Model theoretic and fixpoint semantics for preference queries over imperfect data

Peter Vojtáš

Charles University and Czech Academy of Science, Prague
Peter.Vojtas@mff.cuni.cz

Abstract. We present an overview of our results on model theoretic and fixpoint semantics for a relational algebra using a model of many valued Datalog with similarity. Using our previous results on equivalence of our model and certain variant of generalized annotated programs, we base our querying on fuzzy aggregation operators (also called annotation terms, combining functions, utility functions). Using of fuzzy aggregation operators (distinct from database aggregations) enables us to reduce tuning of various linguistic variables. In practice we can learn fuzzy aggregator operators by an ILP procedure for every user profile. Our approach enables also integration of data from different sources via aggregation and similarity. Extending domains we discuss difference between fuzzy elements and fuzzy subsets. We also discuss an alternative, when all extensional data are stored crisp and fuzziness is in rules interpreting data, context and in user query.
Keywords: fuzzy Datalog, preference querying, correct and complete semantics,

Introduction.

Relational data model of E. F. Codd [3] was closely related to logic of predicate calculus. Connections of model theoretic semantics and proof theoretic (possibly also fixpoint) semantic was considered as a standard part of the development of a formal data model (see e.g. J. D. Ullman [15]).

Necessity to extend the expressivity of data modeling languages to include vagueness, preference, uncertainty and different forms of imperfection was stressed by several authors (see [2], [4], [5], [6], [11], [13], [14], just to mention a few). In this work we would like to present a model theoretic and fixpoint semantics for a relational algebra introduced in [12] and discuss several aspects of the system.

Our motivation is from the work on a system developing methods and tools for acquiring, managing, mining and presenting information in a heterogeneous environment [10]. First pilot application we work on is in the domain of labor market. Here we will apply our techniques of preference queries. Job offer can mention place, required skills, salary and age "young" . Job seeking can be interested in distance "close", salary "high" and specifying age "about_25". The problem is to assign a degree of matching between job offer and the user in a way, he gets best answers first. Moreover our system incorporates combining overall score from score for particular attributes (see e.g. [4]). We model combining

function by a special sort of logical connectives - fuzzy aggregation operators (please do not confuse them with database aggregations) - which can range from conjunctions to disjunctions, but most typically are somewhere "in the middle", expressing fulfillment of most of requirements wrt. some weighting of importance of attributes.

Preference, better answer is modeled using many valued logic - the bigger the truth value the better the answer.

Fuzzy Datalog.

We follow the model of fuzzy logic programming developed in [16] and [8] and implement usual restrictions which make them Datalog programs. Our language is typed $[0,1]$-valued predicate logic. The only connectives are implications and aggregations @. Aggregations have truth functions fuzzy aggregation operators $@^\bullet$ which are monotone in all variables and $@^\bullet(0,...,0) = 0$ and $@^\bullet(1,...,1) = 1$. Aggregations cover all sorts of fuzzy conjunctions and disjunctions. We have no negation here.

Declarative semantics is based on the notion of correct answer. Assume P is a fuzzy positive Datalog program, Q is an atom of our language and θ is a substitution and $x \in [0,1]$ is a real number. We say that θ, x is a correct answer to query Q wrt the program P if for all models $\mathfrak{M}$ of the program P the model assigns to $Q\theta$ a truth value $\mathfrak{M}^\bullet(Q\theta) \geq x$.

Procedural semantics of fuzzy Datalog we are using here was described in [8] and is based on the many valued modus ponens and residual conjunctors $\mathcal{C}_i$ to rule implicators $\rightarrow_i$

$$\frac{(B.b), (H \longleftarrow_i B.r)}{(H.\mathcal{C}_i(b,r))}.$$

The Datalog production operator is defined as follows: Assume P is a fuzzy definite logic program and $\mathcal{F} = \mathcal{B}^{[0,1]}$ be the complete lattice of all fuzzy Herbrand interpretations (ordered coordinate wise). Then for $\mathfrak{H} \in \mathcal{F}$ and a ground atom H the Datalog production operator is defined as follows:

$T_P(\mathfrak{H})(H) = \max\{\sup\{\mathfrak{H}^\bullet(@(B_1,\ldots,B_m)) : (H \leftarrow_L @(B_1,\ldots,B_m)) \text{ is a ground instance of a rule in the program } P\}, \sup\{c : (H.c) \text{ is a ground instance of a fact in the program } P\}\}$.

The T_P operator is continuous provided all truth functions of fuzzy aggregations in body of rules are left continuous (in the sense of functions of real numbers) in all variables.

