UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

THE LOCAL RELATIONAL MODEL: MODEL AND PROOF THEORY

Luciano Serafini, Fausto Giunchiglia, John Mylopoulos
and Philip A . Bernstein

December 2001

Technical Report # DIT-02-009

The Local Relational Model: Model and Proof Theory

Luciano Serafini* Fausto Giunchiglia*,† John Mylopoulos†,‡ Philip A. Bernstein§

Abstract

In this paper we identify desirable data management mechanisms for peer-to-peer (P2P) computing. P2P networks have to remain open and dynamic, while peers remain autonomous and need only be aware of their immediate acquaintances. In such a setting, we argue that one cannot assume the existence of a global schema for all the peer databases. Instead, one needs a data model which views the space of data being managed within the P2P network as an open collection of possibly overlapping and inconsistent databases. Accordingly, the paper proposes the Local Relational Model and offers a formal semantics for coordination between peer databases. Our result generalizes Reiter's characterization of a relational database in terms of a first order theory, by providing a syntactic characterization of a relational space in terms of a multi-context system.

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. 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, where each peer may have data to share with other peers. For simplicity, we assume that each peer's database is relational. Since the data residing in different databases may have semantics inter-dependencies, we allow peers to specify coordination formulas that explain how the data in one peer must relate to data in an acquaintance. For example, the patient database of a family doctor and that of a pharmacy may want to coordinate their information about a particular patient, the prescriptions she has received, and the dates when these prescriptions were filled. 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 a query expressed with respect to one database to be able to use relevant databases at acquaintances, acquaintances of those acquaintances, and so on. To accomplish this, we expect the P2P data management system to use coordination formulas for recursively decomposing the query into subqueries that are evaluated with respect to the databases of acquaintances. Coordination formulas may also act as soft constraints or guide the propagation of updates. In addition, peers need an acquaintance initialization protocol where two peers exchange views of their respective databases and agree on levels of coordination between them. The level of coordination should be dynamic, in the sense that acquaintances may start with little coordination, strengthen it over time with more coordination formulas, and eventually abandon it when tasks and interests change.

In such a dynamic setting, we cannot assume the existence of a global schema for all databases in a P2P network, or even those of all acquainted databases. Moreover, peers should be able to establish and evolve acquaintances, preferably with little human intervention. Thus, we need to avoid protracted tasks by skilled database designers and DBAs required by traditional distributed and multi-database systems [15, 1]. In [2] we

*ITC-IRST, 38050 Povo, Trento, Italy
†University of Trento, 38050 Povo, Trento, Italy
‡University of Toronto M5S 3H5, Toronto, Ontario, Canada
§Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399

introduce the intuitions underlying our proposed data-management model in P2P environment. In this paper we introduce the Local Relational Model (LRM) as a data model specifically designed for P2P applications. LRM assumes that the set of all data in a P2P network consists of local (relational) databases, each with a set of acquaintances, which define the P2P network topology. For each acquaintance link, domain relations define translation rules between data items, and coordination formulas define semantic dependencies between the two databases. Two of the main goals of the data model are to allow for inconsistent databases and to support semantic interoperability in the absence of a global schema.

The paper is structured as follows. Section 2 presents a motivating scenario. In Section 3 we characterize the LRM semantically in terms of *relational spaces*. A relational space is a pair consisting of a set of databases (the peers) and a domain relation which makes explicit the relations among the domains of the databases. The LRM semantics is a variation of the semantic of distributed first order logic [7], which itself is an extension of the *Local Models Semantics*, proposed in [8, 6]. Section 4 introduces coordination formulas that relate the contents of peer databases and define what it means for a coordination formula to be satisfied (with respect to a relational space). The crucial step in this definition is the quantification across the distinct domains of different databases. This section also illustrates the use of coordination formulas as *deductive rules* and it defines what it means to give a *global answer to a query* with respect to a relational space. The intuition is to compute the union of all the answers of the peer databases, taking into account the information carried by domain relations. Finally, Section 5 contains the main technical result in this paper. The section first proposes a calculus for handling coordination formulas which is proved correct and complete with respect to the semantics introduced in the previous sections. In [17], Reiter proves that any partial relational database can be uniquely represented by a generalized relational theory. We generalize this result by showing that a relational space is uniquely represented by a new kind of formal system called a *multi-context system*, consisting of a set of generalized relational theories (one per database) and a set of coordination rules. An important corollary of this result is the syntactic characterization of the notion of global answer to a query. This result can serve as foundation for sound and complete implementations of a query answering mechanism in a P2P environment.

