Journal of Logical and Algebraic Methods in Programming 84 (2015) 456–481

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

An algebra of database preferences

Bernhard Möller *, Patrick Roocks

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

ARTICLE INFO

Article history:
Received 26 May 2013
Received in revised form 31 July 2014
Accepted 19 January 2015
Available online 11 February 2015

Keywords:
Relational algebra
Complex preferences
Preference algebra

ABSTRACT

Preferences allow more flexible and personalised queries in database systems. Evaluation of such a query means to select the maximal elements from the respective database w.r.t. the preference, which is a partial strict-order. We present a point-free calculus of such preferences and exemplify its use in proving algebraic laws about preferences that can be used in query optimisation. We show that this calculus can be mechanised using off-the-shelf automated first-order theorem provers.

1. Introduction

In the field of databases, the relational calculus is a well established discipline which, among other applications, is used in algebraic query optimisation. The classical operations are union, difference, Cartesian product, selection and projection. The queries treated with this calculus mostly pose so-called *hard constraints*, by which the objects sought in the database are clearly and sharply characterised. Hence, if there are no exact matches the empty result set is returned, which is very frustrating for users.

As a remedy, over the last decade queries with *soft constraints* have been studied. These arise from a formalisation of the *user's preferences* in the form of partial strict-orders. A realisation of this idea is presented by the language PREFERENCE SQL [11] and its current implementation [12]. For example, a consumer wants to buy a new car. Her preference relation has two components: she prefers cars with high power and low fuel consumption. Hence, in addition to offering the possibility of specifying simple preferences, one needs flexible and powerful combination operators. With their help hierarchies of user wishes can be handled by *complex preferences*. The bottom of the hierarchy is formed by *base preferences* like "lowest fuel consumption". These are combined using the constructs of the *Pareto* and the *Prioritisation* preferences which model equal and more/less important user preferences. Since the resulting strict-orders are partial, they frequently admit many best or maximal database objects, which helps to avoid empty result sets for queries.

Naturally, also for queries using such preferences efficient optimisation has to be performed, for which an algebraic calculus is indispensable. Although there is by now a well developed set of preference constructors with associated algebraic laws, the underlying theory was based on pointwise definitions with complex first-order formulas, which made proofs of new optimisation rules or the addition of further preference constructors a tedious and error-prone task. The present paper unifies and extends a point-free calculus for preference relations and their laws that has developed over the last two years and is meant to help in resolving the mentioned problems. Not least, it can be easily used with off-the-shelf automated theorem provers, the fact which provides an additional level of trustworthyness.

Before we delve into the technical details, we provide some examples that illustrate the phenomena to be tackled.

* Corresponding author.
E-mail addresses: moeller@informatik.uni-augsburg.de (B. Möller), roocks@informatik.uni-augsburg.de (P. Roocks).

http://dx.doi.org/10.1016/j.jlamp.2015.01.001
2352-2208/

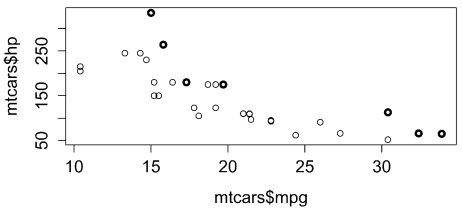

Fig. 1. An example data set of cars where the *horsepower* and the value for *miles per gallon* (inverse fuel consumption) is depicted. The skyline for high power and low consumption is highlighted.

Table 1
Example of a data set of cars.

Model	Fuel	Power	Colour
BMW 5	11.4	230	silver
Mercedes E	12.1	275	black
Audi 6	12.7	225	red

Example 1.1. We return to the sketched example concerning cars. The goals the user wants to achieve are *conflicting*, because cars with high power tend to have a higher fuel consumption. To get the optimal results according to both of these equally important goals from a database, the concept of *skyline queries* [3] is used: A car belongs to the result set if there is no other car which is better in both criteria, i.e. has a lower fuel consumption and a higher power. In a 2-dimensional diagram for both criteria the result set looks like a "skyline", viewed from the hypothetical optimum. Fig. 1 shows such a skyline for the preference mentioned above, where the *mtcars* data set was taken, a standard example in the statistical computing language "R".

Concretely, consider the data set in Table 1. The skyline query for minimal fuel consumption and maximal power returns the "BMW 5" and "Mercedes E", because each of these is better than the other by one criterion. The "Audi 6" is not returned, as it is worse by both criteria. □

Imagine that a large database, for example a catalogue containing all the cars for the European market, returns a quite large result for the above skyline query. Assume that the consumer has even more wishes, for example prefers cars with a specific colour, but this is *less important* than the preference for low fuel and high power. This is formalised as follows.

Example 1.2. In the abstract notation of Preference SQL, the preference for "Lowest fuel consumption and (equally important) highest power, *both more important than* a preference for black cars" can be expressed by

$$P = (\text{LOWEST}(\mathit{fuel}) \otimes \text{HIGHEST}(\mathit{power})) \,\&\, \text{POS}(\mathit{colour}, \{\text{black}\}) \,,$$

where LOWEST and HIGHEST induce the "$<$" and "$>$" orders on their respective numerical domains, while POS creates a preference for values contained in the given set on a discrete domain. Pareto-composition and Prioritisation are denoted by $\otimes$ and & and are defined precisely later on; they might be pronounced "as well as" and "and then", respectively. □

In PREFERENCE SQL many base preferences have the nice property of being *layered* which means that their elements can be grouped into level in each of which the elements are pairwise incomparable. In order theory they are called *strict weak orders*, and prioritisation preserves that property. This allows fast algorithms and a very intuitive way to define *equally good* results: The incomparability relation w.r.t. a layered preference is an equivalence relation. Unfortunately the Pareto preference constructor does not preserve layeredness. This is the technical reason for a counter-intuitive effect which occurs in Preference SQL, shown in the following example.

Example 1.3. The best objects according to P from Example 1.2 in the data set of Table 1 are again "BMW 5" and "Mercedes E". This is quite counterintuitive, because the preference for black cars should decide for the Mercedes only. □

After these motivating examples we present the pointfree calculus of preferences developed in [20] and [18], extended by some additional material.

As our first contribution, we enrich the standard theory of relational databases by an algebraic framework that allows completely point-free reasoning about (complex) preferences and their best matches. This "black-box view" is amenable to a treatment in first-order logic and hence to fully automated proofs using off-the-shelf verification tools. We exemplify the use of the calculus with some non-trivial laws, notably concerning so-called preference prefilters (introduced in [6]), which perform a preselection to speed up the computation of the best matches proper, in particular, for queries involving expensive join operations. It turns out that the original laws hold under much weaker assumptions; moreover, several new ones are derived.