




The Expressive Power of Revised Datalog on Problems with Closure Properties

Shiguang Feng^(✉) 

School of Computer Science and Engineering, Sun Yat-sen University,
Guangzhou 510006, China
fengshg3@mail.sysu.edu.cn

Abstract. In this paper, we study the expressive power of revised Datalog on the problems that are closed under substructures. We show that revised Datalog cannot define all the problems that are in PTIME and closed under substructures. As a corollary, LFP cannot define all the extension-closed problems that are in PTIME.

Keywords: Datalog · preservation theorem · closure property · expressive power

1 Introduction

Datalog and its variants are widely used in artificial intelligence and other fields, such as deductive database, knowledge representation, data integration, cloud computing, etc [8, 11, 13, 18, 19, 23]. As a declarative programming language, it is often used to perform data analysis and create complex queries. The complexity and expressive power is an important issue of the study [1, 5, 19, 21, 22]. With the recursive computing ability, Datalog is more powerful than first-order logic. It defines exactly the polynomial time computable queries on ordered finite structures [9]. Hence, Datalog captures the complexity class PTIME on ordered finite structures. While on all finite structures, the expressive power of Datalog is very limited. It even cannot define the parity of a set [9]. A Datalog program is constituted of a set of Horn clauses. The characteristics of syntax determine the monotonicity properties of its semantics. That is, every Datalog (resp., positive Datalog, the fragment of Datalog where no negated atomic formula occurs in the body of any clauses) definable query is preserved under extensions [3] (resp., homomorphisms [4]). It is natural to ask from the point of view of descriptive complexity that whether Datalog (resp., positive Datalog) captures the polynomial time computable problems that are closed under extensions (resp., homomorphisms). The answer is negative by the work of Afrati et al. who showed that positive Datalog cannot express all monotone queries computable in polynomial time, and the perfect squares problem that is in polynomial time and closed under extensions is not expressible in Datalog [3].

In model theory, many preservation theorems are proved to show the relationship between the closure properties and the syntactic properties of formulas.

Most of these preservation theorems fail when restricted to finite structures. A lot of research about the preservation theorems on Datalog, first-order logic (FO) and least fixpoint logic (LFP) have been conducted on finite structures. Ajtai and Gurevich showed that a positive Datalog formula is bounded iff it is definable in positive existential first-order logic, and every first-order logic expressible positive Datalog formula is bounded [4], where a Datalog formula is bounded if there exists a number n such that the fixpoint of the formula can be reached for any finite structure within n steps. Dawar and Kreutzer showed that the homomorphism preservation theorem fails for LFP, both in general and in restriction to finite structures [6]. That is, there is an LFP formula that is preserved under homomorphisms (in the finite) but is not equivalent (in the finite) to a Datalog formula. The paper [16] studied Datalog with negation and monotonicity, and the expressive power with respect to monotone and homomorphism properties.

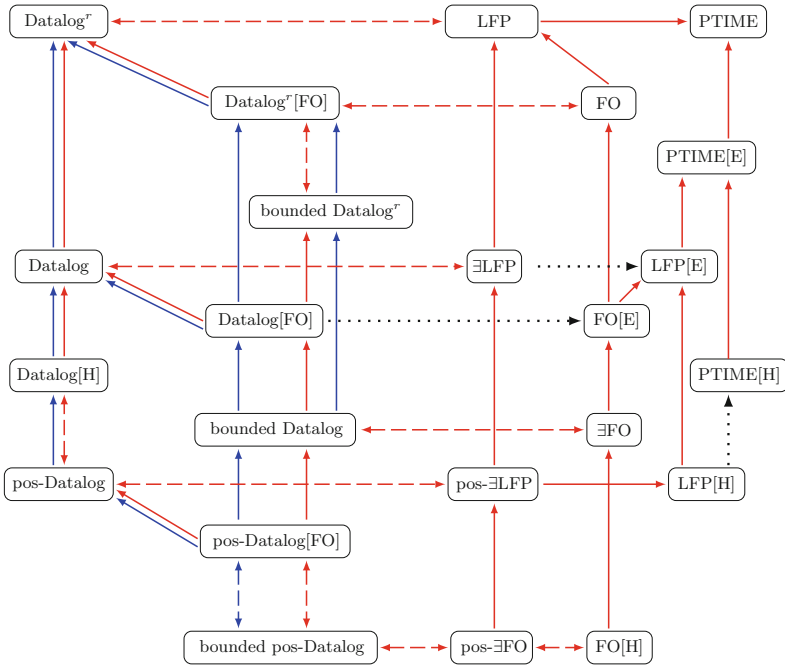


Fig. 1. The relationship of FO, LFP, PTIME, Datalog and its variants. Datalog^r denotes revised Datalog. $\text{pos-}\mathcal{L}$ denotes the positive fragment of \mathcal{L} . $\exists\mathcal{L}$ denotes the existential fragment of \mathcal{L} . $\mathcal{L}[\text{FO}]$ denotes the set of \mathcal{L} formulas that are first-order definable. $\mathcal{L}[\text{H}]$ (resp., $\mathcal{L}[\text{E}]$) denotes the set of \mathcal{L} formulas (or problems computable in \mathcal{L}) that are preserved under homomorphisms (resp., extensions). The blue arrow shows the containment relationship on Datalog and its variants. The red arrow shows the relationship about the expressive power. The solid arrow implies that the relationship is strict, and the dashed bidirectional arrow implies the equality relationship. The black dotted arrow means whether the relationship is strict is still open.

The papers [7, 20] studied the preservation results under extensions for FO and Datalog. All the results are summarized in Fig. 1.

Revised Datalog (Datalog^r) is an extension of Datalog, where universal quantification over intensional relations is allowed in the body of rules. Abiteboul and Vianu first introduced the idea that the body of a rule in Datalog can be universally quantified [2]. The author of the paper showed that Datalog^r equals LFP on all finite structures [10]. In the paper, we study the expressive power of Datalog^r on problems with closure properties, i.e., closed under substructures (or extensions). We conclude that a Datalog^r formula is equivalent to a first-order formula iff it is equivalent to a bounded Datalog^r formula. As the main result of the paper, we show that Datalog^r cannot define all the problems that are in PTIME and closed under substructures. Since Datalog^r equals LFP, and the complement of a substructure-closed problem is extension-closed, as a corollary, LFP cannot define all the extension-closed problems that are in PTIME. This result contributes the strict containment $LFP[E] \subsetneq PTIME[E]$ in Fig. 1. A technique of tree encodings for arbitrary structures is used in the proof. For an arbitrary set of structures $\mathcal{K} \in EXPTIME$, we can encode them into a set of substructure-closed structures \mathcal{K}' , where the tree used to encode the structure in \mathcal{K} is exponentially larger. Therefore, \mathcal{K}' is in PTIME. For every structure in \mathcal{K}' , there is a characteristic structure $S_{\mathbf{T}}$ of it such that they are equivalent with respect to Datalog^r-transformations. Since $S_{\mathbf{T}}$ can be computed from the structure in \mathcal{K} in logspace, this implies that \mathcal{K} is also in PTIME, contrary to the time hierarchy theorem. Figure 2 shows the sketch of the proof.

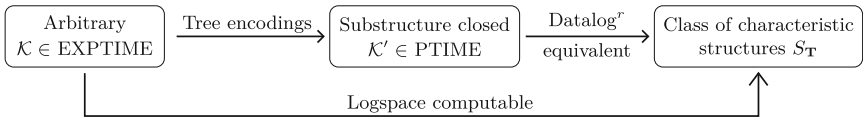


Fig. 2. The idea of the proof for the nondefinability of Datalog^r.

The paper is organized as follows: In Sect. 2, we give the basic definitions and notations. In Sect. 3, we recall invariant relations on perfect binary trees, and introduce the technique of tree encodings for arbitrary structures. And we prove the nondefinability results for Datalog^r on substructure-closed problems. Finally, we conclude the paper in Sect. 4.

