Possibilistic keys

Nishita Balamuralikrishna [c], Yingnan Jiang [c], Henning Koehler [a], Uwe Leck [b], Sebastian Link [c,*], Henri Prade [d]

[a] *School of Engineering & Advanced Technology, Massey University, New Zealand*
[b] *Department of Mathematics, The University of Flensburg, Germany*
[c] *Department of Computer Science, The University of Auckland, New Zealand*
[d] *IRIT, CNRS and Université de Toulouse III, France*

Abstract

Possibility theory is applied to introduce and reason about the fundamental notion of a key for uncertain data. Uncertainty is modeled qualitatively by assigning to tuples of data a degree of possibility with which they occur in a relation, and assigning to keys a degree of certainty which says to which tuples the key applies. The associated implication problem is characterized axiomatically and algorithmically. Using extremal combinatorics, we then characterize the families of non-redundant possibilistic keys that attain maximum cardinality. In addition, we show how to compute for any given set of possibilistic keys a possibilistic Armstrong relation, that is, a possibilistic relation that satisfies every key in the given set and violates every possibilistic key not implied by the given set. We also establish an algorithm for the discovery of all possibilistic keys that are satisfied by a given possibilistic relation. It is shown that the computational complexity of computing possibilistic Armstrong relations is precisely exponential in the input, and the decision variant of the discovery problem is NP-complete as well as W[2]-complete in the size of the possibilistic key. Further applications of possibilistic keys in constraint maintenance, data cleaning, and query processing are illustrated by examples. The computation of possibilistic Armstrong relations and discovery of possibilistic keys from possibilistic relations have been implemented as prototypes. Extensive experiments with these prototypes provide insight into the size of possibilistic Armstrong relations and the time to compute them, as well as the time it takes to compute a cover of the possibilistic keys that hold on a possibilistic relation, and the time it takes to remove any redundant possibilistic keys from this cover.

Keywords: Armstrong relation; Axiomatization; Constraint maintenance; Database; Extremal combinatorics; Discovery; Implication; Key; Possibility theory; Uncertain data

* Corresponding author.
E-mail addresses: h.koehler@massey.ac.nz (H. Koehler), uwe.leck@uni-flensburg.de (U. Leck), s.link@auckland.ac.nz (S. Link), prade@irit.fr (H. Prade).

https://doi.org/10.1016/j.fss.2019.01.008

Table 1
A possibilistic relation and its nested chain of possible worlds.

Possibilistic relation

Zone	*Time*	*rfid*	*Object*	*p-Degree*
Z0	10am	H0	Grizzly	α_1
Z1	10am	H1	Grizzly	α_1
Z1	12pm	H2	Grizzly	α_1
Z3	1pm	H2	Grizzly	α_1
Z3	1pm	H3	Grizzly	α_2
Z3	3pm	H3	Grizzly	α_3
Z4	3pm	H3	Grizzly	α_4

Worlds of possibilistic relation

	zone	*time*	*rfid*	*object*
	Z0	10am	H0	Grizzly
	Z1	10am	H1	Grizzly
	Z1	12pm	H2	Grizzly
r_1	Z3	1pm	H2	Grizzly
r_2	Z3	1pm	H3	Grizzly
r_3	Z3	3pm	H3	Grizzly
r_4	Z4	3pm	H3	Grizzly

1. Introduction

Background. The notion of a key is fundamental for understanding the structure and semantics of data. For relational databases, keys were already introduced in Codd's seminal paper [8]. Here, a key is a set of attributes that holds on a relation if there are no two different tuples in the relation that have matching values on all the attributes of the key. Keys uniquely identify tuples of data, and have therefore significant applications in data processing, such as data cleaning, integration, modeling, and retrieval.

Motivation. Relational databases were developed for applications with certain data, such as accounting, inventory and payroll. Modern applications, such as information extraction, radio-frequency identification (RFID), scientific data management, data cleaning, and financial risk assessment produce large volumes of uncertain data. Probabilistic databases constitute a principled approach towards dealing with uncertainty in data. Suciu et al. emphasize that "the main use of probabilities is to record the degree of uncertainty in the data and to rank the outputs to a query; in some applications, the exact output probabilities matter less to the user than the ranking of the outputs" [61]. This suggests that a qualitative approach to uncertainty, such as the one offered by possibility theory [15], can avoid the high computational complexity in obtaining and processing probabilities, while guaranteeing the same qualitative outcome. For instance, RFID is used to track movements of endangered species of animals, such as Grizzly Bears. The scientists that gather the data keep track of which bear (identified by a unique *RFID* label) is recorded in which *zone* at which *time*. The scientists are aware that sensor readings are not exact, and the quality of the readings may affect their data analysis. The uncertainties result from various sources, including the quality of the rfid readers and tags, the movements of the animals, and the fact that borders of the zones are not clearly defined. For their analysis, they would like to distinguish between four different degrees of uncertainty and have therefore decided to assign one of four degrees of possibility (p-degree) to each record. The assignment of the p-degrees to tuples can be understood as a global function of many (unspecified) factors, or as a simple means to holistically aggregate the various levels of uncertainty associated with the tuple into one score. Table 1 shows a possibilistic relation (p-relation), where each tuple is associated with an element of a finite scale of p-degrees: $\alpha_1 > \ldots > \alpha_{k+1}$. The top degree α_1 is reserved for tuples that are 'fully possible', the bottom degree α_{k+1} for tuples that are 'impossible' to occur. Intermediate degrees, such as 'quite possible' (α_2), 'medium possible' (α_3), and 'somewhat possible' (α_4) are used whenever some linguistic interpretation is preferred.

For the data analysis, different queries may require recordings with different thresholds for data quality, as quantified by the p-degrees. For example a query may be evaluated over all recordings that have a p-degree of at least 'quite possible'. A fundamental question is then how data in recordings with a minimum p-degree can be accessed efficiently. The primary mechanism for efficient data access are keys, as index structures defined on them enable us to identify records uniquely. For our example, we observe that the p-degrees enable us to express keys with different degrees of certainty. For example, to express that it is 'somewhat possible' that the same grizzly is in different zones within an hour we declare the key {*time*, *rfid*} to be 'quite certain', stipulating that no two distinct tuples are at least 'medium possible' and have matching values on *time* and *rfid*. Similarly, to say that it is 'quite possible' that different grizzlies are in the same zone at the same time we declare the key {*zone*, *time*} to be 'somewhat certain', stipulating that no two distinct tuples are at least 'quite possible' and have matching values on *zone* and *time*. These examples

motivate the introduction and study of possibilistic keys (p-keys) as a fundamental notion for identifying records of uncertain data.

Summary. Our article provides a comprehensive study of the new concept of possibilistic keys. The goal is to establish fundamental results about core computational problems associated with possibilistic keys. These include the implication problem, extremal problems, and the problems of computing Armstrong relations for sets of possibilistic keys, and discovering all possibilistic keys that hold on a given p-relation. Solutions to these problems provide computational support to reason about possibilistic keys, acquire possibilistic keys that are meaningful in a given application domain, and estimate worst case scenarios for their maintenance. Detailed experiments complement our theoretical analysis regarding the computational complexity of these problems. Primary application areas of possibilistic keys are found in database design and data cleaning, which have been investigated in related work [35,45]. A case study from web data extraction illustrates the use of possibility degrees in data integration as well as their acquisition. In addition, we will also illustrate the applicability of possibilistic keys in constraint maintenance and query processing.

Contributions. Our contributions can be described in detail as follows.

- First, we define a semantics for possibilistic keys. Here, uncertainty is modeled qualitatively by degrees of possibility. The degrees bring forward a nested chain of possible worlds, with each being a classical relation that has some degree of possibility. Smaller worlds are more possible and satisfy at least as many keys as any of the worlds they are contained in. This "possible world" semantics is quite different from that used in the context of probabilistic databases. Our possible worlds form a linear chain with the smallest world forming a kernel that contains all tuples that are certain to occur, and adding tuples that become less possible whenever less constraints are imposed. Classical relations form the special case of possibilistic relations with only two degrees of possibility, that is, where $k = 1$. For example, the possible worlds of the p-relation from Table 1 are shown in Fig. 1. The key $\{time, rfid\}$ is satisfied by r_3 but not by r_4, and $\{zone, time\}$ is satisfied by r_1 but not by r_2.
- We have conducted a use case study in web data extraction. The case study shows how p-degrees can naturally quantify our confidence in data that is integrated from different sources, which p-keys are satisfied by such possibilistic data, and how the possibilistic data can be summarized by a sample that consists of only 2% of the tuples contained in the original p-relation but satisfies the same p-keys. The sample can be used by business analysts to consolidate which p-keys are meaningful for the application domain.
- We then establish axiomatic and linear-time algorithmic characterizations for the implication problem of p-keys. Our characterizations subsume the axiomatization and linear-time decision algorithm for the implication of classical keys as the special case where $k = 1$.
- Subsequently, we illustrate the applicability of possibilistic keys in the areas of constraint maintenance, data cleaning, and query processing. In constraint maintenance, our algorithm for deciding the implication problem can be used to remove any redundant elements from the set of p-keys whose validity are maintained under updates of possibilistic relations. In classical data cleaning, a minimal set of tuples is usually removed in order to restore the consistency of a given relation with respect to a given set of keys. For possibilistic relations, we minimally lower the degrees of possibility for tuples until consistency of a given possibilistic relation with respect to a given set of p-keys is restored. In query processing, our possibilistic framework makes it possible to rank query answers according to their associated degrees of possibility, and to utilize p-keys for optimizing a given query.
- Using extremal combinatorics, we characterize the non-redundant families of p-keys that attain maximum cardinality. This shows how large sets of possibilistic keys may become, providing insight into the worst-case complexity of modeling required. Families of non-redundant classical keys that attain maximum cardinality were characterized by Sperner's theorem, which is subsumed by our result for the special case where $k = 1$.
- Armstrong relations are data samples that exactly represent a given set of business rules. That is, they satisfy the given set and violate all business rules that are not implied by the given set. As such, Armstrong relations help business analysts with the acquisition of classical keys that are meaningful for a given application domain. A possibilistic relation is said to be Armstrong for a given set Σ of p-keys if and only if for every p-key φ, the possibilistic relation satisfies φ if and only if φ is implied by Σ. We establish structural and computational characterizations of possibilistic Armstrong relations for sets of possibilistic keys. We characterize when a given possibilistic relation is Armstrong for a given set of possibilistic keys. While the problem of finding a possibilistic Armstrong relation is precisely exponential, we establish an algorithm that computes a possibilistic Armstrong

relation whose size is guaranteed to be at most quadratic in the size of a minimum-sized possibilistic Armstrong relation. These results subsume the case of Armstrong relations for classical keys for $k = 1$.

- We then show that the complexity of deciding whether a given possibilistic relation satisfies some possibilistic key with at most n attributes is both NP-complete and $W[2]$-complete in the size of the possibilistic key. It is therefore very unlikely that tractable algorithms for the discovery of possibilistic keys can be found, even when the size of the possibilistic keys are fixed. Using hypergraph transversals, we establish an algorithm that computes a representation for the set of possibilistic keys that hold on a given possibilistic relation. Again, the special case where $k = 1$ covers the discovery of classical keys from relations. While the computation of Armstrong relations turns a set of possibilistic keys into a possibilistic relation that satisfies only those p-keys that are implied by the given set, the discovery algorithm turns a given possibilistic relation into a set of possibilistic keys that only implies those p-keys that hold on the given relation.
- The algorithms for computing possibilistic Armstrong relations and the discovery of possibilistic keys from possibilistic relations have been implemented as prototypes. These prototypes transfer our findings into actual tools that can help business analysts with the acquisition of meaningful possibilistic keys in practice.
- Finally, we have used the prototypes to conduct detailed experiments. For the computation of possibilistic Armstrong relations, we measure the time it takes to compute them as well as their size. We show cases where the size grows exponentially in the input, and other cases where the size grows logarithmically in the input. For randomly created input sets, the output size and computation time each grow low-degree polynomial in the number of given input attributes and for any fixed number of available possibility degrees, and each also grow low-degree polynomial in the number of available possibility degrees and for any fixed number of input attributes. For the discovery problem, we have applied our algorithm to six real-world data sets with varying numbers of attributes and columns after they have been made possibilistic by randomly assigning available possibility degrees to each of their tuples. We compare results for all applicable data sets under different null marker semantics, that is, under $\texttt{null} = \texttt{null}$ and under $\texttt{null} \neq \texttt{null}$. We also compare results between a sequential and a parallel implementation of our algorithms.

Organization. The remainder of this article is organized as follows. In Section 2 we compare our work to previous research, emphasizing the lack of a qualitative framework for data dependencies under uncertain data. In the same section, we also briefly summarize the work on other classes of possibilistic data dependencies. Our possibilistic data model and possibilistic keys are formally introduced in Section 3. The use case study about web data extraction is presented in Section 4. Axiomatic and algorithmic characterizations of the associated implication problem are established in Section 5. In Section 6, applications of possibilistic keys in constraint maintenance, data cleaning, and query processing are illustrated by examples. Combinatorial results on the maximum cardinality of non-redundant families of possibilistic keys are established in Section 7. Possibilistic Armstrong relations are investigated in Section 8, where we also study the problem of discovering the set of possibilistic keys from a given possibilistic relation. The prototype systems for the computation of Armstrong relations and the discovery problem are briefly described in Section 9. Section 10 presents and discussed our experimental results related to the computation of Armstrong relations and the discovery problem. Finally, Section 11 concludes and outlines future work.

2. Related work

This section will elaborate on the wider context of work on uncertain data. We will start with the general trade-offs between the use of probabilistic and possibilistic approaches, and highlight some recent work on constraints over probabilistic databases. Subsequently, we will identify database design and data cleaning as the main target areas of our possibilistic model in contrast to other possibilistic data models that target database queries. We will then comment on recent related work on other classes of database constraints that are founded on the possibilistic model we use. We will say how our contributions to possibilistic keys generalize previous findings for the special case of keys over certain relations. Finally, we will detail how our current submission extends our conference article on possibilistic keys.

2.1. Probabilistic databases

Probabilistic databases have received much interest [61] due to the need to deal with uncertain data. While the many types of uncertainty information make it challenging for uncertain data to become first-class citizens in commercial

database systems, the benefits and applications of uncertain data are broad and deep. For example, the METIS system is "an industrial prototype system for supporting real-time, actionable maritime situational awareness". It uses web harvesting, information extraction, and data integration to collect substantial data on ships, their routes, and actions. The system represents the uncertainty in this data explicitly, and utilizes the information for reasoning purposes with ProbLog. When coast guards notice ships in their waters, they ask queries of the following type to METIS: "What is the probability of smuggling and how reliable is this assessment?". Given there are more than 300,000 ships in the world and each of them creates a lot of data, this is quite an impressive achievement, and a good motivator to conduct research on uncertain data.

Constraints present a key challenge here: "When the data is uncertain, constraints can be used to increase the quality of the data, and hence they are an important tool in managing data with uncertainties" [9]. Suciu et al. emphasize that "the main use of probabilities is to record the degree of uncertainty in the data and to rank the outputs to a query; in some applications, the exact output probabilities matter less to the user than the ranking of the outputs" [61]. This suggests that a qualitative approach to uncertainty, such as possibility theory [15], can avoid the high computational complexity in obtaining and processing probabilities, while guaranteeing the same qualitative outcome.

Research on probabilistic databases has naturally focused on query processing, e.g. [3,30]. Keys and cardinality constraints on probabilistic databases have been studied in [7,6,29,56–58]. Here, [7,6,56–58] study similar problems for probabilistic keys and cardinality constraints that we investigate in this article for possibilistic keys. Probabilistic and possibilistic keys complement one another and are incomparable: Probabilistic keys are based on underlying probability distributions, and possibilistic keys are based on underlying possibility distributions. The use of either probabilistic or possibilistic keys therefore depends on what kind of uncertainty information is available and/or required. The paper [29] focuses on the use of probabilistic keys for query optimization.

2.2. Possibilistic data models and their target use cases

Similar to probabilistic databases, possibilistic approaches to uncertain data have mainly dealt with query languages. Queries are not the main subject of this article, so we refer the reader to a survey on fuzzy approaches to database querying [66]. Nevertheless, we will now comment on various possibilistic data models, their expressiveness, and their main target areas. From the least to the most expressive, we can at least distinguish four possibilistic models for uncertain data [55]:

- databases with layered tuples
- tuples involving certainty-qualified attribute values
- tuples involving attribute values restricted by possibility distributions, and
- possibilistic c-tables.

Layered tuples. This is the data model we use here. The idea is just to provide a total ordering of the tuples in the database according to the confidence we have in their truth. This is encoded by assigning a possibility degree with each tuple. The result is a layered database: all the tuples having the same degree are in the same layer (and only them). The tuples of highest possibility degree are also associated with the highest certainty degree, while tuples with a smaller possibility degree are not certain at all. This means that any possible world contains all the tuples with the highest possibility degree, while the other tuples may or may not be present in a particular possible world, see [44] for details. Our model does not provide information about the uncertainty of specific attribute values. Consequently, enforcing the p-degrees on tuples in situations where the degree of uncertainty is known for some attribute values and can be quantified, we would loose some information. On the other hand, there may well be situations where the sources of uncertainty are unclear, cannot be quantified adequately, or where this is not even desired. In this model, the possibility degree of a tuple can be understood as a global function of many (unspecified) factors, or as a simple means to holistically assess the degree of possibility we associate with a tuple. It is the quantity that results from the combination of the uncertainty information present in the given set of attributes. A different way of interpretation, which is closer to the quantification of p-degrees for a value of an attribute, is to take the minimum p-degree associated with the values on the given attributes. This is a trade-off: if specific information (such as the p-degree of an attribute value) is available, we may not be able to express it in less expressive models, but if specific information is unavailable, then we can still express it in less expressive models. The trade-off can be viewed as a

generalization of the situation that is already apparent in databases with null markers: SQL permits only one type of null marker, so we lose information whenever we know more about the kind of missing value (e.g. the value exists but is unknown, or the value does not exist). The possibilistic grounding of this model was first presented in [44] where the duality between p-degrees and c-degrees is pointed out and exploited, but the model has its roots in an early proposal dealing with weighted tuple databases [31]. The possibilistic model has gained recent momentum in a series of articles that highlight its applications in data modeling [21,38], database design [45], and data cleaning [35]. In [44], a new class of possibilistic functional dependencies was introduced, and the equivalence of their associated implication problem to that of Horn clauses in possibilistic logic was proven. The main target area of this new class of possibilistic functional dependencies was database schema design, since different degrees of data redundancy can be introduced and different normal forms guarantee that no redundant data values of these degrees can occur in any instances of the schemata that meet the normal form condition. This normalization framework for uncertain data has been reported in [45]. Possibilistic keys form the important special case of possibilistic functional dependencies that represent the sub-class of functional dependencies which do not case any form of data redundancy. Possibilistic keys were the first class of database constraints introduced for the underlying possibilistic model [33]. The current article is an extended version of [33]. Another important area of application for any form of constraints over this possibilistic model is data cleaning. Here, the details can be found in [35]. Basically, the p-degrees of tuples provide a new view of cleaning dirty data: Instead of viewing the data itself as dirty, we view the p-degrees associated with the data as dirty. The c-degrees of constraints then provide guidelines for the cleaning of the p-degrees in the sense that the p-degrees must be lowered minimally to restore consistency with respect to the given c-degrees of constraints. In [21,38], the possibilistic model was used to introduce a new class of possibilistic cardinality constraints. While keys stipulate that there cannot be two different tuples with matching values on all the attributes of the key, cardinality constraints stipulate that there cannot be $b+1$ different tuples with matching values on all the attributes of the cardinality constraint. Keys therefore form the important special case of cardinality constraints where $b=1$.

Certainty-qualified attribute values. In this model [53], attribute values (or disjunctions thereof) are associated with a c-degree (which is the lower bound of the value of a necessity function). This amounts to associating each attribute value with a simplified type of possibility distribution restricting it. Different attributes in a tuple may have different c-degrees associated with their respective values. This model has some advantages with respect to querying: i) it constitutes a strong representation system for the whole relational algebra, ii) it does not require the use of any lineage mechanism and the query complexity is close to the classical case, and iii) the approach seems more robust with respect to small changes in the value of degrees than a probabilistic handling of uncertainty.

Attribute values restricted by general possibility distributions. In this "full possibilistic model" [5], any attribute value can be represented by a possibility distribution. Moreover, representing the result of some relational operations (in particular the join) in this model requires the expression of dependencies between candidate values of different attributes in the same tuple, which leads to the use of nested relations. In [5], it is shown that this model is a strong representation system for selection, projection and foreign-key join only. The handling of the other relational operations requires the use of a lineage mechanism as in the probabilistic approaches. This model makes it possible to compute not only the more or less certain answers to a query (as in the previous model), but also the answers which are only possible to some extent.

