Soft Cardinality Constraints on XML Data
How Exceptions Prove the Business Rule

Flavio Ferrarotti[1], Sven Hartmann[2], Sebastian Link[3], Mauricio Marin[4], and Emir Muñoz[5]

[1] Victoria University of Wellington
[2] Clausthal University of Technology
[3] The University of Auckland
[4] Yahoo! Research
[5] DERI, National University of Ireland Galway

Contact e-mail: `flavio.ferrarotti@vuw.ac.nz`

Abstract. We introduce soft cardinality constraints which need to be satisfied on average only, and thus permit violations in a controlled manner. Starting from a highly expressive but intractable class, we establish a fragment that is maximal with respect to both expressivity and efficiency. More precisely, we characterise the associated implication problem axiomatically and develop a low-degree polynomial time decision algorithm. Any increase in expressivity of our fragment results in coNP-hardness of the implication problem. Finally, we extensively test the performance of our algorithm. The performance evaluation provides first-hand evidence that reasoning about expressive notions of soft cardinality constraints on XML data is practically efficient and scales well. Our results unleash soft cardinality constraints on real-world XML practice, where a little more semantics makes applications a lot more effective in contexts where exceptions to common rules may occur.

1 Introduction

Cardinality constraints are a very natural class of constraints that can be observed easily and can express a lot of semantics important to applications such as consistency management, data integration, query optimization, view maintenance and cardinality estimation. Generally speaking, cardinality constraints capture information about the frequency with which certain data items occur in particular contexts.

Example 1. Suppose we use XML to store data about teams involved in projects within a research institute. Figure 1 shows an XML tree representing a small fragment of such an XML document. The nodes are annotated by their type: E for element nodes, A for attribute nodes, and S for text nodes. Of course, in reality there would be far more data stored in the XML document. We use the simplified example to illustrate how cardinality constraints can express important semantic properties of XML data. We assume that each project has a manager and that several research teams (RTeam) and support teams (STeam) can be involved in a given project. Technicians (Tech) belong to support teams, Scientists (Sci) belong to research teams and Engineers (Eng) can belong to both, support and research teams. Some of the semantic properties which can be expressed by means of cardinality constraints are:

1. Every scientist is a member of 2, 3, or 4 research teams.
2. Every technician can work in up to 4 different support teams.
3. Every engineer is in 4 dififerent support teams and 1 or 2 research teams.
4. A project cannot have more than one manager.
5. Every support team is involved in 1, 2, or 3 projects.
6. Every research team is involved in up to 2 different projects.
7. A maximum of 2 support teams and 2 research teams should be involved in a given project.
8. In every team, there should be two employees for each expertise level.
9. Within a given project, an employee cannot belong to more than one group.
10. No more than 5 employees of a given expertise level can be involved in the same project. □

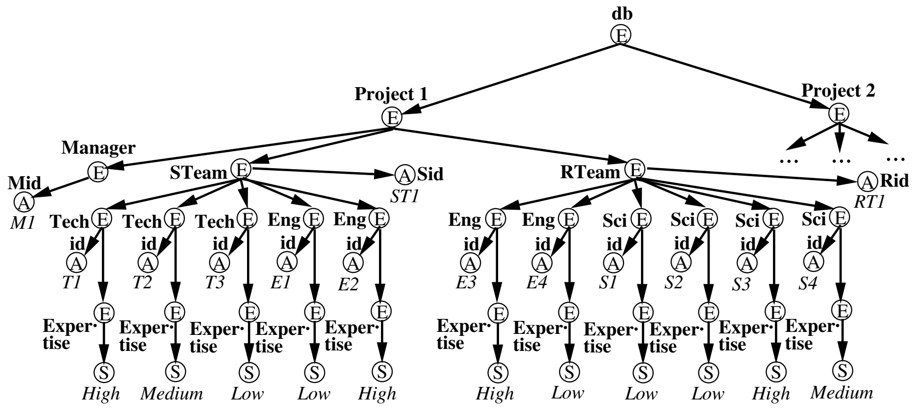

Fig. 1. Fragment of an XML tree with information on projects and teams.

For the effectiveness of XML applications, it is an important problem to identify useful classes of cardinality constraints. This is challenging, since for semi-structured XML data more so than for structured data, the more semantics of application domains can be captured the less efficient reasoning about these constraints usually becomes (see [1, 2, 8] among others).

In addition, exceptions to business rules are the common norm for (XML) data, and are therefore difficult to handle by strictly enforced constraints, which makes them often impractical to exploit. Indeed, it is often difficult to specify exact bounds for cardinality constraints. In many cases we only have a good idea of what rules should apply to most data.

Example 2. The constraints in Example 1 express rules that apply to most projects in the research institute, but clearly exceptions may occur. For instance, according to the first cardinality constraint in that example, every scientist is a member of 2, 3, or 4 research teams. However, it is likely that some scientists participate in 5 research teams or more. Nevertheless this will be an exception to most cases. □

The previous example motivates the idea to treat cardinality constraints as *soft* constraints, which do not strictly enforce the constraint on all data, but on most data. Under this interpretation, soft cardinality constraints express an ideal or a preferred situation while still allowing room for violations of the strict constraints.

Example 3. The first cardinality constraint in Example 1 is better interpreted as a soft cardinality constraint. We can expect that scientists *on average* participate in 2 to 4 research teams, or that the number of researchers working in less than 2 and more than 4 research teams is considerably small. Similar observations apply to the other cardinality constraints in Example 1, too. □

Paper Organization. In Section 2 we assemble useful technical notations on XML trees. In Section 3, we introduce soft cardinality constraints to effectively deal with the problem that exceptions confirm the rule in practice. Soft cardinality constraints need to be satisfied on average only, and thus permit exceptions to the general business rules in a controlled manner. We start by defining a highly expressive class of soft cardinality constraints. This class can be used to capture a wide range of interesting semantic properties of XML data, such as those described in Example 1. It allows data engineers to specify "soft" bounds on the number of nodes in an XML tree that are equal on some of their subnodes. These "soft" bounds can be violated by individual nodes, but should be respected on average. Bounds can be specified with respect to a context node. The soft constraints use a very general notion of value equality which is not restricted to leaf nodes, and an expressive path language to select nodes using single-label wildcards, child navigation, and descendant navigation from XPath.

