The Impact of Virtual Views on Containment

Michael Benedikt
michael.benedikt@comlab.ox.ac.uk

Georg Gottlob
georg.gottlob@comlab.ox.ac.uk

ABSTRACT

Virtual views are a mechanism that facilitates re-use and makes queries easier to express. However the use of iterative view definitions makes very simple query evaluation and analysis problems more complex. In this paper we study classical containment and equivalence problems for queries built up through simple unions of conjunctive queries and view definitions. More precisely, we determine the complexity of containment and equivalence for non-recursive Datalog. We show that the problem is much harder than its classical counterpart – complete for co-NEXPTIME. We then show that this remains true even with restrictions on the schema and queries in place. Finally, we isolate subcases that are more tractable, ranging from NP to PSPACE.

1. INTRODUCTION

The impact of logical views on query processing has been studied extensively in the database community [15]. A standard setting considers the impact of views as a way of *restricting* the information available to queriers, motivated by the need either to hide data from unauthorized users or to reduce space consumption. A fundamental problem is to analyze what queries can be answered exclusively using a set of views – for example, to compare the query answers that can be answered using (only) a view with those that could have been retrieved directly from the base data [21, 4]. The question is particularly relevant when the views are materialized but the base relations from which views are defined are not.

In contrast, virtual views can also play a role in making queries easier to write: this is how they are commonly used by SQL developers – as a concise and modular way of describing common query components. It has been long understood that the use of views allows queries to be more succinctly represented. But we will argue that the impact of the use of views as a device for query-definition has not been fully understood as yet.

Consider the most basic setting: a query Q is defined via a sequence of nested view definitions. We wish to know if a top-level atom of Q is redundant. Can we do this efficiently? As another example, two queries Q and Q' are defined via a set of nested view definitions. How can we determine whether Q is equivalent to Q'? We emphasize here that we use standard view definitions by conjunctive queries and unions of conjunctive queries. Query containment and equivalence analysis has numerous applications, including query optimization, the checking of integrity constraints [14], and independence-checking for updates [20].

Of course, view definitions are a macro facility, and like any such they add no expressiveness: in the case of the view definitions above, the queries can be expanded into traditional unions of conjunctive queries, and the standard results and algorithms can be applied. However, this expansion blows-up the size of the queries – doubly-exponentially, as we will demonstrate. Hence the naive application of query containment and minimization procedures is infeasible. Is there a cleverer way to do analysis on views without expansion?

In this paper, we examine the question in-depth. The containment question over queries built from chains of views is exactly the same as containment problem for the well-known language *nonrecursive Datalog*. Containment for full Datalog – that is, the corresponding language using *recursive* definitions — has been extensively studied in the literature. Shmueli [23] showed that containment (and hence equivalence) of Datalog is undecidable. [3, 2] isolated subclasses which are decidable, focusing on restricting the form of recursion used. Chaudhuri and Vardi [7] give an extensive study of the containment of Datalog queries in nonrecursive Datalog queries, showing in particular that the problem is decidable. They also show that it is 2ExpTime-hard even to decide containment of a Datalog query within a union of conjunctive queries. The lower-bound does not apply to the nonrecursive case (and indeed, we will show that the problem is less complex). Chaudhuri and Vardi also show that for restricted classes the containment and equivalence problems can become simpler [6] – however, the restrictions considered do not subsume the nonrecursive case, and the lower-bounds of [6] make heavy use of recursion.

Surprisingly, the complexity of containment for nonrecursive Datalog has to our knowledge never been explicitly studied. It is known that evaluation of nonrecursive Datalog is PSPACE-complete [25], and this easily implies a PSPACE lower bound for the containment problem, as well as implying upper bounds in some special cases.

The main contribution of this paper is to isolate the ex-

Proceedings of the VLDB Endowment, Vol. 3, No. 1
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

act complexity of nonrecursive Datalog containment. We show in Section 3 (Theorem 3) that the problem is complete for co-NEXPTIME. We then isolate the source of the complexity. In Section 4 we show that the complexity remains similarly high even when drastic restrictions are made on the input schema or in the right-hand "containing" query. In Section 5 we isolate some subcases with lower complexity, focusing on restrictions in the left-hand "contained" query. We show that the source of complexity lies in the nesting of sharing (multiple occurrences of the same predicate in a rule body) and disjunction in the left-hand query. Section 6 gives related work, while Section 7 gives a summary of our results, along with conclusions.

Our work can be seen as an analysis of the complexity of containment as the height of a chain of view definitions grows. It should be noted that current manually-written queries typically would not employ deep hierarchies of nested view definitions. However, in the setting of data integration systems it is natural to build these kinds of hierarchies, and we expect to see more of these as integration systems become more widespread. Our results indicate that the complexity of query containment becomes super-exponential as height increases, unless limitations on sharing or disjunction are imposed.

2. BACKGROUND

Datalog. An *atom* over a relational signature S is an expression $R(x_1 \dots x_n)$ where R is an n-ary predicate of S and the x_i are either variables or constants. A *pure atom* is one in which all x_i are variables.

A Datalog query consists of:

- A relational signature S along with a collection of constants C
- a set of rules of the form $\phi \to$ A, where ϕ is a conjunction of atoms over S, and A is an atom over S. A is the *head* of the rule and ϕ is the *body* of the rule. We will often identify ϕ with the set of atomic formulas in it, writing $A_1(\vec{x}_1), \dots, A_n(\vec{x}_n)$ instead of $A_1(\vec{x}_1) \wedge \dots \wedge A_n(\vec{x}_n)$. A variable that occurs in the head of rule r is a *free variable* of r. Other variables are *bound* in r; we write $bvars(r)$ for the bound variables of r. We require that every free variable occurs in the body.
- A distinguished predicate P of S which occurs in the head of a rule, referred to as the *goal predicate.*

The relational symbols that do not occur in the head of any rule are the *input predicates*, while the others are *intensional predicates.* A predicate P immediately depends on another predicate P' if there is a rule that has P in the head and P' in the body. A Datalog query is *nonrecursive* if this relation is acyclic. We let NRDL denote the language of nonrecursive Datalog queries.

Pure vs. Impure Queries. We say that an NRDL query is *pure* if all atoms in the heads of rules with nonempty bodies are pure, and is impure otherwise. Equivalently impure NRDL can be seen as the language of queries over view definitions where a definition can contain equalities between variables in the view predicate and constants. Impure NRDL is a more succinct language than pure NRDL, and this has impact on the complexity of many computational problems. For example, the complexity of NRDL satisfiability is PSPACE-complete in the impure case, since it can be reduced to evaluation, while satisfiability for pure NRDL queries admits of a simple syntactic check.

We believe that all the results of this paper hold for both the pure and impure case, and certainly for lower bounds it is clear that proving them for the pure case is sufficient. However, for brevity *all the results we state here will be for the pure case only.* In particular, we will assume pure queries in the definition of the semantics and Unf below, which will be a considerable convenience.

Note that in pure queries, constants can still appear in rule bodies. However constants play little role in the complexity of containment for pure queries. Containment questions for pure NRDL queries with constants can generally be reduced to the corresponding questions for NRDL without any constants, by replacing each constant c in Q or Q' by a unary predicate $IsC(x)$; one can enforce that distinct constant symbols represent distinct values by adding a rule to Q' stating that $IsC(x), IsD(x) \to \mathsf{Goal}$.

Semantics. For a NRDL query, we can define the *rank* of an intensional predicate P (with respect to the query, although we omit the argument), denoted $\mathsf{Rk}(P)$, as follows: the rank of an input predicate is 0, the rank of an intensional predicate P is
$\max\{\mathsf{Rk}(P') + 1$: there is a rule with P' in the body and P in the head$\}$.

Given a structure D interpreting the input predicates, an NRDL query Q, and a predicate P of Q, we define the evaluation of P in D, denoted $P(D)$, by induction on $\mathsf{Rk}(P)$. For P an input predicate $P(D)$ is the interpretation of P in D. For P an intensional predicate of Rk $k+1$ and arity l.

- Let D^k be the expansion of D with $P'(D)$ for all intensional P' of rank at most k.
- If r is a rule with $P(x_1 \dots x_l)$ in the head, $\vec{w}$ the bound variables of r, and $\phi(\vec{x}, \vec{w})$ the body of r let $P_r(D)$ be defined by:
$$\{\vec{c} \in Dom(D)^l : (D^k, x_1 \mapsto c_1 \dots x_l \mapsto c_l) \models \exists \vec{w}\ \phi\}$$

We let $P(D)$ denote the union of $P_r(D)$ over all r with P in the head. Finally, the *result* of a query Q on D is the evaluation of the goal predicate of Q on D. We will often assume Goal is nullary, in which case the result of the query on D is the boolean true iff Goal holds in D. All of our complexity upper bounds are stated for boolean evaluation, but they can be restated in terms of membership in a non-boolean result.

Unfoldings. Let *positive existential first-order logic* ($\exists^+$FO) be the fragment of predicate logic built up from atoms using input predicates via $\wedge, \vee$ and existential quantification.

A *conjunctive query* is a $\exists^+$FO query of the form $\exists x_1 \dots \exists x_n \phi$, where ϕ does not contain $\vee$ or $\exists$. A union of conjunctive queries (or UCQ) is a disjunction of conjunctive queries. We take the empty disjunction and the empty $\exists^+$FO query to denote (by convention) the query always evaluating to false.

We give an alternative semantics of NRDL queries Q via translation to $\exists^+$FO. The translation simply expands/inlines each intensional predicate iteratively (we use the term "unfolding" below).