Possibilistic c-tables. This model is outlined in [54]. The possibilistic extension of c-tables preserves all the advantages of classical c-tables (for expressing constraints linking attribute values) while the attribute values are restricted by any kind of possibility distribution. This model generalizes the two previous ones. In fact, possibilistic c-tables, similar to probabilistic c-tables, can be encompassed in the general setting of the semiring framework proposed by Tannen et al. [20].

2.3. Extending results from keys over certain relations

The present article extends the conference version [33] in various directions. Firstly, a new section on extremal combinatorics for possibilistic keys has been added. Here, we characterize which families of non-redundant possibilistic keys attain maximum cardinality. The result can be used by data engineers to simulate scenarios where the largest potential number of possibilistic keys must be maintained, and can therefore provide estimates into the worst-case complexity of managing them. Secondly, we prove a new result that the discovery of possibilistic keys is both NP-complete and $W[2]$-complete in the size of the possibilistic keys. Thirdly, we implemented our algorithms for

computing possibilistic Armstrong relations as well as for computing a cover for the set of possibilistic keys that hold on a given possibilistic relation. The graphical user interfaces for these prototype systems are briefly summarized. Fourthly, we used the prototype systems to conduct extensive experiments regarding the time it takes to compute the outputs of both algorithms as well as the size of these outputs. The findings are also discussed in a new section. Fifthly, we have provided full proofs for all our results. Sixthly, we have included a new result on the equivalence of the finite and unrestricted implication problems for p-keys. Seventhly, we have also included the new application area of constraint maintenance in the section on applications for p-keys. Eighthly, we have included a new section on a use case for our possibilistic model and p-keys from web data integration. Finally, the presentation of our findings in [33] has been substantially revised to add more algorithms, explanations, illustrations, and examples.

Our contributions extend results on keys from classical relations [43], covered by the special case of two possibility degrees where $k = 1$. These include results on the implication problem [1,11], Armstrong relations [2,18,24,47,64] and the discovery of keys from relations [4,28,40,48,51,52], as well as extremal combinatorics [10,23,34,59]. Keys have also been considered in other data models, including incomplete relations [22,34,39,37,36,25,62,65] and XML data [26,27]. Note that Armstrong relations are also an AI tool to acquire and reason about conditional independencies [19,50].

3. Possibilistic keys

In this section we extend the classical relational model of data to model uncertain data qualitatively. Based on the data model, we then introduce the notion of a possibilistic key, and illustrate it by examples.

A relation schema, denoted by R, is a finite non-empty set of *attributes*. Each attribute $a \in R$ has a *domain* $dom(a)$ of values. A *tuple* t over R is an element of the Cartesian product $\prod_{a \in R} dom(a)$ of the attributes' domains. For $X \subseteq R$ we denote by $t(X)$ the *projection* of t on X. A *relation* over R is a finite set r of tuples over R. As example we use the relation schema TRACKING with attributes *zone*, *time*, *rfid*, *object* from before. Tuples either belong or do not belong to a relation. For example, we cannot express that we have less confidence for the Grizzly identified by *rfid* value $H3$ to be in *zone* $Z3$ at $1\,pm$ than for the Grizzly identified by $H2$.

We model uncertain relations by assigning to each tuple some degree of possibility with which the tuple occurs in a relation. Formally, we have a *scale of possibility*, that is, a finite strict linear order $\mathcal{S} = (S, <)$ with $k+1$ elements, denoted by $\alpha_1 > \cdots > \alpha_k > \alpha_{k+1}$. The elements $\alpha_i \in S$ are called *possibility degrees*, or p-degrees. The top p-degree α_1 is reserved for tuples that are 'fully possible' to occur in a relation, while the bottom p-degree α_{k+1} is reserved for tuples that are 'impossible' to occur. Humans like to use simple scales in everyday life to communicate, compare, or rank. Simple means to classify items qualitatively, rather than quantitatively by putting a precise value on it. Classical relations use two p-degrees, that is $k = 1$.

A *possibilistic relation schema* $(R, \mathcal{S})$, or p-relation schema, consists of a relation schema R and a possibility scale $\mathcal{S}$. A *possibilistic relation*, or p-relation, over $(R, \mathcal{S})$ consists of a relation r over R, and a function $Poss_r$ that assigns to each tuple $t \in r$ a p-degree $Poss_r(t) \in \mathcal{S}$. Table 1 shows a p-relation over (TRACKING, $\mathcal{S} = \{\alpha_1, \ldots, \alpha_5\}$).

P-relations enjoy a possible world semantics. For $i = 1, \ldots, k$ let r_i consist of all tuples in r that have p-degree at least α_i, that is, $r_i = \{t \in r \mid Poss_r(t) \geq \alpha_i\}$. Indeed, we have $r_1 \subseteq r_2 \subseteq \cdots \subseteq r_k$. The possibility distribution π_r for this linear chain of possible worlds is defined by $\pi_r(r_i) = \alpha_i$. Note that r_{k+1} is not a possible world, since its possibility $\pi(r_{k+1}) = \alpha_{k+1}$ means 'impossible'. Vice versa, the possibility $Poss_r(t)$ of a tuple $t \in r$ is the maximum possibility $\max\{\alpha_i \mid t \in r_i\}$ of a world to which t belongs. If $t \notin r_k$, then $Poss_r(t) = \alpha_{k+1}$. Every tuple that is 'fully possible' occurs in every possible world, and is therefore also 'fully certain'. Hence, relations are a special case of uncertain relations. Fig. 1 shows the possible worlds $r_1 \subsetneq r_2 \subsetneq r_3 \subsetneq r_4$ of the p-relation of Table 1.

We introduce possibilistic keys, or p-keys, as keys with some degree of certainty. As keys are fundamental to applications with certain data, p-keys will serve a similar role for application with uncertain data. A *key* $K \subseteq R$ is satisfied by a relation r over R, denoted by $\models_r K$, if there are no distinct tuples $t, t' \in r$ with matching values on all the attributes in K. For example, the key {*time, object*} is not satisfied by any relation $r_1, \ldots, r_4$. The key {*zone, time*} is satisfied by r_1, but not by r_2. The key {*zone, rfid*} is satisfied by r_2, but not by r_3. The key {*time, rfid*} is satisfied by r_3, but not by r_4. The key {*zone, time, rfid*} is satisfied by r_4.

The p-degrees of tuples result in degrees of certainty with which keys hold. Since $K_1 = \{zone, time, rfid\}$ holds in every possible world, it is fully certain to hold on r. As $K_2 = \{time, rfid\}$ is only violated in a somewhat possible

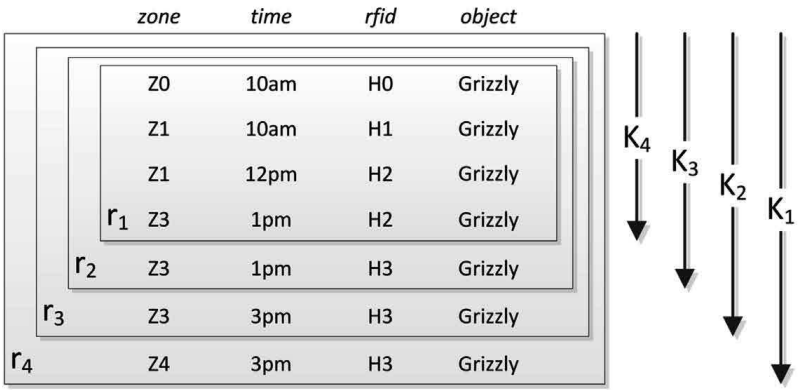


Fig. 1. Worlds of possibilistic relation and scope of keys.

world r_4, it is quite certain to hold on r. Since the smallest relation that violates $K_3 = \{zone, rfid\}$ is the medium possible world r_3, it is medium certain to hold on r. As the smallest relation that violates $K_4 = \{zone, time\}$ is the quite possible world r_2, it is somewhat certain to hold on r. Since $\{time, object\}$ is violated in the fully possible world r_1, it is not certain at all to hold on r. The scope of these keys, that is the largest possible world on which they hold, is illustrated in Fig. 1.

Similar to a scale $\mathcal{S}$ of p-degrees for tuples we use a scale $\mathcal{S}^T$ of certainty degrees, or c-degrees, for keys. We use subscripted versions of the Greek letter β to denote c-degrees. Formally, the correspondence between p-degrees in $\mathcal{S}$ and the c-degrees in $\mathcal{S}^T$ can be defined by the mapping $\alpha_i \mapsto \beta_{k+2-i}$ for $i = 1, \ldots, k+1$. Hence, the *marginal certainty* $C_r(K)$ with which the key K holds on the uncertain relation r is either the top degree β_1 if K is satisfied by r_k, or the minimum amongst the c-degrees β_{k+2-i} that correspond to possible worlds r_i in which K is violated, that is,

$$C_r(K) = \begin{cases} \beta_1 & \text{, if } r_k \text{ satisfies } K \\ \min\{\beta_{k+2-i} \mid \not\models_{r_i} K\} & \text{, otherwise} \end{cases} .$$

For the p-relation r from Table 1, we can observe from Fig. 1 that the marginal certainty of $K_1 = \{zone, time, rfid\}$ is $C_r(K_1) = \beta_1$, the marginal certainty of $K_2 = \{time, rfid\}$ is $C_r(K_2) = \beta_2$, the marginal certainty of $K_3 = \{zone, rfid\}$ is $C_r(K_3) = \beta_3$, and the marginal certainty of $K_4 = \{zone, time\}$ is $C_r(K_4) = \beta_4$.

In the same way keys can ensure the integrity of entities in certain relations, possibilistic keys aim at ensuring the integrity of entities in uncertain relations. More precisely, p-keys aim at ensuring the integrity of entities in some of the possible worlds of uncertain relations. For this purpose, we empower p-keys to stipulate that the marginal certainty by which they hold in some uncertain relation must meet a given minimum threshold. This aim is formalized by the following definition.

Definition 1. Let $(R, \mathcal{S})$ denote a p-relation schema. A possibilistic key (p-key) over $(R, \mathcal{S})$ is an expression (K, β) where $K \subseteq R$ and $\beta \in \mathcal{S}^T$. A p-relation $(r, Poss_r)$ over $(R, \mathcal{S})$ satisfies the p-key (K, β) if and only if $C_r(K) \geq \beta$.

In other words, a p-key (K, β) is violated by a p-relation r if and only if the marginal certainty $C_r(K)$ by which the key K holds in r is lower than the minimum c-degree β that has been specified. We illustrate the notion of a p-key on our running example.

Example 2. The p-relation from Table 1 satisfies the p-key set Σ consisting of

- $(\{zone, time, rfid\}, \beta_1)$,
- $(\{time, rfid\}, \beta_2)$,
- $(\{zone, rfid\}, \beta_3)$, and
- $(\{zone, time\}, \beta_4)$.

It violates the p-key $(\{zone, rfid\}, \beta_2)$ since $C_r(\{zone, rfid\}) = \beta_3 < \beta_2$.

Table 2
An informative Armstrong p-relation for the book p-relation.

Title	*Author*	*Pages*	*Publisher*	*Price*	p-Degree
Harry Potter and the Cursed Child	J. K. Rowling	352	Little Brown	19.19	α_1
Harry Potter and the Philosopher's Stone	J. K. Rowling	256	Bloomsbury	37.32	α_1
Harry Potter Box Set	J. K. Rowling	3422	Bloomsbury	76.04	α_2
Mockingjay	Suzanne Collins	448	Scholastic	8.92	α_2
Catching Fire	Suzanne Collins	448	Scholastic	8.92	α_3
End of Watch	Stephen King	448	Scribner	24.65	α_3
End of Watch	Stephen King	496	Pocket Books	9.2	α_5

4. A real-world example from web data integration

In this section we illustrate the application of our framework on a real-world example from Web data integration. Here, basic information about the top-100 best-selling books was extracted from five online book sellers:

1. eBay book store[1]
2. Amazon book store[2]
3. book directory[3]
4. Barnes & Noble[4]
5. Easons.[5]

For our collection the tool Web Scraper[6] was used on a Chrome extension plugin. It extracted the content from the same HTML tags from every book web page. The extracted results had many mistakes initially, since not all books have the same details and the same HTML tags may also have different values. Errors were fixed manually. In total, 344 tuples were extracted over the p-schema $(\textsc{Books}, \{\alpha_1, \ldots, \alpha_5, \alpha_6\})$ with $\textsc{Books} = \{\mathit{title}, \mathit{author}, \mathit{pages}, \mathit{price}, \mathit{publisher}\}$. For $i = 1, \ldots, 6$, a tuple was assigned p-degree α_i if and only if it was contained in $6 - i$ of the data sets. The p-degree therefore denotes the possibility by which a book is listed in the top-100 of each book seller. From the resulting p-relation we then applied Algorithm 4 to discover the following set Σ of p-keys that are satisfied by it.

- $(\{\mathit{pages}, \mathit{title}\}, \beta_1)$, $(\{\mathit{price}, \mathit{title}\}, \beta_1)$, $(\{\mathit{publisher}, \mathit{title}\}, \beta_1)$,
- $(\{\mathit{title}\}, \beta_2)$,
- $(\{\mathit{pages}\}, \beta_4)$, $(\{\mathit{author}, \mathit{price}\}, \beta_4)$, $(\{\mathit{price}, \mathit{publisher}\}, \beta_4)$, and
- $(\{\mathit{publisher}\}, \beta_5)$.

These p-keys may now be used for the optimization of any queries on the given data set. Note that this is true whether the discovered p-key is meaningful for the given application domain, or not.

With the help of Algorithm 3 we then computed an Armstrong p-relation for the set Σ, and then looked manually for tuples of the original p-relation that had same agree sets as those in the Armstrong p-relation. The benefit is that the resulting Armstrong p-relation consists of real-world tuples and is a subset of the original p-relation. Such Armstrong p-relations are known as informative Armstrong databases [49,64]. Informative Armstrong databases may be understood as semantic samples of the original database, since they represent the same set of constraints. Their big advantage is that they are easier to understand by humans because of their smaller size. In our example, the resulting informative Armstrong p-relation is shown in Table 2.

[1] http://www.half.ebay.com/books-bestsellers.
[2] https://www.amazon.com/best-sellers-books-Amazon/zgbs/books.
[3] https://www.bookdepository.com/bestsellers.
[4] http://www.barnesandnoble.com/b/books/_/N-1fZ29Z8q8.
[5] http://www.easons.com/shop/c-bestselling-books.
[6] http://webscraper.io/.

It consists of seven tuples, which is 2% of the number of tuples in the original p-relation. The sample may serve different purposes. It can be understood as a semantic data profile of the given data set, and thus be used for testing purposes, such as queries or updates. This provides data profiling with a different, more user-friendly, perspective of the result set. Another application is business rule acquisition and data cleaning. The aim of business rule acquisition is to identify those constraints which are meaningful in a given application domain. This is usually done with the help of domain experts, who have commonly no knowledge of database constraints but can readily assess samples and provide feedback. Given the sample above, for instance, a domain expert may spot that different books with the same author and title occur in the list. If the original intention of the integration process was to not list multiple books with the same author and title, then the sample would immediately alert domain experts to a violation of this constraint. The feedback to the business analysts would then result in the specification of the p-key $(\{author, title\}, \beta_1)$. In turn, such p-keys may guide data repairs.

5. Reasoning tools

The primary purpose of any class of data dependencies is the enforcement of data integrity. In the case of keys, data integrity refers to entity integrity. Enforcing a set of keys on a relation means that any updates to the relation should only be permitted when they result in a new relation that still satisfies all the keys in the given set. Validating that the new relation does satisfy all the keys in the given set consumes time, which should be minimized in order to progress data processing. As such, it is important that the given set of keys does not contain any *redundant* keys. Here, a key σ is redundant in a given set Σ if and only if σ is implied by $\Sigma - \{\sigma\}$. Recall that a constraint σ is *implied* by a constraint set Σ, written $\Sigma \models \varphi$, if and only if every relation that satisfies all the constraints in Σ also satisfies σ. In other words, if a key σ is redundant in a given set Σ, then knowing that all the keys in $\Sigma - \{\sigma\}$ are satisfied by the new relation also guarantees us that σ is satisfied by the new relation: We do not need to validate that σ is satisfied by the new relation. This means that the following *implication problem* is fundamental for the class of p-keys.

PROBLEM:	Implication problem
INPUT:	A set $\Sigma \cup \{\varphi\}$ of p-keys over p-relation schema $(R, \mathcal{S})$
OUTPUT:	Yes, if $\Sigma \models \varphi$ No, otherwise

It is the aim of this section to establish axiomatic and algorithmic solutions to the implication problem of p-keys. The solutions capture those for the notion of a classical key by the special case where the number of available p-degree is two, that is, where $k = 1$. We will first introduce some more terminology and show that, for the class of p-keys, finite and unrestricted implication problems coincide. Subsequently, we will establish a strong correspondence between the implication problem for p-keys and that for classical keys. We will then utilize this correspondence to develop the axiomatic and algorithmic solutions to the implication problem.

5.1. Finite and unrestricted implication problems

So far, we have defined relations to be finite set of tuples. In theory, it would also be possible to allow infinite set of tuples as p-relations. This results in two different notions of implication, which we define now. Let $\Sigma \cup \{\varphi\}$ denote a set of p-keys over $(R, \mathcal{S})$. We say Σ *(finitely) implies* φ, denoted by $\Sigma \models_{(f)} \varphi$, if every (finite) p-relation $(r, Poss_r)$ over $(R, \mathcal{S})$ that satisfies every p-key in Σ also satisfies φ. Consequently, for the unrestricted implication problem we consider finite and infinite relations, while for the finite implication problem we restrict ourselves to finite relations only. We use $\Sigma^*_{(f)} = \{\varphi \mid \Sigma \models_{(f)} \varphi\}$ to denote the *(finite) semantic closure* of Σ. We will show now that finite and unrestricted implication problem coincide for the class p-keys. That is, for any given set $\Sigma \cup \{\varphi\}$ of p-keys on any given p-relation schema $(R, \mathcal{S})$ it holds that $\Sigma \models \varphi$ if and only if $\Sigma \models_{(f)} \varphi$. In fact, we can show something stronger: We show that the unrestricted implication problem coincides with the *implication problem in two-tuple relations*, which we denote by $\models_2$. The latter problem is to decide for any given set $\Sigma \cup \{\varphi\}$ of p-keys on any given p-relation schema $(R, \mathcal{S})$, whether $\Sigma \models_2 \varphi$ holds, that is, whether every two-tuple p-relation over $(R, \mathcal{S})$ that satisfies all the p-keys in Σ also satisfies φ. We will now show that all three implication problems coincide for the class of p-keys.

Theorem 1. *Let $\Sigma \cup \{\varphi\}$ denote a set of p-keys over p-relation schema $(R, \mathcal{S})$. Then the following statements are equivalent:*

1. $\Sigma \models \varphi$,
2. $\Sigma \models_f \varphi$, *and*
3. $\Sigma \models_2 \varphi$.

Proof. Since every two-tuple p-relation over $(R, \mathcal{S})$ is a finite p-relation, the following is straightforward: if $\Sigma \models \varphi$, then $\Sigma \models_f \varphi$, and if $\Sigma \models_f \varphi$, then $\Sigma \models_2 \varphi$. It remains to establish the opposite directions. For that purpose, it suffices to show that if $\Sigma \models_2 \varphi$, then $\Sigma \models \varphi$. We show the contraposition: if $\Sigma \models \varphi$ does not hold, then $\Sigma \models_2 \varphi$ does also not hold.

Suppose $\Sigma \models \varphi$ does not hold. Then there is some (possibly infinite) p-relation $(r, Poss_r)$ over $(R, \mathcal{S})$ such that $(r, Poss_r)$ satisfies Σ, but $(r, Poss_r)$ violates $\varphi = (K, \beta_i)$. Since $(r, Poss_r)$ violates φ, there is a smallest possible world r_j that violates K and where $C_r(K) = \beta_{k+2-j} < \beta$. Since r_j violates K, there must be two tuples $t_1, t_2 \in r_j$ such that $t_1(K) = t_2(K)$. Let $(r', Poss_{r'})$ be the two-tuple p-relation over $(R, \mathcal{S})$ where $r' = \{t_1, t_2\}$ and $Poss_{r'}(t_1) = Poss_r(t_1)$ and $Poss_{r'}(t_2) = Poss_r(t_2)$. It follows immediately that for every $(K', \beta') \in \Sigma$, $C_{r'}(K') \geq C_r(K') \geq \beta'$ holds. That is, $(r', Poss_{r'})$ satisfies all p-keys in Σ. Since $(r', Poss_{r'})$ violates $\varphi = (K, \varphi)$ it follows that $\Sigma \models_2 \varphi$ does not hold. This completes the proof. □

Theorem 1 allows us to speak about *the* implication problem for the class of p-keys. The following example shows an instance of the implication problem on our running example.

