, the symmetric pair $J(\mathbf{y}) \leftarrow I(\mathbf{x}) \wedge \bar{E}(\mathbf{z})$ of that rule is also in $\mathcal{P}$. The *width* of a symmetric Datalog program is defined similarly to the width of a linear Datalog program.

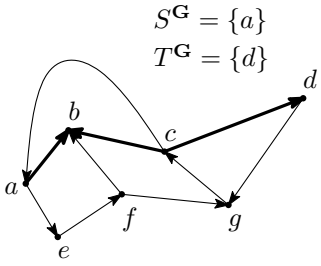
We explain the semantics of linear (symmetric) Datalog using derivations (it could also be explained with *fixed point operators*, but that would be inconvenient for the proofs). Let $\mathcal{P}$ be a linear Datalog program with vocabulary τ. A *$\mathcal{P}$-derivation with codomain D* is a sequence of pairs $\mathscr{D} = (\rho_1, \lambda_1), \ldots, (\rho_q, \lambda_q)$, where ρ_ℓ is a rule of $\mathcal{P}$, and λ_ℓ is a function from the variables V_ℓ of ρ_ℓ to D, $\forall \ell \in [q]$. The sequence $\mathscr{D}$ must satisfy the following properties. Rule ρ_1 is nonrecursive, and ρ_q is a goal rule. For all $\ell \in [q-1]$, the head IDB I of ρ_ℓ is the IDB in the body of $\rho_{\ell+1}$, and if the variables of I in the head of ρ_ℓ and the body of $\rho_{\ell+1}$ are $\mathbf{x}$ and $\mathbf{y}$, respectively, then $\lambda_\ell(\mathbf{x}[i]) = \lambda_{\ell+1}(\mathbf{y}[i])$, $\forall i \in [\mathrm{ar}(I)]$.

Let $R(\mathbf{z})$ be an EDB with variables in some rule ρ_ℓ of a derivation $\mathscr{D}$. Then we write $R(\mathbf{t})$ to denote that $\lambda_\ell(\mathbf{z}) = \mathbf{t}$, i.e. that λ_ℓ *instantiates* the variables of $R(\mathbf{z})$ to $\mathbf{t}$, and we say that $R(\mathbf{t})$ *appears* in ρ_ℓ, or less specifically, that $R(\mathbf{t})$ *appears* in $\mathscr{D}$. Given a structure $\mathbf{A}$ and a derivation $\mathscr{D}$ with codomain A for a program $\mathcal{P}$, we say that $\mathscr{D}$ *is a derivation for* $\mathbf{A}$ if for every $R(\mathbf{t})$ that appears in a rule of $\mathscr{D}$, $(R, \mathbf{t}) \in \tilde{\mathbf{A}}$. We denote a $\mathcal{P}$-derivation for a structure $\mathbf{A}$ with $\mathscr{D}_{\mathcal{P}}(\mathbf{A})$. A linear (symmetric) Datalog program $\mathcal{P}$ *accepts* an input structure $\mathbf{A}$ if there exists a $\mathcal{P}$-derivation for $\mathbf{A}$.

▶ **Definition 2** (Read-Once Derivation)**.** We say that a derivation $\mathscr{D}$ is *read-once* if every $R(\mathbf{t})$ that appears in $\mathscr{D}$ appears exactly once in $\mathscr{D}$, except when R is the special EDB $\mathfrak{suc}$, $\mathfrak{first}$, or $\mathfrak{last}$, defined in Section 4.

An example is given in Fig. 1. The vocabulary is $\tau = \{E^2, S^1, T^1\}$, where the superscripts denote the arity of the symbols. Notice that in the symmetric Datalog program $\mathcal{P}$, rules of types 2 and 3 form a symmetric pair. It is not difficult to see that $\mathcal{P}$ accepts a τ-structure $\mathbf{A}$ iff there is an oriented path (see Section 3.1) in $E^{\mathbf{A}}$ from an element in $S^{\mathbf{A}}$ to an element in $T^{\mathbf{A}}$.

$$
\begin{aligned}
I(x) &\leftarrow S(x) && (1)\\
I(y) &\leftarrow I(x) \wedge E(x,y) && (2)\\
I(x) &\leftarrow I(y) \wedge E(x,y) && (3)\\
G &\leftarrow I(x) \wedge T(x) && (4)
\end{aligned}
$$



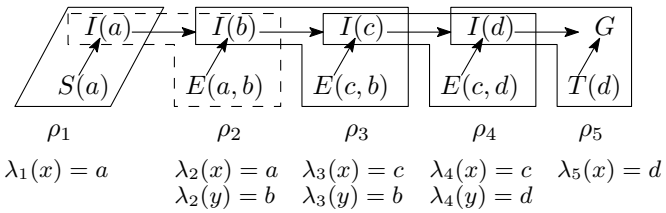


Figure 1 *Top left:* Symmetric Datalog program $\mathcal{P}$. *Top right:* Input structure $\mathbf{G}$ where the binary relation $E^{\mathbf{G}}$ is specified by the digraph. *Bottom:* Visualization of a $\mathcal{P}$-derivation $\mathscr{D}_{\mathcal{P}}(\mathbf{G}) = (\rho_1, \lambda_1), \ldots, (\rho_5, \lambda_5)$ for $\mathbf{G}$, where ρ_1 is nonrecursive, ρ_2, ρ_4 are rules of type 2, ρ_3 is a rule of type 4, and ρ_5 is the goal rule. For example, the dashed box corresponds to rule ρ_2, and it is the rule $I(y) \leftarrow I(x) \wedge E(x,y)$ of $\mathcal{P}$, where λ_2 assigns a to variable x and b to variable y. Observe that $\mathscr{D}_{\mathcal{P}}(\mathbf{G})$ is read-once.

2.3 Path-Decompositions and Derivations

▶ **Definition 3.** [Path-Decomposition] Let $\mathbf{S}$ be a τ-structure. A (j,k)-*path-decomposition* of $\mathbf{S}$ is a sequence $S_0, \ldots, S_{n-1}$ of subsets of A such that

1. For every $(R, (a_1, \ldots, a_{\mathrm{ar}(R)})) \in \tilde{\mathbf{A}}$, $\exists \ell \in \{0, \ldots, n-1\}$ such that $\{a_1, \ldots, a_{\mathrm{ar}(R)}\} \subseteq S_\ell$;
2. If $a \in S_i \cap S_{i'}$ $(i < i')$ then $a \in S_\ell$ for all $i < \ell < i'$;
3. $\forall \ell \in \{0, \ldots, n-1\}$, $|S_\ell| \leq k$, and $\forall \ell \in \{0, \ldots, n-2\}$, $|S_\ell \cap S_{\ell+1}| \leq j$.

