Efficiency Frontiers of XML Cardinality Constraints

Flavio Ferrarotti
School of Information Management,
The Victoria University of Wellington,
Wellington, New Zealand

Sven Hartmann
Department of Informatics,
Clausthal University of Technology,
Clausthal-Zellerfeld, Germany

Sebastian Link
Department of Computer Science,
University of Auckland,
Auckland, New Zealand

CDMTCS-427
October 2012

Centre for Discrete Mathematics and
Theoretical Computer Science

Efficiency Frontiers of XML Cardinality Constraints

Flavio Ferrarotti
School of Information Management
The Victoria University of Wellington
Wellington, New Zealand
flavio.ferrarotti@vuw.ac.nz

Sven Hartmann
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
sven.hartmann@tu-clausthal.de

Sebastian Link
Department of Computer Science
The University of Auckland, Private Bag 92019
Auckland, New Zealand
s.link@auckland.ac.nz

October 10, 2012

Abstract

XML has gained widespread acceptance as a premier format for publishing, sharing and manipulating data through the web. While the semi-structured nature of XML provides a high degree of syntactic flexibility there are significant shortcomings when it comes to specifying the semantics of XML data. For the advancement of XML applications it is therefore a major challenge to discover natural classes of constraints that can be utilized effectively by XML data engineers. This endeavor is ambitious given the multitude of intractability results that have been established. We investigate a class of XML cardinality constraints that is precious in the sense that it keeps the right balance between expressiveness and efficiency of maintenance. In particular, we characterize the associated implication problem axiomatically and develop a low-degree polynomial time algorithm that can be readily applied for deciding implication. Our class of constraints is chosen near-optimal as already minor extensions of its expressiveness cause potential intractability. Finally, we transfer our findings to establish a precious class of soft cardinality constraints on XML data. Soft cardinality constraints need to be satisfied on average only, and thus permit violations in a controlled manner. Soft constraints are therefore able to tolerate exceptions that frequently occur in practice, yet can be reasoned about efficiently.

Keywords: Semi-structured Data and XML, Database semantics, Database constraint, Cardinality constraint, Cardinality estimation

1 Introduction

The unprecedented success of web-based applications has led to a tremendous growth in the data volumes stored and shared on the Web. A major driver of this success were the efforts of the World Wide Web consortium to establish the Extensible Markup Language (XML) as a commonly accepted standard for encoding and retrieving such data. In turn, the XML family of standards has emerged as a key enabling technology for many promising trends in software development, deployment and use, including web services and cloud computing. As web-based applications become ever more data-intensive, qualified support for managing XML data on the Web is in high demand. Consequently, data management support for XML has attracted a huge amount of research over the last decade. Despite the recent incorporation of native XML support into all major database management systems, current database management practice for XML is not yet mature. One of the areas that have been identified as being essential for the advancement of database management techniques for XML are database constraints. For relational databases we have experienced that many recurring tasks in database development, administration, and utilization can greatly benefit from a better understanding of the semantics of the managed data.

Database constraints are used to restrict the instances of a database to those considered meaningful to the application domain. Enriching XML by means to specify relevant database constraints is therefore a worthwhile endeavor of the database community, but faces a major obstacle: XML is often appraised for its syntactical flexibility that allows data engineers to represent highly complex and/or heterogenous application data. This advantage, however, causes the treatment of database constraints for XML to be more complicated than for relational data. A goal must be to identify classes of database constraints that can be readily exploited during database development, administration, and utilization. The delicate interdependencies among XML data items often mean that constraint classes cannot be treated effectively, let alone efficiently. A multitude of infeasibility and intractability results exists, see [4, 42]. For example, the satisfiability problem of keys and foreign keys, as defined by XML Schema, is undecidable, while that of keys alone is still *NP*-hard [3]. Hence, the important challenge is to find classes that are precious, that is, both expressive *and* tractable.

Many areas of database practice have motivated the study of particular kinds of database constraints. Cardinality constraints are a very natural class of constraints that can be observed easily and have many potential applications. Generally speaking, cardinality constraints capture information about the frequency with which certain data items occur in particular contexts.