Example 3. Let Σ be as in Example 2, and $\varphi = (\{zone, rfid, object\}, \beta_2)$. Then Σ does not imply φ as the following p-relation witnesses:

zone	*time*	*rfid*	*object*	p-degree
Z0	10am	H0	Grizzly	α_1
Z0	3pm	H0	Grizzly	α_3

Notice that p-relation is a two-tuple relation.

5.2. The magic of β-cuts

For our axiomatic and algorithmic solutions to the implication problem for p-keys we will establish a strong correspondence to the implication problem for classical keys. This is possible because of the *sub-model property* of p-keys: Whenever a relation satisfies a key, then every sub-relation of the relation will also satisfy the key. A fundamental notion is that of a β-cut, which we define now.

Definition 4. Let Σ denote a set of p-keys over p-relation schema $(R, \mathcal{S})$ with $|\mathcal{S}| = k + 1$, and let $\beta \in \mathcal{S}^T$ denote a c-degree where $\beta > \beta_{k+1}$. Then $\Sigma_\beta = \{K \mid (K, \beta') \in \Sigma \text{ and } \beta' \geq \beta\}$ denotes the set of keys for which a p-key exists in Σ whose c-degree is at least β. The key set Σ_β is called the *β-cut* of the p-key set Σ.

We illustrate the definition of β-cuts on our running example.

Example 5. For the p-key set $\Sigma = \{(K_1, \beta_1), (K_2, \beta_2), (K_3, \beta_3), (K_4, \beta_4)\}$ from Example 2, we obtain for $i = 1, \ldots, 4$ the following β-cuts $\Sigma_{\beta_i} = \bigcup_{j=1}^{i} \{K_j\}$.

The fundamental property of β-cuts is given by the following result.

Theorem 2. *Let $\Sigma \cup \{(K, \beta)\}$ be a p-key set over $(R, \mathcal{S})$ where $\beta > \beta_{k+1}$. Then $\Sigma \models (K, \beta)$ if and only if $\Sigma_\beta \models K$.*

Proof. Suppose $(r, Poss_r)$ is some p-relation over $(R, \mathcal{S})$ that satisfies Σ, but violates (K, β). In particular, $C_r(K) < \beta$ implies that there is some relation r_i that violates K and where $\beta_{k+2-i} < \beta$. Let $K' \in \Sigma_\beta$, where $(K', \beta') \in \Sigma$. Since

Table 3
Axiomatization $\mathfrak{K}' = \{\mathcal{T}', \mathcal{S}'\}$ of keys and $\mathfrak{K} = \{\mathcal{T}, \mathcal{S}, \mathcal{B}, \mathcal{W}\}$ of possibilistic keys.

$\frac{}{R}$ (top, $\mathcal{T}'$)	$\frac{}{(R, \beta)}$ (top, $\mathcal{T}$)	$\frac{}{(K, \beta_{k+1})}$ (bottom, $\mathcal{B}$)
$\frac{K}{K \cup K'}$ (superkey, $\mathcal{S}'$)	$\frac{(K, \beta)}{(K \cup K', \beta)}$ (superkey, $\mathcal{S}$)	$\frac{(K, \beta)}{(K, \beta')}\, \beta' \leq \beta$ (weakening, $\mathcal{W}$)

r satisfies $(K, \beta') \in \Sigma$ we have $C_r(K') \geq \beta' \geq \beta$. If r_i violated K', then $\beta > \beta_{k+2-i} \geq C_r(K') \geq \beta$, a contradiction. Hence, r_i satisfies Σ_β and violates K.

Let r' denote some relation that satisfies Σ_β and violates K, w.l.o.g. $r' = \{t, t'\}$. Let r be the p-relation over $(R, \mathcal{S})$ that consists of r' and where $Poss_{r'}(t) = \alpha_1$ and $Poss_{r'}(t') = \alpha_i$, such that $\beta_{k+1-i} = \beta$. Then r violates (K, β) since $C_r(K) = \beta_{k+2-i}$, as $r_i = r'$ is the smallest relation that violates K, and $\beta_{k+2-i} < \beta_{k+1-i} = \beta$. For $(K', \beta') \in \Sigma$ we distinguish two cases. If r_i satisfies K', then $C_r(K') = \beta_1 \geq \beta$. If r_i violates K', then $K' \notin \Sigma_\beta$, i.e., $\beta' < \beta = \beta_{k+1-i}$. Therefore, $\beta' \leq \beta_{k+2-i} = C_r(K')$ as $r_i = r'$ is the smallest relation that violates K'. We conclude that $C_r(K') \geq \beta'$. Consequently, $(r, Poss_r)$ is a p-relation that satisfies Σ and violates (K, β). □

The following example illustrates an instance of the correspondence that Theorem 2 has established.

Example 6. Let $\Sigma \cup \{\varphi\}$ be as in Example 3. Theorem 2 says that Σ_{β_2} does not imply $(\{zone, rfid, object\}, \beta_2)$. The possible world r_3 of the p-relation from Example 3:

zone	*time*	*rfid*	*object*
Z0	10am	H0	Grizzly
Z0	3pm	H0	Grizzly

satisfies the key {*time, rfid*} that implies both keys in Σ_{β_2}. However, r_3 violates the key $\{zone, rfid, object\}$.

In the following subsections and sections we will make extensive use of Theorem 2.

5.3. Axiomatic characterization

We determine the semantic closure by applying *inference rules* of the form

$$\frac{\text{premise}}{\text{conclusion}}\,\text{condition}\,.$$

For a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\sigma_1, \ldots, \sigma_n$ such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or is the conclusion that results from an application of an inference rule in $\mathfrak{R}$ to some premises in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. Let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is *sound* (*complete*) if for every set Σ over every $(R, \mathcal{S})$ we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

For the set $\mathfrak{K}$ from Table 3 the attribute sets K, K' are subsets of a given R, and β, β' belong to a given $\mathcal{S}^T$. In particular, β_{k+1} denotes the bottom certainty degree.

Theorem 3. *The set $\mathfrak{K}$ forms a finite axiomatization for the implication problem of p-keys.*

Proof. We prove soundness first. For the soundness of the top-rule $\mathcal{T}$ we observe that every possible world is a relation, which means there cannot be any possible world which contains two different tuples that have matching values on all the attributes of the underlying relation schema R. In other words, R is always guaranteed to be a

p-key that holds with c-degree β_1 and thus any c-degree. For the soundness of the bottom-rule $\mathcal{B}$ we observe that the marginal c-degree of any key in any p-relation is at least β_{k+1}. For the soundness of the superkey-rule $\mathcal{S}$, let r be a p-relation that satisfies (K, β). Then every possible world of r that satisfies the key K will also satisfy the superkey $K \cup K'$. Consequently, the marginal certainty of $K \cup K'$ in r is at least as high as the marginal certainty of K in r. Consequently, r will also satisfy $(K \cup K', \beta)$. For the soundness of the weakening-rule $\mathcal{W}$ let r be a p-relation that satisfies (K, β). That is, the marginal certainty $C_r(K)$ of K in r is at least β. Consequently, the marginal certainty $C_r(K)$ of K in r is also at least β' for every β' such that $\beta' \leq \beta$. This shows the soundness of the inference rules.

For completeness, we apply Theorem 2 and the fact that $\mathcal{K}'$ axiomatizes key implication. Let $(R, \mathcal{S})$ be a p-relation schema with $|\mathcal{S}| = k + 1$, and $\Sigma \cup \{(K, \beta)\}$ a p-key set such that $\Sigma \models (K, \beta)$. We show that $\Sigma \vdash_{\mathfrak{K}} (K, \beta)$ holds.

For $\Sigma \models (K, \beta_{k+1})$ we have $\Sigma \vdash_{\mathfrak{K}} (K, \beta_{k+1})$ by applying $\mathcal{B}$. Let now $\beta < \beta_{k+1}$. From $\Sigma \models (K, \beta)$ we conclude $\Sigma_\beta \models K$ by Theorem 2. Since $\mathfrak{K}'$ is complete for key implication, $\Sigma_\beta \vdash_{\mathfrak{K}'} K$ holds. Let $\Sigma_\beta^\beta = \{(K', \beta) \mid K' \in \Sigma_\beta\}$. Thus, the inference of K from Σ_β using $\mathcal{K}'$ can be turned into an inference of (K, β) from Σ_β^β by $\mathfrak{K}$, simply by adding β to each key in the inference. Hence, whenever $\mathcal{T}'$ or $\mathcal{S}'$ is applied, one applies instead $\mathcal{T}$ or $\mathcal{S}$, respectively. Consequently, $\Sigma_\beta^\beta \vdash_{\mathfrak{K}} (K, \beta)$. The definition of Σ_β^β ensures that every p-key in Σ_β^β can be inferred from Σ by applying $\mathcal{W}$. Hence, $\Sigma_\beta^\beta \vdash_{\mathfrak{K}} (K, \beta)$ means that $\Sigma \vdash_{\mathfrak{K}} (K, \beta)$. □

We illustrate the use of the inference rules on our running example.

Example 7. Let Σ denote the p-key set from Example 2: ({*zone, time, rfid*}, β_1), ({*time, rfid*}, β_2), ({*zone, rfid*}, β_3), and ({*zone, time*}, β_4). Independent of the given Σ, the top-rule $\mathcal{T}$ allows us to infer ({*zone, time, rfid, object*}, β_1), and the bottom-rule $\mathcal{B}$ allows us to infer p-keys such as ({*zone*}, β_5) or even ($\emptyset$, β_5). From ({*time, rfid*}, β_2) we can infer ({*time, rfid, object*}, β_2) by means of the superkey-rule $\mathcal{S}$, and from the latter p-key we can infer ({*time, rfid, object*}, β_4) by application of the weakening-rule $\mathcal{W}$.

5.4. Algorithmic characterization

The axiomatization $\mathfrak{K}$ from the last subsection enables us to enumerate all p-keys implied by a p-key set Σ. In practice, however, it often suffices to decide whether a given p-key φ is implied by Σ. Enumerating all implied p-keys and checking whether φ is among them is neither efficient nor does it make good use of the input φ. We will now establish an efficient procedure to decide the implication problem for p-keys. The procedure is based on the following result which says that a p-key is implied by a p-key set if and only if it is trivial (i.e. contains all attributes or has bottom c-degree) or there is some p-key in the p-key set whose attribute set is contained in that of the given p-key and whose c-degree is higher than the c-degree of the given p-key.

Theorem 4. *Let $\Sigma \cup \{(K, \beta)\}$ denote a set of p-keys over $(R, \mathcal{S})$ with $|\mathcal{S}| = k + 1$. Then Σ implies (K, β) if and only if $\beta = \beta_{k+1}$, or $K = R$, or there is some $(K', \beta') \in \Sigma$ such that $K' \subseteq K$ and $\beta' \geq \beta$.*

Proof. Theorem 2 shows for $i = 1, \ldots, k$ that Σ implies (K, β_i) if and only if Σ_β implies K. It is known from the relational model of data [63] (or easy to observe from the axiomatization $\mathfrak{K}'$ of keys) that a key K is implied by a key set Σ_β if and only if $K = R$ or there is some $K' \in \Sigma_\beta$ such that $K' \subseteq K$ holds. From the definition of β-cuts it follows that $K' \in \Sigma_\beta$, if $(K', \beta') \in \Sigma$ for some $\beta' \geq \beta$. Consequently, the theorem holds for $i = 1, \ldots, k$. Furthermore, Σ implies (K, β_{k+1}), so the theorem follows. □

We apply Theorem 4 to our running example.

Example 8. Let Σ denote the p-key set from Example 2: ({*zone, time, rfid*}, β_1), ({*time, rfid*}, β_2), ({*zone, rfid*}, β_3), and ({*zone, time*}, β_4). Independent of the given Σ we can use Theorem 4 to observe that p-keys such as ({*zone, time, rfid, object*}, β_1), ({*zone*}, β_5) or even ($\emptyset$, β_5) are implied. The p-key ({*time, rfid, object*}, β_4) is implied by Σ because the given p-key ({*time, rfid*}, β_2) $\in \Sigma$ meets the conditions of Theorem 4 that {*time, rfid*} $\subseteq$ {*time, rfid, object*} and $\beta_2 \geq \beta_4$. In contrast, the p-key ({*time, rfid, object*}, β_1) is not implied by Σ since $\beta_1 \neq \beta_{k+1} =$

Algorithm 1 Implication.

Input: Set $\Sigma \cup \{(K, \beta)\}$ of p-keys over p-relation schema $(R, \mathcal{S})$ with $|\mathcal{S}| = k + 1$
Output: Yes, if $\Sigma \models (K, \beta)$, and No, otherwise

```
if K = R or β = β_{k+1} then
    return true;
else
    for all (K', β') ∈ Σ do
        if K' ⊆ K and β' ≥ β then
            return true;
    return false;
```

Algorithm 2 Non-redundant cover.

Input: Set Σ of p-keys over p-relation schema $(R, \mathcal{S})$
Output: A non-redundant cover $\Sigma_c \subseteq \Sigma$

```
Σ_c ← Σ;
for all σ ∈ Σ_c do
    if (Σ_c − {σ}) ⊨ σ then
        Σ_c ← (Σ_c − {σ});
return Σ_c;
```

β_5, {*time, rfid, object*} does not include *zone*, and there is no $(K', \beta') \in \Sigma$ such that $K' \subseteq$ {*time, rfid, object*} and $\beta' \geq \beta_1$.

Theorem 4 can be easily converted into an algorithm for deciding the implication problem of p-keys. The pseudo-code for such an algorithm is given in Algorithm 1. The algorithm first checks if the p-key in question is trivial and, if not, then goes through Σ to see if any member satisfies the remaining condition of Theorem 4 for implication. If no member can be found, the p-key is not implied. It is immediate that this algorithm runs in linear time in the input.

Corollary 5. *An instance $\Sigma \models \varphi$ of the implication problem for p-keys can be decided in time $\mathcal{O}(||\Sigma \cup \{\varphi\}||)$ where $||\Sigma||$ denotes the total number of symbol occurrences in Σ.*

This section has established axiomatic and algorithmic characterizations for the implication problem of possibilistic keys. These will form the foundation for subsequent sections.

6. Applications of possibilistic keys

We use this section to outline some important applications of p-keys. This provides further motivation for the study of their properties.

6.1. Constraint maintenance

As stated at the beginning of Section 5, the primary purpose of solving the implication problem for p-keys is to minimize the time required for validating sets of p-keys against an updated p-relation. A set Σ_c of p-keys is called a *non-redundant cover* of a given set Σ of p-keys if and only if $\Sigma_c^+ = \Sigma^+$ and Σ_c does not contain any p-key that is redundant. A p-key $\sigma \in \Sigma_c$ is said to be *redundant* with respect to Σ_c if $(\Sigma_c - \{\sigma\}) \models \sigma$. It is therefore our aim to compute a non-redundant cover for a given set of p-keys. This can be achieved by applying Algorithm 1 to check for each given p-key $\sigma \in \Sigma$ whether $(\Sigma - \{\sigma\}) \models \sigma$, and removing σ from Σ whenever that is the case. The pseudo-code of this algorithm is given in Algorithm 2. Let $|\Sigma|$ denote the cardinality of a given set Σ of p-keys, that is, the number of its elements. Indeed, every set Σ of p-keys has a *unique* non-redundant cover, given by

$$\Sigma_c = \{(K, \beta) \in \Sigma \mid K \neq R \wedge \beta < \beta_{k+1} \wedge \neg\exists (K', \beta') \in \Sigma \, ((K' \subset K \wedge \beta' \leq \beta) \vee (K' \subseteq K \wedge \beta' < \beta))\} \, .$$

For that reason we can speak of *the* non-redundant cover.

Corollary 6. *Algorithm 2 computes the non-redundant cover for a given set Σ of p-keys in time $\mathcal{O}(|\Sigma| \times ||\Sigma||)$.*

Proof. Algorithm 2 removes successively any redundant p-keys from the input Σ. All remaining members of Σ_c are not redundant. The time bound is therefore a consequence of Corollary 5. □

Algorithm 2 is non-deterministic due to the different orders in which redundant p-keys may be selected. However, the algorithm is confluent, that is the result is guaranteed to be the unique non-redundant cover. We illustrate the computation of the non-redundant cover on our running example.

Example 9. Let Σ denote the p-key set from Example 2: ({*zone, time, rfid*}, β_1), ({*time, rfid*}, β_2), ({*zone, rfid*}, β_3), and ({*zone, time*}, β_4). It is easily observed that none of the p-keys in Σ is implied by the subset of the remaining given p-keys. In other words, none of the elements in Σ is redundant. Consequently, Σ is its own non-redundant cover.

6.2. Data cleaning

In this subsection we illustrate an application of possibilistic keys to data cleaning. In [35], possibilistic data cleaning was introduced. In classical data cleaning, a minimal sequence of update operations are applied to the data to restore consistency with respect to given set of constraints. Possibilistic data cleaning offers a new view: It is not the data that is considered to be dirty, but the p-degrees associated with the tuples. In this sense, the p-degrees of tuples are changed minimally such that consistency with respect to the given set of possibilistic constraints is restored. This section illustrates this new view on our running example with respect to the class of possibilistic keys. For details of the algorithm we refer the interested reader to [35].

Given that the permitted update operations are limited to tuple deletions, the classical data cleaning problem with respect to keys can be stated as follows: Given a relation r and a set Σ of keys, find a relation $r' \subseteq r$ of maximum cardinality such that r' satisfies Σ. For example, the relation r (without the two last columns)

r					
zone	*time*	*rfid*	*object*	$Poss_r$	$Poss'_r$
Z3	1pm	H2	Grizzly	α_1	α_1
Z3	1pm	H3	Grizzly	α_1	α_2
Z3	3pm	H3	Grizzly	α_1	α_3
Z4	3pm	H3	Grizzly	α_1	α_4

violates the set $\Sigma = \{zt, zr, tr\}$ of keys. Solutions to the classical data cleaning problem would be the relations r_1 consisting of the first and third tuple, r_2 consisting of the first and last tuple, and r_3 consisting of the second and last tuple. Each solution requires us to remove at least two tuples from the relation. In this sense, classical data cleaning removes valuable information from the given relation.

We now consider possibilistic data cleaning as a means to minimize the removal of tuples from a p-relation. For this purpose, we exploit the c-degrees of p-keys to "reduce" the given p-degrees of tuples such that all p-keys will be satisfied.

Given two p-relations $(r', Poss_{r'})$ and $(r, Poss_r)$ we say that $(r', Poss_{r'})$ is a *p-subrelation* of $(r, Poss_r)$, denoted by $(r', Poss_{r'}) \subseteq_p (r, Poss_r)$, if and only if $r'_i \subseteq r_i$ for $i = 1, \ldots, k$. The p-subset relationship is simply the partial order of functions induced by the ordering on p-degrees, that is, $(r', Poss_{r'}) \subseteq_p (r, Poss_r)$ if and only if $Poss_{r'}(t) \leq Poss_r(t)$ holds for all tuples t. The *p-cardinality* of the p-relation $(r, Poss_r)$ is the mapping $\mathcal{C} : \alpha_i \mapsto |r_i|$ for $i = 1, \ldots, k$. We compare p-cardinalities with respect to the lexicographical order, that is,

$$\mathcal{C}_1 <_L \mathcal{C}_2 :\Leftrightarrow \exists \alpha_i (\mathcal{C}_1(\alpha_i) < \mathcal{C}_2(\alpha_i) \wedge \forall \alpha_j < \alpha_i (\mathcal{C}_1(\alpha_j) = \mathcal{C}_2(\alpha_j)))$$

The *possibilistic data cleaning problem* is: Given a p-relation r and set Σ of p-keys, find a p-subrelation $r' \subseteq_p r$ of maximal p-cardinality so that Σ holds on r'.

A point that might seem perhaps controversial in our problem definition is the use of the lexicographic order $<_L$ in defining our target function to optimize. We chose this linearization of the natural partial order between p-cardinalities over other candidates for several reasons. Firstly, leximin ordering is appropriate for accounting for the cardinality of level-cuts [14]. Secondly, by maximizing $|r'_k| = |r'|$, the number of tuples completely "lost" during data cleaning is minimized. Thirdly, it allows one to develop more efficient algorithms for computing it. For example, the p-relation $(r, Poss_r)$ violates the p-key set

$\Sigma = \{(zt, \beta_4), (zr, \beta_3), (tr, \beta_2)\}$.

