Contracting preference relations for database applications[☆]

Denis Mindolin, Jan Chomicki[*]

Department of Computer Science and Engineering
201 Bell Hall, University at Buffalo
Buffalo, NY 14260-2000, USA

Abstract

The binary relation framework has been shown to be applicable to many real-life preference handling scenarios. Here we study preference contraction: the problem of discarding selected preferences. We argue that the property of minimality and the preservation of strict partial orders are crucial for contractions. Contractions can be further constrained by specifying which preferences should be protected. We consider preference relations that are finite or finitely representable using preference formulas. We present algorithms for computing minimal and preference-protecting minimal contractions for finite as well as finitely representable preference relations. We study relationships between preference change in the binary relation framework and belief change in the belief revision theory. We evaluate the proposed algorithms experimentally and present the results.

Key words: preference contraction, preference change, preference query

1. Introduction

A large number of preference handling frameworks have been developed [16, 7, 20]. In this paper, we work with the *binary relation* preference framework [10, 22]. Preferences are represented as ordered pairs of tuples, and sets

[☆]Research partially supported by NSF grant IIS-0307434. This paper is an extended version of [23].

[*]Corresponding author.

Email addresses: `mindolin@buffalo.edu` (Denis Mindolin), `chomicki@buffalo.edu` (Jan Chomicki)

of preferences form *preference relations.* Preference relations are required to be *strict partial orders (SPO)*: transitive and irreflexive binary relations. The SPO properties are believed to capture the rationality of preferences [16]. This framework can deal with finite as well as infinite preference relations, the latter represented using finite *preference formulas.*

Working with preferences in any framework, it is naive to expect that they never change. Preferences can change over time: if one likes something now, it does not mean one will still like it in the future. Preference change is an active topic of current research [11, 17]. It was argued [15] that along with the discovery of sources of preference change and elicitation of the change itself, it is important to preserve the correctness of the preference model in the presence of change. In the binary relation framework, a natural correctness criterion is the preservation of SPO properties of preference relations.

An operation of preference change – preference revision – has been proposed in [11]. We note that when a preference relation is changed using a revision operator, new preferences are "semantically combined" with the original preference relation. However, combining new preferences with the existing ones is not the only way people change their preferences in real life. Another very common operation of preference change is "semantic subtraction" from a set of preferences another set of preferences one used to hold, if the reasons for holding the *contracted* preferences are no longer valid. That is, we are given an initial preference relation $\succ$ and a subset CON of $\succ$ (called here a *base contractor*) which should not hold. We want to change $\succ$ in such a way that CON does not hold in it. This is exactly opposite to the way the preference revision operators change preference relations. Hence, such a change cannot be captured by the existing preference revision operators.

In multi-agent scenarios, a negotiation between different agents may involve *giving up individual agents' preferences* [1]. In more complex scenarios, preferences may be added as well as given up.

Another reason for contracting user preferences in real-life applications is the need for *widening preference query results.* In many database applications, preference relations are used to compute sets of the best (i.e. the most preferred) tuples, according to user's preferences. Such tuples may represent objects like cars, books, cameras etc. The operator which is used in the binary relation framework to compute such sets is called *winnow* [10] (or *BMO* in [22]). The winnow operator is denoted as $w_{\succ}(r)$, where r is the original set of tuples, and $\succ$ is a preference relation. If the preference relation $\succ$ is large (i.e. the user has many preferences), the result of $w_{\succ}(r)$ may be too narrow. One way to widen the result is by discarding some preferences in $\succ$. Those may be the preferences which do not hold any more or are not longer important.

In this paper, we address the problem of *contraction* of preference relations. We consider it for finitely representable and finite preference relations. We illustrate now preference contraction for finite (Example 1) and finitely representable (Example 2) preference relations.

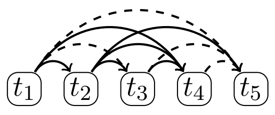


Figure 1: Example 1. Mary's preferences

Example 1 *Assume a car dealer has a web site showing his inventory of cars, and Mary is a customer interested in buying a car. Assume also that Mary has a previous purchase history with the dealer, so her preferences (possibly outdated) over cars are known: she prefers every car t_i to every car t_j (denoted $t_i \succ_1 t_j$) with $i > j$ ($i, j \in [1, 5]$). Let the inventory r_1 consist of four cars ($r_1 = \{t_1, t_3, t_4, t_5\}$), while t_2 is currently missing. The preference relation is illustrated in Figure 1 by the set of all edges, where an edge from t_i to t_j shows that t_i is preferred to t_j. The set of the best cars according to Mary's preference relation is $w_{\succ_1}(r_1) = \{t_1\}$.*

Assume that the dealer observes that while Mary is browsing the web site, she indicates equal interest in three cars: t_1 (as expected according to $\succ_1$), t_3, and t_5. As a result, her preference relation $\succ_1$ has to be changed so that t_1, t_3, and t_5 are all among the best cars, i.e., they must not be dominated by any car in the inventory. That implies that the preferences in the set CON_1 consisting of the following preferences: the preference of t_1 over t_3, and the preference of t_1, t_3, and t_4 over t_5 do not hold any more and need to be contracted *(removed from $\succ_1$). Those preferences are shown as dashed arrows in Figure 1. Notice that since t_2 is not in the inventory, and Mary has not explicitly provided any information regarding her preferences involving t_2, the preferences of t_1 over t_2 and t_2 over t_3, t_4 and t_5 remain unchanged.*

In the example above, we showed a simple scenario of preference contraction. The user preference relation there is a finite relation; and preferences to be contracted are *elicited* from the user-provided feedback. Variations of this scenario are possible. First, the user's preference relation may be *infinite* but representable by a finite preference formula. Second, a possibly infinite set of preferences to discard may also be defined by a formula.

Example 2 *Assume that Bob prefers newer cars, and given two cars made in the same year, the cheaper one is preferred.*

$$t \succ_2 t' \equiv t.year > t'.year \vee t.year = t'.year \wedge t.price < t'.price$$

where $>, <$ *denote the standard orderings of rational numbers, the attribute "year" defines the year when the car was made, and the attribute "price" – its price. The information about all cars which are in stock now is shown in the table* r_2 *below:*

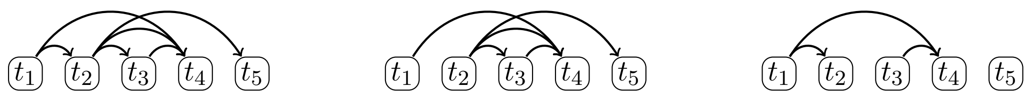
id	*make*	*year*	*price*
t_1	*Kia*	*2007*	*12000*
t_3	*VW*	*2007*	*15000*
t_4	*Kia*	*2006*	*15000*
t_5	*VW*	*2006*	*7000*

Then the set of the most preferred cars according to $\succ_2$ *is* $w_{\succ_2}(r_2) = \{t_1\}$. *Assume that having observed the set* $w_{\succ_2}(r_2)$, *Bob understands that it is too narrow. He decides that the car* t_3 *is not really worse than* t_1. *He generalizes that by stating that the cars made in* 2007 *which cost* 12000 *are not better than the cars made in* 2007 *costing* 15000. *Hence, the set of preferences the user wants to discard can be represented by the relation* CON_2

$$CON_2(t, t') \equiv t.year = t'.year = 2007 \wedge t.price = 12000 \wedge t'.price = 15000.$$

The scenarios illustrated in the examples above have the following in common: we have a (finite or finitely representable infinite) SPO preference relation $\succ$ and a set CON, finite of infinite, of preferences to discard. Our goal is to modify $\succ$, so that the resulting preference relation is an SPO, and the preferences in CON do not hold anymore.

Another important property of preference relation change is *minimality.* Indeed, a simple way of removing a subset of a preference relation without violating its SPO properties is to remove *all* the preferences from this relation. However, most likely it is not what the user expects. Hence, it is important to change the preference relation *minimally.*



(a) Naive contraction (b) Minimal contraction (c) Minimal contraction

Figure 2: Example 3

Example 3 *Take Mary's preferences from Example 1. A naive way to discard CON_1 ($CON_1 = \{t_1t_3, t_1t_5, t_3t_5, t_4t_5\}$) from $\succ_1$ is to represent the contracted preference relation as $\succ'_1 = \succ_1 - CON_1$ (Figure 2(a)). However it turns out that $\succ'_1$ is not transitive (and thus not an SPO): $t_1 \succ'_1 t_2$, $t_2 \succ'_1 t_3$, but $t_1 \not\succ'_1 t_3$; $t_1 \succ'_1 t_2, t_2 \succ'_1 t_5$, but $t_1 \not\succ'_1 t_5$. Hence, this change does not preserve SPO. To make the changed preference relation transitive, some other preferences have to be discarded in addition to CON_1. At the same time, discarding too many preferences is not a good solution since some of them may be important. Therefore, we need to discard a minimal subset of $\succ_1$ which contains CON_1 and preserves SPO in the modified preference relation. Two solutions are possible here: in the first case, we remove the preferences $P_1^- = \{t_1t_2\} \cup CON_1$ (Figure 2(b) shows the contracted relation); in the second – the preferences $P_2^- = \{t_2t_3, t_2t_4, t_2t_5\} \cup CON_1$ (Figure 2(c) shows the contracted relation).*

Similarly, take $\succ_2$ and CON_2 from Example 2. The relation $\succ'_2 \equiv (\succ_2 - CON_2)$ is not transitive: if we take $t_5 = (VW, 2007, 12000)$, $t_6 = (VW, 2007, 14000)$, and $t_7 = (VW, 2007, 15000)$, then $t_5 \succ'_2 t_6$ and $t_6 \succ'_2 t_7$ but $t_5 \not\succ'_2 t_7$. An SPO preference relation which is minimally different from $\succ_2$ and does not contain CON_2 is shown below:

$$\begin{aligned} t \succ^*_2 t' \equiv &(t.y > t'.y \vee t.y = t'.y \wedge t.p < t'.p) \wedge \\ &\neg(t.y = t'.y = 2007 \wedge t.p = 12000 \wedge t'.p > 12000 \wedge t'.p \leq 15000) \end{aligned}$$

As we can see, the relation $\succ^*_2$ is different from the naive solution $\succ'_2$ in the sense that $\succ^*_2$ implies that a car made in 2007 costing 12000 is not better than a car made in 2007 costing from 12000 to 15000. *We note that $\succ^*_2$ is not the only relation minimally different from $\succ_2$ and not containing CON_2.*

The examples above show that when a subset of an SPO preference relation is discarded, the resulting relation may lose its SPO properties: while it is always irreflexive, the transitivity axiom may not be preserved. A possible way to remedy the problem is to relax the SPO requirements imposed on preference relations and allow non-transitive preference relations. However, there is a number of reasons why all SPO axioms are important to preserve. First, the SPO properties are believed to capture the rationality of preferences. The second reason is related to the usage of preferences in database applications: There are many efficient algorithms for preference query evaluation which assume preference relations to be transitive [12, 6].

Example 4 *A popular preference query evaluation algorithm* SFS *[12] works as follows. Given a database table r and a preference relation $\succ$, SFS 1) sorts r according to a weak order consistent with $\succ$, 2) picks every tuple o from r in sorted order and checks if there is any tuple o' in r that appeared* before *o such that $o' \succ o$. If there at least one such tuple, then $o \notin w_{\succ}(r)$ (i.e., not among the best in r according to $\succ$) and is discarded, and otherwise $o \in w_{\succ}(r)$.*

Take $r = \{o_1, o_3\}$, and $\succ = \{o_1 o_2, o_2 o_3\}$ (i.e., not transitive). Applying SFS to r and $\succ$ results in $w_{\succ}(r) = \{o_1, o_3\}$. Note that SFS fails to return the correct answer (which is $w_{\succ}(r) = \{o_1\}$) due to the intransivity of $\succ$: the tuple o_2, the only tuple that dominates o_3, is discarded before it can prevent o_3 from being output.

Hence, relaxing the SPO properties of preference relations would require developing new preference query evaluation algorithms that are likely to be less efficient [8]. Moreover, the approach of contracting preference relations we propose in this paper has the property of *closure*: both the original and the contracted preference relation are SPOs. Closure is important because it makes iterating contraction (or revision [11]) possible.

As illustrated in Examples 1 and 2, the essence of the preference contraction approach we propose here is the following: *when discarding a subset CON of a preference relation $\succ$, some preferences* additional to *CON should be discarded to make the resulting preference relation an SPO*. A subset P^- of $\succ$ which contains CON and whose removal from $\succ$ preserves the SPO properties of the modified preference relation is called a *full contractor of $\succ$ by CON*. The set $\mathcal{P}^m$ of *alternative* minimal full contractors for a given $\succ$ and CON may contain a large or even infinite number of elements. How to perform contraction in such cases?

There are essentially two possibilities:

- *Minimal contraction*: The user does not care which full contractor from $\mathcal{P}^m$ is chosen; the goal is to change the preference relation *minimally*.

- *Meet contraction*: The user does not know (or does not want to reveal) which full contractor from $\mathcal{P}^m$ to choose. So it is only safe to remove *all* $P^- \in \mathcal{P}^m$ from $\succ$. In this case, the minimality of change may be sacrificed.

We notice that the two approaches to preference contraction are similar to *minimal contraction* and *meet contraction* used in belief revision [19]. They are justified by similar reasons. Section 8 contains a comparison analysis of the framework proposed here and preference change in belief revision.

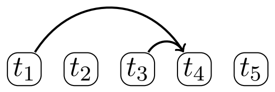


Figure 3: Meet contraction

Example 5 *Consider Example 3 and the contraction of* $\succ_1$ *by* CON_1*. The set of all minimal full contractors* $\mathcal{P}^m$ *of* $\succ_1$ *by* CON_1 *is* $\{P_1^-, P_2^-\}$*. The* meet contraction *corresponds to picking and removing both* P_1^- *and* P_2^- *from* $\succ_1$*. The result of the* meet contraction *is shown in Figure 3.*

The operators of preference contraction – minimal contraction and meet contraction – describe two extreme cases. However, they share an important property: after specifying CON and the type of the contraction (minimal or meet), the user has *no further control* over the result.

To overcome this disadvantage, we also introduce two variants of these contraction operators operators: *preference-protecting minimal contraction* and *preference-protecting meet contraction*. These operators require the user to provide a set of preferences $P^+ \subseteq \succ$ which she believes must hold after the contraction (i.e., none of them should be contracted). This gives the user *limited control* over the result of the contraction.

- *Preference-protecting minimal contraction*: We choose *some* $P^- \in \mathcal{P}^m$ that protects P^+ (i.e., $P^- \cap P^+ = \emptyset$) and remove it from $\succ$. The minimality of change is preserved in this case because P^- is a minimal full contractor (i.e., a member of $\mathcal{P}^m$).

- *Preference-protecting meet contraction*: We choose *all* $P^- \in \mathcal{P}^m$ protecting P^+ (i.e.,$P^- \cap P^+ = \emptyset$) and remove them from $\succ$.

Example 6 *Take the set of minimal contractors $\mathcal{P}^m = \{P_1^-, P_2^-\}$ from Example 3. Assume that the user wants to protect the preference set $P^+ = \{t_2 t_3\}$ from contraction. Out of the full contractors $\{P_1^-, P_2^-\}$, only P_1^- protects it. Hence, the preference-protecting minimal contraction will return P_1^-, and the result of the contraction is shown in Figure 2(b). Similarly, since the set of all minimal full contractors protecting P^+ is a singleton, Figure 2(b) also shows the result of applying the* preference-protecting meet contraction*.*

We observe that the operators of minimal contraction and meet contraction are special cases (i.e., $P^+ = \emptyset$) of preference-protecting minimal contraction and preference-protecting meet contraction, respectively.

Above we showed some simple use cases of preference contraction. In real-life applications, preference contraction can be done in a *step-by-step* manner by collecting user feedback and elaborating contraction: the user can change the sets CON and P^+, undo contraction, or vary the contraction parameters and operators. Such feedback may be collected from the users directly, by asking them questions about relationships of certain objects [4], or indirectly, e.g., by analyzing users' clicks on web pages or critiques of various parameters of objects [9]. However, the details of such usage scenarios are beyond the scope of this work.

The main results of the paper are as follows:

1. We present necessary and sufficient conditions for minimality of full contractors.
2. We propose two algorithms for minimal preference contraction: the first for finitely representable preference relations and the second for finite preference relations. The algorithms require that CON be finitely stratifiable.
3. We show that for the class of preference formulas studied in this paper checking finite stratifiability can be performed using quantifier elimination.
4. We show how to reduce minimal preference-protecting contraction to minimal contraction.
5. We show how meet and preference-protecting meet contraction can be accommodated in our framework.
6. We study the relationship of preference contraction to belief contraction and revision.
7. We perform experimental evaluation of the proposed framework and present the results of the experiments (Appendix A).

2. Basic Notions

The preference relation framework we use in the paper is a variation of the one proposed in [10]. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a fixed set of attributes. Every attribute A_i is associated with a *domain* $\mathcal{D}_i$. We consider here two kinds of infinite domains: C (uninterpreted constants) and Q (rational numbers). Then the universe $\mathcal{U}$ of tuples is defined as

$$\mathcal{U} = \prod_{A_i \in \mathcal{A}} \mathcal{D}_i$$

We assume that two tuples o and o' are equal iff the values of their corresponding attributes are equal.