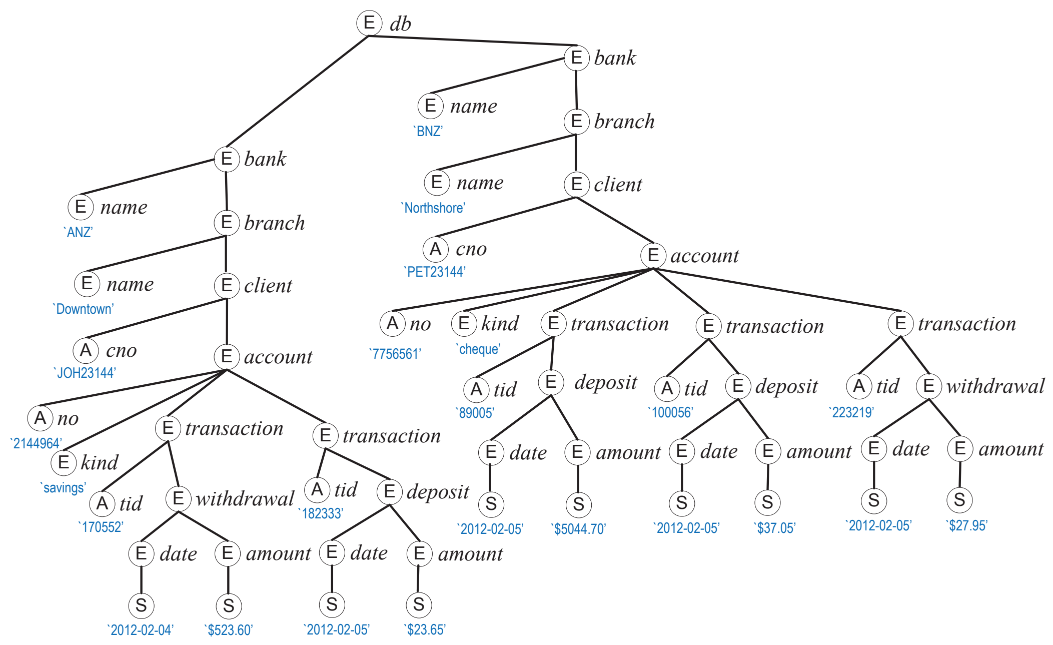

Figure 1: Tree representation of XML data.

Example 1 *Suppose we use XML to store data about bank transactions. Figure 1 shows an artificially small example of such an XML document. In reality there would be of course far more banks, branches, clients, accounts, and transactions. We use the example just to illustrate the general structure of the database. Among other information, the date, type (e.g., deposit or withdrawal) and amount of each transaction are registered in the database. Each transaction belongs to an account which belongs to a branch of a given bank. A cardinality constraint could now state that each bank can handle up to* $300,000$ *transactions per day.*

More generally, cardinality constraints restrict the cardinality of the answer to some query against the database. For any given bank and any given date, we could ask for the number of transactions registered for this bank and date. The cardinality constraint under inspection just states that the answer will consist of at most $300,000$ transactions. It is easy to see that occurrence constraints, as defined by XML Schema, are not expressive enough to capture such properties. The reason is that XML Schema occurrence constraints simply restrict the number of occurrences of XML elements, independently of the data carried by these elements. Instead, a class of constraints is required that restricts the number of XML elements dependent on the data that they carry. Such a class is reminiscent of the cardinality constraints from conceptual modeling [35, 51] and is tailored towards the particularities of XML.

Throughout we will use a simple XML tree model as proposed by DOM [2] and XPath, but independently from schema specifications such as DTDs [7] or XSDs [56]. Figure 2 shows a tree representation of an XML data fragment in which nodes are annotated by their type: E for elements, A for attributes, and S for text (PCDATA) in XML data.