2 Preliminaries

Let $\tau = \{\mathbf{c}_1, \dots, \mathbf{c}_m, P_1, \dots, P_n\}$ be a vocabulary, where $\mathbf{c}_1, \dots, \mathbf{c}_m$ are constant symbols and P_1, \dots, P_n are relation symbols. A τ -structure is a tuple $\mathbf{A} = \langle A, \mathbf{c}_1^A, \dots, \mathbf{c}_m^A, P_1^A, \dots, P_n^A \rangle$ where A is the domain, and $\mathbf{c}_1^A, \dots, \mathbf{c}_m^A, P_1^A, \dots, P_n^A$ are interpretations of the constant and relation symbols over A , respectively. We assume the equality relation “=” is contained in every vocabulary, and omit the

superscript “ A ” when it is clear from context. We call \mathbf{A} *finite* if its domain A is a finite set. Unless otherwise stated, all structures considered in this paper are finite. We use $\text{arity}(R)$ to denote the arity of a relation R , and use “ $|$ ” to indicate the cardinality of a set or the arity of a tuple, e.g., $|A|$ denotes the cardinality of A and $|(x_1, x_2, x_3)| = 3$. A finite structure is *ordered* if it is equipped with a linear order relation “ \leq ”, and the successor relation “SUCC”, the constants “**min**” and “**max**” for the minimal and maximal elements, respectively, with respect to “ \leq ”. Let $\mathbf{A} = \langle A, \mathbf{c}_1^A, \dots, \mathbf{c}_m^A, P_1^A, \dots, P_n^A \rangle$ and $\mathbf{B} = \langle B, \mathbf{c}_1^B, \dots, \mathbf{c}_m^B, P_1^B, \dots, P_n^B \rangle$ be two structures. If $B \subseteq A$, $\mathbf{c}_i^A = \mathbf{c}_i^B$ ($1 \leq i \leq m$), and $P_j^B = P_j^A \cap B^{\text{arity}(P_j)}$ ($1 \leq j \leq n$), then we say that \mathbf{B} is a *substructure* of \mathbf{A} , and \mathbf{A} is an *extension* of \mathbf{B} .

An r -ary *global relation* R of a vocabulary τ is a mapping that assigns to every τ -structure \mathbf{A} an r -ary relation R^A over A such that for every isomorphism $\pi : \mathbf{A} \simeq \mathbf{B}$ and every $a_1, \dots, a_r \in A$, $\mathbf{A} \models R^A a_1 \dots a_r$ iff $\mathbf{B} \models R^B \pi(a_1) \dots \pi(a_r)$. A *query* is a global relation. We say that a query \mathcal{Q} is expressible in a logic \mathcal{L} if there is an \mathcal{L} -formula that defines \mathcal{Q} . Two formulas are *equivalent* if they define the same query. Given two logics \mathcal{L}_1 and \mathcal{L}_2 , we use $\mathcal{L}_1 \leq \mathcal{L}_2$ to denote that every \mathcal{L}_1 -formula is equivalent to an \mathcal{L}_2 -formula. If $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_2 \leq \mathcal{L}_1$, then we denote it by $\mathcal{L}_1 \equiv \mathcal{L}_2$.

Suppose that a relation symbol X occurs positively in $\varphi(\bar{x})$ and $|\bar{x}| = \text{arity}(X)$. Given a structure \mathbf{A} , we can define a monotonic sequence X_0, X_1, X_2, \dots , where $X_0 = \emptyset$ and $X_{i+1} = \{\bar{a} \mid (\mathbf{A}, X_i) \models \varphi[\bar{a}]\}$ for $i \geq 0$, such that $X_i \subseteq X_{i+1}$. Since \mathbf{A} is finite, the sequence will eventually reach a fixpoint.

Definition 1. *The least fixpoint logic LFP is an extension of first-order logic by adding the following rule [9]:*

- If φ is an LFP formula, X occurs positively in φ , and $|\bar{x}| = |\bar{u}| = \text{arity}(X)$, then $[\text{LFP}_{\bar{x}, X} \varphi] \bar{u}$ is an LFP formula.

Given an LFP formula $[\text{LFP}_{\bar{x}, X} \varphi] \bar{u}$, for any structure \mathbf{A} and $\bar{a} \in A^{\text{arity}(\bar{a})}$, we have $\mathbf{A} \models [\text{LFP}_{\bar{x}, X} \varphi] \bar{a}$ iff \bar{a} is in the fixpoint of the sequence induced by X and φ on \mathbf{A} .

Proposition 1. [15, 24] *LFP captures PTIME on ordered finite structures.*

Definition 2. *Let τ be a vocabulary. A Datalog program Π over τ is a finite set of rules of the form*

$$\beta \leftarrow \alpha_1, \dots, \alpha_l$$

where $l \geq 0$ and

- (1) each α_i is either an atomic formula or a negated atomic formula,
- (2) β is an atomic formula $R\bar{x}$, where R doesn't occur negatively in any rule of Π .

β is the head of the rule and the sequence $\alpha_1, \dots, \alpha_l$ constitute the body. Every relation symbol occurring in the head of some rule of Π is intensional, and the other symbols in τ are extensional. We use $(\tau, \Pi)_{\text{int}}$ and $(\tau, \Pi)_{\text{ext}}$ to denote the set of intensional and extensional symbols, respectively. We also allow 0-ary relation symbols. If Q is a 0-ary relation, its value is from $\{\emptyset, \{\emptyset\}\}$. $Q = \emptyset$ means that Q is FALSE and $Q = \{\emptyset\}$ means that Q is TRUE. We use the least fixpoint semantics for Datalog programs. A Datalog formula has the form $(\Pi, P)\bar{x}$, where P is an r -ary intensional relation symbol and $\bar{x} = (x_1, \dots, x_r)$ are variables that do not occur in Π . For a $(\tau, \Pi)_{\text{ext}}$ -structure \mathbf{A} and $\bar{a} = (a_1, \dots, a_r) \in A^r$,

$$\mathbf{A} \models (\Pi, P)\bar{x}[\bar{a}] \text{ iff } (a_1, \dots, a_r) \in P_{(\infty)},$$

where $P_{(\infty)}$ is the least fixpoint for relation P when Π is evaluated on \mathbf{A} . If P is 0-ary, then $\mathbf{A} \models (\Pi, P)$ iff $P_{(\infty)} = \{\emptyset\}$.

3 Datalog^r on Problems with Closure Properties

3.1 Revised Datalog Programs

Definition 3. In Definition 2, if we replace Condition (1) by

(1') each α_i is either an atomic formula, or a negated atomic formula, or a formula $\forall \bar{y} R\bar{y}\bar{z}$, where R occurs in the head of some rule,

then we call this logic program revised Datalog program, denoted by Datalog^r.

Example 1. Let $G = \langle V, E \rangle$ be a directed acyclic graph, and the set of nodes V partitioned into two disjoint sets V_{uni} and V_{exi} . The nodes in V_{uni} (resp., V_{exi}) are universal (resp., existential). The notion of alternating path is defined recursively. There is an alternating path from \mathbf{s} to \mathbf{t} in G if

- $\mathbf{s} = \mathbf{t}$; or
- $\mathbf{s} \in V_{\text{exi}}$, $\exists x \in V$ such that $(\mathbf{s}, x) \in E$ and there is an alternating path from x to \mathbf{t} ; or
- $\mathbf{s} \in V_{\text{uni}}$, $\exists x \in V$ such that $(\mathbf{s}, x) \in E$, and $\forall y \in V$, if $(\mathbf{s}, y) \in E$ then there is an alternating path from y to \mathbf{t} .

The alternating graph accessibility problem is defined as follows:

Input: A directed acyclic graph $G = \langle V_{\text{uni}} \cup V_{\text{exi}}, E \rangle$ and two nodes \mathbf{s}, \mathbf{t} .

Output: Yes if there is an alternating path from \mathbf{s} to \mathbf{t} in G , otherwise no.

This problem is P-complete [14]. The following Datalog^r program Π defines the alternating graph accessibility problem

$$\begin{array}{ll} P_{\text{alt}}xy \leftarrow x = y; & Qxzy \leftarrow P_{\text{uni}}x, Exz, P_{\text{alt}}zy; \\ P_{\text{alt}}xy \leftarrow \neg V_{\text{uni}}x, Exz, P_{\text{alt}}zy; & P_{\text{alt}}xy \leftarrow P_{\text{uni}}x, \forall z Qxzy; \\ P_{\text{uni}}x \leftarrow V_{\text{uni}}x, Exy; & P \leftarrow P_{\text{alt}}\mathbf{st}. \\ Qxzy \leftarrow P_{\text{uni}}x, \neg Exz; & \end{array}$$

We have $(\tau, \Pi)_{\text{int}} = \{P_{\text{alt}}, Q, P_{\text{uni}}, P\}$ and $(\tau, \Pi)_{\text{ext}} = \{E, V_{\text{uni}}, \mathbf{s}, \mathbf{t}\}$. The relation P_{uni} saves the nodes in V_{uni} that have a successor. The relation P_{alt} saves the pairs (x, y) such that there is an alternating path from x to y . We use $Qxzy$ to denote that for any $x \in P_{\text{uni}}$, either there is no edge from x to z , or there is an alternating path from z to y . For any directed acyclic $(\tau, \Pi)_{\text{ext}}$ -structure \mathbf{A} , we have $\mathbf{A} \models (\Pi, P)$ iff there is an alternating path from \mathbf{s} to \mathbf{t} .

