Preference Functional Dependencies for Managing Choices*

Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong
wilfred@cse.ust.hk

Abstract. The notion of user preference in database modeling has recently received much attention in advanced applications, such as personalization of e-services, since it captures the human wishes on querying and managing data. The paradigm of preference-driven choices in the real world requires new semantic constraints in modelling. In this paper, we assume preference constraints can be defined over data domains and thus the assumption gives rise to preference relations as a special case of ordered relations over schemas consisting of the preference, preference-dependent and preference-independent attributes. We demonstrate that Lexicographically Ordered Functional Dependencies (LOFDs) can be employed to maintain the consistency of preference semantics embedded in preference database, since prioritized multiple preferences can be represented. We thus define a useful semantic constraint in terms of a set of LOFDs, called Preference Functional Dependencies (PFDs), in order to capture the semantics of the preference ranked data. We exhibit a sound and complete axiom system for PFDs, whose implication problem is shown to be decidable in polynomial-time. We also confirm the existence of Armstrong preference relations for PFDs, a fundamental result related to the practical use of PFDs in database design.

1 Introduction

Preference is natural in real world. When searching for items to be purchased over the internet, customer wishes and preferences are important, since they relate to managing of goods and developing selling tactics of a business corporation. It is a frustrating experience if one encounters many times some query results like "no match" or "sorry, try again with some other choices". The traditional constraints like functional dependencies model an exact world where the semantics capture the hard fact only, rather than the constraints specified by a list of user preferences. We share the same idea with [5, 6] that the fundamental nature of different preferences in the form of "I like A better than B" can be modelled by a set of orderings defined over data.

* This work is partially supported by RGC CERG under grant number HKUST6185/03E.

We assume here that the data domains of the relational data model are linearly ordered and call the extended model the ordered relational model [11, 12]. Within the model, we have formalised the notion of *Lexicographically Ordered Functional Dependencies* (LOFDs) being satisfied in an ordered database and tackled their implication problem in [10]. The semantics of LOFDs are defined by means of the *lexicographical orderings* on the domains associated with the involved attributes, which resemble the way that words are arranged in a dictionary. For example, the LOFD $\langle POST\_RANK, WORKING\_YEARS\rangle \rightsquigarrow SALARY$ can capture the preference constraint that in a company the higher salary is preferably given for those employees who are in higher post rank, or if they are at the same rank but are more experienced. Here there are two preferences are involved, namely, $POST\_RANK$ and $WORKING\_YEARS$; the former preference is considered more important than the latter. The left-hand side attributes capture the notion of *prioritized preference* (cf. Definition 6 in [5]), i.e. $POST\_RANK$ is considered more important than $WORKING\_YEARS$ and $WORKING\_YEARS$ is respected only where $POST\_RANK$ is the same.

In this paper we apply LOFDs in the context of preference relations and view preference relations as a special class of ordered relations. The underlying idea is that *preference* is inherent to the ordering relationship between the data projected onto the *preference* (e.g. $POST\_RANK$ and $WORKING\_YEARS$) and *preference-dependent* (e.g. $SALARY$) attributes is important to the design of preference databases. We classify the attributes in a preference relation schema according to their preference nature into three categories of the *preference*, *preference-dependent* and *preference-independent* attributes. Such a classification of attributes allows us to express the semantics of a Preference Functional Dependency (PFD) in terms of the satisfaction of a corresponding set of LOFDs in a relation, each of them with its left-hand side restricted to a sequence of prioritized preference attributes and its right-hand side restricted to a single preference-dependent attribute in the canonical form.

A PFD is a semantic constraint that arises naturally from preference relations, since most preference data is dependent on preference in a monotonic manner. For example, the PFD $\langle PRICE, CATEGORY, POWER, MILEAGE\rangle \mapsto MIDDLE\_CLASS$ states the fact that when some middle class customers choose a second-hand car in a market, the complex preference can be prioritized in terms of the sequence of selling price, category, engine power, and used mileage. Note that the order of the attributes here may not follow the alphabetical or numerical order. For example, the engine power preference may be defined as $\{3500cc <_p 1500cc <_p 3000cc <_p 2000cc\}$, where $<_p$ captures the meaning of "is preferable to".

The main result of this paper is fundamental. We establish a simple, sound and complete axiom system for PFDs and show that the implication problem for PFDs is decidable in polynomial-time. This result is significant because it implies that, in principle, by using our established system all possible PFDs being logically implied by a given set of PFDs can be effectively generated in polynomial-time. The simplicity of the system for PFDs is also important from

the point of view of *usability*, since it provides different categories of database users with impetus for accepting and applying PFDs in preference data modelling. In addition, the axiom system provides us with a basis to find a more efficient algorithm for solving the implication problem of PFDs.

In order to incorporate PFDs into preference database design, we tackle the problem whether Armstrong relations exist for PFDs. The importance of Armstrong relations for FDs in the process of database design is well-recognised in conventional database design [9, 2]. In our case, Armstrong preference relations exist and can also be served as *example relations* in the design process. They help the database designers to gain a better insight of PFDs needed in modelling the preference data in an enterprise.

The rest of the paper is organised as follows. In Section 2 we review some related work. In Section 3 we clarify the notion of the extension of linear order to the Cartesian product of linearly ordered sets and then define the ordered relational model. In Section 4 we present the chase rules for LOFDs and, using an extended notion of tableaux for LOFDs, show that the chase is sound and complete for LOFDs. In Section 5 we illustrate the uses of LOFDs and formally define PFDs in preference relations. A sound and complete axiom system is then presented for PFDs. We also show that Armstrong relations exist for PFDs. In Section 6 we give our concluding remarks.