In Section 4 we focus on a particular fragment of soft cardinality constraints, and characterise the associated implication problem axiomatically. In Section 5 we develop a low-degree polynomial time algorithm for deciding implication. The fragment under investigation is still expressive, allowing for instance to express most of the cardinality constraints in Example 1. Any increase in expressivity of our fragment is likely to cause intractability of the implication problem. There is great potential for practical uses of the proposed decision algorithm. For example, the process of checking XML data integrity against soft cardinality constraints can benefit a lot from the ability to decide implication efficiently. Clearly, if a set Σ of soft cardinality constraints implies a soft cardinality constraint φ and we have already checked that an XML data tree satisfies Σ then there is no need to test φ anymore, thus saving considerable resources. Section 5 reports on the outcomes of a performance anaylsis that provides first-hand evidence that reasoning about our fragment of soft cardinality constraints is practically efficient and scales well. Our results unleash cardinality constraints to application contexts where a little more semantics makes applications a lot more effective and where exceptions can be tolerated.

Related Work. The topic of XML constraints has attracted much attention over the last decade (see [5, 10, 11, 17] among others). However, as far as we know, this article is the first to explore the concept of *soft* constraints in the context of XML. It is not the first time that cardinality constraints for XML data are studied [6], and they have a long and successful history in the field of database design (see [16] for a recent survey). Soft constraints are not new in the context of database design, where deontic logic has long been used as a tool to model soft constraints, see [12] for a survey. More recently, soft constraints have also been studied in the context of the constraint satisfaction problem (see [4, 9, 13] among others) where constraints are often soft in the sense that they do not have to be satisfied for a solution to be acceptable. None of these works, however, deals with the implication problem for soft constraints on XML data.

2 Basic Terminology

We use the common tree model of XML for our investigation. Let $\mathbf{E}$ denote a countably infinite set of element tags, $\mathbf{A}$ a countably infinite set of attribute names, and let S represent simple text data in XML (PCDATA). These sets are pairwise disjoint. The elements of $\mathcal{L} = \mathbf{E} \cup \mathbf{A} \cup \{S\}$ are called *labels*. An *XML tree* is a 6-tuple $T = (V, lab, ele, att, val, r)$ where V is a set of nodes, and lab is a mapping $V \to \mathcal{L}$ assigning a label to every node in V. A node $v \in V$ is an *element node* if $lab(v) \in \mathbf{E}$, an *attribute node* if $lab(v) \in \mathbf{A}$, and a *text node* if $lab(v) = S$. Moreover, ele and att are partial mappings defining the edge relation of T: for any node $v \in V$, if v is an element node, then $ele(v)$ is a list of element and text nodes, and $att(v)$ is a set of attribute nodes in V. If v is an attribute or text node, then $ele(v)$ and $att(v)$ are undefined. The partial mapping val assigns a string to each attribute and text node: for each node $v \in V$, $val(v)$ is a string if v is an attribute or text node, while $val(v)$ is undefined otherwise. Finally, r is the unique and distinguished root node.

A *path* p of T is a finite sequence of nodes $v_0, \ldots, v_m$ in V such that (v_{i-1}, v_i) is an edge of T for $i = 1, \ldots, m$. The path p determines a word $lab(v_1). \cdots .lab(v_m)$ over the alphabet $\mathcal{L}$, denoted by $lab(p)$. For navigation in the XML tree, we use the path language $PL^{\{.,\_,\_^*\}}$ consisting of words given by the following grammar: $Q \to \ell \mid \varepsilon \mid Q.Q \mid \_ \mid \_^*$. Here $\ell \in \mathcal{L}$ is any label, ε denotes the empty path expression, "." denotes the concatenation of two path expressions, "_" denotes the *single-label* wildcard, and "_*" denotes the *variable length don't care* wildcard. Let P, Q be words from $PL^{\{.,\_,\_^*\}}$. P is a *refinement* of Q, denoted by $P \lesssim Q$, if P is obtained from Q by replacing variable length wildcards in Q by words from $PL^{\{.,\_,\_^*\}}$ and single-label wildcards in Q by labels from $\mathcal{L}$. Let Q be a word from $PL^{\{.,\_,\_^*\}}$. A path p in the XML tree T is called a Q*-path* if $lab(p)$ is a refinement of Q. For a node v of T, $v[\![Q]\!]$ denotes the set of nodes in T that are reachable from v following any Q-path.

We use $[\![Q]\!]$ as an abbreviation for $r[\![Q]\!]$ where r is the root node. For $\mathcal{S} \subseteq \{., \_, \_^*\}$, $PL^{\mathcal{S}}$ denotes the subset of $PL^{\{.,\_,\_^*\}}$ expressions restricted to the constructs in $\mathcal{S}$. $Q \in PL^{\{.,\_,\_^*\}}$ is *valid* if it does not have labels $\ell \in \mathbf{A}$ or $\ell = S$

in a position other than the last one. Let P, Q be words from $PL^{\{.,-,-^*\}}$. P is *contained* in Q, denoted by $P \subseteq Q$, if for every XML tree T and every node v of T we have $v[\![P]\!] \subseteq v[\![Q]\!]$.

If a node u lies on the path from a node v to the root, then u is an *ancestor* of v, and v a *descendent* of u. An *independent set* J of an XML tree T is a set of pairwise incomparable nodes, i.e., no node in J is an ancestor of any other node in J. Every path from a leaf to the root is a *branch* of T. An independent set intersects a branch at most once. For a node u of T, a u*-independent set* J of T is a set of descendents of u such that each pair of distinct nodes in J has u as their lowest common ancestor. Clearly, u-independent sets are independent.

