LOCAL RELATIONAL MODEL: A LOGICAL FORMALIZATION OF DATABASE COORDINATION

Serafini L., Giunchiglia F.,
Mylopoulos J., Bernstein P.

January 2003

Technical Report # 0301−08

Local Relational Model: a logical formalization of database coordination

L. Serafini[1], F. Giunchiglia[12], J. Mylopoulos[23], and P. Bernstein[4]

[1] ITC-IRST, 38050 Povo, Trento, Italy
[2] University of Trento, 38050 Povo, Trento, Italy
[3] University of Toronto M5S 3H5, Toronto, Ontario, Canada
[4] Microsoft Corporation, One Microsoft Way, Redmond, WA

Abstract. We propose a new data model intended for peer-to-peer (P2P) databases. The model assumes that each peer has a (relational) database and exchanges data with other peers (its *acquaintances*). In this context, one needs a data model that views the space of available data within the P2P network as an open collection of possibly overlapping and inconsistent databases. Accordingly, the paper proposes the Local Relational Model, develops a semantics for coordination formulas. The main result of the paper generalizes Reiter's characterization of a relational database in terms of a first order theory [1], by providing a syntactic characterization of a relational space in terms of a multi-context system. This work extends earlier work by Giunchiglia and Ghidini on Local Model Semantics [2].

1 Introduction

Peer-to-peer (hereafter P2P) computing consists of an open-ended network of distributed computational *peers*, where each peer can exchange data and services with a set of other peers, called *acquaintances*. Peers are fully autonomous in choosing their acquaintances. Moreover, we assume that there is no global control in the form of a global registry, global services, or global resource management, nor a global schema or data repository of the data contained in the network. Systems such as Napster and Gnutella popularized the P2P paradigm as a version of distributed computing lying between traditional distributed systems and the web. The former is rich in services but requires considerable overhead to launch and has a relatively static, controlled architecture. The latter is a dynamic, anyone-to-anyone architecture with little startup costs but limited services. By contrast, P2P offers an evolving architecture where peers come and go, choose whom they deal with, and enjoy some traditional distributed services with less startup cost.

We are interested in data management issues raised by this paradigm. In particular, we assume that each peer has data to share with other nodes. To keep things simple, we further assume that these data are stored in a local relational database for each peer. Since the data residing in different databases may have semantic inter-dependencies, we require that peers can specify coordination rules which ensure that the contents of their respective databases remain "coordinated" as the databases evolve. For example, the patient database of a family doctor and that of a pharmacist may want to coordinate their information about a particular patient, the prescription she has been administered,

the dates when these prescriptions were fulfilled and the like. Coordination may mean something as simple as propagating all updates to the PRESCRIPTION and MEDICATION relations, assumed to exist in both databases. In addition, we'd like to support query processing so that a query expressed with respect to one database fetches information from other relevant databases as well. To accomplish this, we expect the P2P data management system to use coordination rules as a basis for recursively decomposing the query into sub-queries which are translated and evaluated with respect to the databases of acquaintances.

Consider the patient databases example, again. There are several databases that store information about a particular patient (family doctor, pharmacist, hospitals, specialists.) These databases need to remain acquainted and coordinate their contents for every shared patient. Since patients come and go, coordination rules need to be dynamic and are introduced by mutual consent of the peers involved. Acquaintances are dynamic too. If a patient suffers an accident during a trip, new acquaintances will have to be introduced and will remain valid until the patient's emergency treatment is over.

In such a setting, we cannot assume the existence of a global schema for all the databases in a P2P network, or just those of acquainted databases. Firstly, it is not clear what a global schema means for the whole network, given that the network is open-ended and continuously evolves. Secondly, even if the scope of a global schema made sense, it would not be practical to build one (just think of the effort and time required.) Finally, building a global schema for every peer and her acquaintances isn't practical either, as acquaintances keep changing. This means that current approaches to information integration [3, 4], are not applicable because they assume a global schema (and a global semantics) for the total data space represented by the set of peer databases.

Instead, the *Local Relational Model* (hereafter LRM) proposed here only assumes the existence of pairwise-defined *domain relations*, which relate synonymous data items, as well as *coordination formulas*, which define semantic dependencies among acquainted databases. Local relational model is an evolution of a first attempt in this direction presented in [5] which had the main limitation in the languages adopted to express peer's coordination. Among other things, LRM allows for inconsistent databases and supports semantic interoperability in a manner to be spelled out precisely herein. Further motivation for the proposed data model can be found in [6]. The main objective of this paper is to introduce the LRM, focusing on its formal semantics.

The LMS semantics presented in this paper are an extension of the *Local Model Semantics*, a new semantics motivated by the problem of formalizing contextual reasoning in AI [7], which was first introduced in [2].

2   A motivating scenario

Consider, again, the example of patient databases. Suppose that the Toronto General Hospital owns the Tgh database with schema:

```
Patient(TGH#,OHIP#,Name,Sex,Age,FamilyDr,PatRecord)
PatientInfo(OHIP#,Record)
Admission(AdmID,OHIP#,AdmDate,ProblemDesc,PhysID,DisDate)
Treatment(TreatID,TGH#,Date,TreatDesc,PhysID)
Medication(TGH#,Drug#,Dose,StartD,EndD)
```

The database identifies patients by their hospital ID and keeps track of admissions, patient information obtained from external sources, and all treatments and medications administered by the hospital staff. When a new patient is admitted, the hospital may want to establish immediately an acquaintance with her family doctor. Suppose the view exported by the family doctor DB (say, Davis) has schema:

```
Patient(OHIP#,FName,LName,Phone#,Sex,PatRecord)
Visit(OHIP#,Date,Purpose,Outcome)
Prescription(OHIP#,Med#,Dose,Quantity,Date)
Event(OHIP#,Date,Description)
```