2 Related Work

There are a number of different approaches for defining constraints as a fundamental component in databases. Functional Dependencies (FDs) [2] are essentially the kinds of *equality generating dependencies*. It is worth mentioning that in [3, 4] the axiom system for partial order dependencies is co-NP, which limits the usability of such kind of *order comparison dependencies*. However, we impose a preference partial order to override a linearly ordered domain whenever there is a conflict and thus, the linear order assumption simplify most of the complex results.

Comparing to the body of work on data dependencies in databases in literature [2, 7, 8], we believe that our definition of PFDs are a novel constraint that is useful in preference relations for three main reasons. First, in our data model we consider the fundamental feature that linear ordering is an integral part of simple preference domains from which to derive complex preferences in a relation in a straightforward manner. Second, a PFD captures the semantics of ordering relationships that arises naturally from the interaction between preference and preference-dependent data. Third, the concept of PFDs is founded on the formal notion of lexicographical ordering on preference domains, which paves the way to develop more advanced preference constraints.

Soft constraints in the form of numerical preferences have been used in many database and information retrieval applications [1, 8]. Basically, the usual numerical order are used for ranking preference items. For example, in the area of full-text searching [1, 13], where keywords can be understood as implicit score

preferences indicating their relevance. The combining function for ranking is typically some scalar product taking the cosine function commonly used in the vector space model from information retrieval [13]. Preferences are receiving attention as DBMSs need to provide better information service. Preference SQL [6] has been extended by a "preferring" clause that allows user to specify soft constraints reflecting complex preferences.

3 The Ordered Relational Model

We assume throughout that sequences consist only distinct attributes. For any two sequences X and Y, $X \sim Y$ denotes the fact that X and Y have the same elements. XY denotes the *concatenation* of X and Y, where $XY = X(Y - X)$. A *prefix* of X, denoted by $pre(X)$, is a sequence of the form $\langle A_1, \ldots, A_{m_1} \rangle$, where $X = \langle A_1, \ldots, A_m \rangle$ and $1 \leq m_1 \leq m$. A *shuffle* of X and Y, denoted by $shu(X, Y)$, is defined as a sequence of the form $\langle C_1, \ldots, C_{m+n} \rangle$, where there exists two subsequences of attributes $\langle C_{i_1}, \ldots, C_{i_m} \rangle = X$ and $\langle C_{j_1}, \ldots, C_{j_n} \rangle = Y$, and the order of the attributes in X and Y is preserved in $shu(X, Y)$.

As usual a *linear ordering* $\leq$ on a set S is a binary relation on S which satisfies the conditions of *reflexivity*, *anti-symmetry*, *transitivity* and *linearity*. A *linearly ordered set* (or simply an ordered set) is a structure $\langle S, \leq \rangle$. We assume the usual predicates, $=$ and $<$, still applies to ordered sets. We let $D_1, \ldots, D_n$ be n ordered sets, t be an element in the Cartesian product $S = D_1 \times \cdots \times D_n$, and $t[i]$ be the ith coordinate of t. We now define *lexicographical ordering* in order to capture the semantics of data.

Definition 1. (Lexicographical Ordering) Let t_1, $t_2 \in S$. A *lexicographical ordering* on S is a linear ordering $\leq_S^{lex}$ (or simply $\leq_S$ whenever it is clear from the context) such that $t_1 \leq_S t_2$ if either (1) there exists k with $1 \leq k \leq n$ such that $t_1[k] <_{D_k} t_2[k]$, and for all $1 \leq i < k$, $t_1[i] = t_2[i]$, or (2) for all $1 \leq i \leq n$, $t_1[i] = t_2[i]$.

Lexicographical ordering is a fundamental property of many primitive data types, for example the alphabetical ordering over the domain of characters. It is easy to see that the preference of "the earlier the better" can be modelled as an ordering of the domain *DATE*, which is called a *chronological ordering*.

Let D be a countably infinite set of constant values and $\leq_D$ be an ordering on D. We assume that all attributes share the same domain D. A *relation schema* $R = \{A_1, \ldots, A_m\}$, is a non-empty finite subset of a countably infinite set of attributes U. A *database schema* $\mathbf{R}$ is a non-empty finite set of relation schemas. A *tuple* t over R is a member of D^m and $t[A_i]$ denotes the ith coordinate of t, i.e. the *projection* of t onto the attribute A_i. A *relation* r defined over R is a finite set of tuples over R. An *database* over $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a finite set of relations $d = \{r_1, \ldots, r_n\}$ where by convention r_i is a relation over R_i.

We also make two assumptions in our model.

First, given a data domain, $D = \{a, b, c\}$, there is a *background ordering* (linear ordering) such as alphabetical ordering or numerical ordering on the

domains associated with the attributes present in the underlying schema. In this case we have the background ordering $\{a <_{bg} b <_{bg} c\}$ over D.

Second, apart from the standard ordering assumption, according to the preference (modelled as a set of *irreflexive* and *transitive* ordered pairs) such as $P = \{c <_p a, c <_p b\}$ used in an application, we can declare one or more *preference orderings* which override the default standard ordering. A preference ordering is also a linear ordering formed by imposing all the preference on the background ordering. For example, if the background ordering of $\leq_{bg}$ is used, then the preference ordering arising from P is given by $\{c <_p a <_p b\}$ in which $(a <_p b) \notin P$ is derived from $a <_{bg} b$.

These two assumptions are found to be useful in many database applications. The second assumption simplify some technical complications in our subsequent discussion, since all comparison is defined. From the application view point, in most cases we also need to present the ranked (i.e. linearly ordered) result according to user preferences. (Readers may refer to [11, 12] for more detailed discussion of various notions of orderings in our model and their applications.)

