Optimal Broadcasting Strategies for Conjunctive Queries over Distributed Data

Bas Ketsman* **and Frank Neven**

Hasselt University and transnational University of Limburg
Belgium
first.lastname@uhasselt.be

Abstract

In a distributed context where data is dispersed over many computing nodes, monotone queries can be evaluated in an eventually consistent and coordination-free manner through a simple but naive broadcasting strategy which makes all data available on every computing node. In this paper, we investigate more economical broadcasting strategies for full conjunctive queries without self-joins that only transmit a part of the local data necessary to evaluate the query at hand. We consider oblivious broadcasting strategies which determine which local facts to broadcast independent of the data at other computing nodes. We introduce the notion of broadcast dependency set (BDS) as a sound and complete formalism to represent local optimal oblivious broadcasting functions. We provide algorithms to construct a BDS for a given conjunctive query and study the complexity of various decision problems related to these algorithms.

1998 ACM Subject Classification H.2.3 Query Languages, H.2.4 Distributed databases

Keywords and phrases Coordination-free evaluation, conjunctive queries, broadcasting

Digital Object Identifier 10.4230/LIPIcs.ICDT.2015.291

1 Introduction

We assume the setting introduced in the context of declarative networking [6, 14], where queries are specified on a logical level over a global schema and are evaluated by multiple computing nodes over which the input database is distributed. These nodes can perform local computations and communicate asynchronously with each other via messages. The model then operates under the assumption that messages can never be lost but can be arbitrarily delayed. It is known that every monotone query can be evaluated in an eventually consistent and coordination-free manner through a naive broadcasting strategy that makes all data available to all nodes [14].[1] Indeed, every computing node sends all its local data to every other node and reevaluates the query every time new data arrives. This evaluation is eventually consistent as, because of monotonicity, no facts will be derived which later have to be retracted and, furthermore, when all transmitted data has arrived, the output of every node will correspond to the result of the query. In addition, the computation requires no coordination between the nodes.

Obviously, the above strategy leads to a very careless evaluation as the whole database is send to every node and every node independently computes the complete answer for the targeted query. In the present paper, we are interested in more economical broadcasting

* PhD Fellow of the Research Foundation – Flanders (FWO)

[1] Actually, this observation is the straightforward part of the CALM-conjecture [14]. It is the converse direction which is more surprising: that every query which can be evaluated in an eventually consistent and coordination-free manner has to be monotone [6].

18th International Conference on Database Theory (ICDT'15).
Editors: Marcelo Arenas and Martín Ugarte; pp. 291–307

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

strategies where only a subset of the local data is transmitted and where each computing node contributes to the answer of the query by outputting only a subset of the answer tuples. The result of the query then is the union of the tuples output by the computing nodes. In particular, we focus on full conjunctive queries without self-joins and we consider *oblivious broadcasting strategies* where every computing node determines which facts will be broadcast solely on the content of its own local database (so, oblivious of the data at other nodes). These facts are referred to as *broadcast facts*. Facts that are not initially broadcast are called *static*. We illustrate the ideas behind such strategies by means of an example.

▶ **Example 1.** Let Q_1 be the query $Q_1(x, y, z) \leftarrow A(x, y), B(y, x), C(x, z)$ and let $I = \{A(1, 2), A(2, 2), B(2, 1), B(2, 2), B(4, 4), C(1, 3)\}$ be a database instance. Consider a network of two computing nodes c and c' containing the facts $I(c) = \{A(2, 2), B(2, 1), B(2, 2)\}$ and $I(c') = \{A(1, 2), B(4, 4), C(1, 3)\}$, respectively.

Naive broadcasting strategy. The naive broadcasting algorithm outlined above sends all facts in $I(c)$ to c' and all facts in $I(c')$ to c. Eventually, both c and c' receive all data and both of them compute the result of the query, that is, $Q_1(I) = \{(1, 2, 3)\}$.

Improved oblivious broadcasting strategy. The just described strategy is clearly oblivious but also rather wasteful. Therefore consider the following strategy which broadcasts all of the C-facts but none of the A-facts. Furthermore, a B-fact $B(i, j)$ is broadcast only when $A(j, i)$ does not occur in the local database. Executing this strategy for every computing node in our example results in c broadcasting the set $\{B(2, 1)\}$ while c' broadcasts $\{B(4, 4), C(1, 3)\}$. So, eventually, $I^*(c) = \{A(2, 2), B(2, 1), B(2, 2), B(4, 4), C(1, 3)\}$ and $I^*(c') = \{A(1, 2), B(2, 1), B(4, 4), C(1, 3)\}$. Here, we denote by $I^*(d)$ the instance at node d when all transmitted messages have arrived. Therefore, $Q_1(I^*(c)) = \emptyset$ and $Q_1(I^*(c')) = \{(1, 2, 3)\}$, and $Q_1(I)$ equals $Q_1(I^*(c)) \cup Q_1(I^*(c'))$. Intuitively, this strategy is correct in general as the following invariant holds for every computing node d: when a fact $B(i, j)$ is *not* broadcast at a node d, then every satisfying valuation V for Q on I that maps (x, y) to (i, j) can be realized locally in $I^*(d)$. Notice that, a similar strategy reversing the roles of A- and B-facts would work as well.

We will formalize oblivious broadcasting functions as generic mappings. This means that decisions on whether to broadcast facts do not depend only on the name of the predicate but can also depend on the equality type of the fact under consideration. Therefore, the following strategy would be valid as well: always broadcast facts of the form $C(i, j)$ with $i \neq j$ and keep all facts of the form $C(i, i)$ static; broadcast all B-facts; broadcast a fact $A(i, j)$ only when the fact $C(i, i)$ is not present in the local database. While not immediately obvious, this strategy correctly computes Q on every distributed database.

Both strategies will be presented more formally in Section 5 in terms of *broadcast dependency sets* and are formalized further in Example 12(1) and 12(3). ◀

In this paper, we make the following contributions:

(i) We provide a semantical characterization of when an oblivious broadcasting function (OBF) correctly evaluates a given conjunctive query. While it is desirable to construct OBFs that minimize the overall amount of transmitted facts over all distributed databases, we show that there is no optimal OBF for any conjunctive query with at least two distinct atoms in its body. Therefore, we turn to a slightly weaker notion of optimality, called local optimal, which requires that an OBF is optimal w.r.t. the local instance at every computing node. Intuitively, this means that no broadcast fact can be made static without sacrificing correctness. We provide a semantical characterization for when an OBF is local optimal for a given conjunctive query.

(ii) We introduce the notion of a broadcast dependency set (BDS) as a formalism to specify OBFs. In brief, a BDS $\mathcal{S}$ is a set of pairs (τ, T) where τ is a partial atomic type and T is a set of partial atomic types. Every such pair encodes a rule that can be interpreted roughly as follows: when a fact $\mathbf{f}$ matches type τ, it will be broadcast at a computing node c when the set of facts induced by T is not present at c. We present necessary and sufficient syntactic conditions for when a BDS is correct for a given query and also for when it is local optimal w.r.t. that query. Furthermore, we study the complexity of deciding whether a BDS is correct for a query and whether it is local optimal. Finally, and most importantly, we show that the formalism of BDS is expressively complete w.r.t. local optimal OBFs by obtaining that every local optimal OBF can be represented by a BDS. In fact, every local optimal OBF can already be represented by a BDS that only uses complete types, that is, types where the equalities between all variables are fully specified.

(iii) Based on the syntactic criteria of when a BDS is correct for Q and when it is local optimal, we obtain an algorithm BDS-BUILD(Q) that computes a local optimal OBF (represented as a BDS) for a given conjunctive query Q. When restricting to open types (these are types without restrictions on the equalities between variables), BDS-BUILD(Q) computes a local optimal OBF in time polynomial in the size of Q. When considering complete types, BDS-BUILD(Q) computes a local optimal OBF in time exponential in the size of Q simply because there are exponentially many complete types.

Outline. We discuss related work in Section 2 and introduce the necessary definitions and concepts in Section 3. In Section 4, we discuss oblivious broadcasting functions and local optimality. In Section 5, we discuss broadcast dependency sets and study their properties. In Section 6, we provide an algorithm to construct a local optimal oblivious broadcasting function for a given conjunctive query. We conclude in Section 7.

2 Related Work

CALM. The approach in this paper is motivated by the work on the CALM-conjecture. Hellerstein [14] formulated the CALM-principle which suggests a link between logical monotonicity and distributed consistency without the need for coordination. The latter principle is, for instance, embedded in BLOOM, a declarative language for distributed programming, for which practical program analysis technique have been developed detecting potential consistency anomalies [3, 4, 11]. Ameloot et al. [6] formalized (and proved) the CALM-conjecture in terms of relational transducer networks. Zinn et al. [19] showed that the generalization of the conjecture to stronger variants of relational transducer networks fails. Ameloot et al. [5] then subsequently provided a more fine-grained answer to the CALM-conjecture by relating these stronger variants of relational transducer networks to weaker notions of monotonicity. All of these works considered naive evaluation strategies that broadcast *all* of the local data. In particular, none of these works considered more economic broadcasting evaluation of conjunctive queries.

Massive parallel model. The networked relational transducer model is just one paradigm for studying distributed query evaluation. In the massively parallel (MP) model, introduced by Koutris and Suciu [15], computation proceeds in a sequence of parallel steps, each followed by global synchronization of all servers. In this model, evaluation of conjunctive queries [15, 7] as well as skyline queries [2] have been considered. Recently, Beame et al. [8] proved a

matching upper and lower bound for the amount of communication needed to compute a full conjunctive query without self-joins in one communication round. The upper bound is provided by a randomized algorithm called Hypercube which dates back to Ganguli et al. [13] and was described by Afrati and Ullman [1] in the context of MapReduce algorithms. Hypercube is motivated by modern massively distributed systems like, for instance, Spark [18], where entire computations reside in main memory, replay is used to recover, and the dominant cost is that of communication. We note that one-round Hypercube is coordination-free and can be easily employed within the framework of relational transducer networks as well. A characteristic of Hypercube-style algorithms is that the space of computing nodes (over which the input data will be distributed) needs to be known in advance. The broadcasting strategies considered in this paper are motivated by a cloud computing setting where data is already initially distributed and the complete space of computing nodes is not necessarily known in advance. In this respect, Hypercube-style and broadcasting algorithms are orthogonal.