Two nodes u, v are *value equal*, denoted by $u =_v v$, if the subtrees rooted at u and v are isomorphic by an isomorphism that preserves string values. For nodes v and v' of an XML tree T, the *value intersection* of $v[\![Q]\!]$ and $v'[\![Q]\!]$ is given by $v[\![Q]\!] \cap_v v'[\![Q]\!] = \{(w, w') \mid w \in v[\![Q]\!], w' \in v'[\![Q]\!], w =_v w'\}$.

3 Soft Cardinality Constraints

Now we define a highly expressive class of soft cardinality constraints. The first source of expressivity comes from the ability to specify soft upper bounds (soft-max) as well as soft lower bounds (soft-min) on the number of nodes (target nodes) in an XML tree that are value-equal on some of its subnodes (field nodes). These soft bounds can be violated by some individual nodes, but they should be respected in average. There is also the possibility of specifying the soft bounds w.r.t. a context node. The second source of expressivity results from the generality of the path language $PL^{\{.,-,-^*\}}$ used for the selection of nodes. The final source of expressivity is due to the use of the robust notion of value-equality defined in the previous section, which is *not* restricted to leaf or attribute nodes.

Definition 1. *We define a* soft cardinality constraint φ *for XML as an expression of the form soft-card*$(Q, (Q', \{Q_1, \ldots, Q_k\})) = ($*soft-min*, *soft-max*$)$ *where* k *is a non-negative integer, where* $Q, Q', Q_1, \ldots, Q_k \in PL^{\{.,-,-^*\}}$ *such that* $Q.Q'$, $Q.Q'.Q_1, \ldots, Q.Q'.Q_k$ *are valid, and where soft-min* $\in \mathbb{N}$ *and soft-max* $\in \mathbb{N} \cup \{\infty\}$ *with soft-min* $\leq$ *soft-max. Herein,* Q *is called the* context path*,* Q' *is called the* target path*,* $Q_1, \ldots, Q_k$ *are called the* field paths*, soft-min is called the* soft lower bound*, and soft-max the* soft upper bound *of* φ*. If* $Q = \varepsilon$*, we call* φ absolute*; otherwise* φ *is called* relative*.*

In the sequel, for a soft cardinality constraint φ, we denote its context path as Q_φ, its target path as Q'_φ, its field paths as $Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}$ and its soft lower and upper bounds as soft-min$_\varphi$ and soft-max$_\varphi$, respectively.

Definition 2. *Consider a soft cardinality constraint* φ*, an XML tree* T*, a context node* $q \in [\![Q_\varphi]\!]$ *and a target node* $q'_0 \in q[\![Q'_\varphi]\!]$*. We set* $f^\varphi_T(q, q'_0)$ *as the maximum of* $|\{q' \in q[\![Q'_\varphi]\!] \mid \exists y_1, \ldots, y_k. \forall i = 1, \ldots, k.\ y_i \in q'[\![Q^\varphi_i]\!] \wedge x_i =_v y_i\}$ *where* $x_1, \ldots, x_k$ *ranges through all* $x_i \in q'_0[\![Q^\varphi_i]\!]$ *(with* $i = 1, \ldots, k$*). That is,* $f^\varphi_T(q, q'_0)$ *is the maximum number of target nodes* q' *in the sub-tree of* T *rooted* *at the context node q that share with q'_0 the same information on their field paths. We say that T satisfies φ as a* soft cardinality constraint *if*

$$\textit{soft-min}_\varphi \leq \frac{1}{|U|} \sum_{q'_0 \in U} f_T^\varphi(q, q'_0) \leq \textit{soft-max}_\varphi$$

holds for every context node $q \in [\![Q_\varphi]\!]$ and every maximal q-independent set $U \subseteq q[\![Q'_\varphi]\!]$. If there is no target node $q'_0 \in q[\![Q'_\varphi]\!]$ in T for which for all $i = 1, \ldots, k_\varphi$, field nodes $x_i \in q'_0[\![Q_i^\varphi]\!]$ exists in T, then T satisfies the soft cardinality constraint φ by default since it does not apply to T.

Example 4. Following the discussion in Examples 2 and 3 above, the following expressions formalise the cardinality constraints in Example 1 when interpreted as soft cardinality constraints over trees of the form illustrated in Figure 1.

1. soft-card$(\varepsilon, (_.RTeam.Sci, \{id\})) = (2, 4)$ or equivalently soft-card$(\varepsilon, (_^*.RTeam.Sci, \{id\})) = (2, 4)$.
2. soft-card$(\varepsilon, (_.STeam.Tech, \{id\})) = (1, 4)$ or equivalently soft-card$(\varepsilon, (_^*.STeam.Tech, \{id\})) = (1, 4)$
3. soft-card$(\varepsilon, (_.STeam.Eng, \{id\})) = (4, 4)$ and soft-card$(\varepsilon, (_.RTeam.Eng, \{id\})) = (1, 2)$.
4. soft-card$(_, (Manager, \emptyset)) = (1, 1)$.
5. soft-card$(\varepsilon, (_.Steam, \{Sid\})) = (1, 3)$.
6. soft-card$(\varepsilon, (_.Rteam, \{_^*.Rid\})) = (1, 2)$.
7. soft-card$(_, (STeam, \{_^*.Sid\})) = (1, 2)$ and soft-card$(_, (RTeam, \{Rid\})) = (1, 2)$.
8. soft-card$(_._, (_, \{_^*.S\})) = (2, 2)$ or equivalently soft-card$(_._, (_, \{Expertise.S\})) = (2, 2)$.
9. soft-card$(_, (_._, \{id\})) = (1, 1)$.
10. soft-card$(_, (_._, \{Expertise.S\})) = (1, 5)$.

Note that the soft cardinality constraints in point 1–3, 5 and 6 are absolute while the soft cardinality constraints in the remaining points are relative. □

Let $\Sigma \cup \{\varphi\}$ be a finite set of (soft) constraints in a class $\mathcal{S}$. We say that Σ *finitely implies* φ, denoted by $\Sigma \models \varphi$, if every finite XML tree T that satisfies all $\sigma \in \Sigma$ also satisfies φ. The *finite implication problem* for the class $\mathcal{S}$ is to decide whether $\Sigma \models \varphi$. By Σ^* we denote the *(finite) semantic closure* of Σ, i.e., the set of all (soft) constraints finitely implied by Σ.