For ease of notation, it will be useful to introduce a concept closely related to path-decompositions. Let τ be a vocabulary. Let $\mathbf{S}$ be a τ-structure that can be expressed as $\mathbf{S} = \mathbf{S}_0 \cup \cdots \cup \mathbf{S}_{n-1}$, where the $S_0, \ldots, S_{n-1}$ (the universes of the $\mathbf{S}_i$) satisfy properties 2 and 3 above. Note that $\cup$ here denotes union, *not* disjoint union of τ-structures. We say that $\mathbf{S}$ is a (j,k)-*path*, and that $(\mathbf{S}_0, \ldots, \mathbf{S}_{n-1})$ is a (j,k)-*path representation* of $\mathbf{S}$. We denote (j,k)-path representations with script letters, e.g. $\mathscr{S} = (\mathbf{S}_0, \ldots, \mathbf{S}_{n-1})$. The substructure $\mathbf{S}_i \cup \cdots \cup \mathbf{S}_{i'}$ of $\mathbf{S}$ (assuming a (j,k)-representation is fixed) is denoted with $\mathbf{S}_{[i,j]}$. We call n the *length* of the representation. Obviously, a structure is a (j,k)-path iff it admits a (j,k)-path-decomposition.

Let $\mathscr{D} = (\rho_1, \lambda_1), \ldots, (\rho_q, \lambda_q)$ be a derivation for some linear or symmetric program $\mathcal{P}$ with vocabulary τ. We can extract from $\mathscr{D}$ a τ-structure $\mathrm{Ex}(\mathscr{D})$ such that $\mathscr{D}$ is a derivation for $\mathrm{Ex}(\mathscr{D})$. We specify $\mathrm{Ex}(\mathscr{D})$ as a tuple structure $\tilde{\mathbf{A}}$: for each $R(\mathbf{t})$ that appears in $\mathscr{D}$

$(R \in \tau)$, we add the pair $(R, \mathbf{t})$ to $\tilde{\mathbf{A}}$, and set $\tilde{A}$ to be the set of those elements that appear in a tuple.

Let $\mathscr{D} = (\rho_1, \lambda_1), \ldots, (\rho_q, \lambda_q)$ be a derivation. For each x that is in a rule ρ_ℓ for some $\ell \in [q]$, call x^ℓ the *indexed version of* x. We define an equivalence relation $\mathrm{Eq}(\mathscr{D})$ on the set of indexed variables of $\mathscr{D}$. First we define a graph $G = (V, E)$ as:

- V is the set of all indexed versions of variables in $\mathscr{D}$;
- $(x^\ell, y^{\ell'}) \in E$ if $\ell' = \ell + 1$, x is the i-th variable of the head IDB I of ρ_ℓ, and y is the i-th variable of the body IDB I of $\rho_{\ell+1}$.

Two indexed variables x^ℓ and $y^{\ell'}$ are related in $\mathrm{Eq}(\mathscr{D})$ if they are connected in G. Observe that if $C = \left\{x_1^{\ell_1}, x_2^{\ell_2}, \ldots, x_c^{\ell_c}\right\}$ is a connected component of G, then it must be that $\lambda_{\ell_1}(x_1) = \lambda_{\ell_2}(x_2) = \cdots = \lambda_{\ell_c}(x_c)$.

▶ **Definition 4** (Free Derivation)**.** Let $\mathcal{P}$ be a linear Datalog program and $\mathscr{D} = (\rho_0, \lambda_0), \ldots, (\rho_q, \lambda_q)$ be a derivation for $\mathcal{P}$. Then $\mathscr{D}$ is said to be *free* if for any two $(x^\ell, y^{\ell'}) \notin \mathrm{Eq}(\mathscr{D})$, $\lambda_\ell(x) \neq \lambda_{\ell'}(y)$.

Intuitively, this definition says that $\mathscr{D}$ is free if any two variables in $\mathscr{D}$ which are not "forced" to have the same value are assigned different values.

2.4 Canonical Programs

Fix a τ-structure $\mathbf{B}$ and $j \leq k$. Let $Q_1, \ldots, Q_n$ be all possible at most j-ary relations over B. The *canonical linear* (j,k)*-Datalog program for* $\mathbf{B}$ *(*(j,k)-CanL($\mathbf{B}$)*)* contains an IDB I_m of the same arity as Q_m for each $m \in [n]$. The rule $I_c(\mathbf{x}) \leftarrow I_d(\mathbf{y}) \wedge \bar{E}(\mathbf{z})$ belongs to the canonical program if it contains at most k variables, and the implication $Q_c(\mathbf{x}) \leftarrow Q_d(\mathbf{y}) \wedge \bar{E}(\mathbf{z})$ is true for all possible instantiation of the variables to elements of B. The goal predicate of this program is the 0-ary IDB I_g, where $Q_g = \emptyset$.

The *canonical symmetric* (j,k)*-Datalog program for* $\mathbf{B}$ *(*(j,k)-CanS($\mathbf{B}$)*)* has the same definition as (j,k)-CanL($\mathbf{B}$), except that it has less rules due to the following additional restriction. If $I_c(\mathbf{x}) \leftarrow I_d(\mathbf{y}) \wedge \bar{E}(\mathbf{z})$ is in the program, then both $Q_c(\mathbf{x}) \leftarrow Q_d(\mathbf{y}) \wedge \bar{E}(\mathbf{z})$ and $Q_d(\mathbf{y}) \leftarrow Q_c(\mathbf{x}) \wedge \bar{E}(\mathbf{z})$ must hold for all possible instantiation of the variables to elements of B. The program (j,k)-CanS($\mathbf{B}$) is obviously symmetric. When it is clear from the context, we write CanL($\mathbf{B}$) and CanS($\mathbf{B}$) instead of (j,k)-CanL($\mathbf{B}$) and (j,k)-CanS($\mathbf{B}$), respectively.

2.5 Defining CSPs

The following discussion applies not just to Datalog but also to its symmetric and linear fragments. It is easy to see that the class of structures accepted by a Datalog program is homomorphism-closed, and therefore it is not possible to define CSP($\mathbf{B}$) in Datalog. However, it is often possible to define co-CSP($\mathbf{B}$) in Datalog. The following definition is key.

▶ **Definition 5** (Obstruction Set)**.** A set $\mathcal{O}$ of τ-structures is called an *obstruction set* for $\mathbf{B}$, if for any τ-structure $\mathbf{A}$, $\mathbf{A} \not\to \mathbf{B}$ iff there exists $\mathbf{S} \in \mathcal{O}$ such that $\mathbf{S} \to \mathbf{A}$.

If $\mathcal{O}$ above can be chosen to have property X, then we say that $\mathbf{B}$ has X-duality.

3 On CSPs in symmetric Datalog

3.1 Definitions

An *oriented path* is a digraph obtained by orienting the edges of an undirected path, i.e. an oriented path has vertices $v_0, \ldots, v_{q+1}$ and edges $e_0, \ldots, e_q$, where e_i is either (v_i, v_{i+1}),

or (v_{i+1}, v_i). The *length* of an oriented path is the number of edges it contains. We call (v_i, v_{i+1}) a *forward edge* and (v_{i+1}, v_i) a *backward edge*. Oriented paths can be thought of as relational structures over the vocabulary $\{E^2\}$, so we denote them with boldface letters.