The Datalog formulas are preserved under extensions [7], i.e., if a structure \mathbf{B} satisfies a Datalog formula φ and \mathbf{A} is an extension of \mathbf{B} , then \mathbf{A} also satisfies φ . A directed acyclic graph with an alternating path from \mathbf{s} to \mathbf{t} can be extended to a directed acyclic graph without any alternating path from \mathbf{s} to \mathbf{t} by adding new nodes. So Datalog cannot define the alternating graph accessibility problem, which implies that Datalog^r is strictly more expressive than Datalog. Allowing universal quantification over intensional relations is essential for Datalog^r to increase its expressive power. With the help of it, every $\text{FO}(\text{LFP})$ formula can be transformed into an equivalent Datalog^r formula.

Proposition 2. [10] $\text{Datalog}^r \equiv \text{LFP}$ on all finite structures.

A Datalog program is *positive* if no negated atomic formula occurs in the body of any rule. A Datalog formula $(\Pi, P)\bar{t}$ is *bounded* if there is an $n \geq 0$ such that $P_{(n)} = P_{(\infty)}$ for all structures. A bounded (positive) Datalog formula is equivalent to an existential (positive) first-order formula, and vice versa [9]. Furthermore, a positive Datalog formula is bounded iff it is equivalent to a first-order formula. The statement is false for all Datalog formulas. There is an unbounded Datalog formula that is equivalent to an FO formula, but no bounded Datalog formula is equivalent to it [4]. Unlike Datalog, if an unbounded Datalog^r formula is equivalent to an FO formula, then it must be equivalent to a bounded Datalog^r formula.

Proposition 3. A Datalog^r formula is equivalent to a first-order formula iff it is equivalent to a bounded Datalog^r formula.

Proof. Suppose that a Datalog^r formula is equivalent to a first-order formula φ . Using the method in [10] we can construct a bounded Datalog^r formula that is equivalent to φ . For the other direction, the proof in [9] which shows that every bounded Datalog formula is equivalent to an FO formula remains valid for bounded Datalog^r formulas.

3.2 Invariant Relations on Perfect Binary Trees

In [17], Lindell introduced invariant relations that are defined on perfect binary trees, and showed that there are queries computable in PTIME but not definable in LFP. A perfect binary tree is a binary tree in which all internal nodes have two children and all leaf nodes are in the same level. Let $T = \langle V, E, \mathbf{root} \rangle$ be a perfect binary tree, where V is the set of nodes, E is the set of edges and \mathbf{root} is the root node. Suppose that R is an r -ary relation on V and f is an automorphism

of T . Given a tuple $\bar{a} = (a_1, \dots, a_r) \in R$, we write $f(\bar{a}) = (f(a_1), \dots, f(a_r))$ and $f[R] = \{(f(a_1), \dots, f(a_r)) \mid (a_1, \dots, a_r) \in R\}$. We say that R is an invariant relation if for every automorphism f , $R = f[R]$. It is easily seen that the equality $=$ and E are invariant relations.

First we give several technical lemmas. The proofs of Lemmas 1, 2, 3, and 5 can be found in the full arXiv version of the paper.

Lemma 1. *If R_1 and R_2 are r -ary invariant relations, then $\neg R_1$, $R_1 \cap R_2$ and $R_1 \cup R_2$ are also invariant relations.*

Lemma 2. *Suppose that R is an r -ary invariant relation, R' is a k -ary invariant relation and g is a permutation of $\{1, \dots, r\}$. Define*

$$R_1 = \{(a_{g(1)}, \dots, a_{g(r)}) \mid (a_1, \dots, a_r) \in R\},$$

$$R_2 = \{(a_1, \dots, a_r, b_1, \dots, b_k) \mid (a_1, \dots, a_r) \in R \text{ and } (b_1, \dots, b_k) \in R'\}.$$

Then R_1 and R_2 are also invariant relations.

Lemma 3. *Suppose that R is a $(k + r)$ -ary invariant relation. Define*

$$R_1 = \{(a_1, \dots, a_r) \mid (b_1, \dots, b_k, a_1, \dots, a_r) \in R \text{ for all nodes } b_1, \dots, b_k\}$$

$$R_2 = \{(a_1, \dots, a_r) \mid \exists b_1, \dots, b_k \text{ such that } (b_1, \dots, b_k, a_1, \dots, a_r) \in R\}.$$

Then R_1 and R_2 are also invariant relations.

Let a, b be two nodes of a perfect binary tree T , we use $a \bar{\wedge} b$ and $d(a)$ to denote the least common ancestor of a, b and the depth of a , respectively. For example, in Fig. 3 there is a perfect binary tree in which $d(\mathbf{root}) = 0$, $d(a) = 1$, $d(c) = d(e) = 2$, and $c \bar{\wedge} e = \mathbf{root}$.

Let (a_1, \dots, a_r) be an r -ary tuple of nodes, its characteristic tuple is defined as

$$(a_1, \dots, a_r)^* = (d(a_1), d(a_1 \bar{\wedge} a_2), \dots, d(a_1 \bar{\wedge} a_r),$$

$$d(a_2), d(a_2 \bar{\wedge} a_3), \dots, d(a_2 \bar{\wedge} a_r),$$

$$\dots, d(a_r))$$

which is a $\frac{r(r+1)}{2}$ -ary tuple of numbers. Let R be an invariant relation, the characteristic relation of R is defined to be

$$R^* = \{(a_1, \dots, a_r)^* \mid (a_1, \dots, a_r) \in R\}.$$

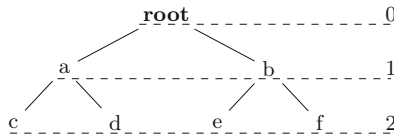


Fig. 3. A perfect binary tree of depth 3.

Proposition 4. [17] Let $\bar{a} = (a_1, \dots, a_r)$ and $\bar{b} = (b_1, \dots, b_r)$ be two tuples, and R an r -ary invariant relation of a perfect binary tree T .

- $(a_1, \dots, a_r)^* = (b_1, \dots, b_r)^*$ iff there is an automorphism f of T such that $f(\bar{a}) = \bar{b}$.
- If $(a_1, \dots, a_r)^* = (b_1, \dots, b_r)^*$, then $\bar{a} \in R$ iff $\bar{b} \in R$.

For any two invariant relations R_1 and R_2 , $R_1 = R_2$ iff $R_1^* = R_2^*$.

3.3 Tree Encodings and Characteristic Structures

This section is devoted to the definitions of tree encodings and characteristic structures, and the propositions about the equivalent relationship between them on Datalog^r programs, which will be used in the next section.

Definition 4. Let T be a perfect binary tree, R an r -ary relation on T . R is a saturated relation if for any nodes $a_1, \dots, a_r, b_1, \dots, b_r$, whenever $d(a_i) = d(b_i)$ ($1 \leq i \leq r$), then $(a_1, \dots, a_r) \in R$ iff $(b_1, \dots, b_r) \in R$.

The following proposition can be proved easily from the definitions of invariant relations and saturated relations.

Proposition 5. A saturated relation is also an invariant relation.