In this article we study cardinality constraints that restrict the number of nodes in

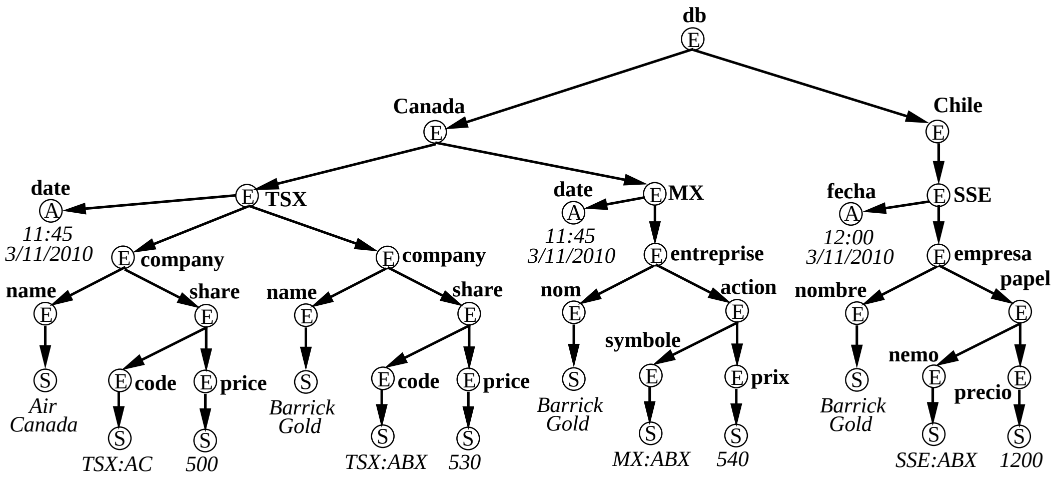

Figure 2: XML data from integration.

an XML tree that have the same (complex) values on some selected descendant nodes. Addressing the tree-like structure of XML data, constraints can impose restrictions for an entire XML document or relatively to selected context nodes.

Evidently, the expressiveness (but also the tractability) of a class of XML cardinality constraints will depend on the query language used to navigate in trees and on the way the cardinalities of XML elements are restricted. As is common in XML data processing, path expressions are used to select the nodes of interest. The preferred languages for selection queries against XML data are XPath [11] and fragments thereof. The flexibility provided by these languages is probably not needed for every application, but there are many applications where flexibility is essential. Taken that XML is widely used to represent heterogeneous data, the expressiveness that comes with wildcards and ancestor-descendent queries is highly appreciated. Just think of data integration where data from different sources are stored in the same document. Though not uniformly structured, the data should still be analyzed using the same 'one fits all' query to avoid extensive maintenance costs. This can only be achieved when using a sufficiently rich query language.

Example 2 *Consider the XML tree T in Figure 2. The XML document is the result of integrating XML data from different sources, i.e., stock markets around the world. Each market provides, in the local official language, information regarding shares. For instance, the Toronto Stock Exchange (TSX) shows this information in English while the Montréal Exchange (MX) chooses French and the Santiago Stock Exchange (SSE) chooses Spanish. It is assumed that the markets provide the information as XML documents with the structure of the subtrees rooted at the nodes at level three (the* company*-nodes, the* entreprise*-node and the* empresa*-node in our example). Suppose Canada has a policy that a company can list its shares in at most two of the stock markets that operate in the country. Further, each company can list its shares in up to four stock markets globally.*

There is a whole range of database tasks that can benefit from cardinality constraints. These include consistency management, integrity enforcement, query optimization, and view maintenance. For example, they can help to predict the number of query answers, updates or de/encryptions. When evaluating queries it is often desirable to estimate the cardinality of the result (or of intermediate results) a priori. Such predictions are crucial for making the right decisions during query optimization, but also for allocating the right resources (such as bandwidth, storage capacity, processing power) in distributed query answering. Resource bottlenecks have been identified as one of the top-ten challenges in cloud computing. We hasten to note that while the focus is often on upper bounds, lower bounds are of interest too, e.g., for ensuring data privacy.