Relevance. One approach to minimize data transfer for a query Q, is to find the smallest subset J of a distributed instance I for which $Q(I) = Q(J)$ and then only broadcast the relevant subset J. Determining which part of a database is relevant for answering a query is a problem that arises in different contexts. For instance, causality in databases aims to determine which tuples in the database instance caused the output to a query [16, 17]. Then, the contingency set asks for the smallest set K such that $Q(I) \neq Q(I - K)$. So, any set $I - K$ extended with one element is relevant. Similarly, "where" and "why" provenance refer to the location(s) in the source databases from which the output was extracted or by which the output was influenced [10, 9]. Fan et al. [12] study the problem of scale independence where, through access patterns, the result of a query depends only on a bounded part of the database. It would be interesting to investigate how these different approaches translate to a distributed setting. Most surely, any lower bounds for the sequential setting imply lower bounds for the distributed setting, but upper bounds need to take into account that the initial database instance I is distributed as well.

3 Preliminaries

Instances and queries. For a finite set S, we denote by $|S|$ its cardinality and by 2^S its powerset. We denote $\{1, \ldots, n\}$ by $[n]$, for $n \in \mathbb{N}$. We assume an infinite set **dom** of data values. A *database schema* σ is a collection of relation names R where every R has arity $ar(R) > 0$. We call $R(\bar{d})$ a *fact* when R is a relation name and $\bar{d}$ is a tuple in **dom**. We say that a fact $R(d_1, \ldots, d_k)$ is *over* a database schema σ if $R \in \sigma$ and $ar(R) = k$. A *(database) instance* I over σ is simply a finite set of facts over σ. We denote by $Adom(I)$ the set of all values that occur in facts of I. When $I = \{\mathbf{f}\}$, we simply write $Adom(\mathbf{f})$ rather than $Adom(\{\mathbf{f}\})$. A *query* over a schema σ to a schema σ' is a generic mapping Q from instances over σ to instances over σ'. Genericity means that for every permutation π of **dom** and every instance I, $Q(\pi(I)) = \pi(Q(I))$. For the remainder of the paper, we assume given a database schema σ over which all queries are defined and do not refer to it anymore. A query Q is *monotone* if $Q(I) \subseteq Q(J)$ for all instances I, J with $I \subseteq J$. We only consider monotone queries in the sequel.

Conjunctive queries. Let **var** be the universe of variables, disjoint from **dom**. An *atom* A is of the form $R(u_1, \ldots, u_k)$ where R is a relation name and each $u_i \in$ **var**. We call R the *predicate* and denote it by $pred(A)$. We denote the variables occurring in A by $Vars(A) =$

$\{u_1, \ldots, u_k\}$. We say that A is an atom *over* the database schema σ if $pred(A) \in \sigma$ and $k = ar(pred(A))$. A *conjunctive query* Q (CQ) is an expression of the form $A_0 \leftarrow A_1, \ldots, A_n$, where for every $i \in [n]$, A_i is an atom over the schema and A_0 is an atom not over the schema. In particular, A_0 is the head of Q, denoted $head_Q$, and $A_1, \ldots, A_n$ is the body of Q, denoted $body_Q$. By $Vars(Q)$ we denote all the variables occurring in Q. A *valuation* for Q on an instance I is a function $V : Vars(Q) \to Adom(I)$. The *application* of V to an atom $A = R(u_1, \ldots, u_k)$, denoted $V(A)$, results in the fact $R(a_1, \ldots, a_k)$ where $a_i = V(u_i)$ for each $i \in [k]$. The valuation V is said to be *satisfying* for Q if $V(A) \in I$ for all atoms A in the body of Q. In that case, V derives the fact $V(A_0)$. The result of Q on I, denoted $Q(I)$ is defined as the set of facts that can be derived by satisfying valuations.

In what follows, we assume that every CQ is full and does not contain self-joins. Formally, we require that $pred(A_i) \neq pred(A_j)$ for $i \neq j$ and $Vars(A_0) = \bigcup_{i \in [n]} Vars(A_i)$. That is, every atom has a unique relation symbol and all variables occurring in the body occur in the head as well. For instance, $Q_1(x, y, z) \leftarrow A(x, y), B(x, z), C(y, y)$ is full and does not contain self-joins, while $Q_2(x, y) \leftarrow A(x, y), B(x, z), C(y, y)$ is not full and $Q_3(x, y, z) \leftarrow A(x, y), A(x, z), C(y, y)$ contains a self-join.

Distributed database. A *network* $\mathcal{N}$ is a nonempty finite set of values from **dom**, which we call *nodes*. A *distribution* of an instance I over $\mathcal{N}$ is a function H that maps each $c \in \mathcal{N}$ to an instance such that $I = \bigcup_{c \in \mathcal{N}} H(c)$. Notice that facts can be replicated. We also refer to each of the $H(c)$ as the *local instances*. We consider a model where nodes have unlimited computational power and can send messages to all other nodes. These messages can never be lost but can be arbitrarily delayed.

4 Oblivious broadcasting

We refrain from introducing the formalism of relational transducer networks from [6], but present a simpler setting more suitable for our needs. In particular, the relational transducer networks needed in this paper only perform two actions: decide which facts to broadcast (and transmit those) and evaluate the query under consideration whenever new data arrives. The only parameter is the used broadcasting strategy and, therefore, forms the focus of our formalization. In brief, we consider broadcasting strategies where computing nodes partition their local database into *static* and *broadcast* facts. Static facts are kept local while broadcast facts, as the name already indicates, are sent to all other nodes in the network. As we only consider conjunctive queries which are monotone, the target query can be recomputed whenever new data arrives.

4.1 Oblivious broadcasting functions

We now formally define oblivious broadcasting function.

▶ **Definition 2.** An *oblivious broadcasting function (OBF)* f is a generic mapping that maps instances to instances such that $f(J) \subseteq J$ for all instances J.

An OBF specifies which local facts are broadcast. Specifically, $f(J)$ are the broadcast facts while $J \setminus f(J)$ are the static facts. We use the term oblivious as broadcast facts only depend on the local database instance and their choice is independent of the facts at other computing nodes. An OBF f is *naive* when there are no static facts, that is, $f(J) = J$ for all instances J.

Given a CQ Q, an instance I, a distribution H of I, and a network $\mathcal{N}$, an OBF f implies a broadcasting algorithm in the following way. Let $B(f, H) = \bigcup_{c \in \mathcal{N}} f(H(c))$ be the set of

broadcast facts. Then, define $\mathit{eval}(Q, f, H) = \bigcup_{c \in \mathcal{N}} Q(H(c) \cup B(f, H)))$ as the union of the query result at every computing node over the local instance extended with all broadcast facts.[2]

▶ Remark. We note that the function $\mathit{eval}(Q, f, H)$ implies an evaluation that can be executed by a transducer program $\pi_{f,Q}$ at every node c as follows: (1) $R = \emptyset$, output $Q(H(c))$, broadcast $f(H(c))$; (2) whenever a fact $\mathbf{f}$ arrives, $R = R \cup \{\mathbf{f}\}$, output $Q(H(c) \cup R)$. Correctness then follows from the genericity and monotonicity of f. We refer to the execution strategy induced by $\mathrm{eval}(Q, f, H)$ as a *broadcasting algorithm*. Coordination-freeness intuitively follows as $\pi_{f,Q}$ never waits. Formally, a transducer is coordination-free [6] if there is a so-called *ideal* distribution, on which the query is already computed by a prefix of a run that does not process any of the incoming facts. For $\pi_{f,Q}$ this is the distribution that puts the complete instance at every node. We refer to [6] for a more formal treatment of coordination-freeness.

▶ **Definition 3.** An OBF f is *correct* for a CQ Q when $Q(I) = \mathit{eval}(Q, f, H)$ for all instances I and all distributions H of I.

When f is correct for Q, we also say that f is an OBF *for* Q. The following lemma characterizes correctness in that two compatible facts residing at different computing nodes can never be both static. Indeed, if they are, then the valuation witnessing compatibility is never realized at any computing node and consequently f can not be correct for Q.

We say that two distinct facts $\mathbf{f}$ and $\mathbf{g}$ are *compatible w.r.t* Q, denoted $\mathbf{f} \sim_Q \mathbf{g}$, when they are assigned to two atoms from the body of Q under one valuation, i.e., there is a valuation V for Q and atoms $A, B \in \mathit{body}_Q$, such that $V(A) = \mathbf{f}$ and $V(B) = \mathbf{g}$.

