Logic In Databases: Report on the LID 2008 Workshop.

Andrea Calì
Oxford-Man Institute of Quantitative Finance
University of Oxford
Wolfson Building, Parks Road
Oxford OX1 3QD
United Kingdom
andrea.cali@comlab.ox.ac.uk

Laks V.S. Lakshmanan
Dept. of Computer Science
University of British Columbia
2366 Main Mall
Vancouver, B.C.
Canada V6T 1Z4
laks@cs.ubc.ca

Davide Martinenghi
Dip. di Elettr. e Informazione
Politecnico di Milano
Via Ponzio, 34/5
I-20133 Milano
Italy
martinen@elet.polimi.it

1. INTRODUCTION

The Logic in Databases (LID'08) workshop was held at the DIS department of "La Sapienza" university, Rome, Italy, between May 19-20, 2008.

LID'08 was established as a forum for bringing together researchers and practitioners, from the academia and the industry, who are focusing on all logical aspects of data management.

LID'08 was a confluence of three successful past events:

- LID'96, an international workshop on Logic in Databases, which LID'08 derives its name from;
- IIDB'06, an international workshop on Inconsistency and Incompleteness in Databases;
- LAAIC'06, an international workshop on Logical Aspects and Applications of Integrity Constraints.

In order to guarantee its continuity, a Steering Committee, chaired by Georg Gottlob, was founded; its members are Andrea Calì, Jan Chomicki, Henning Christiansen, Laks V.S. Lakshmanan, Davide Martinenghi, Dino Pedreschi, Jef Wijsen, and Carlo Zaniolo.

This workshop was organized by Andrea Calì, Laks V.S. Lakshmanan, and Davide Martinenghi; it attracted 18 paper submissions out of which 7 were selected for long presentation and 6 for short presentation at the workshop; the workshop attracted around 50 registered participants. Details and presentations are available at the workshop Web site: `http://conferenze.dei.polimi.it/lid2008/`.

2. WORKSHOP AIMS AND CONTENTS

Ever since Codd's Relational Model, logic has played a major role in the field of databases. The significance and impact of this role have grown stronger over the years as data management research marched through many a data model, with logic keeping up and providing the foundations every step of the way. Some of the latest additions to this long list of models are XML, Semantic Web, Probabilistic Relational models, integrated model of DB+IR, data integration models, and models of unclean data to name a few. For some of these, corresponding logics already exist or are being explored. The significance of logic's role for data management will continue regardless of the data model.

Logic is a fundamental tool for understanding and analyzing several aspects of data management, as Georg Gottlob (University of Oxford) said during his opening remarks. The three keynote presentations, "Changing the Instance Level of Ontologies: A Logic Approach" by Maurizio Lenzerini, "From Consistent Query Answering to Query Rewriting: A Detour around Answer Set Programs" by Leopoldo Bertossi, and "Databases Meet Verification: What do we have in common, and how can we help each other?" by Leonid Libkin, represented well some of the most promising areas of research that emerged during the workshop. Among these, we mention new perspectives on data integration (P2P, ontology-based, etc.), algebraic characterizations of languages, inconsistency and incompleteness in databases (approximation, tolerance, repairs), query rewriting techniques for advanced query processing, characterizations of tractable fragments of languages for XML processing and their use, indexing techniques. In the next sections we summarize the relevant research problems and trends that were identified during the workshop, and organize them in three macro-areas corresponding to the sessions that were held during the workshop: advanced query processing, incompleteness and inconsistency, and semi-structured data.

3. ADVANCED QUERY PROCESSING

Processing queries is the ultimate task in information systems that integrate several data sources [35]. This problem can be often reformulated as the one of answering queries under constraints, in the presence of incomplete information [47]; it has been addressed by several authors in the literature, who have considered standard and less-standard