Definition 1 *A binary relation $\succ \subseteq \mathcal{U} \times \mathcal{U}$ is a* preference relation*, if it is a strict partial order (SPO) relation, i.e., transitive and irreflexive.*

Binary relations $R \subseteq \mathcal{U} \times \mathcal{U}$ considered in the paper are *finite* or *infinite.* Finite binary relations are represented as finite sets of pairs of tuples. The infinite binary relations we consider here are *finitely representable* as *formulas.* Given a binary relation R, its formula representation is denoted by F_R. That is, $R(o, o')$ iff $F_R(o, o')$. A formula representation $F_\succ$ of a preference relation $\succ$ is called a *preference formula.*

We consider two kinds of atomic formulas here:

- *equality constraints*: $o.A_i = o'.A_i$, $o.A_i \neq o'.A_i$, $o.A_i = c$, or $o.A_i \neq c$, where o, o' are tuple variables, A_i is a C-attribute, and c is an uninterpreted constant;
- *rational-order constraints*: $o.A_i \, \theta \, o'.A_i$ or $o.A_i \, \theta \, c$, where $\theta \in \{=, \neq, <, >, \leq, \geq\}$, o, o' are tuple variables, A_i is a Q-attribute, and c is a rational number.

A preference formula whose all atomic formulas are equality (resp. rational-order) constraints will be called an *equality* (resp. *rational order*) preference formula. If both equality and rational order constraints are used in a formula, the formula will be called an *equality/rational-order* formula or simply *ERO* -formula. Without loss of generality, we assume that all preference formulas are quantifier-free because ERO-formulas admit quantifier elimination.

We also use the representation of binary relations as *directed graphs*, both in the finite and the infinite case.

Definition 2 *Given a binary relation $R \subseteq \mathcal{U} \times \mathcal{U}$ and two tuples x and y such that xRy ($xy \in R$), xy is an* R-edge *from x to y. A* path in R *(or an* R-path*) from x to y is a finite sequence of R-edges such that the start node of the first edge is x, the end node of the last edge is y, the end node of every edge (except the last one) is the start node of the next edge in the sequence, and no R-edge appears more than once in it. The sequence of nodes participating in an R-path is an* R-sequence. *The* length of an R-path *is the number of R-edges in the path. The* length of an R-sequence *is the number of nodes in it.*

An element of a preference relation is called *a preference*. We use the symbol $\succ$ with subscripts to refer to preference relations. We write $x \succeq y$ as a shorthand for $(x \succ y \vee x = y)$. We also say that x *is preferred to* y and y *is dominated by* x according to $\succ$ if $x \succ y$.

In this paper, we present several algorithms for finite relations. Such algorithms are implemented using the *relational algebra* operators: selection σ, projection π, join $\bowtie$, set difference $-$, and union $\cup$ [24]. Set difference and union in relational algebra have the same semantics as in set theory, provided the argument relations are compatible. The semantics of the other operators are as follows:

- Selection $\sigma_C(R)$ picks from the relation R all the tuples for which the condition C holds. The condition C is a boolean expression involving comparisons between attribute names and constants.
- Projection $\pi_L(R)$ returns a relation which is obtained from the relation R by leaving in it only the columns listed in L and dropping the others.
- Join of two relations R and S

$$R \underset{R.X_1=S.Y_1,\ldots,R.X_n=S.Y_n}{\bowtie} S$$

  computes a product of R and S, leaves only the tuples in which $R.X_1 = S.Y_1, \ldots, R.X_n = S.Y_n$, and drops the columns $S.Y_1, \ldots, S.Y_n$ from the resulting relation.

When we need more than one copy of a relation R in a relational algebra expression, we add subscripts to the relation name (e.g. R_1, R_2 etc).

3. Preference contraction

Preference contraction is an operation of discarding preferences. We assume that when the user intends to discard some preferences, he or she expresses the preferences to be discarded as a binary relation called a *base contractor*. The interpretation of each pair in a base contractor is that the first tuple should not be preferred to the second tuple. We require base contractor relations to be subsets of the preference relation to be contracted. Hence, a base contractor is irreflexive but not necessary transitive. Apart from the containment in the original preference relation, we impose no other restrictions on the base contractors (e.g., they can be finite or infinite), unless stated otherwise. Throughout the paper, base contractors are typically referred to as CON.

Definition 3 *A binary relation P^- is* a full contractor of a preference relation $\succ$ by CON *if* $CON \subseteq P^- \subseteq \succ$*, and* $(\succ - P^-)$ *is a preference relation (i.e., an SPO). The relation* $(\succ - P^-)$ *is called the* contracted relation*.*

A relation P^- is a minimal full contractor of $\succ$ by CON *if P^- is a full contractor of $\succ$ by CON, and there is no other full contractor P' of $\succ$ by CON s.t. $P' \subsetneq P^-$.*

Definition 4 *A preference relation is* minimally contracted *if it is contracted by a* minimal full contractor*.* Contraction *is the operation of constructing a full contractor.* Minimal contraction *is the operation of constructing a minimal full contractor.*

We notice that the requirement of CON being a subset of $\succ$ introduced above is imposed solely for the sake of simplifying the discussion. Indeed, if according to the user preference relation $\succ$, a tuple o is not better than o', then removing the preference of o over o' from $\succ$ is trivial. Moreover, our definition of preference contraction guarantees that such a preference cannot appear in the contracted $\succ$, because a contracted preference relation is always a subset of the original one. Therefore, contracting $\succ$ by some CON^* not contained in $\succ$ is equivalent to contracting $\succ$ by $CON \ = \succ \cap CON^*$. At the same time, if preference contraction is used in conjunction with preference revision [11] (which may result in adding new preference to the revision preference relation), some special care has to be taken of the preferences in $(CON^* - \succ)$. However, the discussion of such techniques is outside of the scope of the current paper.

According to Definition 3, minimality of full contractor is defined in terms of *set containment*. Obviously, other definitions are possible. For instance, minimality can be defined in terms of the cardinality of a full contractor. However, in this paper, we focus on developing techniques of preference contraction which would work for for *finite* as well as *finitely representable* (i.e., possibly infinite) preference relations. It is clear that the minimality-as-minimum-cardinality criterion cannot be used in the latter case.

The notion of a minimal full contractor narrows the set of full contractors. However, as we illustrate in Example 7, a minimal full contractor is generally not unique for the given preference and base contractor relations. Moreover, the number of minimal full contractors for infinite preference relations can be infinite. Thus, minimal contraction differs from minimal preference revision [11] which is uniquely defined for given preference and revising relations.

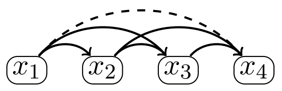


Figure 4: $\succ$ and CON from Example 7

Example 7 *Take the preference relation $\succ$ which is a total order of $\{x_1, \ldots, x_4\}$ (Figure 4). Let the base contractor relation CON be $\{x_1x_4\}$. Then the following sets are minimal full contractors of $\succ$ by CON: $P_1^- = \{x_1x_2, x_1x_3, x_1x_4\}$, $P_2^- = \{x_3x_4, x_2x_4, x_1x_4\}$, $P_3^- = \{x_1x_2, x_3x_4, x_1x_4\}$, and $P_4^- = \{x_1x_3, x_2x_4, x_2x_3, x_1x_4\}$.*

The number of minimal full contractors can be rather large. As the following example illustrates, it is in some cases exponential in the number of edges in base contractor.

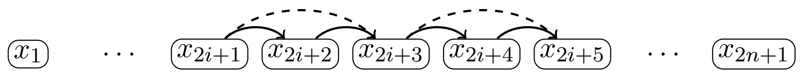


Figure 5: $\succ$ and CON from Example 8. Transitive edges are omitted

Example 8 *Let a preference relation $\succ$ be a total order of $\{x_1, \ldots, x_{2n+1}\}$ for some n (i.e., $x_i \succ x_j$ for $1 \leq i < j \leq 2n+1$), and $CON = \{x_{2i+1}x_{2i+3} \mid i \in [0, n-1]\}$, consisting of n edges. To remove an edge $x_{2i+1}x_{2i+3}$ from $\succ$ and make the resulting relation transitive, we also need to remove either $x_{2i+1}x_{2i+2}$*

or $x_{2i+2}x_{2i+3}$ from it. Thus, we have 2^n possible full contractors. It is easy to show that each of them is minimal.

Another important observation here is that that the contracted preference relation is defined as a subset of the original preference relation. We want to preserve the SPO properties – transitivity and irreflexivity – of preference relations. Since any subset of an irreflexive relation is also an irreflexive relation, no additional actions are needed to preserve irreflexivity during contraction. However, not every subset of a transitive relation is a transitive relation. We need to consider paths in the original preference relation, which, by transitivity, may produce CON-edges to be discarded. We call such paths *CON-detours*.

Definition 5 *Let $\succ$ be a preference relation, and $P \subseteq \succ$. Then a $\succ$-path from x to y is a P-*detour *if $xy \in P$.*

First, let us consider the problem of finding any full contractor, not necessary a minimal one. As we showed above, a contracted preference relation cannot have any CON-detours. To achieve that, some additional edges of the preference relation have to be discarded. However, when we discard these edges, we have to make sure that there are no paths in the contracted preference relation which produce the removed edges. Hence, a necessary and sufficient condition for a subset of a preference relation *to be its full contractor* can be formulated in an intuitive way.

Lemma 1 *Given a preference relation (i.e., an SPO) $\succ$ and a base contractor CON, a relation $P^- \subseteq \succ$ is a full contractor of $\succ$ by CON iff $CON \subseteq P^-$, and for every $xy \in P^-$, $(\succ - P^-)$ contains no paths from x to y.*

Proof.
$\boxed{\Leftarrow}$ Prove that if for all $xy \in P^-$, $(\succ - P^-)$ contains no paths from x to y, then $(\succ - P^-)$ is an SPO. The irreflexivity of $(\succ - P^-)$ follows from the irreflexivity of $\succ$. Assume $(\succ - P^-)$ is not transitive, i.e., there are $xz, zy \in (\succ - P^-)$ but $xy \notin (\succ - P^-)$. If $xy \in P^-$ then the path xz, zy is not disconnected, which contradicts the initial assumption. If $xy \notin P^-$, then the assumption of transitivity of $\succ$ is violated.
$\boxed{\Rightarrow}$ First, $CON \not\subseteq P^-$ implies that P^- is not a full contractor of $\succ$ by CON by definition. Second, the existence of a path from x to y in $(\succ - P^-)$ for

$xy \in P^-$ implies that $(\succ - P^-)$ is not transitive, which violates the SPO properties. □

Now let us consider the property of minimality of full contractors. Let P^- be any minimal full contractor of a preference relation $\succ$ by a base contractor CON. Pick any edge xy of P^-. An important question which arises here is *why is xy a member of P^-?* The answer is obvious if xy is also a member of CON: every CON-edge has to be removed from the preference relation. However, what if xy is not a member of CON? To study this problem, we introduce the notion of the *outer edge set* of an edge belonging to a full contractor relation.

Definition 6 *Let CON be a base contractor of a preference relation $\succ$, and P^- be a full contractor of $\succ$ by CON. Let $xy \in (P^- - CON)$, and*

$$\Phi_0(xy) = \{xy\}, \text{ and}$$
$$\Phi_i(xy) = \{u_iv_i \in P^- | \exists u_{i-1}v_{i-1} \in \Phi_{i-1}(xy) \ . \ u_i = u_{i-1} \wedge v_{i-1}v_i \in (\succ - P^-) \vee v_{i-1} = v_i \wedge u_iu_{i-1} \in (\succ - P^-)\}, \text{ for } i > 0.$$

Then the outer edge set $\Phi(xy)$ for xy *is defined as*

$$\Phi(xy) = \bigcup_{i=0}^{\infty} \Phi_i(xy).$$

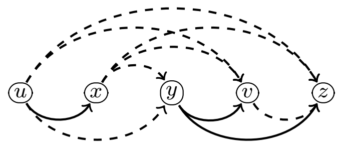


Figure 6: $\Phi(xy)$ for Example 9.

Intuitively, the outer edge set $\Phi(xy)$ of an edge $xy \in (P^- - CON)$ contains all the edges of a full contractor P^- which should be removed from P^- (i.e., added back to the preference relation $\succ$) to preserve the full contractor property of the result, should xy be removed from P^- (i.e., added back to the preference relation). The reasoning here is as follows. When for some i, $\Phi_i(xy)$ is removed from P^-, then $\Phi_{i+1}(xy)$ has also to be removed from P^-. Otherwise, for every edge in $\Phi_{i+1}(xy)$, there is a two-edge path in $\succ$, one of whose edges is in $\Phi_i(xy)$ while the other is not contracted. Hence, if the SPO properties of $(\succ - P^-)$ need to be preserved, removing xy from P^- requires recursively removing the entire $\Phi(xy)$ from P^-.

The next example illustrates the inductive construction of an outer edge set. Some properties of outer edge sets are shown in Lemma 2.

Example 9 *Let a preference relation $\succ$ be the set of all edges in Figure 6, and P^- be defined by the dashed edges. Let us construct $\Phi(xy)$ (assuming that xy is not an edge of the base contractor CON).*

- $\Phi_0(xy) = \{xy\}$*;*
- $\Phi_1(xy) = \{xv, xz\}$*;*
- $\Phi_2(xy) = \{uv, uz\}$*;*

Thus, $\Phi(xy) = \{xy, xv, xz, uv, uz\}$.

Lemma 2 *Let P^- be a full contractor of a preference relation $\succ$ by a base contractor CON. Then for every $xy \in (P^- - CON)$, $\Phi(xy)$ has the following properties:*

1. *for all $uv \in \Phi(xy)$, $u \succeq x$ and $y \succeq v$;*
2. *for all $uv \in \Phi(xy)$, $ux, yv \notin P^-$;*
3. *if $(\Phi(xy) \cap CON) = \emptyset$, then $P' = (P^- - \Phi(xy))$ is a full contractor of $\succ$ by CON.*

Proof. First, we prove that Properties 1 and 2 hold. We do it by induction on the index of $\Phi_i(xy)$ used to construct $\Phi(xy)$. Since by definition $\{xy\} = \Phi_0(xy)$, Properties 1 and 2 hold by the construction of Φ_0. Now let Properties 1 and 2 hold for $\Phi_n(xy)$, i.e.,

$$\forall u_n v_n \in \Phi_n(xy) \rightarrow u_n \succeq x \wedge y \succeq v_n \wedge u_n x, y v_n \notin P^- \qquad (1)$$

Pick any $u_{n+1}v_{n+1} \in \Phi_{n+1}(xy)$. By construction of $\Phi_{n+1}(xy)$, we have

$$\begin{aligned}\exists u_n v_n \in \Phi_i(xy)\ .\ & u_{n+1} = u_n \wedge v_n \succ v_{n+1} \wedge v_n v_{n+1} \notin P^- \vee \\ & u_{n+1} \succ u_n \wedge v_n = v_{n+1} \wedge u_{n+1} u_n \notin P^- \end{aligned} \qquad (2)$$

Note that $u_{n+1} \succeq x$ and $y \succeq v_{n+1}$ follows from (1), (2), and transitivity of $\succeq$. Similarly, $u_{n+1}x, yv_{n+1} \notin P^-$ is implied by (1), (2), and transitivity of $(\succ - P^-)$. Hence, Properties 1 and 2 hold for $\cup_{i=0}^{n}\Phi_i(xy)$ for any n.

Now we prove Property 3: $(\succ - P')$ is an SPO and $CON \subseteq P'$. The latter follows from $CON \subseteq P^-$ and $\Phi(xy) \cap CON = \emptyset$. Irreflexivity of $(\succ - P')$ follows from irreflexivity of $\succ$. Assume $(\succ - P')$ is not transitive, i.e., there are $uv \notin (\succ - P')$ and $uz, zv \in (\succ - P')$. Transitivity of $(\succ - P^-)$ implies that at least one of uz, zv is in $\Phi(xy)$. However, Property 1 implies that *exactly one* of uz, zv is in $\Phi(xy)$ and the other one is not in $\Phi(xy)$ and thus in $(\succ - P^-)$. However, $uz \in \Phi(xy)$ and $zv \in (\succ - P^-)$ imply $uv \in \Phi(xy)$, and thus $uv \in (\succ - (P^- - \Phi(xy))) = (\succ - P')$, i.e., we derive a contradiction. A similar contradiction is derived in the case $uz \in (\succ - P^-)$ and $zv \in \Phi(xy)$. Therefore, $(\succ - P')$ is an SPO and P' is a full contractor of $\succ$ by CON. □

Out of the three properties shown in Lemma 2, the last one is the most important. It says that if an edge xy of a full contractor is not needed to disconnect any CON-detours, then that edge may be dropped from the full contractor along with its entire outer edge set. A more general result which follows from Lemma 2 is formulated in the next theorem. It represents a necessary and sufficient condition for a full contractor to be *minimal*.