4 Lexicographically Ordered Functional Dependencies

In this section, we present lexicographically ordered functional dependency and a chase procedure to maintain its consistency over an ordered relation.

The semantics of a Lexicographically Ordered Functional Dependency (LOFD) with two or more attributes on either the left- or right-hand side is defined according to lexicographical orderings on the Cartesian product of the underlying domains of the attributes in the LOFD.

Definition 2. (Lexicographically Ordered Functional Dependency) A *lexicographically ordered functional dependency* (or simply an LOFD) over a relation schema R, is a statement of the form $R : X \rightsquigarrow Y$ (or simply $X \rightsquigarrow Y$ whenever R is understood from the context), $X, Y \subseteq R$ are sequences of attributes and $X \neq \emptyset$. An LOFD, $X \rightsquigarrow Y$, is satisfied in a relation r over R, denoted by $r \models X \rightsquigarrow Y$, if for all $t_1, t_2 \in r$, $t_1[X] \leq_X^{lex} t_2[X]$ implies that $t_1[Y] \leq_Y^{lex} t_2[Y]$.

We denote $min(a, b)$ and $max(a, b)$ the minimum and maximum of the values a and b, respectively. For any two distinct tuples $t_1, t_2 \in r$ and some $A \in R$, the *equate* of t_1 and t_2 on A, denoted as *equate*($t_1[A]$, $t_2[A]$), is defined by replacing both $t_1[A]$ and $t_2[A]$ by $min(t_1[A], t_2[A])$; the *swap* of t_1 and t_2 on A, denoted as *swap*($t_1[A]$, $t_2[A]$), is defined by replacing $t_1[A]$ by $min(t_1[A], t_2[A])$ and $t_2[A]$ by $max(t_1[A], t_2[A])$. We now extend the classical chase defined over conventional relations with respect to FDs [2] to ordered relations with respect to LOFDs.

Definition 3. (Chase Rules for LOFDs) Let t_1 and t_2 be two tuples in r such that $t_1[X] \leq_X^{lex} t_2[X]$ but $t_1[Y] \not\leq_Y^{lex} t_2[Y]$, A be the first attribute in X such that $t_1[A] \neq t_2[A]$, if such an attribute exists, and B be the first attribute in Y such that $t_1[B] \neq t_2[B]$, then the *chase rules* for the LOFD $X \rightsquigarrow Y$ is defined by the following two rules:

1. **Equate rule:** if $t_1[X] = t_2[X]$ but $t_1[B] \neq t_2[B]$, then $equate(t_1[B], t_2[B])$;
2. **Swap rule:** if $t_1[A] < t_2[A]$ but $t_2[B] < t_1[B]$, then $swap(t_1[B], t_2[B])$, or if $t_2[A] < t_1[A]$ but $t_1[B] < t_2[B]$, then $swap(t_1[A], t_2[A])$.

The said chase rules cater for all of the possible cases when there are two tuples in a relation violating $X \rightsquigarrow Y$ (see [10] for details). We now give the pseudo-code of an algorithm designated $CHASE(r,\mathrm{F})$, which applies the chase rules given in Definition 3 to R for as long as possible, and returns the resulting relation r over R.

Algorithm 1 ($CHASE(r, \mathrm{F})$)

```
1.   begin
2.     Result := r = ⟨t_1, ..., t_n⟩ ;
3.     Tmp:= ∅;
4.     while Tmp ≠ Result do
5.       Tmp := Result;
6.       if ∃X ⇝ Y ∈ F, ∃ t_p, t_q ∈ Result such that t_p[X] ≤_X^lex t_q[X]
         but t_p[Y] ≰_Y^lex t_q[Y]
7.       then apply the appropriate chase rule to Result with
         t_1 = t_min(p,q) and t_2 = t_max(p,q);
8.     end while
9.     return Result;
10.  end.
```

It is easy to verify that $CHASE(r,\mathrm{F})$ in Algorithm 1 satisfies F, which can be computed in preference polynomial in the sizes of r and F. In order to provide a proof procedure for LOFDs, we need to define the notion of *ordered variable domains* as follows. The *variable domain* of a relation schema R, denoted by $vdom(R)$, is the finite set $\{l_1, \ldots, l_m, h_1, \ldots, h_m\}$, where $m = \mid R \mid$. The variables l_i and h_i with $i \in \{1, \ldots, m\}$ are called *low ordered variables* and *high ordered variables*, whose ordering is given by $l_i < h_i$.

Definition 4. (Template Relations for an LOFD) Let f be the LOFD $X \rightsquigarrow Y$ over R with $\mid X \mid = n$ and $\mid R \mid = m$. We use two shorthand symbols, u_i and v_i, to represent one of the following three cases: (1) $u_i = l_i$ and $v_i = l_i$, (2) $u_i = l_i$ and $v_i = h_i$, or (3) $u_i = h_i$ and $v_i = l_i$. A *template relation* (or simply a *template*) with respect to f, denoted as r_f, is a relation consisting of two tuples, t_1 and t_2, whose underlying domain is $vdom(R)$, such that it is equal to either Γ_0 or Γ_k shown in Figure 1 , where $pre(X) = \langle x_1, \ldots, x_k \rangle$ for $1 \leq k \leq n$.