Figuring out patient record correspondences (i.e., doing object identification) is achieved by using the patient's Ontario Health Insurance # (e.g., OHIP# = 1234). Initially, this acquaintance has exactly one coordination formula which states that if there is no patient record at the hospital for this patient, then the patient's record from Davis is added to Tgh in the `PatientInfo` relation, which can be expressed as:

$$\forall(\mathsf{Davis}: fn, ln, pn, sex, pr).(\mathsf{Davis}: \mathtt{Patient}(1234, fn, ln, pn, sex, pr) \rightarrow$$
$$\mathsf{Tgh}: \exists(tghid, n, a).(\mathtt{Patient}(tghid, 1234, n, sex, a, \mathsf{Davis}, pr) \wedge \qquad (1)$$
$$n = concat(fn, ln)))$$

In the above formula the syntax "$\forall(\mathsf{Davis}: fn, ln, pn, sex, pr)\ldots$" is a quantification of the variables fn, ln, pn, sex, pr in the domain of Davis; analogously the syntax $\mathsf{Davis}: \mathtt{Patient}(1234, fn, ln, pn, sex, pr)$ states the fact that the tuple $\langle 1234, fn, ln, pn, sex, pr \rangle$ belongs to the relation `Patient` of the database Davis.

When Tgh imports data from Davis, the existentially quantified variables $tghid$, n and a must be instantiated with some concrete elements of the domain of Tgh database, by generating a new TGH# for $tghid$, by inserting the Skolem constant `<undef-age>` for a and by instantiated n with the concatenation of fn (first name) and ln (last name) contained in Davis. Later, if patient 1234 is treated at the hospital for some time, another coordination formula might be set up that updates the `Event` relation for every treatment or medication she receives:

$$\forall(\mathsf{Tgh}: d, desc).((\mathsf{Tgh}: \exists(tid.tghid.pid.n.sex.a.pr).$$
$$(\mathtt{Treatment}(tid, tghid, d, desc, pid) \wedge$$
$$\mathtt{Patient}(tghid, 1234, n, sex, a, \mathsf{Davis}, pr)) \rightarrow \qquad (2)$$
$$\mathsf{Davis}: \mathtt{Event}(1234, d, desc)))$$

$$\forall(\mathsf{Tgh}: tghid, drug, dose, sd, ed).($$
$$\mathsf{Tgh}: \mathtt{Medication}(tghid, drug, dose, sd, ed) \wedge$$
$$\exists n, sex, a, p.\mathtt{Patient}(tghid, 1234, n, sex, a, \mathsf{Davis}, pr) \rightarrow$$
$$\mathsf{Davis}: \forall d.(sd \leq d \leq ed \rightarrow \exists desc.(\mathtt{Event}(1234, d, desc) \wedge \qquad (3)$$
$$desc = concat(drug, dose, \text{"atTGHDB"}))))$$

This acquaintance is dropped once the patient's hospital treatment is over. Along similar lines, the patient's pharmacy may want to coordinate with Davis. This acquaintance is initiated by Davis when the patient tells Dr. Davis which pharmacy she uses. Once established, the patient's name and phone are used for identification. The pharmacy database (say, Allen) has the schema:

```
Prescription(Prescr#,CustName,CustPhone#,DrugID,Dose,Repeats)
Sales(CustName,CustPhone#,DrugID,Dose,Date,Amount)
```

Here, we want Allen to remain updated with respect to prescriptions in Davis:

$$
\begin{aligned}
\forall(\mathsf{Davis}\!:\!&fn,ln,pn,med,dose,qt)(\\
&\mathsf{Davis}\!:\!\exists ohip,date,sex,pr.(\mathtt{Prescription}(ohip,med,dose,qt,date)\wedge\\
&\qquad\qquad\qquad Patient(ohip,fn,ln,pn,sex,pr))\ \rightarrow\\
&\mathsf{Allen}\!:\!\exists cn,amount.(\mathtt{Prescription}(cn,pn,med,qt,dose,amount)\wedge\\
&\qquad\qquad\qquad cn=concat(fn,ln)))
\end{aligned}
\tag{4}
$$

Of course, this acquaintance is dropped when the patient tells her doctor that she changed pharmacy. Suppose the hospital has no information on its new patient with OHIP# 1234 and needs to find out if she is receiving any medication. Here, the hospital uses its acquaintance with Toronto pharmacies association, say TPhLtd. TPhLtd, is a peer that has acquaintances with most Toronto pharmacists and has a coordination formula that allows it to access prescription information in those pharmacists' databases. For example, if we assume that Tphh consists of a single relation

$$\mathtt{Prescription}(\mathtt{Name},\mathtt{Phone\#},\mathtt{DrugID},\mathtt{Dose},\mathtt{Repeats})$$

then the coordination formula between the two databases might be:

$$
\begin{aligned}
\forall(\mathsf{Davis}\!:\!&fn,ln,pn,med,dose).(\\
&\mathsf{Davis}\!:\!\exists ohip,qt,date,sex,pr.(\ \mathtt{Prescription}(ohip,med,dose,qt,date)\wedge\\
&\qquad\qquad\qquad \mathtt{Patient}(ohip,fn,ln,pn,sex,pr))\ \rightarrow\\
&\mathsf{Tphh}\!:\!\exists name,rep.(\mathtt{Prescription}(name,pn,med,dose,rep)\wedge\\
&\qquad\qquad\qquad name=concat(fn,ln)))
\end{aligned}
\tag{5}
$$

Analogous formulas exist for every other pharmacy acquaintance of TPhLtd. Apart from serving as information brokers, interest groups also support mechanisms for generating coordination formulas from parameterized ones, given exported schema information for each pharmacy database. On the basis of this formula, a query such as "All prescriptions for patient with name N and phone# P" evaluated with respect to Tphh, will be translated into queries that are evaluated with respect to databases such as Allen. The acquaintance between the hospital and TPhLtd is more persistent than those mentioned earlier. However, this one too may evolve over time, depending on what pharmacy information becomes available to TPhLtd. Finally, suppose the patient in question takes a trip to Trento and suffers a skiing accident. Now the Trento Hospital database (TNgh) needs information about the patient from DavisDB. This is a transient acquaintance that only involves making the patient's record available to TNgh, and updating the Event relation in Davis.