To start with a simple example, consider the following nonrecursive Datalog program Q whose input predicates are

B and D and whose intensional predicates are R and G:

$$\begin{array}{l}\mathsf{R}(x,y) \wedge \mathsf{R}(y,z) \rightarrow \mathsf{G}(x,z)\\ \mathsf{B}(x',y',u) \rightarrow \mathsf{R}(y',x')\\ \mathsf{D}(x'',y'') \rightarrow \mathsf{R}(x'',y'')\end{array}$$

The unfolding $\mathsf{Unf}(\mathsf{G}(x,z)),Q)$ of predicate G in Q is then the following $\exists^{+}\mathsf{FO}$ query with free variables x and z:

$$\exists y(((\exists u\,\mathsf{B}(y,x,u)) \vee \mathsf{D}(x,y)) \wedge ((\exists u\,\mathsf{B}(z,y,u)) \vee \mathsf{D}(y,z))).$$

As mentioned previously, we give the formal definition of Unf only in the case where all atoms are pure, leaving the extension to the general case (i.e. with constants) to the reader. We will also assume that rules use distinct variables: this will prevent capture of free variables in the unfolding process below.

If $P(\vec{x})$ and $P(\vec{y})$ are atoms with the same predicate, we let $\sigma : \vec{y} \mapsto \vec{x}$ indicate that σ is a bijection mapping the variable in the i^{th} place of $P(\vec{y})$ to the variable in the i^{th} place of $P(\vec{x})$, where $i \leq \text{arity}(P)$. Given a first-order logic formula ϕ whose free variables overlap with $\vec{y}$, and σ a mapping as above, we let $\sigma(\phi)$ be the formula obtained by replacing a variable y_i in $\vec{y}$ with $\sigma(y_i)$. The *unfolding* of an atom $P(\vec{x})$ with respect to a NRDL query Q, denoted $\mathsf{Unf}(P(\vec{x}),Q)$ is defined by induction on $rk(P)$:

- if P is an input predicate, $\mathsf{Unf}(P(\vec{x}),Q) = P(\vec{x})$;
- if P is intentional, then $\mathsf{Unf}(P(\vec{x}),Q) =$

$$\bigvee_{\substack{r:\,rule\ with\ head\ P(\vec{y})\\ \sigma:\vec{y}\mapsto\vec{x},\vec{w}=bvars(r)}} \left(\exists\vec{w} \bigwedge_{P'(\vec{u})\in body(r)} \sigma(\mathsf{Unf}(P'(\vec{u}),Q))\right).$$

One can easily verify by induction on rank that $\mathsf{Unf}(P(\vec{x}),Q)$ is a $\exists^{+}\mathsf{FO}$ formula with exactly $\vec{x}$ free.

We can now state our alternative way to give the semantics of an NRDL query via unfoldings (the proof is immediate from the definitions):

A NRDL query Q is true in a model D iff $\mathsf{Unf}(\mathsf{Goal},Q)$, is true in D, where Goal is the goal predicate and $\mathsf{Unf}(\mathsf{Goal},Q)$ is evaluated under the usual semantics of $\exists^{+}\mathsf{FO}$.

Containment. A query Q is contained in another query Q' if on any input the result of Q is contained in the result of Q'. Let $\mathsf{NRDLCon}$ denote the problem:

Input: two nonrecursive boolean Datalog queries Q,Q'
Output: true iff Q is contained in Q'

The containment problem was first studied by Chandra and Merlin [5] for conjunctive queries. Their results were extended to unions of conjunctive queries by Sagiv and Yannakakis [22]. These prior results are summarized by the following result:

THEOREM 1. *There is an NP algorithm for deciding containment of* UCQ*'s, while the problem of containment is* NP*-hard even for conjunctive queries.*

The results make use of the *canonical database (DB)* of a conjunctive query: the database whose elements are the variables of x, such that for every input relation R in Q, R holds of $\vec{x}$ iff $R(\vec{x})$ is an atom of Q; i.e. the query seen as a database. By a canonical database of a UCQ, we mean a canonical DB of one of its disjuncts. Then the NP upper bound of [22] follows from the following: Q is contained in Q' iff Q' holds true on every canonical database of Q.

We can extend the notation to say that a canonical db of an NRDL query is any a canonical db of its unfolding as a UCQ. The result above still applies. Unfortunately, applying the unfolding mapping followed by a conversion of $\exists^{+}\mathsf{FO}$ to UCQ, and then applying the technique above, would lead to a doubly-exponential blow-up.

EXAMPLE 1. *Consider the following family of queries, taken from [1]. Let A and B be unary predicates and R a binary predicate. Consider the query asking whether there is a path in the interpretation of R consisting of 2^n nodes, with each node satisfying $A \vee B$. One can write this using an* NRDL *query of size $O(n)$, as follows:*

$$\begin{array}{l}A(x) \rightarrow N(x)\\ B(x) \rightarrow N(x)\\ N(x), N(y), R(x,y) \rightarrow P_1(x,y)\end{array}$$

For each $1 \leq i \leq n-1$ the rules:
$P_i(x,w_1), P_i(w_1,y) \rightarrow P_{i+1}(x,y)$

$P_n(x,y) \rightarrow \mathsf{Goal}$
Checking equivalence of queries such as these could clearly not be done feasibly via unfolding.

The following result shows that this can not be avoided (see also [1]):

PROPOSITION 2. *There are* NRDL *queries $Q_n : n \in N$ of size $O(n)$ such that any sequence of* UCQ*'s $Q'_n : n \in N$ with Q'_i equivalent to Q_i must have size doubly-exponential in n.*

Proof: Consider any UCQ equivalent to the query in Example 1. Each disjunct D_i consists of a collection of existentially quantified variables $\vec{x}$ followed by a conjunction C_i. Note that for any simple path p of size 2^n there is a model that has an isomorphic copy of that path, and no other path of this size. For every such path p, let D_p be the disjunct that is satisfied in the corresponding model. We can see that no two paths can map to the same disjunct, and hence there must be doubly exponentially many disjuncts. □

3. THE COMPLEXITY OF CONTAINMENT

The doubly-exponential succinctness of NRDL with respect to unions of conjunctive queries suggests that containment could be significantly harder than the NP-bound one has for UCQ's. The goal of this section is to prove that it is indeed hard, although not 2EXPTIME as would be obtained via naively unfolding and applying the worst-case bounds for UCQ containment.

THEOREM 3. $\mathsf{NRDLCon}$ *is* $\mathsf{co\text{-}NEXPTIME}$*-complete*

We outline the proof here, leaving some details for the appendix. We start with membership in $\mathsf{co\text{-}NEXPTIME}$. We note that Unf is an $\mathsf{EXPTIME}$ function converting a nonrecursive Datalog query into an equivalent formula in $\exists^{+}\mathsf{FO}$. Thus we have an $\mathsf{EXPTIME}$ function that reduces $\mathsf{NRDLCon}$ to the problem: given a $\exists^{+}\mathsf{FO}$ query Q and a NRDL query Q', is Q contained in Q'.

We will need the well-known fact (see, e.g. [9]): the combined complexity of evaluating an NRDL query on a finite structure is in PSPACE.

Membership in $\mathsf{co\text{-}NEXPTIME}$ will follow from combining the exponential time unfolding function, applied to Q only, with the algorithm produced by the following lemma:

Lemma 4. *There is a non-deterministic algorithm that take as input $\exists^+\mathsf{FO}$ query Q and NRDL query Q' and returns true iff Q is not contained in Q', whose running time is $P(|Q|, 2^{|Q'|})$ for P a polynomial.*

Proof: In polynomial time in Q we can convert the $\exists^+\mathsf{FO}$ formula Q to the form $\exists x_1 \dots x_n \phi$, for some n where ϕ is quantifier-free.

We first note that if containment fails, then there is a D satisfying $Q \wedge \neg Q'$ whose size (in number of tuples) is bounded by n. Indeed, we claim that for any such D, there is a submodel D_0 of this size: we let $d_1 \dots d_n$ be such that $D, d_1 \dots d_n \models \phi$ and let D_0 be the submodel of D induced by $d_1 \dots d_n$.

Our non-deterministic machine first guesses a D_0 of size at most n, and also guesses a homomorphism h from $x_1 \dots x_n$ onto elements of D_0. It then verifies that h is a homomorphism for Q into D_0, which can be done in polynomial time, and then verifies that D does not satisfy Q' in space polynomial in Q' (hence in $\mathsf{EXPTIME}$ in $|Q'|$), using the result on evaluation mentioned above.

This completes the proof of the lemma and thus the proof of membership in $\mathsf{co\text{-}NEXPTIME}$. □

We now turn to the more difficult direction, showing hardness.

All of our $\mathsf{co\text{-}NEXPTIME}$-hardness results will follow via reduction from the *Exponential Tiling Problem.* An instance I of this problem is of the form (n, r, H, V, w) where n and r are numbers (in unary), H, V are subsets of $[1, r] \times [1, r]$ and w is a sequence of n numbers each in the range $[1, r]$. Intuitively, we are specifying that we desire a $[0, 2^n - 1] \times [0, 2^n - 1]$ corridor, where each cell is tiled with one of r tiles. w represents a constraint on the initial part of the first row, H is a constraint on any two tiles that are horizontally adjacent, while V is a constraint on vertically-adjacent tiles.

A *solution* for I is a function F from $[0, 2^n - 1] \times [0, 2^n - 1]$ to $[1, r]$ such that $F(i, 1) = w(i)$ for each $i \leq n$, $(F(i, j), F(i+1, j)) \in H$ for $i \leq 2^n - 2$ and $(F(i, j), F(i, j + 1)) \in V$ for $i \leq 2^n - 2$. We will often refer to $[0, 2^n - 1] \times [0, 2^n - 1]$ as a *corridor*, with the pairs in it being *cells.* A cell consists of two *co-ordinates*, and any function on a corridor is a *tiling.* Exponential Tiling ($\mathsf{ExpTile}$) is the problem:

Input: I as above.
Output: True iff I has a solution.

It is known to be complete for $\mathsf{co\text{-}NEXPTIME}$ (see, e.g., Section 3.2 of [17]). It is easy to see that it remains complete if we restrict H, V and their complements to be non-empty, and we will generally assume this in the constructions in the paper.

We will often abuse notation below by referring to 2^n as both a number (as above) and the space of functions from n into $\{0, 1\}$. Given a function f from n into $\{0, 1\}$, we let $\mathsf{Bin}(f)$ be the corresponding number in $[0, 2^n - 1]$.