If we want to take advantage of the proposed soft cardinality constraints in real-world XML applications, then we must be able to reason about them efficiently. Central to this task is the finite implication problem describe above. Unfortunately the implication problem for the general class of soft cardinality constraints introduced in Definition 1, is likely intractable. In fact, as stated in the next theorem, there are at least three different sources of intractability:

i. the simultaneous use of both soft lower and soft upper bounds (as permitted in $\mathcal{S}_1$ in Theorem 1),

ii. the complete absence of field paths (as permitted in $\mathcal{S}_2$ in Theorem 1), and *iii.* the simultaneous use of arbitrary length wildcards in both target and field paths (as permitted in $\mathcal{S}_3$ in Theorem 1).

Theorem 1. *The finite implication problem for each of the following fragments of soft cardinality constraints is coNP-hard.*

$$\begin{aligned}\mathcal{S}_1 =&\{\textit{soft-card}(\varepsilon, (P', \{P_1, \ldots, P_k\})) = (\textit{soft-min}, \textit{soft-max})\\ &\mid P', P_1, \ldots, P_k \in PL^{\{.\}}, k \geq 1, \textit{soft-max} \leq 5\},\\ \mathcal{S}_2 =&\{\textit{soft-card}(\varepsilon, (P', \{P_1, \ldots, P_k\})) = (1, \textit{soft-max})\\ &\mid P', P_1, \ldots, P_k \in PL^{\{.\}}, k \geq 0, \textit{soft-max} \leq 6\},\\ \mathcal{S}_3 =&\{\textit{soft-card}(\varepsilon, (Q', \{Q_1, \ldots, Q_k\})) = (1, \textit{soft-max})\\ &\mid Q', Q_1, \ldots, Q_k \in PL^{\{.,\_^*\}}, k \geq 1, \textit{soft-max} \leq 4\}.\end{aligned}$$

We note that for each of the classes considered in Theorem 1, the 3-colorability problem over graphs can be polynomially transformed to the complement of the implication problem for soft cardinality constraints, but due to space limitations we omit the formal proof.

To avoid the sources of intractability pointed out in Theorem 1, we will consider a fragment of soft cardinality constraints that provides an optimal balance with respect to expressivity and efficiency.

Definition 3. *We define the fragment* $\mathfrak{M}^{\text{soft}}$ *of* soft-max cardinality constraints *as follows.* $\mathfrak{M}^{\text{soft}} = \{\textit{soft-card}(Q, (Q', \{Q_1, \ldots, Q_k\})) = (1, \textit{soft-max}) \mid Q, Q', Q_1, \ldots, Q_k \in PL^{\{.,\_,\_^*\}}$ *but s.t.* Q' *or* $Q_1. \cdots .Q_k \in PL^{\{.,\_\}}\}$*. Since soft-min is always set to* 1*, we use the abbreviation* $\textit{soft-card}(Q, (Q', \{Q_1, \ldots, Q_k\})) \leq \textit{soft-max}$ *to denote the soft constraints in* $\mathfrak{M}^{\text{soft}}$*.*

The fragment of soft-max cardinality constraints is still expressive, allowing for instance to express most of the cardinality constraints in Example 1.

Example 5. The soft cardinality constraints in points 2, 4–7, 9 and 10 in Example 4 belong to $\mathfrak{M}^{\text{soft}}$. Also the second soft cardinality constraint in point 3 belongs to $\mathfrak{M}^{\text{soft}}$. The remaining three soft cardinality constraints can still be partially expressed as soft-max cardinality constraints if we change the lower bound soft-min to 1. $\square$

Note that by Theorem 1, any increase in expressivity of the fragment of soft-max constraints results in coNP-hardness of the implication problem.

4 Axiomatization

Table 1 shows a set of inference rules which constitutes a finite axiomatization for the implication of soft-max cardinality constraints. Each inference rule has the form $\frac{\textit{premises}}{\textit{conclusion}}\textit{condition}$ with premises from $\mathfrak{M}^{\text{soft}}$. That is, the path expressions used in the premises are always chosen such that the respective soft cardinality constraint lies in $\mathfrak{M}^{\text{soft}}$.

$$\frac{}{\text{soft-card}(Q,(Q',S)) \le \infty}\; {\scriptstyle Q' \in PL^{\{.,\_\}} \text{ or } \emptyset \neq S \subseteq PL^{\{.,\_\}}}$$
(infinity)

$$\frac{}{\text{soft-card}(Q,(\epsilon,S)) \le 1}$$
(epsilon)

$$\frac{\text{soft-card}(Q,(Q'.Q'',S)) \le \text{soft-max}}{\text{soft-card}(Q.Q',(Q'',S)) \le \text{soft-max}}$$
(target-to-context)

$$\frac{\text{soft-card}(Q,(Q',S)) \le \text{soft-max}}{\text{soft-card}(Q,(Q',S)) \le \text{soft-max}+1}$$
(weakening)

$$\frac{\text{soft-card}(Q,(Q',S \cup \{\epsilon,P\})) \le \text{soft-max}}{\text{soft-card}(Q,(Q',S \cup \{\epsilon,P.P'\})) \le \text{soft-max}}$$
(prefix-epsilon)

$$\frac{\text{soft-card}(Q,(Q',S)) \le \text{soft-max}}{\text{soft-card}(Q,(Q',S \cup \{P\})) \le \text{soft-max}}\; {\scriptstyle Q' \text{ or } P \in PL^{\{.,\_\}}}$$
(superfield)

$$\frac{\text{soft-card}(Q,(Q'.P,\{P'\})) \le \text{soft-max}}{\text{soft-card}(Q,(Q',\{P.P'\})) \le \text{soft-max}}\; {\scriptstyle \text{at least 2 of } Q',P,P' \in PL^{\{.,\_\}}}$$
(subnodes)

$$\frac{\text{soft-card}(Q,(Q',S)) \le \text{soft-max}}{\text{soft-card}(Q'',(Q',S)) \le \text{soft-max}}\; {\scriptstyle Q'' \subseteq Q}$$
(context-path-containment)