3   Relational spaces

Traditionally, federated and multi-database systems have been treated as extensions of conventional databases. Unfortunately, formalizations of the relational model (such as [1]) hardly apply to these extensions where there are multiple overlapping and heterogeneous databases, which may be inconsistent and may use different vocabularies and different domains. We launch the search for implementation solutions that address the scenario described in the previous section with a formalization of LRM.

The model-theoretic semantics for LRM is defined in terms of relational spaces each of which models the state of the databases in a P2P system. These are mathematical structures generalizing the model-theoretic semantics for the Relational Model, as defined by Reiter in [1]. Coordination between databases in a relational space is expressed in terms of coordination formulas that describe dependencies between a set of databases. Let us start by recalling Reiter's key concepts.

Definition 1 (Relational Language). *A* relational language *is first order language L with equality, a finite set of constants, denoted dom, no function symbols and finite set* **R** *of predicate symbols.*

The set *dom* of constants is called the domain and represents the total set of data contained in a database, while the predicates in **R** represent its relations. For instance, the language of Davis contains the constant symbol `1234`, the relational symbols such as `Patient`, the unary predicates `OHIP#`, `FName`, `LName`, `Phone#`, `Sex`, and `PatRecord`; $\alpha(\texttt{Patient}) = \langle \texttt{OHIP\#}, \texttt{FName}, \texttt{LName}, \texttt{Phone\#}, \texttt{Sex}, and \texttt{PatRecord} \rangle$.

We use the notation **x** for a sequence of variables $\langle x_1, \ldots, x_n \rangle$ and **d** for a sequence of elements $\langle d_1, \ldots, d_n \rangle$, each of which belongs to the domain *dom*; $\phi(x)$ is a formula with the free variable x, and $\phi(\mathbf{x})$ is a formula with free variables in **x**.

Definition 2 (Relational Database). *A* relational database *is a first order interpretation m of a relational language L on the set of constants dom, such that $m(d) = d$, for all constant d of L.*

Definition 2 does not properly represent partial databases, i.e., database that contain null values or partial tuples. Indeed, if m is a relational database, $m \models \phi$ or $m \models \neg\phi$ (where "$\models$" stands for "first order satisfiability"). In an incomplete database we would like to have for instance that neither ϕ not $\neg\phi$ are trie. A common approach is to model incomplete databases as a set of first order structures, also called a state of information. We follow this approach, and formalize an incomplete database on a relational language L as a set of relational databases on L. Notice that the set of relational databases corresponding to an incomplete database all share the same domain, consisting of the set of constants contained in the database. The partiality, therefore, concerns only the interpretation of the relational symbols. With this generalization we can capture inconsistent, complete, and incomplete databases. For instance, if db_a, db_b and db_c are three (partial) relational databases defined as

$$db_a = \{m_1\}, db_b = \{m_2, m_3\}, db_c = \emptyset$$

where m_1, m_2, and m_3 are relational databases, we have that they are respectively, complete, incomplete, and inconsistent. Generally, db_i is complete if $|db_i| = 1$, incomplete if $|db_i| > 1$ and inconsistent if $db_i = \emptyset$.

Since we are interested in modelling P2P applications, we take a further step and consider, rather than a single database, a family (indexed with a set of peers I) of database. We call such of these databases a *local database* when we want to stress that it is a member of a set of (coordinated) databases.

When we consider a set of databases the same information could be represented twice in two databases. In this case we say that they *overlap*. Overlapping databases

have nothing to do with the fact that the same symbols appear in both databases—the same constant can have completely different meanings in two databases—overlap occurs when the real world entities denoted by a symbol in different databases are somehow related. To represent the overlap of two local databases, one may use a global schema, with suitable mappings to/from each local database schema. As argued earlier, this is not feasible in a P2P setting. Instead, we adopt a localized solution to the overlap problem, defined in terms of pair-wise mappings from the elements of the domain of database i to elements of the domain of database j.

Definition 3 (Domain relation). *Let L_i and L_j be two relational languages, with domains dom_i and dom_j respectively; a* domain relation r_{ij} *from i to j is any subset of $dom_i \times dom_j$.*

The domain relation r_{ij} represents the ability of database j to import (and represent in its domain) the elements of the domain of database i. In symbols, $r_{ij}(d_i) = \{d_j | \langle d_i, d_j \rangle \in r_{ij}\}$ represents the set of elements in which j translates the constant d of i's domain. In many cases, domain relations are not, one to one, for instance if two databases represent a domain at a different level of details. Domain relation are not symmetric, for instance when r_{ij} represents a currency exchange, a rounding function, or a sampling function. In a P2P setting, domain relations need only be defined for acquainted pairs of peers. Domain relations between databases are conceptually analogous to *conversion functions* between semantic objects, as defined in [8]. The domain relation defined above formalizes the case where a single attribute of one database is mapped into single attribute of another database. It is often the case, however, that two (or more) attributes of a database correspond to a single attribute in another one. An obvious is when the attributes `first-name` and `last-name` in a database i are merged in the unique attribute `name` of a database j. Domain relation can be generalized to deal with these cases by allowing, for instance, a domain relation $r_{i:(\texttt{first-name},\texttt{last-name}),j:\texttt{name}}$ to be a subset of $dom_i^2 \times dom_j$.