2 A motivating scenario

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

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

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:

$\texttt{Patient}(\underline{\texttt{OHIP\#}}, \texttt{FName}, \texttt{LName}, \texttt{Phone\#}, \texttt{Sex}, \texttt{PatRecord})$ $\texttt{Visit}(\underline{\texttt{OHIP\#}}, \underline{\texttt{Date}}, \texttt{Purpose}, \texttt{Outcome})$
$\texttt{Prescription}(\underline{\texttt{OHIP\#}}, \underline{\texttt{Med\#}}, \texttt{Dose}, \texttt{Quantity}, \underline{\texttt{Date}})$ $\texttt{Event}(\underline{\texttt{OHIP\#}}, \underline{\texttt{Date}}, \texttt{Description})$

Figuring out patient record correspondences (i.e., doing object identification) is achieved by using the patient's Ontario Health Insurance # (e.g., $\texttt{OHIP\#} = \texttt{1234}$). 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:

$$\begin{array}{l}\forall fn.\forall ln.\forall pn.\forall sex.\forall pr.(\mathsf{Davis} : \texttt{Patient}(\texttt{1234}, fn, ln, pn, sex, pr) \rightarrow \\ \quad \mathsf{Tgh} : \exists tghid.\exists n.\exists a.(\texttt{Patient}(tghid, \texttt{1234}, n, sex, a, \texttt{Davis}, pr) \wedge n = concat(fn, ln)))\end{array} \tag{1}$$

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. This amounts to generating a new TGH# for *tghid*, inserting the Skolem constant <undef-age> for *a* (which will be further instantiated as the patient's age) and generating name *n* by concatenating her first name *fn* and last name *ln* 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:

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

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

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:

$$\mathtt{Prescription}(\mathtt{Prescr\#},\mathtt{CustName},\mathtt{CustPhone\#},\mathtt{DrugID},\mathtt{Dose},\mathtt{Repeats})$$
$$\mathtt{Sales}(\mathtt{CustName},\mathtt{CustPhone\#},\mathtt{DrugID},\mathtt{Dose},\mathtt{Date},\mathtt{Amount})$$

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

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

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 an interest group of Toronto pharmacies, 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:

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

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 [17]) 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 [17]. 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 3.1 (Relational Language). A first order language L is a relational language if:

1. L contains a finite set of unary predicates $\mathbf{A}$;
2. for each $A \in \mathbf{A}$, L contains a finite set of constant symbols dom_A; we suppose that for each $A \neq B$, dom_A and dom_B are disjoint sets;
3. L does not contain functional symbols;
4. L contains a finite set of predicate symbols $\mathbf{R}$.

$\mathbf{A}$ is the set of attributes, dom_A is the domain of attribute A and $\mathbf{R}$ is the set of relations of L. Furthermore, there is a mapping $\alpha : R \rightarrow \mathbf{A}^*$, such that, for each n-ary predicate symbol R, $\alpha(R) = \langle A_1, ..., A_n \rangle$ is an n-tuple of attributes, called the *attributes of R*. We use the following notation: $dom = \cup_{A \in \mathbf{A}} dom_A$ is called the *domain* of L; for each $R \in \mathbf{R}$, with $\alpha(R) = \langle A_1, \ldots, A_n \rangle$, $dom_R = dom_{A_1} \times \ldots \times dom_{A_n}$; $\mathbf{x}$ denotes a sequence of variables $\langle x_1, \ldots, x_n \rangle$; $\mathbf{d}$ denotes a sequence of elements $\langle d_1, \ldots, d_n \rangle$, each of which belongs to some domain; $\phi(x)$ is a formula with the free variable x, and $\phi(\mathbf{x})$ is a formula with free variables in $\mathbf{x}$. 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`; $\alpha(\texttt{Patient}) = \langle \texttt{OHIP\#}, \texttt{FName}, \texttt{LName}, \texttt{Phone\#}, \texttt{Sex}, and \texttt{PatRecord} \rangle$.