For an oriented path $\mathbf{P}$, we can find a mapping level : $P \to \{0, 1, 2, \dots\}$ such that $\text{level}(b) = \text{level}(a) + 1$ whenever (a, b) is an edge of $\mathbf{P}$. Clearly, there is a unique such mapping with the smallest possible values. The *level of an edge* (a, b) of $\mathbf{P}$ is $\text{level}(a)$, i.e. the level of the starting vertex of (a, b). The $\text{height}(\mathbf{P})$ of an oriented path $\mathbf{P}$ is $\max_{a \in P} \text{level}(a)$. We say that an oriented path $\mathbf{P}$ is *minimal* if there is precisely one vertex a such that $\text{level}(a) = 0$, and precisely one vertex b such that $\text{level}(b) = \text{height}(\mathbf{P})$.

A *zigzag operator* ξ takes a (j, k)-path representation $\mathscr{S} = (\mathbf{S}_0, \dots, \mathbf{S}_{n-1})$ of a (j, k)-path $\mathbf{S}$ and a minimal oriented path $\mathbf{P} = e_0, \dots, e_q$ such that $\text{height}(\mathbf{P}) = n$, and it returns another (j, k)-path $\xi(\mathscr{S}, \mathbf{P})$. Intuitively, $\xi(\mathscr{S}, \mathbf{P})$ is the (j, k)-path $\mathbf{S}$ "modulated" by $\mathbf{P}$ such that the forward and backward edges e_i of $\mathbf{P}$ are mimicked in $\xi(\mathscr{S}, \mathbf{P})$ by "forward and backward" copies of $\mathbf{S}_{\text{level}(e_i)}$. Before the formal definition, it could help the reader to look at the right side of Fig. 2, where the oriented path used to modulate the (j, k)-path over the vocabulary E^2 (i.e. digraphs) with representation $(\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2)$ is $\mathbf{P}$ on the left side. The right side is a more abstract example, and the reader might find it useful after reading the definition.

We inductively define the (j, k)-path $\xi(\mathscr{S}, \mathbf{P})$ as $(\mathbf{S}_{e_0}, \mathbf{S}_{e_1}, \dots, \mathbf{S}_{e_q})$ together with a sequence of isomorphisms $\varphi_{e_0}, \varphi_{e_1}, \dots, \varphi_{e_q}$, where φ_{e_i} is an isomorphism from $\mathbf{S}_{e_i}$ to $\mathbf{S}_{\text{level}(e_i)}$, $0 \leq i \leq q$. For the base case, we define $\mathbf{S}_{e_0}$ to be an isomorphic copy of $\mathbf{S}_0$, and φ_{e_0} to be the isomorphism that maps $\mathbf{S}_{e_0}$ back to $\mathbf{S}_0$. Assume inductively that $\mathbf{S}_{e_0}, \dots, \mathbf{S}_{e_{i-1}}$ and $\varphi_{e_0}, \dots, \varphi_{e_{i-1}}$ are already defined. Let $\mathbf{S}'_{e_i}$ be an isomorphic copy of $\mathbf{S}_{\text{level}(e_i)}$ with domain disjoint from $S_{e_0} \cup \dots \cup S_{e_{i-1}}$, and fix φ'_{e_i} to be the isomorphism that maps back S'_{e_i} to $S_{\text{level}(e_i)}$. We "glue" $\mathbf{S}'_{e_i}$ to $\mathbf{S}_{e_{i-1}}$ by renaming some elements of $\mathbf{S}'_{e_i}$ to elements of $\mathbf{S}_{e_{i-1}}$. To facilitate understanding, we can think of the already constructed structures $\mathbf{S}_{e_0}, \dots, \mathbf{S}_{e_{i-1}}$ as labels of the edges $e_0, \dots, e_{i-1}$ of $\mathbf{P}$, respectively, and we want to determine $\mathbf{S}_{e_i}$, the label of the next edge. The connection between $\mathbf{S}_{e_{i-1}}$ and $\mathbf{S}_{e_i}$ will be defined such that $\mathbf{S}_{e_{i-1}}$ and $\mathbf{S}_{e_i}$ "mimic" the orientation of the edges e_{i-1} and e_i.

We resume our formal definition. Set $\ell = \text{level}(e_i)$, and let $\ell' = \ell - 1$ if e_i is a forward edge, and $\ell' = \ell + 1$ if e_i is a backward edge. If an element $x \in S'_{e_i}$ and an element $y \in S_{e_{i-1}}$ are both copies of the same element $a \in S_\ell \cap S_{\ell'}$, then rename x to y in S'_{e_i}. After all such elements are renamed, $\mathbf{S}'_{e_i}$ becomes $\mathbf{S}_{e_i}$. That is, for all $a \in S_\ell \cap S_{\ell'}$, rename $\varphi'^{-1}_{e_i}(a)$ in $\mathbf{S}'_{e_i}$ to $\varphi^{-1}_{e_{i-1}}(a)$ to obtain $\mathbf{S}_{e_i}$.

We define the isomorphism φ_{e_i} from $\mathbf{S}_{e_i}$ to $\mathbf{S}_{\text{level}(e_i)}$ as:

$$\varphi_{e_i}(x) = \begin{cases} \varphi'_{e_i}(x) & \text{if } x \in S_{e_i} \text{ and } x \notin S_{e_{i-1}} \\ \varphi_{e_{i-1}}(x) & \text{if } x \in S_{e_i} \cap S_{e_{i-1}}. \end{cases}$$

3.2 Two Dualities for Symmetric Datalog

The two main theorems (Theorems 9 and 16) of this section can be combined to obtain:

▶ **Theorem 6.** *For a finite structure* $\mathbf{B}$*, TFAE:*

1. *There is a symmetric Datalog program that defines* co-CSP($\mathbf{B}$)*;*
2. $\mathbf{B}$ *has symmetric bounded pathwidth duality (for some parameters);*
3. $\mathbf{B}$ *has piecewise symmetric bounded pathwidth duality (for some parameters).*

Details follow.