Example 1. Let us consider how domain relations can represent different data integration scenaria. The situation where two databases have *different but equivalent representations of the same domain* can be represented by taking r_{ij} and r_{ji} as the translation function from dom_i to dom_j and vice-versa, namely $r_{ij} = r_{ji}^{-1}$. Likewise, disjoint domains can be represented by having $r_{ij} = r_{ji} = \emptyset$. Transitive mappings between the domains of three databases are represented by imposing $r_{13} = r_{12} \circ r_{23}$. Suppose instead that dom_i and dom_j are ordered according to two orders $<_i$ and $<_j$. A relation that satisfies the property: $\forall d_1, d_2 \in dom_i, d_1 <_i d_2 \Rightarrow \forall d_1' \in r_{ij}(d_1), \forall d_2' \in r_{ij}(d_2). \ d_1' <_j d_2'$ formalizes a mapping which preserves the orders, such as currency exchange. Finally, suppose that a peer with database i doesn't want to export any information about a certain object d_s in its database. To accomplish this, it is sufficient to ensure that the domain relations from i to any other database j, do not associate any element to d_s, namely $r_{ij}(d_s) = \emptyset$.

Definition 4 (Relational space). *A relational space is a pair $\langle db, r \rangle$, where db is a set of local relational databases on I and r is a function that associates to each $i, j \in I$, a domain relation r_{ij}.*

Example 2. A relational space modeling the states of the database described in Section 2, is a pair

$$\left\langle \begin{array}{l} db = \langle db_{\mathsf{Tgh}}, db_{\mathsf{Davis}}, db_{\mathsf{Allen}}, db_{\mathsf{Tphh}}, db_{\mathsf{TNgh}} \rangle \\ r = \langle r_{\mathsf{DavisTgh}}, \ r_{\mathsf{TghDavis}}, \ r_{\mathsf{DavisAllen}}, \ r_{\mathsf{DavisTphh}} \rangle \end{array} \right\rangle$$

where the first component, the local databases, contains five sets of interpretations of the relational languages associated to Tgh, Davis, Allen and Tphh and TNgh, respectively; and the second component, the domain relation, contains four domain relations between those databases which have to coordinate according to constraints (1–5).

The fact that $t = \langle 1234, \mathtt{Pippo}, \mathtt{Inzaghi}, 444, \mathtt{M}, \mathtt{Rec\_23} \rangle$ is a tuple of the relation $\mathtt{PatRecord}$ of the Davis database, if formalized by requiring $t \in m(\mathtt{PatRecord})$ for each interpretation $m \in db_{\mathsf{Davis}}$.

The fact that $t = \langle \mathtt{TG64}, 1234, ``\mathtt{PippoInzaghi}", \mathtt{M}, \mathtt{<undef\text{-}age>}, \mathtt{Davis}, \mathtt{Rec\_23} \rangle$ is a tuple of the relation $\mathtt{Patient}$ of Tgh database, is represented by requiring that, for each natural number n, with $0 \le n \le \mathtt{MaxAge}$, db_{Tgh} contains a model a model m, with $t[\mathtt{<undef\text{-}age>}/n] \in m(\mathtt{Patient})$ ($t[\mathtt{<undef\text{-}age>}/n]$ is the result of substituting n for $\mathtt{<undef\text{-}age>}$ in t).

The fact that the $\mathtt{TGH\#}$ 1234 uniquely identifies a patient in both Tgh and Davis, is represented by requiring $r_{\mathsf{DavisTgh}}(1234) = r_{\mathsf{TghDavis}}(1234) = \{1234\}$.

4 Coordination in relational spaces

Two (or more) peers who want to coordinates each other, need a language in which they can express the inter-dependencies between the information stored in their database. To this purpose, we define a declarative language by which it is possible to express semantic relations between local databases. The formulas of this language, called *coordination formulas* can be used to describe cross-database views and cross-databases constraints.

Definition 5 (Coordination formula). *The set of* coordination formulas *CF on the family of relational languages* $\{L_i\}_{i \in I}$ *is defined as follows for each* $i \in I$ *and each formula* ϕ *of* L_i[5].

$$CF ::= i : \phi \mid CF \to CF \mid CF \wedge CF \mid CF \vee CF \mid \exists i : x.CF \mid \forall i : x.CF$$

We use Greek letters ϕ, ψ, to denote formulas of any languages L_i $i \in I$, and Latin capital letters A, B, and C to denote coordination formulas. The basic building blocks of coordination formulas are expressions of the form $i : \phi$ and are called *atomic coordination formulas*. An occurrence of a variable x in a coordination formula is a *free occurrence*, if it is not in the scope of a quantifier. Examples of coordination formulas are shown in Section 2.

To give an interpretation of coordination formulas in relational spaces, let us start by considering Definition 5 in detail. Item 1 states that coordination formulas are defined on the basis of atomic formulas of the form $i : \phi$, where ϕ is any formula of L_i. $i : \phi$

---

[5] The following precedence rules apply: $i : \dots$ has the highest precedence, followed by quantifiers, then $\wedge$, then $\vee$, and finally $\to$. For instance, $\forall i : x.i : \phi \wedge j : \psi \to k : \theta \vee h : \eta$, stands for: $((\forall(i : x).(i : \phi)) \wedge (j : \psi)) \to ((k : \theta) \vee (h : \eta))$.

intuitively means "ϕ is true in database i" and its interpretation follows the standard rules of first order logic. Thus, in particular, if ϕ is of the form $\forall x.\psi(x)$ or of the form $\exists x.\psi(x)$ then its interpretation is given in terms of the possible assignments of x to elements of dom_i.

The crucial observation for the evaluation of quantified formulas is that a free occurrence of a variable can be quantified in four different ways: by $\forall x$, $\exists x$ within an atomic coordination formula (as from Item 1), and by $\forall i : x$ or $\exists i : x$, within a coordination formula. In the two latter cases the index i tells us the domain where we interpret x. Thus, the formula $\forall i : x.A(x)$ (where $A(x)$ is a coordination formula and not a formula!) must be read as "for all elements d of the domain dom_i, A is true for d". Likewise, $\exists i : x.A(x)$, must be read as "there is an element in the domain dom_i such that A is true". The trick is that A, being a coordination formula, may contain atomic coordination formulas of the form $j : \phi(x)$, with $j \neq i$. For instance in the coordination formula (5) the variables fn and ln occur free a coordination formula with index Tgh (the consequence of the implication), while they are bound by the quantifiers $\forall(\mathsf{Davis} : fn, ln, \ldots)$.