Definition 3.2 (First Order Interpretation). A *first order interpretation* $\langle D, m \rangle$ of a relational language L is a pair composed of a non empty set D, called its *domain*, and a function m that maps every constant d of L into an element $m(d) \in dom$, and every n-ary predicate R in L into an n-ary relation $m(R) \subseteq dom^n$.

Definition 3.3 (Relational Database). A first order interpretation $\langle D, m \rangle$ of a relational language L is a *relational database* if:

1. D is equal to dom;
2. for each $A \in \mathbf{A}$, $m(A) = dom_A$;
3. for each $d \in dom$, $m(d) = d$;
4. $m(=) = \{\langle d, d \rangle | d \in dom\}$;
5. for each $R \in \mathbf{R}$, $m(R) \subseteq dom_R$.

Since the domain of a relational database on L is fixed (i.e., the set of constants of L), a relational database on L is uniquely identified by the interpretation function m. Notice that the term "database" is used informally while the corresponding (semantic) formal notion is that of "relational database". In the following, when no confusion arises, we use the term "database" meaning also its formalization in terms of relational databases.

A *complete database* is one which does not contain null values or partial tuples. Notice that if m is a relational database, then, for any formula ϕ, either $m \models \phi$ or $m \models \neg\phi$ (where "$\models$" stands for "first order satisfiability"). In many cases, however, we have to deal with *incomplete databases*. A common approach is to characterize an incomplete database 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.

Since we are interested in modelling P2P applications, we take a further step and consider multiple, possibly incomplete, possibly (partially) overlapping, and possibly inconsistent databases. 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. We model this by assuming that there is a non-empty set I of indices/names of databases, and that, for each $i \in I$, L_i is the relational language associated with the local database i. Then, we associate to each L_i a set db_i of relational databases on L_i. We call each element of db_i a *local relational database*. Each local database is therefore characterised by a set of local relational databases, as follows.

Definition 3.4 (Sets of local relational databases). Given a family of relational languages $\{L_i\}_{i \in I}$, db is a total function which associates to each $i \in I$ a set db_i of local relational databases on L_i. db is called a *set of local databases (defined on $\{L_i\}_{i \in I}$)*.

In LRM, there is no notion of global consistency for a set of local databases. However, we do retain a notion of local consistency. Each local database can be in a (locally) consistent or inconsistent state, and consistent and inconsistent databases can coexist in a single relational space. For instance the local databases $db_a = \{m_1\}, db_b = \{m_2, m_3\}, db_c = \emptyset$ are respectively, complete, incomplete, and inconsistent Generally, db_i is complete if $|db_i| = 1$, incomplete if $|db_i| > 1$ and inconsistent if $db_i = \emptyset$.

In a relational space, overlapping databases represent information about a common part of the world. This overlap has nothing to do with the fact that the same constant appears in both databases. For instance, the fact that the constant `Apple` appears in a database describing computers and another describing Italian agricultural products does not imply that these databases overlap. Rather, overlap is determined by the meaning of constants, i.e., when the entities denoted by constants in different databases are 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 3.5 (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 $dom_i \times dom_j$. If r_{ij} is a domain relation, then $r_{ij}(d_i) = \{d_j | \langle d_i, d_j \rangle \in r_{ij}\}$.

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 many cases, domain relations 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 [18].

Example 3.1. *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 following 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 3.6 (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 3.2. *A relational space modeling the states of the database described in Section 2, is a pair $\langle db, r \rangle$, where the first component is a tuple $\langle db_{\mathsf{Tgh}}, db_{\mathsf{Davis}}, db_{\mathsf{Allen}}, db_{\mathsf{Tphh}}, db_{\mathsf{TNgh}} \rangle$ containing five sets of interpretations of the relational languages associated to* Tgh, Davis, Allen *and* Tphh *and* TNgh, *respectively; and the second component, r, is the tuple $\langle r_{\mathsf{DavisTgh}}, r_{\mathsf{TghDavis}}, r_{\mathsf{DavisAllen}}, r_{\mathsf{DavisTphh}} \rangle$ containing four domain relations between those databases which have to coordinate according to constraints (1–5).*