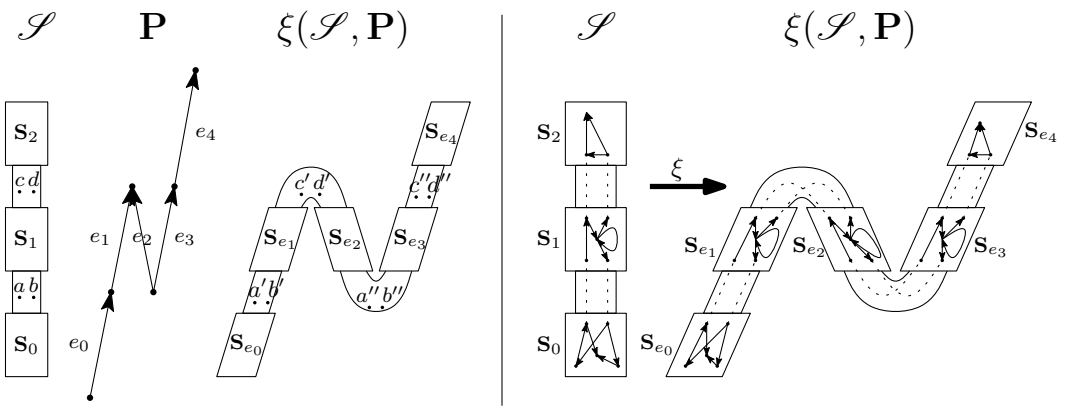


Figure 2 *Left:* Applying a zigzag operator to the (j,k)-path $\mathbf{S}$ with the (j,k)-representation $\mathscr{S} = (\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2)$. Suppose that $S_0 \cap S_1 = \{a, b\}$ and $S_1 \cap S_2 = \{c, d\}$. We demonstrate how $\mathbf{S}_{e_0}$ and $\mathbf{S}_{e_2}$ are obtained. $\mathbf{S}_{e_0}$ is a disjoint copy of $\mathbf{S}_0$ (and the copy of a and b in $\mathbf{S}_{e_0}$ are a' and b', respectively). To obtain $\mathbf{S}_{e_2}$, first make a disjoint copy $\mathbf{S}'_{e_2}$ of $\mathbf{S}_{\text{level}(e_2)} = \mathbf{S}_1$. Set $\ell = \text{level}(e_2) = 1$. Since e_1 is a forward edge and e_2 is a backward edge, $\ell' = \ell + 1 = 2$. Therefore to "glue" $\mathbf{S}'_{e_2}$ to $\mathbf{S}_{e_1}$, we need to look at $S_\ell \cap S_{\ell'} = \{c, d\}$. Assume that the copy of c and d in $\mathbf{S}_{e_1}$ are c' and d', respectively. Furthermore, assume that the copy of c and d in $\mathbf{S}'_{e_2}$ are $\tilde{c}$ and $\tilde{d}$, respectively. To obtain $\mathbf{S}_{e_2}$, we rename $\tilde{c}$ to c', and $\tilde{d}$ to d' in $\mathbf{S}'_{e_2}$. *Right:* A specific example when $\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2$ are the digraphs in the boxes. The dashed lines indicate identification of vertices.

3.2.1 Symmetric Bounded Pathwidth Duality

▶ **Definition 7** ((j,k)-symmetric)**.** Assume that $\mathcal{O}$ is a set of (j,k)-paths. Suppose furthermore that a (j,k)-path representation can be fixed for each structure in $\mathcal{O}$ such that the following holds. For every $\mathbf{S} \in \mathcal{O}$ with representation $\mathscr{S}$ of some length n, and every minimal oriented path $\mathbf{P}$ of height n, it holds that $\xi(\mathscr{S}, \mathbf{P}) \in \mathcal{O}$. Then $\mathcal{O}$ is said to be (j,k)-*symmetric*.

▶ **Definition 8** (SBPD)**.** A structure $\mathbf{B}$ has (j,k)-*symmetric bounded pathwidth duality* ((j,k)-*SBPD*) if there is an obstruction set $\mathcal{O}$ for $\mathbf{B}$ that consists of (j,k)-paths, and in addition, $\mathcal{O}$ is (j,k)-symmetric.

▶ **Theorem 9.** *For a finite structure* $\mathbf{B}$, co-CSP($\mathbf{B}$) *can be defined by a symmetric* (j,k)*-Datalog program if and only if* $\mathbf{B}$ *has* (j,k)*-SBPD.*

To prove Theorem 9, first we prove Lemma 10 using the standard canonical Datalog argument:

▶ **Lemma 10.** *If* CanS($\mathbf{B}$) *accepts a structure* $\mathbf{A}$*, then* $\mathbf{A} \not\to \mathbf{B}$.

The following is the main technical lemma of the section.

▶ **Lemma 11.** *For any* τ*-structures* $\mathbf{A}$ *and* $\mathbf{B}$*, if there exists a structure* $\mathbf{S}$ *with a* (j,k)*-path representation* $\mathscr{S}$ *of some length* n *such that* $\mathbf{S} \to \mathbf{A}$*, and for any minimal oriented path* $\mathbf{P}$ *of height* n*, it holds that* $\xi(\mathscr{S}, \mathbf{P}) \not\to \mathbf{B}$*, then* (j,k)-CanS($\mathbf{B}$) *accepts* $\mathbf{A}$.

Proof of Theorem 9. If CSP($\mathbf{B}$) is defined by a symmetric (j,k)-Datalog program $\mathcal{P}$, then using the symmetric property of $\mathcal{P}$, it is laborious but straightforward to show that

$$\mathcal{O} = \bigcup_{\substack{\mathscr{D} \text{ is a free} \\ \text{derivation of } \mathcal{P}}} \{\text{Ex}(\mathscr{D})\}$$

is a (j,k)-symmetric obstruction set for $\mathbf{B}$.

For the converse, assume that $\mathbf{B}$ has (j,k)-SBPD. Let $\mathcal{O}$ be a symmetric obstruction set of width (j,k) for $\mathbf{B}$. We claim that (j,k)-CanS($\mathbf{B}$) defines CSP($\mathbf{B}$). Assume that $\mathbf{A} \to \mathbf{B}$. Then by Lemma 10, (j,k)-CanS($\mathbf{B}$) does not accept $\mathbf{A}$. Suppose now that $\mathbf{A} \not\to \mathbf{B}$. Then by assumption, there exists a (j,k)-path $\mathbf{S} \in \mathcal{O}$ with a representation $\mathscr{S}$ of length n such that $\mathbf{S} \to \mathbf{A}$. Furthermore, since $\mathcal{O}$ is symmetric, for any minimal oriented path $\mathbf{P}$ of height n, $\xi(\mathscr{S}, \mathbf{P}) \not\to \mathbf{B}$. It follows from Lemma 11 that CanS($\mathbf{B}$) accepts $\mathbf{A}$. ◀

From the above proof it is obvious that:

▶ **Corollary 12** ([8]). *If a symmetric (j,k)-Datalog program defines* CSP($\mathbf{B}$)*, then so does* (j,k)-CanS($\mathbf{B}$).

3.2.2 Piecewise Symmetric Bounded Pathwidth Duality

Piecewise symmetric bounded pathwidth duality (PSBPD) for symmetric Datalog is less stringent than SBPD; however, the price is larger program width. Although the following definitions might seem technical, the general idea is simple: a piecewise symmetric obstruction set $\mathcal{O}$ does not need to contain all (j,k)-paths obtained by zigzagging (j,k)-paths in $\mathcal{O}$ in all possible ways. It is sufficient to zigzag a (j,k)-path $\mathbf{S}$ using only oriented paths which "avoid" certain segments of $\mathbf{S}$: some constants c and d are fixed for $\mathcal{O}$, and there are at most c fixed segments of $\mathbf{S}$ that are avoided by the zigzag operator, each of size at most d. We give the formal definitions.