database constraints, for instance, inclusion dependencies, functional dependencies, tuple-generating and equality-generating dependencies. At the same time, Description Logics (DLs) [5] seem to be a natural candidate as a constraint languages for representing possibly incomplete information residing at different sources, as well as being a suitable language for several applications in the Semantic Web. *Conjunctive query rewriting* under DL-constraints has been considered in [13] for the tractable DL-Lite family of DLs; query rewriting consists of rewriting a given query into another one that encodes information about the constraints; then, the evaluation of the latter on the data returns the correct answer to the former with respect to the constraints. The paper [43] deals with answering queries on databases that are incomplete with respect to a knowledge base expressed in Description Logics. The authors here consider the description logic $\mathcal{ELHIO}^-$ to express constraints; such formalism is capable of expressing several description logics in the DL-Lite and the $\mathcal{EL}$ family. In the paper, a general query rewriting technique is presented, that shows that the query answering problem is PTIME-complete in data complexity, i.e., considering as input only the data. The result extends to the aforementioned description logics captured by $\mathcal{ELHIO}^-$, being worst-case optimal at the same time for all of them.

In a *peer-to-peer* setting, several peer nodes store information, and each peer provides answers to queries posed in a shared language based on its locally stored data and by querying other peers; *semantic mappings* specify the correspondence among data at different peers. Works in the literature address the problem of peer-based information integration by adopting epistemic logic to account for the modular nature of peer information sharing in peer-to-peer systems [14]. The paper [48] points out that such works consider the information residing at a peer as facts with respect to *possible worlds* that the peer can access. Therefore, the problem of dealing with conflicting information between peers arises in this approach; while some works propose the notion of *repair* of inconsistent data, the problem has been so far addressed by proposing a normative strategy for all peers. [48] instead proposes a more general way of modelling peer knowledge, in particular, by separating *knowledge of statements* from *knowledge of facts*. This approach is called *doxastic* from the Greek δόξα (opinion), and allows for a variety of strategies for integrating information in a peer-to-peer environment. The authors set up a formal framework for the doxastic approach to peer-based information integration, where each peer combines its direct knowledge with the indirect knowledge at other peers, and related to the one at the peer by means of the mapping assertions. The proposed approach is highly flexible and allows for a consistent, integrated account of the knowledge gathered from other peers and a peer's own knowledge; in particular, this formalization overcomes the limitations of current approaches such as *transfer* [36] or *routing* [38].

$\mathcal{K}$-relations [33] are relations where a value belonging to a semiring is assigned to each tuple. $\mathcal{K}$-relations are able to model bag and set semantics in the standard relational model, imcomplete databases, and probabilistic data. In $\mathcal{K}$-relations, common operations on tuples are represented by operations in the semiring. In [30] the authors first present different extensions of the positive relational algebra on $\mathcal{K}$-relations, showing additional conditions required by the corresponding semirings. Then, they extend the *provenance semiring* found in the literature, so as to record the provenance of results of queries in the aforementioned extended relational algebras. Finally, they extend the notion of BP-completeness [42], which is language-independent, to $\mathcal{K}$-relations, determining which of the introduced extended languages are BP-complete, depending on the properties of the corresponding semiring.

In certain settings, *preferences* on objects (or tuples) are relevant [34]. Sometimes, in making a decision based on relational data, a user has to consider properties of *sets* of objects, and expresses preferences over such sets [9]. The paper [50] focuses on set preferences, and considers two main components: *(1) profiles*, which are collections of features, each of which representing a quantity of interest; *(2) profile preference relations*, which specify values or orders. The preferences are defined over sets that have all the same, fixed cardinality. A feature, in particular, is a function $\mathcal{A} : k\text{-}subsets(r) \rightarrow U$, where $k\text{-}subsets(r)$ is the set of all subsets of cardinality k of tuples of relation r, and U is either the set of rational numbers or uninterpreted constants. A profile relation can be specified as a set $\{\langle \mathcal{A}_1(s), \ldots, \mathcal{A}_m(s)\rangle \mid s \in k\text{-}subsets(r)\}$, where $\mathcal{A}_1(s), \ldots, \mathcal{A}_m(s)$ are features. Profile-based set preferences on r are then expressed by ordering tuples in a profile relation. In [50], the authors present a heuristic algorithm for computing the "best" sets.

4. INCOMPLETENESS AND INCONSISTENCY

A database instance is said to be *inconsistent* if it does not satisfy its integrity constraints (ICs). Inconsistent databases arise in a variety of contexts and for different reasons, e.g., in data warehousing of heterogeneous data obeying different integrity constraints or for lack of support for particular integrity constraints. Two different approaches to inconsistency were given particular emphasis during the session: database *repairs* and *consistent query answering* (CQA) [3].