▶ **Lemma 4.** *Let Q be a CQ and f be an OBF. Then, the following are equivalent:*

1. *f is correct for Q; and*
2. *there are no instances I, J, and facts $\boldsymbol{f}, \boldsymbol{g}$, with $\boldsymbol{f} \sim_Q \boldsymbol{g}$, $\boldsymbol{g} \notin I, \boldsymbol{f} \notin J$ such that $\boldsymbol{f} \notin f(I \cup \{\boldsymbol{f}\})$ and $\boldsymbol{g} \notin f(J \cup \{\boldsymbol{g}\})$.*

Proof. (1)⇒(2) We start by showing that every OBF for Q satisfies the above condition. The proof is by contraposition, so we assume that there are instances I and J and compatible facts $\mathbf{f}$ and $\mathbf{g}$ w.r.t. Q, where $\mathbf{g} \notin I$ and $\mathbf{f} \notin J$, but $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$ and $\mathbf{g} \notin f(J \cup \{\mathbf{g}\})$. Let K be an instance and let V be a satisfying valuation for Q on K witnessing compatibility of $\mathbf{f}$ and $\mathbf{g}$. Then consider a network $\mathcal{N} = \{1, 2, 3\}$ and an instance $L = I \cup J \cup V(\mathit{body}_Q)$ with the following distribution H: $H(1) = I \cup \{\mathbf{f}\}$, $H(2) = J \cup \{\mathbf{g}\}$, and $H(3) = V(\mathit{body}_Q) \setminus \{\mathbf{f}, \mathbf{g}\}$. Clearly, $V(\mathit{head}_Q) \in Q(L)$. As Q is full, $V(\mathit{head}_Q) \notin \bigcup_{c \in \mathcal{N}} Q(H(c) \cup B(f, H))$ because none of the computing nodes contain both $\mathbf{f}$ and $\mathbf{g}$, and $\mathbf{f}$ and $\mathbf{g}$ are not broadcast. Thus, $Q(I) \neq \bigcup_{x \in \mathcal{N}} Q(H(x) \cup B(f, H)) = \mathit{eval}(Q, f, H)$ and f is not an OBF for Q.

(2)⇒(1) It remains to show that if the above condition is satisfied, then f is an OBF for Q. For this, let I be an instance, $\mathcal{N}$ a network, and H a distribution of I over $\mathcal{N}$. We prove that $Q(I) = \mathit{eval}(Q, f, H) = \bigcup_{c \in \mathcal{N}} Q(H(c) \cup B(f, H))$. As Q is monotone, $Q(H(c) \cup B(f, H)) \subseteq Q(I)$ for every $c \in \mathcal{N}$. Hence, it suffices to show that $Q(I) \subseteq \bigcup_{c \in \mathcal{N}} Q(H(c) \cup B(f, H))$. Thereto, let $\mathbf{f} \in Q(I)$, let V be a satisfying valuation for Q over I for which $V(\mathit{head}_Q) = \mathbf{f}$. Let $J = V(\mathit{body}_Q) \setminus B(f, H)$, and c a node for which $|H(c) \cap J|$ is maximal. We claim that $J \subseteq H(c)$, obviously implying that $\mathbf{f}$ will be derived at node c. Towards a contradiction, assume there is an $\mathbf{f}_i \in J \setminus H(c)$. As $\mathbf{f}_i \in I$ there is a $d \in \mathcal{N}$, $c \neq d$, such that $\mathbf{f}_i \in H(d)$. Moreover, by choice of c, $|H(d) \cap J| \leq |H(c) \cap J|$ and thus there must be a fact $\mathbf{f}_j \in H(c)$

[2] To simplify notation, in the definition of B and *eval*, we do not mention I and $\mathcal{N}$ as they are implied by H.

that is not in $H(d)$. However, as $\mathbf{f}_i \sim_Q \mathbf{f}_j$, $\mathbf{f}_i \notin H(c)$, and $\mathbf{f}_j \notin H(d)$, instances $H(d)$, $H(c)$, and facts $\mathbf{f}_i, \mathbf{f}_j$ contradict condition (2). ◀

4.2 Local optimality

We are interested in OBFs that transmit as little data as possible. Thereto, we investigate sensible notions of optimality. We fix a query Q, an instance I, a distribution H of I, and a network $\mathcal{N}$. The total number of transmitted facts equals $||B(f,H)|| = \sum_{c\in\mathcal{N}} |f(H(c))|$. Of course, $||B(f,H)|| \geq |B(f,H)|$.

▶ **Definition 5.** An OBF f for a CQ Q is *optimal* iff $||B(f,H)|| \leq ||B(g,H)||$ for every other OBF g for Q and for every instance I and distribution H.

Intuitively, an OBF is optimal when it transmits the least amount of data over all instances and all distributions. The next result, however, shows that this notion of optimality, although desirable, is unattainable.

▶ **Lemma 6.** *There is no optimal OBF for any conjunctive query with at least two distinct atoms in its body.*

Proof. Let Q be the conjunctive query $A_0 \leftarrow A_1, \ldots, A_n$ with $n \geq 2$. Towards a contradiction assume there is an optimal OBF f for Q. Let I be the canonical instance for Q where for every $i \in [n]$, the relation $pred(A_i)$ is interpreted by the fact A_i.[3] Now, consider a network $\mathcal{N} = [n]$ and a distribution H that places every fact in I on a distinct node. As all of the n facts in I need to be gathered at one node, at least $n-1$ facts must be broadcast. Let $\mathbf{g}$ be the fact in I that is not broadcast by f and assume w.l.o.g. that $pred(\mathbf{g}) = A_n$. As the OBF that broadcasts all A_i-facts for $i < n$ and keeps all A_n-facts static is correct for Q and only transmits $n-1$ facts on I, by assumption on the minimality of f, $||B(f,H)|| = n-1$. Now, consider $I' = I \setminus \{\mathbf{g}\}$. And let H' equal H restricted to only facts in I' over $\mathcal{N}$. Then, as $\mathbf{g}$ is not broadcast in H, $||B(f,H)|| = ||B(f,H')||$. However, the OBF that broadcasts all A_i-facts for $i > 1$ and keeps all A_1-facts static is correct for Q and only broadcasts $n-2$ facts on I' contradicting the optimality of f. ◀

We next turn to a different form of optimality. For two OBFs f and g, we say that f is *included* in g, denoted $f \subseteq g$, iff $f(I) \subseteq g(I)$ for every instance I.

▶ **Definition 7.** An OBF f for a CQ Q is *local optimal* iff for every other broadcasting function g for Q, $g \subseteq f$ implies $f = g$.

Intuitively, when f is local optimal there is no subdivision of f that transmits only a strict subset of the facts broadcast by f.

The next lemma gives a sufficient criteria for when an OBF can not be local optimal. Specifically, a condition is given for when a broadcast fact $\mathbf{f}$ can be kept static and a more economical OBF f' can be derived.

▶ **Lemma 8.** *Let Q be a CQ and let f be an OBF for Q. If there is an instance I and fact $\boldsymbol{f}$ for which $\boldsymbol{f} \in f(I \cup \{\boldsymbol{f}\})$, but there is no instance J and no fact $\boldsymbol{g}$ for which $\boldsymbol{f} \sim_Q \boldsymbol{g}$, $\boldsymbol{g} \notin I$, $\boldsymbol{f} \notin J$, and $\boldsymbol{g} \notin f(J \cup \{\boldsymbol{g}\})$, then there is an OBF f' for Q for which $f' \subsetneq f$.*

[3] Notice that we abuse the notation and interpret variables as values.

Proof. Assume f, I, and $\mathbf{f}$ as given by the statement of the lemma. The proof is now by construction. Let $I_{\mathbf{f},J}$ be the set of facts that (by genericity) relate the same way to J, as $\mathbf{f}$ to I. That is, $I_{\mathbf{f},J} = \{\pi(\mathbf{f}) \mid \pi \text{ a permutation s.t. } \pi(I) = J\}$. Then, define f' as the mapping where for every instance J, $f'(J) = f(J) \setminus I_{\mathbf{f},J}$. Notice that $f' \subsetneq f$ by construction of f'. Furthermore, f' is generic and is an oblivious broadcasting function. It remains to show that f' is an oblivious broadcasting function for Q. Towards a contradiction, assume that f' is not an oblivious broadcasting function for Q. Then, by Lemma 4, there are instances J_1 and J_2 and facts $\mathbf{g}_1$ and $\mathbf{g}_2$, for which $\mathbf{g}_1 \sim_Q \mathbf{g}_2, \mathbf{g}_2 \notin J_1$, $\mathbf{g}_1 \notin J_2$, and $\mathbf{g}_1 \notin f'(J_1 \cup \{\mathbf{g}_1\})$ and $\mathbf{g}_2 \notin f'(J_2 \cup \{\mathbf{g}_2\})$. As f is an oblivious broadcasting function for Q, it holds that

$$\mathbf{g}_1 \in f(J_1 \cup \{\mathbf{g}_1\}) \text{ or } \mathbf{g}_2 \in f(J_2 \cup \{\mathbf{g}_2\}).$$

Say that $\mathbf{g}_1 \in f(J_1 \cup \{\mathbf{g}_1\})$. Then, $\mathbf{g}_1 \in I_{\mathbf{f},J_1}$, implying $J_1 = \pi(I)$ and $\mathbf{g}_1 = \pi(\mathbf{f})$ for some permutation π. As Q does not contain self-joins and $\mathbf{g}_1 \sim_Q \mathbf{g}_2$, this means that $\mathbf{g}_2 \notin I_{\mathbf{f},J}$. Therefore, $\mathbf{g}_2 \notin f(J_2 \cup \{\mathbf{g}_2\})$ which contradicts the condition of the lemma (taking $\pi^{-1}(\mathbf{g}_1)$ and $\pi^{-1}(J_2)$ as $\mathbf{g}$ and J, respectively). ◀

The following lemma now characterizes when an OBF for a query is local optimal.

▶ **Lemma 9.** *Let Q be a CQ and let f be an OBF for Q. The following are equivalent:*

1. *f is local optimal; and*
2. *for every instance I and fact $\boldsymbol{f}$ for which $\boldsymbol{f} \in f(I \cup \{\boldsymbol{f}\})$, there is an instance J and a fact $\boldsymbol{g}$ such that $\boldsymbol{f} \sim_Q \boldsymbol{g}$, $\boldsymbol{g} \notin I$, $\boldsymbol{f} \notin J$, and $\boldsymbol{g} \notin f(J \cup \{\boldsymbol{g}\})$.*

Proof. We can assume that Q contains at least two atoms. Indeed, when Q contains one atom, the only local optimal OBF is the one that broadcasts no facts and the lemma trivially holds. The direction from (1) to (2) follows from Lemma 8.