To represent the fact that $\langle \texttt{1234}, \text{"}\texttt{Pippo}\text{"}, \text{"}\texttt{Inzaghi}\text{"}, \texttt{444}, \texttt{M}, \texttt{Rec\_23} \rangle$ *is a row of the relation* `PatRecord` *of the* Davis *database, we impose* $\langle \texttt{1234}, \texttt{Pippo}, \texttt{Inzaghi}, \texttt{444}, \texttt{M}, \texttt{Rec\_23} \rangle \in m(\texttt{PatRecord})$ *for each interpretation $m \in db_{\mathsf{Davis}}$,*

To represent the fact that $\langle \texttt{TG64}, \texttt{1234}, \text{"}\texttt{PippoInzaghi}\text{"}, \texttt{M}, \texttt{<undef-age>}, \texttt{Davis}, \texttt{Rec\_23} \rangle$ *is a row of the relation* `Patient` *of* Tgh *database, we impose that, for each natural number n, with* $0 \leq n \leq \texttt{MaxAge}$, db_{Tgh} *contains a model a model m, with* $\langle \texttt{TG64}, \texttt{1234}, \text{"}\texttt{PippoInzaghi}\text{"}, \texttt{M}, n, \texttt{Davis}, \texttt{Rec\_23} \rangle \in m(\texttt{Patient})$.

To represent the fact that the `TGH# 1234` *uniquely identifies a patient in both* Tgh *and* Davis, *we impose that* $r_{\mathsf{DavisTgh}}(\texttt{1234}) = r_{\mathsf{TghDavis}}(\texttt{1234}) = \{\texttt{1234}\}$.

4 Coordination in relational spaces

Semantic inter-dependencies between local databases are expressed in a declarative language, independent of the languages supported by local databases. The formulas of this language describe properties of schemas as well as the contents of local databases in a relational space. This language is a generalization of interpretation constraints defined in [7].

Definition 4.1 (Coordination formula). The set of *coordination formulas CF* on the family of relational languages $\{L_i\}_{i \in I}$ is defined as follows:

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

where $i \in I$ and ϕ is a formula of L_i, and x is an individual variable of L_i[1].

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 $i : \phi$, also called *atomic coordination formulas* which means "ϕ is true in database i". Connectives have the usual meaning, while quantifiers require further consideration. The formula $\forall i : x.A(x)$ should be read as "for all elements of the domain dom_i, A is true". Likewise, $\exists i : x.A(x)$, is read as "there is an element in the domain dom_i such that A is true". Notice that a variable x in the scope of a quantifier $\forall i : x$ or $\exists i : x$ can occur in an atomic coordination formula $j : \phi(x)$ with $i \neq j$, allowing quantification across domains. Specifically, we allow that within the scope of a dom_i formula, one can quantify over another domain dom_j exploiting the domain relations r_{ij} and r_{ji}. We say that an occurrence of a variable x in a coordination formula is a *free occurrence*, if it is not in the scope of a quantifier

Example 4.1. *Consider the coordination between* Davis *and* Tgh *(5). Its reformulation in terms of coordination formula is:*

$$\begin{array}{l}\forall \mathsf{Davis} : fn.\forall \mathsf{Davis} : ln.\forall \mathsf{Davis} : pn.\forall \mathsf{Davis} : sex.\forall \mathsf{Davis} : pr.(\mathsf{Davis} : \texttt{Patient}(1234, fn, ln, pn, sex, pr) \rightarrow \\ \quad \mathsf{Tgh} : \exists tghid.\exists n.\exists a.(\texttt{Patient}(tghid, 1234, n, sex, a, \texttt{Davis}, pr) \wedge n = concat(fn, ln)))\end{array} \tag{6}$$

The issue now is to provide an interpretation of coordination formulas in terms of relational spaces. Let us start by considering Definition 4.1 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. $i : \phi$ 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 $j \neq i$. One such case can be found in Example 4.1, where, for instance, the variables fn and ln occur free in the consequence of the implication of (6 within a coordination formula with index Tgh, while they are bound by the quantifiers $\forall \mathsf{Davis} : fn$ and $\forall \mathsf{Davis} : ln$.