However, if we change the p-degree of the second tuple to α_2, the p-degree of the third tuple to α_3, and the p-degree of the last tuple to α_4, then the resulting p-relation $(r, Poss'_r)$ satisfies Σ. Note that none of the p-degrees had to be set to the bottom degree α_5. That is, every tuple in the cleaned p-relation $(r, Poss'_r)$ is at least somewhat possible to occur.

We refer the interested reader to [35] in which a fixed parameter-tractable algorithm has been established for the possibilistic data cleaning problem.

6.3. Query processing

We demonstrate the benefit of p-keys on query processing. Therefore, we add the attribute *p-degree* to the relation schema TRACKING with attributes *zone*, *time*, *rfid*, *object*.

Suppose we are interested in finding out which grizzly bears have been tracked in which zone, but we are only interested in answers that come from 'certain' or 'quite possible' tuples in the database. A user might enter the following SQL query:

```
SELECT DISTINCT zone, rfid, p-degree
FROM TRACKING
WHERE p-degree = α1 OR p-degree = α2
ORDER BY p-degree ASC
```

zone	*rfid*	*p-degree*
Z0	H0	α_1
Z1	H1	α_1
Z1	H2	α_1
Z3	H2	α_1
Z3	H3	α_2

which removes duplicate answers, and orders them with decreasing p-degree. When applied to the p-relation from Table 1, the query returns the answers on the right.

Firstly, our framework allows users to ask such queries, since the p-degrees of tuples is available. Secondly, answers can be ordered according to the p-degree, which is a huge benefit for users in terms of ranking outputs. Thirdly, the example shows how our framework can be embedded with standard technology, here SQL. Finally, recall our p-key $(\{zone, rfid\}, \beta_3)$ which holds on the set of tuples that have p-degree α_1 or α_2. Consequently, the query answers satisfy the key $\{zone, rfid\}$ and the DISTINCT clause becomes superfluous. A query optimizer, capable of reasoning about p-keys, can remove the DISTINCT clause from the input query without affecting its output. This optimization saves response time when answering queries, as duplicate elimination is an expensive operation and therefore not executed by default in SQL databases. P-keys, and the ability to reason about them, have therefore direct applications to query processing.

If we want to use the data model to physically store and manage data, an operational language is required to evaluate queries. Despite being slightly out of scope, we are providing the basic definitions here for selection, projection, and join operations. For a p-relation $(r, Poss_r)$ and $X \subseteq R$, we define the *projection* $\pi_X(r) = \{t(X) \mid t \in r\}$ of r onto X, and $Poss_{\pi_X(r)}$ as $Poss_{\pi_X(r)}(t) = \max\{Poss_r(t') \mid t' \in r, t'(X) = t\}$. For a p-relation $(r, Poss_r)$, attribute $A \in R$, and constant $c \in dom(A)$, we define the *constant selection* $\sigma_{A=c}(r) = \{t \mid t \in r, t(A) = c\}$, and $Poss_{\sigma_{A=c}(r)}$ as $Poss_{\sigma_{A=c}(r)}(t) = Poss_r(t)$ for all $t \in \sigma_{A=c}(r)$. For a p-relation $(r, Poss_r)$, and attributes $A, B \in R$, we define the *attribute selection* $\sigma_{A=B}(r) = \{t \mid t \in r, t(A) = t(B)\}$, and $Poss_{\sigma_{A=B}(r)}$ as $Poss_{\sigma_{A=B}(r)}(t) = Poss_r(t)$ for all $t \in \sigma_{A=B}(r)$. Finally, for p-relations $(r, Poss_r)$ and $(s, Poss_s)$ over p-schemata $\sharp r$ and $\sharp s$ and the same linear orders of p-degrees, we define the *join* $r \bowtie s = \{t \mid \exists t' \in r, t'' \in s, t'(\sharp r) = t(\sharp r), t''(\sharp s) = t(\sharp s)\}$, and $Poss_{r \bowtie s}$ as $Poss_{r \bowtie s}(t) = \min\{Poss_r(t'), Poss_s(t'') \mid t' \in r, t'' \in s, t'(\sharp r) = t(\sharp r), t''(\sharp s) = t(\sharp s)\}$ for all $t \in r \bowtie s$. Note that these operators can also be defined by their possible worlds. That is, the ith possible world for each of these operations is equal to the classical relational algebra operation applied to the ith possible world of its operands. In particular, $(\pi_r)_i = \pi_{r_i}$, $(\sigma_{A=c}(r))_i = \sigma_{A=c}(r_i)$, $(\sigma_{A=B}(r))_i = \sigma_{A=B}(r_i)$, and $(r \bowtie s)_i = r_i \bowtie s_i$.

7. Extremal combinatorics for possibilistic keys

In this section we provide answers to fundamental questions concerning the maximum cardinality that non-redundant families of p-keys with at most ℓ attributes can have, and which families attain this cardinality. The result shows data engineers how large families of p-keys can potentially grow, which gives them insight into the complexity

of modeling required and clues for reducing this complexity. A characterization of non-redundant families enables us to apply techniques from extremal set theory to answer our questions. The main result is interesting from a combinatorial perspective itself, as it generalizes the famous theorem by Sperner [60].

We use the following notations: $[n] = \{1, 2, \ldots, n\}$ represents sets of attributes, $2^{[n]} = \{S : S \subseteq [n]\}$ the powerset of $[n]$, and $2^{[n]}_\ell = \{S : S \subseteq [n],\ |S| \leq \ell\}$ the elements of $2^{[n]}$ with at most ℓ attributes. Recall from before that a set Σ of constraints is *non-redundant* if for all $\sigma \in \Sigma$, σ is not implied by $\Sigma - \{\sigma\}$. For $0 \leq \ell < n$ and a set $\Sigma \subseteq 2^{[n]}_\ell \times [k]$ of p-keys (K, i) where $i \in [k]$ denotes the index of its c-degree β_i, Σ is non-redundant if and only if Σ does not contain two distinct elements (X, i) and (Y, j) with $X \subseteq Y$ and $i \leq j$.

The proof of the following theorem uses the properties that $p(\mathcal{F}) = \{X \mid (X, i) \in \mathcal{F}\}$ is a *k-Sperner family* [17], i.e. does not contain chains of length $k + 1$ and $2^{[n]}_\ell$ has the *strong Sperner property* w.r.t. set inclusion [32,60], i.e. the size of the largest k-Sperner family in $2^{[n]}_\ell$ is the sum of the $\min\{k, \ell + 1\}$ largest of the binomial coefficients $\binom{n}{i}$, $i = 0, 1, \ldots, \ell$.

Theorem 7. *Let n, ℓ, k be integers with $0 \leq \ell < n$ and $k \geq 1$, and let*

$$\mathcal{G}_1 = \left\{(X, i) \in 2^{[n]} \times [k] : |X| + i = \left\lfloor \tfrac{n+k+1}{2} \right\rfloor\right\},$$
$$\mathcal{G}_2 = \left\{(X, i) \in 2^{[n]} \times [k] : |X| + i = \left\lceil \tfrac{n+k+1}{2} \right\rceil\right\},$$
$$\mathcal{H} = \{(X, i) \in 2^{[n]} \times [k] : |X| + i = \ell + 1\}.$$

Let $\mathcal{F} \subseteq 2^{[n]}_\ell \times [k]$ such that $\mathcal{F}$ does not contain two distinct elements (X, i) and (Y, j) with $X \subseteq Y$ and $i \leq j$.

(i) *If $\ell \geq (n + k - 1)/2$, then $|\mathcal{F}| \leq |\mathcal{G}_1|$, where equality holds if $\mathcal{F} \in \{\mathcal{G}_1, \mathcal{G}_2\}$.*
(ii) *If $\ell < (n + k - 1)/2$, then $|\mathcal{F}| \leq |\mathcal{H}|$, where equality holds if $\mathcal{F} = \mathcal{H}$.*

Proof. For $F \in \mathcal{F}$ let $p(F)$ be its projection to $2^{[n]}_\ell$, i.e., if $F = (X, i)$, then $p(F) = X$. By the properties of $\mathcal{F}$, the mapping $p : \mathcal{F} \mapsto 2^{[n]}$ is a bijection. Hence, $|p(\mathcal{F})| = |\mathcal{F}|$.

Assume that $p(\mathcal{F})$ contains a chain of length $k + 1$, that is, there are distinct sets $X_1, \ldots, X_k, X_{k+1} \in p(\mathcal{F})$ with $X_1 \subset X_2 \subset \cdots \subset X_{k+1}$. By the pigeonhole principle, there must be distinct $s, t \in [k + 1]$ such that $p^{-1}(X_s) = (X_s, i)$ and $p^{-1}(X_t) = (X_t, i)$ for some $i \in [k]$, contradicting the choice of $\mathcal{F}$. Consequently, $p(\mathcal{F})$ does not contain a chain of length $k + 1$. In other words, $p(\mathcal{F})$ is a *k-Sperner family* [17].

$2^{[n]}_\ell$ has the *strong Sperner property* w.r.t. set inclusion, i.e., for any k the size of the largest k-Sperner family in $2^{[n]}_\ell$ is the sum of the $\min\{k, \ell + 1\}$ largest of the binomial coefficients $\binom{n}{i}$, $i = 0, 1, \ldots, \ell$. Clearly, $|p(\mathcal{F})| = |\mathcal{F}|$ attains this maximum size if $\mathcal{F} = \mathcal{G}_1$ or $\mathcal{F} = \mathcal{G}_2$ when $\ell \geq \frac{n+k-1}{2}$ and if $\mathcal{F} = \mathcal{H}$ when $\ell < \frac{n+k-1}{2}$.

To see that $2^{[n]}_\ell$ has the strong Sperner property, note that [60] proved that the poset $(2^{[n]}, \subseteq)$ has the *normalized matching property*. Such posets are called *normal*. We refer to Section 4.5 of Engel's book for definition. Now $(2^{[n]}_\ell, \subseteq)$ as a rank-selected subposet of $(2^{[n]}, \subseteq)$ is normal, too. (See Proposition 4.5.3 in [17].) Finally, normal posets have the strong Sperner property [32]. □

From the above proof and the LYM inequality [17] it is easy to derive the following characterization of the extremal $\mathcal{F}$ in Theorem 7.

(a) For $n > \ell \geq (n + k - 1)/2$, equality holds in Theorem 7(i) if and only if $\mathcal{F} \in \{\mathcal{G}_1, \mathcal{G}_2\}$.
(b) For $(n + k - 1)/2 > \ell \geq k - 1$, equality holds in Theorem 7(ii) if and only if $\mathcal{F} = \mathcal{H}$.
(c) For $k - 2 \geq \ell \geq 0$, equality in Theorem 7(ii) if and only if $p(\mathcal{F}) := \{X \mid (X, i) \in \mathcal{F}\} = 2^{[n]}_\ell$.

Note that the condition in (a) implies $\ell \geq k$, and the condition in (c) implies $\ell < (n + k - 1)/2$ and $p(\mathcal{H}) = 2^{[n]}_\ell$.

Fig. 2 shows the case of $n = 4$ attributes, $k = 2$ c-degrees other than β_{k+1}, and $\ell = 2$ the maximum size of a key. Unary/binary keys (marked red/green) have c-degree β_2/β_1, respectively.

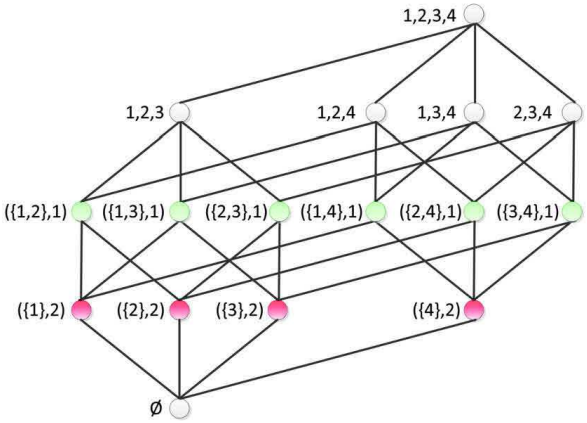


Fig. 2. Case $n = 4$, $k = 2$, $\ell = 3$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

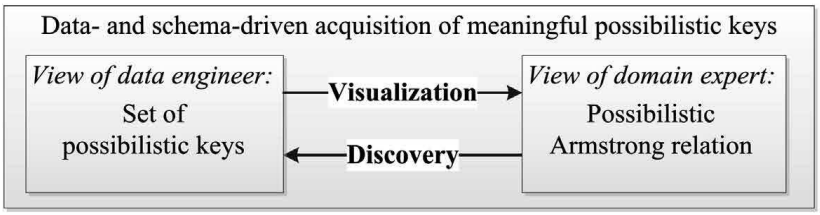


Fig. 3. Acquisition framework for possibilistic keys.

8. Acquisition tools

New applications benefit from the ability of data engineers to acquire the p-keys that are semantically meaningful in the domain of the application. For that purpose, data engineers communicate with domain experts. Unfortunately, they have to overcome a communication problem which is caused by a mismatch in expertise. Indeed, data engineers know database concepts but not the domain, while domain experts know the domain but not database concepts. It is therefore helpful to translate the knowledge of the data engineers into a format that can be understood by the domain experts. As humans learn a lot from good examples, the goal is to visualize the abstract set of p-keys that data engineers currently perceive as meaningful in the form of a data sample. Here, the data sample should be representative of the abstract p-key set. This means the data sample satisfies all the elements of the set, but violates all the p-keys not implied by the abstract set. The intuitive idea is that domain experts will easily spot cases in which actually meaningful p-keys are violated by the data sample, because they are incorrectly perceived as meaningless by the data engineers. The notion of a representative data sample is also known as an Armstrong database in the research literature.

In this section we will establish two major tools that help data engineers to effectively communicate with domain experts. We follow the framework in Fig. 3. We will first establish an algorithm that data engineers can use to visualize abstract p-key sets Σ in the form of some Armstrong p-relation r_Σ. This p-relation is then inspected jointly with the domain experts. The domain experts may change r_Σ or supply entirely new data samples to the engineers. For that case we establish an algorithm that computes a non-redundant cover of the set of p-keys that hold in the data sample. In subsequent sections we will describe prototypes that have implemented these algorithms, and report on results we obtained by experimenting with these prototypes.

8.1. Structure and computation of Armstrong relations for keys and Armstrong p-relations for p-keys

8.1.1. Keys and relations

We first recall the definition and results for Armstrong relations in the context of relations and keys [2,47]. Here, a relation r over relation schema R is Armstrong for a given set Σ of keys over R if and only if for every key K over R it is true that r satisfies K if and only if K is implied by Σ. In this sense, the implication problem for any key by a given key set Σ reduces to checking whether K holds on an Armstrong relation for Σ.

Example 10. Let Σ denote the p-key set from Example 2. Then a non-redundant cover of Σ_{β_4} consists of the keys {*time, rfid*}, {*zone, rfid*}, and {*zone, time*}. The relation r_1 of Table 1 is an Armstrong relation for Σ_{β_4}. It satisfies all the given keys, and violates all keys that are not implied by Σ_{β_4}. For example, the keys {*time*, *object*}, {*zone*, *object*}, and {*rfid*, *object*} are violated by r_1. In fact, these are the maximal keys (with respect to subset order) that are not implied by Σ_{β_4}, also known as anti-keys.

In the relational model, a given relation is Armstrong for a given key set if and only if the relation satisfies every given key and violates every anti-key for the given key set. The computation of an Armstrong relation for a given key set therefore involves the computation of the anti-keys for the given key set, followed by the insertion of a new tuple for each anti-key that has matching values with the previous tuple on exactly those attributes that belong to the anti-key. In Example 10, pairs of consecutive tuples have matching values on exactly those attributes that belong to the anti-keys {*time*, *object*}, {*zone*, *object*}, and {*rfid*, *object*}.

8.1.2. P-keys and p-relations

Our goal is now to extend these results to p-relations and p-keys. A p-relation $(r, Poss_r)$ over $(R, \mathcal{S})$ is *Armstrong* for a p-key set Σ if and only if for all p-keys φ over $(R, \mathcal{S})$, $(r, Poss_r)$ satisfies φ if and only if $\Sigma \models \varphi$. Armstrong p-relations therefore have the beautiful property that the maximum c-degree β by which a p-key (K, β) is implied by Σ can 'simply be read-off' as the marginal certainty $C_r(K)$ of K in any Armstrong p-relation $(r, Poss_r)$ for Σ.

Example 11. Let Σ denote the p-key set from Example 2: ({*zone, time, rfid*}, β_1), ({*time, rfid*}, β_2), ({*zone, rfid*}, β_3), and ({*zone, time*}, β_4). Then the p-relation r in Table 1 is Armstrong for Σ. For example, the key {*zone, time, object*} has marginal certainty C_r(*zone, time, object*) $= \beta_4$, since the smallest possible world that violates this key is r_2. Indeed, $\beta = \beta_4$ is the largest c-degree such that the p-key ({*zone, time, object*}, β) is implied by Σ.

Our first aim is to characterize the structure of Armstrong p-relations. We recall two notions from relational databases. The *agree set* of two tuples t, t' over R is the set $ag(t, t') = \{a \in R \mid t(a) = t'(a)\}$ of attributes on which t and t' have matching values. The agree set of a relation is the set $ag(r) = \{ag(t, t') \mid t, t' \in r \wedge t \neq t'\}$. Let Σ denote a set of keys over relation schema R. An *anti-key* of R with respect to Σ is a subset $A \subseteq R$ such that Σ does not imply the key A over R and for all $a \in R - A$, Σ implies the key $A \cup \{a\}$ over R. We denote by Σ^{-1} the set of all anti-keys of R with respect to Σ.

Example 12. Consider the possible world r_4 of the p-relation r in Table 1. Then the set of agree sets of r_4 consists of {*time, object*}, {*object*}, {*zone, object*}, {*rfid, object*}, {*zone, time, object*}, {*zone, rfid, object*}, and {*time, rfid, object*}.

Let Σ_{β_1} denote the β_1-cut of the p-key set from Example 2, that is, $\Sigma_{\beta_1} =$ {*zone, time, rfid*}. Then the set $\Sigma_{\beta_1}^{-1}$ of anti-keys consists of {*zone, time, object*}, {*zone, rfid, object*}, and {*time, rfid, object*}.

We will now characterize when a given p-relation is Armstrong with respect to a given set of p-keys. This is the case precisely when every possible world r_{k+1-i} of the given p-relation is an Armstrong relation for the β_i-cut of the given p-key set Σ.

Theorem 8. *Let Σ denote a set of p-keys, and let $(r, Poss_r)$ denote a p-relation over $(R, \mathcal{S})$ with $|\mathcal{S}| = k + 1$. Then $(r, Poss_r)$ is Armstrong for Σ if and only if for all $i = 1, \ldots, k$, the relation r_{k+1-i} is Armstrong for Σ_{β_i}. That is, for all $i = 1, \ldots, k$, $\Sigma_{\beta_i}^{-1} \subseteq ag(r_{k+1-i})$, and for all $K \in \Sigma_{\beta_i}$ and for all $X \in ag(r_{k+1-i})$, $K \not\subseteq X$.*

Proof. $(r, Poss_r)$ is Armstrong for Σ if and only if for all $i = 1, \ldots, k$, for all $K \subseteq R$, $\models_{(r, Poss_r)} (K, \beta_i)$ iff $\Sigma \models (K, \beta_i)$. However, $\models_{(r, Poss_r)} (K, \beta_i)$ iff $\models_{r_{k+1-i}} K$, and $\Sigma \models (K, \beta_i)$ iff $\Sigma_{\beta_i} \models K$. Therefore, $(r, Poss_r)$ is Armstrong for Σ if and only if for all $i = 1, \ldots, k$, r_{k+1-i} is an Armstrong relation for Σ_{β_i}. The second statement follows straight from the well-known result that a relation r is Armstrong for a set Σ of keys if and only if $\Sigma^{-1} \subseteq ag(r)$ and for all $K \in \Sigma$ and all $X \in ag(r)$, $K \not\subseteq X$ [10]. □

We illustrate Theorem 8 on our running example.

Example 13. Let Σ denote the p-key set from Example 2: ($\{zone, time, rfid\}, \beta_1$), ($\{time, rfid\}, \beta_2$), ($\{zone, rfid\}, \beta_3$), and ($\{zone, time\}, \beta_4$). Let r denote the p-relation from Table 1.