From now on we make the assumption: τ is the vocabulary $\{R_1, \dots, R_k\}$, and $\tau' = \tau \cup \{\mathbf{root}, E\}$, where \mathbf{root} is a constant symbol and E is a binary relation symbol that is not in τ . We define a class of τ' -structures

$$\mathcal{T} = \{ \langle V, \mathbf{root}, E, R_1, \dots, R_k \rangle \mid \langle V, E, \mathbf{root} \rangle \text{ is a perfect binary tree, } R_1, \dots, R_k \text{ are saturated relations on it} \}.$$

Definition 5. Let $\mathbf{A} = \langle \{0, 1, \dots, h-1\}, R_1^A, \dots, R_k^A \rangle$ be a τ -structure. The tree encoding of \mathbf{A} is a τ' -structure $C(\mathbf{A}) = \langle V, \mathbf{root}, E, R_1^T, \dots, R_k^T \rangle \in \mathcal{T}$, such that $\langle V, E, \mathbf{root} \rangle$ is a perfect binary tree of depth h , and for any relation symbol R_i ($1 \leq i \leq k$) and any nodes $a_1, \dots, a_{r_i} \in V$,

$$C(\mathbf{A}) \models R_i^T a_1 \cdots a_{r_i} \text{ iff } \mathbf{A} \models R_i^A d(a_1) \cdots d(a_{r_i})$$

where r_i is the arity of R_i , and $d(a_j)$ ($1 \leq j \leq r_i$) is the depth of a_j .

Roughly speaking, $C(\mathbf{A})$ encodes \mathbf{A} in a tree, but its size is exponentially larger. Conversely, given a τ' -structure $\mathbf{T} = \langle V, \mathbf{root}, E, R_1^T, \dots, R_k^T \rangle \in \mathcal{T}$, we can compute the τ -structure \mathbf{A} encoded by \mathbf{T} as follows:

- (1) The domain is $\{0, \dots, h-1\}$, where h is the depth of \mathbf{T} ;
- (2) For each $i = 1, \dots, k$,

$$R_i^A = \{ (d(a_1), \dots, d(a_{r_i})) \mid \exists a_1, \dots, a_{r_i} \in V \text{ such that } \mathbf{T} \models R_i^T a_1 \cdots a_{r_i} \}.$$

We use $C^{-1}(\mathbf{T})$ to denote the corresponding τ -structure \mathbf{A} encoded by \mathbf{T} . Let $\text{FUL}_m = V^m$ be a relation of arity m , where $m \geq 1$ and V is the domain of \mathbf{T} . Define the vocabulary

$$\sigma = \{\mathbf{0}, \text{SUCC}, R_{\neq}, R_{\neg e}, \text{FUL}_m^*\} \cup \{R_1^*, \dots, R_k^*, (\neg R_1)^*, \dots, (\neg R_k)^*\}$$

where FUL_m^* has arity $\frac{m(m+1)}{2}$, R_{\neq} and $R_{\neg e}$ have arity 3, R_i^* or $(\neg R_i)^*$ has arity $\frac{r_i(r_i+1)}{2}$ ($1 \leq i \leq k$ and r_i is the arity of R_i).

Definition 6. *Given a τ' -structure $\mathbf{T} = \langle V, \text{root}, E, R_1, \dots, R_k \rangle \in \mathcal{T}$, the characteristic structure $S_{\mathbf{T}}$ of \mathbf{T} is a σ -structure*

$$\langle \{0, 1, \dots, h-1\}, \mathbf{0}, \text{SUCC}, R_{\neq}, R_{\neg e}, \text{FUL}_m^*, R_1^*, \dots, R_k^*, (\neg R_1)^*, \dots, (\neg R_k)^* \rangle$$

where h is the depth of \mathbf{T} , $\mathbf{0}$ is a constant interpreted by 0, SUCC is the successor relation on the domain, and $R_{\neq}, R_{\neg e}, \text{FUL}_m^*, R_1^*, \dots, R_k^*, (\neg R_1)^*, \dots, (\neg R_k)^*$ are the characteristic relations of $\neq, (\neg E), \text{FUL}_m, R_1, \dots, R_k, \neg R_1, \dots, \neg R_k$, respectively.

In the following we show that for every Datalog^r program Π on the tree encodings, there is a Datalog^r program Π^* on the corresponding characteristic structures such that Π^* simulates Π . More precisely, Π^* handles the characteristic relations of the relations in Π . Let $\Pi = \{\gamma_1, \dots, \gamma_s\}$ be a Datalog^r program on \mathcal{T} . Suppose X_1, \dots, X_w are all intensional relation symbols in Π and for each rule γ_i , let n_{γ_i} be the number of free variables occurring in γ_i . Set

$$m = \max\{n_{\gamma_1}, \dots, n_{\gamma_s}, \text{arity}(R_1), \dots, \text{arity}(R_k), \text{arity}(X_1), \dots, \text{arity}(X_w)\}.$$

We shall construct, based on Π , a Datalog^r program Π^* such that for any Datalog^r formula (Π, P) , there exists a Datalog^r formula (Π^*, P^*) , and $\mathbf{T} \models (\Pi, P)$ iff $S_{\mathbf{T}} \models (\Pi^*, P^*)$ for any $\mathbf{T} \in \mathcal{T}$, where P and P^* are 0-ary.

Every element of \mathbf{T} is a node of a perfect binary tree, while every element of $S_{\mathbf{T}}$ is a number which can be treated as the depth of some node. Hence, for each variable x in Π , we introduce a new variable i_x , and for any two variables x_1, x_2 in Π , we introduce a new variable $i_{x_1 \bar{\wedge} x_2}$. For a tuple of variables $\bar{x} = x_1 \cdots x_r$, we use the following abbreviations:

$$\begin{aligned} (\bar{x})^* &= i_{x_1} i_{x_1 \bar{\wedge} x_2} \cdots i_{x_1 \bar{\wedge} x_r} i_{x_2} i_{x_2 \bar{\wedge} x_3} \cdots i_{x_{r-1} \bar{\wedge} x_r} i_{x_r}, \\ \forall(\bar{x})^* &= \forall i_{x_1} \forall i_{x_1 \bar{\wedge} x_2} \cdots \forall i_{x_1 \bar{\wedge} x_r} \forall i_{x_2} \forall i_{x_2 \bar{\wedge} x_3} \cdots \forall i_{x_{r-1} \bar{\wedge} x_r} \forall i_{x_r}. \end{aligned}$$

Without loss of generality, we treat $i_{u \bar{\wedge} v}$ and $i_{v \bar{\wedge} u}$ as the same variable. Additionally, we assume that the 0-ary relation is also an invariant relation, and the characteristic relation of a 0-ary relation is itself.

First we construct a quasi-Datalog^r program Π' as follows. For each rule $\beta \leftarrow \alpha_1, \dots, \alpha_l$ in Π , suppose that v_1, \dots, v_n are the free variables in it, we add the formula

$$\text{FUL}_m v_1 v_2 \cdots v_{n-1} v_n v_n \cdots v_n$$

to the body and obtain a new rule

$$\beta \leftarrow \alpha_1, \dots, \alpha_l, \text{FUL}_m v_1 v_2 \cdots v_{n-1} v_n \cdots v_n.$$

For each new rule, we

- replace $x = y$ by $i_x = i_{x\bar{\wedge}y}, i_{x\bar{\wedge}y} = i_y$ (reason: $d(x) = d(x\bar{\wedge}y) = d(y)$), for constant **root**, we replace i_{root} by constant **0**, and replace $i_{\text{root}\bar{\wedge}x}$ also by **0**, since **root** $\bar{\wedge} a = \text{root}$ for any node a ;
- replace Exy by $i_x = i_{x\bar{\wedge}y}, \text{SUCC}i_{x\bar{\wedge}y}i_y$ (reason: $d(y) = d(x\bar{\wedge}y) + 1 = d(x) + 1$);
- replace $x \neq y$ by $R_{\neq}i_x i_{x\bar{\wedge}y}i_y$ (reason: R_{\neq} is the characteristic relation of \neq);
- replace $\neg Exy$ by $R_{\neg e}i_x i_{x\bar{\wedge}y}i_y$ (reason: $R_{\neg e}$ is the characteristic relation of $\neg E$);
- replace $P\bar{x}$ by $P^*(\bar{x})^*$, where P is in $\{R_1, \dots, R_k, \text{FUL}_m\}$, or an intensional relation symbol;
- replace $\neg R\bar{x}$ by $(\neg R)^*(\bar{x})^*$, where R is a symbol in $\{R_1, \dots, R_k\}$;
- replace $\forall y_1 \cdots \forall y_t P y_1 \cdots y_t z_1 \cdots z_s$ by

$$\Psi_P = \left(\begin{array}{l} \text{FUL}_m^*(z_1 z_2 \cdots z_{s-1} z_s \cdots z_s)^* \wedge \\ \forall (y_1 \cdots y_t)^* \forall i_{y_1 \bar{\wedge} z_1} \cdots \forall i_{y_1 \bar{\wedge} z_s} \forall i_{y_2 \bar{\wedge} z_1} \cdots \forall i_{y_t \bar{\wedge} z_{s-1}} \forall i_{y_t \bar{\wedge} z_s} \\ (\text{FUL}_m^*(y_1 \cdots y_t z_1 \cdots z_s \cdots z_s)^* \rightarrow P^*(y_1 \cdots y_t z_1 \cdots z_s)^* \end{array} \right)$$

where P is an intensional relation symbol.