(2)⇒(1) Let f be an OBF for Q. Towards a contradiction assume that f is not local optimal. That is, there exists another OBF f' for Q such that $f' \subsetneq f$. In particular, there is an instance I and a fact $\mathbf{f}$ such that $\mathbf{f} \notin f'(I \cup \{\mathbf{f}\})$, while $\mathbf{f} \in f(I \cup \{\mathbf{f}\})$. By Lemma 4, for every fact $\mathbf{g}$ with $\mathbf{f} \sim_Q \mathbf{g}$ where $\mathbf{g} \notin I$, and for every instance J, where $\mathbf{f} \notin J$, it must be that $\mathbf{g} \in f'(J \cup \{\mathbf{g}\})$. The latter then implies that for every such $\mathbf{g}$ and J, $\mathbf{g} \in f(J \cup \{\mathbf{g}\})$ which contradicts condition (2) of the present lemma. ◀

5 Broadcasting functions based on dependency sets

In this section, we introduce the notion of a broadcast dependency set (BDS) as a formalism to specify OBFs. We present necessary and sufficient conditions for when a BDS induces an OBF which is correct for a given query and also for when it is local optimal. Furthermore, we study the complexity of the corresponding decision problems. Finally, we show that every local optimal OBF can be represented by a BDS thereby obtaining that BDS is complete as a representation formalism for local optimal OBFs.

5.1 Broadcast dependency sets

Let Q be the CQ $A_0 \leftarrow A_1, \ldots, A_n$. We assume Q is full and does not contain self-joins. Therefore an atom A_i in $body_Q$ is uniquely identified by its predicate $pred(A_i)$. For a predicate R, we denote by $atom(R)$ the unique atom $A \in body_Q$ for which $pred(A) = R$.

For a finite set of variables X, a *partial (equality) type over* X is a pair of binary relations $\varphi = (E_\varphi, I_\varphi)$ representing equalities and inequalities among elements in X. Formally, we require that $E_\varphi \cup I_\varphi \subseteq X \times X$, E_φ is an equivalence relation, and I_φ is irreflexive and

symmetric. We abuse notation and also use φ to denote the formula $\bigwedge\{x = y \mid (x, y) \in E_\varphi\} \wedge \bigwedge\{x \neq y \mid (x, y) \in I_\varphi\}$. We tacitly assume that partial types are always consistent. That is, we always assume that there is a tuple $\bar{a}$ such that the formula $\varphi(\bar{a})$ evaluates to true. When for all $(x, y) \in X \times X$, either $(x, y) \in E_\varphi$ or $(x, y) \in I_\varphi$, then φ completely specifies all relations between variables in X and we call φ a *type*. For emphasis, we sometimes say *complete type* rather than just *type* even though type always means complete type.

A *partial atomic type* (over Q) is a pair $\tau = (R_\tau, \varphi_\tau)$, where R_τ is a database predicate and φ_τ is a partial type over $\mathit{Vars}(\mathit{atom}(R_\tau))$, that is, the variables occurring in the unique atom $A \in \mathit{body}_Q$ for which $\mathit{pred}(A) = R_\tau$. By $\mathit{Vars}(\tau)$ we denote the variables over which τ is defined, i.e., $\mathit{Vars}(\tau) = \mathit{Vars}(\mathit{atom}(R_\tau))$. Sometimes we write $\mathit{atom}(\tau)$ to abbreviate $\mathit{atom}(R_\tau)$. We say that τ is an *atomic type* when φ_τ is a type. To improve readability, we denote partial atomic types with τ and (complete) atomic types with ω. We denote by $\mathit{PTypes}(Q)$ and $\mathit{Types}(Q)$ the set of all partial atomic types and atomic types over Q, respectively.

A fact $\mathbf{f}$ *is of type* τ or *satisfies* τ, denoted $\mathbf{f} \models \tau$, when there is a valuation h from the variables in $\mathit{atom}(R_\tau)$ onto $\mathit{Adom}(\mathbf{f})$ such that $h(\mathit{atom}(R_\tau)) = \mathbf{f}$ and the formula φ_τ evaluates to true where each x_i is substituted by $h(x_i)$. Notice that h is unique for $\mathbf{f}$. Hereafter we will refer to h as $V_\mathbf{f}$. By $\mathit{type}(\mathbf{f})$, we denote the unique atomic type satisfied by $\mathbf{f}$ when it exists. As atomic types are defined w.r.t. Q, $\mathit{type}(\mathbf{f})$ is not always defined. Indeed, when $\mathbf{f} = R(a, b)$ (with $a \neq b$) and $\mathit{atom}(R) = R(x, x)$, then there is no τ with $\mathbf{f} \models \tau$. Two partial atomic types τ, τ' are *compatible w.r.t.* Q, denoted $\tau \sim_Q \tau'$, when there are facts $\mathbf{f}$ and $\mathbf{g}$ with $\mathbf{f} \models \tau$ and $\mathbf{g} \models \tau'$ such that $\mathbf{f} \sim_Q \mathbf{g}$. We say that τ *implies* τ', denoted $\tau \models \tau'$, if for all facts $\mathbf{f}$, $\mathbf{f} \models \tau$ implies $\mathbf{f} \models \tau'$. We can think of a partial atomic type as a disjunction of types for a shared predicate symbol. Define $\mathit{Types}(\tau) = \{\omega \in \mathit{Types}(Q) \mid \omega \models \tau\}$ as the set of all atomic types ω which imply τ. Notice that, $\omega \models \tau$ iff $\omega \in \mathit{Types}(\tau)$ for any atomic type ω. For a set of partial atomic types T, we use $\mathit{Types}(T)$ as an abbreviation for $\bigcup_{\tau \in T} \mathit{Types}(\tau)$.

For a set of variables X and Y, and a partial atomic type τ, $X \subseteq_\tau Y$ if for all $x \in X$ either $x \in Y$ or there is an $y \in Y$ such that $(x, y) \in E_{\varphi_\tau}$. That is, X is a subset of Y when taking the equalities in E_{φ_τ} into account. For instance, let τ be a type such that $(y, z) \in E_{\varphi_\tau}$, then $\{x, y, z\} \subseteq_\tau \{x, y\}$.

For a set of pairs $\mathcal{S}$, we define $\mathit{Keys}(\mathcal{S}) = \{a \mid (a, b) \in \mathcal{S}\}$ and $\mathit{Values}(\mathcal{S}) = \{b \mid (a, b) \in \mathcal{S}\}$.

▶ **Definition 10.** A *broadcast dependency set (BDS)* for a CQ Q is a set $\mathcal{S}$ of pairs (τ, T), where $\tau \in \mathit{PTypes}(Q)$ is a *key*, and $T \in 2^{\mathit{PTypes}(Q)}$ is a *dependency set*, such that the following holds:

1. $(\tau, T) \in \mathcal{S}$ and $(\tau, T') \in \mathcal{S}$ implies $T = T'$;
2. $\tau, \tau' \in \mathit{Keys}(\mathcal{S})$ implies $\mathit{Types}(\tau) \cap \mathit{Types}(\tau') = \emptyset$; and,
3. $(\tau, T) \in \mathcal{S}$ implies $\mathit{Vars}(\tau') \subseteq_{\tau'} \mathit{Vars}(\tau)$ for every $\tau' \in T$.

The above definition states that (1) every key can have at most one value in $\mathcal{S}$; (2) every complete type implies at most one partial type $\tau \in \mathit{Keys}(\mathcal{S})$; and, (3) the set of variables of $\mathit{atom}(\tau')$ is included in the set of variables of $\mathit{atom}(\tau)$ taking into account the equalities in $E_{\tau'}$. We first explain informally how a BDS represents an OBF. Let $\mathbf{f}$ be a fact in the local instance at a computing node. When $\mathit{type}(\mathbf{f})$ is undefined, then $\mathbf{f}$ is static as $\mathbf{f}$ can never participate in any satisfying valuation. For instance this happens when $\mathbf{f} = R(a, b)$ with $a \neq b$ and Q contains the atom $R(x, x)$. Every pair $(\tau, T) \in \mathcal{S}$ now specifies a condition on facts: when $\mathbf{f} \models \tau$ then $\mathbf{f}$ is broadcast only if a set of facts implied by T (to be formalized below) is not present at the local instance. Furthermore, when there is no $\tau \in \mathit{Keys}(\mathcal{S})$ for which $\mathbf{f} \models \tau$, $\mathbf{f}$ is broadcast as well. In this light, conditions (1) and (2) ensure that every

local fact $\mathbf{f}$ is matched by at most one partial type $\tau \in \mathit{Keys}(S)$; and, condition (3) ensures that when $\mathbf{f} \models \tau$ then $V_{\mathbf{f}}$ can be extended in a unique way to a valuation for every $\tau' \in T$ that is consistent with $\mathbf{f}$, that is, for which $\mathit{type}(\mathbf{f}) \sim_Q \tau'$.