Database repairs provide a framework for coping with inconsistent databases in a principled way. Informally, a repair is a new instance D' obtained from the initial database D such that D' satisfies the ICs and D' differs from D in a minimal way. Several different types of repairs have been considered: subset-repairs; $\oplus$-repairs (symmetric-difference-repairs); cardinality-based repairs; attribute-based repairs. A tuple t is then a *consistent answer* to a query q in D if t is an answer to q in every repair D' of D.

The CQA problem (finding all consistent answers to a query) has traditionally been tackled by *query rewriting* based on a fixpoint operator T^ω: given a query q, find q' such that the answers to q' in D are the consistent answers to q in D. However, the T^ω rewriting approach has limitations and does not work for full first-order queries and ICs.

An alternative view to the described model-theoretic definition of consistent answer consists in representing the class of all database repairs in a compact form by using disjunctive logic programs (DLP) with stable model semantics [31], a.k.a. *Answer Set Programs* [20]. Repairs correspond then to distinguished models of the program. An ASP-based specification of repairs and consistent answers as consequences

from a program provide a sort of (non-classical) logic for CQA, with the advantage that DLP is a general methodology that, unlike previous approaches, works for universal ICs, referential ICs, and general FO queries. However, instead of completely computing all the stable models, it is preferable to try to generate "partial" repairs to optimize the access to the DB [25, 40].

It is pointed out in [1] that, although it underlies CQA, the repair checking problem (i.e., given D, the ICs, and D', tell whether D' is a repair of D w.r.t. the ICs) has received less attention than CQA so far. It is therefore advocated to embark on a systematic investigation of the algorithmic aspects of the repair checking problem, by studying classes of integrity constraints that have been considered in information integration and data exchange. This leads to the study of subset-repairs and $\oplus$-repairs, and to the introduction of the new CC-repairs (component-cardinality repairs), a new type of cardinality-based repairs that have a Pareto-optimality character. Several complexity results are known both for CQA and for repair checking [16, 29, 4, 12, 10], and new results are presented in [1], with particular attention to *weakly acyclic sets of tuple-generating dependencies*. The DLP approach and the traditional CQA approach often coincide, but, for some classes of queries and ICs, CQA has a lower complexity. This aspect should be investigated further. New avenues of research are opened by the use of the ASP-based logic for CQA. For example, the new problem of *CQA under updates* could be analyzed by taking advantage of results about updates of logic programs. Also, known rewritings for CQA (such as those given by T^ω in [3]) as well as new ones can be obtained by using first-order specification of repairs via elimination of SO quantifiers [41] and subsequent application of Ackermann's lemma.

CQA has also been applied to *database histories* (i.e., finite sequences of states) under primary keys. Histories can be represented by words. In particular, in [21], violations of key constraints in a database history are be modeled by *multiwords*, which are a compact representation of sets of possible words. Questions on multiwords suitably characterize queries on database histories; if words are also allowed to contain variables, i.e., placeholders for constants, larger classes of queries can be captured. Similarly to the notion of certain answer, a word w is *certainly contained* in a multiword M if w is a sub-word of every possible word in M. The problem at stake here is to characterize the language $CERTAIN(w)$, defined as the set of multiwords that certainly contain w. Results on complexity, regularity and FO-definability of $CERTAIN(w)$ have been presented. Among the open questions, [21] presents a conjecture that $CERTAIN(w)$ is FO-definable if w is variable-free, and discusses the need for a syntactic characterization of the words v for which $CERTAIN(v)$ is FO-definable.

Inconsistency in databases can also be tackled in a more lenient way that tolerates the presence of IC violations rather than trying to repair them. The paper [22] classifies several methods of checking integrity with respect to so-called *inconsistency tolerance*, a notion that has been gaining momentum [8]. Among the aforementioned methods, a new one has emerged that tries to apply the principle of inconsistency tolerance to those tools that are actually used to prevent inconsistency from sneaking in after an update, i.e., integrity checking methods [23]. Traditional methods are capable of incrementally checking integrity by assuming that the initial database state fully satisfies the ICs. Many such methods have been shown to be inconsistency-tolerant, and can thus be applied also to inconsistent databases; in this case, they can be used to guarantee that an update will not introduce *new* inconsistency. Doors are open to combine inconsistency-tolerant integrity checking with Knowledge Assimilation, Semantic Query Optimization, CQA, and Inconsistency Measuring [24].