▶ **Definition 13** ((c,d)-filter). Let $\mathbf{S}$ be a (j,k)-path with a representation $\mathscr{S} = \mathbf{S}_0, \ldots, \mathbf{S}_{n-1}$. A (c,d)-*filter* $\mathscr{F}$ for $\mathscr{S}$ is a set of intervals $\{[s_1,t_1],[s_2,t_2],\ldots,[s_{c'},t_{c'}]\}$ such that

- $c' \leq c$; $0 \leq s_1$; $t_{c'} \leq n-1$; $s_i \leq t_i, \forall i \in [c']$; and $t_\ell + 2 \leq s_{\ell+1}, \forall \ell \in [c'-1]$;
- $|\bigcup_{i \in [s_\ell, t_\ell]} S_i| \leq d, \forall \ell \in [c']$.

Elements of $\mathscr{F}$ are called *delimiters*. An oriented path $\mathbf{P}$ of height n *obeys* a (c,d)-filter $\mathscr{F}$ if for any delimiter $[s_i,t_i] \in \mathscr{F}$, the set of edges e of $\mathbf{P}$ such that $s_i \leq \text{level}(e) \leq t_i$ form a (single) directed path. A demonstration is given in Fig. 3.

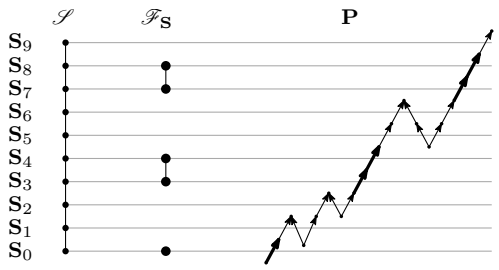


Figure 3 $\mathscr{S}$ is a (j,k)-path representation of $\mathbf{S}$. $\mathscr{F}_\mathbf{S}$ is the $(3,2k)$-filter $\{[0,0][3,4][7,8]\}$ for $\mathscr{S}$. $\mathbf{P}$ is an oriented path that obeys the filter. For example, observe that the edges at levels 3 and 4 form a directed subpath, and that "zigzagging" happens only at those parts of $\mathbf{P}$ that do not fall into the intervals of the filter.

▶ **Definition 14** (Piecewise Symmetric). Assume that $\mathcal{O}$ is a set of (j,k)-paths, and c and d are nonnegative integers. Suppose furthermore that for each $\mathbf{S} \in \mathcal{O}$, there is a (j,k)-path representation $\mathscr{S}$, and a (c,d)-filter $\mathscr{F}_\mathbf{S}$ such that the following holds. For every $\mathbf{S} \in \mathcal{O}$ of some length n, and every minimal oriented path $\mathbf{P}$ of height n that obeys the filter $\mathscr{F}_\mathbf{S}$, it holds that $\xi(\mathscr{S}, \mathbf{P}) \in \mathcal{O}$. Then $\mathcal{O}$ is (j,k,c,d)-*piecewise symmetric*.

Roughly speaking, an oriented path $\mathbf{P}$ is allowed to modulate only those segments of $\mathscr{S}$ which do not correspond to any delimiters in $\mathscr{F}_{\mathbf{S}}$. Compare Definition 14 with Definition 7, and observe that the only difference is that in the piecewise case, the oriented paths must be of a restricted form. Therefore a set that is (j,k)-symmetric is also (j,k,c,d)-piecewise symmetric for any c and d. We simply associate the empty (c,d)-filter with each structure.

▶ **Definition 15** (PSBPD). A structure $\mathbf{B}$ has *(j,k,c,d)-piecewise symmetric bounded pathwidth duality ((j,k,c,d)-PSBPD)* if there is an obstruction set $\mathcal{O}$ for $\mathbf{B}$ that consists of (j,k)-paths, and in addition, $\mathcal{O}$ is (j,k,c,d)-piecewise symmetric.

▶ **Theorem 16.** *For a finite structure $\mathbf{B}$, $\mathbf{B}$ has SBPD (for some parameters) if and only if $\mathbf{B}$ has PSBPD (for some parameters).*

3.3 Applications

3.3.1 Datalog + Maltsev ⇒ symmetric Datalog

Using SBPD, we give a short and simple re-proof of the main result of [8]:

▶ **Theorem 17** ([8]). *Let $\mathbf{B}$ be a finite core structure. If $\mathbf{B}$ is invariant under a Maltsev operation and* co-CSP($\mathbf{B}$) *is definable in Datalog, then* co-CSP($\mathbf{B}$) *is definable in symmetric Datalog (and therefore* CSP($\mathbf{B}$) *is in* L *by [10]).*

We only need to show that if co-CSP($\mathbf{B}$) is in linear Datalog and $\mathbf{B}$ is preserved by a Maltsev operation, then co-CSP($\mathbf{B}$) is in symmetric Datalog. The "jump" from Datalog to linear Datalog essentially follows from already established results, as observed in [8]. Therefore to re-prove Theorem 17, we show the following lemma using an SBPD argument.

▶ **Lemma 18.** *If* co-CSP($\mathbf{B}$) *is definable by a linear Datalog program and $\mathbf{B}$ is invariant under a Maltsev operation m, then* co-CSP($\mathbf{B}$) *is definable by a symmetric Datalog program.*

To get ready for the proof of Lemma 18, we define an N of size s as an oriented path that consists of s forward edges, followed by s backward edges, followed by another s forward edges. Proposition 19 is easy to prove, and the Maltsev properties are used in Lemma 20.

▶ **Proposition 19.** *A minimal oriented path is either a directed path, or it contains a subpath which is an N.*

▶ **Lemma 20.** *Let $\mathbf{B}$ be a structure invariant under a Maltsev operation m, $\mathbf{S}$ be a (j,k)-path with a (j,k)-representation $\mathscr{S} = (\mathbf{S}_0,\dots,\mathbf{S}_{n-1})$, and $\mathbf{P} = e_0,\dots,e_q$ be a minimal oriented path of height n. If $\xi(\mathscr{S},\mathbf{P}) \to \mathbf{B}$, then $\mathbf{S} \to \mathbf{B}$.*

Proof. Using Proposition 19, there is an index t such that $\mathbf{Q} = e_t, e_{t+1},\dots,e_{t+(3s-1)}$ is an N of size s in $\mathbf{P}$. Assume that the first and last vertices of $\mathbf{Q}$ are v and w, respectively. Let $\mathbf{P}'$ be the oriented path obtained from $\mathbf{P}$ by removing $\mathbf{Q}$, and adding a directed path $\mathbf{Q}' = f_t, f_{t+1},\dots,f_{t+(s-1)}$ of length s from v to w. We claim that there is a homomorphism γ from $\xi(\mathscr{S},\mathbf{P}')$ to $\mathbf{B}$. Once this is established, a repetition of this argument sufficiently many times yields that $\mathbf{S} \to \mathbf{B}$ because if we repeatedly "remove" N-s from a minimal oriented path, eventually we must reach a directed path.