The agree sets of r_1 consists of *to*, *o*, *zo*, and *ro*. The agree sets of r_2 include those of r_1 and *zto*. The agree sets of r_3 include those of r_2 and *zro*. The agree sets of r_4 include those of r_3 and *tro*. The β-cuts Σ_β of Σ and anti-key sets Σ_β^{-1} are (using the first letters of the attributes):

- $\Sigma_{\beta_1} = \{ztr\}$ and $\Sigma_{\beta_1}^{-1} = \{zto, tro, zro\}$
- $\Sigma_{\beta_2} = \{tr\}$ and $\Sigma_{\beta_2}^{-1} = \{zto, zro\}$
- $\Sigma_{\beta_3} = \{tr, zr\}$ and $\Sigma_{\beta_3}^{-1} = \{zto, ro\}$, and
- $\Sigma_{\beta_4} = \{tr, zr, zt\}$ and $\Sigma_{\beta_4}^{-1} = \{to, zo, ro\}$.

For r_1 we can see that every anti-key in $\Sigma_{\beta_4}^{-1}$ is also an agree set of r_1, and that no key in Σ_{β_4} is contained in an agree set of r_1. Hence, r_1 is Armstrong for Σ_{β_4}. For r_2 we can see that every anti-key in $\Sigma_{\beta_3}^{-1}$ is also an agree set of r_2, and that no key in Σ_{β_3} is contained in an agree set of r_2. Hence, r_2 is Armstrong for Σ_{β_3}. For r_3 we can see that every anti-key in $\Sigma_{\beta_2}^{-1}$ is also an agree set of r_3, and that no key in Σ_{β_2} is contained in an agree set of r_3. Hence, r_3 is Armstrong for Σ_{β_2}. For r_4 we can see that every anti-key in $\Sigma_{\beta_1}^{-1}$ is also an agree set of r_4, and that no key in Σ_{β_1} is contained in an agree set of r_4. Hence, r_4 is Armstrong for Σ_{β_1}. We conclude, by Theorem 8, that r is an Armstrong p-relation for Σ.

We can now use Theorem 8 to develop an algorithm that computes for a given p-key set Σ an Armstrong p-relation for Σ. The pseudo-code of this algorithm is given as Algorithm 3. It computes for $i = 1, \ldots, k$ the sets $\Sigma_{\beta_i}^{-1}$ of anti-keys incrementally. Starting with a tuple of p-degree α_1, for $i = k, \ldots, 1$, each $A \in \Sigma_{\beta_i}^{-1}$ is realized as an agree set by introducing a tuple that agrees with the previous tuple on A and has p-degree α_{k+1-i}, as long as A did not already occur for some larger i. In fact, if A already occurred for some larger i, then the corresponding agree set has already been realized.

We illustrate Algorithm 3 on our running example.

Example 14. We apply Algorithm 3 to the set Σ from Example 2. Using the first letters of each attribute we obtain

- $\Sigma_1 = \{ztr\}$ and $\Sigma_{\beta_1}^{-1} = \{zto, \underline{tro}, zro\}$
- $\Sigma_2 = \{tr\}$ and $\Sigma_{\beta_2}^{-1} = \{zto, \underline{zro}\}$
- $\Sigma_3 = \{tr, zr\}$ and $\Sigma_{\beta_3}^{-1} = \{\underline{zto}, ro\}$, and
- $\Sigma_4 = \{tr, zr, zt\}$ and $\Sigma_{\beta_4}^{-1} = \{\underline{to}, \underline{zo}, \underline{ro}\}$.

Anti-keys are underlined when they are realized as agree sets of tuples in the Armstrong p-relation:

zone	*time*	*rfid*	*object*	*Poss. degree*
$c_{z,0}$	$c_{t,0}$	$c_{r,0}$	$c_{o,0}$	α_1
$c_{z,1}$	$c_{t,0}$	$c_{r,1}$	$c_{o,0}$	α_1
$c_{z,1}$	$c_{t,2}$	$c_{r,2}$	$c_{o,0}$	α_1
$c_{z,3}$	$c_{t,3}$	$c_{r,2}$	$c_{o,0}$	α_1
$c_{z,3}$	$c_{t,3}$	$c_{r,4}$	$c_{o,0}$	α_2
$c_{z,3}$	$c_{t,5}$	$c_{r,4}$	$c_{o,0}$	α_3
$c_{z,6}$	$c_{t,5}$	$c_{r,4}$	$c_{o,0}$	α_4

Fitting substitution yields the p-relation from Table 1.

Finally, we derive some results regarding the space and time complexity of our algorithm. Here, an Armstrong p-relation for Σ is said to be *minimum-sized* if and only if there is no Armstrong p-relation for Σ with fewer tuples.

Algorithm 3 Visualize.

Input: P-key set Σ over p-relation schema $(R, \{\beta_1, \ldots, \beta_k, \beta_{k+1}\})$
Output: Armstrong p-relation $(r, Poss_r)$ for Σ

$\Sigma_0^{-1} \leftarrow \{R - \{a\} \mid a \in R\}$;
for $i = 1, \ldots, k$ **do** ▷ Compute $\Sigma_{\beta_i}^{-1}$ incrementally
  $\Sigma_i \leftarrow \{K \mid (K, \beta_j) \in \Sigma \text{ and } j \leq i\}$,
  $\Sigma_i^{-1} \leftarrow \text{AntiKeys}(R, \Sigma_i, \Sigma_{i-1}^{-1})$,
for all $a \in R$ **do**
  $t_0(a) \leftarrow c_{a,0}$; ▷ $c_{a,i}$ are fresh constants
$j \leftarrow 0$; $r \leftarrow \{t_0\}$; $Poss_r(t_0) \leftarrow \alpha_1$; $\Sigma_0 \leftarrow \emptyset$;
for $i = k$ downto 1 **do**
  for all $A \in \Sigma_i^{-1} - \Sigma_0$ **do**
    $j \leftarrow j + 1$;
    for all $a \in R$ **do** ▷ New tuple with agree set A
      if $a \in A$ **then** $t_j(a) \leftarrow t_{j-1}(a)$;
      else $t_j(a) \leftarrow c_{a,j}$;
    $Poss_r(t_j) \leftarrow \alpha_{k+1-i}$; ▷ and p-degree α_{k+1-i}
    $r \leftarrow r \cup \{t_j\}$;
  $\Sigma_0 \leftarrow \Sigma_0 \cup \Sigma_i^{-1}$;
return $(r, Poss_r)$;

Subroutine AntiKeys(R, Σ, Σ^{-1})
Input: R, Σ set of keys in Σ_i, Σ^{-1} set of anti-keys in Σ_{i-1}^{-1}
Output: Σ^{-1} set of anti-keys for Σ_{β_i}
for all $K \in \Sigma$, $A \in \Sigma^{-1}$ with $K \subseteq A$ **do**
  $\Sigma^{-1} \leftarrow (\Sigma^{-1} - \{A\}) \cup \bigcup_{a \in K} \{A - \{a\}\}$;
$\Sigma^{-1} \leftarrow \{A \mid \forall B \in \Sigma^{-1} - \{A\}(A \not\subseteq B)\}$;
return Σ^{-1};

Theorem 9. *Algorithm 3 computes an Armstrong p-relation for Σ whose size is at most quadratic in that of a minimum-sized Armstrong p-relation for Σ.*

Proof. The soundness of Algorithm 3 follows from Theorem 8, which also shows that for $\Sigma^{-1} = \bigcup_{i=1}^{k} \Sigma_i^{-1}$ we have $|\Sigma^{-1}| \leq ag(r) \leq \binom{|r|}{2}$. The inequalities establish the lower bound in $\frac{1}{2} \cdot \sqrt{1 + 8 \cdot |\Sigma^{-1}|} \leq |r| \leq |\Sigma^{-1}| + 1$. The upper bound follows from Algorithm 3. Hence, the p-relation computed by Algorithm 3 is at most quadratic in the size of a minimum-sized Armstrong p-relation for Σ. □

Finding Armstrong p-relations is precisely exponential. That means that there is an algorithm for computing an Armstrong p-relation whose running time is exponential in the size of Σ, and that there is some set Σ in which the number of tuples in each minimum-sized Armstrong p-relation for Σ is exponential — thus, an exponential amount of time is required in this case simply to write down the p-relation.

Theorem 10. *Finding an Armstrong p-relation for a p-key set Σ is precisely exponential in the size of Σ.*

Proof. Algorithm 3 computes an Armstrong p-relation for Σ in time at most exponential in its size. Some p-key sets Σ have only Armstrong p-relations with exponentially many tuples in the size of Σ. For $R = \{a_1, \ldots, a_{2n}\}$, $\mathcal{S} = \{\alpha_1, \alpha_2\}$ and $\Sigma = \{(\{a_1, a_2\}, \beta_1), \ldots, (\{a_{2n-1}, a_{2n}\}, \beta_1)\}$ with size $2 \cdot n$, Σ^{-1} consists of the 2^n anti-keys $\bigcup_{j=1}^{n} X_j$ where $X_j \in \{a_{2j-1}, a_{2j}\}$. □

Armstrong *p*-relations for some other p-key sets Σ' only require a number of tuples that is logarithmic in the size of Σ'. Such a set Σ' is given by the 2^n p-keys $(\bigcup_{j=1}^{n} X_j, \beta_1)$ where $X_j \in \{a_{2j-1}, a_{2j}\}$. In fact, Algorithm 3 computes an Armstrong p-relation for Σ' with $n + 1$ tuples. For $n = 3$, for example, we obtain the eight p-keys $(\{a_1, a_3, a_5\}, \beta_1)$, $(\{a_1, a_3, a_6\}, \beta_1)$, $(\{a_1, a_4, a_5\}, \beta_1)$, $(\{a_1, a_4, a_6\}, \beta_1)$, $(\{a_2, a_3, a_5\}, \beta_1)$, $(\{a_2, a_3, a_6\}, \beta_1)$,

$(\{a_2, a_4, a_5\}, \beta_1)$, and $(\{a_2, a_4, a_6\}, \beta_1)$. These result in the three anti-keys $(\{a_1, a_2, a_3, a_4\}, \beta_1)$, $(\{a_1, a_2, a_5, a_6\}, \beta_1)$, and $(\{a_3, a_4, a_5, a_6\}, \beta_1)$, which produces an Armstrong p-relation with 4 tuples.

It is our recommendation to use both representation of the constraints: the abstract set Σ and an Armstrong p-relation for Σ. Later sections will discuss an implementation of Algorithm 3 and results we obtained from experiments with the implementation.

8.2. Discovery

We are now addressing the discovery part of our acquisition framework illustrated in Fig. 3. The problem is, given a p-relation over some p-relation schema, compute a cover of the set of p-keys that hold on the p-relation. In other words, given a p-relation we would like to compute a set Σ of p-keys for which the p-relation is Armstrong. This problem was first introduced in [46] under the name *dependency inference*. It is also known as *dependency mining* or *constraint mining*, and in recent times the problem is considered to be an important part of *data profiling*. We will first show that the decision variant of this problem is both NP-complete and $W[2]$-complete in the size of the p-key, thus illustrating that it is unlikely that there are tractable algorithms to solve this problem, even in case the size of the p-key is fixed. Subsequently, we develop an algorithm based on hypergraph transversals.

8.2.1. The computational complexity of the discovery problem

Before we can state and prove the (parameterized) computational complexity of the p-key discovery problem, we formally state its decision variant and recall some necessary definitions from the area of (parametrized) complexity theory [12,13].

The problem $(\textsc{Key}, n)$ is to decide whether for a given relation r over relation schema R and a given positive integer n there is some $X \subseteq R$ of size $|X| \leq n$ such that r satisfies the key X. Note that $(\textsc{Key}, n)$ does not depend upon asking for a solution of size at most n as opposed to one of size exactly n, since adding attributes to a satisfied key X will result in a satisfied key XY.

Similarly, the problem $(\text{p-}\textsc{Key}, n)$ is to decide whether for a given p-relation $(r, Poss_r)$ over p-relation schema $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$, a given c-degree β_i with $1 \leq i \leq k$, and a given positive integer n, there is some $X \subseteq R$ of size $|X| \leq n$ such that $(r, Poss_r)$ satisfies the p-key (X, β_i). Again, $(\text{p-}\textsc{Key}, n)$ does not depend upon asking for a solution of size at most n as opposed to one of size exactly n, since adding attributes to a satisfied p-key (X, β_i) will result in a satisfied p-key (XY, β_i).

For an instance I of a decision problem and a parameter $n \in \mathbb{N}^+$, the pair (I, n) is an instance of the corresponding parameterized problem. The running time of an algorithm is then considered not only in terms of the input size $|I|$ but also in terms of n. A parameterized problem is fixed-parameter tractable, that is, it belongs to the complexity class FPT, if a given instance can be solved in time $\mathcal{O}(f(n) \cdot p(|I|))$, where p is a polynomial while f is an arbitrary computable function. We then also say that the algorithm runs in FPT-time.

Let P and P' be two parameterized problems. A parameterized reduction from P to P' is an algorithm running in FPT-time that maps an instance (I, n) of P to an equivalent instance (I', n') of P' such that the parameter n' depends only on the value of n (and not on $|I|$). Note that an (hypothetical) FPT-algorithm for P' would also yield an FPT-algorithm for P via this reduction. Hence, considering their parameterized complexity, P is at most as hard as P', which we denote by $P \leq_{FPT} P'$. If conversely $P' \leq_{FPT} P$ also holds, we say that P and P' are FPT-equivalent.

The parameterized reduction leads to a hierarchy of complexity classes, the so-called W-hierarchy, by specifying a complete problem for each class. To define the desired family of problems, we employ Boolean formulas in propositional logic. Let φ be such a formula. A satisfying truth assignment for φ has Hamming weight n if exactly n variables are set to true in this assignment; we also call the set of these n variables a solution for φ. The formula φ is t-normalized if it can be written as a conjunction of disjunctions of conjunctions of disjunctions (and so on) of literals with $t - 1$ alternations between conjunction and disjunction. Observe that a Boolean formula is 2-normalized if it is in conjunctive normal form (CNF) and 3-normalized if it is a conjunction of subformulas in disjunctive normal form (DNF).

The problem WEIGHTED T-NORMALIZED SATISFIABILITY is to decide for a given t-normalized formula φ and a positive integer n whether φ has a weight n satisfying assignment; here n serves as the parameter. For any $t \geq 1$, a parameterized problem P is said to be in the complexity class $W[t]$ in case $P \leq_{FPT}$ WEIGHTED T-NORMALIZED SATISFIABILITY.

The classes $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots$ form an ascending hierarchy and all inclusion are assumed to be proper, which is however still unproven. The higher a problem ranks in the W-hierarchy, the lower the chances are generally considered of finding an FPT-algorithm to solve it.

We are now in a position to state and prove our result about the computational complexity of the p-key discovery problem.

Theorem 11. *The problem* P-KEY *is NP-complete, and* $W[2]$*-complete in the size of the p-key.*

Proof. For the $W[2]$-completeness it suffices to show that $(\text{KEY}, k) \leq_{FPT} (\text{P-KEY}, k) \leq_{FPT} (\text{KEY}, k)$, since the result follows then from the $W[2]$-completeness of (KEY, k) in k [4].

For $(\text{KEY}, k) \leq_{FPT} (\text{P-KEY}, k)$ we assume an instance (r, k) of (KEY, k) is given with r being a relation over relation schema R. Now we simply transform r into a p-relation $(r, Poss_r)$ over $(R, \{\alpha_1, \alpha_2\})$ by assigning the α_1-degree to each tuple $t \in r$, and assigning the α_2-degree to each tuple $t \notin r$. Consequently, a $k(X)$ over R with $|X| \leq k$ is satisfied by r if and only if the p-key $(k(X), \beta_1)$ over $(R, \{\alpha_1, \alpha_2\})$ with $|X| \leq k$ is satisfied by $(r, Poss_r)$. Note that the transformation is the identity on k.

For $(\text{P-KEY}, k) \leq_{FPT} (\text{KEY}, k)$ we assume an instance $((r, Poss_r), \beta_i, k)$ of $(\text{P-KEY}, k)$ is given with $(r, Poss_r)$ being a p-relation over p-relation schema $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ and $1 \leq i \leq k$. Now we simply transform $(r, Poss_r)$ into the relation $r_{k+1-i} = \{t \in r \mid Poss_r(t) \leq \alpha_{k+1-i}\}$ over R. Consequently, a p-key $(k(X), \beta_i)$ over $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ with $|X| \leq k$ is satisfied by $(r, Poss_r)$ if and only if the key $k(X)$ over R with $|X| \leq k$ is satisfied by r_{k+1-i}. Note that the transformation is the identity on k.

The FPT-reductions show, in particular, that (KEY, k) and $(\text{P-KEY}, k)$ are PTIME-equivalent, too. Since (KEY, k) is NP-complete [2], it follows that $(\text{P-KEY}, k)$ is NP-complete, too. □

The experimental results from Section 10 regarding the discovery of p-keys should be viewed in light of Theorem 11.

8.2.2. Algorithmic solution

In the relational model, a key ensures that for each pair of distinct tuples there is some attribute of the key on which the tuples have different values. A good strategy to discover all (minimal) keys is therefore to compute for all pairs of distinct tuples their disagree set, that is, the set of attributes on which they have different values, and then compute all minimal sets of attributes which intersect non-trivially with every disagree set. The latter problem is also known as the hypergraph transversal problem [16], whose exact complexity is still open. We will now adopt this strategy to the computation of all p-keys that hold on a given p-relation.

More formally, a *hypergraph* (V, E) consists of a vertex set V and a set E of subsets of V, called hyperedges. A set $T \subseteq V$ is a *transversal* of (V, E) if for all $H \in E$, $T \cap H \neq \emptyset$ holds. A transversal T of (V, E) is *minimal* if there is no transversal T' of (V, E) such that $T' \subsetneq T$ [16]. With this terminology, we can now explain the pseudo-code of Algorithm 4, which computes a non-redundant cover for the set of p-keys that hold on the given p-relation. In lines (1–4), Algorithm 4 computes, for $i = 1, \ldots, k$, the minimal transversals of the hypergraph that has the underlying attributes as vertex set and minimal disagree sets of tuples from world r_i as hyperedges. These form a cover of the set of minimal keys that hold on r_i. The marginal certainty of those keys in the given p-relation is thus at least β_{k+1-i}. Line (5) takes the union of the p-keys that originate from the possible worlds. Finally, Theorem 4 is used to select only those p-keys that are not implied by the other p-keys.

We illustrate Algorithm 4 on our running example.

Example 15. We apply Algorithm 4 to the p-relation from Table 1. Using the first letters of each attribute we obtain

- $dis\text{-}ag(r_1) = \{zr, tr, zt\}$ and $\Sigma_1 = \{(zr, \beta_4), (tr, \beta_4), (zt, \beta_4)\}$
- $dis\text{-}ag(r_2) = \{zt, r\}$ and $\Sigma_2 = \{(zr, \beta_3), (tr, \beta_3)\}$
- $dis\text{-}ag(r_3) = \{t, r\}$ and $\Sigma_3 = \{(tr, \beta_2)\}$, and
- $dis\text{-}ag(r_4) = \{z, t, r\}$ and $\Sigma_4 = \{(ztr, \beta_1)\}$.

A non-redundant cover Σ for the p-keys that hold on the p-relation consists of (ztr, β_1), (tr, β_2), (zr, β_3), and (zt, β_4).

Algorithm 4 Discover.

Input: $(r, Poss_r)$ over $(R, \{\beta_1, \ldots, \beta_{k+1}\})$
Output: Non-redundant cover Σ of the set of p-keys that are satisfied by $(r, Poss_r)$

for $i = 1, \ldots, k$ **do**
  $dis\text{-}ag(r_i) \leftarrow \min\{X \subseteq R \mid \exists t, t' \in r_i \forall a \in R(t(a) \neq t'(a) \leftrightarrow a \in X)\}$;
  $\mathcal{H}_i \leftarrow (R, dis\text{-}ag(r_i))$;
  $\Sigma_i \leftarrow \{(K, \beta_{k+1-i}) \mid K \in Tr(\mathcal{H}_i)\}$;
$\Sigma \leftarrow \bigcup_{i=1}^{k} \Sigma_i$;
$\Sigma \leftarrow \{(K, \beta) \in \Sigma \mid \neg\exists (K', \beta') \in \Sigma (K' \subseteq K \wedge \beta' > \beta)\}$;
return Σ;

Finally, we state the correctness and an upper bound of the time complexity of Algorithm 4.