By adding FUL_m to each rule of Π and replacing it with FUL_m^* in Π' , we can restrict to characteristic tuples. Π' is not a Datalog^r program because of Ψ_P . Note that Ψ_P is equivalent to the Datalog^r formula $(\Pi_1, Q_2)\bar{t}$, where

$$\begin{aligned} \Pi_1 : Q(y_1 \cdots y_t z_1 \cdots z_s)^* &\leftarrow \neg \text{FUL}_m^*(y_1 \cdots y_t z_1 \cdots z_s z_s \cdots z_s)^*; \\ Q(y_1 \cdots y_t z_1 \cdots z_s)^* &\leftarrow P^*(y_1 \cdots y_t z_1 \cdots z_s)^*; \\ Q_1(z_1 \cdots z_s)^* &\leftarrow \forall (y_1 \cdots y_t)^* \forall i_{y_1 \bar{\wedge} z_1} \cdots \forall i_{y_1 \bar{\wedge} z_s} \\ &\quad \forall i_{y_2 \bar{\wedge} z_1} \cdots \forall i_{y_t \bar{\wedge} z_{s-1}} \forall i_{y_t \bar{\wedge} z_s} Q(y_1 \cdots y_t z_1 \cdots z_s)^*; \\ Q_2(z_1 \cdots z_s)^* &\leftarrow Q_1(z_1 \cdots z_s)^*, \text{FUL}_m^*(z_1 z_2 \cdots z_{s-1} z_s \cdots z_s)^*. \end{aligned}$$

The Datalog^r program Π^* can be obtained by adding Π_1 to Π' and changing Ψ_P to $Q_2(z_1 \cdots z_s)^*$.

Remark 1. We cannot replace $\forall y_1 \cdots \forall y_t P y_1 \cdots y_t z_1 \cdots z_s$ directly by

$$\forall (y_1 \cdots y_t)^* \forall i_{y_1 \bar{\wedge} z_1} \cdots \forall i_{y_1 \bar{\wedge} z_s} \forall i_{y_2 \bar{\wedge} z_1} \cdots \forall i_{y_t \bar{\wedge} z_{s-1}} \forall i_{y_t \bar{\wedge} z_s} P^*(y_1 \cdots y_t z_1 \cdots z_s)^*$$

since there may be $\mathbf{T} \in \mathcal{T}$, $\bar{a} \in \mathbf{T}$, and an invariant relation P such that

$$\begin{aligned} \mathbf{T} &\models \forall y_1 \cdots \forall y_t P y_1 \cdots y_t z_1 \cdots z_s [\bar{a}], \text{ and} \\ S_{\mathbf{T}} &\not\models \forall (y_1 \cdots y_t)^* \forall i_{y_1 \bar{\wedge} z_1} \cdots \forall i_{y_1 \bar{\wedge} z_s} \forall i_{y_2 \bar{\wedge} z_1} \cdots \forall i_{y_t \bar{\wedge} z_{s-1}} \forall i_{y_t \bar{\wedge} z_s} \\ &\quad P^*(y_1 \cdots y_t z_1 \cdots z_s)^* [(\bar{a})^*]. \end{aligned}$$

For example, let \mathbf{T} be the structure with the perfect binary tree of Fig. 3 and relation

$$P = \{(\mathbf{root}, \mathbf{root}), (\mathbf{root}, a), (\mathbf{root}, b), (\mathbf{root}, c), (\mathbf{root}, d), (\mathbf{root}, e), (\mathbf{root}, f)\}.$$

Obviously, we have $\mathbf{T} \models \forall y P(\mathbf{root}, y)$. But $S_{\mathbf{T}} \not\models P^*(\mathbf{0}, 1, 1)$ since $(\mathbf{0}, 1, 1)$ is not the characteristic tuple of any tuple in \mathbf{T} . This problem can be solved by changing $\forall y_1 \cdots \forall y_t P y_1 \cdots y_t z_1 \cdots z_s$ to the equivalent formula

$$\text{FUL}_m z_1 \cdots z_{s-1} z_s \cdots z_s \wedge \forall y_1 \cdots \forall y_t (\text{FUL}_m y_1 \cdots y_t z_1 \cdots z_s \cdots z_s \rightarrow P y_1 \cdots y_t z_1 \cdots z_s).$$

We replace FUL_m and P by their characteristic relations FUL_m^* and P^* , respectively, to obtain Ψ_P . This guarantees that only characteristic tuples are considered.

Example 2. The following Datalog^r program Π computes the transitive closure R of edges E

$$\begin{aligned} \Pi : R x_1 x_2 &\leftarrow E x_1 x_2; \\ R x_1 x_3 &\leftarrow R x_1 x_2, E x_2 x_3. \end{aligned}$$

The corresponding Datalog^r program Π^* below computes the characteristic relation R^* of R .

$$\begin{aligned} \Pi^* : R^* i_{x_1} i_{x_1 \bar{\wedge} x_2} i_{x_2} &\leftarrow i_{x_1} = i_{x_1 \bar{\wedge} x_2}, \text{SUCC} i_{x_1 \bar{\wedge} x_2} i_{x_2}, \\ &\quad \text{FUL}_3^* i_{x_1} i_{x_1 \bar{\wedge} x_2} i_{x_1 \bar{\wedge} x_2} i_{x_2} i_{x_2} i_{x_2}; \\ R^* i_{x_1} i_{x_1 \bar{\wedge} x_3} i_{x_3} &\leftarrow R^* i_{x_1} i_{x_1 \bar{\wedge} x_2} i_{x_2}, i_{x_2} = i_{x_2 \bar{\wedge} x_3}, \text{SUCC} i_{x_2 \bar{\wedge} x_3} i_{x_3}, \\ &\quad \text{FUL}_3^* i_{x_1} i_{x_1 \bar{\wedge} x_2} i_{x_1 \bar{\wedge} x_3} i_{x_2} i_{x_2 \bar{\wedge} x_3} i_{x_3}. \end{aligned}$$

Lemma 4. *Given $\psi_P = \forall y_1 \cdots \forall y_t P y_1 \cdots y_t z_1 \cdots z_s$, a structure $\mathbf{T} \in \mathcal{T}$, let Ψ_P be defined as above, and $Q_1 = \{\bar{a} \mid \mathbf{T} \models \psi_P[\bar{a}]\}$, $Q_2 = \{\bar{e} \mid S_{\mathbf{T}} \models \Psi_P[\bar{e}]\}$. If P is an invariant relation on \mathbf{T} , then $(Q_1)^* = Q_2$.*

Proof. Because P is an invariant relation, by Lemma 3 and the definition of Q_1 , we know that Q_1 is also an invariant relation. We first show that $(Q_1)^* \subseteq Q_2$. Suppose that $\bar{e} \in (Q_1)^*$ for some $\bar{e} \in S_{\mathbf{T}}$, there must exist a tuple \bar{a} from \mathbf{T} such that $\bar{a} \in Q_1$, $(\bar{a})^* = \bar{e}$, and $\bar{b}\bar{a} \in P$ for all tuples \bar{b} of \mathbf{T} , i.e.,

$$\begin{aligned} \mathbf{T} \models &\left(\text{FUL}_m z_1 \cdots z_{s-1} z_s z_s \cdots z_s \wedge \right. \\ &\left. \forall y_1 \cdots \forall y_t (\text{FUL}_m y_1 \cdots y_t z_1 \cdots z_s z_s \cdots z_s \rightarrow P y_1 \cdots y_t z_1 \cdots z_s) \right) [\bar{a}]. \end{aligned}$$

By the definition of Ψ_P we see that $S_{\mathbf{T}} \models \Psi_P[(\bar{a})^*]$, which implies $\bar{e} \in Q_2$.