$\Gamma_0 =$

	X	$R - X$
t_1	$l_1 \cdots l_n$	$u_{n+1} \cdots u_m$
t_2	$l_1 \cdots l_n$	$v_{n+1} \cdots v_m$

$\Gamma_k =$

	$x_1 \cdots x_{k-1}$	x_k	$R - pre(X)$
t_1	$l_1 \cdots l_{k-1}$	l_k	$u_{k+1} \cdots u_m$
t_2	$l_1 \cdots l_{k-1}$	h_k	$v_{k+1} \cdots v_m$

Fig. 1. Template relations for an LOFD

We remark that in Definition 4, the symbols u_i and v_i represent three possibilities of combinations of l_i and h_i; it is easy to verify that the order of the upper bound of the number of templates is $O(3^m)$, where m is the number of attributes in R. Note that m is usually small for a relation schema in practice. We will also show in Theorem 3 that the complexity of chasing all the templates can be greatly reduced when the templates are applied for PFDs.

A template can be viewed as a relation instance consisting of two tuples by mapping values in D to ordered variables. We define *tableaux*, denoted by Γ_f, to be the set of all template relations in Definition 4. The tableaux in our case is different from that for FDs, which requires just a single template for FDs (see Theorem 4.2 in [2]). The chase of Γ_f, denoted as $CHASE(\Gamma_f, F)$, is defined by $CHASE(\Gamma_f, F) = \{CHASE(r_f, F) \mid r_f \in \Gamma_f\}$. $CHASE(\Gamma_f, F)$ satisfies $X \rightsquigarrow Y$, denoted by $CHASE(\Gamma_f, F) \models X \rightsquigarrow Y$, if for all $r_f \in \Gamma_f$, $CHASE(r_f, F) \models X \rightsquigarrow Y$. Furthermore, $CHASE(\Gamma_f, F)$ satisfies F, denoted by $CHASE(\Gamma_f, F) \models$ F, if for all $X \rightsquigarrow Y \in$ F, $CHASE(\Gamma_f, F) \models X \rightsquigarrow Y$. We now re-state the main theorem for LOFDs (c.f. see the proof of Theorem 9, page 550 in [10]), which shows that the chase procedure is a decidable, sound and complete inference algorithm for LOFDs.

Theorem 1. Let F be a set of LOFDs over R and f be a LOFD $X \rightsquigarrow Y$. Then $CHASE(\Gamma_f, F) \models f$ if and only if F $\models f$. $\square$

5 Applications of LOFDs in Preference Relations

In this section we discuss the applications of LOFDs in preference relations. We define a novel constraint called Preference Functional Dependencies (PFDs) and establish a corresponding set of axiom system that tackle the implication problem of PFDs.

We adopt a natural view of a user preference, which is perceived as being a declaration of a set of *irreflexive* and *transitive* ordered pairs specified by users [5, 6]. The specification includes non-numerical and numerical ranking methods. This view affords us a general way to define preference as a sequence of ordered domains of preference related attributes (or simply preference domains) and further classify preference domains into *simple* and *complex* ones.

Definition 5. (Simple and Complex Preference Domains) A preference domain is said to be *simple* if its data elements (i.e. preference items) are *atomic*, meaning that the elements in such a domain are indivisible as far as the DBMS is concerned. A *complex* preference domain is defined as a sequence of more than one simple preference domain. The ordering of a complex preference domain is defined according to the lexicographical ordering on the Cartesian product of the involved simple preference domains.

Some examples of commonly used simple preference domains related to second handed cars are $PRICE\_RANGE = \{(1000 - 2000) < (2001 - 3000) < \cdots < (9001 - 10000)\}$ and $ENGINE\_POWER = \{1500cc < \cdots < 3000cc\}$. We

can also make use $I(n) = \{1, \ldots, n\}$ as simple domains to construct a complex preference domain, which provides a convenient way to model a complex preference system as well as to define an arbitrary preference granularity (cf. the formation of complex preference in [6]).

We assume that there is a partition on the countably infinite set of attributes U, which forms the three distinguished classes of the *preference* (PE), the *preference-dependent* (PD), and the *preference-independent* (PI) attributes. Under this classification, preference is specified by choosing only the attribute names defined in PE. Furthermore, we are able to differentiate the data elements that represent preference (such as $PRICE\_RANGE$ or $ENGINE\_POWER$), that are dependent on preference (such as *FAMILY_CHOICE* or *BUSINESS_CHOICE*), and that are invariant with respect to preference (such as *PAYMENT_METHOD* or *SALES_MANAGER*).

Definition 6. (Preference Relation Schema and Preference Relation) Let PE, PD and PI be three distinguished non-empty subsets of U such that they are pairwise disjoint. A *preference relational schema* R is a schema which satisfies $R \subseteq PE \cup PD \cup PI$ and $R \cap PE \neq \emptyset$. A *preference relation* is a relation r over R. For simplicity in notation, we denote $R \cap PE = R^{PE}$, $R \cap PD = R^{PD}$ and $R \cap PI = R^{PI}$. We call r a *PE relation* when $R = R^{PE}$.

The following example illustrates that LOFDs are employed to maintain the consistency of the data elements in different preference domains.

Example 1. Suppose a customer preference relation is defined by the preference attributes $PRICE\_RANGE$, $ENGINE\_POWER$ and $MILEAGE$. The following LOFD, $\langle PRICE\_RANGE,\ ENGINE\_POWER,\ MILEAGE\rangle \rightsquigarrow YOUTH\_CHOICE$, asserts the preference specification defined by $PRICE\_RANGE$, $ENGINE\_POWER$ and $MILEAGE$, such that the rank of $YOUTH\_CHOICE$ (the choice of young customers) increases with first the price range and then the engine power and finally the car milage. In addition, we use the PREFERRING clause in [5] to express the preference terms, which essentially impose the preference order over their corresponding data domains given by (1) LOWEST(price), (2) HIGHEST(power) and (3) mileage AROUND 30000km.