Theorem 12. *Algorithm 4 computes a non-redundant cover of the set of p-keys that are satisfied by the given p-relation r in time $\mathcal{O}(m + n^2)$ where $m := |R|^2 \times |r_k|^2 \times |dis\text{-}ag(r_k)|$ and $n := \prod_{X \in dis\text{-}ag(r_k)} |X|$.*

Proof. The soundness follows from the result that the keys of a relation are the minimal transversals of the disagree sets in the relation [10,48], and Theorem 4. The collection $dis\text{-}ag(r_i)$ is computed in time $\mathcal{O}(m)$. The set of all minimal transversals for the simple hypergraph $\mathcal{H}_i$ is computed in time $\mathcal{O}(n^2)$. Algorithm 4 can compute the minimal hypergraphs incrementally with additional disagree sets discovered from tuples with lower p-degrees. □

Subsequent sections will discuss an implementation of Algorithm 4 and results we obtained from experiments with the implementation.

9. Prototype systems for visualization and discovery

This section briefly introduces two graphical user interfaces that provide access to implementations of Algorithm 3 and Algorithm 4, respectively. Section 10 will discuss the results of various experiments with these interfaces. The following section will discuss the outcomes of detailed experimental studies with the implementations of this section.

9.1. Possibilistic Armstrong relation tool

Fig. 4 shows the GUI of our Possibilistic Armstrong Relation tool. It provides access for users to an implementation of Algorithm 3. The tool computes for an input set Σ of p-keys on a p-relation schema, an Armstrong p-relation for Σ.

We briefly outline the steps, which are illustrated on our running example in Fig. 4. In step 1 (top left of Fig. 4), the user enters the column names of the p-relation schema, the index of the bottom c-degree, together with the column names of the c-degree of a new p-key. Pressing the 'insert' button shows in step 2 (top right of Fig. 4) the p-relation schema together with all the p-keys that have been entered so far. Pressing the 'save' button in step 2, will invoke the execution of Algorithm 3, and produce several results.

Result 3 (bottom right of Fig. 4) shows the corresponding Armstrong p-relation for the input set of p-keys from step 2. Intermediate results of the computation are shown in the bottom left of Fig. 4. These include the anti-keys (Result 1) and the agree sets (Result 2).

The bottom right of Fig. 5 shows the GUI after executing Algorithm 3 on our running example. The output shows a p-relation that is Armstrong for the input, and has the same agree sets as our p-relation from Table 1.

9.2. Possibilistic key finder tool

Fig. 5 shows the GUI of our Possibilistic Key Finder tool. It provides access for users to an implementation of Algorithm 4. The tool computes for an input p-relation r a non-redundant cover for the set of p-keys that hold on r.

The input p-relation can be supplied in the form of a csv file (left of Fig. 5), or entered manually (right of Fig. 5) in the GUI. In both cases, the input is expected to have a column called p_degree which contains the index of the p-degree for each tuple. If such a column is not present, the implementation will assume that the p-degree of each tuple is β_1. The GUI also contains an input field called 'Maximum P Degree'. If no value is entered, then the implementation

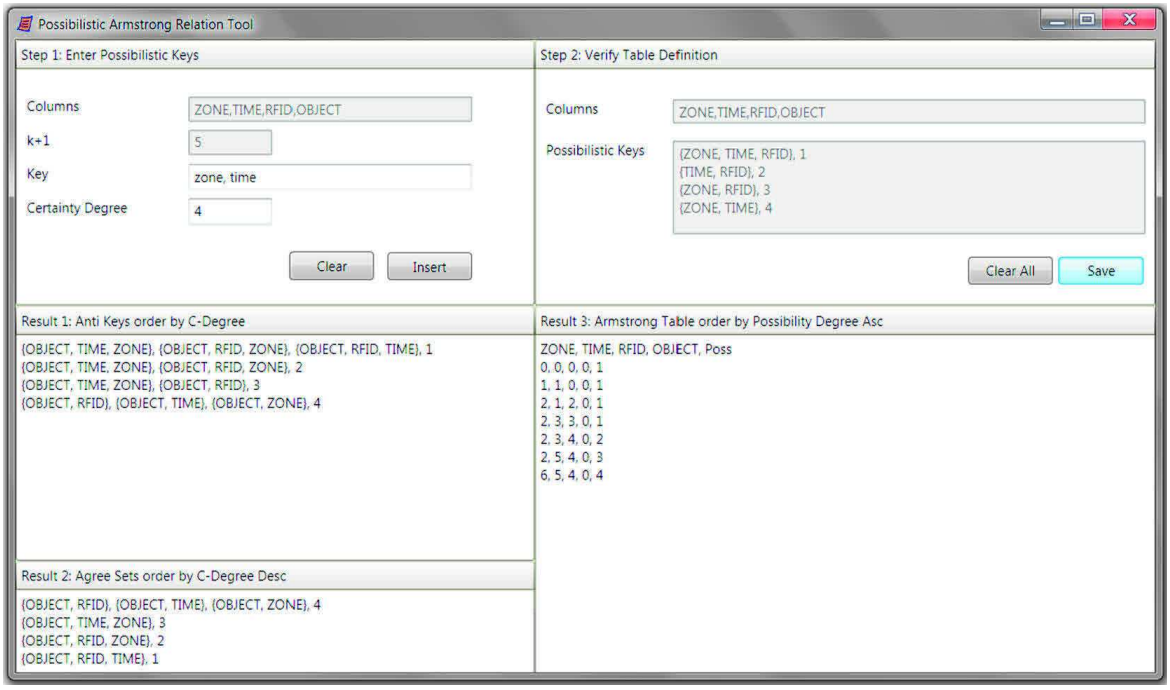


Fig. 4. Graphical user interface for visualization.

will pick the lowest p-degree (the one with the highest index) associated with any tuple in the input p-relation (and β_1 in case that no column of name p_degree exists). Using our notation from before, 'Maximum P Degree' denotes the index k. It should be noted that k can be higher than any p-degree found in the input p-relation.

The bottom of the GUI displays several results. It shows the time taken to compute (a possibly redundant) cover of the set of p-keys that hold on the input p-relation, the set of p-keys itself, the cardinality of this set, and the total number of key attributes in this set. It then shows the time taken to convert the possibly redundant cover into a non-redundant cover, the p-keys in the non-redundant cover, the cardinality of the non-redundant cover, and the total number of key attributes in the non-redundant cover.

The right of Fig. 5 shows the GUI after executing Algorithm 4 on our running example. The output shows exactly the non-redundant cover that we have worked with throughout the article.

10. Experimental results with visualization and discovery

This section will discuss the outcomes of detailed experimental studies we have conducted with the implementations of Algorithm 3 for the computation of Armstrong p-relations, and Algorithm 4 for the computation of the p-keys that hold on a given p-relation. They provide some insight on the actual size of Armstrong p-relations and the time it takes to compute them, as well as on the time to find the p-keys that hold on a given p-relation and the savings one can achieve by computing non-redundant covers.

10.1. Visualization

We will first explain our experiments with the computation of Armstrong p-relations using Algorithm 3. Here we distinguish between three different cases. Firstly, we will look at a worst-case in which the size of the output grows exponential in the size of the input, and thus Algorithm 3 also takes exponential time. Secondly, we will look at a good case in which the size of the output grows logarithmically in the size of the input. Finally, we will look at a random case in which the input set of p-keys is randomly generated.

10.1.1. Exponential case

The exponential case is taken from the proof of Theorem 10, where for positive integers n, $R = \{a_1, \ldots, a_{2n}\}$, $\mathcal{S} = \{\alpha_1, \alpha_2\}$ and

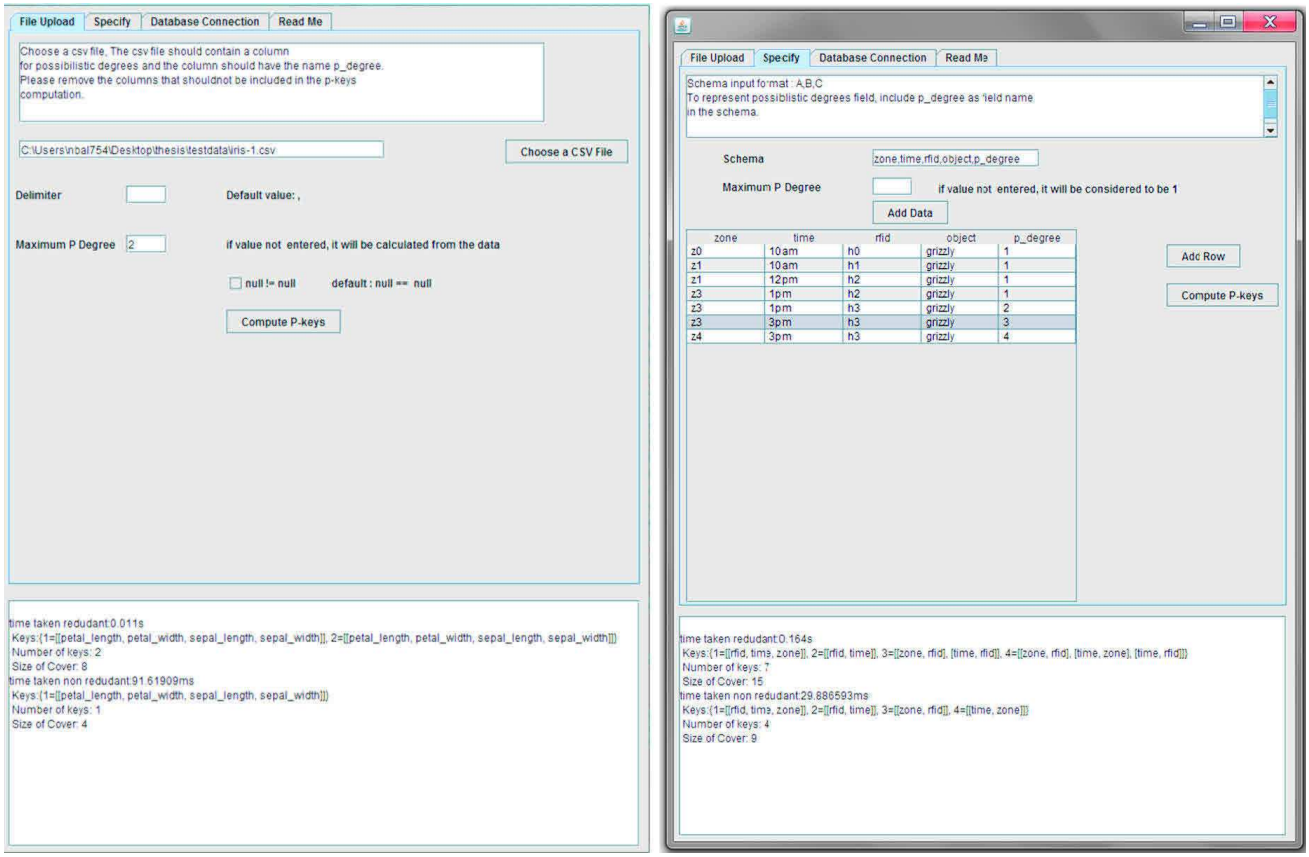


Fig. 5. Possibilistic key finder tool on csv file and direct input.

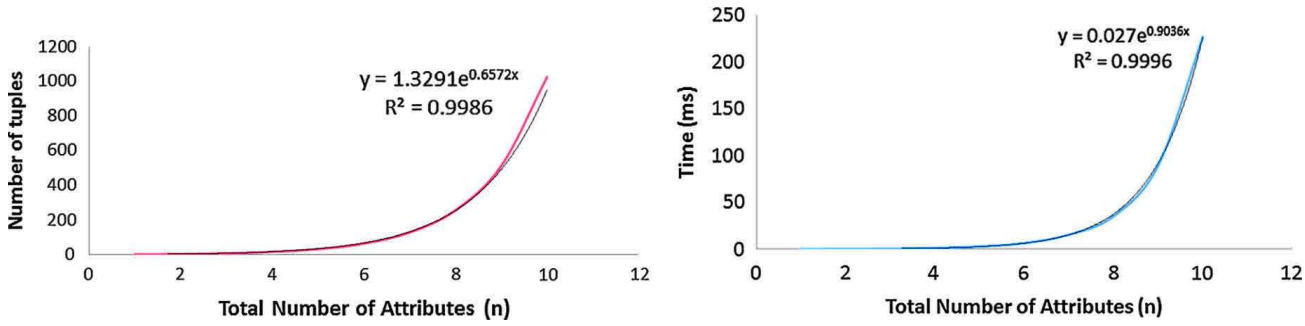


Fig. 6. Size of Armstrong p-relation, and time to compute it in exponential case.

$$\Sigma = \{(\{a_1, a_2\}, \beta_1), \ldots, (\{a_{2n-1}, a_{2n}\}, \beta_1)\}$$

forms the input of size $2 \cdot n$. The set Σ^{-1} consists of the 2^n anti-keys $\bigcup_{j=1}^{n} X_j$ where $X_j \in \{a_{2j-1}, a_{2j}\}$ for $j = 1, \ldots, n$.

Fig. 6 shows the size of the resulting Armstrong p-relations the case where $n = 1, \ldots, 10$, as well as the times (in ms) to compute them. For each the size and time, the actual experimental results are approximated (very closely) by an exponential function.

10.1.2. Logarithmic case

The logarithmic case is taken from the paragraph following Theorem 10, where for positive integers n, $R = \{a_1, \ldots, a_{2n}\}$, $\mathcal{S} = \{\alpha_1, \alpha_2\}$ and

$$\Sigma' = \{(\cup_{j=1}^{n} X_j, \beta_1) \mid X_j \in \{a_{2j-1}, a_{2j}\}\}$$

forms the input with 2^n p-keys. The set Σ^{-1} consists of the n anti-keys $R - \{a_{2j-1}, a_{2j}\}$ for $j = 1, \ldots, n$.

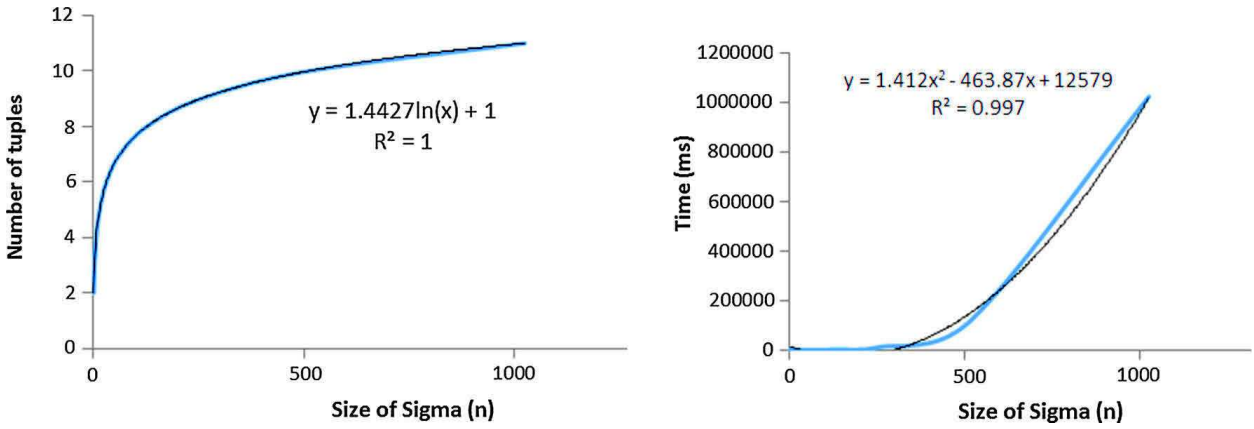


Fig. 7. Size of Armstrong p-relation, and time to compute it in logarithmic case.

Fig. 7 shows the size of the resulting Armstrong p-relations for the cases where $n = 1, \ldots, 10$, as well as the times (in ms) to compute them. For the size, the actual experimental results are approximated perfectly by a logarithmic function. For the time, the experimental results are closely approximated by a quadratic function.

10.1.3. Average case

The most interesting case may be the average-case behavior, that we mimicked by randomly creating for a given number $n = 2, \ldots, 15$ of attributes and a given bottom p-degree of $k + 1 = 2, \ldots, 20$, 500 sets Σ of p-keys, and then taking the average size (number of tuples) and time to compute an Armstrong p-relation for Σ by Algorithm 3. Each of the input sets Σ contained up to n^2 p-keys with two attributes on average. All experiments were conducted on an Intel Core i7-4600U 2.10 GHz machine that has 8 GB of RAM.

We were interested in the impact of the given number of attributes as well as the given number of available p-degrees for both measures, size and time. For that purpose, we display our results in two different figures.

Fig. 8 shows the size and time in terms of the growing number of attributes, for different fixed numbers of available p-degrees. The graphs suggest that both, the size of the output and the time to compute the output, grow low-degree polynomial in the input.

Fig. 9 shows the size and time in terms of the growing number of available p-degrees, for different fixed number of given attributes. As before, the graphs suggest that both, the size of the output and the time to compute the output, grow low-degree polynomial in the input.

10.2. Discovery

We will now discuss our experiments with the possibilistic key finder tool that implemented Algorithm 4. The algorithm was applied to possibilistic variants of seven real-world data sets plus one synthetic data set, called fd-reduced-30. The synthetic data set was generated with the dbtesma data generator [51,52]. All the other data sets have been obtained from the UCI Machine Learning Repository [42]. Table 4 shows the basic characteristics of the various data sets. They comprise a reasonable range of column and row numbers. It should be pointed out here that big data sets, with say 50 columns and 1 million rows, resist state-of-the-art data dependency mining algorithms [51,52]. As indicated previously, the data sets are not possibilistic, so they were made possibilistic by randomly assigning values between 1 and k for each tuple in the data set. For our experiments, k was either 1, 5, 10, or 15. All the experiments were performed on a Windows 7, 64 bit machine with four 3.30 GHz processors, 8 GB RAM, 930 GB hard disk and 64bit Java 7.0.

Apart from discussing various measures on the basis of the given data sets, we will also briefly discuss the semantics that is associated with null marker occurrences, as well as a parallel approach towards the discovery of p-keys from the data sets. Note that it is not our aim to judge the meaningfulness of the discovered keys. One cannot determine the meaningfulness of a constraint from whether it holds or does not hold on a given data set. Such judgment would require domain expertise. In the possibilistic case, it is even more difficult to comment on the meaningfulness since

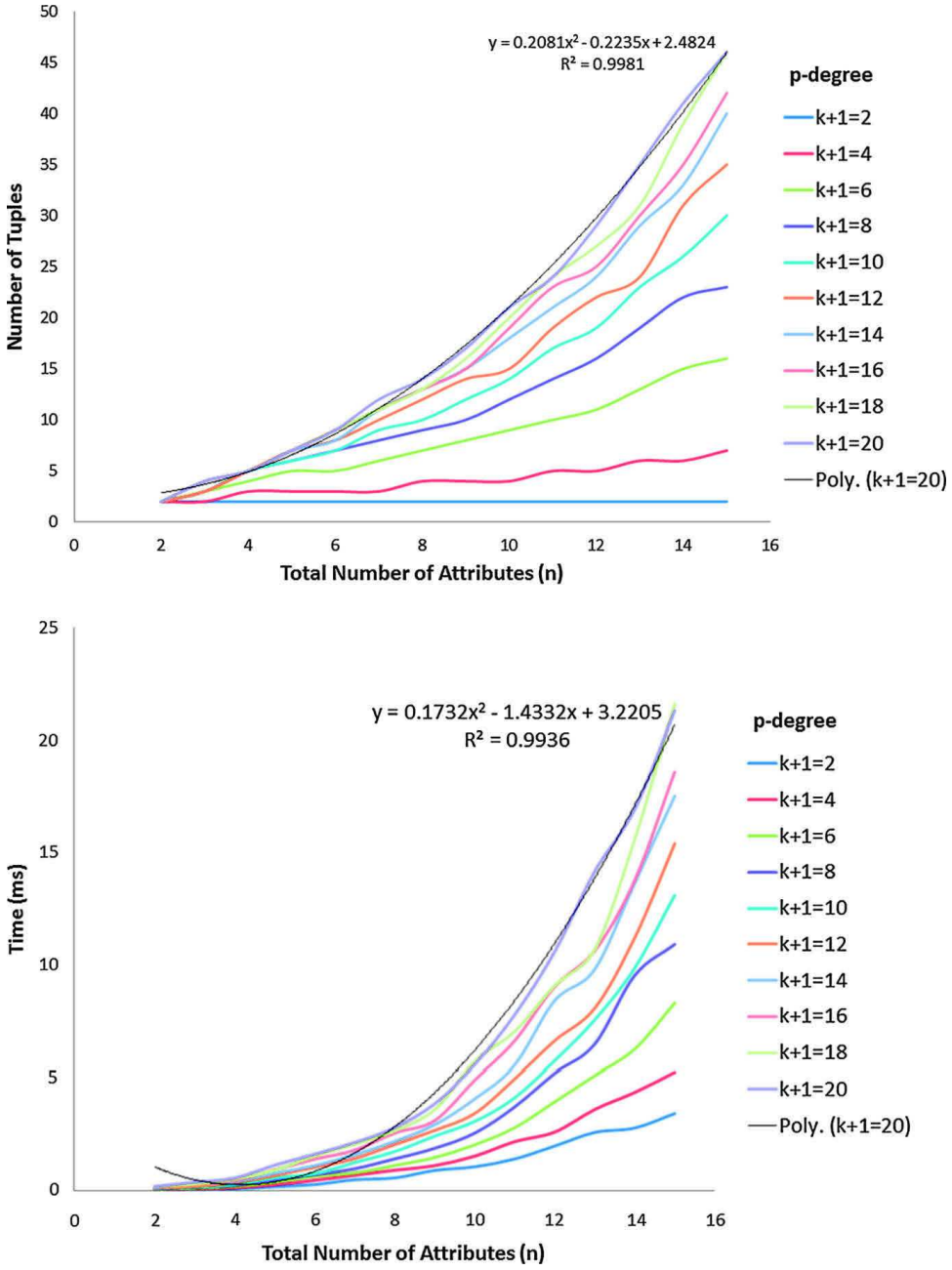


Fig. 8. Average size of p-Armstrong relation with growing number of attributes and fixed p-degree.

the concept of p-relations and p-constraints are new, so even domain expertise would not really be available. Just like previous research [51,52] we therefore focus on the performance of our discovery algorithms.