Our proofs will be polynomial time reductions from $\mathsf{ExpTile}$ to the complement of $\mathsf{NRDLCon}$. That is, given I we form queries Q_I and Q'_I such that I is a tiling iff there is a database satisfying $Q_I \wedge \neg Q'_I$.

In the basic construction, the input signature common to Q_I and Q'_I consists of:

- unary predicates $\mathsf{0}_i, \mathsf{1}_i$ for $i \leq n$. Informally, these represent the i^{th} bit of a coordinate in $[0, 2^n - 1]$. For example, an element x with $\mathsf{0}_1(x), \dots \mathsf{0}_n(x)$ represents the co-ordinate 0, an x with $\mathsf{1}_1(x), \mathsf{0}_2(x), \dots \mathsf{0}_n(x)$ represents the co-ordinate 1, while an x with $\mathsf{1}_1(x), \dots \mathsf{1}_n(x)$ represents the coordinate $2^n - 1$.

- binary predicates $\mathsf{TiledBy}_i$ for $i \leq r$; $\mathsf{TiledBy}_i(x, y)$ will be used to indicate that the cell with co-ordinates x, y is tiled by tile i.

Q_I will have intensional predicates:

- binary predicates Eq_i for $i \leq n$

- binary predicates $\mathsf{TiledAboveCol}_i$ for $i \leq n$ and unary predicate $\mathsf{RowTiled}$

- unary predicates $\mathsf{TiledAboveRow}_i$ for $i \leq n$ and nullary predicate $\mathsf{AllTiled}$

- 0-ary predicate Goal, which will be the goal predicate.

The role of Q_I is to assert the existence of tiles for the whole corridor, ignoring all constraints. The predicate Eq_i will represent equality on the first i bits. The predicate $\mathsf{TiledAboveCol}_i(x_0, y_0)$ will say that for y-coordinate y_0 there are tiled cells with coordinates (x, y_0) for every x value that agrees with x_0 on the first $i - 1$ bits: that is, for the row corresponding to y_0, every column extending the first $i - 1$ bits of x_0 is tiled. In particular $\mathsf{TiledAboveCol}_1$ will say that the entire row for y_0 is tiled. Using $\mathsf{TiledAboveCol}_1$ we will define $\mathsf{TiledAboveRow}_i(y_0)$, which will assert that for every y' whose value agrees with y_0 on the first $i - 1$ bits, the row with y-co-ordinate y' is fully tiled.

Formally, the rules for Q_I are divided into the following sets of rules:

1. ($\mathsf{Equality}$)

   $\mathsf{0}_1(x), \mathsf{0}_1(y) \rightarrow \mathsf{Eq}_1(x, y)$
   $\mathsf{1}_1(x), \mathsf{1}_1(y) \rightarrow \mathsf{Eq}_1(x, y)$

   For $1 \leq i \leq n - 1$ the rules:
   $\mathsf{Eq}_i(x, y), \mathsf{0}_{i+1}(x), \mathsf{0}_{i+1}(y) \rightarrow \mathsf{Eq}_{i+1}(x, y)$
   $\mathsf{Eq}_i(x, y), \mathsf{1}_{i+1}(x), \mathsf{1}_{i+1}(y) \rightarrow \mathsf{Eq}_{i+1}(x, y)$

2. ($\mathsf{AllTiledRow}$)

   For $j, k \leq r$ the rules:
   $\mathsf{TiledBy}_j(x_1, y), \mathsf{TiledBy}_k(x_2, y), \mathsf{Eq}_{n-1}(x, x_1), \mathsf{Eq}_{n-1}(x, x_2), \mathsf{1}_n(x_1), \mathsf{0}_n(x_2) \rightarrow \mathsf{TiledAboveCol}_n(x, y)$

   For $j, k \leq r$ and $2 \leq i \leq n$ the rules:
   $\mathsf{Eq}_{i-1}(x, x_1), \mathsf{Eq}_{i-1}(x, x_2),$
   $\mathsf{TiledAboveCol}_i(x_1, y), \mathsf{TiledAboveCol}_i(x_2, y),$
   $\mathsf{1}_i(x_1), \mathsf{0}_i(x_2) \rightarrow \mathsf{TiledAboveCol}_{i-1}(x, y)$

   $\mathsf{TiledAboveCol}_1(x, y) \rightarrow \mathsf{RowTiled}(y)$

3. ($\mathsf{AllTiled}$)

   $\mathsf{Eq}_{n-1}(y, y_1), \mathsf{Eq}_{n-1}(y, y_2), \mathsf{1}_n(y_1), \mathsf{0}_n(y_2),$
   $\mathsf{RowTiled}(y_1), \mathsf{RowTiled}(y_2) \rightarrow \mathsf{TiledAboveRow}_n(y)$

For each $2 \leq i \leq n$, the rules:
$\mathsf{Eq}_{i-1}(y, y_1), \mathsf{Eq}_{i-1}(y, y_2), \mathsf{1}_i(y_1), \mathsf{0}_i(y_2),$
$\mathsf{TiledAboveRow}_i(y_1), \mathsf{TiledAboveRow}_i(y_2)$
$\rightarrow \mathsf{TiledAboveCol}_{i-1}(y)$

$\mathsf{TiledAboveRow}_1(y) \rightarrow \mathsf{AllTiled}$

$\mathsf{AllTiled} \rightarrow \mathsf{Goal}$

Q'_I is designed so that it fails exactly when one of the constraints on the tiles does not hold. Its intensional predicates are:

- binary predicates $\mathsf{Succ}_i, \mathsf{Eq}_i$ for $i \leq n$. Eq_i is the i-bit equality as before, while Succ_i states that the first i bits are in the successor relation, (hence Succ_n will be the usual successor, which corresponds to the coding of integers in strings we use here).

- nullary predicate Goal, which will be the goal predicate.

The rules for Q'_I include the (Equality) rules as in Q_I, and also:

1. (ValuesDisjoint)

   For each $i \leq n$ the rule:
   $\mathsf{0}_i(x), \mathsf{1}_i(x) \rightarrow \mathsf{Goal}$

2. (Successor)

   $\mathsf{0}_1(x), \mathsf{1}_1(y) \rightarrow \mathsf{Succ}_1(x, y)$

   For each $2 \leq i \leq n$, the rules:
   $\mathsf{1}_1(x), \dots, \mathsf{1}_{i-1}(x), \mathsf{0}_i(x), \mathsf{0}_1(y), \dots, \mathsf{0}_{i-1}(y), \mathsf{1}_i(y)$
   $\rightarrow \mathsf{Succ}_i(x, y)$
   $\mathsf{Succ}_{i-1}(x, y), \mathsf{1}_i(x), \mathsf{1}_i(y) \rightarrow \mathsf{Succ}_i(x, y)$
   $\mathsf{Succ}_{i-1}(x, y), \mathsf{0}_i(x), \mathsf{0}_i(y) \rightarrow \mathsf{Succ}_i(x, y)$

3. (TileConsistency)

   For each $i \neq j \leq r$, we have the rule:
   $\mathsf{Eq}_n(x, x'), \mathsf{Eq}_n(y, y'), \mathsf{TiledBy}_i(x, y), \mathsf{TiledBy}_j(x', y') \rightarrow$ Goal

4. (TileCompatibility)

   For each $(i, j) \notin V$ we have a rule:
   $\mathsf{Succ}_n(y, y'), \mathsf{TiledBy}_i(x, y), \mathsf{TiledBy}_j(x, y') \rightarrow \mathsf{Goal}$

   and for each $(i, j) \notin H$ we have a rule:
   $\mathsf{Succ}_n(x, x'), \mathsf{TiledBy}_i(x, y), \mathsf{TiledBy}_j(x', y) \rightarrow \mathsf{Goal}$

5. (InitialTile)

   For each $j \leq n$, let $f_j \in 2^n$ be such that $\mathsf{Bin}(f_j) = j$ and let k be a number in $[1, r]$ other than $w(j)$ (recall that $w(1) \dots w(n)$ is part of the ExpTile instance I). Then we have the rule:
   $\mathsf{0}_1(y), \dots \mathsf{0}_n(y), A_1(x), \dots, A_n(x), \mathsf{TiledBy}_k(x, y) \rightarrow \mathsf{Goal}$

   where predicate A_i is $\mathsf{0}_i$ if $f_j(i) = 0$ and $\mathsf{1}_i$ if $f_j(i) = 1$.

We show in the appendix that the mapping from I to Q_I, Q'_I given above is a polynomial time reduction from ExpTile to the complement of NRDLCon, which proves Theorem 3. □

4. THE PERSISTENCE OF HARDNESS

We now ask whether simplifications to the queries might allow hardness to dissipate.

Our previous construction used input schemas of arbitrary arity. We now show that the problem remains hard even for fixed input schema.

THEOREM 5. *There is a fixed input signature, such that* NRDLCon *restricted to this signature remains* co-NEXPTIME-*hard. For hardness the intensional predicates can be restricted to be binary.*

Proof Sketch. The same proof technique is used as in Theorem 3, but now with fixed signature consisting of predicates: $\mathsf{1}(i), \mathsf{0}(i), \mathsf{NumSucc}(i, j), \mathsf{BitSet}(x, i, b), \mathsf{TiledBy}(x, y, i)$.

Think of the domain as consisting of co-ordinates (which will represent exponentially-sized numbers) and "small" (poly-sized) numbers. Using 1 and NumSucc, we will be able to define the "small" numbers $1 \dots max(n, r)$ using intensional predicates. The predicate $\mathsf{BitSet}(x, i, b)$ will relate a co-ordinate x, a small number i and b that is either 0 or 1: intuitively, it states that the i^{th} bit of x is b. $\mathsf{TiledBy}(x, y, i)$ will indicate that the cell with co-ordinates (x, y) is tiled by tile i, where i is again a small number. Details are in the appendix. □

The next result states that hardness still holds when we restrict the input signature to be *monadic* (all predicates have arity at most 1).