Theorem 1 (Full-contractor minimality test). *Let P^- be a full contractor of $\succ$ by CON. Then P^- is a* minimal *full contractor of $\succ$ by CON iff for every $xy \in P^-$, there is a CON-detour in $\succ$ in which xy is the only P^--edge.*

PROOF.
$\boxed{\Leftarrow}$ The proof in this direction is straightforward. Assume that for every edge of the full contractor P^- there exists at least one CON-detour in which only that edge is in P^-. If P^- loses a subset P containing that edge, then there is a CON-detour in $\succ$ having no edges in $(P^- - P)$, and thus $(P^- - P)$ is not a full contractor of $\succ$ by CON by Lemma 1. Hence, P^- is a minimal full contractor.
$\boxed{\Rightarrow}$ Let P^- be a minimal full contractor. For the sake of contradiction, assume for some $xy \in P^-$, 1) there is no CON-detour which xy belongs to, or 2) any CON-detour xy belongs to has at least one more P^--edge. If 1) holds, then $\Phi(xy)$ has no edges in CON by construction. Thus, Lemma 2 implies that $(P^- - \Phi(xy))$ is a full contractor of $\succ$ by CON. Since $\Phi(xy)$

is not empty, we get that P^- is not a minimal full contractor which is a contradiction. If 2) holds, then we use the same argument as above and show that $\Phi(xy) \cap CON = \emptyset$. If $\Phi(xy) \cap CON$ is not empty (i.e., some $uv \in \Phi(xy) \cap CON$), then by Lemma 2,

$$u \succeq x \land x \succ y \land y \succeq v \land ux, yv \notin P^-,$$

and thus there is a CON-detour going from u to v in which xy is the only P^--edge. This contradicts the initial assumption. $\square$

Note that using directly Definition 3 to check the minimality of a full contractor P^- requires checking the full contractor properties of *all subsets* of P^-. In contrast, the minimality checking method shown in Theorem 1 requires checking properties of *distinct elements* of P^- with respect to its other members.

Sometimes a direct application of the minimality test from Theorem 1 is hard because it does not give any bound on the length of CON-detours. Hence, it is not clear how the test can be formulated in terms of validity of finite formulas. Fortunately, the transitivity of preference relations implies that the minimality condition from Theorem 1 can be stated in terms of paths of length at most three.

Corollary 1 *A full contractor P^- of $\succ$ by CON is* minimal *iff for every edge $xy \in P^-$, there is a CON-detour consisting of at most three edges among which only xy is in P^-.*

PROOF.
$\boxed{\Leftarrow}$ Trivial.
$\boxed{\Rightarrow}$ For every $xy \in P^-$, pick any CON-detour T in which the only P^--edge is xy. If its length is less or equal to three, then the corollary holds. Otherwise, x is not the start node of T, or y is not the end node of T, or both. Let the start node u of T be different from x. Since the only common edge of T and P^- is xy, every edge in the path from u to x is an element of $(\succ - P^-)$. Transitivity of $(\succ - P^-)$ implies $ux \in (\succ - P^-)$. Similarly, $yv \in (\succ - P^-)$ for the end node of T if y is different from v. Hence, there is a CON-detour of length at most three in which xy is the only element of P^-. $\square$

As a result, the following tests can be used to check the minimality of a full contractor P^-. In the finite case, P^- is minimal if the following relational algebra expression results in an empty set

$$\begin{aligned}\text{P - }[&\pi_{\text{P}_2.\text{X},\text{P}_2.\text{Y}}((\text{R}_1 - \text{P}_1) \underset{\text{R}_1.\text{Y}=\text{P}_2.\text{X}}{\bowtie} \text{P}_2 \underset{\text{P}_2.\text{Y}=\text{R}_3.\text{X}}{\bowtie} (\text{R}_3 - \text{P}_3) \underset{\text{R}_1.\text{X}=C.X,\ \text{R}_3.\text{Y}=\text{C.Y}}{\bowtie} \\ &\text{C}) \cup \pi_{\text{P}_2.\text{X},\text{P}_2.\text{Y}}(\text{P}_2 \underset{\text{P}_2.\text{Y}=\text{R}_3.\text{X}}{\bowtie} (\text{R}_3 - \text{P}_3) \underset{\text{P}_2.\text{X}=C.X,\ \text{R}_3.\text{Y}=\text{C.Y}}{\bowtie} \text{C}) \cup \\ &\pi_{\text{P}_2.\text{X},\text{P}_2.\text{Y}}((\text{R}_1 - \text{P}_1) \underset{\text{R}_1.\text{Y}=\text{P}_2.\text{X}}{\bowtie} \text{P}_2 \underset{\text{R}_1.\text{X}=C.X,\ \text{P}_2.\text{Y}=\text{C.Y}}{\bowtie} \text{C}) \cup \text{C}],\end{aligned}$$

for the tables R, C and P with columns X and Y, storing $\succ$, CON, and P^- correspondingly. R_1, R_3, and P_1, P_2, P_3 refer to renamings of R and P, respectively. Applying the minimality test to finite relations is illustrated in the next example.

Example 10 *Take a preference relation represented by the table R, and a base contractor represented by the table C (Figure 7(a)). Consider the table P representing a full contractor of R by C. Then the result of the relational algebra expression above evaluated for these tables is shown in the table D. Since it is not empty, the full contractor represented by P is not minimal. The minimality of P can be achieved by removing from it any (but only one) tuple from D.*

In the finitely representable case, P^- is minimal if the following formula is valid

$$\begin{aligned}\forall x, y\ (F_{P^-}(x,y) \Rightarrow F_\succ(x,y) \wedge \exists u, v\ .\ F_{CON}(u,v) \wedge (F_\succ(u,x) \vee u = x) \wedge \\ (F_\succ(y,v) \vee y = v) \wedge \neg F_{P^-}(u,x) \wedge \neg F_{P^-}(y,v)).\end{aligned}$$

We note that when the relations are definable using ERO-formulas, checking minimality of a full contractor can be done by performing quantifier elimination on the above formula.

Example 11 *Let a preference relation $\succ$ be defined by the formula $F_\succ(o,o') \equiv o.d < o'.d$, where d is a Q -attribute. Let a base contractor CON of $\succ$ be defined by the formula*

$$F_{CON}(o,o') \equiv (1 \le o.d \le 2 \wedge o'.d = 4) \vee (o.d = 0 \wedge o'.d = 3)$$

(Figure 7(b)). Denote the relation represented by the first and second disjuncts of F_{CON} as CON_1 and CON_2 correspondingly. The relation P^- defined by F_{P^-} is a full contractor of $\succ$ by CON

$$F_{P^-}(o,o') \equiv (1 \le o.d \le 2 \wedge 2 < o'.d \le 4) \vee (o.d = 0 \wedge 0 < o'.d \le 3).$$

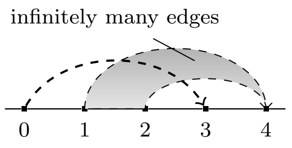
R	X	Y
	u	x
	x	y
	y	v
	u	y
	x	v
	u	v

C	X	Y
	u	v

P	X	Y
	u	x
	y	v
	x	v
	u	v

D	X	Y
	u	x
	x	v

(a) Finite case, Example 10



(b) Infinite case, Example 11

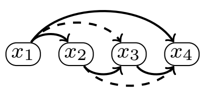
Figure 7: Checking minimality of a full contractor

Similarly, denote the relations represented by the first and the second disjuncts of F_{P^-} as P_1^- and P_2^- correspondingly. We use Corollary 1 to check the minimality of P^-. By the corollary, we need to consider CON-detours of length at most three. Note that every P_1^--edge starts a one- or two-edge CON-detour with the corresponding CON_1-edge. Moreover, the second edge of all such two-edge detours is not contracted by P^-. Hence, the minimal full contractor test is satisfied for P_1^--edges. Now we consider P_2^--edges. All CON-detours, which these edges belong to, correspond to CON_2-edges and are started by P_2^--edges. Hence, we need to consider only CON_2-detours of length at most two. When a P_2^--edge ends in o' with the value of d in $(0,1)$ and $(2,3]$, the second edge in the corresponding two-edge CON_2-detour is not contracted by P^-. However, when d is in $[1,2]$, the second edge is already in P^-. Hence, P^- is not minimal by Corollary 1. To minimize it, we construct P^* by removing the edges from P^- which end in o' with d in $[1,2]$

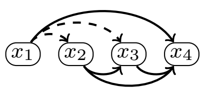
$$\begin{aligned} F_{P^*}(o,o') \equiv\ & (1 \leq o.d \leq 2 \land 2 < o'.d \leq 4) \lor \\ & (o.d = 0 \land (0 < o.d' < 1 \lor 2 < o'.d \leq 3)) \end{aligned}$$

4. Construction of a minimal full contractor

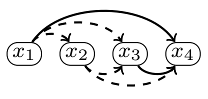
In this section, we propose a method of computing a minimal full contractor. An approach for minimally contracting a preference relation $\succ$ by CON that seems intuitive is *incremental contraction.* Namely, one may try to partition CON arbitrarily into subsets $CON = \cup_{i=0}^{n} CON_i$, and in every i-th iteration ($i = 0, \ldots, n$), compute a minimal full contractor P_i^- of the intermediate preference relation $(\succ - P_{i-1})$ by CON_i (given that $P_{-1} = \emptyset$), expecting that $P^- = \cup_{i=0}^{n} P_i^-$ will be a minimal full contractor of $\succ$ by the entire CON. However, in the following example we show that to guarantee the minimality of P^-, the ways in which CON is partitioned and the intermediate preference relations are contracted by individual partitions have to be chosen carefully.



(a) Preference relation $\succ$



(b) Preference relation $(\succ - P_0^-)$



(c) Preference relation $\succ - (P_0^- \cup P_1^-)$

Figure 8: Preference contraction

Example 12 *Let the preference relation $\succ$ be a total order shown in Figure 8(a), and a base contractor CON be $\{x_1x_3, x_2x_4\}$. Let us partition CON into $CON_0 = \{x_1x_3\}$ and $CON_1 = \{x_2x_4\}$. Then a minimal full contractor P_0^- of $\succ$ by CON_0 is $\{x_1x_3, x_1x_2\}$, and a* minimal *full contractor P_1^- of $\succ -P_0^-$ is $\{x_2x_4, x_2x_3\}$. However, the relation $P^- = P_0^- \cup P_1^-$ is* not *a minimal full contractor of $\succ$ by CON because its subset $\{x_1x_3, x_2x_4, x_2x_3\}$ is a full contractor.*

In the algorithms for computing minimal full contractors proposed in this section, we essentially follow the approach described above. First, we show a method of computing a full contractor of $\succ$ by CON. Then we show why such full contractor may fail to be minimal. Subsequently, we propose a method for partitioning CON into *strata,* such that an iterative contraction of $\succ$ stratum-by-stratum results in a minimal full contractor.

4.1. Stratification of base contractor

We illustrate the idea of computing full contractors using the set P_1^- from Example 7. The set P_1^- was constructed as follows: we took the CON-edge x_1x_4 and put in P_1^- all the edges which start some path from x_1 to x_4. For the preference relation $\succ$ in Example 7, P_1^- turned out to be a minimal full contractor. As shown in the next lemma, the set consisting of all edges starting CON-detours is a full contractor by CON.

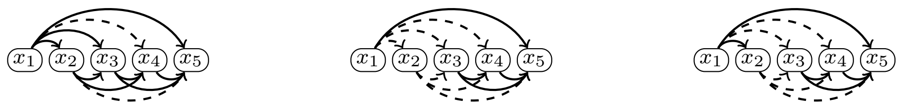
Lemma 3 *Let $\succ$ be a preference relation and CON be a base contractor relation of $\succ$. Then*

$$P^- := \{\ xy \mid \exists x'v \in CON\ .\ x' = x \wedge x' \succ y \wedge y \succeq v\}$$

is a full contractor of $\succ$ by CON.

PROOF. By construction of P^-, $CON \subseteq P^-$. Lemma 1 implies that $(\succ - P^-)$ is an SPO. Indeed, given any $xy \in P^-$, every path from x to y is disconnected by its starting edge. Hence, P^- is a full contractor of $\succ$ by CON. □

However, in the next example we show that such a full contractor is not always minimal. Recall that by Theorem 1, for every edge of a minimal full contractor there should be a CON-detour which shares only that edge with the contractor. However, it may be the case that an edge starting a CON-detour does not have to be discarded because the CON-detour is already disconnected by another edge of the full contractor.



(a) Preference relation $\succ$ (b) $(\succ - P^-)$ (c) Minimally contracted $\succ$

Figure 9: Preference contraction

Example 13 *Let a preference relation $\succ$ be a total order of $\{x_1, \ldots, x_5\}$ (Figure 9(a)). Let a base contractor CON be $\{x_1x_4, x_2x_5\}$. Let P^- be defined as in Lemma 3. That is $P^- = \{x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_2x_5\}$. Then $(\succ - P^-)$ is shown in Figure 9(b) as the set of solid edges. P^- is not minimal because $(P^- - \{x_1x_2\})$ (Figure 9(c)) is also a full contractor of $\succ$ by CON. In fact, $(P^- - \{x_1x_2\})$ is a* minimal *full contractor of $\succ$ by CON. As we can see, having the edge x_1x_2 in P^- is not necessary. First, it is not a CON-edge. Second, the edge x_2x_4 of the CON-detour $x_1 \succ x_2 \succ x_4$ is already in P^-.*

As we have shown in Example 13, a minimal full contractor can be constructed by including in it only the edges which start some CON-detour, if the detour is not already disconnected. Thus, before adding such an edge to a full contractor, we need to know if an edge in the detour but not starting it is already in the full contractor. So instead of contracting $\succ$ by CON at once, we split CON into *strata,* and contract $\succ$ incrementally by the strata of CON. Essentially, a stratum of CON consists of the edges whose detours can be disconnected in a single iteration without violating the minimality of the full contractor computed so far. The method of splitting a base contractor into strata we propose to use is as follows.

Definition 7 *The* stratum index *of $xy \in CON$ is* the maximum length of a $\succ$-path started by y and consisting of the end nodes of CON-edges. *A* stratum *is the set of all CON-edges with the same* stratum index.

This method of stratification has the following useful property. If a preference relation is minimally contracted by the strata with indices of up to n, then minimally contracting that relation by the stratum with the index $n+1$ guarantees the minimality of the entire contraction.

Clearly, if a preference relation is infinite, a tuple can start $\succ$-paths of arbitrarily large lengths. Therefore, the stratum index of a CON-edge may be undefined. We exclude such cases here, so we can assume that for each edge of CON relations, the stratum index is defined.

Definition 8 *Let CON be a base contractor of a preference relation $\succ$. Let $K_{CON} = \{y \mid \exists x \ . \ \ xy \in CON\}$, and $\succ_{CON} = \succ \cap K_{CON} \times K_{CON}$. Then CON is* stratifiable *iff for every $y \in K_{CON}$ there is an integer k such that all the paths started by y in $\succ_{CON}$ are of length at most k. CON is* finitely stratifiable *iff there is a constant k such that all paths in $\succ_{CON}$ are of length at most k.*

The intuition beyond the definition above is as follows. The K_{CON} defines the set of all the end nodes of CON-edges, i.e., all the nodes which will lose *incoming* edges after the contraction. The relation $\succ_{CON}$ is the restriction of $\succ$ to the set K_{CON}.

Definition 8 implies that for every edge of a stratifiable CON, the stratum index is defined. Since the shortest path in $\succ_{CON}$ is of length 0, the least stratum index for stratifiable relations is 0.

Example 14 *Take a preference relation* $\succ_2$

$$t \succ_2 t' \equiv t.price < t'.price$$

where the domain is the set of rational numbers Q *, and base contractors* CON_1 *and* CON_2

$$CON_1(t, t') = t.price < 10,000;$$
$$CON_2(t, t') = t.price < t'.price \wedge (t'.price = 5,000 \vee t'.price = 6,000).$$

Then K_{CON_1} *and* K_{CON_2} *are defined by* $F_{CON_{K_1}}(x) = \top$ *and* $F_{CON_{K_2}}(x) = x.price = 5,000 \vee x.price = 6,000$*, respectively. Hence,* $\succ_{CON_1}$ *and* $\succ_{CON_2}$ *are defined by*

$$F_{\succ_{CON_1}}(t, t') = t.price < t'.price;$$
$$F_{\succ_{CON_2}}(t, t') = t.price = 5,000 \wedge t'.price = 6,000.$$

Clearly, the length of paths in $\succ_{CON_1}$ *is unbounded. Hence,* CON_1 *is not finitely stratifiable. The relation* CON_2 *is finitely stratifiable – the longest path in* $\succ_{CON_2}$ *is of length* 1.

Above we illustrate the finite stratifiability property of a base contractor. The preference relation and the base contractors are represented as ERO formulas. It is an open question whether there are stratifiable relations, defined using ERO formulas, which are not finitely stratifiable.

4.2. Computation of minimal full contractor

Below we present an approach of constructing a minimal full contractor for a *stratifiable* relation CON.

Theorem 2 (Minimal full contractor construction). *Let* $\succ$ *be a preference relation, and* CON *be a stratifiable base contractor of* $\succ$*. Let* L_i *be the set of the end nodes of all* CON*-edges of stratum* i*. Then* P^-*, defined as follows, is a* minimal full contractor *of* $\succ$ *by* CON

$$P^- = \bigcup_{i \in 0}^{\infty} E_i,$$

where

$$E_i = \{xy \mid \exists v \in L_i \,.\, xv \in CON \wedge x \succ y \wedge y \succeq v \wedge yv \notin (P^-_{i-1} \cup CON)\}$$

$$P^-_{-1} = \emptyset,$$

$$P^-_i = \bigcup_{j=0}^{i} E_i$$

Intuitively, the set E_i contains all the CON edges of stratum i along with the edges of $\succ$ which need to be discarded to contract the preference relation by that stratum. P^-_i is the union of all such sets up to stratum i.