Next, we formally define how every BDS $\mathcal{S}$ implies an OBF $f_{\mathcal{S}}$. Given a fact $\mathbf{f}$, if there is no $\tau \in \mathit{Keys}(\mathcal{S})$ for which $\mathbf{f} \models \tau$ then $\mathbf{f}$ is always broadcast. Otherwise, by condition (1) and (2) of Definition 10, there is exactly one $\tau \in \mathit{Keys}(\mathcal{S})$ such that $\mathbf{f} \models \tau$. Recall that $V_{\mathbf{f}}$ is the valuation (defined above) such that $V_{\mathbf{f}}(\mathit{atom}(\tau)) = \mathbf{f}$. Then, by condition (3) of Definition 10, $V_{\mathbf{f}}$ can also be interpreted as a valuation for every $\mathit{atom}(\tau')$ for every $\tau' \in T$ for which $\mathit{type}(\mathbf{f}) \sim_Q \tau'$. Indeed, for every $y \in \mathit{Vars}(\tau') \setminus \mathit{Vars}(\tau)$ there is a variable $x \in \mathit{Vars}(\tau)$ for which $(x, y) \in E_{\tau'}$. Therefore, define for every $y \in \mathit{Vars}(\tau')$,

$$V_{\mathbf{f},\tau'}(y) = \begin{cases} V_{\mathbf{f}}(y) & \text{if } y \in \mathit{Vars}(\tau); \text{ and,} \\ V_{\mathbf{f}}(x) & \text{if } y \notin \mathit{Vars}(\tau) \text{ and } (x, y) \in E_{\tau'}. \end{cases}$$

As we only consider $V_{\mathbf{f},\tau'}$ for which $\mathit{type}(\mathbf{f}) \sim_Q \tau'$, the above is well-defined.

Now, $\mathbf{f}$ is broadcast when the local instance does not contain all the facts $V_{\mathbf{f},\tau'}(\mathit{atom}(\tau'))$ for which $\tau' \in T$ and $\mathit{type}(\mathbf{f}) \sim_Q \tau'$. We refer to these facts as the *dependency fact set*. To formally define $f_{\mathcal{S}}$, we set $\mathit{Dep}(\mathbf{f}, T) = \{V_{\mathbf{f},\tau'}(\mathit{atom}(\tau')) \mid \tau' \in T \text{ and } \mathit{type}(\mathbf{f}) \sim_Q \tau'\}$. Then, define $\mathit{Dep}(\mathbf{f}, \mathcal{S})$ as $\mathit{Dep}(\mathbf{f}, T)$ when there is a $(\tau, T) \in \mathcal{S}$ for which $\mathbf{f} \models \tau$. Otherwise, $\mathit{Dep}(\mathbf{f}, \mathcal{S})$ is undefined.

▶ **Definition 11.** For a CQ Q and a BDS $\mathcal{S}$ for Q, define $f_{\mathcal{S}}$ as the function that maps every instance J to the set $f_{\mathcal{S}}(J)$ of those facts $\mathbf{f} \in J$ for which (1) $\mathit{type}(\mathbf{f}) \in \mathit{Types}(Q)$; and, (2) $\mathit{Dep}(\mathbf{f}, \mathcal{S})$ is undefined or $\mathit{Dep}(\mathbf{f}, \mathcal{S}) \not\subseteq J$.

Intuitively, $\mathbf{f}$ is static only when $\mathit{type}(\mathbf{f}) \notin \mathit{Types}(Q)$ ($\mathbf{f}$ can not participate in any satisfying valuation) or the dependency fact set $\mathit{Dep}(\mathbf{f}, \mathcal{S})$ is present at the local instance.

▶ **Example 12.** (1) For a simple example of a BDS $\mathcal{S}$ and OBF $f_{\mathcal{S}}$, recall query Q_1 from Example 1, being $Q_1(x, y, z) \leftarrow A(x, y), B(y, x), C(x, z)$. Let $\varphi = (\emptyset, \emptyset)$, that is, φ imposes no restrictions. Let $\tau_A = (A, \varphi)$ and $\tau_B = (B, \varphi)$. Then, $\mathcal{S} = \{(\tau_B, \{\tau_A\}), (\tau_A, \emptyset)\}$ is a BDS for Q_1. Indeed, every partial atomic type occurs at most once as a key. There is no (complete) atomic type that implies both τ_A and τ_B. Furthermore, the variable containment condition between τ_A and τ_B is satisfied. Notice that $f_{\mathcal{S}}$ simulates exactly the broadcast dependency function which is described in Example 1.

(2) Consider the query $Q_2(x, y, z) \leftarrow A(x, y, z), B(x, y, z), C(z, z)$. For simplicity, we define partial types through formulas. Then, define $\tau_B = (B, \text{true})$, $\tau_A^{x=y} = (A, x = y)$, $\tau_A^{y=z} = (A, y = z)$, $\tau_A^{\neq} = (A, x \neq y \wedge y \neq z)$, $\tau_B^{\neq} = (B, x \neq y \wedge y \neq z)$. Then, $\mathcal{S} = \{(\tau_B, \{\tau_A^{x=y}, \tau_A^{y=z}\}), (\tau_A^{\neq}, \{\tau_B^{\neq}\})\}$ is a BDS for Q_2. To illustrate how OBF $f_{\mathcal{S}}$ works, let $I = \{A(1,2,3), B(1,2,3), A(1,1,2), B(1,1,2), A(1,2,2), B(1,2,2), C(3,4), C(3,3)\}$ be a database instance. Then, $f_{\mathcal{S}}(I) = \{A(1,1,2), A(1,2,2), C(3,3)\}$. Indeed, the facts $A(1,1,2)$, $A(1,2,2)$, $C(3,3)$ do not match a key in $\mathcal{S}$ and their type occurs in $\mathit{Types}(Q)$. So they are broadcast. The fact $C(3,4)$ is not broadcast as its type does not occur in $\mathit{Types}(Q)$ ($C(3,4)$ does not match $C(z,z)$). The fact $\mathbf{f}_1 = B(1,1,2)$ matches τ_B and $\mathit{Dep}(\mathbf{f}_1, \{\tau_A^{x=y}, \tau_A^{y=z}\}) = \{A(1,1,2)\} \subseteq I$. Therefore, $B(1,1,2)$ is static. Similarly, the fact $\mathbf{f}_2 = B(1,2,2)$ matches τ_B and $\mathit{Dep}(f_2, \{\tau_A^{x=y}, \tau_A^{y=z}\}) = \{A(1,2,2)\} \subseteq I$. Therefore, $B(1,2,2)$ is static as well. The fact $\mathbf{f}_3 = A(1,2,3)$ is static as it matches τ_A and $\mathit{Dep}(\mathbf{f}_3, \{\tau_b^{\neq}\}) = \{B(1,2,3)\} \subseteq I$. The fact $\mathbf{f}_4 = B(1,2,3)$ is static as it matches τ_B and $\mathit{Dep}(\mathbf{f}_4, \{\tau_A^{x=y}, \tau_A^{y=z}\}) = \emptyset$.

(3) For an example where condition (3) of Definition 10 does not reduce to ordinary variable containment, consider again query Q_1 from Example 1. Let $\tau_C = (C, x = z)$, and

$\tau_A = (A, \text{true})$. Then, $\mathcal{S} = \{(\tau_A, \{\tau_C\}), (\tau_C, \emptyset)\}$ is a BDS for Q_1. Notice that condition $Vars(C) \not\subseteq Vars(A)$ but $Vars(\tau_C) \subseteq_{\tau_C} Vars(\tau_A)$.

(4) Our final example shows that dependencies can be circular. Let $Q_3(x, y, z) \leftarrow A(x, y), B(y, z), C(z, x)$. Let $\tau_A = (A, x = y)$, $\tau_B = (B, x = y)$, and $\tau_C = (C, x = y)$. Then, $\mathcal{S} = \{(\tau_A, \{\tau_B\}), (\tau_B, \{\tau_C\}), (\tau_C, \{\tau_A\})\}$ is an OBF for Q_1. Though correctness of $\mathcal{S}$ for Q follows from Lemma 13, we provide some intuition. Let $I = \{A(1,1), B(1,1), C(1,1)\}$ be a database instance. Consider a network containg the nodes c_1, c_2, and c_3. When $I(c_1) = \{A(1,1)\}$, $I(c_2) = \{B(1,1)\}$, and $I(c_3) = \{C(1,1)\}$, then all three facts will be broadcast. Now, assume one of the nodes contains two of the facts in I, w.l.o.g., say $I(c_1) = \{A(1,1), B(1,1)\}$. Then, exactly one of the facts in $I(c_1)$ is broadcast; i.e., $B(1,1)$. Now, suppose that $C(1,1)$ is mapped on some node, say c_2, but that $C(1,1)$ is not broadcast. Then it must be that $A(1,1)$ is mapped on c_2 as well. So, broadcasting $B(1,1)$ indeed suffices to guarantee correctness. ◀

Note that not every BDS for Q induces an OBF which is correct for Q. Indeed, the following lemma provides equivalent semantic and syntactic conditions for an OBF $f_{\mathcal{S}}$ to be correct for a query.

▶ **Lemma 13.** *Let Q be a CQ and let $\mathcal{S}$ be a BDS for Q. Then the following are equivalent:*

1. *$f_{\mathcal{S}}$ is an OBF for Q;*
2. *there are no instances I, J, and facts $\boldsymbol{f}, \boldsymbol{g}$, with $\boldsymbol{f} \sim_Q \boldsymbol{g}$, $\boldsymbol{g} \notin I, \boldsymbol{f} \notin J$ such that $\boldsymbol{f} \notin f_{\mathcal{S}}(I \cup \{\boldsymbol{f}\})$ and $\boldsymbol{g} \notin f_{\mathcal{S}}(J \cup \{\boldsymbol{g}\})$; and*
3. *there are no (complete) atomic types ω_1, ω_2, and pairs $(\tau_1, T_1), (\tau_2, T_2) \in \mathcal{S}$, with $\omega_1 \sim_Q \omega_2$, $\omega_1 \models \tau_1$, $\omega_2 \models \tau_2$ such that $\omega_1 \notin Types(T_2)$ and $\omega_2 \notin Types(T_1)$.*

Proof. (1)⇔(2) Because $f_{\mathcal{S}}$ is an OBF, the equivalence follows immediately from Lemma 4.