THEOREM 6. NRDLCon *restricted to signatures for Q in which input predicates are monadic is* co-NEXPTIME-*complete.*

Proof Sketch. The input predicates are now:

- $\mathsf{0^x}_i(x), \mathsf{1^x}_i(x), \mathsf{0^y}_i(x), \mathsf{1^y}_i(x)$ for $i \leq n$
- $\mathsf{TiledBy}_i(x)$ for every $i \leq r$

Here the variation on the idea of Theorem 3 is quite simple. In Theorem 3 the elements of the domain of a model were co-ordinates, and a cell was represented by a pair of domain elements. Here domain elements will represent cells. Each cell will be associated with predicates $\mathsf{0^x}_i(x), \mathsf{1^x}_i(x)$ describing the i^{th} bit of the x coordinate of the cell, while the predicates $\mathsf{0^y}_i(x), \mathsf{1^y}_i(x)$ describe the i^{th} bit of the y coordinate. The details are given in the appendix. □

An obvious question is whether the complexity of containment is due to the use of iterated view definitions in Q, or Q', or both. Towards resolving this, we show that the problem is still just as hard when Q' is restricted to be merely a union of conjunctive queries. The proof is in the appendix.

THEOREM 7. *The following problem is* co-NEXPTIME-*hard: given Q an* NRDL *query and Q' a* $\exists^+$FO *query, determine whether Q is contained in Q'. Hardness holds even when Q' is a conjunctive query.*

The previous results show that restricting the right-hand side does not help matters. Some restrictions on the left-hand side query do not help either – for example, restricting Q to have only monadic intensional predicates. The following is proven in the appendix:

THEOREM 8. NRDLCon *restricted to signatures for Q in which all intensional predicates are monadic is* co-NEXPTIME-*complete.*

5. REDUCING THE COMPLEXITY

In contrast to our previous results, we demonstrate one way of reducing the complexity of containment – by restricting the shape of Q.

Recall again the doubly-exponential succinctness of NRDL queries with respect to UCQ's (Proposition 2). This is caused by combining two different exponential blow-ups. The use of *sharing*, due to repetition of literals in rules blows up the size of individual canonical dbs of Q. The use of *nesting of disjunctions and conjunctions* in Q, as in $\exists^+$FO, allows there to be *exponentially many* canonical databases. We examine limiting these features to lower the complexity below.

5.1 Restricting sharing in Q

A simple argument shows that containment is in PSPACE when the NRDL query Q is *linear* – i.e. every rule contains at most one intensional atom.

We start with the following:

Theorem 9. NRDLCon *restricted to signatures for Q in which all intensional predicates are linear is* PSPACE*-complete.*

Proof: PSPACE-hardness follows easily from the fact that NRDL-evaluation is PSPACE-hard [25]. We show membership in PSPACE. It suffices to show that there is a polynomial P such that this containment fails iff there is a database D of size $P(|Q|)$ that witnesses the failure, since (again by [25]) we can test whether $Q \wedge \neg Q'$ holds on D in PSPACE.

This follows immediately from the following simple proposition and the corresponding small property for $\exists^+$FO:

Proposition 10. *There is a polynomial time function P that takes a linear* NRDL *query Q and an atom $P(\vec{x})$ and produces the unfolding* $\mathsf{Unf}(P(\vec{x}), Q)$.

A variation of the above argument shows:

Theorem 11. *The problem of determining if $Q \subseteq Q'$ where Q and Q' are linear is Π^p_2-complete. Hardness holds even when Q' is a conjunctive query.*

Proof: To see membership in Π^p_2, note again that if there is a counterexample to containment for such queries, then it can be taken to be of size polynomial in Q. Thus we have to check every possible counterexample, and then evaluate Q' on it. Each individual test can be done in NP, since by the same argument we need only guess polynomially many witnesses (in $|Q'|$) to check. The hardness follows from the result of [22] (see also [1]) that membership of a $\exists^+$FO query in a CQ is Π^p_2-hard. □

5.2 Restricting the use of disjunction in Q

We now examine what happens when the use of disjunction in Q is restricted.

The *conjunctive* NRDL queries (denoted $\wedge$NRDL) are those in which every intensional predicate appears in the head of at most one rule. The unfolding of such a query is a single conjunctive query. The following proposition is not difficult:

Proposition 12. *The problem of determining, given a $\wedge$NRDL query Q and an arbitrary NRDL Q', whether or not Q is contained in Q', is* PSPACE*-complete.*

Proof: Membership in PSPACE follows because the unfolding of Q (considered as a single canonical db) can be produced by a PSPACE-transducer, which can be composed with the PSPACE algorithm for evaluation of Q'. PSPACE-hardness follows from the PSPACE-hardness of NRDL evaluation. □

A variation of the argument above shows that by *simultaneously* restricting disjunction in Q and sharing in Q', we get down to the complexity of classical conjunctive query containment:

Proposition 13. *The problem of determining, given a $\wedge$NRDL query Q and a linear NRDL Q' is NP-complete, and it is NP-hard even when Q' is a conjunctive query.*

Proof: We can unfold Q' to $\exists^+$FO using Proposition 10. Membership in NP follows because every element of the canonical db can be represented in polynomial space; thus for any $\exists^+$FO Q' we can guess a polynomial space representation of at most $|Q'|$ elements of the canonical db, a conjunctive query in the flattening of Q', and a homomorphism of the conjunctive query to these elements. To check that a mapping is a homomorphism into the canonical db can be done efficiently, since it requires only looking at the part of the db containing the paths to those elements.

NP-hardness follows because this subsumes containment of conjunctive queries. □

Pragmatics of $\wedge$NRDL containment. Let us look at the case where the left-hand side is in $\wedge$NRDL in more detail. We know it is in PSPACE, and hence in principle reducible in polynomial time to the PSPACE-complete problem of NRDL evaluation. We now give an explicit linear-time reduction. Although this does not improve the worst-case bound, we believe it will be useful in practice. It allows us to reduce containment in this case to evaluation of an NRDL program, for which there are many optimisation techniques available (e.g. magic set and Query/Subquery transformations).

Let Q be a $\wedge$NRDL query. Let k be the rank of Q, and assume that each rule uses a distinct set of variable names. An element of the unique canonical db of $\mathsf{Unf}(Q)$ consists of a sequence of pairs $(r_1, v_1), \dots (r_j, v_j)$ for $j \leq k$ where v_i is a variable of rule r_i and r_i is associated with a head predicate contained in rule r_{i-1} for $i > 1$. A vector of such sequences $s^1 \dots s^k$ satisfies input predicate $A(s^1 \dots s^k)$ in the canonical db if there are variables $v_1 \dots v_k$ with v_i in s^i and $A(v_1 \dots v_k)$ occurs in some rule.

Of course, the canonical db is big, and thus we can not produce it in its entirety efficiently. But we can efficiently produce an NRDL program that in turn produces the canonical db:

Claim 14. *There is a linear-time program v that takes as input an $\wedge$NRDL program Q with input predicates $R_1 \dots R_j$ and forms an impure NRDL program Q^* over the empty input signature, which has distinguished intensional predicates $Canon_R$ of arity $m \cdot k$ for every input predicate R of arity m in Q (for some k linear in the size of Q), with the following property: Q^* will produce relations $Canon_{R_1}, \dots Canon_{R_n}$ isomorphic to the interpretation of $R_1 \dots R_n$ in the canonical database of Q.*

The idea is to just "invert the rules of Q". Officially, we work inductively top-down on Q. As an example consider the program with rules:
$R_1 : R(x, z), R(z, y), T(z, z) \to Q(x, y)$
$R_2 : S(u, w), S(w, v) \to R(u, v)$
$R_3 : S(r, s), S(s, t) \to T(r, s)$

We first generate a rule that creates the canonical db for the program consisting only of R_1, with R and T considered as input predicates. This will have rules:

$\rightarrow Canon_R(c_x, c_z)$, $\rightarrow Canon_R(c_z, c_y)$, and $\rightarrow Canon_T(c_z, c_z)$

Here $Canon_R, Canon_T$ are intensional predicates and c_x, c_z, c_y are constants.

Continuing top-down to incorporate rule R_2, we generate two rules:

$Canon_R(u, v) \rightarrow Canon_S(c_\flat, c_\flat, u, u, v, c_w)$

$Canon_R(u, v) \rightarrow Canon_S(u, v, c_w, c_\flat, c_\flat, u)$

Here $c_\flat$ is a constant. Informally, this says that the canonical db now interprets S by pairs of triples (i.e. a 6-tuple). These triples are formed from any names u and v in the canonical db's interpretation for R; we create triples by padding u and v by a new constant $c_\flat$ (e.g. the triple $c_\flat, c_\flat, u$ is the padding of u), and also triples formed from u and v by appending a new constant c_w.

Continuing to rule R_3, we create the rules:

$Canon_T(r, s) \rightarrow Canon_S(c_\flat, c_\flat, r, c_\flat, c_\flat, s)$

$Canon_T(r, s) \rightarrow Canon_S(c_\flat, c_\flat, s, r, s, c_t)$

One can verify that this program produces a $Canon_S$ that is isomorphic to the interpretation of S in the canonical db of the NRDL query $Q(x, y)$. The general procedure requires additional case distinctions, mostly related to padding predicates that occur at different levels.

Note that this program makes heavy use of constants in the head, and that it introduces disjunction. The procedure is used to show the following:

THEOREM 15. *Given* $\wedge$NRDL Q *and* NRDL Q' *we can produce in linear time an impure input-free* NRDL *boolean query* $Prod(Q^*, Q')$ *such that* $Prod(Q^*, Q')$ *evaluates to true iff* Q *is contained in* Q'.