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

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

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

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

$r_{ik}(d)$, 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

$$\exists i : x.(i : P(x) \wedge j : Q(x) \wedge k : R(x)) \tag{8}$$

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 (8) 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 $i : P(x) \wedge j : Q(x) \wedge k : R(x)$ is $\{i, j, k\}$.

Definition 4.2 (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 (7), (8). In Equation (7) 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 (8), 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 4.3 (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 4.4 (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. $\langle db, r \rangle \models A \wedge B[a]$, if $\langle db, r \rangle \models A[a]$ and $\langle db, r \rangle \models B[a]$;
4. $\langle db, r \rangle \models A \vee B[a]$, 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 4.5 (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 4.6 (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

$$(i : P(x) \vee j : Q(x)) \wedge k : R(x,y) \rightarrow h : q(x,y)$$

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 4.6 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 $\{q_h := A_h(\mathbf{x}_h) \rightarrow i_h : q_h(\mathbf{x}_h)\}_{1\leq h\leq n}$ 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

$$\dfrac{\begin{array}{c} i:\phi_1 \quad \ldots \quad i:\phi_n \\ \boxed{\begin{array}{c}\phi_1 \quad \ldots \quad \phi_n \\ \vdots \; i\text{-rules} \\ \phi\end{array}} \end{array}}{i:\phi} \qquad \dfrac{}{i:x=y \vee i:x \neq y}\ (=\vee) \qquad \dfrac{}{\exists i:x.i:x=t}\ (dom_i)$$

$$\dfrac{A \quad B}{A \wedge B}\ (\wedge\mathrm{I}) \qquad \dfrac{A/B}{A \vee B}\ (\vee\mathrm{I}) \qquad \dfrac{\begin{array}{c}[A] \\ B\end{array}}{A \to B}\ (\to\mathrm{I})$$

$$\dfrac{A \wedge B}{A/B}\ (\wedge\mathrm{E}) \qquad \dfrac{A \vee B \quad \begin{array}{c}[A] \\ C\end{array} \quad \begin{array}{c}[B] \\ C\end{array}}{C}\ (\vee\mathrm{E}) \qquad \dfrac{A \quad A \to B}{B}\ (\to\mathrm{E})$$

$$\dfrac{\begin{array}{c}[\bigwedge_{k=1}^{n} \exists i_k : y.i:y=x] \\ A^x_{x^{i\to}}\end{array}}{\forall i:x.A}\ (\forall i\mathrm{I}) \qquad \dfrac{A^x_{x^{\to i}}/A^x_{x^{i\to}}}{\exists i:x.A}\ (\exists i\mathrm{I})$$

$$\dfrac{\forall i:x.A \quad \bigwedge_{k=1}^{n} \exists i_k : y.i:y=x}{A^x_{x^{\to i}}/A^x_{x^{i\to}}}\ (\forall i\mathrm{E}) \qquad \dfrac{\exists i:x.A \quad \begin{array}{c}[A^x_{x^{\to i}}] \\ C\end{array}}{C}\ (\exists i\mathrm{E})$$

Notation:
***i*-rules:** If ϕ can be derived from $\phi_1, \ldots, \phi_n$ by applying ND rules for classical logic, then $i:\phi$ can be inferred from $i:\phi_1, \ldots i:\phi_n$.
X/Y stands for both X and Y.
Square brackets around formulas represent assumption discharging.
For each connective and quantifier there is an introduction and an elimination rule denoted by the connectives or quantifier followed by I and E respectively.

Restriction: In $(\forall i\mathrm{I})$ and $(\forall i\mathrm{E})$, J is the coordination space of x in A. $(\forall i\mathrm{I})$ is applicable only if x does not occur free in any assumption $A^x_{x^{i\to}}$ depends on. $(\exists i\mathrm{E})$ is applicable only if x does not occurs free in C, nor in any assumption, different from $A^x_{x^{\to i}}$ C, depends on.