$$\frac{\text{soft-card}(Q,(Q'.P,\{\epsilon,P'\})) \le \text{soft-max}}{\text{soft-card}(Q,(Q',\{\epsilon,P.P'\})) \le \text{soft-max}}\; {\scriptstyle \text{at least 2 of } Q',P,P' \in PL^{\{.,\_\}}}$$
(subnodes-epsilon)

$$\frac{\text{soft-card}(Q,(Q',S)) \le \text{soft-max}}{\text{soft-card}(Q,(Q'',S)) \le \text{soft-max}}\; {\scriptstyle Q'' \subseteq Q'}$$
(target-path-containment)

$$\frac{\begin{array}{c}\text{soft-card}(Q,(Q',\{P.P_1,\dots,P.P_k\})) \le \text{soft-max},\\ \text{soft-card}(Q.Q',(P,\{P_1,\dots,P_k\})) \le \text{soft-max}'\end{array}}{\text{soft-card}(Q,(Q'.P,\{P_1,\dots,P_k\})) \le \text{soft-max}\cdot\text{soft-max}'}$$
(multiplication)

$$\frac{\text{soft-card}(Q,(Q',S \cup \{P\})) \le \text{soft-max}}{\text{soft-card}(Q,(Q',S \cup \{P'\})) \le \text{soft-max}}\; {\scriptstyle P' \subseteq P}$$
(field-path-containment)

Table 1. A finite axiomatization for soft cardinality constraints in $\mathfrak{M}^{\text{soft}}$.

Example 6. Let us define the soft-max cardinality constraints $\sigma_1 = \text{soft-card}(\_, (RTeam, \{Eng.\_^*.S\})) \le 2$ and $\sigma_2 = \text{soft-card}(\_.\_, (Eng, \{\_^*.S\})) \le 2$, which are applicable to XML documents that are structured in the way schematised by the tree in Figure 1. The soft constraint σ_1 states that in a given project, it is rare that more than two research teams have engineers of a same expertise level. The soft constraint σ_2 states that it is unusual that there is more than two engineers of a same expertise level within a given team. By applying the *context-path-containment* rule to σ_2 we derive $\sigma_3 = \text{soft-card}(\_.RTeam, (Eng, \{\_^*.S\})) \le 2$. Then, by applying the *multiplication* rule to σ_1 and σ_3 we derive $\varphi = \text{soft-card}(\_, (RTeam.Eng, \{\_^*.S\})) \le 4$, which expresses that it is infrequent to find more than 4 engineers of a same expertise level if we look at all the engineers in all the research teams involved in a given project. □

We omit the tedious, but not very difficult proof of the soundness of the inference rules. Our next goal is to demonstrate that the set $\mathfrak{R}$ of inference rules in Table 1 is complete for the implication of soft-max constraints in the class $\mathfrak{M}^{\text{soft}}$. Completeness means we need to show that for an arbitrary finite set $\Sigma \cup \{\varphi\}$ of soft-max constraints in the class $\mathfrak{M}^{\text{soft}}$, if φ is not derivable from Σ by $\mathfrak{R}$, then there is some XML tree T that satisfies all members of Σ but violates φ. That is, T is a counter-example tree for the implication of φ by Σ.

In a first step, we represent φ as a finite node-labeled tree $T_{\Sigma,\varphi}$, which we call the *φ-tree*.

Definition 4. *(φ-**tree**). Let $\Sigma \cup \{\varphi\}$ be a finite set of soft-max constraints in the class $\mathfrak{M}^{\text{soft}}$. Let $\mathcal{L}_{\Sigma,\varphi}$ denote the set of all labels $\ell \in \mathcal{L}$ that occur in path expressions of members in $\Sigma \cup \{\varphi\}$, and fix a label $\ell_0 \in \boldsymbol{E} - \mathcal{L}_{\Sigma,\varphi}$. First we* *transform the path expressions occurring in φ into simple path expressions in $PL^{\{.\}}$. For that purpose we replace each single-label wildcard "_" by ℓ_0 and each variable-length wildcard "_*" by a sequence of $l+1$ labels ℓ_0, where l is the maximum number of consecutive single-label wildcards that occurs in any soft constraint in $\Sigma \cup \{\varphi\}$. This transformation turns Q_φ into O_φ, Q'_φ into O'_φ, and each Q_i^φ into O_1^φ for $i = 1, \ldots, k_\varphi$. The path expressions after the transformation do not contain any more wildcards (neither single-label nor variable-length ones). Let p be an O_φ-path from a node r_φ to a node q_φ, let p' be an O'_φ-path from a node r'_φ to a node q'_φ and, for $i = 1, \ldots, k_\varphi$, let p_i be a O_i^φ-path from a node r_i^φ to a node x_i^φ, such that the paths $p, p', p_1, \ldots, p_{k_\varphi}$ are mutually node-disjoint. From the paths $p, p', p_1, \ldots, p_{k_\varphi}$ we obtain the* φ-tree $T_{\Sigma,\varphi}$ *by identifying the node r'_φ with q_φ, and by identifying each of the nodes r_i^φ with q'_φ.*

The *marking* of the φ-tree $T_{\Sigma,\varphi}$ is a subset $\mathcal{M}$ of the node set of $T_{\Sigma,\varphi}$: if for all $i = 1, \ldots, k_\varphi$ we have $Q_i^\varphi \neq \varepsilon$, then $\mathcal{M}$ consists of the leaves of $T_{\Sigma,\varphi}$, and otherwise $\mathcal{K}$ consists of all descendant nodes of q'_φ in $T_{\Sigma,\varphi}$.

We use φ-trees to calculate the impact of soft-max constraints in Σ on a possible counter-example tree T for the implication of φ by Σ. To distinguish soft-max constraints that have an impact from those that do not, we introduce the notion of *applicability*. Intuitively, when a soft-max constraint is not applicable, then we do not need to satisfy its soft upper bound in a counter-example tree as it does not require all its field paths.