(2)⇒(3) The proof is by contraposition. So, assume that there are two (complete) atomic types ω_1, ω_2, and pairs $(\tau_1, T_1), (\tau_2, T_2) \in \mathcal{S}$, with $\omega_1 \sim_Q \omega_2$, $\omega_1 \in Types(\tau_1)$, $\omega_2 \in Types(\tau_2)$ such that $\omega_1 \notin Types(T_2)$ and $\omega_2 \notin Types(T_1)$. Now, because $\omega_1 \sim_Q \omega_2$, there are facts $\mathbf{f}$ and $\mathbf{g}$, with $\mathbf{f} \sim_Q \mathbf{g}$, $type(\mathbf{f}) = \omega_1$, and $type(\mathbf{g}) = \omega_2$. Define $I = Dep(\mathbf{f}, \mathcal{S})$ and $J = Dep(\mathbf{g}, \mathcal{S})$. Observe that by definition of Dep, $\omega_1 \notin Types(T_2)$ implies $\mathbf{f} \notin Dep(\mathbf{g}, \mathcal{S})$ and $\omega_2 \notin Types(T_1)$ implies $\mathbf{g} \notin Dep(\mathbf{f}, \mathcal{S})$. Hence, $\mathbf{f} \notin J$ and $\mathbf{g} \notin I$. Moreover, by definition of $f_{\mathcal{S}}$, it is always the case that $\mathbf{f} \notin f_{\mathcal{S}}(Dep(\mathbf{f}, \mathcal{S}) \cup \{\mathbf{f}\})$ and $\mathbf{g} \notin f_{\mathcal{S}}(Dep(\mathbf{g}, \mathcal{S}) \cup \{\mathbf{g}\})$. Therefore, $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$ and $\mathbf{g} \notin f_{\mathcal{S}}(J \cup \{\mathbf{g}\})$, which contradicts condition (2).

(3)⇒(2) Again, the proof is by contraposition. So, assume that there is an instance I and J and facts $\mathbf{f}$ and $\mathbf{g}$ where $\mathbf{f} \sim_Q \mathbf{g}$, $\mathbf{g} \notin I$ and $\mathbf{f} \notin J$, but $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$ and $\mathbf{g} \notin f_{\mathcal{S}}(J \cup \{\mathbf{g}\})$. As $\mathbf{f} \sim_Q \mathbf{g}$, we have $\omega_1 \sim_Q \omega_2$ for $\omega_1 = type(\mathbf{f})$ and $\omega_2 = type(\mathbf{g})$. Then, by construction of $f_{\mathcal{S}}$ there are $(\tau_1, T_1), (\tau_2, T_2) \in \mathcal{S}$ with $type(\mathbf{f}) \in Types(\tau_1)$ and $type(\mathbf{g}) \in Types(\tau_2)$. Now, $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$ and $\mathbf{g} \notin f_{\mathcal{S}}(J \cup \{\mathbf{g}\})$ implies $Dep(\mathbf{f}, \mathcal{S}) \subseteq I$ and $Dep(\mathbf{g}, \mathcal{S}) \subseteq J$. If we assume that $type(\mathbf{g}) \in Types(T_1)$ then $\mathbf{g} \in Dep(\mathbf{f}, \mathcal{S})$ (as $\mathbf{g} = V_{\mathbf{f}, type(\mathbf{g})}(atom(type(\mathbf{f})))$), and therefore $\mathbf{g} \in I$ which leads to a contradiction. Hence, $type(\mathbf{g}) \notin Types(T_1)$. A similar argument shows that $type(\mathbf{f}) \notin Types(T_2)$. So, we have found ω_1, ω_2, (τ_1, T_2), and (τ_2, T_2) contradicting condition (3). ◀

Notice that the OBFs of Example 12 are all correct for the given query.

Two partial atomic types τ_1, τ_2 are said to be *equal*, denoted $\tau_1 = \tau_2$, when $Types(\tau_1) = Types(\tau_2)$. We say that a BDS $\mathcal{S}$ is *harmonious* when every two partial types in $\mathcal{S}$ are either disjoint or equal. That is, when for every two partial atomic types $\tau_1, \tau_2 \in Keys(\mathcal{S}) \cup \{\tau' \in T \mid T \in Values(\mathcal{S})\}$, either $\tau_1 = \tau_2$ or $Types(\tau_1) \cap Types(\tau_2) = \emptyset$.

▶ **Theorem 14.** *Let Q be a CQ and let $\mathcal{S}$ be a BDS for Q. Deciding whether $f_{\mathcal{S}}$ is correct for Q is* CONP*-complete and in* PTIME *when $\mathcal{S}$ is harmonious.*

5.2 Local optimality

Next, we turn to local optimal OBFs. The following lemma provides equivalent semantic and syntactic conditions for an OBF to be local optimal. Regarding condition (3), the intuition is as follows. While condition (3c) is the syntactic counterpart of condition (2), conditions (3a) and (3b) specify optimality requirements which are inherent to the formalism of BDS. More specifically, condition (3a) specifies that every atomic type implying a partial type in a dependency set in $\mathcal{S}$ must also imply a key in $\mathcal{S}$. Indeed, when an atomic type does not imply a key, every local fact of this type is always broadcast and therefore present at every computing node. The atomic type can therefore be removed from every dependency set it occurs in. When Condition (3b) fails for an atomic type ω, $\mathcal{S}$ can be adapted to broadcast less while preserving correctness for Q by adding the pair $(\omega, \{\tau \mid \tau \sim_Q \omega, \tau \in \mathit{Types}(\mathit{Keys}(\mathcal{S}))\})$.

▶ **Lemma 15.** *Let Q be a CQ, $\mathcal{S}$ a BDS for Q, and $f_{\mathcal{S}}$ an OBF for Q. The following are equivalent:*

1. *$f_{\mathcal{S}}$ is local optimal;*
2. *for every instance I and fact $\boldsymbol{f}$ for which $\boldsymbol{f} \in f_{\mathcal{S}}(I \cup \{\boldsymbol{f}\})$, there is an instance J and a fact $\boldsymbol{g}$ such that $\boldsymbol{f} \sim_Q \boldsymbol{g}$, $\boldsymbol{g} \notin I$, $\boldsymbol{f} \notin J$, and $\boldsymbol{g} \notin f_{\mathcal{S}}(J \cup \{\boldsymbol{g}\})$; and,*
3. *$\mathcal{S}$ satisfies the following conditions:*
   - **(a)** *for $(\tau, T) \in \mathcal{S}$ and $\omega \in \mathit{Types}(T)$, $\omega \sim_Q \tau$ implies $\omega \models \tau'$ for some $\tau' \in \mathit{Keys}(\mathcal{S})$;*
   - **(b)** *for every $\omega \in \mathit{Types}(Q) \setminus \mathit{Types}(\mathit{Keys}(\mathcal{S}))$, there is a partial atomic type $\tau_1 \in \mathit{Keys}(\mathcal{S})$ and a $\omega_1 \in \mathit{Types}(\tau_1)$ such that $\omega \sim_Q \omega_1$ and $\mathit{Vars}(\omega_1) \not\subseteq_{\omega_1} \mathit{Vars}(\omega)$; and*
   - **(c)** *for $(\tau_1, T_1), (\tau_2, T_2) \in \mathcal{S}$, where $\omega_1 \in \mathit{Types}(\tau_1)$, $\omega_2 \in \mathit{Types}(\tau_2)$, and $\omega_1 \sim_Q \omega_2$: $\omega_1 \in \mathit{Types}(T_2)$ implies $\omega_2 \notin \mathit{Types}(T_1)$.*

Deciding whether $f_{\mathcal{S}}$ is local optimal for arbitrarily given BDS $\mathcal{S}$ turns out to be hard (c.f., Theorem 16). Therefore, we also consider the special case of open BDSs. We say that a partial type $\varphi = (E, I)$ is *open* when it enforces no restrictions. That is, when $E = I = \emptyset$. A partial atomic type (R, φ) is *open* when φ is. We say that a BDS $\mathcal{S}$ is *open* when it only contains open partial atomic types. Notice that a BDS that is open is also harmonious (but not vice versa).

Similarly to Theorem 14, we have the following decidability result for local optimal OBFs.

▶ **Theorem 16.** *Let Q be a CQ and let $\mathcal{S}$ be a BDS for Q for which $f_{\mathcal{S}}$ is correct for Q. Deciding whether $f_{\mathcal{S}}$ is local optimal is in* CONP *and in* PTIME *when $\mathcal{S}$ is open.*

It remains open though whether deciding local optimality is CONP-complete or in PTIME (even for harmonious BDS). For harmonious BDS, condition (1) and (3) of Lemma 15 are verifiable in polynomial time.

Next, we show that every local optimal OBF can be represented by a BDS thereby obtaining that BDSs (satisfying the conditions in Lemma 15) are a complete representation of local optimal OBFs. Let Q be a CQ and let f be an OBF for Q. We call a fact $\mathbf{f}$ *semi-static* for f when there is an atomic type ω and an instance I such that $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$ and $\mathit{type}(\mathbf{f}) = \omega$. That is, $\mathbf{f}$ has an atomic type and there is an instance for which $\mathbf{f}$ is not broadcast. We say that a semi-static fact $\mathbf{f}$ (for f) *depends* on a fact $\mathbf{g}$, when $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$ implies $\mathbf{g} \in I$ for every instance I. With every semi-static fact $\mathbf{f}$, we associate the set $D_{\mathbf{f}}$ containing exactly all facts on which $\mathbf{f}$ depends. Thus, $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$ implies $D_{\mathbf{f}} \subseteq I$.

We make use of the following lemma in the proof of Theorem 18.