PROOF OF THEOREM 2. Every E_i contains the CON-edges of stratum i. Thus, P^- contains CON. Now we prove that $(\succ - P^-)$ is an SPO. Its irreflexivity follows from the irreflexivity of $\succ$. Transitivity is proved by induction on stratum index.

It is given that $\succ$ is transitive. Assume $(\succ - P^-_n)$ is transitive. Prove that $(\succ - P^-_{n+1}) = (\succ -(P^-_n \cup E_{n+1}))$ is transitive. For the sake of contradiction, assume

$$\exists x, y, z \;.\; xy \notin (\succ - P^-_{n+1}) \wedge xz, zy \in (\succ - P^-_{n+1}) \qquad (1)$$

which implies

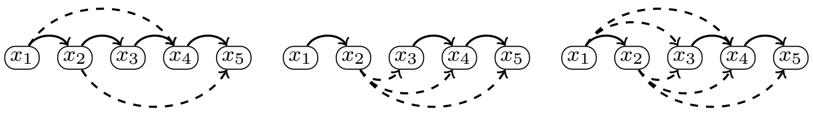
$$xz, zy \notin E_{n+1} \cup P^-_n \qquad (2)$$

Transitivity of $(\succ - P^-_n)$ and (1) imply $xy \in (\succ - P^-_n)$ and thus $xy \in E_{n+1}$. Hence,

$$\exists v \in L_n \;.\; xv \in CON \wedge x \succ y \wedge y \succeq v \wedge yv \notin (P^-_n \cup CON) \qquad (3)$$

According to (3), $y \succeq v$. If $y = v$, then (2), (1) and (3) imply $xz \in E_{n+1}$ which is a contradiction. If $y \succ v$, then $xz \notin E_{n+1}$ implies $zv \in P^-_n \cup CON$ by the construction of E_{n+1}. Note that $zv \in CON$ implies zv is a CON-edge of stratum index $n+1$ and thus either $zy \in E_{n+1}$ or $yv \in P^-_n \cup CON$, which contradicts (2) and (3). If $zv \in P^-_n$, then $zy, yv \notin P^-_n$ implies intransitivity of $(\succ - P^-_n)$, which contradicts the inductive assumption. Thus, $(\succ - P^-_{n+1})$ is transitive by induction. Assume that $(\succ - P^-)$ is not transitive. The violation of transitivity means that there is an edge $xy \in P^-$ such that there exists a path from x to y none of whose edges is in P^- (Lemma 1). Since xy must be in P^-_n for some n, that implies intransitivity of $(\succ - P^-_n)$, which is a contradiction. Thus P^- is a full contractor of $\succ$ by CON.

Now we prove that P^- is a *minimal* full contractor. If it is not, then by Theorem 1, there is $xy \in P^-$ for which there is no CON-detour which shares with P^- only the edge xy. Note that $xy \in P^-$ implies $xy \in E_n$ for some n. By definition of E_n, there is a CON-detour $x \succ y \succeq v$ which shares with P_n^- only xy. Since all CON-detours which xy belongs to have other P^--edges, $yv \in P^-$. Since $yv \notin P_n^-$, there must exist $k > n$ such that $yv \in E_k$. However, that is impossible by construction: every CON-detour which may be started by yv must have the stratum index not greater than n. $\square$



(a) $\succ$ and CON (b) P_0^- (c) P_1^-

Figure 10: Using Theorem 2 to compute a minimal full contractor

Example 15 *Let $\succ$ and CON be as in Example 13 (Figure 10(a), the transitive edges are omitted for clarity). We use Theorem 2 to construct a minimal full contractor of $\succ$ by CON. The relation CON has two strata with the end nodes $L_0 = \{x_5\}$, $L_1 = \{x_4\}$. Then $E_0 = \{x_2x_3, x_2x_4, x_2x_5\}$, $P_0^- = E_0$, $E_1 = \{x_1x_3, x_1x_4\}$, $P_1^- = E_0 \cup E_1$, and a minimal full contractor of $\succ$ by CON is $P^- = P_1^-$.*

4.3. Prefix and suffix full contractors

It is easy to observe that the minimal full contractor P^- constructed in Theorem 2 has the property that its every edge xy *starts* at least one CON-detour in which xy is the only P^--edge. Full contractors which have this property are called *prefix.* Prefix full contractors are minimal by Theorem 1. It turns out that for a given preference relation and a given base contractor, a prefix full contractor is unique.

Proposition 1 *Given a preference relation $\succ$ and a stratifiable base contractor CON, there exists a unique prefix full contractor P^- of $\succ$ by CON.*

Proof. The existence of a prefix full contractor follows from Theorem 2. The fact that every prefix full contractor is equal to P^- constructed by Theorem 2 can be proved by induction on the stratum index of CON. Namely, we show that for every n, P_n^- is contained in every prefix full contractor of $\succ$

by CON. Clearly, the set E_0 contracting $\succ$ by the 0th stratum of CON has to be in any prefix full contractor. Assume every edge in P_n^- is in any prefix full contractor of $\succ$ by CON. If an edge $xy \in E_{n+1} - CON$, then there is a CON-detour $x \succ y \succ v$ in which xy is the only P^--edge (i.e., $yv \notin P^-$). Hence if xy is not in some prefix full contractor P', then yv has to be in P' by Lemma 1. However, $P_n^- \subsetneq P'$ is enough to disconnect every CON-detour with index up to n, and yv can only start a CON-detour with the stratum index up to n. Hence P' is not a minimal full contractor and P^- is a unique prefix full contractor. $\square$

A natural question which arises after the discussion of prefix full contractors is whether the *suffix* full contractor can be constructed similarly to the prefix full contractor above. Analogously, a full contractor P^- of $\succ$ by CON is *suffix* if every edge xy of P^- *ends* at least one CON-detour of $\succ$ in which xy is the only P^--edge. Note that as in the case of the prefix full contractor, the suffix full contractor is minimal by Theorem 1. It turns out that the connection between prefix and suffix full contractors is straightforward. To define it, we use the notion of the inverse of a binary relation.

Definition 9 *Given a binary relation $R \subseteq \mathcal{U} \times \mathcal{U}$, the* inverse *$R_{inv}$ of R is*

$$R_{inv} = \{xy \mid yx \in R\}$$

Take a preference relation $\succ$ and its base contractor CON. It is clear that $\succ_{inv}$ is a preference relation (i.e., an SPO), and CON_{inv} is a base contractor of $\succ_{inv}$ (i.e., $CON_{inv} \subseteq \succ_{inv}$).

Proposition 2 *Take a preference relation $\succ$ and its base contractor CON. Let P^- be the* prefix *full contractor of $\succ_{inv}$ by CON_{inv}. Then ${P^-}_{inv}$ is the* suffix *full contractor of $\succ$ by CON.*

Proof. The fact that ${P^-}_{inv}$ is a full contractor of $\succ$ by CON follows from the definitions of SPO and the inverse of a binary relation. To prove that ${P^-}_{inv}$ is the *suffix* full contractor of $\succ$ by CON, recall that the prefix property of P^- implies that every edge xy of P^- *starts* some CON_{inv}-detour of $\succ_{inv}$ in which xy is the only P^--edge. In $\succ$, the edge $yx \in P^-$ *ends* the *inverse* of that detour and is the only ${P^-}_{inv}$-edge in it. $\square$

Note that Proposition 2 and the fact that the inverse of the inverse of a relation R is R itself imply that the suffix full contractor of $\succ$ by CON is unique. To compute the suffix full contractor of $\succ$ by CON, one may compute $\succ_{inv}$ and CON_{inv}, use Theorem 2 to compute the prefix full contractor P^- of $\succ_{inv}$ by CON_{inv}, and compute the inverse of P^-. At the same time, it is important to remember that one of the preconditions of that theorem is that CON must be stratifiable w.r.t. $\succ$. By definition of inverse, CON being stratifiable w.r.t. $\succ$ does not imply that CON_{inv} is stratifiable w.r.t. $\succ_{inv}$ and vice versa.

5. Contraction by finitely stratifiable base contractors

In this section, we consider the practical issues of computing minimal full contractors. In particular, we show how the method of constructing a *prefix* full contractor we have proposed in Theorem 2 can be adapted to various classes of preference and base contractor relations.

Due to the connection between prefix and suffix full contractors discussed in the previous section, the same methods can be used to compute *suffix* full contractors (with the overhead of computing inverses). From now on, we focus on prefix full contractors only.

Observe that the definition of the minimal full contractor in Theorem 2 is recursive. Namely, to find the edges we need to discard for contracting the preference relation by the stratum $n + 1$ of CON, we need to know which edges to discard for contracting it by all the previous strata. It means that for base contractor relations which are not finitely stratifiable (i.e., CON has infinite number of strata), the corresponding computation will never terminate.

Assume that CON is a finitely stratifiable relation. First we note that any base contractor of a finite preference relation is finitely stratifiable: all paths in such preference relations are not longer than the size of the relation, and base contractors are required to be subsets of the preference relations. At the same time, if CON is a base contractor of an infinite preference relation, then it can be finitely stratifiable without being finite. In particular, it may be the case that the *length* of all paths in $\succ_{CON}$ is bounded, but the *number* of paths is infinite.

Below we consider the cases of finite and finitely representable, finitely stratifiable base contractors separately.

5.1. Computing the prefix full contractor: finitely representable relations

Here we assume that the relations CON and $\succ$ are represented by finite ERO-formulas F_{CON} and $F_{\succ}$. We aim to construct a finite ERO-formula F_{P^-} which represents the prefix full contractor of $\succ$ by CON. The function `minContr(`$F_{\succ}$`, `F_{CON}`)` shown below exploits the method of constructing prefix full contractors from Theorem 2, adapted to formula representations of relations. All the intermediate variables used in the algorithm store formulas. Hence, for example, any expression in the form $''F(x,y) := \ldots''$ means that the formula-variable F is assigned the formula written in the right-hand side, which has two free tuple variables x and y. The operator QE used in the algorithm computes a quantifier-free formula equivalent to its argument formula. For ERO-formulas, the operator QE runs in time polynomial in the size of its argument formula (if the number of attributes in $\mathcal{A}$ is fixed), and exponential in the number of attributes in $\mathcal{A}$.

The function `minContr` (Algorithm 1) starts with an empty full contractor P^-_{-1} (line 2), then the formulas $F_{K_{CON}}$ and $F_{\succ_{CON}}$ representing K_{CON} and $\succ_{CON}$ are computed (lines 3 and 4) by definition of those sets. To get the 0-th stratum of CON, the function `getStratum` is used (line 5). After that, in every iteration of the `while`-loop, we compute the formula defining E_i (line 7) as in Theorem 2, compute the formula representing the intermediate full contractor $F_{P^-_i}$ (line 8), and compute the formula F_{L_i} representing the next stratum. The `while`-loop terminates when the next stratum contains no edges (i.e., F_{L_i} is not defined). Finally, the full contractor of $\succ$ by CON is returned.

To compute formulas representing different strata of CON, `getStratum` (Algorithm 2) is used. It takes three parameters: the formula $F_{\succ_{CON}}$ representing the relation $\succ_{CON}$, the formula $F_{K_{CON}}$ representing the set of the end nodes of CON-edges, and the stratum index i. It returns a formula which represents the set of the end nodes of CON-edges of stratum i , or `undefined` if the corresponding set is empty. That formula is computed according to the definition of a stratum.

Algorithm 1 `minContr`$(F_\succ, F_{CON})$

```
1:  $i = 0$
2:  $F_{P^-_{-1}}(x,y) := false$
3:  $F_{K_{CON}}(y) := QE(\exists x \,.\, F_{CON}(x,y))$
4:  $F_{\succ_{CON}}(x,y) := F_{CON}(x,y) \wedge F_{K_{CON}}(x) \wedge F_{K_{CON}}(y)$
5:  $F_{L_i}(y) :=$ getStratum$(F_{\succ_{CON}}, F_{K_{CON}}, i)$
6:  while $F_{L_i}$ is defined do
7:      $F_{E_i}(x,y) := QE(\exists v \,.\, F_{L_i}(v) \wedge F_{CON}(x,v) \wedge F_\succ(x,y) \wedge$
                  $(y = v \vee F_\succ(y,v) \wedge \neg(F_{P^-_{i-1}}(y,v) \vee F_{CON}(y,v))))$
8:      $F_{P^-_i}(x,y) := F_{P^-_{i-1}}(x,y) \vee F_{E_i}(x,y)$
9:      $i := i + 1;$
10:     $F_{L_i}(y) :=$ getStratum$(F_{\succ_{CON}}, F_{K_{CON}}, i)$
11: end while
12: return $F_{P^-_i}$
```

Algorithm 2 `getStratum`$(F_{\succ_{CON}}, F_{K_{CON}}, i)$

```
Require: $i \geq 0$
1:  if i = 0 then
2:      $F_{L_i}(y) := QE(\ F_{K_{CON}}(y) \wedge \neg\exists x_1(F_{\succ_{CON}}(y,x_1)))$
3:  else
4:      $F_{L_i}(y) := QE(\ \exists x_1, \ldots, x_i \,.\, F_{\succ_{CON}}(y,x_1) \wedge F_{\succ_{CON}}(x_1,x_2) \wedge \ldots$
                  $\wedge F_{\succ_{CON}}(x_{i-1},x_i)) \wedge \neg\exists x_1, \ldots, x_{i+1} \,.\, F_{\succ_{CON}}(y,x_1)$
                  $\wedge F_{\succ_{CON}}(x_1,x_2) \wedge \ldots \wedge F_{\succ_{CON}}(x_i,x_{i+1})))$
5:  end if
6:  if $QE(\exists y \,.\, F_{L_i}(y))$ then
7:      return $F_{L_i}$
8:  else
9:      return undefined
10: end if
```

Proposition 3 *Let CON be a finitely stratifiable base contractor of a preference relation $\succ$. Then Algorithm 1 terminates and computes the prefix full contractor of $\succ$ by CON.*

Proposition 3 holds because Algorithm 1 uses the construction from Theorem 2. Below we show an example of computing the prefix full contractor for a finitely representable preference relation.

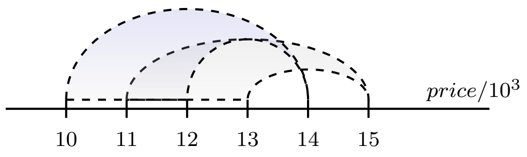


Figure 11: Example 16

Example 16 *Let a preference relation $\succ$ be defined by the following formula*

$$F_{\succ}(o,o') \equiv o.price < o'.price$$

and a base contractor CON (Figure 11) be defined by

$$\begin{aligned} F_{CON}(o,o') \equiv {} & (11000 \leq o.price \leq 13000 \wedge o'.price = 15000) \vee \\ & (10000 \leq o.price \leq 12000 \wedge o'.price = 14000) \end{aligned}$$

where price is a Q -attribute. Then $F_{K_{CON}}(o) \equiv o.price = 14000 \vee o.price = 15000$ and $F_{\succ_{CON}}(o,o') \equiv F_{\succ}(o,o') \wedge F_{K_{CON}}(o) \wedge F_{K_{CON}}(o')$ (Figure 11). The end nodes of the CON strata are defined by the following formulas:

$$\begin{aligned} F_{L_0}(o) &\equiv o.price = 15000 \\ F_{L_1}(o) &\equiv o.price = 14000 \end{aligned}$$

The relations contracting all CON strata are defined by the following formulas

$$\begin{aligned} F_{E_0}(o,o') &\equiv 11000 \leq o.price \leq 13000 \wedge 13000 < o'.price \leq 15000 \\ F_{E_1}(o,o') &\equiv 10000 \leq o.price \leq 11000 \wedge 13000 < o'.price \leq 14000 \end{aligned}$$

Finally, a full contractor P^- of $\succ$ by CON is defined by $F_{P^-}(o,o') \equiv F_{E_1}(o,o') \vee F_{E_2}(o,o')$.

The finite stratifiability property of CON is crucial for the termination of the algorithm: the algorithm does not terminate for relations which are not finitely stratifiable. Hence, given a base contractor relation, it is useful to know if it is finitely stratifiable or not. Let us consider the formula $F_{\succ_{CON}}$. Without loss of generality, we assume it is represented in DNF. By definition, CON is a finitely stratifiable relation iff there is a constant k such that all $\succ_{CON}$ paths are of length at most k. In the next theorem, we show that this property can be checked by a single evaluation of the quantifier elimination operator.

Theorem 3 (Checking finite stratifiability property). *Let F_R be an ERO-formula, representing an SPO relation R, in the following form*

$$F_R(o, o') = F_{R_1}(o, o') \vee \ldots \vee F_{R_l}(o, o'),$$

where F_{R_i} is a conjunction of atomic formulas. Then checking if there is a constant k such that the length of all R-paths is at most k can be done by a single evaluation of QE over a formula of size linear in $|F_R|$.

In Theorem 3, we assume that each atomic formula using the operators $\leq, \geq$ is transformed to a disjunction of two formulas, one which uses the strict comparison operator and the other using the equality operator. The proof of Theorem 3 and the details of the corresponding finite stratifiability property test are provided in Appendix B.