Figure 1: Inference rules for Coordination Formulas.

Theorem 4.2. *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 $ans_1, \ldots, ans_n$, 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

Let us now provide a proof-theoretic (deductive) account of coordination among databases. We define the derivability relation $\vdash$ in terms of a Natural Deduction (ND) system [16] with rules as described in Figure 1. To define the calculus we need to introduce a new set of variables, called *arrow variables*. For any variable x, the expression $x^{\to i}$ and $x^{i\to}$ is an arrow variable. Intuitively, the variable $x^{\to i}$ occurring in an atomic coordination formula with index j denotes any element of dom_j that is the pre-image (via r_{ji}) of the element of dom_i denoted by x. Analogously the variable $x^{i\to}$ occurring in an atomic coordination formula with index j, denotes any element of dom_j that is the image (via r_{ij}) of the element of dom_i denoted by x.

Definition 5.1 (Universal and existential substitution). Let A be a coordination formula and x a variable. The *from-i-substitution* of x in A, denoted $A^x_{x^{i\to}}$, is the formula obtained by replacing each free occurrence of x in an atomic coordination formula with index j distinct from i with $x^{i\to}$. Likewise, the *to-i-substitution* of x in A, denoted $A^x_{x^{\to i}}$, is obtained by replacing each occurrence of x in an atomic coordination formula with index j distinct from i with $x^{\to i}$.

Examples of deductions will be given in the final version of the paper.

Theorem 5.1 (Soundness and completeness). *A coordination formula A is a logical consequence of a set of coordination formulas Γ if and only if there is a deduction of A from Γ.*

Proof outline. The proof is a standard soundness and completeness proof for a deduction system. Soundness is proved by induction by showing that each rule preserves satisfiability under assignments. Completeness is proved by showing that an i-consistent set of formulas Γ, i.e., $i:\bot$ is not derivable from Γ, has a canonical relational space. A similar construction, restricted to the propositional case is given in [8, 6]. Details of the proof will be provided in the full paper. □

The completeness result given above allows us to generalize Reiter's syntactic characterization of relational databases to relational spaces. We start by recalling Reiter's result (in a slightly different, but equivalent, formulation).

Definition 5.2 (Generalized relational theory). A theory T on the relational language L is a *generalized relational theory* if the following conditions hold.

Domain closure: if $dom = \{d_1, \ldots d_n\}$, then T contains the axiom $\forall x (x = d_1 \vee \ldots \vee x = d_n)$.

Unique names: For any $d, d' \in dom$, T contains the axiom $d \neq d'$.

Predicate extension: For any relational symbol $R \in \mathbf{R}$, there is a finite number of finite sets of tuples $E_R^1, \ldots, E_R^n$ (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 5.3 (Domain relation extension). Let r_{ij} be a domain relation. The *set of coordination formulas for the extension of* r_{ij} is a set R_{ij} that contains the following coordination formulas for any $d \in dom_i$:

1. if $d' \in r_{ij}(d)$ the coordination formula $\exists j : x.(i : x = d \wedge j : x = d')$;
2. if $d' \notin r_{ij}(d)$ the coordination formula $\forall i : x.(i : x = d \rightarrow j : x \neq d')$

Definition 5.3 axiomatizes r_{ij} as a relation between the domains i and j.

Lemma 5.2. *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 $r_{ij} = r'_{ij}$.*

Lemma 5.2 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}. With this lemma we can state the main representation theorem (Theorem 5.3). A corollary of this theorem (Corollary 5.4) provides a proof-theoretic characterization of a global answer to a i-query.

Definition 5.4 (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 5.3 (Representation of relational space). *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 *$\langle db, r \rangle$.*

Proof outline. The proof is a composition of the previous lemma on R_{ij} and Reiter's result on T_i. Details of this and the previous proofs will be provided in the final version of the paper. □

Corollary 5.4 (Syntactic characterization of queries). *For any relational space $\langle db, r \rangle$, let $\langle T, R \rangle$ be the relational multi-context system that represents $\langle db, r \rangle$. Then, 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*

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

Corollary (5.4) 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 [7]. The main improvements concern the language of the coordination formulas, their semantics and the calculus. In [7] 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 $\forall i : x \exists j : y i : x = y$ and $\forall j : x \exists i : y i : x = y$, 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 ([12, 13]. 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. In any case, such query processing issues are beyond the scope of the current paper, whose main focus is the formal definition of LRM and a proof of its soundness and completeness.