Example 3 *Assume we would like to provide a service where clients receive up-to-date information regarding company shares that are traded in different stock markets worldwide. Suppose a client is using a smart phone to receive the current value for all shares of the multinational company* Barrick Gold*. Clients will only use the service if the costs are reasonable, but our service provider prefers to integrate data from different sources only when the service has been paid for. If reasoning about cardinality constraints can be automated, then it is likely that we can estimate the maximum number of answers to a client's query and hence the maximum cost for the service. Thus, the client is able to make an informed choice and the service provider has minimized its cost for unpaid services, without querying or integrating data.*

As a more concrete example, we show next how the creation of XML views based on cardinality constraints may lead to simplified integrity checking, and more efficient processing of common queries and updates.

Example 4 *Let us consider the following XQuery, which expresses over XML trees with the structure of the tree in Figure 2, the user request for the current value of all shares of* Barrick Gold*.*

```
for $s in doc("shares.xml")/*/*/*
where $s[1]="Barrick Gold"                    (1)
return ⟨share⟩{$s[2]}⟨/share⟩
```

It can be expensive to process an XQuery such as this one over the original XML tree since in the worst case, for all "stock market" nodes, it has to be decided if it has a descendant "company" node which corresponds to the particular company. Taking into account the cardinality constraint in Example 2 which stipulates that each company can list its shares in up to four stock markets globally, a better approach could be to create an XML view as in Figure 3 (only the fragment for Barrick Gold *is shown). Then, we could rewrite query (1) as follows so that it can be evaluated on this view instead.*

```
for $c in doc("shares_view.xml")
      //company[@name]="Barrick Gold"
return ⟨share⟩{$c/*/*/*}⟨/share⟩
```

On the XML view the constraint translates into the condition that under each company-*node there are up to four nodes corresponding to shares, independently of any other data. Thus, it is clearly better to pose queries regarding shares of a specific company against this XML view. This view would also simplify partial updates of the share values corresponding to specific companies, by allowing us to easily identify the external resources (share markets) from which we need to request updated information.*

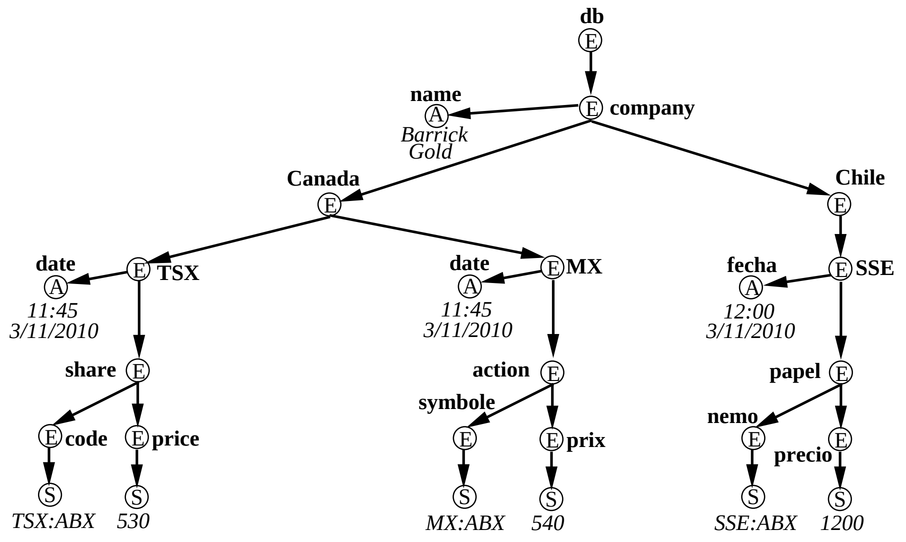

Figure 3: XML view fragment.