Similarly as in the classical case fuzzy models are characterized by following:

Theorem. Assume P is a definite fuzzy logic program. A Herbrand structure $\mathfrak{H}$ is a model of P iff $T_P(\mathfrak{H}) \leq \mathfrak{H}$ (hence the minimal fixpoint of T_P is a minimal Herbrand model of P).

For corresponding fuzzy logic programming we have following

Theorem. Assume our language contains only left continuous annotations, P is a fuzzy definite logic program, θ is a substitution and $x \in [0,1]$. Then

(soundness) if θ, x is a computed answer to $? - Q$ with respect to P, then θ, x is a correct answer

(approximate completeness) if θ, x is a correct answer to $? - Q$ with respect to P, then for every $\epsilon > 0$ there is a computed answer (y, ϑ) to $? - Q$ with respect to P such that $x - \epsilon < y$ and $\vartheta = \theta\gamma$ (for some γ).

Our system captures arbitrary fuzzy similarity, not only max-min (see [9])simply extending arbitrary logic program by axioms of similarity (which have form of rules).

Fuzzy relational algebra.

Fuzzy relational algebra was developed in [12]. The main point we would like to stress here is our join parameterized by a fuzzy aggregation operator. It is defined wrt crisp equality and a fuzzy aggregation operator which tells us how to calculate the truth value degree of a tuple in the join. Assume we have relations $R_1, ..., R_n$ which evaluate predicates $r_1, ..., r_n$. Moreover assume that first k-attributes in each R_i are the same and $(b_1, \ldots, b_k, b^i_{k+1}, \ldots, b^i_{m_i}, \beta^i) \in R_i$ then $(b_1, \ldots, b_k, b^1_{k+1}, \ldots, b^1_{m_1}, \ldots, b^n_{k+1}, \ldots, b^n_{m_n}, @^\bullet(\beta^1, \ldots, \beta^n))$ is in the relation $\bowtie_@ (R_1, ..., R_n)$, that is the truth value attributes in our join do not behave as in classical join, they disappear, forwarding the respective truth values to the new aggregated truth value

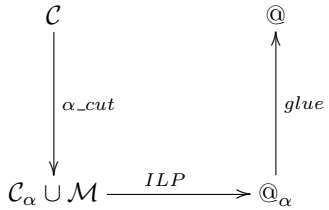
Besides this, our algebra contains also selection $\sigma_{TruthV \geq t}(r)$, similarity closure $\mathcal{SYM}_{s^r_A, A}(R)$, classical projection$\Pi_{X_1,\ldots,X_h}(R)$ and max-union.

Theorem. Every query over the fuzzy knowledge base represented by a fuzzy positive Datalog program (possibly with recursion, without negation) extended by rules describing the equational theory of predicate calculus and properties of fuzzy similarities can be arbitrarily exactly evaluated by iterating described operations of fuzzy relational algebra.

Learning the fuzzy aggregation operator.

An advantage of our approach is that we can learn the aggregation function from user's classification by a fuzzy ILP system developed in [17]. We use equivalence ([8]) of our programs to GAP ([7]) and inducing annotation terms.

We assume a set of job offers $\mathcal{J}$ with database attributes ([[10]]). Assume a user classifies (a representative sample of) job offers by $\mathcal{C} : \mathcal{J} \longrightarrow [0, 1]$. We translate this to multiple classical ILP task using α-cuts and special monotonicity axioms



Background knowledge has to be extended by monotonicity axioms $\mathcal{M}$. For each predicate touched by user preference and fuzziness, we add e.g.

$near(h, x) \leftarrow near(h, y)$ for all $x \leq y$.

Then classical ILP learning takes into account "better" examples and does not violate ordering of classification.

Some problems and future research directions.

Fuzzy tuples versus fuzzy attributes. Some fuzzy data approaches differ in the way assigning fuzzy truth value to the whole tuple or to each attribute value. In our approach we assign fuzzy truth value to the whole tuple. In our opinion it covers the fuzzy attribute case in the following sense. Many n+1-ary relations $p(A_0, \ldots, A_n)$ from practical applications can be decomposed into n-many binary relations $p_i(A_0, A_i)$. Here A_0 plays the role of a resource URI and A_i is the property, which in instances corresponds to property value (this corresponds to modeling sufficiency of the RDF data model used in the semantic web, [18]). Moreover in our applications we do not store any fuzzy data. Even values like "close" and "young" are stored as terms (ASCI strings) and only in the time of query evaluation (or query preprocessing) we interpret them and depending on the context and users understanding a fuzzy interpretation is assigned.