A preference relation which is described by the preference attributes $PRICE\_RANGE$ and $ENGINE\_POWER$, where $YOUTH\_CHOICE = I(5)$ (i.e. 5 possible ranks are used for labelling the preference of the second-hand cars) is used for defining the overall preference ranking, as shown in Figure 2. Note that some attributes are abbreviated in the table due to width limit. The attributes $SALES\_MANAGER$ and $PAYMENT\_METHOD$ are preference-independent attributes.

For the choice from middle class customers, we may need to change the preference terms, which give rise to a different consistent preference relation as shown in Figure 3. The preference order is (1) price AROUND \$4000-\$5000, (2) HIGHEST(power) and (3) LOWEST(mileage).

Next, we define an interesting class of LOFDs, called Preference Functional Dependencies (PFDs), to order to capture the ordering semantics between the

two sets of values projected onto the preference and preference-dependent attributes. A PFD is essentially an LOFD that has a sequence of the preference attributes in the left-hand side, constituting a simple or a compound preference domain, and the preference-dependent attributes in the right-hand side, capturing the semantics of preference-dependent data.

Finally, for the choice of pensioner customers, we may have another set of preference terms and also need to impose the new LOFD, $\langle PRICE\_RANGE$, $MILEAGE$, $ENGINE\_POWER\rangle \rightsquigarrow PENSIONER\_CHOICE$, which give rise to a different consistent preference relation shown in Figure 4. The preference order is (1) LOWEST(price), (2) mileage BETWEEN 20000km AND 30000km, and (3) power AROUND 2000cc.

$PRICE$	$ENGINE$	$MILEAGE$	$YOUTH$	$MANAGER$	$PAYMENT$
1001-2000	1500cc	20000km	1	Jane	Installment
4001-5000	3000cc	30000km	2	Jane	Installment
4001-5000	2000cc	20000km	3	Ken	Installment
4001-5000	1500cc	10000km	4	Ken	Installment
5001-6000	3000cc	10000km	5	Larry	Installment

Fig. 2. An example showing $\langle PRICE\_RANGE$, $ENGINE\_POWER$, $MILEAGE\rangle \rightsquigarrow YOUTH\_CHOICE$ in a second-hand car preference relation

$PRICE$	$ENGINE$	$MILEAGE$	$MIDDLE\_CLASS$	$MANAGER$	$PAYMENT$
1001-2000	1500cc	20000km	5	Jane	Installment
4001-5000	3000cc	30000km	1	Jane	Installment
4001-5000	2000cc	20000km	2	Ken	Installment
4001-5000	1500cc	10000km	3	Ken	Installment
5001-6000	3000cc	10000km	4	Larry	Installment

Fig. 3. An example showing $\langle PRICE\_RANGE$, $ENGINE\_POWER$, $MILEAGE\rangle \rightsquigarrow MIDDLECLASS\_CHOICE$ in a second-hand car preference relation

$PRICE$	$ENGINE$	$MILEAGE$	$PENSIONER$	$MANAGER$	$PAYMENT$
1001-2000	1500cc	20000km	1	Jane	Installment
4001-5000	3000cc	30000km	3	Jane	Installment
4001-5000	2000cc	20000km	2	Ken	Installment
4001-5000	1500cc	10000km	4	Ken	Installment
5001-6000	3000cc	10000km	5	Larry	Installment

Fig. 4. An example showing $\langle PRICE\_RANGE$, $MILEAGE$, $ENGINE\_POWER\rangle \rightsquigarrow PENSIONER\_CHOICE$ in a second-hand car preference relation

Definition 7. (Preference Functional Dependency) A *Preference Functional Dependency* (PFD) over a preference relation schema R, denoted as $R : T \mapsto X$ (or simply $T \mapsto X$ whenever R is understood from the context), $T \subseteq R^{PE}$ is a sequence of the preference attributes and $X \subseteq R^{PD}$ is a sequence of the preference-dependent attributes. A PFD, $T \mapsto X$, is satisfied in a preference relation r over R, if and only if, for all $A \in X$, $r \models T \rightsquigarrow A$.

From now on, we use $F' = \{T \mapsto A \mid T \mapsto X \in \mathrm{F} \text{ and } A \in X\}$ to represent the set of *unary* PFDs corresponding to F. The following proposition immediately follows from Definition 7, which justifies the equivalence in semantics between F and F'.

Proposition 1. $r \models \mathrm{F}$ if and only if $r \models F'$. □

An *axiom system* $\mathcal{A}$ for F is a set of inference rules (or simply rules) that can be used to *derive* PFDs from F over R. We denote by $\mathrm{F} \vdash f$ the fact that f is *derivable* from F by a specified axiom system [2, 10].

Definition 8. (Inference Rules for PFDs) Let F be a set of PFDs over R and T_1, T_2 be subsets of R^{PE}. The inference rules for PFDs are defined as follows:

(PFD1) *Shuffle*: If $\mathrm{F} \vdash T_1 \mapsto X$ and $\mathrm{F} \vdash T_2 \mapsto X$, then $\mathrm{F} \vdash shu(T_1, pre(T_2)) \mapsto X$.
(PFD2) *Left Expansion*: If $\mathrm{F} \vdash T_1 \mapsto X$, then $\mathrm{F} \vdash T_1 T_2 \mapsto X$.
(PFD3) *Decomposition*: If $\mathrm{F} \vdash T_1 \mapsto X$, then $\mathrm{F} \vdash T_1 \mapsto Y$, where $Y \subseteq X$.
(PFD4) *Union*: If $\mathrm{F} \vdash T_1 \mapsto X$ and $\mathrm{F} \vdash T_1 \mapsto Y$, then $\mathrm{F} \vdash T_1 \mapsto XY$.