The theorem is proven by unioning the rules of the program Q^* produced by the claim above for Q with the modification of Q' which we now describe. Let k be such that an atomic predicate A of arity m in in the original signature corresponds to an intensional predicate $Canon_A$ of arity $m \cdot k$ (such a k is linear in the depth of Q). Replace every free or bound variable x in Q' by a vector of k variables $\vec{x}$, and every input predicate $A(x, y)$ of Q' by the intensional predicate $Canon_A(\vec{x}, \vec{y})$. The correctness of this construction follows from Claim 14 above, along with the fact that Q is contained in Q' iff Q' evaluates to true on the canonical db of Q. $\Box$

5.3 Fixing parameters in the problem

We now look at another method to gain tractability, fixing one of the parameters of the problem. We have seen that fixing the input signature is not sufficient to decrease complexity. We thus consider fixing either Q or Q'.

If Q is fixed, then Q can be taken to be a UCQ. By the homomorphism theorem of Chandra and Merlin [5], containment of Q in Q' reduces to evaluation of Q' on every canonical database of Q. Since the number of canonical databases is fixed with Q, we have a reduction to the evaluation problem of NRDL, which is known to be in PSPACE [25]. On the other hand, the problem of evaluating an NRDL program on a fixed model is known to be PSPACE-hard [25]. Using this we see that:

PROPOSITION 16. *For any fixed* NRDL Q*, the problem of determining whether or not* Q *is contained in* NRDL Q' *is in PSPACE. There are* NRDL *queries* Q *such that the problem is PSPACE-hard.*

We now turn to the situations where the target query Q' is fixed. We show the following:

THEOREM 17. *For any fixed* NRDL *query* Q'*, the problem of checking whether or not* NRDL Q *is contained in* Q' *is* PSPACE*-complete.*

In the appendix we explain an alternating polynomial time algorithm that searches for a counterexample canonical db of Q satisfying $\neg Q'$. This could be implemented deterministically using a stack-based algorithm, where the stack contains guesses for the rules generating the currently-examined portion of the witness canonical db.

6. RELATED WORK

As mentioned in Section 2, work on the query containment problem began with the seminal paper of Chandra and Merlin [5] for conjunctive queries. The homomorphism technique introduced there was extended to unions of conjunctive queries by Sagiv and Yannakakis [22], yielding the same complexity bound. Later work analyzed the complexity for richer languages, including extensions of unions of conjunctive queries with inequalities [18], safe negation [24, 19], and regular expressions [11]. More recent work has looked at containment and equivalence for queries involving even more advanced features of SQL, including nesting [16] and aggregation [8]. But view definitions are not addressed in any of these works.

The impact of *recursive* view definitions on containment has received considerable attention. After Shmueli's initial undecidability result for Datalog equivalence [23], Bonatti et. al. [3] and Calvanese et. al [2] traced the borderline of decidability. Chaudhuri and Vardi [6, 7] studied the relationship of recursive and nonrecursive Datalog extensively. However, none of their results gives a tight bound in the nonrecursive case. Similarly Dong and Su [10] consider containment problems in a setting that generalizes the one here, considering more complex view definitions (admitting recursion) as well as restrictions on predicates via constraints. Due to the increased power of the formalism, they show that the most general case of containment is undecidable. They isolate collections of constraints that admit lower complexity, but restrictions on the view definitions are not explored.

Oddly, even in very exhaustive surveys of the complexity of logic programming [9] the complexity of equivalence and containment for nonrecursive Datalog is not touched upon.

7. SUMMARY AND CONCLUSIONS

In this paper, we have given the first full analysis of the complexity of the most basic query language with simple view definitions – nonrecursive Datalog. The table below gives a picture of the main results (i.e. excluding our results for fixed input signature and for monadic queries). In the table Arb stands for arbitrary signature for intensional predicates, Lin stands for linear rules, Fixed stands for fixed signature (not necessarily monadic) for intensional predicates, and CQ means that we restrict the query to be a conjunctive query. The "reference" column points to the result that proves completeness. Note that the PSPACE result for arbitrary NRDL Q and fixed Q' implies a polynomial time bound when we restrict further – so we omit the last three rows of the table. Two results have no reference: the upper bound in (1) follows since evaluation of linear NRDL is in NP. The lower bound follows since this still subsumes conjunctive query containment. In (2) the lower bound follows since the query complexity of CQ's is NP-hard, while the upper follows as in (1).

Q	Q'	complexity	reference
Arb	Arb	co-NEXPTIME	Thm 3
Lin	Arb	PSPACE	Thm 9
$\wedge$NRDL	Arb	PSPACE	Prop 12
CQ	Arb	PSPACE	Derivable from [25]
Fixed	Arb	PSPACE	Derivable from [25]
Arb	Lin	co-NEXPTIME	Thm 7
Lin	Lin	Π_2^p	Thm 11
$\wedge$NRDL	Lin	NP	Prop 13
CQ	Lin	NP	See note (1)
Fixed	Lin	NP	See note (2)
Arb	CQ	co-NEXPTIME	Thm 7
Lin	CQ	Π_2^p	Thm 11
$\wedge$NRDL	CQ	NP	Prop 13
CQ	CQ	NP	[5]
Fixed	CQ	NP	[5]
Arb	Fixed	PSPACE	Thm 17

Table 1: Complexity of Containment

We intend to revisit several applications of nonrecursive Datalog in the data integration literature in light of our results. For example, [21] makes use of a variant of NRDL containment in tackling questions on querying with materialized views. It would be interesting to compare our results with the much lower complexity bounds of [21] (e.g. Theorem 3.2) to understand the distinction between the two scenarios.

In the case of $\wedge$NRDL Q and "small" Q', we believe the algorithms we provide here – although PSPACE in the worst case – show promise. In particular the $\wedge$NRDL algorithms can be implemented via standard Datalog evaluators. Although the worst-case bounds that we establish here for the general problem are daunting, we are investigating heuristics that exhibit low complexity in common cases, and that could be shown complete for restricted classes of queries. One particularly interesting direction is to devise a generalization of acyclic queries to the context of NRDL that implies tractability of the containment problem (for the evaluation problem, requiring acyclicity in each local rule is sufficient to achieve tractability [12]).

8. REFERENCES

[1] M. Benedikt, G. Puppis, and H. Vuy. Positive Higher Order Queries. In *PODS*, 2010. To appear.

[2] P. A. Bonatti. On the decidability of containment of recursive datalog queries. In *PODS*, 2004.

[3] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Decidable containment of recursive queries. *Theor. Comput. Sci.*, 336(1):33–56, 2005.

[4] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query containment. In *PODS*, 2003.

[5] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.

[6] S. Chaudhuri and M. Y. Vardi. On the complexity of equivalence between recursive and nonrecursive Datalog programs. In *PODS*, 1994.

[7] S. Chaudhuri and M. Y. Vardi. On the equivalence of recursive and nonrecursive Datalog programs. *JCSS*, **54**(1):61–78, 1997.

[8] S. Cohen, W. Nutt, and Y. Sagiv. Deciding equivalences among conjunctive aggregate queries. *J. ACM*, 54(2):5, 2007.

[9] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. *ACM Comp. Sur.*, **33**(3):374–425, Sept. 2001.

[10] G. Dong and J. Su. Conjunctive query containment with respect to views and constraints. *Inf. Process. Lett.*, 57(2):95–102, 1996.

[11] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *PODS*, 1998.

[12] J. Flum, M. Frick, and M. Grohe. "Query Evaluation via Tree-Decompositions". *Journal of the ACM*, **49**(6):716–752, 2002.

[13] G. Gottlob and C. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1), 2003.

[14] A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. Constraint checking with partial information. In *PODS*, 1994.

[15] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[16] T. S. Jayram, P. G. Kolaitis, and E. Vee. The containment problem for real conjunctive queries with inequalities. In *PODS*, 2006.

[17] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theor. Comp. Sci.* MIT Press, 1990.

[18] A. Klug. On Conjunctive Queries Containing Inequalities. *Journal of the ACM*, **35**(1):146–160, Jan. 1988.

[19] G. Lausen and F. Wei. On the containment of conjunctive queries. In *Computer Science in Perspective*, 2003.

[20] A. Y. Levy and Y. Sagiv. Queries independent of updates. In *VLDB*, 1993.

[21] T. Millstein, A. Halevy, and M. Friedman. Query containment for data integration systems. *JCSS*, 66(1):20–39, 2003.

[22] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *JACM*, **27**(4):633–655, 1980.

[23] O. Shmueli. Equivalence of datalog queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.

[24] J. D. Ullman. Information integration using logical views. In *ICDT*, 1997.

[25] S. Vorobyov and A. Voronkov. Complexity of nonrecursive logic programs with complex values. In *PODS*, 1998.

9. APPENDIX

9.1 Proof of Theorem 3

We refer to the construction of Q_I, Q'_I from an $\mathsf{ExpTile}$ instance I given in the body of the paper. Clearly the function giving the reduction in the previous subsection can be implemented in polynomial time. We show that Q_I is not contained in Q'_I iff I is a yes instance of $\mathsf{ExpTile}$.

Suppose that F is a solution to the tiling problem I. It is easy to construct a D that satisfies $Q_I \wedge \neg Q'_I$:

- Let the elements of 2^n be the domain of the model.
- Let $\mathsf{0}_i(f)$ hold if $f(i) = 0$ and $\mathsf{1}_i(f)$ hold if $f(i) = 1$
- Let $\mathsf{TiledBy}_i(f, g)$ hold iff $F(f, g) = i$

It is easy to verify that the rules of Q_I hold and that Q'_I does not hold.

Suppose now that D is a database that satisfies $Q_I \wedge \neg Q'_I$. We show how to construct a tiling from D.

Let $\mathsf{Indexed}(D)$ be the set of $x \in D$ such that for each $i \leq n$ either $\mathsf{0}_i(x)$ holds or $\mathsf{1}_i(x)$ holds.

Note that it is clear that: no x can satisfy both $\mathsf{1}_i(x)$ and $\mathsf{0}_i(x)$, by Rule ($\mathsf{ValuesDisjoint}$). Let $\mathsf{Hom}(x)$ associate each $x \in \mathsf{Indexed}(D)$ with the unique function f in 2^n such that $\mathsf{1}_i(x)$ holds iff $f(i) = 1$.