Uniform fuzzy small, medium, large and special indexing for efficient search. In many approaches to fuzzy data management we can see various implementations of linguistic variables (like young) and hedges (like very young). In our approach, in each domain with some natural ordering, we use only one universal fuzzy linguistic variable for small, large. Those attain value 1 only at max or min element of the domain. Reason is, that every original rule system can be rewritten using universal fuzzy sets and a new fuzzy aggregation operator. Roughly speaking, the aggregation operator can compensate all differences between users different intentions (unique for users with same profile). Further consideration are devoted to implementation of "medium". A special indexing structure B_d^+ tree is constructed to traverse along the ordering induced by fuzzy sets on domains and to find best (top-k) answers.

Extending domains by fuzzy elements and sets and selection conditions. Another problem occurring in fuzzy data modeling is extensions of domain by fuzzy values. We have made some initial acquaintance with using different models for fuzzy elements of domains (like "about_25" to be compared with other domain values) and fuzzy subsets for range queries (like "young"). We think, that calculating degree of "about_25 $\in$ young" is more appropriate than degree of "about_25 = young". Moreover pure set theoretic semantics of handling those values is not satisfactory for application. In some case we have to use approach form metric spaces (measuring distance between sets), measure theory and other Information Retrieval measures. We mention many possible definitions of fuzzy less (greater) or equal.

Future research directions. In future we plan practical experiments on a system developed at our department. In [1] authors describe a system which implements integration between a heterogeneous set of databases. One of key attribute of their solution is conflict resolution. In future we would like to experiment with some fuzzy similarities learned to fit requirements.

In [10] some initial experiments with real data were done and tests with fuzzy ILP and variants of heuristics arising from Fagin's threshold algorithm ([4]). We would like to use this to push forward the model of fuzzy data and query model.

Conclusion. In this paper we have reviewed some of our former models giving a model theoretic and fixpoint semantics for fuzzy Datalog programs with similarity and discussed some related issues and future work.

Acknowledgement. Supported by Czech IT projects 1ET100300517, 1ET100300419 and the center of excellence MS0021620838.

References

1. D. Bednárek, D. Obdržálek, J. Yaghob, F. Zavoral. Data Integration Using Data Pile Structure, In ADBIS '2005, J. Eder et al eds. Published on CEUR-WS, 2005, 178-188, ONLINE: http://CEUR-WS.org/Vol-152/
2. S. Borzsonyi, D. Kossmann, and K. Stocker. The Skyline Operator. In Proceedings of the 17th International Conference on Data Engineering, pages 421–430, Heidelberg, Germany, Apr. 2001
3. E. F. Codd. A relational model for large scale shared data bases. CACM 13 (1970) 377-387
4. R. Fagin, Combining fuzzy information from multiple systems, Proc. ACM SIGMOD, Montreal, 216-226 (1996)
5. J. Galindo, J. M. Medina, O. Pons, J. C. Cubero. A server for fuzzy SQL queries, In FQAS'98, . Andreasen et al eds. LNAI 1495, Springer Verlag 1998, 164-174
6. I. F. Ilyas, R. Shah, W. G. Aref, J. S. Vitter, A. K. Elmagarmid. Rank-aware query optimization, In Proc. 2004 ACM SIGMOD, ACM, 2004, 203 - 214
7. M. Kifer, V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications, J. Logic Programming, 12 (1992) pp 335–367
8. S. Krajči, R. Lencses, P. Vojtáš, A comparison of fuzzy and annotated logic programming, Fuzzy Sets and Systems, 144, 173-192 (2004)
9. J. Medina, M. Ojeda-Aciego, P. Vojtáš, Similarity-Based Unification: A Multi-Adjoint Approach, Fuzzy Sets and Systems, 146, 43-62 (2004)
10. P. Návrat, M. Bieliková, V. Rozinajová. Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: Proc. International Conference on Computer Systems and Technologies - CompSysTech' 2005, Varna 2005
11. D. Papadias, Y. Tao, G. Fu, B. Seeger. Progressive skyline computation in database systems ACM TODS, 2005, 41 - 82
12. J. Pokorný, P. Vojtáš. A data model for flexible querying. In Proc. ADBIS'01, A. Caplinskas and J. Eder eds. Lecture Notes in Computer Science 2151, Springer Verlag, Berlin 2001, 280-293
13. Sonalyst - fuzzy query http://fuzzy.sonayst.com/fuzzysql1.htm
14. Y. Takahashi. Fuzzy Database Query Languages and Their Relational Completeness Theorem, IEEE Transactions on Knowledge and Data Engineering archive 5 (1993) 122 - 125
15. J. D. Ullman. Principles of Database and Knowledge-Base Systems, Comp. Science Press. 1989
16. P. Vojtáš. Fuzzy logic programming, Fuzzy Sets and Systems, 124(3), 361-370 (2001)
17. P. Vojtáš, T. Horváth, S. Krajči, R. Lencses. An ILP model for a monotone graded classification problem, Kybernetika 40(3), 317-332 (2004)
18. http://www.w3c.org