▶ **Lemma 17.** *Let Q be a CQ, and f be a local optimal OBF for Q. Let $\boldsymbol{f}$ be semi-static for f. Then, $\boldsymbol{f} \notin f(D_{\boldsymbol{f}} \cup \{\boldsymbol{f}\})$. Furthermore, $\boldsymbol{g} \in D_{\boldsymbol{f}}$ implies*

1. *$\boldsymbol{g}$ is semi-static and $\boldsymbol{g} \sim_Q \boldsymbol{f}$;*
2. *$Adom(\boldsymbol{g}) \subseteq Adom(\boldsymbol{f})$;*
3. *$Vars(atom(\boldsymbol{g})) \subseteq_{type(\boldsymbol{g})} Vars(atom(\boldsymbol{f}))$; and*
4. *$\boldsymbol{g} = V_{\boldsymbol{f},type(\boldsymbol{g})}(atom(\boldsymbol{g}))$;*

We are now ready to prove completeness. The proof of the following theorem shows that the formalism of BDS that only uses complete atomic types can already represent every local optimal OBF.

▶ **Theorem 18** (Completeness). *Let Q be a CQ and f a local optimal OBF for Q. Then, there is a BDS $\mathcal{S}$ for Q such that $f = f_{\mathcal{S}}$.*

Proof. We start by noting that if $\mathbf{f}$ is semi-static for f, then every $\mathbf{g}$ with $type(\mathbf{f}) = type(\mathbf{g})$ is semi-static for f as well. Therefore, we say that an atomic type τ is semi-static for f when there is a semi-static fact $\mathbf{f}$ with $type(\mathbf{f}) = \tau$. The proof is by construction. Let $\mathcal{S}$ be the set of pairs (τ, D_τ) where τ is semi-static for f and $D_\tau = Types(D_{\mathbf{f}})$, where $\mathbf{f}$ is a fact with atomic type τ.

We first show that $\mathcal{S}$ is a BDS and then that $f = f_{\mathcal{S}}$. Notice that, $\mathcal{S}$ has only finitely many pairs, because there are only finitely many distinct atomic-types, and every set in $Values(\mathcal{S})$ is finite by construction. Let $(\tau, T) \in \mathcal{S}$, and $\tau' \in T$. By construction of $\mathcal{S}$, τ is a semi-static atomic type for f and for every atomic type τ there is at most one pair $(\tau, T) \in \mathcal{S}$. Furthermore, $T = D_\tau$. Let $\mathbf{f}$ be a fact of type τ. Then, $\mathbf{f}$ is a semi-static fact for f and there is a $\mathbf{g} \in D_{\mathbf{f}}$, such that $type(\mathbf{g}) = \tau'$. By Lemma 17(3), $Vars(atom(\tau')) = Vars(atom(\mathbf{g})) \subseteq_{type(\mathbf{g})} Vars(atom(\mathbf{f})) = Vars(atom(\tau))$. So, $\mathcal{S}$ is a broadcast dependency set for query Q.

Next, we show that $f = f_{\mathcal{S}}$. For this, we assume $D_{\mathbf{f}} = Dep(\mathbf{f}, D_{type(\mathbf{f})})$ (which is argued below) and show that $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$ iff $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$.

Let $\mathbf{f}$ be a fact and I an instance, such that $\mathbf{f} \notin f(I \cup \{\mathbf{f}\})$. If $\mathbf{f}$ has no atomic type, then it is never broadcast by $f_{\mathcal{S}}$. So, assume $\mathbf{f}$ has an atomic type. Then it must be that $D_{\mathbf{f}} \subseteq I$. However, because $(type(\mathbf{f}), D_{type(\mathbf{f})}) \in \mathcal{S}$ and $D_{\mathbf{f}} = Dep(\mathbf{f}, D_{type(\mathbf{f})})$, $Dep(\mathbf{f}, \mathcal{S}) \subseteq I$. Hence, by definition of $f_{\mathcal{S}}$, $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$.

For fact $\mathbf{f}$ and instance I, where $\mathbf{f} \in f(I \cup \{\mathbf{f}\})$, Lemma 9 implies that $\mathbf{f}$ has an atomic type. Either, $\mathbf{f}$ is always broadcast by f, or it is semi-static for $\mathbf{f}$. The former implies that there is no pair in $\mathcal{S}$ of the form $(type(\mathbf{f}), T)$. So, $\mathbf{f}$ is broadcast by $f_{\mathcal{S}}$ as well. The latter implies by Lemma 17 that $D_{\mathbf{f}} \not\subseteq I$ and there is a pair $(type(\mathbf{f}), D_{type(\mathbf{f})}) \in \mathcal{S}$. In particular, because $Dep(\mathbf{f}, D_{type(\mathbf{f})}) = D_{\mathbf{f}}$, $Dep(\mathbf{f}, D_{type(\mathbf{f})}) \not\subseteq I$, which implies that $\mathbf{f} \notin f_{\mathcal{S}}(I \cup \{\mathbf{f}\})$.

It remains to show that $D_{\mathbf{f}} = Dep(\mathbf{f}, D_{type(\mathbf{f})})$. Because $\mathbf{g} \in D_{\mathbf{f}}$, implying $type(\mathbf{g}) \in D_{type(\mathbf{f})}$, it follows by Lemma 17(4) that $\mathbf{g} \in Dep(\mathbf{f}, D_{type(\mathbf{f})})$. For the reverse direction, let $\mathbf{g} \in Dep(\mathbf{f}, D_{type(\mathbf{f})})$, which implies $type(\mathbf{g}) \in D_{type(\mathbf{f})}$. So, there must be some fact $\mathbf{g}'$, which is of the same type as $\mathbf{g}$, in $D_{\mathbf{f}}$. In particular, because $D_{\mathbf{f}} \subseteq Dep(\mathbf{f}, D_{type(\mathbf{f})})$, $\mathbf{g}' = V_{\mathbf{f},type(\mathbf{g}')}(atom(\mathbf{g}'))$. However, because $\mathbf{g} = V_{\mathbf{f},type(\mathbf{g})}(atom(\mathbf{g}))$, $atom(\mathbf{g}) = atom(\mathbf{g}')$, and $type(\mathbf{g}') = type(\mathbf{g})$, it must be that $\mathbf{g} = \mathbf{g}'$. So, indeed $\mathbf{g} \in D_{\mathbf{f}}$. ◀

6 Algorithms for constructing a BDS

Lemma 13 and Lemma 15 yield a natural algorithm for constructing a local optimal OBF for a given conjunctive query Q by simply starting from $\mathcal{S} = \emptyset$ and adding new pairs in a one by one fashion till no more pairs can be added. More formally, we introduce the algorithm

Input: conjunctive query Q
Param: sequence of partial types $\mathcal{R}$
$\mathcal{S} = \emptyset$;
foreach $\tau \in \mathcal{R}$ **do**
  addPair = *true*;
  Values = $\emptyset$;
  foreach $\tau' \in \mathit{Keys}(\mathcal{S})$, *where* $\tau' \sim_Q \tau$ **do**
    Values = Values $\cup\{\tau'\}$;
    if $\mathit{Vars}(\tau') \not\subseteq_{\tau'} \mathit{Vars}(\tau)$ **then**
      addPair = *false*;
    end
  end
  if addPair **then**
    $\mathcal{S} = \mathcal{S} \cup \{(\tau, \text{Values})\}$;
  end
end
return $\mathcal{S}$

Algorithm 1: Algorithm BDS-BUILD.

BDS-BUILD, given in Algorithm 1. As there are exponentially many (in the size of Q) partial atomic types, we parameterize BDS-BUILD by a sequence $\mathcal{R}$ of partial atomic types.[4] The algorithm then produces a set of pairs $(\tau, T) \in \mathit{PTypes}(Q) \times 2^{\mathit{PTypes}(Q)}$.

The following theorem obtains the correctness of BDS-BUILD. The complexity follows directly from the size of $\mathcal{R}$ which is polynomial in the size of Q for open types and exponential for complete types.

▶ **Theorem 19.** *For a conjunctive query Q and a sequence $\mathcal{R}$ consisting of exactly the complete (respectively, open) types,* BDS-BUILD(Q) *computes a BDS $\mathcal{S}$ for Q in time exponential (respectively, polynomial) in the size of Q such that $f_{\mathcal{S}}$ is correct for Q and local optimal.*

▶ **Example 20.** We illustrate BDS-BUILD by means of an example.
Consider the conjunctive query $Q(x, y, z, w) \leftarrow A(x, y, z), B(x, y, z), C(z, w)$.

1. **Open types.** Observe that query Q has three open types, being $\tau_A = (A, \mathit{true})$, $\tau_B = (B, \mathit{true})$, and $\tau_C = (C, \mathit{true})$. Let $\mathcal{R} = (\tau_A, \tau_B, \tau_C)$. Then, BDS-BUILD computes a BDS by starting from $\mathcal{S} = \emptyset$, expanding $\mathcal{S}$ to $\{(\tau_A, \emptyset)\}$ in the first iteration and to $\{(\tau_A, \emptyset), (\tau_B, \{\tau_A\})\}$ in the second iteration. During the last iteration, $\mathcal{S}$ is not changed anymore, because $\mathit{Vars}(\tau_A) \not\subseteq_{\tau_A} \mathit{Vars}(\tau_C)$.
2. **Complete types.** The (complete) atomic types for Q are

$$\tau_X^{\neq} = (X, x \neq y \wedge y \neq z \wedge x \neq z), \qquad \tau_X^{x=z} = (X, x = z \wedge z \neq y \wedge y \neq z),$$
$$\tau_X^{x=y} = (X, x = y \wedge x \neq z \wedge y \neq z), \qquad \tau_X^{y=z} = (X, x \neq y \wedge y = z \wedge z \neq x),$$
$$\tau_X^{=} = (X, x = y \wedge x = z \wedge y = z), \qquad \tau_C^{=} = (C, z = w), \text{ and } \tau_C^{\neq} = (C, z \neq w),$$

where $X \in \{A, B\}$.[5] Let $\mathcal{R} = (\tau_B^{\neq}, \tau_C^{=}, \tau_C^{\neq}, \tau_B^{x=z}, \tau_A^{x=y}, \tau_A^{\neq}, \tau_A^{x=z}, \tau_A^{=}, \tau_B^{=}, \tau_A^{y=z}, \tau_B^{x=y}, \tau_B^{y=z})$.