Data are naturally characterized as *incomplete* in several contexts, for instance when data from different sources are integrated. In an arbitrary relational database, the *Closed World Assumption* (CWA) tags as false all those tuples that are not in the database; however, the CWA is not the correct approach when the database is incomplete. The *Open-World Assumption* (OWA), common in data integration systems, assumes that the world can be in any state in which all database atoms are true. The OWA is often too incomplete and underestimates the knowledge in a database, as pointed out in [17]. A compromise between CWA and OWA should better identify those parts of the database that are complete. For example, we may assume that the database of the computer science department knows *all* the telephone numbers of people working there; complete knowledge may not hold for other kinds of facts. Different approaches exist to specify that the database is partially complete, e.g., the *Local Closed-World Assumption* (LCWA) [37, 18]. Tractable methods based on fixpoint techniques for finding under-approximations of certain answers and over-approximations of possible answers are discussed in [17]. Ongoing work in the field includes *i)* refinements of the class of LCWA for which the query answering methods are complete, and *ii)* integration of integrity constraints and views into the techniques.

In the setting of incomplete data with an underlying schema, techniques have been developed to directly query the data through the schema. Lately, particular attention has been devoted to a practically relevant extension of the ER model known as *Extended Entity-Relationship* (EER) model [11], which can be translated to suitable logic programs and queried. This context has been regarded from a different perspective in [2]: support (EER) design activity via automated snapshot generation where a formal validation would not be easily available. In particular, in [2] it is shown that, given an EER schema, this can be translated in a way that allows generating small *informative example instances* for the schema, based on the recent notion of *Informative Armstrong database* [39]. This provides information about the structure of the database by letting the user inspect a suitably small instance that satisfies all the constraints implied by the schema.

5. SEMI-STRUCTURED DATA

Semi-structured data have been playing an important role for several years both in academia and in industry. Several efforts have been done to "export" database operations from the well-established relational database world to semistructured data, in particular in the XML format. XML is becoming a standard language for semistructured data, and

several formalisms are emerging for querying XML, and for expressing integrity constraints on XML data. An XML document can be seen as a labeled tree with a finite number of nodes. The W3C Resource Description Framework (RDF) represents data in the form of triples, by flattening the hierarchy between objects and relationships among them; therefore, it also blurs the distinction between data and metadata. Querying RDF is a topic of practical relevance that has raised significant interest.

The paper [6] addresses XPath, a language for navigating XML documents, and investigates some of its foundational aspects. Core XPath, previously introduced by Gottlob and Koch [32], captures the navigational core of XPath 1.0. A complete axiomatization is a set of valid equivalence schemes between XPath expressions, such that every equivalence is derivable from those in the set by repeatedly applying the equivalences; axiomatizations of XPath fragments have been proposed in the past [7, 46]. As shown in [6], it is possible to have an axiomatization for the *single-axis* fragments of Core XPath, both for node and path expressions. Another axiomatization is presented for the full language Core XPath, which however is *non-orthodox*, i.e., it requires an additional inference rule that has extra syntactic conditions. This sets the basis for XPath query optimization, where equivalent but more efficient queries are to be determined.

Constraints in XML are important to specify characteristics of documents in a specific application domain. Constraints can be classified into *structured* and *data value* constraints [26]. *Regular XPath (RXPath)*, introduced in [15], is a novel language for both expressing XML structural constraints and to express queries over XML trees. Regular XPath is derived from XPath, in particular by extending XPath with nominals and binary relations on XML nodes that are expressed as two-way regular expressions over XPath axes. Nominals denote a single node in an XML document, similarly to the `ID` XML construct. It can be shown that satisfiability of RXPath constraints can be reduced to reasoning in *Repeat-Converse-Deterministic PDL*; however, the reduction is of little practical use, due to the poor efficiency of reasoners in Repeat-Converse-Deterministic PDL. The work [15] presents therefore a direct EXPTIME decision algorithm for the problem; the algorithm is based on checking emptiness of two-way alternating automata on finite trees [19]. The technique is shown to be practically applicable, since it can be implemented symbolically, based on Binary Decision Diagrams. Moreover, query containment and view-based query answering in the language RXPath can both be reduced to the aforementioned satisfiability checking problem.