We remark that the axiom system comprising these rules is *minimal*, since the four rules given in Definition 8 are primitive. The *reflexivity* rule is not applicable for PFDs, since the preference attributes exist only in the left-hand side of LOFDs. We still need the following inference rule to deal with the sequences of attributes in the right-hand side of LOFDs, which can be derived from the inference rules PFD3 and PFD4.

Proposition 2. The following inference rule is sound.

(PFD5) *Permutation*: If $\mathrm{F} \vdash T_1 \mapsto X$, then $\mathrm{F} \vdash T_1 \mapsto X'$, where $X \sim X'$. □

We now show in next theorem that the axiom system comprising the inference rules in Definition 8 is sound and complete for PFDs, holding in preference relations. The underlying idea in this proof is first to assume that a PFD $T \mapsto A$ cannot be inferred from the axiom system, and then to present a relation as a counter-example in which all the PFDs of F' hold except $T \mapsto A$ (c.f. see Theorem 3.21 in [2]). The result is significant since it indicates that the axiom system can be employed as a theorem-proving tool for PFDs.

Theorem 2. The axiom system comprising from PFD1 to PFD4 is sound and complete for PFDs.

Proof. It is easy to show that the inference rules from PFD1 to PFD4 are sound. We now establish the completeness by showing that if $F' \nvdash T \mapsto A$, then $F' \not\models T \mapsto A$. Equivalently for the latter, it is sufficient to exhibit a relation as a counter-example r^c, such that $r^c \models F'$ but $r^c \not\models T \mapsto A$. Assuming that for all $Q \subseteq R^{PE}$, P is the largest prefix of T such that $F' \vdash PQ \mapsto A$. Let us call this the *P-assumption*. There are two cases to consider.

In the first case, we assume $P = T$. We consider the relation r^c shown in Figure 5, where $A \in R^{PD}$ and $Z = R^{PD} R^{PI} - A$. Obviously, we have $r^c \not\models$

$T \mapsto A$. It remains to show that $r^c \models F'$. Assume to the contrary that $r^c \not\models F'$. So $\exists f \in F'$ such that $r^c \not\models f$. Let $f = T' \mapsto A'$. By the construction of r^c, we have $T' \subseteq T$ and $A' = A$. By the P-assumption and PFD1, it follows that $F' \vdash PT' \mapsto A$. So we have $F' \vdash P \mapsto A$, which is a contradiction, since we derive $T \mapsto A$ from F'.

	T	$R^{PE} - T$	A	Z
t_1	$0 \cdots 0$	$0 \cdots 0$	0	$0 \cdots 0$
t_2	$0 \cdots 0$	$1 \cdots 1$	1	$0 \cdots 0$

Fig. 5. A counter-example relation r^c used in the case of $P = T$

	P	B	$R^{PE} - BP$	A	Z
t_1	$0 \cdots 0$	1	$0 \cdots 0$	0	$0 \cdots 0$
t_2	$0 \cdots 0$	0	$1 \cdots 1$	1	$0 \cdots 0$

Fig. 6. A counter-example relation r^c used in the case of $P \neq T$

In the second case, we assume $P \neq T$. Let $T = PBQ$ where $B \notin P$ and $BQ \subseteq R^{PE}$. We construct the relation r^c shown in Figure 6, in which $r^c \not\models T \mapsto A$.

We now show that $r^c \models F'$. Assuming to the contrary that $\exists f \in F'$ such that $r^c \not\models f$, where $f = T' \mapsto A'$. By the construction of r^c, we have $A' = A$ and the following two possible cases concerning T'.

(Case of $T' \subseteq P$). By PFD2, we expand f by attaching the attribute B on its left-hand side. It follows that $F' \vdash T'B \mapsto A$. By the P-assumption and PFD1, it follows that $F' \vdash PT'BQ \mapsto A$. So we have $F' \vdash PBQ \mapsto A$. But PB is the prefix of T and strictly contains P. This leads to a contradiction, since we violate the P-assumption.

(Case of $T' \not\subseteq P$). Let $T' = VBW$ where $V \subseteq P$ and $W \subseteq R^{PE}$. By the P-assumption and PFD1, it follows that $F' \vdash PT' \mapsto A$. So we have $F' \vdash PBW \mapsto A$. But PB is the prefix of T. This leads to the same contradiction, since we violate the P-assumption. □

As we discussed in Section 3, the number of possible templates used in $CHASE(\Gamma_f, F)$ is $O(3^m)$. We now show that the complexity of applying the chase for PFDs is much better than exponential-time. Let $T = \langle B_1, \ldots, B_n \rangle$ (i.e. a sequence of n preference attributes for some positive integer n) and $f = T \mapsto A$. For $k \in \{0, \ldots, n\}$, we define the *k-th reduced form* of a given set of PFDs F with respect to f by $\Delta_k(F) = \{T' \mapsto A \in F \mid T' \subseteq T\}$ when $k = 0$, and $\Delta_k(F) = \{T' \mapsto A \in F \mid T' = pre(WB_kQ)$ such that $W \subseteq \{B_1, \ldots, B_{k-1}\}$ and $Q \subseteq R^{PE}\}$ when $n \geq k > 0$.

Theorem 3. Let F be a set of PFDs and $f = T \mapsto A$ where $\mid T \mid = n$. Then $\Delta_k(F) \neq \emptyset$ for all $k \in \{0, \ldots, n\}$ if and only if $CHASE(\Gamma_f, F) \models T \mapsto A$.

Proof. (*IF:*) Let $A \in R^{PD}$ and $Z = R^{PD}R^{PI} - A$. From Definition 4, a template r_f in Γ_f can be equal to either Γ_0 or Γ_k with $n \geq k > 0$, which give rise to the following two cases.