5.2. Computing prefix full contractor: finite relations

In this section, we consider finite relations $\succ$ and CON. We assume that the relations are stored in separate tables: the preference relation table R and the base contractor table C, each having two columns X and Y. Every tuple in a table corresponds to an element of the corresponding binary relation. Hence, R has to be an SPO and $C \subseteq R$. Here we present an algorithm, Algorithm 3, for computing the prefix full contractor of a preference relation $\succ$ by CON represented by such tables. Essentially, the algorithm is an adaptation of Theorem 2.

The function `minContrFinite` takes two arguments: R and C. The function is implemented in terms of relational algebra operators. First, it constructs the table K with one column Y storing the end nodes of all C-edges. K corresponds to K_{CON} (Definition 8) and is computed analogously. Second, it computes the table RC storing a restriction of the original preference relation R to K (as $\succ_{CON}$ in Definition 8). These two tables are needed for obtaining the strata of C. After that, the function picks all strata of C one by one and contracts the original preference relation by each stratum in turn, as shown in Theorem 2. In every iteration of the `while`-loop, it computes the end nodes E of the current stratum (line 8), removes them from K (line 10) and removes the C-edges of the current stratum from RC, to prepare them for the next iteration. In line 13, we compute the table P with two columns X and Y, which represents the minimal full contractor computed so far. The `while`-loop terminates when the table K is exhausted (i.e., all strata have been processed).

The extraction of the strata of CON in the order of the stratum index is performed as follows. It is clear that the nodes ending CON-edges of stratum 0 do not start any edge in RC. The set E computed in line 8 is the difference of the set K of nodes ending C-edges and the set of nodes starting some edges in RC. Hence, the initial value of E contains all the nodes ending C-edges of stratum 0. To get the end nodes of the next stratum of C, we need to remove all the edges from RC which end in members of E, and remove E from K. After the stratum with the highest index is obtained, the relation K becomes empty.

Algorithm 3 uses two renamings of K (K_1 and K_2) and two renamings of R (R_1 and R_1).

Algorithm 3 `minContrFinite`(R, C)

```
Require: $R$ is transitive, $C \subseteq R$
 1: $P \leftarrow C$
 2: /* Get the end nodes of all $C$-edges */
 3: $K \leftarrow \pi_Y(C)$
 4: /*  $RC$ is related to $R$ as $\succ_{CON}$ to $\succ$ in Definition 8 */
 5: $RC \leftarrow \pi_{R.X,\ R.Y}\ (K_1 \underset{K_1.Y=R.X}{\bowtie} R \underset{K_2.Y=R.Y}{\bowtie} K_2)$
 6: while $K$ not empty do
 7:    /* Get the end nodes of the next stratum $C$-edges */
 8:    $E \leftarrow K - \pi_X(RC)$
 9:    /* Prepare $K$ and $RC$ for the next iteration */
10:    $K \leftarrow K - E$
11:    $RC \leftarrow RC - RC \underset{RC.Y=E.Y}{\bowtie} E$
12:    /* Add to $P$ the $R$-edges contracting the current stratum of $C$*/
13:    $P \leftarrow P \cup \pi_{R_1.X,\ R_1.Y}\ (R_1 \underset{R_1.Y\ =\ R_2.X}{\bowtie} (R_2 - P) \underset{R_1.X\ =\ C.X,\ R_2.Y\ =\ C.Y}{\bowtie}$
       $(C \underset{C.Y\ =\ E.Y}{\bowtie} E))$
14: end while
15: return $P$
```

Proposition 4 *Algorithm 3 computes the prefix full contractor of R by C. It can be implemented in $\mathcal{O}(|C|^2 \cdot |R| \cdot log|R|)$ time.*

The correctness of Proposition 4 holds because Algorithm 3 follows from Theorem 2. To compute the running time, we assume that the cost of binary operation (join, union, difference) of two relations T and S to be $O(|T+S|)$ if both arguments are sorted on the same key, and $O(|T| \cdot log|S|)$ otherwise. That cost can be clearly reduced with an appropriate use of indexing and hashing. All the input arguments and the intermediate relations used in Algorithm 3 are kept sorted. The relation P, containing the intermediate full contractor edges, and the relation R are stored as a single relation, in which edges belonging to P are marked.

In line 1, the relation C is sorted on (X, Y) and copied to P. In line 3, the projection of C is computed, the result is sorted on Y and copied to K. The relation RC computed in line 5 is sorted on (X, Y). The processing of lines 1-5 takes time $O(|R \cdot log|R|)$ (given that $|C| \leq |R|$). The running time of the body of the `while`-loop is clearly dominated by the running time of line 13, which is $O(|C| \cdot |R| \cdot log|R|)$. Finally, since the size of K is $O(|C|)$, the running time of the algorithm is as stated in Proposition 4.

6. Preference-protecting contraction

Here we propose an operator of *preference-protecting* contraction. In addition to a base contractor CON, a subset P^+ of the original preference relation to be *protected from removal* in the contracted preference relation may also be specified. Such a relation is complementary with respect to the base contractor: the relation CON defines the preferences to discard, whereas the relation P^+ defines the preferences to protect.

Definition 10 *Let $\succ$ be a preference relation and CON be a base contractor of $\succ$. Let a relation P^+ be such that $P^+ \subseteq \succ$. A full contractor P^- of $\succ$ by CON such that $P^+ \cap P^- = \emptyset$ is called a P^+-*protecting full contractor of $\succ$ by *CON. A minimal full contractor P^- of $\succ$ by CON such that $P^+ \cap P^- = \emptyset$ is called a P^+-*protecting minimal full contractor of $\succ$ by *CON.*

Given any full contractor P^- of $\succ$ by CON, by Lemma 1, P^- must contain at least one edge from every CON-detour. Thus, if P^+ contains an entire CON-detour, protecting P^+ while contracting $\succ$ by CON is not possible.

Theorem 4 *Let CON be a stratifiable base contractor relation of a preference relation $\succ$ such that $P^+ \subset \succ$. There exists a minimal full contractor of $\succ$ by CON that protects P^+ iff $P^+_{TC} \cap CON = \emptyset$, where P^+_{TC} is the transitive closure of P^+.*

As we noted, the necessary condition of the theorem above follows from Lemma 1. The sufficient condition follows from Theorem 5 that we prove further.

A naive way of computing a preference-protecting minimal full contractor is by finding a minimal full contractor P^- of $(\succ - P^+)$ and then adding P^+ to P^-. However, $(\succ - P^+)$ is not an SPO in general, thus obtaining SPO of $\succ - (P^- \cup P^+)$ becomes problematic.

The solution we propose here uses the following idea. First, we find a base contractor CON' such that minimal contraction of $\succ$ by CON' is equivalent to minimal contraction of $\succ$ by CON with protected P^+. After that, we compute a minimal full contractor of $\succ$ by CON' using Theorem 2.

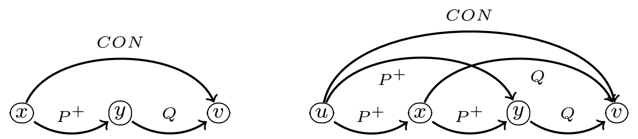
Recall the full contractor P^- constructed using Theorem 2. The prefix property of P^- implies that if edges of P^+ do not start CON-detours in $\succ$, then $P^- \cap P^+ = \emptyset$. Otherwise, assume that an edge $xy \in P^+$ starts a CON-detour in $\succ$. By Lemma 1, any P^+-protecting full contractor P^- has to contain an edge (*different* from xy) which belongs to a CON-detour started by xy. For a CON-detour of length two started by xy, P^- has to contain the *edge ending the CON-detour*. The set of all such edges can be defined as follows:

$$Q = \{xy \mid \exists u : u \succ x \succ y \wedge uy \in CON \wedge ux \in P^+\}.$$

It turns out that the set Q has a very useful property: it is not only contained in any P^+-protecting full contractor, but it can also be used to construct a P^+-protecting minimal full contractor as shown in the next theorem.

Theorem 5 *Let $\succ$ be a preference relation, and CON be a stratifiable base contractor of $\succ$. Let also P^+ be a transitive relation such that $P^+ \subseteq \succ$ and $P^+ \cap CON = \emptyset$. Then the prefix full contractor of $\succ$ by $CON \cup Q$ is a P^+-protecting minimal full contractor of $\succ$ by CON.*

Proof. Let P^- be the prefix full contractor of $\succ$ by $CON' = CON \cup Q$. We prove that $P^- \cap P^+ = \emptyset$, i.e., P^- protects P^+. For the sake of contradiction, assume there is $xy \in P^+ \cap P^-$. We show that this contradicts the prefix property of P^-. Since P^- is the prefix full contractor, there is a CON'-detour from x to some v in $\succ$, started by xy and having only the edge xy in P^-. We have two choices: either it is a CON-detour or a Q-detour. Consider the first case. Clearly, $y \neq v$, otherwise $P^+ \cap CON \neq \emptyset$. Thus, $xv \in CON$



(a) P^+-protection: case 1 (b) P^+-protection: case 2

Figure 12: Proof of Theorem 5

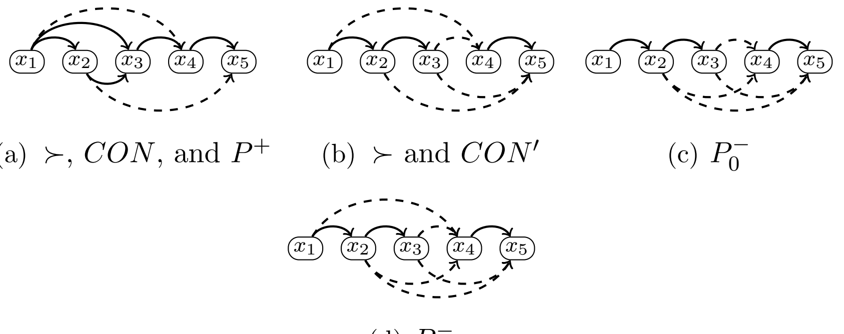
and $x \succ y \succ v$ (Figure 12(a)). $yv \in Q$ follows from $xy \in P^+$, $xv \in CON$ and the construction of Q. Note that every path from y to v in $\succ$ contains a P^--edge because P^- is a full contractor of $\succ$ by $CON \cup Q$. That implies that no CON-detour from x to v started by xy has only xy in P^-, which contradicts the initial assumption.

Consider the second case, i.e., there is a Q-detour from x to some v started by xy and having only the edge xy in P^-. Since $xv \in Q$, there is $uv \in CON$ such that $ux \in P^+$ (Figure 12(b)). $ux, xy \in P^+$ imply $uy \in P^+$ by transitivity of P^+. $uy \in P^+$ and $uv \in CON$ imply $yv \in Q$. That along with the fact that P^- is a full contractor of $\succ$ by $CON \cup Q$ implies that every path in $\succ$ from y to v contains a P^--edge. Hence, there is no Q-detour from x to v started by xy and having only xy in P^-. That contradicts the initial assumption about xy.

Now we prove that P^- is a minimal full contractor of $\succ$ by CON. The fact that it is a full contractor of $\succ$ by CON follows from the fact that it is a full contractor of $\succ$ by a superset CON' of CON. We prove now its minimality. Since P^- is the prefix full contractor of $\succ$ by CON', for every $xy \in P^-$, there is $xv \in CON'$ such that there is a corresponding detour T in which xy is the only P^--edge. If it is a CON-detour, then xy satisfies the minimality condition from Theorem 1. If it is a Q-detour, then there is a CON-edge uv such that $ux \in P^+$. We showed above that P^- protects P^+. Hence, the CON-detour obtained by joining the edge ux and T has only xy in P^-. Therefore, P^- is a minimal full contractor of $\succ$ by CON. $\square$

Note that the sets of the end nodes of $(CON \cup Q)$-edges and the end nodes of CON-edges coincide by the construction of Q. Therefore, $(CON \cup Q)$ *is stratifiable or finitely stratifiable iff* CON *is stratifiable or finitely stratifiable,* correspondingly. Hence, if CON is a finitely stratifiable relation, Algorithms 1 and 3 can be used to compute a preference-protecting minimal full contractor of $\succ$ by CON. If the relations $\succ$ and CON are finite, then Q can be constructed in polynomial time in the size of $\succ$ and CON by a relational algebra expression constructed from its definition. If the relations are finitely representable, then Q may be computed using the quantifier elimination operator QE.

For Theorem 5 to apply, the relation P^+ has to be transitive. Non-transitivity of P^+ implies that there are two edges $xy, yz \in P^+$ which should be protected while transitive edge xz is not critical. However, a relation obtained as a result of preference-protecting contraction is a preference relation (i.e., SPO). Hence, the edge xz will also be protected in the resulting preference relation. This fact implies that *protecting any relation is equivalent to protecting its minimal transitive extension: its transitive closure.* Therefore, if P^+ is not transitive, one needs to compute its transitive closure to use Theorem 5. For finite relations, transitive closure can be computed in polynomial time [13]. For finitely representable relations, *Constraint Datalog* [21] can be used to compute transitive closure.



(a) $\succ$, CON, and P^+ (b) $\succ$ and CON' (c) P_0^-

(d) P_1^-

Figure 13: Using Theorem 5 to compute a preference-protecting minimal full contractor

Another important observation here is that the P^+-protecting minimal full contractor of $\succ$ by CON computed according to Theorem 5 is not necessarily a *prefix* full contractor of $\succ$ by CON. This fact is illustrated in the following example.

Example 17 *Let a preference relation $\succ$ be a total order of $\{x_1, \ldots, x_5\}$ (Figure 13(a), the transitive edges are omitted for clarity). Let a base contractor CON be $\{x_1x_4, x_2x_5\}$, and $P^+ = \{x_1x_3, x_2x_3, x_4x_5\}$.*

The existence of a minimal P^+-protecting full contractor of $\succ$ by CON follows from Theorem 4. We use Theorem 5 to construct it. The set Q is equal to $\{x_3x_4, x_3x_5\}$ and $CON' = \{x_1x_4, x_2x_5, x_3x_4, x_3x_5\}$. We construct the *prefix full contractor of $\succ$ by CON'. The relation CON' has two strata whose end nodes are $L_0 = \{x_5\}$, $L_1 = \{x_4\}$. Then $E_0 = \{x_2x_4, x_3x_4, x_2x_5, x_3x_5\}$, $P_0^- = E_0$, $E_1 = \{x_1x_4\}$, $P_1^- = E_0 \cup E_1$, and $P^- = P_1^-$. By Theorem 5, P^- is a P^+-protecting minimal full contractor of $\succ$ by CON. However, P^- is not a prefix full contractor of $\succ$ by CON, because the edges x_3x_4, x_3x_5 do not start any CON-detour.*

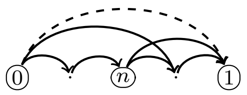


Figure 14: $\succ$ (transitive edges omitted), CON, and P_n^+

In conclusion of this section, we formally show that the number of minimal full contractors by stratifiable base contractor can be infinite. We have mentioned that fact several times above, but (even though it seems reasonable) we have not proved it yet. To do that, we use Theorem 5. Take a preference relation $t \succ t' \equiv t.p < t'.p$ (for a Q -domain attribute p) and a base contractor $CON(t, t') = t.p = 0 \wedge t'.p = 1$. Take a subset P_b^+ of $\succ$, defined as

$$P_b^+(t, t') \equiv t.p = 0 \wedge t'.p > 0 \wedge t'.p < 1 \wedge t'.p \neq n \vee t.p = n \wedge t'.p = 1,$$

for some b such that $0 < b < 1$ (Figure 14). It is easy to check that P_b^+ is transitive and does not intersect $\succ$. Hence by Theorem 5, there is a P_b^+-protecting minimal full contractor of $\succ$ by CON. Denote it as P_b^- and the set of P_b^- for all b as $\mathcal{P}$. Since the number of rational numbers b between 0 and 1 is infinite, the set $\mathcal{P}$ is of infinite size. By Lemma 1, $P_b^- \in \mathcal{P}$ contains the edge $(0, b)$ that is not in $P_a^- \in P$ for $a \neq b$, because $(0, b) \in P_a^+$. Hence, $\mathcal{P}$ contains an infinite number of *different* minimal full contractors.

7. Meet preference contraction

In this section, we consider the operation of *meet preference contraction.* In contrast to the preceding sections, where the main focus was the minimality of preference relation change, the contraction operation considered here changes a preference relation not necessarily in a minimal way. A full meet contractor of a preference relation is the *union* of all minimal full contractors. When a certain set of preferences is required to be protected while

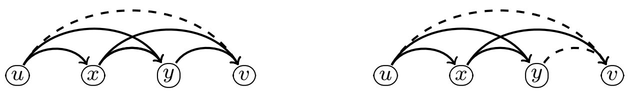

(a) $\succ$ and CON_1 (b) $\succ$ and CON_2

Figure 15: Example 18

contracting a preference relation, the operation of *preference-protecting meet contraction* may be used.

Definition 11 *Let $\succ$ be a preference relation, CON a base contractor of $\succ$, and $P^+ \subseteq \succ$. The relation P^{meet} is a* full meet contractor *of $\succ$ by CON iff*

$$P^{meet} = \bigcup_{P^- \in \mathcal{P}^{meet}} P^-,$$

for the set $\mathcal{P}^{meet}$ of all minimal full contractors of $\succ$ by CON. The relation $P^{meet}_{P^+}$ is a full P^+-protecting meet contractor of $\succ$ by CON *iff*

$$P^{meet}_{P^+} = \bigcup_{P^- \in \mathcal{P}^{meet}_{P^+}} P^-,$$

for the set $\mathcal{P}^{meet}_{P^+}$ of all P^+-protecting minimal full contractors of $\succ$ of CON.