Let $\xi(\mathscr{S},\mathbf{P}) = (\mathbf{S}_{e_0},\dots,\mathbf{S}_{e_q})$, and $\varphi_{e_0},\dots,\varphi_{e_q}$ be the corresponding isomorphisms (recall the zigzag operator definition in Section 3.1). Similarly, let $\xi(\mathscr{S},\mathbf{P}') = (\mathbf{S}_{f_0},\dots,\mathbf{S}_{f_{q-2s}})$, and $\psi_{f_0},\dots,\psi_{f_{q-2s}}$ be the corresponding isomorphisms. Because $\mathbf{S}_{[e_0,e_{t-1}]}$ and $\mathbf{S}_{[e_{t+3s},e_q]}$ are isomorphic to $\mathbf{S}_{[f_0,f_{t-1}]}$ and $\mathbf{S}_{[f_{t+s},f_{q-2s}]}$, respectively, γ for elements in $S_{[f_0,f_{t-1}]} \cup S_{[f_{t+s},e_{q-2s}]}$ is defined in the natural way. It remains to define γ for every $d \in S_{[f_t,f_{t+(s-1)}]}$.

Assume that $d \in S_{f_{t+\ell}}$ for some $\ell \in \{0, \dots, s-1\}$. Find the original of d in $\mathbf{S}$ and let it be d_o, i.e. $d_o = \psi_{f_{t+\ell}}(d)$. Then we find the three copies d_1, d_2, d_3 of d_o in $\mathbf{S}_{[f_t, f_{t+(3s-1)}]}$. That is, first we find the three edges $e_{\ell_1}, e_{\ell_2}, e_{\ell_3}$ of $\mathbf{Q}$ which have the same level as $f_{t+\ell}$ (all levels are with respect to $\mathbf{P}$ and $\mathbf{P}'$). Then $d_i = \varphi^{-1}_{e_{\ell_i}}(d_o)$, $i \in [3]$. We define $\gamma(d) = m(d_1, d_2, d_3)$. By the Maltsev properties of m, γ is well-defined. As $\mathbf{B}$ is invariant under m, $\xi(\mathscr{S}, \mathbf{P}') \xrightarrow{\gamma} \mathbf{B}$. ◀

Proof of Lemma 18. If co-CSP($\mathbf{B}$) can be defined by a linear (j,k)-Datalog program, then there is an obstruction set $\mathcal{O}$ for $\mathbf{B}$ in which every structure is a (j,k)-path by [6]. We construct a symmetric obstruction set $\mathcal{O}_{sym}$ for $\mathbf{B}$ as follows. First we define a sequence of sets $\mathcal{O}_1, \mathcal{O}_2, \dots$ inductively, where $\mathcal{O}_1 = \mathcal{O}$. To construct $\mathcal{O}_{i+1}$, for every (j,k)-path $\mathbf{S}$ with a (j,k)-representation $\mathscr{S} = \mathbf{S}_0, \dots, \mathbf{S}_{n-1}$ in $\mathcal{O}_i$, and any minimal oriented path $\mathbf{P}$ of height n, place $\xi(\mathscr{S}, \mathbf{P})$ into $\mathcal{O}_{i+1}$. Set $\mathcal{O}_{sym} = \bigcup_{1 \le i} \mathcal{O}_i$. By construction $\mathcal{O}_{sym}$ is symmetric.

Observe that $\mathcal{O} \subseteq \mathcal{O}_{sym}$, so it remains to show that no element of $\mathcal{O}_{sym}$ maps to $\mathbf{B}$. For contradiction, take an element $\mathbf{T} \in \mathcal{O}_{sym}$ such that $\mathbf{T} \to \mathbf{B}$. By definition of $\mathcal{O}_{sym}$, there is a $\mathbf{T}_0 \in \mathcal{O}$ and a sequence of oriented paths $\mathbf{P}_1, \dots, \mathbf{P}_d$ such that $\mathbf{T}$ is obtained from $\mathbf{T}_0$ as follows. First $\mathbf{T}_1 = \xi(\mathscr{T}_0, \mathbf{P}_1)$ was constructed (and placed into $\mathcal{O}_1$), where $\mathscr{T}_0$ is a (j,k)-path representation of $\mathbf{T}_0$. Then $\mathbf{T}_2 = \xi(\mathscr{T}_1, \mathbf{P}_2)$ was constructed (and placed into $\mathcal{O}_2$), where $\mathscr{T}_1$ is a (j,k)-path representation of $\mathbf{T}_1$, and so on, until $\mathbf{T}_d = \mathbf{T}$ is constructed. We find the largest i such that $\mathbf{T}_i \not\to \mathbf{B}$. Lemma 20 tells us that if $\mathbf{T}_{i+1} \to \mathbf{B}$, then $\mathbf{T}_i \to \mathbf{B}$, a contradiction. ◀

3.3.2 A class of oriented paths for which the CSP is in L, and a class for which the CSP is NL-complete

In this section we define a class $\mathcal{C}$ of oriented paths such that if $\mathbf{B} \in \mathcal{C}$ then co-CSP($\mathbf{B}$) is in symmetric Datalog. Our strategy is to find an obstruction set $\mathcal{O}$ for $\mathbf{B} \in \mathcal{C}$, and then to show that our obstruction set is piecewise symmetric. We need some notation.

We say that a directed path is *forward* to mean that its first and last vertices are the vertices with indegree zero and outdegree zero, respectively. Let $\mathbf{P}$ be an oriented path with first vertex v and last vertex w. Then the *reverse* of $\mathbf{P}$, denoted with $\bar{\mathbf{P}}$, is a copy of the oriented path $\mathbf{P}$ in the reverse direction, i.e. the first vertex of $\bar{\mathbf{P}}$ is a copy of w and its last vertex is a copy of v. Let $\mathbf{Q}$ be another oriented path. The *concatenation* of $\mathbf{P}$ and $\mathbf{Q}$ is the oriented path $\mathbf{PQ}$ in which the last vertex of $\mathbf{P}$ is identified with the first vertex of $\mathbf{Q}$. For a nonnegative integer r, $\mathbf{P}^r$ denotes $\mathbf{P}_1\mathbf{P}_2 \dots \mathbf{P}_r$, where the $\mathbf{P}_\ell$ are disjoint copies of $\mathbf{P}$. Given two vertices v and w, we denote the presence of an edge from v to w with $v \to w$.