For $f, g \in 2^n$, let $f \equiv_i g$ mean that $f(j) = g(j)$ for all $j \in [1, i]$. We now note the following:

Claim 18. *Let Eq_i be the intensional predicate in either Q or Q'. For any $x, y \in \mathsf{Indexed}(D)$ $\mathsf{Eq}_i(x, y)$ holds iff $\mathsf{Hom}(x) \equiv_i \mathsf{Hom}(y)$.*

The proof of the claim is a straightforward induction, using Rules ($\mathsf{Equality}$) in Q and Q'. It follows from the claim that $\forall x, y \in \mathsf{Indexed}(D)$ $\mathsf{Eq}_1(x, y) \leftrightarrow \mathsf{Hom}(x) = \mathsf{Hom}(y)$.

Let $\mathsf{TiledPair}(D)$ be the set of $x, y \in \mathsf{Indexed}(D)$ such that $\mathsf{TiledBy}_j(x, y)$ holds for some j.

Let $\mathsf{DomTiling}(D)$ be the function mapping $(x, y) \in \mathsf{TiledPair}(D)$ to the unique j such that $\mathsf{TiledBy}_j(x, y)$ holds.

By rule ($\mathsf{TileConsistency}$) and Claim 18, we see not only is there a unique such j (i.e. the function above is well-defined), but that: if $\mathsf{Hom}(x) = \mathsf{Hom}(x')$, $\mathsf{Hom}(y) = \mathsf{Hom}(y')$ and $\mathsf{TiledBy}_j(x, y)$ holds, then we cannot have $\mathsf{TiledBy}_k(x', y')$ holding for any $k \neq j$. Thus $\mathsf{DomTiling}$ depends only on the image of each element in the pair under H.

We now claim:

Claim 19. *For every $f, g \in 2^n$, there are $x, y \in \mathsf{TiledPair}(D)$ with $\mathsf{Hom}(x) = f, \mathsf{Hom}(y) = g$*

To prove Claim 19, we prove by downward induction on $i \leq n$ that for every x, y satisfying $\mathsf{TiledAboveCol}_i(x, y)$ for every $f \in 2^n$ such that $f \equiv_{i-1} \mathsf{Hom}(x)$ there is some x' with $\mathsf{Hom}(x') = f$ and $(x', y) \in \mathsf{TiledPair}(D)$.

We first show the base case $i = n$. If $\mathsf{TiledAboveCol}_n(x, y)$ holds, this can only be derived from the first set of rules in Ruleset ($\mathsf{AllTiledRow}$). This asserts the existence of x_1 and x_2 such that:

- $(x_1, y), (x_2, y) \in \mathsf{TiledPair}(D)$,
- $\mathsf{Eq}_{n-1}(x, x_1), \mathsf{Eq}_{n-1}(x, x_2)$,
- $\mathsf{Hom}(x_1)$ has n^{th} bit 0 and $\mathsf{Hom}(x_1)$ has n^{th} bit 1.

The desired conclusion now follows using Claim 18. For the inductive step, we note that if $\mathsf{TiledAboveCol}_i(x, y)$ holds, it must follow from application of a rule in the second set in ($\mathsf{AllTiledRow}$). Hence there are x_1 and x_2 such that:

- $\mathsf{Eq}_{i-1}(x, x_1), \mathsf{Eq}_{i-1}(x, x_2)$,
- $\mathsf{TiledAboveCol}_i(x_1, y), \mathsf{TiledAboveCol}_i(x_2, y)$,
- $\mathsf{Hom}(x_1)$ has i^{th} bit 0 and $\mathsf{Hom}(x_2)$ has i^{th} bit 1.

By Claim 18 $x_1, x_2 \equiv_{i-1} x$. If $f \in 2^n$ with $f \equiv_{i-1} \mathsf{Hom}(x)$, then $f \equiv_i \mathsf{Hom}(x_1)$ or $f \equiv_i \mathsf{Hom}(x_2)$. Hence the conclusion follows by induction.

From the above lemma it follows, using the last rule of Rules ($\mathsf{AllTiledRow}$), that for every x satisfying $\mathsf{ColumnTiled}$ (y) and for every $f \in 2^n$ there is some x with $\mathsf{Hom}(x) = f$ and $(x, y) \in \mathsf{TiledPair}(D)$.

We now prove by downward induction on i that for every y satisfying $\mathsf{TiledAboveRow}_i$, for every $f \in 2^n$ such that $f \equiv_i \mathsf{Hom}(y)$ there is some y' with $\mathsf{Hom}(y') = f'$ and such that $\mathsf{RowTiled}(y)$ holds.

The base case for $i = n$ follows from the first rule in ($\mathsf{AllTiled}$), while the inductive case follows from the second set of rules in Ruleset ($\mathsf{AllTiled}$).

Now, using the final rule in ($\mathsf{AllTiled}$) and the two facts above, we see that:

If predicate $\mathsf{AllTiled}$ holds in a database D, then the conclusion of Claim 19 holds. But D was assumed to satisfy goal predicate Goal of query Q, and thus $\mathsf{AllTiled}$ must hold.

This completes the proof of Claim 19 □

From Claim 19 it follows that the function H maps the domain of the function $\mathsf{DomTiling}$ onto 2^n. Thus we can define a function $\mathsf{CellTiling}(D)$ that maps a pair $(f, g) \in 2^n \times 2^n$ to the unique $\mathsf{TiledBy}_j$ such that there is are (x, y) with $\mathsf{Hom}(x) = f, \mathsf{Hom}(y) = g$ and $\mathsf{DomTiling}(x, y) = \mathsf{TiledBy}_j$.

We next claim:

Claim 20. *The relation $\mathsf{Succ}_n(x, y)$ holds in D iff $\mathsf{Bin}(\mathsf{Hom}(y))$ is the successor of $\mathsf{Bin}(\mathsf{Hom}(y))$.*

Proof: For $f \in 2^n$ and $i \leq n$ let $\restriction_i (f)$ be the restriction of f to $[1, i]$, seen as a string.

We show by induction that for $i \leq n$ $\mathsf{Succ}_i(x, y)$ holds in D iff $\mathsf{Bin}(\restriction_i (\mathsf{Hom}(y)))$ is the successor of $\mathsf{Bin}(\restriction_i (\mathsf{Hom}(x)))$.

For $i = 1$ this is clear from the first rule in Ruleset ($\mathsf{Successor}$). For the induction step, we see from the second set of rules in Ruleset ($\mathsf{Successor}$) that $\mathsf{Succ}_i(x, y)$ holds in D iff one of the following holds:

- $x = 1^{i-1}.0$ and $y = 0^{i-1}.1$. Clearly in this case y is the successor of x.
- $\mathsf{Succ}_{i-1}(x, y)$, and the $(i)^{th}$ bits are the same. We conclude that y is a successor of x via induction and the rules of addition.

It is likewise easy to show by induction that if the first i bits of $\mathsf{Hom}(y)$ are the binary successor of the corresponding bits of $\mathsf{Hom}(x)$, then the antecedent of one of the above rules must hold, and hence $\mathsf{Succ}_i(x, y)$ holds in D. □

We now claim that $\mathsf{CellTiling}(D)$ is a tiling.

The fact that the vertical and horizontal constraints are satisfied follows immediately from Rules ($\mathsf{TileCompatibility}$), given Claim 20. The fact that the initial tiles are respected follows from Rules ($\mathsf{InitialTile}$). □

9.2 Proof of Theorem 5

Recall the statement of the result: There is a fixed input signature, such that NRDLCon restricted to this signature remains co-NEXPTIME-hard. For hardness the intensional predicates can be restricted to be binary.

For convenience we will give the proof with a signature including a ternary predicate. The extension where all predicates are binary is straightforward.

Again fix I to be an instance of the exponential tiling problem. Recall that our input signature will contain predicates:

$\mathsf{1}(x), \mathsf{0}(x), \mathsf{NumSucc}(x, y), \mathsf{BitSet}(x, y, z), \mathsf{TiledBy}(x, y, z)$.

Note that, as promised, the signature is now independent of the instance I. The intensional predicates for Q_I will be as before, but will also include predicates

- $\mathsf{IsNum}_i(x)$ for every $i \leq n$
- $\mathsf{0}_i(x), \mathsf{1}_i(x)$ for every $i \leq n$
- $\mathsf{TiledBy}_i(x, y)$ for every $i \leq r$

The idea will be that IsNum_i is true of elements that play the role of number i. Predicates $\mathsf{TiledBy}_i, \mathsf{0}_i, \mathsf{1}_i$ will be forced to have the same role as the input predicates with those names in the proof of Theorem 3.

This will be achieved by having the following rules in Q_I:

- (NumberDefinition)
  $\mathsf{1}(x) \to \mathsf{IsNum}_1(x)$
  $\mathsf{NumSucc}(x, y), \mathsf{IsNum}_i(y) \to \mathsf{IsNum}_{i+1}(x)$
- (BitSettingPredicates)
  $\mathsf{BitSet}(x, y, z), \mathsf{IsNum}_i(y), \mathsf{0}(z) \to \mathsf{0}_i(x)$
  $\mathsf{BitSet}(x, y, z), \mathsf{IsNum}_i(y), \mathsf{1}(z) \to \mathsf{1}_i(x)$
- (TilingPredicates)
  $\mathsf{TiledBy}(x, y, z), \mathsf{IsNum}_i(z) \to \mathsf{TiledBy}_i(x, y)$

In addition, Q_I will contain the rules from Theorem 3, with intensional predicates $\mathsf{TiledBy}_i, \mathsf{0}_i, \mathsf{1}_i$ replacing the extensional predicates.

Q'_I will have the rules above and also those from Theorem 3.

The reader can easily verify that the mapping from I to Q_I, Q'_I is a polynomial time reduction.

9.3 Proof of Theorem 6

Recall the statement: NRDLCon restricted to signatures for Q in which input predicates are monadic is co-NEXPTIME-complete.