Note that the relations $(\succ - P^{meet})$ and $(\succ - P^{meet}_{P^+})$ can be represented as intersections of preference (i.e., SPO) relations and thus are also preference (i.e., SPO) relations. Let us first consider the problem of constructing full meet contractors.

By the definition above, an edge xy is in the full meet contractor of a preference relation $\succ$ by CON if there is a minimal full contractor of $\succ$ by CON which contains xy. Theorem 1 implies that if there is no CON-detour in $\succ$ containing xy, then xy is not in the corresponding full meet contractor. However, the fact that xy belongs to a CON-detour is not a sufficient condition for xy to be in the corresponding full meet contractor.

Example 18 *Let a preference relation $\succ$ be a total order of $\{u, x, y, v\}$. Let also $CON_1 = \{uv\}$ (Figure 15(a)) and $CON_2 = \{uv, yv\}$ (Figure 15(b)). There is only one CON_1- and CON_2-detour containing xy: $u \succ x \succ y \succ v$. There is also a minimal full contractor of $\succ$ by CON_1 which contains xy: $P^-_1 = \{uy, xv, xy, uv\}$. However, there is no minimal full contractor of $\succ$ by*

CON_2 which contains xy because the edge yv of the CON_2-detour $u \succ x \succ y \succ v$ is in CON_2.

In Theorem 6, we show how full meet contractors can be constructed in the case of *finitely stratifiable base contractors* . According to that theorem, a $\succ$-edge xy is in the full meet contractor of $\succ$ by CON iff there is a full contractor P^- of $\succ$ by CON such that xy is the only P^--edge in some CON-detour. We use Theorem 4 to show that there is a minimal full contractor of $\succ$ by CON which contains xy while the other edges of the detour are protected.

Theorem 6 *Let CON be a finitely stratifiable base contractor of a preference relation $\succ$. Then the full meet contractor P^{meet} of $\succ$ by CON is*

$$R^{meet} = \{xy \mid \exists uv \in CON \;.\; u \succeq x \succ y \succeq v \wedge (ux \in (\succ - CON) \vee u = x) \wedge \\ (yv \in (\succ - CON) \vee y = v)\}$$

Proof. By Corollary 1, an edge xy is in a minimal full contractor P^- of $\succ$ by CON, iff there is a CON-detour of at most three edges in $\succ$ in which xy is the only P^--edge. Hence any minimal full contractor is a subset of R^{meet}. Now take every edge xy of R^{meet} and show there is a minimal full contractor of $\succ$ by CON which contains xy. Let $u \succeq x \succ y \succeq v$ for $uv \in CON$. Let us construct a set P' as follows:

$$P' = \begin{cases} \{ux, yv\} & \text{if } u \succ x \wedge y \succ v \\ \{ux\} & \text{if } u \succ x \wedge y = v \\ \{yv\} & \text{if } u = x \wedge y \succ v \\ \emptyset & \text{if } u = x \wedge y = v \end{cases}$$

P' is transitive, $P' \cap CON = \emptyset$, and $P' \subseteq \succ$. Theorem 4 implies that there is a P'-protecting minimal full contractor P^- of $\succ$ by CON. Since P^- protects P', there is a CON-detour in $\succ$ from u to v in which xy is the only P^--edge. This implies that $xy \in P^-$. $\square$

Now consider the case of P^+-protecting full meet contractors. A naive solution is to construct it as the difference of P^{meet} defined above and P^+. However, in the next example we show that such solution does not work in general.

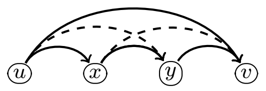


Figure 16: $\succ$, CON, and P^+ from Example 19

Example 19 *Let a preference relation $\succ$ be a total order of $\{u, x, y, v\}$ (Figure 16). Let also $CON = \{uy, xv\}$ and $P^+ = \{ux\}$. Note that $yv \notin P^+$, and by Theorem 6, $yv \in P^{meet}$. Hence, $yv \in (P^{meet} - P^+)$. However, note that $ux \in P^+$ implies that xy must be a member of every P^+-protecting full contractor in order to disconnect the path from u to y. Hence, there is no CON-detour in which yv is the only edge of the full contractor, and yv is not a member of any P^+-protecting minimal full contractor.*

The next theorem shows how a P^+-protecting full contractor may be constructed. The idea is similar to Theorem 6. However, to construct a full meet contractor, we used the set CON as a common part of all minimal full contractors. In the case of P^+-protecting full meet contractor, a superset C_{P^+} of CON is contained in all of them. Such a set C_{P^+} may be viewed as a union of CON and the set of all edges of $\succ$ that *must be discarded due to the protection of P^+.*

Theorem 7 *Let CON be a finitely stratifiable base contractor of a preference relation $\succ$, and P^+ a transitive relation such that $P^+ \subseteq \succ$ and $P^+ \cap CON = \emptyset$. Then the P^+-protecting full meet contractor $P^{meet}_{P^+}$ of $\succ$ by CON is*

$$R^{meet}_{P^+} = \{xy \mid xy \notin P^+ \wedge \exists uv \in CON \;.\; u \succeq x \succ y \succeq v \wedge \\ (ux \in (\succ - C_{P^+}) \vee u = x) \wedge (yv \in (\succ - C_{P^+}) \vee y = v)\},$$

where

$$C_{P^+} = \{xy \mid \exists uv \in CON \;.\; u \succeq x \succ y \succeq v \wedge (ux \in P^+ \vee u = x) \wedge \\ (yv \in P^+ \vee y = v)\}$$

Proof. First, it is easy to observe that C_{P+} is a subset of any P^+-protecting full contractor of $\succ$ by CON. It is constructed from the edges xy which participate in CON-detours of length at most three where all the other edges have to be protected. Since every CON-detour has to have at least one edge in a full contractor, xy has to be a member of every full contractor.

Second, we show that every P^+-protecting minimal full contractor P^- of $\succ$ by CON is a subset of $R^{meet}_{P^+}$. If some $xy \in P^-$, then by Corollary 1 there is an edge $uv \in CON$ such that $u \succeq x \succ y \succeq v$ and $ux, yv \notin P^-$. We show that $xy \in R^{meet}_{P^+}$. That holds if $xy \notin P^+$ (which holds for P^- by definition) and

$$(ux \in (\succ - C_{P^+}) \vee u = x) \wedge (yv \in (\succ - C_{P^+}) \vee y = v)$$

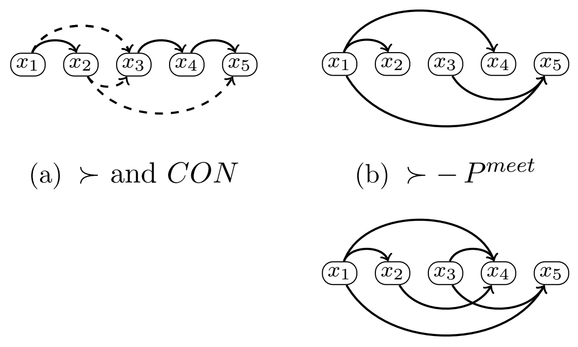
If both $u = x$ and $y = v$ hold then the expression above holds. Assume $u \succ x$ (the case $y \succ v$ is similar). If $ux \in C_{P^+}$ then, as we showed above, $ux \in P^-$ which is a contradiction. Hence, $ux \in (\succ - C_{P^+})$ and $xy \in R^{meet}_{P^+}$. Finally, $P^- \subseteq R^{meet}_{P^+}$.

Third, we show that every $xy \in R^{meet}_{P^+}$ is contained in some P^+-protecting minimal full contractor of $\succ$ by CON. The proof is similar to the proof of Theorem 6. By definition of $R^{meet}_{P^+}$, xy is such that $u \succeq x \succ y \succeq v$. Construct the set P' for xy as in the proof of Theorem 6. We show that for the set P'' which is the transitive closure of $(P^+ \cup P')$ we have $P'' \cap CON = \emptyset$. For the sake of contradiction, assume $P'' \cap CON \neq \emptyset$. This implies that there is a CON-detour consisting of P^+ and P' edges. Having only P^+-edges in the detour contradicts the initial assumption that $P^+ \cap CON = \emptyset$. Having a single edge of P' in the detour implies that the edge (either ux or yv) is in C_{P^+}, which contradicts the definition of $R^{meet}_{P^+}$. Having both ux and yv in the detour implies that $xy \in P^+$ which also contradicts the definition of $R^{meet}_{P^+}$. Hence, $P'' \cap CON = \emptyset$, and by Theorem 5, there is a P''-protecting minimal full contractor P^- of $\succ$ by CON which is also a P^+-protecting minimal full contractor. Since there is a CON-detour in which xy is unprotected by P^-, $xy \in P^-$. $\square$

We note that given the expressions for the meet and P^+-protecting full meet contractors in Theorems 6 and 7, one can easily obtain such contractors for finite and finitely representable relations: by evaluation of a relational algebra query in the former case and by quantifier elimination in the latter case.

Example 20 *Let a preference relation $\succ$ be a total order of $\{x_1, \ldots, x_5\}$ (Figure 17(a), the transitive edges are omitted for clarity). Let a base contractor CON be $\{x_1x_3, x_2x_3, x_2x_5\}$, and $P^+ = \{x_2x_4\}$.*

A full meet contractor P^{meet} of $\succ$ by CON is $\{x_1x_3, x_2x_3, x_2x_5, x_2x_4,$ x_3x_4, $x_4x_5\}$. The resulting contracted preference relation is shown in Figure 17(b). A P^+-protecting full meet contractor of $\succ$ by CON is $\{x_1x_3$, x_2x_3,



(a) $\succ$ and CON (b) $\succ - P^{meet}$ (c) $\succ$, CON, and P^+

(d) $\succ - P^{meet}_{P^+}$

Figure 17: Computing full meet contractor and P^+-protecting full meet contractor

x_2x_5, $x_4x_5\}$. *The resulting contracted preference relation is shown in Figure 17(d). Note that* C_{P^+} *here is* $CON \cup \{x_4x_5\}$.

8. Binary preference relations vs. preference states

The topic of the current paper is preference contraction in the *binary preference relation framework*. However, our paper is not the only one touching the subject of contracting preferences. Some relevant papers are considered in Section 9. One of the most fundamental works in this area is [18], where Hansson introduced the *preference state* framework, which is based on the belief revision theory. Even though the preference state and the binary preference relation frameworks are quite different, the fundamental principles of preference change operators in both are similar. Below we present some connections between the frameworks. In particular, we show an adaptation of the preference state framework to the preference relation framework. As a result, we obtain a framework that encompasses preference contraction and restricted preference revision.

User preferences in [18] are represented as *preference states*. A preference state is a logically closed set of sentences describing the preferences of an agent. Every preference state has an underlying set of preference *relations*. The connection between states and relations is as follows. A preference relation (which is an order on tuples) is an unambiguous description of the preferences of an agent. A preference relation induces a set of logical sentences which describe the relations. However, it is not always the case that the preferences of an agent are unambiguous. Hence, every preference *state* is associated with a *set* of possible preference relations.

Definition 12 *An* alternative *is an element of $\mathcal{U}$ (the universe of tuples). Nonempty subsets of $\mathcal{U}$ are called* sets of alternatives*. The language $\mathcal{L}_{\mathcal{U}}$ of sentences is defined as*

- *if $A, B \in \mathcal{U}$ then $A > B \in \mathcal{L}_{\mathcal{U}}$,*
- *if $A > B \in \mathcal{L}_{\mathcal{U}}$ then $\neg(A > B) \in \mathcal{L}_{\mathcal{U}}$,*
- *no other sentence is in $\mathcal{L}_{\mathcal{U}}$.*

A subset of $\mathcal{L}_{\mathcal{U}}$ is called a *restricted preference set.* The language defined above is a very restricted version of the language in [18] since the only Boolean operator allowed is negation. Throughout the discussion, we assume that the set of alternatives is fixed to be a nonempty subset $\mathcal{U}_r$ of $\mathcal{U}$.

Definition 13 *Let R be a subset of $\mathcal{U}_r \times \mathcal{U}_r$. The set $[R]$ of sentences is defined as follows:*

- *$A > B \in [R]$ iff $AB \in R$*
- *$\neg(A > B) \in [R]$ iff $A, B \in \mathcal{U}_r$ and $A > B \notin [R]$*

Definition 14 *A binary relation $R \subseteq \mathcal{U}_r \times \mathcal{U}_r$ is a* restricted preference model *iff it is a strict partial order. Given a restricted preference model R, the corresponding $[R]$ is called a* restricted preference state*.*

A relation R_S is a minimal representation *of a restricted preference state S iff R_S is a minimal relation such that $S \subseteq [R_S]$.*

In contrast to the definition above, the preference model in [18] is defined as a *set* of SPO relations, and a preference state is an intersection of $[R]$ for all members R of the corresponding preference model.

We define two operators of change of restricted preference states: revision and contraction. Restricted states are changed by sets of sentences. In [18], a change of a preference state by a set of sentences is defined as the corresponding change by the conjunction of the corresponding statements. Moreover, a change by any set of sentences is allowed. In the adaptation of that framework that we define here, conjunctions of statements are not a part of the language. Moreover, preference revision [11] only allows for adding new preferences, and preference contraction we have proposed in this paper allows only discarding existing preferences. Here we aim to define the operator of restricted preference set revision which captures the semantics of those two operators.

Definition 15 *A restricted preference set S is called* positive *iff it contains only sentences of the form $A > B$ for some $A, B \in \mathcal{U}_r$. Analogously, S is* negative *iff it contains only sentences of the form $\neg(A > B)$ for some $A, B \in \mathcal{U}_r$.*

A restricted preference set is a complement *of S (denoted as $\overline{S}$) if for all $A, B \in \mathcal{U}_r$, $A > B \in S$ iff $\neg(A > B) \in \overline{S}$ and $\neg(A > B) \in S$ iff $A > B \in \overline{S}$.*

Positive and negative restricted preference sets are used to change restricted preference states. Intuitively, a positive preference set represents the existence of preferences while a negative set represents a lack of preferences.

Definition 16 *Let R be a restricted preference model. Then the operator $*$ on R is a* restricted preference revision on R *iff for all positive or negative restricted preference sets S, $R * S = \cap\{R'\}$ for all R' such that*

1. *$S \subseteq [R']$*
2. *R' is an SPO*
3. *there is no SPO R'' with $S \subseteq [R'']$ such that $R \subseteq R'' \subsetneq R'$ (if S is positive) or $R' \subsetneq R'' \subseteq R$ (if S is negative).*

The last condition in the definition above expresses the minimality of restricted preference state change. This condition is different for positive and negative sets: when we add positive statements, we do not want to discard any existing positive sentences, and when negative statements are added, no new positive sentences should be added. The restricted preference revision operator defined above is different from preference state revision in [18]. First, preference state revision allows for revision by (finite) sets of arbitrary sentences, not only positive or negative sets of sentences, as here. Second, the minimality condition here is defined using set containment while in [18] it is defined as a function of symmetric set difference of the original preference relations and R'. The last difference is due to the preference state representation: the result of preference revision in [18] is the union of relations R' while in our case it is the intersection.

Below we define the operator of contraction for restricted preference states which is similar to the contraction of preference states.

Definition 17 *Let R be a restricted preference model. Then the operator $\div$ on R is* restricted preference contraction on R *iff for all positive or negative restricted preference sets S, $R \div S = R \ast \overline{S}$.*

Given the operators on restricted preference states we have defined here, their relationships with the preference change framework are straightforward.

Proposition 5 *Let R be a restricted preference model, S be a positive or negative restricted preference set, and R_S be a minimal representation of S. Then $R \ast S$ is*

1. *$\emptyset$, if S is a positive restricted preference set and $R \cup R_S$ has a cyclic path,*
2. *$TC(R \cup R_S)$, if S is a positive restricted preference set and $R \cup R_S$ has no cyclic paths,*
3. *$\cap\{R - P^- \mid P^-$ is a minimal full contractor of R by $\overline{R_S}\}$, if S is a negative restricted preference set,*

where TC is the transitive closure operator.

Proof. When a restricted preference model is revised by a positive preference set, the resulting relation $R \ast S$ is the intersection of all minimal SPO extensions R' of R and R_S (i.e., R' has to contain an edge from A to B if $A > B \in S$). Such an extension R' does not exist if there is a cyclic path in $R \cup R_S$. However, if no cyclic paths exist, then there is only one such a minimal extension R' which is equal to the transitive closure of $R \cup R_S$. Hence, $R \ast S = TC(R \cup R_S)$. We note that this result is equivalent to the result of the *union preference revision* [11].

When a restricted preference model is revised by a negative preference set, the resulting relation $R \ast S$ has to be a subset of R. Moreover, for all $\neg(A > B) \in S$, the pair (A, B) should not be in $R \ast S$. Hence, $R \ast S$ is the intersection of minimal contractions of R by $R_{\overline{S}}$, which is the result of the full meet contraction of R by $R_{\overline{S}}$. □

Below we list some properties of the revision and contraction operators of restricted preference states.