10.2.1. Results of experiments with given data sets

We will now present our findings of applying Algorithm 4 to the possibilistic variants of the given data sets. Several of the data sets contained duplicate tuples, i.e. tuples which had matching values on all the underlying attributes. In such cases, no key can exist, and we therefore only kept one of the tuples for which duplicate tuples exist. The efficiency of the key finder tool is measured in terms of the computation time in seconds, the number of p-keys found, and the total number of attributes that occur in the output. We provide these measures for each data set, and each value of $k = 1, 5, 10$, and 15, and distinguish between the output (RC) derived directly from merging the p-keys found in each possible world, and the non-redundant cover (NRC) derived from the previous output by removing any redundant p-keys. For each data set, and for each value of k, the experiments were repeated 5 times and the

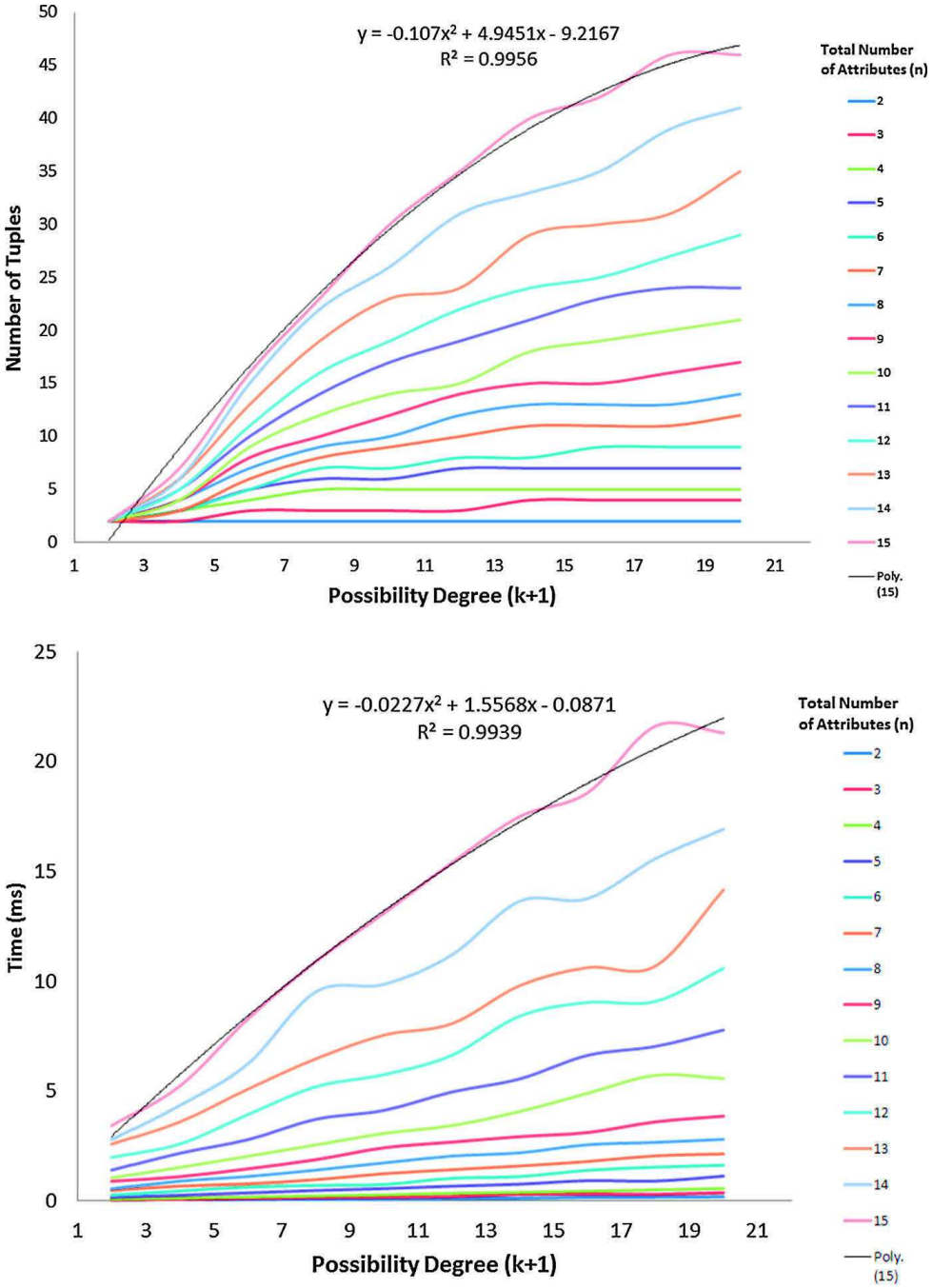


Fig. 9. Average size of p-Armstrong relation with growing p-degree and fixed number of attributes.

Table 4
Basic characteristics of data sets used in discovery experiments.

Data set	Columns (#)	Rows (#)
iris	5	150
Abalone	9	4,177
Breast-Cancer	11	699
Adult	14	48842
Letter	17	20000
Hepatitis	20	155
Horse	27	368
fd-reduced-30	30	250000

Table 5
Efficiency measures for various data sets.

Dataset		k = 1			k = 5		
		time [s]	p-keys (♯)	size	time [s]	p-keys (♯)	size
Iris	RC	0.14	1	4	0.06	7.60	24.80
	NRC	0.015	1	4	8.18	4.80	12.20
Breast-Cancer	RC	0.58	2	10	0.59	14.40	47.20
	NRC	0.01	2	10	2.53	11.20	38.20
Abalone	RC	52.22	29	129	55.21	159.00	638.00
	NRC	0.03	29	129	3.66	86.80	351.60
Adult	RC	1791.53	2	20	1770.88	14.00	112.20
	NRC	0.02	2	20	7.49	11.20	88.20
Letter	RC	784.51	1	16	821.64	5.20	82.60
	NRC	0.02	1	16	7.64	1.80	28.20
Hepatitis	RC	3.04	104	499	10.14	592.60	2703.20
	NRC	0.05	104	499	25.99	456.00	2149.00
Horse	RC	1.73	53	248	27454.07	5534.75	29169.50
	NRC	0.03	53	248	1787.38	5034.25	26812.50
fd-reduced-30	RC	ML	ML	ML	ML	ML	ML
	NRC	ML	ML	ML	ML	ML	ML

Table 6
Efficiency measures for various data sets.

Dataset		k = 10			k = 15		
		time [s]	p-keys (♯)	size	time [s]	p-keys (♯)	size
Iris	RC	0.06	15.80	50.40	0.06	25.80	76.40
	NRC	4.98	6.40	17.80	7.94	8.00	20.60
Breast-Cancer	RC	0.54	18.00	62.80	0.55	40.40	134.80
	NRC	4.08	8.80	30.00	6.35	16.80	56.60
Abalone	RC	55.39	329.60	1289.40	53.39	483.20	1870.20
	NRC	8.38	115.00	448.00	11.87	134.60	512.80
Adult	RC	1814.78	43.60	310.40	1805.73	72.40	470.40
	NRC	7.08	33.20	226.80	9.80	51.60	316.60
Letter	RC	808.21	14.80	217.20	801.67	32.80	435.40
	NRC	6.48	7.00	92.80	4.02	21.00	246.80
Hepatitis	RC	39.31	1472.40	7164.60	43.02	2194.00	10614.60
	NRC	112.27	883.40	4501.20	190.08	1072.20	5360.80
Horse	RC	36334.03	12138.20	64089.00	126486.77	24683.00	137663.20
	NRC	8780.14	8629.40	46550.20	26126.39	12599.40	71332.00
fd-reduced-30	RC	ML	ML	ML	ML	ML	ML
	NRC	ML	ML	ML	ML	ML	ML

averages were computed for time taken, the number of p-keys discovered, and the size of the output, for RC and NRC, respectively.

Table 5 shows all these measures for the k-values 1 and 5, and Table 6 shows the measures for the k-values of 10 and 15. The value ML refers to the event that the experiment exceeded the memory limit of 4 GB. For the non-redundant cover (RC), computation time refers to the time taken to compute the non-redundant cover (NRC) from the redundant cover (RC).

The mining work the fastest on the smallest data set, *Iris*, which also has the least number of p-keys found. The *Abalone* data set has significantly more rows than iris and breast-cancer. The values of k have limited impact on the computation times for this data set. For the *breast-cancer* data set the computation time is faster than for abalone. For this data set, the value of k does again not affect the computation time much. The *adult* data set more rows and columns than the previous data sets. In this case, the computation time is also much higher than that on the previous data sets. The value of k is seen to have a considerable effect on the efficiency. The *letter* data set has more columns than the above mentioned data sets, but has only 20,000 rows while adult has 48,842 rows. For this data set, p-key mining takes less time than that of the adult data set, even though there are more columns. The *hepatitis* data set has considerably more columns (20) and few rows. In this case, the computation time increases proportionally with the increases of k. The *horse* data set has a high number of columns (27) but few rows. In this case the computation

time increases substantially with increases in k. The synthetic data set *fd-reduced-30* has a high number of rows and columns. In this scenario, the algorithm does not return an output as the data set does not fit into memory.

Regarding our experiments we make the following observations. The algorithm performs well on data sets which have a modest number of rows or columns. Data sets that have a larger number of columns (e.g. 30) and a larger number of rows (e.g. 250,000) are difficult to handle. The savings achieved by removing redundant p-keys can be significant (and the larger data sets are the more savings will be made in terms of the time required to validate the p-keys). In addition, the removal of redundant p-keys takes little time in comparison to the computation of the p-keys. The time taken to remove redundant p-keys increases with the value of k. This can be attributed to the fact that the number of p-keys increases as the value of k grows. For $k = 1$, the possibilistic relation corresponds to data coming from a relational table and the p-key cover represents the minimal candidate keys of the corresponding relational table. In this case there are no redundant p-keys. For data sets with more columns and few tuples, such as horse and hepatitis, the performance degrades as the k value increases. This is attributed to the disagree sets being larger on average and hence requiring more time to compute the hypergraph transversals. Fig. 10a shows that for the *hepatitis* data set, hypergraph transversal consumes more time than the time taken to obtain the minimal disagree sets. For data sets with more columns and few rows, such as horse and hepatitis, the size of the cover is larger than that for data sets with few columns and more rows, such as adult and letter. This may be attributed to the fact that with fewer tuples it takes more columns to generate non-empty intersections for the participating larger disagree sets. For larger numbers of rows, for example in the letter and adult data sets, it can be seen from Fig. 10c and Fig. 10b that the main contribution to computation time comes from tuple comparisons during the computation of the disagree sets. As the value of k increases, it is expected that the number of keys increase. This behavior is attributed to the increase in possible worlds when k grows. The algorithm conforms to this behavior.

10.2.2. Null semantics

The basis for our experiments were data sets from the real world. It is therefore natural for null markers, denoted by `null`, to occur. For two tuples t and t' with null marker occurrences in column a, two semantics are possible: $t[a] == t'[a]$ or $t[a] \neq t'[a]$. Since these semantics result in the discovery of different p-keys, the user can choose which semantics is adopted by the implementation of our algorithm. The semantics where `null` $==$ `null` results in more p-keys, as shown in Fig. 11.

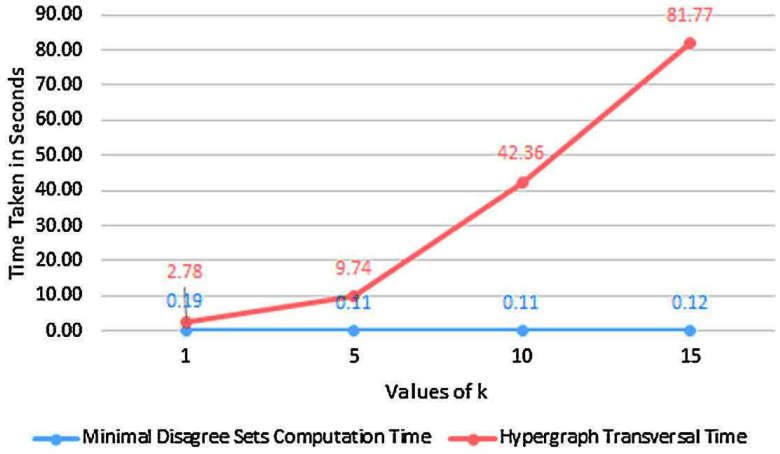
10.2.3. Parallelization

So far, our experiments have been conducted with a sequential implementation of Algorithm 4. Here, the classical key discovery algorithm is applied to each possible world, and the results are merged by assigning corresponding c-degrees, and then removing any redundant p-keys from the result. As the possible worlds are nested, all tuples of a smaller world are included in a bigger world. In particular, it is unnecessary to re-compute the disagree sets of tuples for which this has already been done beforehand. As an alternative approach, one may apply the classical key discovery algorithm to the possible worlds in parallel. The disadvantage is that for each tuple pair that is contained in several possible worlds, their disagree sets must be computed for each of these possible worlds. Table 7 shows the average number of p-keys and the size of the redundant p-key cover on several data sets. These numbers deviate slightly between the sequential and parallel approach, because of the different assignments of p-degrees to tuples. The computation time for both approaches is compared in Fig. 12.

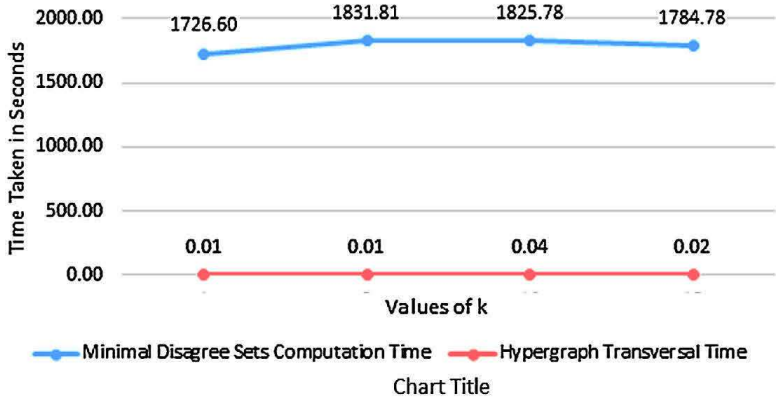
For data sets with few rows and many columns, such as hepatitis, the computation time for the hypergraph transversal is high in comparison to the time for computing minimal disagree sets, as shown in Fig. 10a. In such cases, the time taken for redundant disagree set computations is negligible compared to the hypergraph transversal times. Hence, data sets with few rows but many columns show a better performance in the parallel approach. However, when the value of k increases, then the number of redundant disagree set computations increases as well, and will become a bottleneck in the parallel approach. Indeed, Fig. 12 shows for most data sets that the sequential approach becomes more efficient than the parallel approach from a value of $k = 5$ onwards.

11. Conclusion and future work

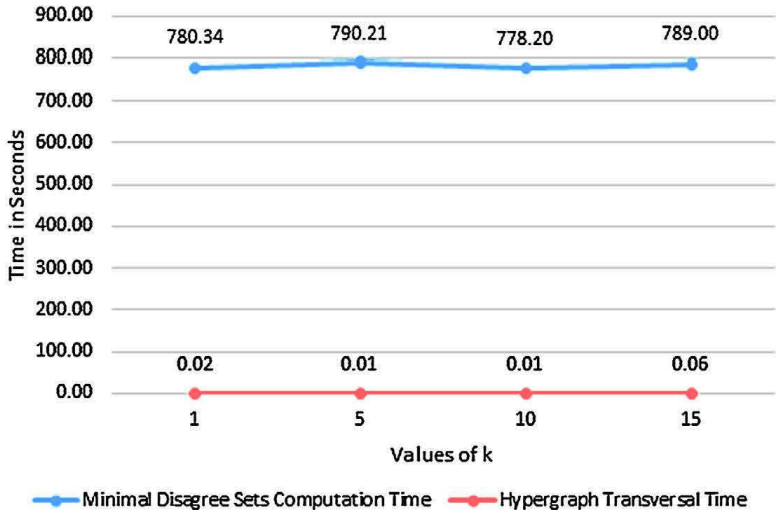
Possibilistic keys have been introduced to efficiently identify tuples in uncertain data. Uncertainty is modeled qualitatively by applying the AI framework of possibilistic logic to the fundamental database concept of keys. Tools



(a) Hepatitis Dataset



(b) Adult Dataset



(c) Letter Dataset

Fig. 10. Comparison of minimal disagree set computation times and hypergraph transversal time for various data sets.

were established to efficiently reason about possibilistic keys, and applications of p-keys in constraint maintenance, data cleaning, and query processing were outlined in detail. It was characterized which non-redundant families of p-keys attain maximum cardinality, a result that enables data engineers to simulate data processing under heavily

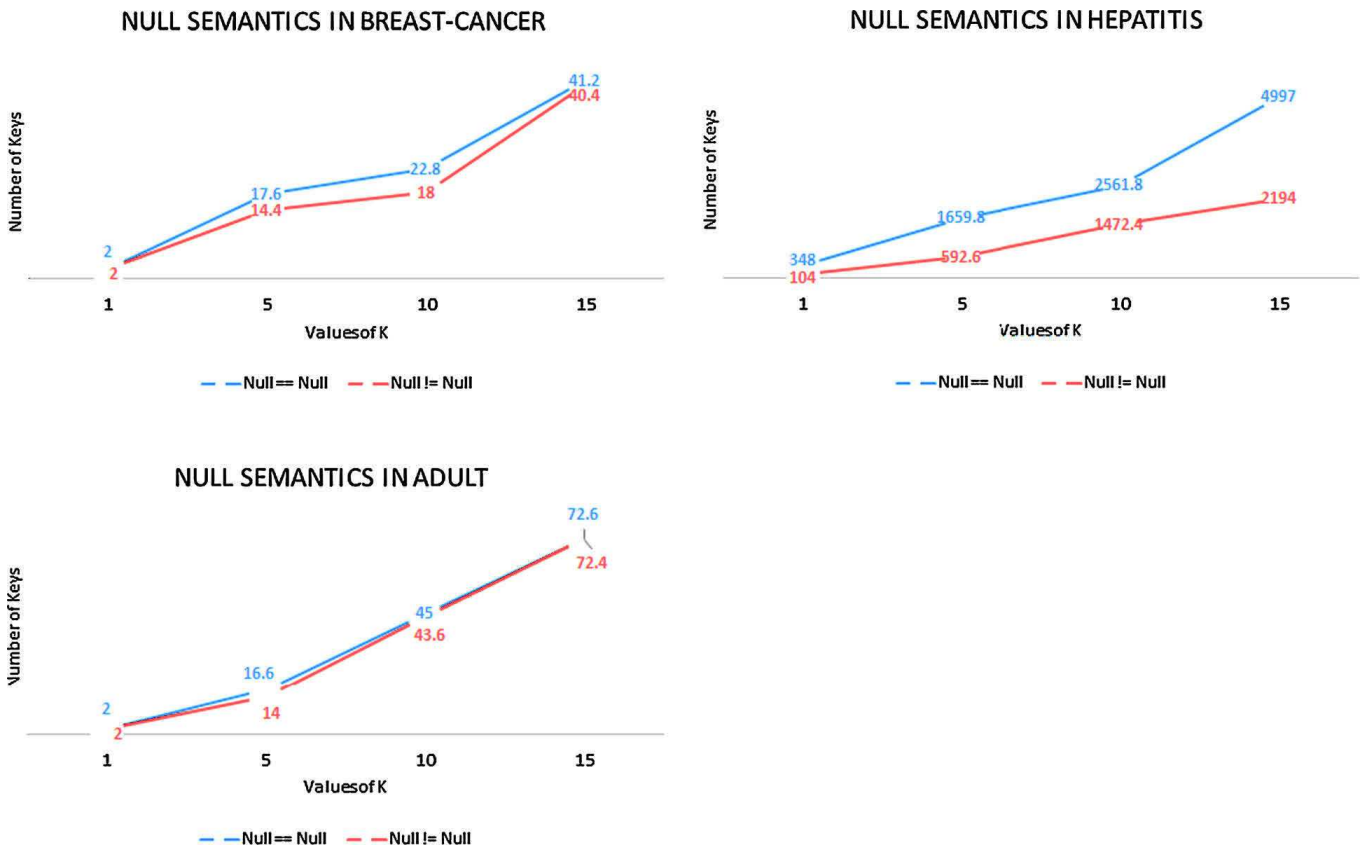


Fig. 11. Impact of null semantics on numbers of P-keys discovered.