The intuition underlying the interpretation of quantified indexed variables is that, if x is a variable being quantified with index i and occurring free in a coordination formula with index j, then we must find a way to relate the interpretation of x in dom_i to the interpretation of x in dom_j using the mapping defined by r_{ij}. More precisely, the coordination formula $\forall i : x.j : P(x)$, means, "for each object of dom_i, the *corresponding object* w.r.t. the domain relation r_{ij} in dom_j has the property P". Thus, for instance, in order to check whether the coordination formula

$$\forall i : x.(i : P(x) \rightarrow j : Q(x) \wedge k : R(x)) \tag{6}$$

is true in a relational space, one has to consider all the assignments that associate to the occurrence of x in $i : P(x)$ any element of $d \in dom_i$, and to the occurrences of x in $j : Q(x)$ and $k : R(x)$ any element of $r_{ij}(d)$ and $r_{ik}(d)$, respectively. Dually, the coordination formula $\exists i : x.j : P(x)$, means "there is an element in dom_j that corresponds w.r.t. the domain relation r_{ji} to an element of dom_i with property P". Thus, for instance, in order to check whether the coordination formula

$$\exists i : x.(i : P(x) \wedge j : Q(x) \wedge k : R(x)) \tag{7}$$

is true in a relational space, one has to find an assignment that associates to the occurrence of x in $i : P(x)$ an element d of dom_i, and to the occurrences of x in $j : Q(x)$ and $k : R(x)$ two elements $d' \in dom_j$ and $d'' \in dom_k$, respectively, such that $d \in r_{ji}(d')$ and $d \in r_{ki}(d'')$.

Notice that in our explanation of the universal quantification we used r_{ij}, while for existential quantification we used r_{ji}. This asymmetry is necessary to maintain the dual intuitive readings of existential and universal quantifiers. Indeed, the intuitive meaning of the formula $\forall i : x.j : P(x)$ is "*for all $d \in dom_i$, if $d' \in r_{ij}(d)$ then d' is in P*", which can be rephrased in its dual existential statement "*there does not exist any element $d' \in r_{ij}(d)$, which is not in P*". Notice that in this last sentence, the quantification is on the elements of dom_j, namely *on the elements in the codomain* of the domain relation r_{ij}, just like in the explanation of Equation (7) above.

To formalize the intuitions given above concerning the interpretation of coordination formulas, we need two notions. The first is *coordination space of a variable x in*

a coordination formula. Intuitively this is the set of indexes of the atomic coordination formulas that contain a free occurrence of x. The coordination space is the set of domains where x must be interpreted. Thus, for instance, the coordination space of x in the $i : P(x) \land j : Q(x) \land k : R(x)$ is $\{i, j, k\}$.

Definition 6 (Coordination space). *The* coordination space *of a variable x in a coordination formula A is a set of indexes $J \subseteq I$, defined as follows:*

1. *the coordination space of x in $i : \phi$ is $\{i\}$, if x occurs free in ϕ according to the usual definition of free occurrence in a first order formula, and the empty set, otherwise;*
2. *the coordination space of x in $A \circ B$ (for any connective $\circ$) is the union of the coordination spaces of x in A and B;*
3. *the coordination space of x in $Qi : y.A$ (for any quantifier Q) is the empty set, if x is equal to y, and the coordination space of x in A, otherwise.*

The second notion is that of *assignment* for a free occurrence of a variable in a coordination formula. To evaluate a formula A quantified over x with index i, an assignment must consider dom_i but also all the domains in the coordination space. To understand how assignments work, look at Equations (6), (7). In Equation (6) we proceed "forward" from dom_i to reach dom_j and dom_k, by applying r_{ij} and r_{ik}. In this case we say that we have an i-to-$\{j, k\}$-assignment. Instead, in Equation (7), we proceed "backward" from dom_j and dom_k to reach dom_i by applying r_{ji} and r_{ki}. In this case we say that we have an i-from-$\{j, k\}$-assignment. If J is a coordination space, i-to-J-assignments take care of the assignments due to universal quantification, while i-from-J-assignments take care of those due to existential quantification.

Definition 7 (Assignment, x-variation i-to-J-assignment, i-from-J-assignment). *An assignment $a = \{a_i\}_{i \in J}$ is a family of functions a_i, where a_i assigns to any variable x an element of dom_i. An assignment a' is an x-variation of an assignment a, if a and a' differ only on the assignments to the variable x. Given a set $J \subseteq I$ and an index $i \in I$, an assignment a is an i-to-J-assignment of x if, for all $j \in J$ distinct from i, $\langle a_i(x), a_j(x) \rangle \in r_{ij}$. An assignment a is an i-from-J-assignment of x if, for all $j \in J$ distinct from i, $\langle a_j(x), a_i(x) \rangle \in r_{ji}$.*

Definition 8 (Satisfiability of coordination formulas). *The relational space $\langle db, r \rangle$ satisfies a coordination formula A under the assignment $a = \{a_i\}_{i \in J}$, in symbols $\langle db, r \rangle \models A[a]$, according to the following rules:*

1. $\langle db, r \rangle \models i : \phi[a]$, *if for each $m \in db_i$, $m \models \phi[a_i]$;*
2. $\langle db, r \rangle \models A \rightarrow B[a]$, *if $\langle db, r \rangle \models A[a]$ implies that $\langle db, r \rangle \models B[a]$;*
3. $\langle db, r \rangle \models A \land B[a]$, *if $\langle db, r \rangle \models A[a]$ and $\langle db, r \rangle \models B[a]$;*
4. $\langle db, r \rangle \models A \lor B[a]$, *if $\langle db, r \rangle \models A[a]$ or $\langle db, r \rangle \models B[a]$;*
5. $\langle db, r \rangle \models \forall i : x.A[a]$, *if $\langle db, r \rangle \models A[a']$ for all assignments a' that are x-variations of a and that are i-to-J-assignments on x, where J is the coordination space of x in A.*
6. $\langle db, r \rangle \models \exists i : x.A[a]$, *if $\langle db, r \rangle \models A[a']$ for some assignment a' that is an x-variation of a and that is an i-from-J-assignment on x, where J is the coordination space of x in A.*