Definition 5. *(***Applicability***). Consider a φ-tree $T_{\Sigma,\varphi}$, and let $\mathcal{M}$ be its marking. A soft-max constraint σ is said to be* applicable *to φ if there are nodes $w_\sigma \in [\![Q_\sigma]\!]$ and $w'_\sigma \in w_\sigma[\![Q'_\sigma]\!]$ in $T_{\Sigma,\varphi}$ such that $w'_\sigma[\![P_i^\sigma]\!] \cap \mathcal{M} \neq \emptyset$ for all $i = 1, \ldots, k_\sigma$. We say that w_σ and w'_σ* witness *the applicability of σ to φ.*

Then, we reverse the edges of the φ-tree and add to the resulting tree downward edges for the applicable members of Σ. Finally, each upward edge receives a label of 1 and each downward edge resulting from $\sigma \in \Sigma$ a label of soft-max$_\sigma$. This final directed graph $G_{\Sigma,\varphi}$ is called the *cardinality network*. A downward edge resulting from σ tells us that under each source node there can be at most soft-max$_\sigma$ target nodes.

Definition 6. *(***Cardinality Network***). We define the* cardinality network *$G_{\Sigma,\varphi}$ of φ and Σ as the node-labeled directed graph obtained from $T_{\Sigma,\varphi}$ as follows: the nodes and node-labels of $G_{\Sigma,\varphi}$ are exactly the nodes and node-labels of $T_{\Sigma,\varphi}$, respectively. The edges of $G_{\Sigma,\varphi}$consist of the reversed edges from $T_{\Sigma,\varphi}$. Furthermore, for each soft-max constraint $\sigma \in \Sigma$ that is applicable to φ and for each pair of nodes $w_\sigma \in [\![Q_\sigma]\!]$ and $w'_\sigma \in w_\sigma[\![Q'_\sigma]\!]$ that witness the applicability of σ to φ we add a directed edge (w_σ, w'_σ) to $G_{\Sigma,\varphi}$. We refer to these additional edges as* witness edges *while the reversed edges from $T_{\Sigma,\varphi}$ are referred to as* upward edges *of $G_{\Sigma,\varphi}$. This is the case since for every witness w_σ and w'_σ the node w'_σ is a descendant of the node w_σ in $T_{\Sigma,\varphi}$, and thus the witness edge (w_σ, w'_σ) is a downward edge or loop in $G_{\Sigma,\varphi}$. We now introduce weights as edge-labels: every upward edge e of $G_{\Sigma,\varphi}$ has weight $\omega(e) = 1$, and every witness edge (u, v) of $G_{\Sigma,\varphi}$ has weight $\omega(u, v) = \min\{$soft-max$_\sigma \mid (u, v)$ witnesses the applicability of some $\sigma \in \Sigma$ to $\varphi\}$.*

The *weight* of a path t in the cardinality network is defined as the product of the weights of its edges, i.e., $\omega(t) = \prod_{i=1}^{n} \omega(v_{i-1}, v_i)$, or $\omega(t) = 1$ if t has no edges. The *distance* $d(v, w)$ from a node v to a node w is the minimum over the weights of all paths from v to w, or ∞ if no such path exists. When the target node q'_φ of constraint φ can be reached from its context node q_φ along a path of weight at most soft-max$_\varphi$ in the cardinality network $G_{\Sigma,\varphi}$ then there exists no counter-example tree T.

Example 7. Figure 2 shows the cardinality network $G_{\Sigma,\varphi}$ obtained for $\Sigma = \{\sigma_1, \sigma_2\}$ and φ, where σ_1, σ_2, and φ are the soft-max constraints used in Example 6. Note that the distance $d(q_\varphi, q'_\varphi) = 4$ and soft-max$_\varphi = 4$. Thus, there is no counter-example tree T, which is correct since φ is indeed implied by Σ. $\square$

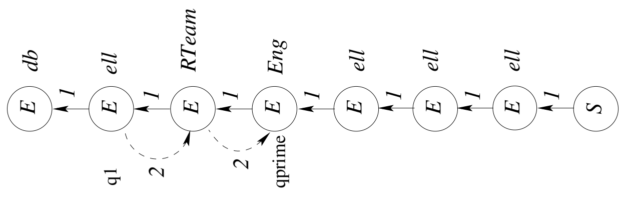

Fig. 2. Cardinality Network.

The result below prove the following crucial observation. If φ is not derivable from Σ by $\mathfrak{R}$, then every path from q_φ to q'_φ in $G_{\Sigma,\varphi}$ has distance at least soft-max$_\varphi + 1$.

Lemma 1. *Let $\Sigma \cup \{\varphi\}$ be a finite set of soft-max cardinality constraints in the class $\mathfrak{M}^{\text{soft}}$. If the distance $d(q_\varphi, q'_\varphi) \leq$ soft-max$_\varphi$ in the cardinality network $G_{\Sigma,\varphi}$, then φ is derivable from Σ by $\mathfrak{R}$.*

The strategy to prove this lemma is to encode an inference by $\mathfrak{R}$ by witness edges of the cardinality network. We omit this proof as it is technical and lengthy.

If $\Sigma \cup \{\varphi\}$ is a finite set of soft-max constraints in the class $\mathfrak{M}^{\text{soft}}$ such that φ is not derivable from Σ by $\mathfrak{R}$, then the previous lemma allows us to construct a finite XML tree T which satisfies all soft-max constraints in Σ but does not satisfy φ. This fact proves the following important result.

Theorem 2. *The inference rules in Table 1 are complete for the implication of soft-max constraints in $\mathfrak{M}^{\text{soft}}$.*

5 An Algorithm for Deciding Implication

Our Algorithm 1 for deciding the implication of soft-max constraints is similar to the corresponding algorithms in [11, 6] for deciding the implication of the strictly less expressive classes of numerical keys and (strict) cardinality constraints, respectively. However, the construction of the cardinality network $G_{\Sigma,\varphi}$, which is central to the algorithms, requires considerably more effort for (strict) cardinality constraints as studied in [6] as well as for the soft-max cardinality constraints as studied here. This effort results in an increase in the worst-case time complexity of the algorithm compared to numerical keys. Nevertheless, Algorithm 1 enables us to conclude that the implication of soft cardinality constraints in $\mathfrak{M}^{\text{soft}}$ can be decided in low-degree polynomial time in the worst case.