Often, potentially large sets of XML documents can have a compact representation in terms of a finite tree automaton, which serves as a schema. In such cases, evaluating a query usually assumes that the documents all satisfy the schema; in case the evaluation is made on a document that instead does not satisfy the schema, results can be incorrect. Similar to what is done in relational databases, a notion of *repair* can be adopted here (see, e.g., [3] and Section 4, and also [27] in the XML context): an answer is consistent if it is an answer to the query in all repairs. It can be shown that, also in this setting, the set of repairs admits a compact representation in terms of a finite, weighted tree automaton [45]. Together with *universal* answers (the aforementioned consistent answers), *existential* answers are treated, which correspond to possible answers (that are answers to the query in at least one repair). In the work [45], a thorough study on the complexity of evaluating universal and existential answers is carried out.

Query processing on RDF data can benefit from research on data modeling with ternary relations, which was very early recognized as an important tool in logic. SPARQL is the W3C recommended language for querying RDF triple stores. As shown in [28], at the core of SPARQL stands a small logic, called BGP (Basic Graph Pattern). BGP is suitable for extracting subsets of related nodes in an RDF graph. The optimization of SPARQL query evaluation can take advantage of a fundamental investigation on BGP. The work [28] presents an algebraization of BGP and introduces the basis for the development of structural indexes for RDF, with the aim of speeding up query processing (see, e.g., [44, 49]).

6. REFERENCES

[1] F. Afrati and P. Kolaitis. On the complexity of repair checking in inconsistent databases. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[2] M. Amalfi and A. Provetti. From extended entity-relationship schemata to illustrative instances. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS'99*, pages 68–79, 1999.

[4] M. Arenas and M. I. Schwartzbach, editors. *Database Programming Languages, 11th International Symposium, DBPL 2007, Vienna, Austria, September 23-24, 2007, Revised Selected Papers*, volume 4797 of *Lecture Notes in Computer Science*. Springer, 2007.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[6] M. M. Balder ten Cate, Tadeusz Litak. Complete axiomatizations for XPath fragments. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[7] M. Benedikt, W. Fan, and G. M. Kuper. Structural properties of XPath fragments. *Theor. Comput. Sci.*, 336(1):3–31, 2005.

[8] L. E. Bertossi, A. Hunter, and T. Schaub, editors. *Inconsistency Tolerance [result from a Dagstuhl seminar]*, volume 3300 of *Lecture Notes in Computer Science*. Springer, 2005.

[9] M. Binshtok, R. I. Brafman, S. E. Shimony, A. Mani, and C. Boutilier. Computing optimal subsets. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007)*, pages 1231–1236, 2007.

[10] L. Bravo and L. E. Bertossi. Semantically correct query answers in the presence of null values. In *EDBT Workshops*, pages 336–357, 2006.

[11] A. Calì. Querying incomplete data with logic programs: Er strikes back. In *ER*, pages 245–260, 2007.

[12] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, pages 260–271, 2003.

[13] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[14] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the 23rd ACM-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004)*, pages 241–251, 2004.

[15] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Regular XPath: Constraints, query containment and view-based answering for XML documents. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[16] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.

[17] A. Cortés-Calabuig, M. Denecker, O. Arieli, and M. Bruynooghe. Efficient fixpoint methods for approximate query answering in locally complete databases. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[18] A. Cortés-Calabuig, M. Denecker, O. Arieli, B. V. Nuffelen, and M. Bruynooghe. On the local closed-world assumption of data-sources. In *LPNMR*, pages 145–157, 2005.

[19] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proc. of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 477–490, 1988.

[20] V. Dahl and P. Wadler, editors. *Practical Aspects of Declarative Languages, 5th International Symposium, PADL 2003, New Orleans, LA, USA, January 13-14, 2003, Proceedings*, volume 2562 of *Lecture Notes in Computer Science*. Springer, 2003.

[21] A. Decan and J. Wijsen. On first-order query rewriting for incomplete database histories. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[22] H. Decker. Classifying integrity checking methods with regard to inconsistency tolerance. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[23] H. Decker and D. Martinenghi. A relaxed approach to integrity and inconsistency in databases. In M. Hermann and A. Voronkov, editors, *13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Phnom Penh, Cambodia, 13-17 November 2006*, volume 4246 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2006.