▶ **Definition 21** (Wave)**.** If an oriented path $\mathbf{Q}$ can be expressed as $\mathbf{E}_1(\mathbf{P}\bar{\mathbf{P}})^r\mathbf{P}\mathbf{E}_2$, where $\mathbf{E}_i$ ($i \in [2]$) denotes the forward directed path that is a single edge, $\mathbf{P}$ is a forward directed path of length ℓ, and $r \ge 0$, then $\mathbf{Q}$ is called an *ℓ-wave*. A 2-wave is shown in Fig. 4, *1.*

▶ **Theorem 22.** *Let* $\mathbf{Q}$ *be a wave. Then* $\mathbf{Q}$ *has PSBPD,* co-CSP($\mathbf{Q}$) *is definable in symmetric Datalog, and* CSP($\mathbf{Q}$) *is in* L.

We state the following generalization of waves.

▶ **Definition 23** (Staircase)**.** A *monotone wave* is an oriented path of the form $(\bar{\mathbf{P}}\mathbf{P})^r\bar{\mathbf{P}}$, where $\mathbf{P}$ is a forward directed path and $r \ge 0$. We call the vertices of a monotone wave in the topmost level *peaks*, and the vertices in the bottommost level *troughs*.

If a minimal oriented path $\mathbf{Q}$ can be expressed as $\mathbf{P}_1\mathbf{W}_1\mathbf{P}_2\mathbf{W}_2 \dots \mathbf{P}_{n-1}\mathbf{W}_{n-1}\mathbf{P}_n$, where $\mathbf{P}_1, \dots, \mathbf{P}_n$ are forward directed paths, $\mathbf{W}_1, \dots, \mathbf{W}_{n-1}$ are monotone waves, and for any $i \in [n-1]$, the troughs of $\mathbf{W}_i$ are in a level strictly below the level of the troughs of $\mathbf{W}_{i+1}$,

and also, the peaks of $\mathbf{W}_i$ are in a level strictly below the level of the peaks of $\mathbf{W}_{i+1}$, then $\mathbf{Q}$ is called a *staircase*. An example is given in Fig. 4, *2*.

▶ **Theorem 24.** *Let* $\mathbf{Q}$ *be a staircase. Then* $\mathbf{Q}$ *has PSBPD,* co-CSP($\mathbf{Q}$) *is definable in symmetric Datalog, and* CSP($\mathbf{Q}$) *is in* L.

We also give a large class of oriented paths for which the CSP is NL-complete.

▶ **Theorem 25.** *Let* $\mathbf{B}$ *be a core oriented path that contains a subpath* $\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$ *of some height* h *with the following properties:* $\mathbf{P}_1, \mathbf{P}_2$ *and* $\mathbf{P}_3$ *are minimal oriented paths, they all have height* h*, and there is a minimal oriented path* $\mathbf{Q}$ *of height* h *such that* $\mathbf{Q} \to \mathbf{P}_1$, $\mathbf{Q} \to \mathbf{P}_3$ *but* $\mathbf{Q} \not\to \mathbf{P}_2$*. Then* CSP($\mathbf{B}$) *is* NL*-complete.*

An example is given in Fig. 4, *3* and *4*.

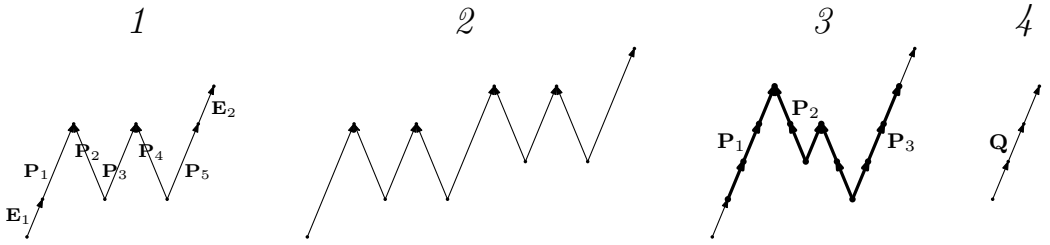


Figure 4 *1:* A 2-wave. *2:* A staircase. *3:* An example oriented path for which the CSP is NL-complete. *4:* The oriented path $\mathbf{Q}$ in Theorem 25 corresponding to the oriented path in *3*.

4 On CSPs in NL

4.1 Preliminaries and Definitions

Let τ be a vocabulary. A *successor* τ*-structure* $\mathbf{S}$ is a relational structure with vocabulary $\tau \cup \{\mathfrak{first}, \mathfrak{last}, \mathfrak{suc}\}$, where $\mathfrak{first}$ and $\mathfrak{last}$ are unary symbols and $\mathfrak{suc}$ is a binary symbol. The domain S is defined as $\{1, \ldots, n\}$, $\mathfrak{first}^{\mathbf{S}} = \{1\}$, $\mathfrak{last}^{\mathbf{S}} = \{n\}$, and $\mathfrak{suc}^{\mathbf{S}}$ contains all pairs $(i, i+1)$, $i \in [n-1]$. Because $\mathfrak{first}^{\mathbf{S}}, \mathfrak{last}^{\mathbf{S}}$ and $\mathfrak{suc}^{\mathbf{S}}$ depend only on n, they are called *built-in* relations. When we say that a class of successor structures is homomorphism/isomorphism-closed, all structures under consideration are successor structures, and we understand that homomorphism/isomorphism closure, respectively, is required only for non-built-in relations.

▶ **Definition 26** (Split Operation)**.** A *split operation* produces a τ-structure $\mathbf{A}'$ from a τ-structure $\mathbf{A}$ as follows. For an element $a \in A$ let T_a be defined as

$$T_a = \left\{(\mathbf{t}, R, i) \mid \mathbf{t} = (t_1, \ldots, t_r) \in R^{\mathbf{A}} \text{ where } R \in \tau, \text{ and } t_i = a\right\}.$$

If $|T_a| \leq 1$, no split operation can be applied. Otherwise we choose a strict nonempty subset T of T_a, and for each triple $(\mathbf{t}, R, i) \in T$, we replace $\mathbf{t} = (t_1, \ldots, t_r)$ in $R^{\mathbf{A}}$ with $(t_1, \ldots, t_{i-1}, a', t_{i+1}, \ldots, t_r)$ to obtain $\mathbf{A}'$ (and $A' = A \cup \{a'\}$).

▶ **Definition 27** (Split-Minimal, Critical)**.** Let $\mathcal{C}$ be a class of structures over the same vocabulary. We say that a structure $\mathbf{A} \in \mathcal{C}$ is *split-minimal in* $\mathcal{C}$ if for every possible nonempty sequence of split operations applied to $\mathbf{A}$, the resulting structure is not in $\mathcal{C}$. We say that a structure $\mathbf{A} \in \mathcal{C}$ is *critical in* $\mathcal{C}$ if no proper substructure of $\mathbf{A}$ is in $\mathcal{C}$.