Q_I will have intensional predicates:

- binary predicates $\mathsf{Eqx}_i, \mathsf{Eqy}_i$ for $i \leq n$
- binary predicates $\mathsf{TiledAboveCol}_i$ for $i \leq n$ and unary predicate $\mathsf{RowTiled}$
- unary predicates $\mathsf{TiledAboveRow}_i$ for $i \leq n$ and nullary predicate $\mathsf{AllTiled}$
- 0-ary predicate Goal, which will be the goal predicate.

We give the rules for Q_I, using $c, c', c_1, c_2, \ldots$ as variables, in order to suggest cells rather than co-ordinates:

1. (Equality)
   $\mathsf{0^x}_1(c), \mathsf{0^x}_1(c') \to \mathsf{Eqx}_1(c, c')$
   $\mathsf{1^x}_1(c), \mathsf{1^x}_1(c') \to \mathsf{Eqx}_1(c, c')$
   $\mathsf{0^y}_1(c), \mathsf{0^y}_1(c') \to \mathsf{Eqy}_1(c, c')$
   $\mathsf{1^y}_1(c), \mathsf{1^y}_1(c') \to \mathsf{Eqy}_1(c, c')$

   For $1 \leq i \leq n-1$ the rules:

   $\mathsf{Eqx}_i(c, c'), \mathsf{0^x}_{i+1}(c), \mathsf{0^x}_{i+1}(c') \to \mathsf{Eqx}_{i+1}(c, c')$
   $\mathsf{Eqx}_i(c, c'), \mathsf{1^x}_{i+1}(c), \mathsf{1^x}_{i+1}(c') \to \mathsf{Eqx}_{i+1}(c, c')$
   $\mathsf{Eqy}_i(c, c'), \mathsf{0^y}_{i+1}(c), \mathsf{0^y}_{i+1}(c') \to \mathsf{Eqy}_{i+1}(c, c')$
   $\mathsf{Eqy}_i(c, c'), \mathsf{1^y}_{i+1}(c), \mathsf{1^y}_{i+1}(c') \to \mathsf{Eqy}_{i+1}(c, c')$

2. (AllTiledRow)
   For $j, k \leq r$ the rules:
   $\mathsf{TiledBy}_j(c_1), \mathsf{TiledBy}_k(c_2), \mathsf{Eqy}_n(c, c_1), \mathsf{Eqx}_{n-1}(c, c_1), \mathsf{Eqy}_n(c, c_2), \mathsf{Eqx}_{n-1}(c, c_2), \mathsf{1^x}_n(c_1), \mathsf{0^x}_n(c_2)$
   $\to \mathsf{TiledAboveCol}_n(c)$
   For $j, k \leq r$ and $2 \leq i \leq n$ the rules:
   $\mathsf{Eqy}_n(c, c_1), \mathsf{Eqy}_n(c, c_2),$
   $\mathsf{Eqx}_{i-1}(c, c_1), \mathsf{Eqx}_{i-1}(c, c_2),$
   $\mathsf{TiledAboveCol}_i(c_1), \mathsf{TiledAboveCol}_i(c_2),$
   $\mathsf{1^x}_i(c_1), \mathsf{0^x}_i(c_2) \to \mathsf{TiledAboveCol}_{i-1}(c)$
   $\mathsf{TiledAboveCol}_1(c) \to \mathsf{RowTiled}(c)$
3. (AllTiled)
   $\mathsf{Eqy}_{n-1}(c, c_1), \mathsf{Eqy}_{n-1}(c, c_2), \mathsf{1^y}_n(c_1), \mathsf{0^y}_n(c_2),$
   $\mathsf{RowTiled}(c_1), \mathsf{RowTiled}(c_2) \to \mathsf{TiledAboveRow}_n(c)$
   For each $2 \leq i \leq n$, the rules:
   $\mathsf{Eqy}_{i-1}(c, c_1), \mathsf{Eqy}_{i-1}(c, c_2), \mathsf{1^y}_i(c_1), \mathsf{0^y}_i(c_2),$
   $\mathsf{TiledAboveRow}_i(c_1), \mathsf{TiledAboveRow}_i(c_2)$
   $\to \mathsf{TiledAboveRow}_{i-1}(c)$
   $\mathsf{TiledAboveRow}_1(x) \to \mathsf{AllTiled}$
   $\mathsf{AllTiled}\ () \to \mathsf{Goal}$

Q'_I will have intensional predicates:

- binary predicates $\mathsf{Succx}_i, \mathsf{Eqx}_i, \mathsf{Succy}_i, \mathsf{Eqy}_i$ for $i \leq n$
- nullary predicate Goal, which will be the goal predicate.

The rules for Q'_I are modified in a similar way:

1. (ValuesDisjoint)
   For each $i \leq n$ the rule:
   $\mathsf{0^x}_i(c), \mathsf{1^x}_i(c) \to \mathsf{Goal}$
   $\mathsf{0^y}_i(c), \mathsf{1^y}_i(c) \to \mathsf{Goal}$
2. (Successor)
   The rules (Equality) as in Q_I, and in addition rules for Succx_i and Succy_i that are copies of the (Successor) for Succ_i in Theorem 3.
3. (TileConsistency)
   For each $i \neq j \leq r$, we have the rule:
   $\mathsf{Eqx}_n(c, c'), \mathsf{Eqy}_n(c, c'), \mathsf{TiledBy}_i(c), \mathsf{TiledBy}_j(c') \to \mathsf{Goal}$
4. (TileCompatibility)
   For each $(i, j) \notin H$ we have a rule:
   $\mathsf{Succx}_n(c, c'), \mathsf{Eqx}_n(c, c'), \mathsf{TiledBy}_i(c), \mathsf{TiledBy}_j(c') \to \mathsf{Goal}$
   and for each $(i, j) \notin V$ we have a rule
   $\mathsf{Succy}_n(c, c'), \mathsf{Eqy}_n(c, c'), \mathsf{TiledBy}_i(c), \mathsf{TiledBy}_j(c') \to \mathsf{Goal}$
5. (InitialTile)
   For each $j \leq n$, let $f_j \in 2^n$ be such that $\mathsf{Bin}(f_j) = j$ and let k be any element $\leq r$ other than $w(j)$. Then we have the rule:
   $\mathsf{0^y}_1(c) \ldots \mathsf{0^y}_n(c), A_1(c), \ldots, A_n(c), \mathsf{TiledBy}_k(c) \to \mathsf{Goal}$
   where predicate A_i is $\mathsf{0^x}_i$ if $f_j(i) = 0$ and $\mathsf{1^y}_i$ if $f_j(i) = 1$

The correctness of the reduction now follows that of Theorem 3. $\square$

9.4 Proof of Theorem 7

Recall the statement:

The following problem is co-NEXPTIME-hard: given Q an NRDL query and Q' a $\exists^+$FO query, determine whether Q is contained in Q'. Hardness holds even when Q' is a conjunctive query.

We will give the proof only in the case where Q' is a UCQ; the extension to eliminate disjunction uses the same idea that is demonstrated below – coding the results of complex boolean formulas using additional "boolean arguments" to predicates; the extension is left to the reader.

We first show why the problem is hard in the case where Q' is a conjunction of UCQs. In this case we can reduce from ExpTile using exactly the same signature as in Theorem 3. For this argument to hold, we need Q'_I to have as atoms the predicates Eq_i for $i \leq n$ defining equality on the first n bits. These do not require views per se, rather each $\mathsf{Eq}_i(x, y)$ can be defined as a UCQ over the input predicates in the obvious way $((\mathsf{0}_1(x) \wedge \mathsf{0}_1(y)) \vee \ldots)$ leading to the conclusion above.

We now show how to modify the proof when Q' is a UCQ.

We use the idea of "coding boolean operations in the input", also exploited in [13, 25]. As mentioned in Section 2, we do make use of constants in this argument, but they can be eliminated by making use of predicates, as described in Section 2.

Again we will be coding an ExpTile instance I. We use the input signature of Theorem 3, but with the following modifications:

- We add ternary input predicates And and Or to the input signature.
- We modify the prior input bit predicates by adding an additional argument, so we now have $\mathsf{0}_j(x, t), \mathsf{1}_j(x, t)$.

The tiling predicates $\mathsf{TiledBy}_i(x, y)$ for $i \leq r$ are left as they are, with no additional argument.

We modify Q_I from Theorem 3 as follows:

- replacing all occurrences of $\mathsf{Eq}_i(x, y)$ with $\mathsf{Eq}_i(x, y, 1)$, $\mathsf{0}_j(x)$ with $\mathsf{0}_j(x, 1)$, and $\mathsf{1}_j(x)$ with $\mathsf{1}_j(x, 1)$.
- adding atomic facts for the truth table of And and Or. That is, we add
  $\mathsf{And}(1, 1, 1), \mathsf{And}(0, 1, 0), \mathsf{And}(1, 0, 0)$
  $\mathsf{Or}(0, 1, 1), \mathsf{Or}(1, 0, 1), \mathsf{Or}(0, 0, 0)$
  to the antecedent of the Goal predicate in Q_I.

Let $\mathsf{And}(x_1, \ldots x_n, v_n)$ abbreviate:
$\mathsf{And}(x_1, x_2, v_1) \wedge \mathsf{And}(v_1, x_3, v_2) \ldots \wedge \mathsf{And}(v_{n-1}, x_n, v_n)$
and similarly for $\mathsf{Or}(x_1 \ldots x_n, v_n)$.

We modify the construction of Q'_I as follows.