A coordination formula A is valid *if it is true in all the relational spaces. A coordination formula A is a* logical consequence *of a set of coordination formulas* Γ *if, for any relational space* $\langle db, r \rangle$ *and for any assignment a, if* $\langle db, r \rangle \models \Gamma[a]$ *then* $\langle db, r \rangle \models A[a]$.

Item 1 states that an atomic coordination formula is satisfied (under the assignment a) if all the relational databases $m \in db_i$ satisfy it. Items 2–4 enforce the standard interpretation of the boolean connectives. Item 5 states that a universally quantified coordination formula is satisfied if all its instances, obtained by substituting the free occurrence of x in the atomic coordination formulas with index i with all the elements of dom_i, and the free occurrences of x in the atomic coordination formulas with index j different from i, with all the elements of dom_j, obtained by applying r_{ij} to the elements of dom_i, are satisfied. Item 6 has the dual interpretation.

Finally, notice that the language of coordination formulas does not include negation. The addition of negation with the canonical interpretation "$\neg A$ is true iff A is *not true*", implies the possibility to define the notion of "Global inconsistency", i.e., there are sets of inconsistent coordination formulas (e.g., $\{i : \phi, \neg i : \phi\}$). These sets are not satisfiable by any relational space. On the other hand, we have that the relational space composed of all inconsistent databases, is the "most inconsistent object that we can have (not allowing global inconsistency), we therefore should allow that this vacuous distributed interpretation satisfies any setxte of coordination formulas. Indeed we have that, in absence of negation, if $db_i^0 = \emptyset$ and $r_{ij}^0 = \emptyset$, $\langle db^0, r^0 \rangle \models A$ for any coordination formula A.

Coordination formulas can be used in two different ways. First, they can be used to define constraints that must be satisfied by a relational space. For instance, the formula $\forall 1 : x. (1 : p(x) \vee 2 : q(x))$ states that any object in database 1 either is in table p or its corresponding object in database 2 is in table q. This is a useful constraint when we want to declare that certain data are available in a set of databases, without declaring exactly where. As far as we know, other proposals in the literature for expressing inter-database constraints can be uniformly represented in terms of coordination formulas.

Coordination formulas can also be used to express queries. In this case, a coordination formula is interpreted as a deductive rule that derives new information based on information already present in other databases. For instance, a coordination formula $\forall i : x. (1 : \exists y. p(x, y) \rightarrow 2 : q(x))$ allows us to derive $q(b)$ in database 2, if $p(a, c)$ holds in database 1 for some c, and $b \in r_{12}(a)$.

Definition 9 (*i*-query). *An i-query on a family of relational languages* $\{L_i\}_{i \in I}$, *is a coordination formula of the form* $A(\mathbf{x}) \rightarrow i : q(\mathbf{x})$, *where* $A(\mathbf{x})$ *is a coordination formula, and q is a new n-ary predicate symbol of* L_i *and* $\mathbf{x}$ *contains n variables.*

Definition 10 (**Global answer to an** *i*-**query**). *Let* $\langle db, r \rangle$ *be a relational space on* $\{L_i\}_{i \in I}$. *The* global answer *of an i-query of the form* $A(\mathbf{x}) \rightarrow i : q(\mathbf{x})$ *in* $\langle db, r \rangle$ *is the set:*

$$\{\mathbf{d} \in dom_i^n \,|\, \langle db, r \rangle \models \exists i : \mathbf{x}. (A(\mathbf{x}) \wedge i : \mathbf{x} = \mathbf{d})\}$$

Notationally $\mathbf{x} = \mathbf{d}$ stands for $x_1 = d_1 \wedge \ldots \wedge x_n = d_n$, and $\exists i : \mathbf{x}$ stands for $\exists i : x_1 \ldots \exists i : x_n$. Intuitively, the global answer to an *i*-query is computed by locally evaluating in db_j all the atomic coordination formulas with index j contained in A, and by recursively composing and mapping (via the domain relations) these results according to the connectives and quantifiers that compose the coordination formula A. For instance to evaluate

the query

$$(i : P(x) \lor j : Q(x)) \land k : R(x,y) \to h : q(x,y)$$

we separately evaluate $P(x)$, $Q(x)$ and $R(x,y)$ in i, j and k respectively, we map these results via r_{ih}, r_{jh}, and r_{kh} respectively obtaining three sets $s_i \subseteq dom_h$ $s_j \subseteq dom_h$ and $s_k \subseteq dom_h^2$. We then compose s_i, s_j and s_k following the connectives obtaining $(s_i \times s_j) \cap s_k$, which is the global answer of q.