[4] We use a sequence rather than a set $\mathcal{R}$ to keep BDS-BUILD deterministic.
[5] For convenience we represent atomic types here by partial atomic types with sufficient (but not complete) conditions; e.g., we write $(C, x = y)$ to denote $(C, x = y \wedge y = x)$. Nevertheless, all of the listed pairs indeed correspond to a single (complete) atomic type.

Then, the output of algorithm BDS-BUILD(Q) is the BDS $\mathcal{S} = \{(\tau_B^{\neq}, \emptyset), (\tau_B^{x=z}, \emptyset), (\tau_A^{x=y}, \emptyset), (\tau_A^{\neq}, \{\tau_B^{\neq}\}), (\tau_A^{x=z}, \{\tau_B^{x=z}\}), (\tau_A^{=}, \emptyset), (\tau_B^{=}, \{\tau_A^{=}\}), (\tau_A^{y=z}, \emptyset), (\tau_B^{x=y}, \{\tau_A^{x=y}\}), (\tau_B^{y=z}, \{\tau_A^{y=z}\})\}$. Observe that the atomic types $\tau_C^{=}$ and $\tau_C^{\neq}$ are not part of $\mathcal{S}$ because the variable containment condition is not satisfied by the earlier included atomic type $\tau_B^{\neq}$.
Observe that the constructed BDS $\mathcal{S}$ can be simplified by merging multiple atomic types into partial atomic types; e.g., for $\mathcal{S}' = \{(\tau_A, \{\tau_B^{\neq}, \tau_B^{x=z}\}), (\tau_B, \{\tau_A^{x=y}, \tau_A^{=}, \tau_A^{y=z}\})\}$, we have $f_{\mathcal{S}} = f_{\mathcal{S}'}$. ◀

Notice that when $\mathcal{R}$ consists of the complete or open atomic types, adding pairs to a given BDS $\mathcal{S}$ as is done by BDS-BUILD(Q) results in a BDS $\mathcal{S}'$ that describes an OBF that broadcasts strictly less facts, i.e., $f_{\mathcal{S}'} \subsetneq f_{\mathcal{S}}$. That is, adding pairs optimizes the OBF.

▶ Remark. By construction, BDS-BUILD(Q) prevents any circular dependencies by stratifying the construction of $\mathcal{S}$ so that partial atomic types can only depend on partial atomic types that where added before. As illustrated in Example 12(4), dependencies in a BDS can also be circular. To allow for these BDS-BUILD can be modified as follows: as an alternative for adding pairs (τ, T) where every existing key that is compatible with τ is included in T, we can allow adding pairs where some keys that are compatible with τ are in T, and for every other compatible key, their respective value set is expandend to contain τ; i.e., allowing pairs of the form (τ, D), where D is a subset of $C = \{\omega' \in \mathit{Keys}(\mathcal{S}) \mid \omega' \sim_Q \omega\}$ satisfying $\mathit{Vars}(\omega') \subseteq_{\omega'} \mathit{Vars}(\omega)$ for every $\omega' \in D$, and where every existing pair (ω', T), where $\omega' \in C \setminus D$, is expanded to $(\omega', T \cup \{\omega\})$. Particularly notice that when a given BDS $\mathcal{S}$ is changed to $\mathcal{S}'$ by adding a pair and expanding at least one of the existing pairs as described above, the inherent nature of the described OBF changes, so that not necessarily $f_{\mathcal{S}'} \subsetneq f_{\mathcal{S}}$.

▶ Remark. Although the machinery developed throughout this paper is motivated by gaining a better understanding of the spectrum of local optimal OBFs, the reader may notice that when no (statistical) information on the actual distribution of the data is available, there is no basis to favor one local optimal OBF over another.

In fact, there is already a very simple algorithm to find an arbitrary local optimal OBF for given CQ Q which is as good as any local optimal one (when no additional information on the distribution of the data is available). Indeed, consider an arbitrary order on the predicates of Q:

> For every local fact $\mathbf{f}$, with predicate R, if there is an earlier predicate S such that some variable in $\mathit{Vars}(S)$ is not in $\mathit{Vars}(R)$, $\mathbf{f}$ is broadcast; otherwise, $\mathbf{f}$ is broadcast only if all the facts induced by $V_{\mathbf{f}}$ on query Q are in the local instance.

Of course, not every local optimal OBF can take this form.

7 Discussion

We investigated local optimal oblivious broadcasting functions represented by the formalism of broadcast dependency sets. We obtained semantical and syntactical characterizations, showed completeness of BDSs for representing local optimal OBFs, and gave an algorithm for constructing local optimal OBFs for a given conjunctive query. We present several directions for future work: more expressive query languages, incorporating background knowledge, and non-oblivious broadcast functions.

An obvious question is how to generalize our results to the class of all conjunctive queries (possibly extended with negation) or even to (subsets of) Datalog. Of course, to evaluate non-

monotonic queries in a coordination-free manner, computing nodes need more information on how data is distributed (c.f., [6]).

We only discussed how to build a BDS when no information about the way data is distributed is available. Indeed, the best one can do is to let a BDS cover as much types as possible, but at the same time introduce as little dependencies as possible, as these are likely to fail when data is arbitrarily distributed. It would be interesting to devise optimal broadcasting algorithms taking more background knowledge into account like information about clustering of attributes, foreign keys, or cardinality of relations.

Another interesting direction for future work is to investigate non-oblivious broadcasting functions where over time, when new messages arrive, static facts can become broadcast facts (but not vice versa). Such functions are initially more conservative keeping more facts static and only broadcast facts when there is some evidence that they can be used at another computing node. For instance, consider the setting of Example 1. Rather than immediately sending $B(i, j)$ whenever $A(j, i)$ is locally absent, broadcasting is suspended until a C-fact of the form $C(j, k)$ is received. The rationale is that a B-fact that can not contribute to a locally satisfying valuation, should only be broadcast when some evidence is received that it could potentially contribute to a satisfying valuation on a remote node. For our example this means that c waits to send $B(2, 1)$ until $C(1, 3)$ arrives. Moreover, $B(4, 4)$ is never sent. While non-oblivious strategies might seem more attractive as they transmit fewer tuples, such strategies, while remaining coordination-free, can increase the overall evaluation time.

Acknowledgment. We thank Phokion Kolaitis for raising the question whether it is always necessary to broadcast all the data in the context of the work in [5]. We thank the reviewers for their in-depth comments and numerous suggestions for improving the presentation of the results.

References

1 F. N. Afrati and J. D. Ullman. Optimizing joins in a map-reduce environment. In *International Conference on Extending Database Technology (EDBT)*, pages 99–110, 2010.

2 Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. Parallel skyline queries. In *International Conference on Database Theory (ICDT)*, pages 274–284, 2012.

3 Peter Alvaro, Neil Conway, Joe Hellerstein, and William R. Marczak. Consistency analysis in bloom: a CALM and collected approach. In *Conference on Innovative Data Systems Research (CIDR)*, pages 249–260, 2011.

4 Peter Alvaro, Neil Conway, Joseph M. Hellerstein, and David Maier. Blazes: Coordination analysis for distributed programs. In *International Conference on Data Engineering (ICDE)*, pages 52–63. IEEE, 2014.

5 Tom J. Ameloot, Bas Ketsman, Frank Neven, and Daniel Zinn. Weaker forms of monotonicity for declarative networking: a more fine-grained answer to the CALM-conjecture. In *Symposium on Principles of Database Systems (PODS)*, pages 64–75. ACM, 2014.

6 Tom J. Ameloot, Frank Neven, and Jan Van den Bussche. Relational transducers for declarative networking. *Journal of the ACM*, 60(2):15, 2013.

7 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Symposium on Principles of Database Systems (PODS)*, pages 273–284, 2013.

8 Paul Beame, Paraschos Koutris, and Dan Suciu. Skew in parallel query processing. In *Symposium on Principles of Database Systems (PODS)*, pages 212–223, 2014.

9 Peter Buneman, James Cheney, Wang Chiew Tan, and Stijn Vansummeren. Curated databases. In *Symposium on Principles of Database Systems (PODS)*, pages 1–12. ACM, 2008.

10 Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *International Conference on Database Theory (ICDT)*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2001.

11 Neil Conway, William R. Marczak, Peter Alvaro, Joseph M. Hellerstein, and David Maier. Logic and lattices for distributed programming. In *Symposium on Cloud Computing (SoCC)*, page 1. ACM, 2012.

12 Wenfei Fan, Floris Geerts, and Leonid Libkin. On scale independence for querying big data. In *Symposium on Principles of Database Systems (PODS)*, pages 51–62. ACM, 2014.

13 Sumit Ganguly, Abraham Silberschatz, and Shalom Tsur. Parallel bottom-up processing of datalog queries. *Journal of Logic Programming*, 14(1&2):101–126, 1992.

14 Joseph M. Hellerstein. The declarative imperative: experiences and conjectures in distributed logic. *SIGMOD Record*, 39(1):5–19, 2010.

15 Paraschos Koutris and Dan Suciu. Parallel evaluation of conjunctive queries. In *Symposium on Principles of Database Systems (PODS)*, pages 223–234, 2011.

16 Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, and Dan Suciu. Causality in databases. *IEEE Data Engineering Bulletin*, 33(3):59–67, 2010.

17 Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proceedings of the VLDB Endowmen (PVLDB)*, 4(1):34–45, 2010.

18 Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 15–28. USENIX Association, 2012.

19 Daniel Zinn, Todd J. Green, and Bertram Ludäscher. Win-move is coordination-free (sometimes). In *International Conference on Database Theory (ICDT)*, pages 99–113, 2012.