To prove $Q_2 \subseteq (Q_1)^*$, consider an arbitrary tuple $\bar{e} \in S_{\mathbf{T}}$ such that $\bar{e} \in Q_2$. By the definition of Q_2 and Ψ_P , we have $S_{\mathbf{T}} \models \text{FUL}_m(z_1 \cdots z_s z_s \cdots z_s)^*[\bar{e}]$, so there exists a tuple \bar{a} of \mathbf{T} such that $\bar{e} = (\bar{a})^*$. On the contrary, assume $\bar{a} \notin Q_1$, then there is a tuple \bar{b} such that $\bar{b}\bar{a} \notin P$. Because P is an invariant relation, for any tuples \bar{b}' and \bar{a}' , if $(\bar{b}\bar{a})^* = (\bar{b}'\bar{a}')^*$ then $\bar{b}'\bar{a}' \notin P$. Combing that P^* is the characteristic relation of P we conclude that

$$S_{\mathbf{T}} \not\models P^*(y_1 \cdots y_t z_1 \cdots z_s)^*[(\bar{b}\bar{a})^*], \text{ and} \quad (1)$$

$$S_{\mathbf{T}} \models \text{FUL}_m^*(y_1 \cdots y_t z_1 \cdots z_s z_s \cdots z_s)^*[(\bar{b}\bar{a})^*]. \quad (2)$$

(1) and (2) give $S_{\mathbf{T}} \not\models \Psi_P[(\bar{a})^*]$. Hence, $\bar{e} \notin Q_2$, contrary to the assumption that $\bar{e} \in Q_2$. Therefore, \bar{a} must be in Q_1 , which implies $\bar{e} \in (Q_1)^*$. \square

Let P be an intensional relation symbol in Π , and \mathbf{T} a structure in \mathcal{T} . We use $P_{(n)}$ ($n > 0$) to denote the relation obtained in the n -th evaluation of Π on \mathbf{T} for P , and $P^{\mathbf{T}[\Pi]}$ to denote the relation obtained by applying Π on \mathbf{T} for P , i.e., the fixpoint of the sequence $P_{(0)}, P_{(1)}, P_{(2)}, \dots$

Proposition 6. *For any intensional relation symbol P in Π and any $\mathbf{T} \in \mathcal{T}$, $P^{\mathbf{T}[\Pi]}$ is an invariant relation on \mathbf{T} and $(P^{\mathbf{T}[\Pi]})^* = (P^*)^{S_{\mathbf{T}}[\Pi^*]}$. Moreover, if P is a 0-ary intensional relation symbol, then $\mathbf{T} \models (\Pi, P)$ iff $S_{\mathbf{T}} \models (\Pi^*, P^*)$.*

Proof. We first show that if P is an intensional relation symbol in Π and \mathbf{T} is a structure in \mathcal{T} , then $P^{\mathbf{T}[\Pi]}$ is an invariant relation on \mathbf{T} . Let $P^1, \dots, P^{m'}$ be all intensional relation symbols in Π . Consider the following formula constructed for each P^i

$$\phi_{P^i}(\bar{x}_{P^i}) = \bigvee \{ \exists \bar{v} (\alpha_1 \wedge \dots \wedge \alpha_l) \mid P^i \bar{x}_{P^i} \leftarrow \alpha_1, \dots, \alpha_l \in \Pi \text{ and } \bar{v} \text{ are the free variables in } \alpha_1 \wedge \dots \wedge \alpha_l \text{ that are different from } \bar{x}_{P^i} \}.$$

If the relation defined by each α_s is an invariant relation, then by Lemmas 1, 2 and 3, we know that the relation defined by ϕ_{P^i} is also an invariant relation. Each α_s is either an atomic (or negated atomic) formula with the relation symbol from $\{=, E, R_1, \dots, R_k\}$ where the relations defined by them are all invariant relations, or an atomic formula $P^j \bar{x}$, or a formula $\forall \bar{y} P^j \bar{y} \bar{z}$ ($1 \leq j \leq m'$).

When computing the fixpoint of $P^1, \dots, P^{m'}$, we set $P_{(0)}^i = \emptyset$ ($1 \leq i \leq m'$), where \emptyset is an invariant relation. By Lemma 3 we know that if P^j is an invariant relation then the relation defined by $\forall \bar{y} P^j \bar{y} \bar{z}$ is also an invariant relation. We proceed by induction on n . Suppose that $P_{(n)}^1, \dots, P_{(n)}^{m'}$ are invariant relations, then each

$$\begin{aligned} P_{(n+1)}^i &= \{ \bar{a} \mid (\mathbf{T}, P_{(n)}^1, \dots, P_{(n)}^{m'}) \models \phi_{P^i}(\bar{x}_{P^i})[\bar{a}] \}, \text{ or} \\ P_{(n+1)}^i &= \{ \emptyset \mid (\mathbf{T}, P_{(n)}^1, \dots, P_{(n)}^{m'}) \models \phi_{P^i} \} \end{aligned}$$

is also an invariant relation. Therefore, the fixpoints $P_{(\infty)}^1, \dots, P_{(\infty)}^{m'}$ are invariant relations, i.e., $P^{\mathbf{T}[\Pi]}$ is an invariant relation on \mathbf{T} .

Next we shall show that $(P^{\mathbf{T}[\Pi]})^* = (P^*)^{S_{\mathbf{T}}[\Pi^]}$. It suffices to prove that $(P_{(n)}^i)^* = (P_{(n)}^i)_{(n)}^*$ ($1 \leq i \leq m'$) for each $n \geq 0$. The proof is by induction on n .

Basis: If $n = 0$, then $P_{(0)}^i = \emptyset$, $(P_{(0)}^i)_{(0)}^* = \emptyset$ ($1 \leq i \leq m'$). We have $(P_{(0)}^i)^* = (P_{(0)}^i)_{(0)}^*$ ($1 \leq i \leq m'$).

Inductive Step: Assuming $(P_{(k)}^i)^* = (P_{(k)}^i)_{(k)}^*$ ($1 \leq i \leq m'$), we show that $(P_{(k+1)}^i)^* = (P_{(k+1)}^i)_{(k+1)}^*$ ($1 \leq i \leq m'$). The case where P^i is 0-ary is trivial, in the following we only consider the relation P^i of no 0-ary.

To prove $(P^i_{(k+1)})^* \subseteq (P^i_{(k+1)})^*$, suppose $\bar{e} \in (P^i_{(k+1)})^*$ for some $\bar{e} \in S_{\mathbf{T}}$. There must be a tuple \bar{a} of \mathbf{T} such that $\bar{a} \in P^i_{(k+1)}$ and $\bar{e} = (\bar{a})^*$. By the semantics of Datalog^r we know that

$$P^i_{(k+1)} = \{\bar{a} \mid (\mathbf{T}, P^1_{(k)}, \dots, P^{m'}_{(k)}) \models \phi_{P^i}(\bar{x}_{P^i})[\bar{a}]\}.$$

By the definition of ϕ_{P^i} , there is a rule $P^i \bar{x}_{P^i} \leftarrow \alpha_1, \dots, \alpha_l$ in Π such that

$$\langle \mathbf{T}, P^1_{(k)}, \dots, P^{m'}_{(k)} \rangle \models \exists \bar{v}(\alpha_1 \wedge \dots \wedge \alpha_l)[\bar{a}].$$

Thus, there exists some \bar{b} such that

$$\langle \mathbf{T}, P^1_{(k)}, \dots, P^{m'}_{(k)} \rangle \models (\alpha_1 \wedge \dots \wedge \alpha_l)[\bar{a}\bar{b}].$$

Because $P^i \bar{x}_{P^i} \leftarrow \alpha_1, \dots, \alpha_l$ is a rule of Π , we can infer that

$$(P^i)^*(\bar{x}_{P^i})^* \leftarrow \alpha'_1, \dots, \alpha'_l, \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*$$

is a rule of Π^* , where $\alpha'_1, \dots, \alpha'_l$ and $\text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*$ are obtained by replacing $\alpha_1, \dots, \alpha_l, \text{FUL}_m$ with the corresponding formulas respectively in the construction of Π^* . Note that we replace $\forall \bar{y} P \bar{y} \bar{z}$ by Ψ_P , and by Lemma 4 the relation defined by Ψ_P is the characteristic relation of that defined by $\forall \bar{y} P \bar{y} \bar{z}$. By the definition of $S_{\mathbf{T}}$ and the induction hypothesis $(P^i_{(k)})^* = (P^i_{(k)})^*$ ($1 \leq i \leq m'$) we deduce that

$$\begin{aligned} \langle S_{\mathbf{T}}, (P^1_{(k)})^*, \dots, (P^{m'}_{(k)})^* \rangle &\models (\alpha'_1 \wedge \dots \wedge \alpha'_l \wedge \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*)[(\bar{a}\bar{b})^*], \text{ i.e.,} \\ \langle S_{\mathbf{T}}, (P^1_{(k)})^*, \dots, (P^{m'}_{(k)})^* \rangle &\models \exists \bar{u}(\alpha'_1 \wedge \dots \wedge \alpha'_l \wedge \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*)[(\bar{a})^*] \end{aligned}$$

where \bar{u} are the free variables in $\alpha'_1 \wedge \dots \wedge \alpha'_l \wedge \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*$ that are different from $(\bar{x}_{P^i})^*$. Combining $\bar{e} = (\bar{a})^*$ we obtain $\bar{e} \in (P^i_{(k+1)})^*$.