Notice that the same query q has different answers depending on the database it is asked to (because of the quantification over $i : \mathbf{x}$). Notice also that Definition 10 reduces to the usual notion of answer to a query when A is an atomic coordination formula $i : \phi$ (case of a single database i). Finally, but most importantly, queries can be recursively composed. Indeed, a *recursive query* can be defined as a set of queries $\{q_h := A_h(\mathbf{x}_h) \to i_h : q_h(\mathbf{x}_h)\}_{1 \leq h \leq n}$ such that $A_h(\mathbf{x}_h)$ can contain of an atomic coordination formula $i_k : q_k(\mathbf{x}_k)$ for some $1 \leq k \leq n$. The evaluation of a query q_h in the i_h-th database is done by evaluating its body, i.e., the coordination formula A_h, which contains the query q_k. This forces the evaluation of the query q_k in the i_k-th database, and so in P2P network. We can prove the following theorem

Theorem 1. *Let $\langle db, r \rangle$ be a relational space and $rq = \{q_h := A_h(\mathbf{x}_h) \to i_h : q_h(\mathbf{x}_h)\}_{1 \leq h \leq n}$ be a recursive query. If $A(\mathbf{x})$ does not contain any $\to$ symbol, then there are n minimal sets $ans_1, \dots, ans_n$, such that each ans_h is the global answer of the query q_h, in the relational space $\langle db', r \rangle$, where db' is obtained by extending every relational database $m \in db_{i_k}$ with $m(q_k) = ans_k$, for each $k \neq h$.*

5 Representation theorems

In this section we generalize Reiter's semantic characterization of relational databases to relational spaces. We start by recalling Reiter's result (in a slightly different, but equivalent, formulation).

Definition 11 (Generalized relational theory). *A theory T on the relational language L is a* generalized relational theory *if the following conditions hold.*

- *if $dom = \{d_1, \dots d_n\}$, $\forall x (x = d_1 \lor \dots \lor x = d_n) \in T$;*
- *for any $d, d' \in dom$, $d \neq d' \in T$;*
- *for any relational symbol $R \in \mathbf{R}$, there is a finite number of finite sets of tuples $E_R^1, \dots, E_R^n$ (the possible extensions of R) such that T contains the axiom:*

$$\bigvee_{1 \leq k \leq n} \left(\forall \mathbf{x} \left(R(\mathbf{x}) \leftrightarrow \bigvee_{\mathbf{d} \in E_R^k} \mathbf{x} = \mathbf{d} \right) \right)$$

Reiter proves that any partial relational database can be uniquely represented by a generalized relational theory. The generalization to the case of multiple partial databases models each of them as a generalized relational theory, and "coordinates" them using an appropriate coordination formula which axiomatizes the domain relation.

Definition 12 (Domain relation extension). *Let r_{ij} be a domain relation. The* set of coordination formulas for the extension of r_{ij} *is the set R_{ij} that contains the coordination formula $\exists j : x. (i : x = d \land j : x = d')$ if $d' \in r_{ij}(d)$, and the coordination formula $\forall i : x. (i : x = d \to j : x \neq d')$ if $d' \notin r_{ij}(d)$.*

Theorem 2 (Characterization of domain relations). *Let R_{ij} be the set of coordination formulas for the extension of r_{ij}. For any relational space $\langle db, r' \rangle$ with db_i and db_j different from the empty set, $\langle db, r' \rangle \models R_{ij}$ if and only if $r_{ij} = r'_{ij}$.*

Theorem 2 states that, when db_i and db_j are consistent databases, the only domain relation from i to j that satisfies the coordination formulas for the extension of r_{ij} (i.e., R_{ij}) is r_{ij} itself. This means that R_{ij} uniquely characterizes r_{ij}. The characterization of a relational space (Theorem 3) is obtained by composing the characterization of local databases (Reiter's result) and the characterization of the domain relation (Theorem 2). A corollary of the relational space's characterization (Corollary 1) provides a characterization in terms of logical consequence of a global answer to a i-query.

Definition 13 (Relational multi-context system). *A relational multi-context system for a family of relational languages $\{L_i\}$ is a pair $\langle T, R \rangle$, where T is a function that associates to each i, a generalized relational theory T_i on the language L_i, and R is a set that contains all the coordination formulas for the extension of a domain relation from i to j for any $i, j \in I$.*

Theorem 3 (Representation of relational spaces). *For any relational multi-context system $\langle T, R \rangle$ there is a unique (up to isomorphism) relational space $\langle db, r \rangle$, with the following properties:*

1. *$\langle db, r \rangle \models i : T_i$ and $\langle db, r \rangle \models R$.*
2. *For each $i \in I$, db_i is different from the empty set.*
3. *$\langle db, r \rangle$ is maximal, i.e., for any other relational space $\langle db', r' \rangle$, satisfying condition 1 and 2, $db'_i \subseteq db_i$, and $r_{ij} = r'_{ij}$ for all $i, j \in I$.*

Vice-versa, for any relational space $\langle db, r \rangle$, there is a relational multi-context system $\langle T, R \rangle$ such that the maximal model of $\langle T, R \rangle$ is $\langle db, r \rangle$. We say that $\langle T, R \rangle$ is the multi-context system that represents *$\langle db, r \rangle$.*

Corollary 1 (Semantic characterization of queries). *Let $\langle T, R \rangle$ be the relational multi-context system that represents the relational space $\langle db, r \rangle$. for any i-query $q := A(\mathbf{x}) \rightarrow i : q(\mathbf{x})$, the n-tuple $\mathbf{d}$ belongs to the global answer of q, if and only if*

$$\{i : T_i\}_{i \in I}, R \models \exists i : \mathbf{x}(A(\mathbf{x}) \wedge i : \mathbf{x} = \mathbf{d})$$

Corollary 1 provides us with the basis for a correct and complete implementation of a query answering mechanism in a P2P environment.