($k = 0$). Consider the following template r_f that is derived from Γ_0.

$r_f =$

	T	$R^{PE} - T$	A	Z
t_1	$l_1 \cdots l_n$	$l_{n+1} \cdots l_p$	l_{p+1}	$l_{p+2} \cdots l_m$
t_2	$l_1 \cdots l_n$	$h_{n+1} \cdots h_p$	h_{p+1}	$l_{p+2} \cdots l_m$

By the assumption of $CHASE(\Gamma_f, F) \models T \mapsto A$, it follows that $CHASE(r_f, F) \models T \mapsto A$. Clearly, there exists at least one application of a chase rule in order to fix the violation of $T \mapsto A$ in r_f. From the construction of r_f, it can check that the only possible way to initialise the chase is to have some $T' \mapsto A$ in F such that $T' \subseteq T$. So $\Delta_0(F) \neq \emptyset$.

$(n \geq k > 0)$. Consider the following template r_f derived from Γ_k, where $P = \langle B_1, \ldots, B_{k-1} \rangle$.

$r_f =$

	P	B_k	$R^{PE} - PB_k$	A	Z
t_1	$l_1 \cdots l_{k-1}$	l_k	$h_{k+1} \cdots h_p$	h_{p+1}	$l_{p+2} \cdots l_m$
t_2	$l_1 \cdots l_{k-1}$	h_k	$l_{k+1} \cdots l_p$	l_{p+1}	$l_{p+2} \cdots l_m$

Again, by the given assumption, it follows that $CHASE(r_f, F) \models T \mapsto A$. Thus, there exists at least one application of a chase rule in order to fix the violation of $T \mapsto A$ in r_f. From the construction of r_f, it can check that the only possible way to initialise the chase is to have some $T' \mapsto A$ in F such that one of the following conditions holds: (i) $T' = W$, (ii) $T' = WB_k$, or (iii) $T' = WB_kQ$, where $W \subseteq P$ and $Q \subseteq R^{PE}$. It follows that $T' = pre(WB_kQ)$. So $\Delta_k(F) \neq \emptyset$. (*ONLY IF:*) It follows from the antecedent that, for all $k \in \{0, \ldots, n\}$, there exists a PFD $f' = T' \mapsto A \in \Delta_k$ in F such that f' is violated in the templates derived from Γ_k. From Definition 4, there are two cases concerning the templates in Γ_f to consider.

First, if a template r_f is derived from Γ_0, then we have $t_1[T] = t_2[T] = \langle l_1, \ldots, l_n \rangle$ in r_f, whose equality will not be changed by any chase rules. So we have $t_1[A] = t_2[A] = l_p$ in $CHASE(r_f, F)$. It follows that $CHASE(r_f, F) \models f$. Second, if a template r_f is derived from Γ_k where $n \geq k > 0$, then we have $t_1[PB_k] = \langle l_1, \ldots, l_{k-1}, l_k \rangle$ and $t_2[PB_k] = \langle l_1, \ldots, l_{k-1}, h_k \rangle$ in r_f. The relative ordering $t_1[PB_k] <^{lex} t_2[PB_k]$ will not be changed by any chase rules, since the attributes in PB_k exists in the left-hand side of an PFD. So we have $t_1[A] = l_p$ and $t_2[A] \in \{l_p, h_p\}$ in $CHASE(r_f, F)$. It follows that $CHASE(r_f, F) \models f$.

The result thus follows, since $f = T \mapsto A$ is satisfied in $CHASE(\Gamma_f, F)$. □

We now formally state the result concerning the complexity of $CHASE(\Gamma_f, F)$. The next corollary follows from Algorithm 1 and Theorem 3.

Corollary 1. Let F be a set of PFDs over R and f be a PFD. Then $CHASE(\Gamma_f, F)$ can be computed in polynomial-time in the sizes of R and F.
Proof. Let $f = T \mapsto A$ and $T = \mid n \mid$. It is easy to check that the decision of whether $\Delta_k(F)$ being empty for a given $k \in \{0, \ldots, n\}$ depends linearly on the number of PFDs in F. Consider that the maximum size of n is equal to $\mid R \mid$. The result is then followed by Algorithm 1 and Theorem 3. □

In order to incorporate PFDs into preference database design, we now tackle the problem whether Armstrong relations exist for PFDs. The importance of such relations for standard FDs in the process of database design is well-recognised [9, 2]. Essentially, Armstrong relations can be served as *example relations* in the design process. They help the designers to gain a better insight of the data

dependencies needed in modelling the data in an enterprise. We now give the definition of Armstrong relations in the context of preference relations.

Definition 9. (**Armstrong Preference Relation**) Let F^* be the set of all PFDs that are logically implied by a given set of PFDs F over R. An *Armstrong preference relation* for F is a preference relation r^{Arm} over a preference schema R such that $r^{Arm} \models T \mapsto X$ if and only if $T \mapsto X \in F^*$.

We will use $poss(R)$ in Lemma 1 and Theorem 4 to represent the set of all PFDs that can be defined over a schema R.

Lemma 1. Given preference relations r_1 and r_2 over R. Then there exists a preference relation r_3 over R such that $F_3 = F_1 \cap F_2$, where $F_i = \{f \mid r_i \models f\}$ for $i = 1, 2, 3$.