For a class of isomorphism-closed successor τ-structures, criticality and split-minimality is meant only with respect to the non-built-in relations.

▶ **Definition 28** (Read-Once Datalog)**.** Let $\mathcal{P}$ be a (linear, symmetric) Datalog program that defines a class of structures $\mathcal{C}$. If for every critical and split-minimal element of $\mathcal{C}$ there is a $\mathcal{P}$-derivation that is read-once, then we say that $\mathcal{P}$ is *read-once*.

▶ **Definition 29** (Read-Once mnBP1)**.** A *monotone nondeterministic branching program* (mnBP) H with variables $X = \{x_1, \ldots, x_n\}$ computes a Boolean function $f_H : \{0,1\}^n \to \{0,1\}$. H is a directed graph with distinguished nodes s and t and some arcs labeled with variables from X (not all arcs must be labeled). An assignment σ to the variables in X defines in a natural way a subgraph H_σ of H. The function f_H is defined as $f_H(\sigma) = 1$ iff H_σ has a directed path from s to t (an *accepting path*). The size of an mnBP is $|V_H|$.

Let $\mathcal{F}$ be a poly-size family of mnBP1s (mnBP1(poly)) that defines a class of structures $\mathcal{C}$ over a vocabulary τ. (The encoding is done in the straightforward manner, i.e. there is a variable for every possible $(R, \mathbf{t})$ where $R \in \tau$ and $\mathbf{t}$ is a tuple.) If for every structure in $\mathcal{C}$ there is an accepting path that queries every variable at most once, then we say that $\mathcal{F}$ is *read-once*. (This read-once condition can be made a bit weaker.)

We give some examples of problems definable by a 1-linDat($\mathfrak{suc}$) program or by an mnBP1(poly). The program in Section 2.2, Fig. 1 without rule 3 is a read-once linear Datalog($\mathfrak{suc}$) program that defines the problem directed *st*-Conn. To see that this program $\mathcal{P}_{st-\text{Conn}}$ is read-once, let $\mathbf{G}$ be any input that is accepted (we do not even need $\mathbf{G}$ to be critical and split-minimal). Then we find a directed path in $E^{\mathbf{G}}$ connecting an element of $S^{\mathbf{G}}$ to an element of $T^{\mathbf{G}}$ without repeated edges. We build a $\mathcal{P}_{st-\text{Conn}}$-derivation for this path in the obvious way.

Let EvenCliques be the class of undirected graphs which are cliques of even size. A bit of work shows that EvenCliques can be defined with a 1-linDat($\mathfrak{suc}$) program. In fact, we can easily test much more complicated arithmetic properties than the property of being even (e.g. being a power of k) with a 1-linDat($\mathfrak{suc}$) program. We note that EvenCliques or "cliques with any domain size property" cannot be defined by a linear Datalog program because a (nontrivial) set of cliques is never closed under homomorphisms. Since a 1-linDat($\mathfrak{suc}$) program can be converted into an mnBP1(poly), the aforementioned problems can also be defined with an mnBP1(poly).

4.2 Main Results

We simply state the results for 1-linDat($\mathfrak{suc}$) and poly-size families of mnBP1s discussed in the Introduction.

▶ **Theorem 30.** *Let $\mathcal{C}$ be a homomorphism-closed class of successor τ-structures. If $\mathcal{C}$ can be defined by a 1-linDat($\mathfrak{suc}$) program of width (j,k), then every critical and split-minimal element of $\mathcal{C}$ has a $(j, k+j)$-path-decomposition.*

▶ **Corollary 31.** *If* co-CSP($\mathbf{B}$) *can be defined by a 1-linDat($\mathfrak{suc}$) program of width (j,k), then* co-CSP($\mathbf{B}$) *can also be defined by a linear $(j, k+j)$-Datalog program.*

▶ **Theorem 32.** *Let $\mathcal{C}$ be a homomorphism-closed class of successor τ-structures. If $\mathcal{C}$ can be defined by a family of mnBP1s of size $O(n^j)$, then every critical and split-minimal element of $\mathcal{C}$ has a $(j, r+j)$-path-decomposition, where r is the maximum arity of the symbols in τ.*

▶ **Corollary 33.** *If* co-CSP($\mathbf{B}$) *can be defined by a family of mnBP1s of size $O(n^j)$, then* co-CSP($\mathbf{B}$) *can also be defined by a linear $(j, r+j)$-Datalog program, where r is the maximum arity of the relation symbols in the vocabulary of $\mathbf{B}$.*

As discussed before, a wide class of CSPs–CSPs whose associated *variety admits the unary, affine or semilattice types*–does not have bounded pathwidth duality [16]. It follows that all these CSPs are not definable by any 1-linDat($\mathfrak{suc}$) program, or with any mnBP1 of poly-size. An example of such a CSP is the P-complete CSP Horn-3Sat.

References

1. F. Afrati and S. S. Cosmadakis. Expressiveness of restricted recursive queries. In *Proceedings of STOC*, pages 113–126, 1989.
2. Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. Syst. Sci.*, 75(4):245–254, 2009.
3. L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of The 50th Annual Symposium on Foundations of Computer Science (FOCS)*, 2009.
4. A. Bulatov, A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *LNCS Surveys on Complexity of Constraints*, volume 5250, pages 93–124. 2008.
5. C. Carvalho, L. Egri, M. Jackson, and T. Niven. On Maltsev digraphs. In *Proceedings of the 6th International Computer Science Symposium in Russia*, 2011. To appear.
6. Víctor Dalmau. Constraint satisfaction problems in non-deterministic logarithmic space. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 414–425. Springer-Verlag, 2002.
7. Víctor Dalmau and Andrei Krokhin. Majority constraints have bounded pathwidth duality. *Eur. J. Comb.*, 29(4):821–837, 2008.
8. Víctor Dalmau and Benoit Larose. Maltsev + Datalog → symmetric Datalog. In *LICS*, pages 297–306, 2008.
9. László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems (Special Issue on STACS 2010)*. To appear.
10. László Egri, Benoît Larose, and Pascal Tesson. Symmetric Datalog and constraint satisfaction problems in logspace. In *LICS*, pages 193–202, 2007.
11. Tomás Feder. Classification of homomorphisms to oriented cycles and of k-partite satisfiability. *SIAM J. Discrete Math.*, 14(4):471–480, 2001.
12. Tomas Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1999.
13. Erich Grädel. Capturing complexity classes by fragments of second-order logic. *Theor. Comput. Sci.*, 101(1):35–57, 1992.
14. D. Hobby and R.N. McKenzie. *The Structure of Finite Algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, R.I., 1988.
15. Neil Immerman. *Descriptive complexity*. Graduate Texts in Computer Science. Springer, 1999.
16. Benoit Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009.
17. Benoit Larose and László Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3-4):439–466, 2007.
18. Leonid Libkin. *Elements of finite model theory*. Springer, 2004.
19. T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of STOC*, pages 216–226, 1978.
20. Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.