To prove $(P^i_{(k+1)})^* \subseteq (P^i_{(k+1)})^*$, suppose $\bar{e} \in (P^i_{(k+1)})^*$ for some $\bar{e} \in S_{\mathbf{T}}$. There must exist a rule

$$(P^i)^*(\bar{x}_{P^i})^* \leftarrow \alpha'_1, \dots, \alpha'_l, \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^* \quad (3)$$

in Π^* such that

$$\langle S_{\mathbf{T}}, (P^1_{(k)})^*, \dots, (P^{m'}_{(k)})^* \rangle \models \exists \bar{u}(\alpha'_1 \wedge \dots \wedge \alpha'_l \wedge \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*)[\bar{e}].$$

Hence there exists a tuple $\bar{f} \in S_{\mathbf{T}}$ such that

$$\langle S_{\mathbf{T}}, (P^1_{(k)})^*, \dots, (P^{m'}_{(k)})^* \rangle \models (\alpha'_1 \wedge \dots \wedge \alpha'_l \wedge \text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*)[\bar{e}\bar{f}].$$

The formula $\text{FUL}_m^*(\bar{x}_{P^i} \bar{v} \bar{v}')^*$ guarantees that $(\bar{a}\bar{b})^* = \bar{e}\bar{f}$ and $(\bar{a})^* = \bar{e}$ for some tuple $\bar{a}\bar{b}$ of \mathbf{T} . Because $P^1_{(k)}, \dots, P^{m'}_{(k)}$ are invariant relations, by the induction hypothesis $(P^i_{(k)})^* = (P^i_{(k)})^*$ ($1 \leq i \leq m'$) we know that

$$\langle \mathbf{T}, P^1_{(k)}, \dots, P^{m'}_{(k)} \rangle \models (\alpha_1 \wedge \dots \wedge \alpha_l)[\bar{a}\bar{b}]$$

where $\alpha_1, \dots, \alpha_l$ occur in the rule $P^i \bar{x}_{P^i} \leftarrow \alpha_1, \dots, \alpha_l$ that is the original of (3) in Π . Hence $\bar{a} \in P^i_{(k+1)}$, which implies $\bar{e} \in (P^i_{(k+1)})^*$. This completes the proof. \square

3.4 Nondefinability Results for Datalog^r

The complexity class EXPTIME contains the decision problems decidable by a deterministic Turing machine in $O(2^{n^c})$ time. By the time hierarchy theorem, we know that PTIME is a proper subset of EXPTIME. In this section, for every class of structures $\mathcal{K} \in \text{EXPTIME}$, we construct a class \mathcal{K}' of structures that is in PTIME and closed under substructures, and show that if \mathcal{K}' is definable by a Datalog^r formula, then \mathcal{K} is in P, which is impossible.

Let c be a constant, and $\mathbf{A} = \langle \{0, \dots, h-1\}, R_1^A, \dots, R_k^A \rangle$ a τ -structure. The *trivial extension* $\mathbf{A}^+ = \langle \{0, \dots, h-1, h, \dots, h+h^c-1\}, R_1^A, \dots, R_k^A \rangle$ of \mathbf{A} is a τ -structure obtained by adding h^c dummy elements to the domain of \mathbf{A} and keeping all other relations unchanged.

For technical reasons we introduce a new unary relation symbol U and let $\tau_U = \tau \cup \{U\}$, $\tau'_U = \tau \cup \{\mathbf{root}, E, U\}$. From now on when we speak of a τ'_U -structure

$$\mathbf{G} = \langle V, \mathbf{root}, E, U, R_1, \dots, R_k \rangle$$

we assume that

- (1) $\langle V, E, \mathbf{root} \rangle$ is a directed acyclic graph and the nodes reachable from \mathbf{root} form a binary tree, and
- (2) all relations U, R_1, \dots, R_k are saturated relations restricted on $T(\mathbf{G})$, which is the largest perfect binary subtree of \mathbf{G} with \mathbf{root} as the root.

It is easy to check that if a τ'_U -structure \mathbf{G} satisfies the aforementioned two conditions, then all its substructures also satisfy the two conditions.

Definition 7. Let \mathcal{K} be a class of τ -structures. Define a class \mathcal{K}' of τ'_U -structures such that, for any $\mathbf{G} = \langle V, \mathbf{root}, E, U, R_1, \dots, R_k \rangle$, let h be the largest number where all nodes in the first h levels of $T(\mathbf{G})$ are marked by U , $\mathbf{G} \in \mathcal{K}'$ iff the following Condition (1) or Condition (2) holds.

Condition (1)

- (a) The depth of $T(\mathbf{G})$ is $h + h^c$.
- (b) The relations R_1, \dots, R_k do not hold on any tuple that contains a node in the last h^c consecutive levels of $T(\mathbf{G})$.
- (c) $C^{-1}(T(\mathbf{G}))$ is the trivial extension of $C^{-1}(T_h(\mathbf{G}))$, where $T_h(\mathbf{G})$ is the subtree of $T(\mathbf{G})$ by restricting to the first h levels.
- (d) $C^{-1}(T_h(\mathbf{G})) \in \mathcal{K}$ when ignoring the relation U .

Condition (2)

- (a) The depth of $T(\mathbf{G})$ is strictly less than $h + h^c$.

Proposition 7. Let \mathcal{K} be an arbitrary class of τ -structures decidable in 2^{n^c} time, where n is the cardinality of the structure's domain, and \mathcal{K}' defined as above. Then

- (i) \mathcal{K}' is closed under substructures;
- (ii) \mathcal{K}' is decidable in PTIME.

Proof. To prove (i), suppose that \mathbf{G} is a τ'_U -structure in \mathcal{K}' , then it satisfies either Condition (1) or Condition (2) in Definition 7. Let \mathbf{H} be an arbitrary substructure of \mathbf{G} . If \mathbf{G} satisfies Condition (2), then \mathbf{H} also satisfies Condition (2), and is in \mathcal{K}' . Suppose that \mathbf{G} satisfies Condition (1), then the perfect binary tree $T(\mathbf{H})$ either equals $T(\mathbf{G})$, which implies \mathbf{H} satisfies Condition (1), or the depth of $T(\mathbf{H})$ is less than that of $T(\mathbf{G})$, which implies \mathbf{H} satisfies Condition (2). Altogether, $\mathbf{H} \in \mathcal{K}'$.

To prove (ii), let \mathbf{G} be an arbitrary τ'_U -structure, we just need to do the following steps to check whether $\mathbf{G} \in \mathcal{K}'$:

- (1) Check that $\langle V, E \rangle$ is a directed acyclic graph.
- (2) Check that all nodes reachable from **root** form a binary tree.
- (3) Compute $T(\mathbf{G})$, the largest perfect binary subtree with **root** as root.
- (4) Check that U, R_1, \dots, R_k are saturated relations on $T(\mathbf{G})$.
- (5) Compute the largest number h such that all nodes in the first h levels of $T(\mathbf{G})$ have property U .
- (6) Check whether the depth of $T(\mathbf{G})$ is less than $h + h^c$.
- (7) If the depth of $T(\mathbf{G})$ is $h + h^c$, then check whether (b), (c) and (d) in Condition (1) of Definition 7 hold.