The correctness of Algorithm 1 is due to the fact that for $\Sigma \cup \{\varphi\}$ a finite set of soft-max cardinality constraints in $\mathfrak{M}^{\text{soft}}$, $\Sigma \models \varphi$ holds if and only if $d(q_\varphi, q'_\varphi) \leq \text{soft-max}_\varphi$ in the cardinality network $G_{\Sigma,\varphi}$. This fact can be easily proved from the results in the previous section.

Algorithm 1 Soft-max constraint implication.

```
Input: a finite set $\Sigma \cup \{\varphi\}$ of soft-max cardinality constraints in $\mathfrak{M}^{\text{soft}}$
Output: yes, if $\Sigma \models \varphi$; no, otherwise
 1: Construct $G_{\Sigma,\varphi}$ for $\Sigma$ and $\varphi$;
 2: Find the shortest path $P$ from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$;
 3: if $\omega(P) \leq \text{soft-max}_\varphi$ then return(yes); else return(no).
```

Interestingly, the algorithm has the same worst-case complexity as the algorithm for the class of cardinality constraints in [6], which is clearly less expressive since it does not allow variable length wildcards to appear in the field paths, and does not cater for soft bounds. In fact, if $\Sigma \cup \{\varphi\}$ is a finite set of soft-max cardinality constraints in $\mathfrak{M}^{\text{soft}}$, then the implication problem $\Sigma \models \varphi$ can be decided in time $\mathcal{O}(|\varphi| \times l \times (||\Sigma|| + |\varphi| \times l))$, where $|\varphi|$ is the sum of the lengths of all path expressions in φ, $||\Sigma||$ is the sum of all sizes $|\sigma|$ for $\sigma \in \Sigma$, and l is the maximum number of consecutive single-label wildcards that occur in Σ.

It is important to note the blow-up in the size of the counter-example with respect to φ. This is due to the occurrence of consecutive single-label wildcards. If the number l is fixed in advance, then Algorithm 1 establishes a worst-case time complexity that is quadratic in the input. In particular, if the input consists of (numerical) keys, as studied in [10, 11], then the worst-case time complexity of Algorithm 1 is that of the algorithm dedicated to (numerical) keys only [10].

6 Experimental Evaluation

We have amply tested our decision algorithm and analysed its performance. We compare the performance against the implementation presented in [7] which is optimised for deciding implication of the strictly less expressive class of XML keys from [10]. The performance results were obtained in a fairly modest Intel Core i7 2.8 GHz machine, with 4 GB of RAM, running a Linux kernel 2.6.32. We compiled our C++ implementation of the algorithms using the standard g++ compiler from the GNU Compiler Collection 4.6.3.

Test Cases. To generate realistic sets of soft constraints to test our algorithm, we generated soft-max constraints applicable to large XML documents from [15]: *321gone.xml* and *yahoo.xml* (auction data), *dblp.xml* (bibliographic information on CS), *nasa.xml* (astronomical data), *SigmodRecord.xml* (articles from SIGMOD Record), and *mondial-3.0.xml* (world geographic db).

We started by writing, for each document in the collection, a corresponding set of around 10 appropriate (in the context of the document) soft-max cardinality constraints. On adapting the strategy from [7], we generated large sets of soft-max cardinality constraints as follows. Firstly, using the manually defined sets of soft-max constraints as seeds, we computed new implied soft-max constraints by successively applying the inference rules from the axiomatization of soft-max cardinality constraints shown in Table 1. Each constraint generated by this method was added to the original set. We applied the *multiplication*, *target-to-context*, *prefix-epsilon*, *subnodes*, *subnodes-epsilon*, *superfield*, *context-path containment*, *target-path containment*, and *field-path-containment* rules whenever possible, since those are the rules which can produce implied soft-max cardinality constraints with corresponding non trivial cardinality networks. Secondly, we defined some non-implied (by the soft constraint defined previously) soft-max cardinality constraints. We did that by taking non-implied soft cardinality constraints φ, building their corresponding cardinality networks $G_{\Sigma,\varphi}$, adding several witness edges to them while keeping the weights $\omega(P) >$ soft-max$_\varphi$, for P the shortest path from q_φ to q'_φ, and finally defining new non-implied soft-max cardinality constraints corresponding to those witness edges. This process gave us a robust collection of soft-max cardinality constraints to thoroughly test the performance of the implication algorithm[6].

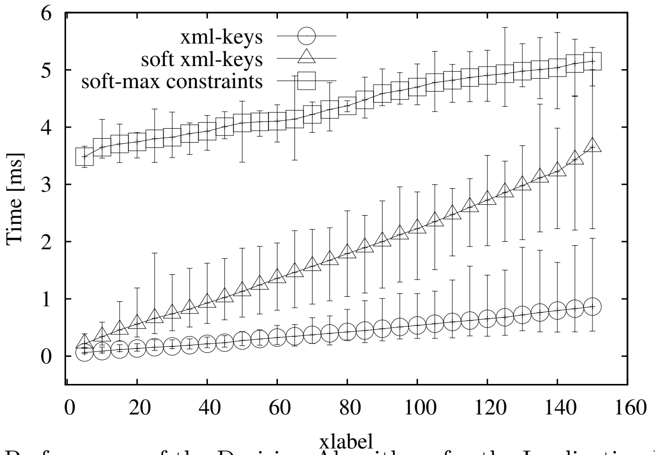

Fig. 3. Performance of the Decision Algorithms for the Implication Problem.