Proof Sketch. We start by assuming that all the attributes share with a common (ordered) domain of integers. Let $r_1 = \{t_1, \ldots, t_m\}$ and $r_2 = \{s_1, \ldots, s_n\}$ where $1 \leq m, n$. We define the *active domain* of a relation r over R by $adom(r)$ $= \{v \mid \exists A \in R, \exists t \in r$ such that $t[A] = v\}$. We define the *safe distance* of two relations r_1 and r_2 over R by $sdist(r_1, r_2) = max(adom(r_1)) - min(adom(r_2)) + 1$. Intuitively, *sdist* guarantees that the satisfaction of a PFD in a relation can be preserved under the union operation. For each tuple $s_i \in r_2$, we define t'_i as follows: $t'_i[A] = s_i[A] + sdist(r_1, r_2)$ for each $A \in R$. We construct $r_3 = r_1 \cup r_4$, where $r_4 = \{t'_1, \ldots, t'_n\}$ being a preference relation over R. Then we can show that $F_3 = F_1 \cap F_2$ by establishing the fact that $F_3 \subseteq F_1 \cap F_2$ and $F_3 \supseteq F_1 \cap F_2$. □

We now show that Armstrong preference relations exist. Our construction is essentially to adapt the classical techniques in [9, 2] for constructing Armstrong relations in our context.

Theorem 4. Given a set of PFDs F over R. There exists an Armstrong preference relation for F.

Proof. Let $F^* = \{f \mid F \models f\}$ and $\bar{F} = \{f \in poss(R) \mid f \notin F^*\}$. We now construct a preference relation r^{Arm} such that $r^{Arm} \models f'$ if and only if $f' \in F^*$. For the "if" case we let $f \in \bar{F}$. We then have a preference relation $r_f \models F$ but $r_f \not\models f$. Let $F_f = \{f \mid r_f \models f\}$. By repeated application of Lemma 1 running for all $f \in \bar{F}$, we have a relation r^{Arm} such that $r^{Arm} \models \bigcap_{f \in \bar{F}} F_f$. Since $F^* \subseteq F_f$, we have $F^* \subseteq \bigcap_{f \in \bar{F}} F_f$. It thus follows that $r^{Arm} \models F^*$. It remains to show the "only if" case. Let $f \notin F^*$. Then $f \in \bar{F}$. So there exists F_f such that $f \notin F_f$. It follows that $f \notin \bigcap_{f \in \bar{F}} F_f$. Thus, $r^{Arm} \not\models f$. □

6 Concluding Remarks

We consider preference handling in relations in order to express the daily and business preferences in practice. Within the context of preference relations, we discussed the applications of LOFDs in the areas of (1) maintaining the consistency of preference data, and (2) capturing the semantics of the ordering relationship between preference and preference-dependent data. In order to establish

an in-depth study for the latter case, we defined PFDs in Definition 7 based on the notions of LOFDs and ordered relations, and presented a sound and complete axiom system for PFDs in Theorem 2. We showed in Theorem 3 that the complexity of chasing the tableaux given in Definition 4 is equivalent to decide whether all the reduced forms of a given set of PFDs are empty. As a result, the complexity of the implication problem for PFDs is found to be polynomial-time in the sizes of the relation schema and the given set of PFDs. We also showed in Theorem 4 that Armstrong relations exist for PFDs, an important result for incorporating PFDs into the process of database design in practice. We remark that the issues related to the interaction between PFDs and FDs is an interesting area to explore. For example, if $r \models \{T \mapsto X, T \mapsto Y\}$, then an equivalence class s in a partition of r, whose tuples have the same value of $t[T]$ for some $t \in r$, satisfies the FD $X \rightarrow Y$. In order to enhance the expressive power of our PFDs we are considering to generalise PFDs to incorporate vagueness semantics [8] in complex preference in the future work. Another more challenging issue is to explore the possibilities of efficient evaluating preference SQL [5] by using PFDs. The motivation is that, if PFDs can be applied to maintain a preference view of data then it may help some commercial web sites to give a more effective and efficient answer when facing a heavy load of preference SQL queries.

References

1. S. Amer-Yahia et al. *Structure and Content Scoring for XML.* In: Proc. of VLDB, (2005).
2. P. Atzeni and V. De Antonellis. *Relational Database Theory.* Benjamin/ Cummings Publishing Company, Inc., (1993).
3. S. Ginsburg and R. Hull. Order Dependency in the Relational Model. *Theoretical Computer Science* **26**(1-2), pp. 149-195, (1983).
4. S. Ginsburg and R. Hull. Sort Sets in the Relational Model. *Journal of the Association for Computing Machinery* **33**(3), pp. 465-488, (1986).
5. W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experiences. In: *Proc. of VLDB*, (2002).
6. W. Kießling and G. Köstler. Foundations of Preference in Database Systems. In: *Proc. of VLDB*, (2002).
7. M. Levene and G. Loizou. *A Guided Tour of Relational Databases and Beyond.* Springer-Verlag, London, (1999).
8. A. Lu and W. Ng. *Vague sets or intuitionistic fuzzy sets for handling vague data: Which one is better?* Proc of ER 2005. LNCS Vol 3716, pp. 401–416,(2005).
9. H. Mannila and K-J Raiha. *The Design of Relational Databases.* Addison-Wesley, (1992).
10. W. Ng. Ordered Functional Dependencies in Relational Databases. *Information Systems* **24**(7), pp. 535-554, (1999).
11. W. Ng and M. Levene. The Development of Ordered SQL Packages to Support Data Warehousing. Journal of Database Management **12**(4), pp. 27-49, (2001).
12. W. Ng. An Extension of the Relational Data Model to Incorporate Ordered Domains. *ACM Transactions on Database Systems* **26**(3), (2001).
13. Q. Tan et al. *Applying Co-training to Clickthrough Data for Search Engine Adaptation.* In: Proc. of DASFAA, LNCS Vol 2973, pp. 519-532, (2004).