Let the rule (PseudoSuccessor) be formed from (Successor) in the proof of Theorem 3, but replacing every atom $\mathsf{Eq}_i(x, y)$ with the sequence of atoms $\mathsf{Eq}'_i(x, y, 1)$ described below. $\mathsf{Eq}'_i(x, y, v)$ has distinguished variables x, y, v and abbreviates:
$\mathsf{0}_1(x, s_1), \mathsf{0}_1(y, s'_1), \mathsf{1}_1(x, t_1), \mathsf{1}_1(y, t'_1), \ldots,$
$\mathsf{0}_i(x, s_i), \mathsf{0}_i(y, s'_i), \mathsf{1}_i(x, t_i), \mathsf{0}_i(y, t'_i),$
$\mathsf{And}(s_1, s'_1, s''_1), \mathsf{And}(t_1, t'_1, t''_1), \mathsf{Or}(s''_1, t''_1, s'''_1), \ldots,$
$\mathsf{And}(s_i, s'_i, s''_i), \mathsf{And}(t_i, t'_i, t''_i), \mathsf{Or}(s''_i, t''_i, s'''_i),$
$\mathsf{Or}(s'''_1, \ldots, s'''_i, v)$

Q'_I will also contain variants of the rules (InitialTile) and (ValuesDisjoint), with the predicates $\mathsf{0}_i, \mathsf{1}_i$ modified to include the additional argument, as in Q_I.

The argument for correctness is as follows.

If we have a tiling, then we can turn it into a model of $Q_I \wedge \neg Q'_I$ in the obvious way, by letting $\mathsf{0}_i(x, 1)$ hold iff the i^{th} bit of x is 0 and similarly for 1, and letting $\mathsf{Eq}_i(x, y, 1)$ hold iff x and y agree on the first i bits. The tiling predicate $\mathsf{TiledBy}_i(x, y)$ holds iff the cell with co-ordinates (x, y) is tiled with tile i.

We now consider the other direction. If we have a model of $Q_I \wedge \neg Q'_I$, then we wish to construct a tiling.

Let (TrueSuccessor) be the modification of the (Successor) where the predicates $\mathsf{0}_i, \mathsf{1}_i$ are modified for the new signature. This is the rule that we would have *liked* to express in Q'_I, but could not do directly since Q'_I could not make use of view definitions.

We can proceed as in Theorem 3, provided that we know that (TrueSuccessor) fails in the model.

If the rule for (TrueSuccessor) fired, then we would have (PseudoSuccessor) holding, using the truth tables for And, Or in Q_I to compute witnesses for the intermediate variables. Thus the model satisfies Q'_I, a contradiction. $\square$

We do not know if hardness holds for Q' a CQ and Q an NRDL query without constants at all.

9.5 Proof of Theorem 8

Recall the statement of the theorem:

NRDLCon restricted to signatures for Q in which all intensional predicates are monadic is co-NEXPTIME-complete.

We will reduce to the exponential tiling problem as before. The input signature has binary predicates $\mathsf{0}_i(x, y), \mathsf{1}_i(x, y)$ for $i \leq n$ and unary predicates $\mathsf{TiledBy}_i(x)$ for $i \leq r$;

Intuitively the models of $Q \wedge \neg Q'$ will represent a node- and edge-labeled tree where the edges are labeled by 0 and 1 while the leaves only are labeled with a color $i < r$. Q will enforce that the depth of the tree is at most $2n$ and will enforce that the graph contains forward paths with all combinations of edge labels (i.e. all binary sequences of length $2n$). A leaf node will represent a tiled element, with the first n edges in the path from the root to the leaf representing the x coordinate of the element, the next n edges representing the y coordinate. Q' will enforce that the graph is a tree, by forbidding cyclic paths, and will also ensure a correct tiling.

Q will have unary intensional predicates U_i, V_i for $i \leq n$, and will have rules:
$V_n(x) \rightarrow \mathsf{Goal}$
$\mathsf{0}(x, y), V_{i-1}(y), \mathsf{1}(x, z), V_{i-1}(z) \rightarrow V_i(x)$ for $1 \leq i \leq n$
$U_n(x) \rightarrow V_0(x)$
$\mathsf{0}(x, y), U_{i-1}(y), \mathsf{1}(x, z), U_{i-1}(z) \rightarrow U_i(x)$ for $1 \leq i \leq n$
$\mathsf{Tiled}_i(x) \rightarrow U_0(x)$ for $i \leq r$.

Q' contains axioms whose negation will guarantee that:

- there is no nontrivial path – an upward path followed by a downward path, where the first downward edge does not match the last upward edge in label – of length at most $2n$ from a node to itself.
- there is no non-leaf node that satisfies a tiling predicate.
- there are no downward paths of size above $2n$.
- there cannot be two paths of size $2n$ with the same edge labels leading to leaves with different tilings.

These rules can be easily encoded in NRDL.

Now it remains for Q' to contain a family of axioms (Successor) whose negation enforces that adjacent cells are tiled correctly. We explain how this works for vertically adjacent pairs below. Using the fourth axiom above, it suffices to assert that we cannot have an intermediate node v at depth n

which has two downward paths p, p' from it where the edge labels of p' are the successor of the edge labels of p, where p and p' lead to leaves with incompatible tiles. So we will have a rule:

$\mathsf{DepthGte}_n(x), \mathsf{Extends}(y_1, x), \mathsf{Extends}(y_2, x),$
$\mathsf{Succ}_y(y_1, y_2, x), \mathsf{Tiled}_i(y_1), \mathsf{Tiled}_j(y_2) \rightarrow \mathsf{Goal}$

for every incompatible pair of tiles i and j, where the first four predicates in the body are intensional, and are defined so as to guarantee the following:

- $\mathsf{DepthGte}_n(x)$ will state that x is at least n from the root.
- $\mathsf{Extends}(z, x)$ will state that there is a downward path from x to z.
- $\mathsf{Succ}_y(y_1, y_2, x)$ will state that the last n edges leading from x to y_1 form a successor of those leading from x to y_2.

All of these are coded straightforwardly in NRDL.

The case for horizontally adjacent nodes is slightly more complicated, since we cannot fix a single stem, but is handled similarly. □

9.6 Proof of Theorem 17

Recall the result:

For any fixed NRDL query Q', the problem of checking whether or not NRDL Q is contained in Q' is PSPACE-complete.

We first give the lower bound. Given any NRDL boolean query Q and input database D for Q we can construct in polynomial time a NRDL query Q^* such that Q is true on D iff Q^* is contained in Q' the empty query – that is, Q is true on D iff Q^* is unsatisfiable. Q^* has an intensional predicate R^* for every input predicate R of Q, and constants for all elements of D. It contains a copy of every rule of Q, modified so that input predicates R are replaced by R^* and the goal predicate of Q is replaced by a new predicate Goal^-. It also contains a rule $\mathsf{Goal}^-, build_1 \ldots build_n \rightarrow \mathsf{Goal}$, where $build_i$ are atoms $R^*(c_1...c_n)$ for every atom $R(c_1 \ldots c_n)$ in D. That is, we build a copy of D in intensional predicates, and then assert that Q holds on this copy. This shows that the containment problem for fixed Q' is at least as hard as the complexity of NRDL evaluation for fixed D. But the query complexity of NRDL evaluation is known to be PSPACE-complete [25].

We now turn to the upper bound. For simplicity, we take Q' to be a CQ here, leaving the UCQ extension to the reader. We can normalize Q so that it has at most two atoms in each body, which are either both intensional or both input predicates. For U an intensional predicate of Q, and Q' any other conjunctive query, a U, Q' subgoal is any conjunction of atoms C from Q' along with a mapping h from a subset of the variables of C to either free variables of U or constants of Q. A U, Q' subgoal family is a finite set of U-subgoals $(C_i, h_i) : i \leq n$. Note that if Q' is fixed the collection of possible C_i's is fixed, along with the number of domains for the mappings h_i. Hence the number of possible h_i's is at most $|Q|^j$ for some j, and therefore the maximal size of a U, Q' subgoal family is polynomial in the size of Q. A U, Q' subgoal family is said to be *omittable* if there is a canonical db D for U in which no h_i cannot be extended to a homomorphism of C_i onto D.

We create an alternating PTIME algorithm that takes as input the program Q, intensional predicate U of Q, and a U, Q' subgoal family and determines whether it is omittable. This can then be applied to the subgoal family consisting of one subgoal, the full query Q' and h the empty mapping onto the goal predicate of Q.

Given U, Q' subgoal family $(C_i, h_i) : i \leq n$, the algorithm guesses a rule $U(\vec{x}) \rightarrow B_1(\vec{x}, \vec{y}), B_2(\vec{x}, \vec{y})$ of Q. If B_1 and B_2 are input predicates, the algorithm just checks that each $h_i(C_i)$ is not satisfied in the canonical db given by $B_1(\vec{x}, \vec{y}), B_2(\vec{x}, \vec{y})$. If B_1 and B_2 are intensional, then the algorithm considers any pair h', f where h' extends h by mapping additional variables in C to elements of $\vec{x} \cup \vec{y} \cup$ (constants of Q), and f assigns each atom in C to one of $B_1(\vec{x}, \vec{y}), B_2(\vec{x}, \vec{y})$. Let D_1 be the B_1, Q' subgoal whose query consists of all atoms that f assigns to B_1 and whose mapping is h'. Let D_2 be similarly defined for B_2. Then the algorithm chooses one of the two subgoals D_1 and D_2. Let F_1 be all the subgoals that are chosen for B_1 during this process, and F_2 be defined similarly for B_2. The algorithm recurses on the two B_1, Q' subgoal family F_1 and the B_2 subgoal family F_2. The fact that the algorithm returns true iff the subgoal is omittable follows easily by induction on the rank of Q'. □

Acknowledgements. The authors wish to thank Huy Vu for invaluable help and corrections on earlier drafts of the paper. Michael Benedikt is affiliated with the Oxford University Computing Laboratory. Benedikt is supported in part by EPSRC EP/G004021/1 (the Engineering and Physical Sciences Research Council, UK). Georg Gottlob is affiliated with the Oxford University Computing Laboratory and the Oxford Man Institute of Quantitative Finance. Gottlob's work was supported by the European Research Council under the European Communitys Seventh Framework Programme (FP7/2007- 2013)/ERC grant no. 246858 DIADEM. G. Gottlob currently holds of a Royal Society Wolfson Research Merit Award.