Proposition 6 *Let R be a restricted preference model and S be a positive or negative restricted preference set. Then*

1. *$R \ast S$ is an SPO (closure)*

2. *$S \subseteq [R\ ^*\ S]$ unless S is positive and $R_S \cup R$ has a cyclic path (limited success)*
3. *If $S \subseteq [R]$, then $R = R\ ^*\ S$ (vacuity)*

Proof. All the properties here follow from Proposition 5. Namely, property 1 follows from the fact that the result of $R\ ^*\ S$ is an SPO in every case of Proposition 5. Property 2 follows from Proposition 5 and the definition of $[R\ ^*\ S]$. Property 3 follows from Proposition 5 and 1) $S \subseteq [R]$ implies $R_S \subseteq R$ (if S is positive), and 2) a minimally contracted preference relation is equal to itself if contracted by non-existent edges (if S is negative). □

Proposition 7 *Let R be a restricted preference model and S be a restricted positive or negative preference set. Then*

1. *$R \div S$ is an SPO (closure)*
2. *$S \subseteq [R \div S]$ unless S is negative and $R_{\overline{S}} \cup R$ has a cyclic path (limited success)*
3. *If $S \cap [R] = \emptyset$, then $R = R \div S$ (vacuity)*
4. *$R\ ^*\ S = (R \div \overline{S})\ ^*\ S$ unless S is positive and $R_S \cup R$ has a cyclic path (limited Levi identity)*
5. *$R \div S = R\ ^*\ \overline{S}$ (Harper identity, by definition)*

Proof. Properties 1, 2, and 3 follow from Proposition 6. Property 4 follows from the fact that $R \div \overline{S} = R\ ^*\ S$ by definition, and Proposition 6 implies $R\ ^*\ S = (R\ ^*\ S)\ ^*\ S$ when either S is negative or S is positive but $R_S \cup R$ has no cyclic path. □

An important difference between the restricted preference-set change operators and the corresponding change operators from [18] is that the restricted versions are not always successful (property 2 in Proposition 5), and Levi identity holds for a certain class of restricted preference sets. In addition to that, the operator of preference set contraction in [18] has the property of inclusion ($R \subseteq R \div S$) and recovery (if $S \subseteq [R]$, then $R = (R \div S)\ ^*\ S$). As for the restricted framework defined here, inclusion does not hold due to the representation of a preference model as a single SPO relation. Recovery does not hold here due to the restrictions on the language (namely, not allowing disjunctions of sentences).

We note that one of the main targets of our current work was the development of an efficient and practical approach to contracting preference relations in the binary relation framework for the finite and the finitely representable cases. In addition to defining the semantics of the preference contraction operators, we have also developed a set of algorithms which can be used to compute contractions. We have tested them on real-life data and demonstrated their efficiency. In contrast, [18] focuses more on semantical aspects of preference change and does not address computational issues of preference change operators. In particular, finite representability is not addressed.

9. Related work

9.1. Other operators of preference relation change

A number of operators of preference relation change have been proposed in the literature. An operator of preference revision is defined in [11]. A preference relation there is *revised* by another preference relation called the *revising relation*. The result of revision is still another preference relation. [11] defines three versions of preference revision – union, prioritized, and Pareto – which are different in the way the original and the revising preference relations are composed. For all of these semantics, [11] identifies cases (called 0-*,* 1-*, and* 2-*conflicts*) when the revision fails, i.e., when there is no SPO preference relation satisfying the operator semantics. This work considers revising preference relations only by preference relations. Although it does not address the problem of discarding subsets of preference relations explicitly, revising a preference relation using Pareto and prioritized revision operators may result in discarding a subset of the original preference relation. It has been shown in [11] that the revised relation is an SPO for limited classes of the composed relations.

Another operator of preference relation change is defined in [5]. This work deals with a special class of preference relations called *skylines* [6]. (A tuple t is preferred to another tuple t' according to a *skyline preference relation* iff t is not worse that t' w.r.t. every attribute and better than t' w.r.t. to at least one attribute.) Preference relations in [5] are changed by *equivalence relations*. In particular, the modified preference relation is an extension of the original relation in which pairs of equivalent or incompatible tuples are ordered according to the new preferences. This preference change operator only adds new edges to the original preference relation, and thus, preference relation contraction cannot be expressed using this operator.

In [23], we introduced the operation of minimal preference contraction for preference relations. We studied properties of this operation and proposed algorithms for computing full contractors and preference-protecting full contractors for finitely stratifiable base contractors. In the current paper, we generalize this approach and we develop a method of checking the finite stratifiability property for finitely representable base contractors. We introduce the operations of meet and meet preference-protecting contraction, and propose methods for computing them. We also provide experimental evaluation of the framework and a comprehensive discussion of related work.

9.2. Belief revision and contraction

Preferences can be considered as a special form of human *beliefs*, and thus their change may be modeled in the context of belief change theory. The approach is to represent beliefs as truth-functional logical sentences. A *belief set* is the set of sentences believed by an agent. A common assumption is that belief sets are closed under logical consequence. The most common operators of belief set change are *revision* and *contraction* [3]. A number of versions of those operators have been proposed [19] to capture various real-life scenarios.

This approach is quite different from the preference relation approach. First, the language of truth functional sentences is rich and allows for rather complex statements about preferences: conditional preferences ($a > b \rightarrow c > d$), indefinite ($a > b \vee c > d$) etc. In contrast, preferences in the preference relation framework used in this paper are certain: given a preference relation $\succ$, it is only possible to check if a tuple is preferred (or not) to another tuple. In addition, belief revision is generally restricted to finite domains. We have proposed here algorithms for contracting finite and infinite preference relations.

9.3. Other frameworks

An approach to preference change is proposed in [9]. Preferences are changed via *interactive example critiques.* This paper identified three types of common critique models: similarity-based, quality-based, and quantity-based. However, no formal framework is provided. [17] describes revision of *rational* preference relations over propositional formulas. The proposed revision operator satisfies the postulates of success and minimal change. The author shows that the proposed techniques work in case of revision by a single statement and can be extended to allow revisions by multiple statements.

[14] proposes algorithms of incremental maintenance of the transitive closure of graphs using relational algebra. The graph modification operations

are edge insertion and deletion. Transitive graphs in [14] consist of two kinds of edges: the edges of the original graph and the edges induced by its transitive closure. When an edge xy of the original graph is contracted, the algorithm also deletes all the transitive edges uv such that all the paths from u to v in the original graph go through xy. As a result, such contraction is not minimal according to our definition of minimality. Moreover, [14] considers only finite graphs, whereas our algorithms can work with infinite relations.

10. Conclusions and future work

In this paper, we have presented an approach to contracting preference relations. We have considered several operators of preference contraction: minimal preference contraction, minimal preference-protecting contraction, and (preference-protecting) meet contraction, inspired by different scenarios of preference change. We have proposed algorithms and techniques for computing contracted preference relations given finitely stratifiable base contractors. We have also evaluated the proposed algorithms experimentally (Appendix A) and showed that they can be used in real-life database applications.

One of the areas of future work is to relax the finite stratifiability property property and consider more general base contractors.

Another interesting direction of future work is to design an operator of generalized preference relation change that allows to change preference relations by discarding existing as well as adding new preferences at the same time. The current approaches of preference relation change are restricted to only one type of change.

As we showed in the discussion of related work, the existing preference revision approach [11] fails to work in the presence of conflicts (cycles). A promising direction here is to use the preference contraction operators presented here to resolve such conflicts.

In this paper, we assume that the relations defining the preferences to discard are explicitly formulated by the user. However, such an assumption hardly works in practical scenarios of preference change: formulating such a relation requires a full knowledge of his or her preferences, which may not be the case. Hence, a promising direction is to perform interactive preference contraction or change.

Acknowledgements

We greatly appreciate the conversations with Matteo Cristani and the comments of the anonymous reviewers.

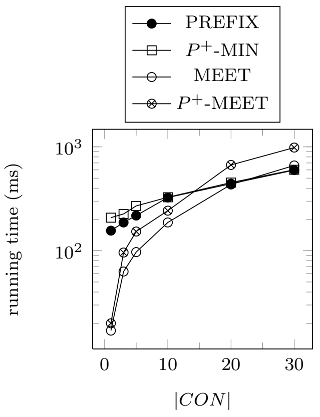
Appendix A. Experimental evaluation

In this section, we present the results of an experimental evaluation of the preference contraction framework proposed here. We implemented the following operators of preference contraction: prefix contraction (denoted as PREFIX), preference-protecting minimal contraction (P^+-MIN), meet contraction (MEET), and preference-protecting meet contraction (P^+-MEET). PREFIX was implemented using Algorithm 3, P^+-MIN according to Theorem 5, MEET according to Theorem 6, and P^+-MEET according to Theorem 7. We used these operators to contract finite preference relations stored in a database table $R(X, Y)$. The preference relations used in the experiments were finite *skyline preference relations* [6], defined in Section 9. Skyline preference relations are often used in database applications. We note that such relations are generally not materialized (as database tables) when querying databases with skylines. However, they may be materialized in scenarios of preference elicitation [4]. To generate such relations, we used the NHL 2008 Player Stats dataset [2]. Each tuple has 18 different attributes out of which we used 5. All algorithms used in the experiments were implemented in Java 6. We ran the experiments on Intel Core 2 Duo CPU 2.1 GHz with 2.0 GB RAM. All tables were stored in a PostgreSQL 8.3 database.

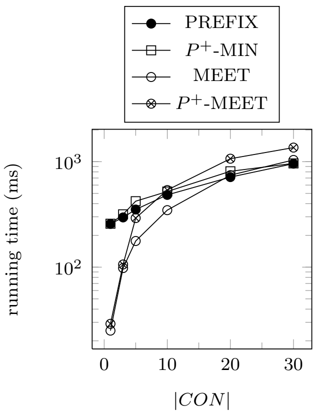
We have carried out two sets of experiments with the preference contraction algorithms. In the first set, we model the scenario where base contractors are manually constructed by user. Thus, we assume that such base contractors are of comparatively small size. In the second set of experiments, we assume that base contractors are constructed automatically and hence may be of larger size.

10.1. Small base contractors

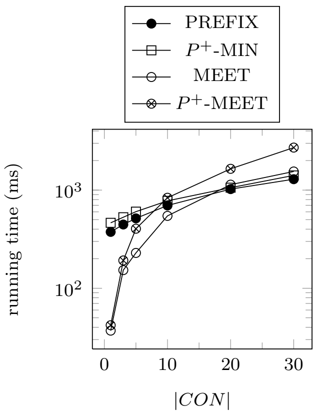
The aim of the experiments shown here is twofold. First, they demonstrate that the algorithms of preference contraction we have proposed have good performance (given base contractors of small size). Second, they show that the difference between the sizes of full contractors computed by different algorithms may be significant. It implies that in real-life applications, an



(a) $|\succ| = 51887$



(b) $|\succ| = 80612$



(c) $|\succ| = 116411$

Figure 18: Contraction performance. Small base contractors

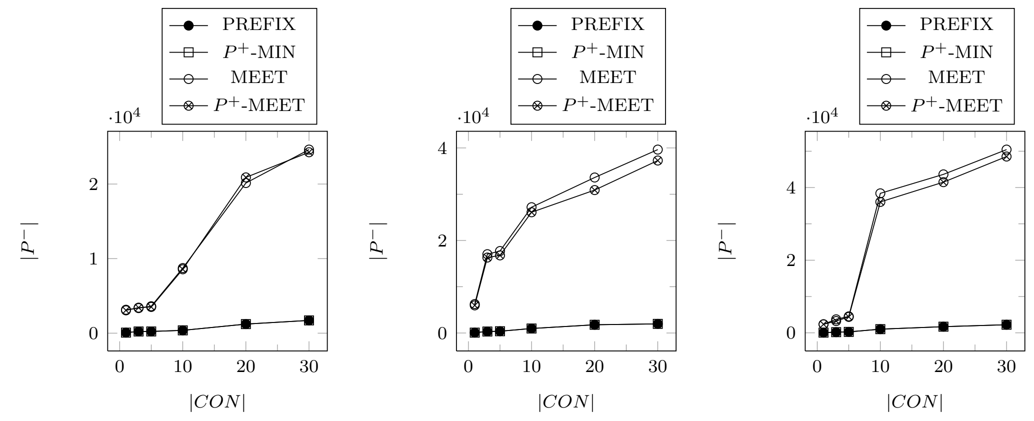
appropriate contraction algorithm needs to be selected carefully depending on the required semantics.

The skyline preference relations we use here consist of 51887, 80612, and 116411 edges. To generate them, we used 400, 500, and 600 tuples, respectively, out of 852 tuples in [2].

The size of base contractors ranges from 1 to 30 edges. We did not pick more than 30 edges, assuming that in this scenario the user is unlikely to provide a large set of preferences to be discarded. For every base contractor size, we randomly generated 10 different base contractors and computed the average time spent to compute full contractors and their average size. The relations P^+, storing preferences to protect, were transitive relations containing from 1% to 5% of edges of the corresponding preference relation.

Figure 18 shows how the running time of contraction operators depends on the size of the preference relation to contract and the size of the base contractor. Here, P^+ contains 1% of $\succ$. As we can observe, the performance of P^+-MIN is slightly worse than the one of PREFIX, due to the need of computing the set Q. Similarly, the running time of P^+-MEET is worse than the running time of MEET, due to the computation of C_{P^+}.

Figure 19 shows the dependency of the minimal full contractor size on the size of the preference relation and the size of the base contractor. For every value of the base contractor size, the charts show the average size of the corresponding full contractor. Notice that preference protection does not much affect the size of the full contractors computed by PREFIX and P^+-



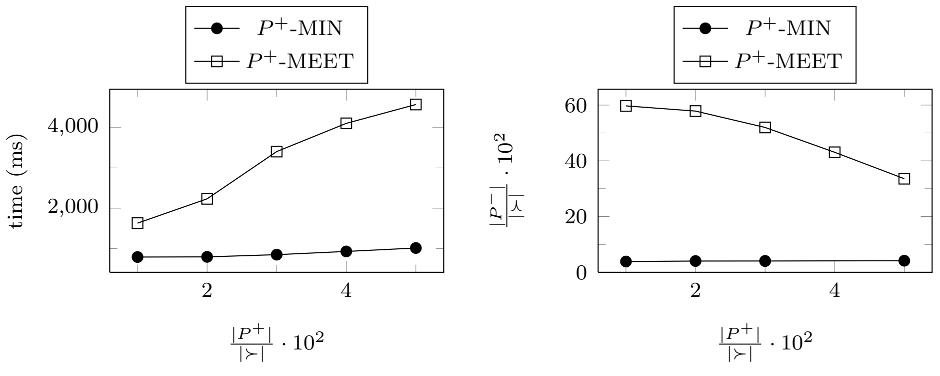
(a) $|\succ| = 51887$ (b) $|\succ| = 80612$ (c) $|\succ| = 116411$

Figure 19: Full contractor size. Small base contractors

MIN – they almost always coincide. That is due to the fact that even though the full contractors computed by these algorithms have different properties, they are both *minimal*. The size of full contractors computed by MEET is strictly less than the size of full contractors computed by P^+-MEET. This is justified by the semantics of those full contractors: a full meet contractor is a union of all minimal full contractors of $\succ$ by CON, while a full P^+-protecting meet contractor is a subset of full meet contractor.

Another important observation is that the size of minimal full contractors (PREFIX and P^+-MIN) and the size of full meet contractors (MEET and P^+-MEET) differ significantly. Hence, minimality has a significant effect on the size of full contractors when base contractors are small.

Figure 20 shows how the algorithm performance and the size of computed full contractors depend on the size of the protected preference set P^+. As we can observe in Figure 20(a), the size of P^+ mostly affects the running time of P^+-MEET, while the running time of P^+-MIN grows slowly. The reason is that the computation of C_{P^+} (used in P^+-MEET) involves more joins of the tables representing P^+ and $\succ$ than the computation of Q used in P^+-MIN. Figure 20(b) shows how the size of a full contractor varies with the size of P^+. As expected, the size of a P^+-protecting minimal full contractor is almost always the same. The size of the full P^+-protecting meet contractor goes down when P^+ grows, because then fewer minimal full contractors protect P^+.



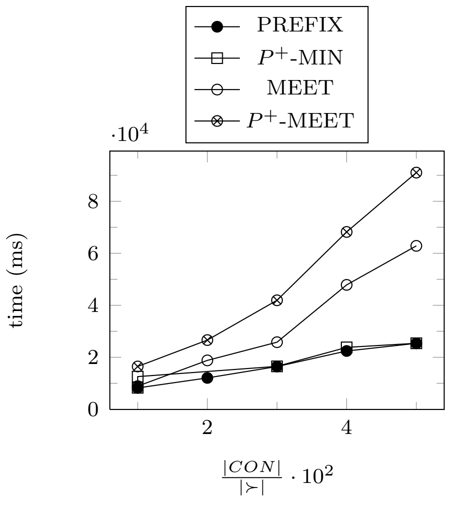
(a) Contraction performance (b) Full contractor size

Figure 20: Small base contractors. Varying $|P^+|$

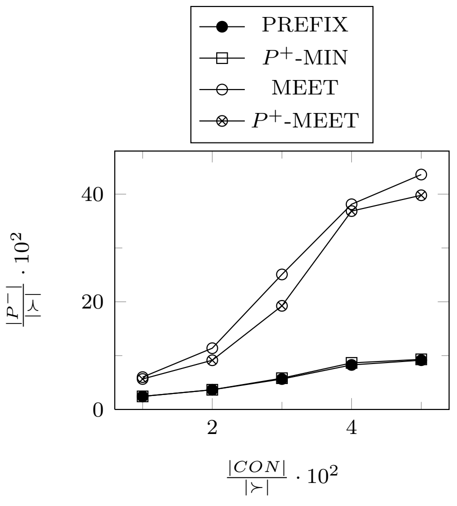
10.2. Medium size base contractors

We notice that according to Figure 18, the time spent to compute a full contractor using any algorithm does not exceed 3 seconds. In Figure 21(a), we show the running time of the algorithms versus the relative size of CON, which is larger in this experiment than in the previous one. The size of the preference relation here is 80612, the size of P^+ is 1% of $|\succ|$, and the size of CON varies from 1% (806 edges) to 5% of (4030 edges) of $|\succ|$. As we can see, the running time grows quadratically with the size of CON, which is consistent with Proposition 4. Figure 21(b) shows how the size of the full contractors changes with $|CON|$. Notice that the size of the minimal full contractors grows significantly slower with $|CON|$: when $|CON|$ is 5% of $|\succ|$, the size of the full meet contractor exceeds 40% of $|\succ|$, while the size of the minimal full contractors is not greater than 10% of $|\succ|$. As in the case of small CON, the sizes of minimal and meet full contractors differ greatly. Hence, in real-life scenarios, it is important to know the semantics of preference contraction the user intends, since that has a high effect on the contraction result.