[24] H. Decker and D. Martinenghi. Classifying integrity checking methods with regard to inconsistency tolerance. In *Proceedings of PPDP 2008, 15-17 July 2008, Valencia, Spain.*, pages 195–204, 2008.

[25] T. Eiter, M. Fink, G. Greco, and D. Lembo. Efficient evaluation of logic programs for querying data integration systems. In *(ICLP 2003)*, pages 163–177, 2003.

[26] W. Fan. Xml constraints: Specification, analysis, and applications. In *Proc. of the 1st International Workshop on Logical Aspects and Applications of Integrity Constraints (LAAIC 2005)*, pages 805–809, 2005.

[27] S. Flesca, F. Furfaro, S. Greco, and E. Zumpano. Repairs and consistent answers for xml data with functional dependencies. In *Proc. of the 1st International XML Database Symposium (Xsym 2003)*, pages 238–253, 2003.

[28] G. H. L. Fletcher. An algebra for basic graph patterns. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[29] A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. In *Proc. of ICDT 2005*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.

[30] F. Geerts and A. Poggi. On bp-complete query languages on k-relations. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[31] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th Logic Programming Symposium*, pages 1070–1080. The MIT Press, 1988.

[32] G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *Proc. of the 17th IEEE Symposium on Logic in Computer Science (LICS 2002)*, pages 189–202, 2002.

[33] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proc. of the 26th ACM-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2007)*, pages 31–40, 2007.

[34] W. Kießling. Foundations of preferences in database systems. In *Proc. of 28th International Conference on Very Large Data Bases (VLDB 2002)*, pages 311–322, 2002.

[35] M. Lenzerini. Information integration: A theoretical perspective. In *Proc. of the 21st ACM-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[36] M. Lenzerini. Principles of p2p data integration. In *Third International Workshop on Data Integration over the Web (DIWeb 2004)*, pages 7–21, 2004.

[37] A. Y. Levy. Obtaining complete answers from incomplete databases. In *VLDB*, pages 402–412, 1996.

[38] Z. Majkic. Intensional semantics for p2p data integration. *Journal of Data Semantics*, 6:47–66, 2006.

[39] F. D. Marchi and J.-M. Petit. Semantic sampling of existing databases through informative armstrong databases. *Inf. Syst.*, 32(3):446–457, 2007.

[40] M. C. Marileo and L. E. Bertossi. The consistency extractor system: Querying inconsistent databases using answer set programs. In H. Prade and V. S. Subrahmanian, editors, *SUM*, volume 4772 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 2007.

[41] A. Nonnengart and A. Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In E. Orłowska, editor, *Logic at Work: Essays Dedicated to the Memory of*

Helena Rasiowa, volume 24 of *Studies in Fuzziness and Soft Computing*, pages 307–328. Springer Physica-Verlag, 1998.

[42] J. Paredaens. On the expressive power of the relational algebra. *Information Processing Letters*, 7(2):107–111, 1978.

[43] H. Perez-Urbina, B. Motik, and I. Horrocks. Rewriting conjunctive queries under description logic constraints. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[44] M. Sintek and M. Kiesel. RDFBroker: A signature-based high-performance RDF store. In *Proc. of the 3rd European Semantic Web Conference (ESWC 2006)*, pages 363–377, 2006.

[45] S. Staworko, E. Filiot, and J. Chomicki. Querying regular sets of XML documents. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[46] B. ten Cate and M. Marx. Axiomatizing the logical core of xpath 2.0. In *Proc. of the 11th International Conference on Database Theory (ICDT 2007)*, pages 134–148, 2007.

[47] R. van der Meyden. Logical approaches to incomplete information. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publishers, 1998.

[48] G. Vetere, F. Venditti, and A. Faraotti. A doxastic approach to P2P information integration. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.

[49] D. Wood. Scaling the kowari metastore. In *WISE 2005 Workshops*, pages 193–198, 2005.

[50] X. Zhang and J. Chomicki. Profiling sets for preference queries. In *Proc. of the Workshop on Logic in Databases (LID 2008)*, 2008.