Table 7
Number of P-keys and size of cover in sequential and parallel approaches for data sets.

Dataset	k = 1			k = 5		k = 10		k = 15	
		p-keys (♯)	size	p-keys (♯)	size	p-keys (♯)	size	p-keys (♯)	size
Iris	Seq	1	4	7.6	24.8	15.8	50.4	25.8	76.4
	Par	1	4	8.6	27.2	18.4	52.8	24.2	73.2
Abalone	Seq	29	129	159	638	329.6	1289.4	483.2	1870.2
	Par	29	129	158.2	633.2	313	1210.8	481.6	1883.2
Breast-Cancer	Seq	2	10	14.4	47.2	18	62.8	40.4	134.8
	Par	2	10	12.8	42.8	20.6	76	46	176
Letter	Seq	1	16	5.20	82.60	14.80	217.20	32.80	435.40
	Par	1	16	5	79.6	12.2	183	23.8	340.4
Hepatitis	Seq	104	499	592.6	2703.2	1472.4	7164.6	2194	10614.6
	Par	104	499	549.6	2483	1673.8	8170.6	2466.8	12357.6

constrained databases. Tools for the visualization of abstract sets of p-keys and for the discovery of p-keys from given p-relations were established. Together, these two tools can be used by data engineers to help acquire the possibilistic keys that are semantically meaningful for a given application domain. Visualization and discovery were implemented in prototype systems, and detailed experimental results with both systems rounded off our theoretical findings on the inherent exponential complexity of these problems.

There are different directions that warrant future research. While keys constitute the simplest yet most important class of constraints, classes such as foreign keys and functional dependencies deserve particular attention as well. In fact, possibilistic functional dependencies were introduced in [44] and their main application to relational database schema design discussed in [45]. The class of foreign keys or, more generally, inclusion dependencies is an interesting next target. Since these referential constraints are defined with respect to different relation schemata, different semantics can be defined. On the basis of our algorithms for computing Armstrong p-relations and discovering p-keys, it

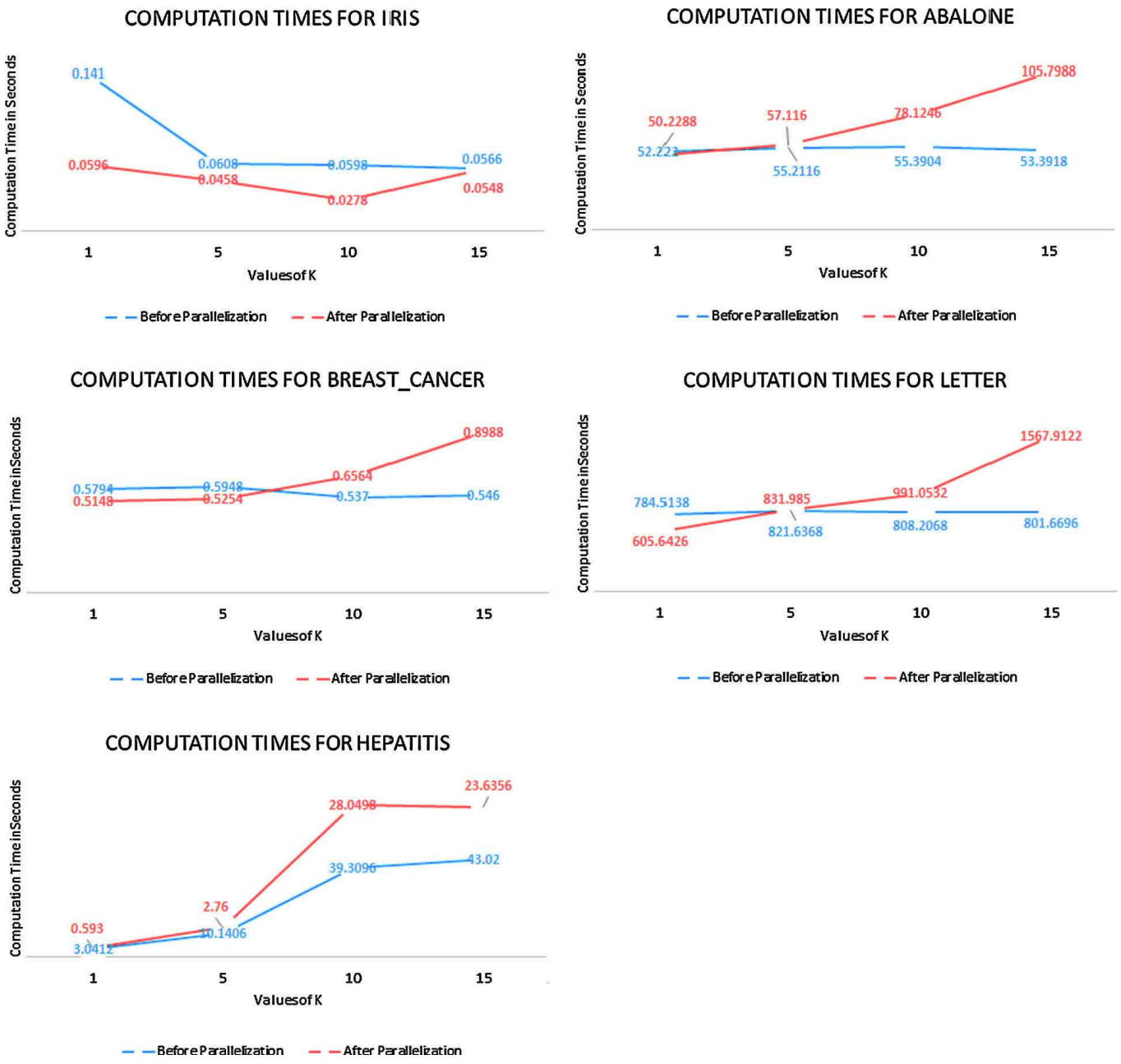


Fig. 12. Sequential versus parallel approach.

would be interesting to conduct empirical studies about the usefulness of our acquisition framework for the identification of semantically meaningful p-keys. Another interesting application is consistent query answering [41]. Here, a given data set may violate a given set of constraints. Instead of repairing the data set with some minimal effort, consistent query answering only returns those answers that are present in all repairs. The possibilistic framework is interesting because consistent query answers would be answers attributed with the minimum p-degree across all repairs. Possibilistic data cleaning has already been mentioned as another application area, and first results have been announced in [35]. It would be interesting to investigate possibilistic variants of data dependencies, in particular keys, in other data models. For example, keys in SQL [34,39,40] or XML [26].

Acknowledgement

The authors would like to thank a reviewer whose comments led to a clearer positioning of the contributions of the research. In particular, the model of uncertainty is primarily intended for the application areas that have been

identified, including database design. We remark that the model of uncertainty is only of limited use in the area of query processing, due to its limited expressiveness.

This research is partially supported by the Marsden Fund Council (grant no 3701176) from New Zealand Government funding.

References

[1] W.W. Armstrong, Dependency structures of data base relationships, in: IFIP Congress, pp. 580–583.

[2] C. Beeri, M. Dowd, R. Fagin, R. Statman, On the structure of Armstrong relations for functional dependencies, J. ACM 31 (1984) 30–46.

[3] O. Benjelloun, A.D. Sarma, A.Y. Halevy, M. Theobald, J. Widom, Databases with uncertainty and lineage, VLDB J. 17 (2008) 243–264.

[4] T. Bläsius, T. Friedrich, M. Schirneck, The parameterized complexity of dependency detection in relational databases, in: 11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24–26, 2016, Aarhus, Denmark, 2016, pp. 6:1–6:13.

[5] P. Bosc, O. Pivert, About projection–selection–join queries addressed to possibilistic relational databases, IEEE Trans. Fuzzy Syst. 13 (2005) 124–139.

[6] P. Brown, S. Link, Probabilistic keys for data quality management, in: Proceedings of the 27th International Conference on Advanced Information Systems Engineering, CAiSE 2015, Stockholm, Sweden, June 8–12, 2015, 2015, pp. 118–132.

[7] P. Brown, S. Link, Probabilistic keys, IEEE Trans. Knowl. Data Eng. 29 (2017) 670–682.

[8] E.F. Codd, A relational model of data for large shared data banks, Commun. ACM 13 (1970) 377–387.

[9] N.N. Dalvi, D. Suciu, Management of probabilistic data: foundations and challenges, in: PODS, 2007, pp. 1–12.

[10] J. Demetrovics, G.O.H. Katona, Extremal combinatorial problems of database models, in: MFDBS, 1987, pp. 99–127.

[11] J. Diederich, J. Milton, New methods and fast algorithms for database normalization, ACM Trans. Database Syst. 13 (1988) 339–365.

[12] R.G. Downey, M.R. Fellows, Parameterized Complexity, Monographs in Computer Science, Springer, 1999.

[13] R.G. Downey, M.R. Fellows, Fundamentals of Parameterized Complexity, Texts in Computer Science, Springer, 2013.

[14] D. Dubois, H. Fargier, H. Prade, Refinements of the maximin approach to decision-making in a fuzzy environment, Fuzzy Sets Syst. 81 (1996) 103–122.

[15] D. Dubois, H. Prade, Possibility theory, in: R.A. Meyers (Ed.), Computational Complexity, Springer, New York, 2012, pp. 2240–2252.

[16] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, SIAM J. Comput. 24 (1995) 1278–1304.

[17] K. Engel, Sperner Theory, Cambridge Uni Press, 1997.

[18] R. Fagin, Horn clauses and database dependencies, J. ACM 29 (1982) 952–985.

[19] D. Geiger, A. Paz, J. Pearl, Axioms and algorithms for inferences involving probabilistic independence, Inf. Comput. 91 (1991) 128–141.

[20] T.J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proceedings of the Twenty-Sixth ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, Beijing, China, June 11–13, 2007, 2007, pp. 31–40.

[21] N. Hall, H. Köhler, S. Link, H. Prade, X. Zhou, Cardinality constraints on qualitatively uncertain data, Data Knowl. Eng. 99 (2015) 126–150.

[22] M. Hannula, S. Link, Automated reasoning about key sets, in: D. Galmiche, S. Schulz, R. Sebastiani (Eds.), Proceedings of the 9th International Joint Conference on Automated Reasoning, IJCAR 2018, held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14–17, 2018, in: Lecture Notes in Computer Science, vol. 10900, Springer, 2018, pp. 47–63.

[23] S. Hartmann, M. Kirchberg, H. Koehler, U. Leck, S. Link, Extremal combinatorics of SQL keys, in: A. Mashkoor, Q. Wang, B. Thalheim (Eds.), Models: Concepts, Theory, Logic, Reasoning and Semantics—Essays Dedicated to Klaus-Dieter Schewe on the Occasion of His 60th Birthday, College Publications, 2018, pp. 75–91.

[24] S. Hartmann, M. Kirchberg, S. Link, Design by example for SQL table definitions with functional dependencies, VLDB J. 21 (2012) 121–144.

[25] S. Hartmann, U. Leck, S. Link, On Codd families of keys over incomplete relations, Comput. J. 54 (2011) 1166–1180.

[26] S. Hartmann, S. Link, Efficient reasoning about a robust XML key fragment, ACM Trans. Database Syst. 34 (2009).

[27] S. Hartmann, S. Link, Numerical constraints on XML data, Inf. Comput. 208 (2010) 521–544.

[28] A. Heise, Jorge-Arnulfo, Quiane-Ruiz, Z. Abedjan, A. Jentzsch, F. Naumann, Scalable discovery of unique column combinations, Proc. VLDB Endow. 7 (2013) 301–312.

[29] A.K. Jha, V. Rastogi, D. Suciu, Query evaluation with soft-key constraints, in: PODS, 2008, pp. 119–128.

[30] A.K. Jha, D. Suciu, Probabilistic databases with markoviews, Proc. VLDB Endow. 5 (2012) 1160–1171.

[31] A. Kiss, λ-decomposition of fuzzy relational data, Ann. Univ. Sci. Bp. 12 (1991) 133–142.

[32] D. Kleitman, On an extremal property of antichains in partial orders. The Lym property and some of its implications and applications, in: M. Hall Jr., J. Lint (Eds.), Combinatorics, in: NATO Advanced Study Institutes Series, vol. 16, Springer, Netherlands, 1975, pp. 277–290.

[33] H. Köhler, U. Leck, S. Link, H. Prade, Logical foundations of possibilistic keys, in: Proceedings of the 14th European Conference on Logics in Artificial Intelligence, JELIA 2014, Funchal, Madeira, Portugal, September 24–26, 2014, 2014, pp. 181–195.

[34] H. Köhler, U. Leck, S. Link, X. Zhou, Possible and certain keys for SQL, VLDB J. 25 (2016) 571–596.

[35] H. Köhler, S. Link, Qualitative cleaning for uncertain data, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, Indiana, USA, October 24–28, 2016, 2016, pp. 2269–2274.

[36] H. Köhler, S. Link, SQL schema design: foundations, normal forms, and normalization, in: Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26–July 01, 2016, 2016, pp. 267–279.

[37] H. Köhler, S. Link, SQL schema design: foundations, normal forms, and normalization, Inf. Syst. 76 (2018) 88–113.

[38] H. Köhler, S. Link, H. Prade, X. Zhou, Cardinality constraints for uncertain data, in: Proceedings of the 33rd International Conference on Conceptual Modeling, ER 2014, Atlanta, GA, USA, October 27–29, 2014, 2014, pp. 108–121.

[39] H. Köhler, S. Link, X. Zhou, Possible and certain SQL keys, Proc. VLDB Endow. 8 (2015) 1118–1129.ä

[40] H. Köhler, S. Link, X. Zhou, Discovering meaningful certain keys from incomplete and inconsistent relations, IEEE Data Eng. Bull. 39 (2016) 21–37.

[41] P. Koutris, J. Wijsen, Consistent query answering for primary keys, SIGMOD Rec. 45 (2016) 15–22.

[42] M. Lichman, UCI Machine Learning Repository, 2013.

[43] S. Link, Old keys that open new doors, in: F. Ferrarotti, S. Woltran (Eds.), Proceedings of the 10th International Symposium on Foundations of Information and Knowledge Systems, FoIKS 2018, Budapest, Hungary, May 14–18, 2018, in: Lecture Notes in Computer Science, vol. 10833, Springer, 2018, pp. 3–13.

[44] S. Link, H. Prade, Possibilistic functional dependencies and their relationship to possibility theory, IEEE Trans. Fuzzy Syst. 24 (2016) 757–763.

[45] S. Link, H. Prade, Relational database schema design for uncertain data, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, Indiana, USA, October 24–28, 2016, 2016, pp. 1211–1220.

[46] H. Mannila, K. Räihä, Dependency inference, in: Proceedings of 13th International Conference on Very Large Data Bases, VLDB'87, Brighton, England, September 1–4, 1987, 1987, pp. 155–158.

[47] H. Mannila, K.J. Räihä, Design by example: an application of Armstrong relations, J. Comput. Syst. Sci. 33 (1986) 126–141.

[48] H. Mannila, K.J. Räihä, Algorithms for inferring functional dependencies from relations, Data Knowl. Eng. 12 (1994) 83–99.

[49] F.D. Marchi, J. Petit, Semantic sampling of existing databases through informative Armstrong databases, Inf. Syst. 32 (2007) 446–457.

[50] M. Niepert, M. Gyssens, B. Sayrafi, D.V. Gucht, On the conditional independence implication problem: a lattice-theoretic approach, Artif. Intell. 202 (2013) 29–51.

[51] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J. Rudolph, M. Schönberg, J. Zwiener, F. Naumann, Functional dependency discovery: an experimental evaluation of seven algorithms, Proc. VLDB Endow. 8 (2015) 1082–1093.

[52] T. Papenbrock, F. Naumann, A hybrid approach to functional dependency discovery, in: Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26–July 01, 2016, 2016, pp. 821–833.

[53] O. Pivert, H. Prade, A certainty-based model for uncertain databases, IEEE Trans. Fuzzy Syst. 23 (2015) 1181–1196.

[54] O. Pivert, H. Prade, Possibilistic conditional tables, in: M. Gyssens, G.R. Simari (Eds.), Proceedings of the 9th International Symposium on Foundations of Information and Knowledge Systems, FoIKS 2016, Linz, Austria, March 7–11, 2016, in: Lecture Notes in Computer Science, vol. 9616, Springer, 2016, pp. 42–61.

[55] O. Pivert, H. Prade, Handling uncertainty in relational databases with possibility theory — A survey of different modelings, in: D. Ciucci, G. Pasi, B. Vantaggi (Eds.), Proceedings of the 12th International Conference on Scalable Uncertainty Management, SUM 2018, Milan, Italy, October 3–5, 2018, in: Lecture Notes in Computer Science, vol. 11142, Springer, 2018, pp. 396–404.

[56] T. Roblot, M. Hannula, S. Link, Probabilistic cardinality constraints, VLDB J. 27 (6) (2018) 771–795.

[57] T. Roblot, S. Link, Probabilistic cardinality constraints, in: P. Johannesson, M. Lee, S.W. Liddle, A.L. Opdahl, O.P. López (Eds.), Proceedings of the 34th International Conference on Conceptual Modeling, ER 2015, Stockholm, Sweden, October 19–22, 2015, in: Lecture Notes in Computer Science, vol. 9381, Springer, 2015, pp. 214–228.

[58] T.K. Roblot, S. Link, Urd: a data summarization tool for the acquisition of meaningful cardinality constraints with probabilistic intervals, in: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19–22, 2017, IEEE Computer Society, San Diego, CA, USA, 2017, pp. 1379–1380.

[59] A. Sali, K.D. Schewe, Keys and Armstrong databases in trees with restructuring, Acta Cybern. 18 (2008) 529–556.

[60] E. Sperner, Ein Satz über Untermengen einer endlichen Menge, Math. Z. 27 (1928) 544–548.

[61] D. Suciu, D. Olteanu, C. Ré, C. Koch, Probabilistic databases, in: Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2011.

[62] B. Thalheim, On semantic issues connected with keys in relational databases permitting null values, Elektron. Inf.verarb. Kybern. 25 (1989) 11–20.

[63] B. Thalheim, Dependencies in Relational Databases, Teubner, 1991.

[64] Z. Wei, S. Link DataProf, Semantic profiling for iterative data cleansing and business rule acquisition, in: G. Das, C.M. Jermaine, P.A. Bernstein (Eds.), Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10–15, 2018, ACM, 2018, pp. 1793–1796.

[65] Z. Wei, S. Link, J. Liu, Contextual keys, in: H.C. Mayr, G. Guizzardi, H. Ma, O. Pastor (Eds.), Proceedings of the 36th International Conference on Conceptual Modeling, ER 2017, Valencia, Spain, November 6–9, 2017, in: Lecture Notes in Computer Science, vol. 10650, Springer, 2017, pp. 266–279.

[66] S. Zadrozny, G.D. Tré, R.M.M.D. Caluwe, J. Kacprzyk, An overview of fuzzy approaches to flexible database querying, in: Database Tech-nologies: Concepts, Methodologies, Tools, and Applications (4 vols.), 2009, pp. 135–156.ää