6 Related work

The formalism presented in this paper is an extension of the Distributed First Order Logics formalism proposed in [5]. The main improvements concern the language of the coordination formulas, their semantics and the calculus. In [5] indeed, relation between databases were expressed via *domain constraints* and *interpretation constraints*. These latter correspond to particular coordination formulas: namely domain constraints from i to j corresponds to the coordination formulas $\forall i : x \exists j : yi : x = y$ and $\forall j : x \exists i : yi : x = y$,

while interpretation constraints can be translated in the coordination formulas $\forall i : \mathbf{x}. (i : \phi(\mathbf{x}) \rightarrow j : \psi(\mathbf{x}))$. This limitation on the expressive power, does not allow to express in DFOL the fact that a table, say p, of a database i is the union of two tables, say p_1 and p_2 of two different databases j and k. This constraint can be easily expressed by the following coordination formula:

$$\forall i : x. (p(x) \leftrightarrow j : p_1(x) \vee k : p_2(x))$$

As far as the query language is concerned, our approach is similar in some ways to view-based data integration techniques, in the following sense. The process of translating a query against a local database into queries against an acquaintance would be driven by the coordination formulas that relate those two databases. If one thinks of our coordination formulas as view definitions, then the translation process is comparable to ones used for rewriting queries based view definitions in the local-as-view (LAV) and global-as-view (GAV) approaches ([9, 10]. Although standard approaches cannot be applied directly to LRM, due to our use of domain relations and context-dependent coordination formulas, we expect it is possible to modify LAV/GAV query processing strategies for LRM. For example, one could define a sublanguage of LRM whose power is comparable to a tractable view definition language used for LAV/GAV query processing. One could then apply a modified LAV/GAV algorithm to that language. Or perhaps one could translate formulas and queries from the LRM sublanguage into the standard (non-LRM) language and apply a conventional LAV/GAV query processing algorithm.

Finally our approach provide a general theoretical reference framework where many forms of inter-schema constraints defined in the literature, such as [11–13, 3, 14, 15]. For lack of space we briefly show only one case. Consider for instance directional existence dependences defined in [12]. Let $T_1[X_1, Y_1]$ and $T_2[X_2, Y_2]$ be two tables of a source database (let's say 1), and that $T[C_1, C_2, C_3]$ is a table of the target database (let say 2). An example of directional existence dependence is:

$$T.(C_1, C_2) \Leftarrow \textbf{select } X_1, X_2 \textbf{ from } T_1, T_2 \textbf{ where } T_1.X_1 \leq T_2.X_2 \qquad (8)$$

The informal semantics of (**??**) is that for each tuple of value $\langle V_1, V_2 \rangle$ produced by the RHS select statement, there is a tuple t in table T such that t projected on columns C_1, C_2 has the value $\langle V_1, V_2 \rangle$. The existence dependence (8), can be rewritten in terms of coordination formulas as

$$\forall 1 : x_1 x_2 (1 : \exists y_1 y_2 (T_1(x_1, y_1) \wedge T_2(x_2, y_2) \wedge x_1 \leq x_2) \rightarrow$$
$$\exists 2 : c_1 c_2 (1 : x_1 = c_1 \wedge x_2 = c_2 \wedge 2 : \exists c_3. T(c_1, c_2, c_3))) \qquad (9)$$

When the domain relation are identity functions, (9) capture the intuitive reading of (8).

7 Conclusion

We have argued that emerging computing paradigms, such as P2P computing, call for new data management mechanisms which do away with the global schema assumption inherent in current data models. Moreover, in a P2P setting the emphasis is on *coordinating* databases, rather than *integrating* them. This coordination is defined by an evolving set of coordination formulas which are used both for constraint enforcement

and query processing. To meet these challenges, the paper proposes, the paper proposes the local relational model, LRM, where the data to be managed constitute a relational space, conceived as a collection of local databases inter-related through coordination formulas and domain relations. The main result of the paper is to define a model theory for the LRM. We use this semantics to generalize an earlier result due to Reiter which characterizes a relational space as a multi-context system. The results of this paper offer a sound springboard in launching a study of implementation techniques for the LRM, its query processing and constraint enforcement.

References

1. Reiter, R.: Towards a Logical Reconstruction of Relational Database Theory. In Brodie, M., Mylopoulos, J., Schmidt, J., eds.: On Conceptual Modelling. Springer-Verlag (1984) 191–233
2. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence **127** (2001) 221–259
3. Ullman, J.D.: Information Integration Using Logical Views. In: Proc. of the 6th International Conference on Database Theory (ICDT'97). (1997)
4. Florescu, D., Levy, A., Mendelzon, A.: Database techniques for the World-Wide Web: A survey. SIGMOD Record (ACM Special Interest Group on Management of Data) **27** (1998) 59–74
5. Ghidini, C., Serafini, L.: Distributed First Order Logics. In Gabbay, D., de Rijke, M., eds.: Frontiers Of Combining Systems 2. Studies in Logic and Computation. Research Studies Press (1998) 121–140
6. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data management for peer-to-peer computing: A vision. In: WebDB 2002: Fifth International Workshop on the Web and Databases. (2002)
7. Giunchiglia, F.: Contextual reasoning. Epistemologia, special issue on I Linguaggi e le Macchine **XVI** (1993) 345–364 Short version in Proceedings IJCAI'93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.
8. Sciore, E., Siegel, M., Rosenthal, A.: Using semantic values to facilitate interoperability among heterogeneous information systems. ACM Transactions on Database Systems **19** (1994) 254–290
9. Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal **10** (2001)
10. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS. (2002) 233–246
11. Carey, M., Haas, L., Schwarz, P., Arya, M., Cody, W., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., II, J.T., Williams, J., Wimmers, E.: Towards heterogeneous multimedia information systems: The garlic approach. RIDE-DOM (1995) 124–131
12. Ceri, S., Widom, J.: Managing semantic heterogeneity with production rules and persistent queues. In Agrawal, R., Baker, S., Bell, D., eds.: 19th International Conference on Very Large Data Bases, August 24–27, 1993, Dublin, Ireland, Proceedings, Morgan Kaufmann (1993) 108–119
13. Levy, A., Rajaraman, A., Ordille, J.: Querying Heterogeneous Information Sources Using Source Descriptions. In: Proceedings of the 22nd VLDB Conference, Bombay, India (1996)
14. Gupta, A., Widom, J.: Local verification of global integrity constraints in distributed databases. In: ACM SIGMOD International Conference on Management of Data. (1993) 49–58
15. Grefen, P., Widom, J.: Integrity constraint checking in federated databases. In: Proceedings 1st IFCIS International Conference on Cooperative Information Systems. (1996) 38–47