Finally our approach provide a general theoretical reference framework where many forms of inter-schema constraints defined in the literature, such as [3, 4, 5, 14, 19, 11, 10, 9]. For lack of space we briefly show only one case. Consider for instance directional existence dependences defined in [5]. 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:

$$T.(C_1, C_2) \Leftarrow \textbf{ select } X_1, X_2 \textbf{ from } T_1, T_2 \textbf{ where } T_1.X_1 \leq T_2.X_2 \tag{9}$$

The informal semantics of (9) 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 (9), can be rewritten in terms of coordination formulas as

$$\begin{aligned} \forall 1 : x_1 x_2 (1 : \exists y_1 y_2 (T_1(x_1, y_1) \wedge T_2(x_2, y_2) \wedge x_1 \leq x_2) \rightarrow \\ \exists 2 : c_1 c_2 (1 : x_1 = c_1 \wedge x_2 = c_2 \wedge 2 : \exists c_3 . T(c_1, c_2, c_3))) \end{aligned} \tag{10}$$

When the domain relation are identity functions, (10) capture the intuitive reading of (9).

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 and proof theory for the LRM, and prove the latter sound and complete with respect to the former. The paper also generalizes 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] *Principles of Distributed Database Systems*. Prentice-Hall, 1999.

[2] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *WebDB 2002: Fifth International Workshop on the Web and Databases*, 2002.

[3] M.J. Carey, L.M. Haas, P.M. Schwarz, Manish Arya, W.F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas II, J.H. Williams, and E.L. Wimmers. Towards heterogeneous multimedia information systems: The garlic approach. *RIDE-DOM*, pages 124–131, 1995.

[4] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[5] S. Ceri and J. Widom. Managing semantic heterogeneity with production rules and persistent queues. In R. Agrawal, S. Baker, and D.A. Bell, editors, *19th International Conference on Very Large Data Bases, August 24–27, 1993, Dublin, Ireland, Proceedings*, pages 108–119. Morgan Kaufmann, 1993.

[6] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.

[7] C. Ghidini and L. Serafini. Distributed First Order Logics. In D. Gabbay and M. de Rijke, editors, *Frontiers Of Combining Systems 2*, Studies in Logic and Computation, pages 121–140. Research Studies Press, 1998.

[8] F. Giunchiglia and C. Ghidini. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 282–289. Morgan Kaufmann, 1998. Also IRST-Technical Report 9701-07, IRST, Trento, Italy.

[9] Paul W. P. J. Grefen and Jennifer Widom. Protocols for integrity constraint checking in federated databases. *Distributed and Parallel Databases*, 5(4):327–355, 1997.

[10] P.W.P.J. Grefen and J. Widom. Integrity constraint checking in federated databases. In *Proceedings 1st IFCIS International Conference on Cooperative Information Systems*, pages 38–47, 1996.

[11] A. Gupta and J. Widom. Local verification of global integrity constraints in distributed databases. In *ACM SIGMOD International Conference on Management of Data*, pages 49–58, 1993.

[12] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10, 2001.

[13] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[14] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of the 22nd VLDB Conference*, Bombay, India, 1996.

[15] W. Litwin, L. Mark, and N. Roussapoulos. Interoperability of multiple autonomous databases. *ACM Computing Survey*, 22(1):267–293, 1990.

[16] D. Prawitz. *Natural Deduction - A proof theoretical study*. Almquist and Wiksell, Stockholm, 1965.

[17] R. Reiter. Towards a Logical Reconstruction of Relational Database Theory. In M.L. Brodie, J. Mylopoulos, and J.W. Schmidt, editors, *On Conceptual Modelling*, pages 191–233. Springer-Verlag, 1984.

[18] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, June 1994.

[19] Jeffrey D. Ullman. Information Integration Using Logical Views. In *Proc. of the 6th International Conference on Database Theory (ICDT'97)*, 1997.