Note that $C^{-1}(T_h(\mathbf{G}))$ has h elements and \mathcal{K} is decidable in 2^{n^c} time, the statement (d) in Condition (1) of Definition 7 can be verified in polynomial time since if the depth of $T(\mathbf{G})$ is $h + h^c$ then the input size is at least 2^{h+h^c} . \square

Let \mathcal{K} and \mathcal{K}' be defined as in Proposition 7. For an arbitrary τ -structure \mathbf{A} , let \mathbf{A}_U be the τ_U -structure obtained by marking every element in \mathbf{A} by U , \mathbf{A}_U^+ the trivial extension of \mathbf{A}_U by adding $|A|^c$ elements, and \mathbf{T} the τ'_U -structure such that $C^{-1}(\mathbf{T}) = \mathbf{A}_U^+$. If \mathcal{K}' is axiomatizable by a Datalog^r formula (Π, Q) , then

$$\mathbf{A} \in \mathcal{K} \quad \text{iff} \quad \mathbf{T} \in \mathcal{K}' \quad \text{iff} \quad \mathbf{T} \models (\Pi, Q). \quad (4)$$

Define the vocabulary

$$\sigma_U = \{\mathbf{0}, \text{SUCC}, R_{\neq}, R_{-e}, \text{FUL}_m^*, U^*, R_1^*, \dots, R_k^*, (\neg U)^*, (\neg R_1)^*, \dots, (\neg R_k)^*\}.$$

By Definition 6, we can compute \mathbf{T} 's characteristic structure that is a σ_U -structure

$$S_{\mathbf{T}} = \langle \{0, 1, \dots, |A| + |A|^c - 1\}, \mathbf{0}, \text{SUCC}, R_{\neq}, R_{-e}, \text{FUL}_m^*, U^*, R_1^*, \dots, R_k^*, (\neg U)^*, (\neg R_1)^*, \dots, (\neg R_k)^* \rangle$$

where $\mathbf{0}$ is interpreted by 0, SUCC is the successor relation on the domain and $R_{\neq}, R_{-e}, \text{FUL}_m^*, U^*, R_1^*, \dots, R_k^*, (\neg U)^*, (\neg R_1)^*, \dots, (\neg R_k)^*$ are the characteristic relations of $\neq, (\neg E), \text{FUL}_m, U, R_1, \dots, R_k, \neg U, \neg R_1, \dots, \neg R_k$, respectively. By Proposition 6, we know there is a Datalog^r formula (Π^*, Q^*) such that $\mathbf{T} \models (\Pi, Q)$ iff $S_{\mathbf{T}} \models (\Pi^*, Q^*)$. Combining (4), we have $\mathbf{A} \in \mathcal{K}$ iff $S_{\mathbf{T}} \models (\Pi^*, Q^*)$.

Lemma 5. $S_{\mathbf{T}}$ is logspace computable from \mathbf{A} .

By Lemma 5, we know that $S_{\mathbf{T}}$ is computable from \mathbf{A} in polynomial time. Hence, \mathcal{K} is in PTIME. Since \mathcal{K} is an arbitrary class in EXPTIME, this would imply EXPTIME=PTIME, which contradicts the time hierarchy theorem. So we must have:

Proposition 8. *There is a problem in PTIME and closed under substructures but not definable in Datalog^r .*

If a problem is closed under substructures, then its complement is closed under extensions. By Proposition 2, we can obtain the following corollary.

Corollary 1. $\text{DATALOG}^r[\mathbf{E}] = \text{LFP}[\mathbf{E}] \subsetneq \text{PTIME}[\mathbf{E}]$.

4 Conclusion

Revised Datalog is an extension of Datalog by allowing universal quantification over intensional relations in the body of rules. On all finite structures, Datalog^r is strictly more expressive than Datalog, and has the same expressive power as that of LFP. In classical model theory, the closure properties of a formula are usually related to some syntactic properties. Many preservation theorems have been proven to reflect this relationship. When restricted to finite structures, most of these preservation theorems fail. Due to the syntax and semantics of Datalog, we can treat it as the dual of SO-HORN logic, which is closed under substructures [12]. It follows that Datalog is closed under extensions. A lot of work has been conducted between Datalog and FO (or LFP) to study the closure property. In this paper, we study the expressive power of revised Datalog on the problems that are closed under substructures. We show that Datalog^r cannot define all the problems that are in PTIME and closed under substructures. As a corollary, LFP cannot define all the extension-closed problems that are in PTIME. A method of tree encodings for arbitrary structures is used in the proof. If we replace the extension closure property by the homomorphism closure property, it is still open whether the statement also holds. This is desirable for future work.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995)
2. Abiteboul, S., Vianu, V.: Datalog extensions for database queries and updates. J. Comput. Syst. Sci. **43**(1), 62–124 (1991)
3. Afrati, F.N., Cosmadakis, S.S., Yannakakis, M.: On datalog vs polynomial time. J. Comput. Syst. Sci. **51**(2), 177–196 (1995)
4. Ajtai, M., Gurevich, Y.: Datalog vs first-order logic. J. Comput. Syst. Sci. **49**(3), 562–588 (1994)
5. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. **33**(3), 374–425 (2001)

6. Dawar, A., Kreutzer, S.: On datalog vs. LFP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 160–171. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_14
7. Dawar, A., Sankaran, A.: Extension preservation in the finite and prefix classes of first order logic. In: Baier, C., Goubault-Larrecq, J. (eds.) 29th EACSL Annual Conference on Computer Science Logic, CSL 2021, 25–28 January 2021, Ljubljana, Slovenia. LIPIcs, vol. 183, pp. 18:1–18:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
8. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Elsevier, Amsterdam (2012)
9. Ebbinghaus, H.D., Flum, J.: Finite Model Theory. Springer-Verlag, Berlin (1995)
10. Feng, S., Zhao, X.: The complexity and expressive power of second-order extended logic. *Stud. Logic* **5**(1), 11–34 (2012)
11. Fernandes, A.A., Paton, N.W.: Databases. In: Meyers, R.A. (ed.) *Encyclopedia of Physical Science and Technology* (Third Edition), pp. 213–228. Academic Press, New York (2003)
12. Grädel, E.: The expressive power of second order horn logic. In: Choffrut, C., Jantzen, M. (eds.) STACS 1991. LNCS, vol. 480, pp. 466–477. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0020821>
13. Gurevich, Y.: Datalog: a perspective and the potential. In: Barceló, P., Pichler, R. (eds.) *Datalog 2.0 2012*. LNCS, vol. 7494, pp. 9–20. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32925-8_2
14. Immerman, N.: Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.* **22**(3), 384–406 (1981)
15. Immerman, N.: Relational queries computable in polynomial time. In: Fourteenth Annual ACM Symposium on Theory of Computing. pp. 147–152. ACM (1982)
16. Ketsman, B., Koch, C.: Datalog with negation and monotonicity. In: Lutz, C., Jung, J. (eds.) 23rd International Conference on Database Theory, ICDT 2020. Leibniz International Proceedings in Informatics, LIPIcs, Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing (2020)
17. Lindell, S.: An analysis of fixed-point queries on binary trees. *Theoret. Comput. Sci.* **85**(1), 75–95 (1991)
18. Marinescu, D.C.: Chapter 6 - cloud data storage. In: Marinescu, D.C. (ed.) *Cloud Computing* (Second Edition), pp. 195–233. Morgan Kaufmann (2018)
19. Revesz, P.Z.: Constraint databases: a survey. In: Thalheim, B., Libkin, L. (eds.) *SiD 1995*. LNCS, vol. 1358, pp. 209–246. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0035010>
20. Rosen, E.: Finite Model Theory and Finite Variable Logics. Ph.D. thesis, University of Pennsylvania (1995)
21. Schlipf, J.S.: Complexity and undecidability results for logic programming. *Ann. Math. Artif. Intell.* **15**, 257–288 (1995)
22. Shmueli, O.: Decidability and expressiveness aspects of logic queries. In: Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1987, pp. 237–249. Association for Computing Machinery, New York (1987)
23. Siekmann, J.H.: *Computational Logic, Handbook of the History of Logic*, vol. 9. North Holland (2014)
24. Vardi, M.Y.: The complexity of relational query languages. In: Fourteenth Annual ACM Symposium on Theory of Computing, pp. 137–146. ACM (1982)