Tests Results. To have a base line for determining how much the increase in expressivity of the considered class of constraints affects the performance of the decision algorithm, we first measured the performance of the algorithm in [7] which is optimised for deciding implication of the strictly less expressive class of

[6] All sets of constraints generated to test the performance of the decision algorithms as well as the full set of results from those experiments and the binary codes, can be downloaded from `http://emir-munoz.github.com/xml-constraints`

XML keys from [10]. For this we used the same sets of XML keys with simple non empty field paths in $PL^{\{\cdot\}}$ and context and target paths in $PL^{\{\cdot,\_,{}^*\}}$ than in [7]. We then ran our algorithm for soft-max constraints using the same sets of XML keys but treating them as soft-max cardinality constraints with soft-max = 1 (i.e., "soft keys"). This allowed us to quantify the gain produced by the optimization of the algorithm for XML keys presented in [7] and produced a base line to measure the effect of introducing single label wildcards in the soft-max cardinality constraints. Finally we measured the performance of the algorithm for soft-max cardinality constraints over the set of full soft-max constraints obtained with the process described at the beginning of this section.

The results are shown in Figure 3. The x-axis corresponds to the number of (soft) constraints in Σ, and the y-axis corresponds to the *average* running time required to decide whether Σ implies a given (soft) constraint φ. More precisely, let $time(\Sigma, \varphi)$ be the running time required to decide $\Sigma \models \varphi$ and let Φ be a set of (soft) constraints such that $\Sigma \cap \Phi = \emptyset$, the running time shown in Figure 3 corresponds to $\big(\sum_{\varphi_i \in \Phi} time(\Sigma, \varphi_i)\big)/|\Phi|$. In our experiments the sets Φ were composed of 20 fixed (soft) constraints. We tested the scalability of the algorithms by adding, in each iteration, 5 new (soft) constraints to the corresponding Σ sets. Each of the experiments was executed 5 times. The resulting error bars are include in the graph. They are consistent with time variations commonly produced by the scheduling of the operating system and the use of the *time*() function to measure the experiments [14].

From the experiments it is clear that the implication algorithm is practically efficient and scales well in all three cases. Notably, the extra price to pay for the added expressivity provided by the class of soft-max cardinality constraints is in the order of just 5 milliseconds for a considerable big set of 150 constraints.

7 Conclusion

We have introduced an expressive class of *soft* cardinality constraints that is sufficiently flexible to advance XML data processing in important areas of XML application such as data exchange and integration, where exceptions to strict rules are the common norm and are therefore difficult to handle by strict constraints. The flexibility results from the right balance between expressivity and efficiency of maintenance. While slight extensions result in the intractability of the associated implication problem, we have shown that our class is finitely axiomatizable, robust and decidable in low-degree polynomial time. Thus, our class forms a precious class of soft cardinality constraints that can be utilised effectively by data engineers. Indeed, the performance tests presented in this paper for its associated implication problem, clearly indicate that it can be maintained efficiently by database systems for XML applications.

Future work can go into various directions. XML practice may well warrant the study of other soft classes of cardinality constraints that require different paradigms to specify soft bounds and to select and compare nodes. It would be interesting to investigate soft cardinality constraints with regard to data cleaning, where one of the most important questions is how to model the consistency of the data; for instance exploring conditional XML constraints in connection with the idea of conditional functional dependencies [3]. Finally, it would also be interesting to explore practical applications of the decision algorithm for the implication problem in areas such as cardinality estimation and optimization of XPath queries, XML constraint mining and validation of XML documents.

References

1. Arenas, M., Fan, W., Libkin, L.: What's Hard about XML Schema Constraints? In: DEXA. LNCS, vol. 2453, pp. 269–278. Springer (2002)
2. Arenas, M., Fan, W., Libkin, L.: On the Complexity of Verifying Consistency of XML Specifications. SIAM J. Comput. 38(3), 841–880 (2008)
3. Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional Functional Dependencies for Data Cleaning. In: ICDE. pp. 746–755. IEEE (2007)
4. Brown, K.: Soft consistencies for weighted csps. In: In Proceedings of Soft'03: 5th International Workshop on Soft Constraints. Kinsale, Ireland (2003)
5. Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., Tan, W.C.: Keys for XML. Computer Networks 39(5), 473–487 (2002)
6. Ferrarotti, F., Hartmann, S., Link, S.: A Precious Class of Cardinality Constraints for Flexible XML Data Processing. In: ER. LNCS, vol. 6998, pp. 175–188. Springer (2011)
7. Ferrarotti, F., Hartmann, S., Link, S., Marín, M., Muñoz, E.: Performance Analysis of Algorithms to Reason about XML Keys. In: DEXA. LNCS, vol. 7446, pp. 101–115. Springer (2012)
8. Franceschet, M., Gubiani, D., Montanari, A., Piazza, C.: From Entity Relationship to XML Schema: A Graph-Theoretic Approach. In: XSym. LNCS, vol. 5679, pp. 165–179. Springer (2009)
9. Hartmann, S.: Soft Constraints and Heuristic Constraint Correction in Entity-Relationship Modelling. In: Semantics in Databases. LNCS, vol. 2582, pp. 82–99. Springer (2001)
10. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
11. Hartmann, S., Link, S.: Numerical constraints on XML data. Inf. Comput. 208(5), 521–544 (2010)
12. Meyer, J.J.C., Wieringa, R., Dignum, F.: The Role of Deontic Logic in the Specification of Information Systems. In: Logics for Databases and Information Systems. pp. 71–115. Kluwer (1998)
13. Preece, A.D., Chalmers, S., McKenzie, C., Pan, J.Z., Gray, P.M.D.: A semantic web approach to handling soft constraints in virtual organisations. Electronic Commerce Research and Applications 7(3), 264–273 (2008)
14. Stewart, D.B., Khosla, P.K.: Mechanisms for Detecting and Handling Timing Errors. Commun. ACM 40(1), 87–93 (1997)
15. Suciu, D.: XML Data Repository, University of Washington. `http://www.cs.washington.edu/research/xmldatasets/www/repository.html` (2002)
16. Thalheim, B.: Integrity Constraints in (Conceptual) Database Models. In: The Evolution of Conceptual Modeling. LNCS, vol. 6520, pp. 42–67. Springer (2008)
17. Yu, C., Jagadish, H.V.: XML schema refinement through redundancy detection and normalization. VLDB J. 17(2), 203–223 (2008)