In the experiments above, we use a real-life data set of NHL player stats. As preference relations, we used skylines. As we have observed, with CON of small size, the time spent to compute any full contractor did not go beyond a few seconds, regardless of the size of $\succ$. Hence, the algorithms we proposed to contract finite relations can be used efficiently in such scenarios in database applications. However, when CON is large, additional optimization



(a) Contraction performance



(b) Full contractor size

Figure 21: Medium size base contractors

techniques are needed.

Appendix B. Proof of Theorem 3

Theorem 3. (Checking the finite stratifiability property). *Let F_R be an ERO-formula in DNF, representing an SPO relation R, of the following form*

$$F_R(o, o') = F_{R_1}(o, o') \vee \ldots \vee F_{R_l}(o, o'),$$

where F_{R_i} is a conjunction of atomic formulas. Then checking if there is a constant k such that the length of all R-paths is at most k can be done by a single evaluation of QE over a formula of size linear in $|F_R|$.

Let R_i be a binary relation represented by the formula F_{R_i} for all $i \in [1, l]$. We split the proof of Theorem 3 into several lemmas. In Lemma 4, we show that the length of all R-paths is bounded by a constant iff the length of all R_i-paths is bounded by a constant for every disjunct F_{R_i} of F_R. Lemma 5 shows that the length of all R_i-paths is bounded by a constant iff there is a bound on the length of all paths in the graph of a binary relation represented by at least one conjunct of F_{R_i}. In Lemma 6, we show how to check if the length of all paths in the graph represented by F_{R_i} is bounded.

To prove the first lemma, we use the following idea. Let a sequence $S = (o_1, \dots, o_n)$ of $n \geq 2$ tuples be an R-sequence, i.e.,

$$(o_1, o_2), \dots, (o_{n-1}, o_n) \in R \tag{1}$$

The transitivity of R implies that there is an R-edge from o_1 to all other tuples in S, i.e.,

$$(o_1, o_2), \dots, (o_1, o_n) \in R \tag{2}$$

Note that (2) contains only edges started by o_1. Since $R = \cup_{i=1}^{l} R_i$, for every R-edge in (2), there is $i \in [1, l]$ such that it is also an R_i-edge. Let R_j (not necessarily unique) for some $j \in [1, l]$ be such that the number of R_j-edges in (2) is maximum. Such R_j is called *a major component of S*. Let the sequence S' consist of the end nodes of all these R_j-edges in the order they appear in S. Such S' is called *a major subsequence of S*.

Observation 1 *Let S be an R-sequence, R_{i^*} a major component of S, and S' the corresponding major subsequence of S. Then*

1. *S' is an R-sequence*
2. *if the length of S is n, then the length of S' is at least $\lceil \frac{n-1}{l} \rceil$*

The first fact of Observation 1 follows from transitivity of R, and the second fact follows from the definition of major subsequence. Note that a major subsequence is an R-sequence too. Hence, if it has at least two tuples, we can construct its major subsequence.

Observation 2 *Let $S_0, \dots, S_t$ be R-sequences such that for all $i \in [1, t]$, S_i is a major subsequence of S_{i-1} with the corresponding major components R_{j_i}. Let o_0, o_t be the first tuples of S_0 and S_t correspondingly. Then $R_{j_1}(o_0, o_t)$.*

Observation 2 follows from the definition of major subsequence.

Example 21 *Let $S_0 = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12})$ be an R-sequence. Figure 22 illustrates a possible construction of a major subsequence S_1 of S_0, a major subsequence S_2 of S_1, and a major subsequence S_3 of S_2. The edges on Figure 22 correspond to the major-component edges. In every sequence, a node is dark if it is in the major subsequence of the sequence. Note that S_3 does not have a major subsequence because a subsequence has to have at least two nodes.*

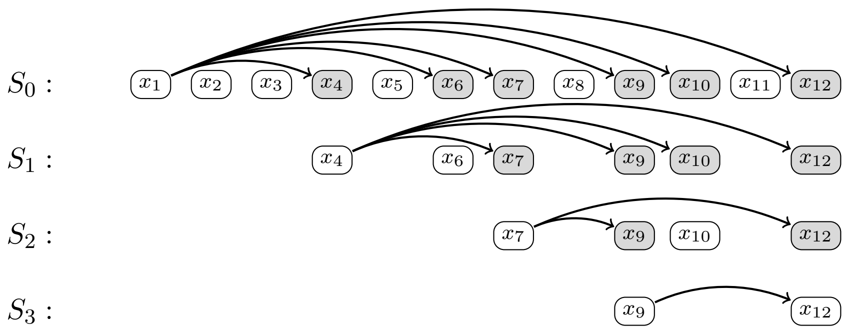


Figure 22: Major subsequences

Lemma 4 *There is a constant bounding the length of all R-paths iff for all $i \in [1, l]$, there is a constant bounding the length of all R_i-paths.*

PROOF. In the case when $l = 1$, the lemma trivially holds. Further we assume $l > 1$.
$\boxed{\Rightarrow}$ If for some $i \in [1, l]$, the length of R_i-paths cannot be bounded, neither can the length of R-paths.
$\boxed{\Leftarrow}$ Assume that for all $i \in [1, l]$, all R_i-paths are of length at most k. Show that the length of all R-paths is not more than $\sum_{i=1}^{(k+2)l+1} l^i - 2$. For the sake of contradiction, let there be an R-path of length $\sum_{i=1}^{(k+2)l+1} l^i - 1$. Let S_0 be the corresponding R-sequence. The length of S_0 is $\sum_{i=0}^{(k+2)l+1} l^i$. Let S_1 be a major subsequence of S_0. By Observation 1, S_1 is also an R-sequence, and its length is at least $\sum_{i=0}^{(k+2)l} l^i$. Following that logic, let S_t be a major subsequence of S_{t-1} with the corresponding major component $R_{j_{t-1}}$. The size of S_t is at least $\sum_{i=0}^{(k+2)l-t+1} l^i$. Such computation may continue while the size of S_t is greater than one, i.e., while $t \leq (k+2)l$. Let the major components of $S_1, \ldots, S_{(k+2)l}$ be $R_{j_1}, \ldots, R_{j_{(k+2)l}}$ correspondingly. Note that there are at most l possible different major components. Thus, at least $k+2$ major components in $R_{j_1}, \ldots, R_{j_{(k+2)l}}$ are the same. Let us denote the first $k+2$ of them as $R_{t_1}, \ldots, R_{t_{k+2}}$ and the tuples which start the corresponding major sequences as $o_{t_1}, \ldots, o_{t_{k+2}}$. By Observation 2,

$$R_{t_1}(o_{t_1}, o_{t_2}) \wedge R_{t_2}(o_{t_2}, o_{t_3}) \wedge \ldots \wedge R_{t_{k+1}}(o_{t_{k+1}}, o_{t_{k+2}})$$

Since all $R_{t_1}, \ldots, R_{t_{k+2}}$ are the same, the expression above implies that there is an R_i-path of length $k+1$ for some $i \in [1, l]$ which is a contradiction. $\square$

In Lemma 4, we showed that the problem of checking the bounded-length property of all R-paths can be reduced to the problem of testing the same property for R_i-paths. Note that R_i is represented by a formula F_{R_i} which is a conjunction of atomic formulas. Let the set of all attributes which are present in the formula F_{R_i} be defined as $\mathcal{A}_{F_{R_i}}$. Then F_{R_i} can be represented as

$$F_{R_i}(o, o') = \bigwedge_{A \in \mathcal{A}_{F_{R_i}}} \lambda_A(o, o'),$$

where $\lambda_A(o, o')$ is a conjunction of all atomic formulas in which the attribute A is used. Note that the structure of the preference formula language implies that *every atomic formula belongs to exactly one* λ_A.

Denote the relation represented by λ_A as Λ_A. In the next lemma, we show that the problem of checking the finite stratifiability property of all R_i-paths can be reduced to the same problem for Λ_A-paths.

Lemma 5 *There is a constant bounding the length of all R_i-paths iff for some $A \in \mathcal{A}_{F_{R_i}}$, there is a constant bounding the length of all Λ_A-paths.*

Proof.
$\boxed{\Leftarrow}$ Let for every k, there be an R_i-path of length at least k

$$R_i(o_1, o_2) \wedge R_i(o_2, o_3) \wedge \ldots \wedge R_i(o_k, o_{k+1})$$

Then for all $A \in \mathcal{A}_{F_{R_i}}$, we have a Λ_A-path of length at least k

$$\Lambda_A(o_1, o_2) \wedge \Lambda_A(o_2, o_3) \wedge \ldots \wedge \Lambda_A(o_k, o_{k+1})$$

$\boxed{\Rightarrow}$ Let for every k and $A \in \mathcal{A}_{F_{R_i}}$, there be an Λ_A-path of length at least k

$$\Lambda_A(o_1^A, o_2^A) \wedge \Lambda_A(o_2^A, o_3^A) \wedge \ldots \wedge \Lambda_A(o_k^A, o_{k+1}^A)$$

Construct a sequence of tuples $(o_1, o_2, o_3, \ldots)$ as follows. Let $o_j.A = o_j^A.A$ if $A \in \mathcal{A}_{F_{R_i}}$. Otherwise, let $o_j.A$ be any value from the domain $\mathcal{D}_A$ of A. Clearly, the following R_i-path is of length at least k

$$R_i(o_1, o_2) \wedge R_i(o_2, o_3) \wedge \ldots \wedge R_i(o_k, o_{k+1}).$$

□

Lemma 6 *There is a constant bounding the length of all Λ_A-paths iff there is no Λ_A-path of length three, i.e.,*

$$\neg\exists o_1, o_2, o_3, o_4 \in \mathcal{U} \ . \ \Lambda_A(o_1, o_2) \wedge \Lambda_A(o_2, o_3) \wedge \Lambda_A(o_3, o_4)$$

Proof.
$\boxed{\Leftarrow}$ If for every constant k, there is a Λ_A-path of length at least k, then there is a Λ_A-path of length three.
$\boxed{\Rightarrow}$ If λ_A is unsatisfiable, then there are no Λ_A-paths. Thus, we assume that λ_A is satisfiable. Based on the preference formula language, the formula $\lambda_A(o, o')$ can be split into at most three conjunctive formulas:

1. ϕ_L: a conjunction of all atomic formulas $o.A\theta c$,
2. ϕ_R: a conjunction of all atomic formulas $o'.A\theta c$,
3. ϕ_M: a conjunction of all atomic formulas $o.A\theta o'.A$

for $\theta \in \{=, \neq, <, >\}$ and a C - or Q -constant c. Any of these three formulas may be missing because λ_A may not contain atomic formulas of the specified type. ϕ_L and ϕ_R capture the range of the left and the right argument in λ_A, correspondingly, and ϕ_M constrains their relationship.

Here we assume that A is a Q -attribute, and the case of C -attributes is similar. Note that if ϕ_L is defined, then the range r_L of ϕ_L is 1) an open rational number interval with a finite number of holes (due to possible atomic formulas $o.A \neq c$), or 2) a single rational value (due to the formula $o.A = c$). If ϕ_L is undefined, then r_L is the entire set of rational numbers. Thus, the number of distinct elements $|r_L|$ in r_L is either ∞ or 1. The same holds for the number of distinct elements $|r_R|$ in r_R. Hence for our class of formulas, $|r_L \cap r_R| \in \{0, 1, \infty\}$. Clearly, if $|r_L \cap r_R| = 0$, then no Λ_A-paths exist. So we we assume that $|r_L \cap r_R| \in \{1, \infty\}$.

Consider the structure of ϕ_M. If ϕ_M is undefined, then $|r_L \cap r_R| > 0$ implies that there are Λ_A-paths of length at least k for every k, consisting of tuples whose A-values are arbitrary elements of $r_L \cap r_R$. If "$o.A = o'.A$" $\in \phi_M$, then no other atomic formula is in ϕ_M (otherwise, Λ_A is unsatisfiable). Since $|r_L \cap r_R| > 0$, Λ_A-paths of length at least k for every k can be constructed of tuples with the value of A all equal to any member of $r_L \cap r_R$. If "$o.A > o'.A$" $\in \phi_M$, then "$o.A = o'.A$", "$o.A < o'.A$" $\notin \phi_M$ (otherwise λ_A is unsatisfiable). However, "$o.A \neq o'.A$" may be in ϕ_M and is implied by "$o.A > o'.A$" $\in \phi_M$ so can be dropped. The existence of a Λ_A-path of length three implies that $|r_L \cap r_R| > 1$ and thus $|r_L \cap r_R| = \infty$. Hence there are Λ_A-paths of length at least k for every k. The case of "$o.A < o'.A$" $\in \phi_M$ is analogous. The last case is when only "$o.A \neq o'.A$". The existence of a Λ_A-path of length three implies that there are two different values $c_1, c_2 \in r_L \cap r_R$. Hence, Λ_A-paths of length at least k for every k can be constructed by taking any sequence of tuples in which the value of A of every even tuple is c_1 and of every odd tuple is c_2. $\square$

Proof of Theorem 3. Here we show how to construct a formula which is true iff there is a constant k such that the length of all R-paths is bounded by k. By Lemma 4, such a formula can be written as a conjunction of l formulas each of which represents the fact that the length of all R_i-paths is bounded. By Lemma 5, such a formula can be written as a disjunction of formulas each of which represents the fact that the length of all Λ_A-paths is bounded. By Lemma 6, such formulas are of size linear in the size of Λ_A. Hence, the resulting formula is linear in the size of F_R. Due to the construction in Lemma 6, the formula has quantifiers. They can be eliminated using QE. $\square$

References

[1] Matteo Cristani, personal communication.

[2] NHL.com Stats, 2008. `http://www.nhl.com/ice/playerstats.htm`.

[3] C. E. Alchourron, P. Gardenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[4] W.-T. Balke, U. Güntzer, and C. Lofi. Eliciting matters - controlling skyline sizes by incremental integration of user preferences. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 551–562. Springer, 2007.

[5] W.-T. Balke, U. Guntzer, and W. Siberski. Exploiting indifference for customization of partial order skylines. In *Proceedings of the Tenth International Database Engineering and Applications Symposium (IDEAS)*, pages 80–88, Delhi, India, 2006. IEEE Computer Society.

[6] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[7] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research (JAIR)*, 21:135–191, 2004.

[8] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 503–514. Chicago, Illinois, USA, 2006.

[9] L. Chen and P. Pu. Evaluating critiquing-based recommender agents. In *Proceedings of AAAI-2006*, pages 157–162, Boston, Massachusetts, USA, 2006. AAAI Press.

[10] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[11] J. Chomicki. Database querying under changing preferences. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):79–109, 2007.

[12] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 717–719. IEEE Computer Society, March 2003.

[13] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, September 2001.

[14] G. Dong, L. Libkin, J. Su, and L.Wong. Maintaining the transitive closure of graphs in SQL. *Int. Journal of Information Technology*, 5:46–78, 1999.

[15] J. Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.

[16] P. C. Fishburn. *Utility theory for decision-making.* Wiley, New York, 1970.

[17] M. Freund. On the revision of preferences and rational inference processes. *Artificial Intelligence*, 152(1):105–137, 2004.

[18] S. O. Hansson. Changes in preference. *Theory and Decision*, 38(1):1–28, 1995.

[19] S. O. Hansson. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 3: Belief Change, chapter Revision of belief sets and belief bases, pages 17–75. Springer, Dordrecht, October 1998.

[20] S. O. Hansson and T. Grüne-Yanoff. Preferences. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, page http://plato.stanford.edu/archives/win2006/entries/preferences/. 2006.

[21] P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, pages 26–52, 1995.

[22] W. Kießling. Foundations of preferences in database systems. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, pages 311–322, Hong Kong, China, 2002. Morgan Kaufmann.

[23] D. Mindolin and J. Chomicki. Minimal contraction of preference relations. In *Proceedings of AAAI-2008*, pages 492–497, Chicago, Illinois, USA, 2008. AAAI Press.

[24] R. Ramakrishnan and J. Gehrke. *Database Management Systems.* McGraw-Hill Science/Engineering/Math, August 2002.