1.1 Contributions

If we want to make effective use of cardinality constraints for particular database tasks then we should be able to reason efficiently about sets of cardinality constraints. This ability is particularly important if we need to make decisions during the runtime of the database. Therefore, we aim to identify precious classes of cardinality constraints. We will first introduce an expressive class of cardinality constraints. This class can be used to capture a wide range of interesting properties of the data under discussion. The first source of expressiveness is the ability to specify lower and upper bounds on the number of nodes (called target nodes) in an XML tree that are value-equal on some of its subnodes (called field nodes). The bounds can be specified with respect to a context node. The second source results from the very general notion of value equality: two element nodes v and w are considered value equal, if the subtrees rooted on v and w are isomorphic by an isomorphism that is the identity on string values. This contrasts with more restrictive notions, for instance value equality on leaf nodes. The third source is a result of the path language we use to select nodes. This includes the single-label wildcard, child navigation, and descendant navigation as known from XPath.

However, this high level of expressiveness comes at a price. For our general class of cardinality constraints we will single out three fragments that are likely to be intractable as their associated implication problems are all *coNP*-hard. The fragments direct our attention towards the class of *max-constraints* which allow the specification of upper bounds and use the general notion of value-equality. In the conference version [14] we have considered max-constraints whose context and target nodes can be selected by our general path language, and field nodes by using single-label wildcards and child navigation. While this class of cardinality constraints can already capture many desirable properties of XML data (e.g. the business rules in our example), it is still tractable.

In the current article we consider an even wider class of max-constraints where we also permit the use of variable-length wildcards for the selection of field nodes. We do that in a controlled manner so that tractability is still guaranteed. Moreover, we permit the use of max-constraints that have no field paths: These constraints impose upper bounds on the total number of targets in the scope of a context node. Again we do that in a controlled way to ensure tractability. This class includes maxOccurs constraints, as defined by XML Schema, as a special case, but its expressiveness goes far beyond that.

For our new, wider class of max-constraints we characterize the implication problem axiomatically as well as algorithmically using shortest path methods in a suitable graph. This constitutes a well-founded framework for developing a compact, low-degree polynomial time algorithm that decides implication efficiently. Hence, we establish a precious class of cardinality constraints that is effective for flexible XML data processing.

Finally, we discuss the use of max-constraints as soft constraints. These constraints impose bounds on the average number of targets that are in the scope of the same context node and value-equal on their field nodes, but tolerate violations for some targets. Soft constraints are particularly attractive in database practice, where violations to common rules frequently occur.

1.2 Organization

Section 2 assembles preliminary terminology and formal notation to be used later on. In Section 3 we introduce an expressive class of cardinality constraints and investigate the tractability of their implication problem. To overcome potential intractability we introduce the class of max-constraints. In Section 4 we establish a finite set of inference rules for deriving new max-constraints, and verify its completeness in Section 5. Section 6 presents an efficient algorithm for deciding implication. In Section 7 we discuss a weaker interpretation of max-constraints as soft constraints and show that it is still tractable. In Section 8 we survey some recent work in the literature on cardinality constraints and on database management support for XML that is related to our study here. Section 9 concludes our work.

2 Terminology

2.1 The XML Tree Model

Let $\mathbf{E}$ denote a countably infinite set of element tags, $\mathbf{A}$ a countably infinite set of attribute names, and $\{S\}$ a singleton set denoting text (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.

For a node $v \in V$, each node w in $ele(v)$ or $att(v)$ is called a *child* of v, and we say that there is an edge (v, w) from v to w in T. A *path* p of T is a finite sequence of nodes $v_0, \dots, v_m$ in V such that (v_{i-1}, v_i) is an edge of T for $i = 1, \dots, m$. The path p determines a word $lab(v_1). \cdots .lab(v_m)$ over the alphabet $\mathcal{L}$, denoted by $lab(p)$. For a node $v \in V$, each node w reachable from v is called a *descendant* of v. Note that every XML tree has a tree structure: for each node $v \in V$, there is a unique path from the root node r to v.

Two nodes $u, v \in V$ are *value equal*, denoted by $u =_v v$, whenever the following conditions are satisfied: (a) $lab(u) = lab(v)$, (b) if u, v are attribute or text nodes, then $val(u) = val(v)$, (c) if u, v are element nodes, then *(i)* if $att(u) = \{a_1, \dots, a_m\}$, then $att(v) = \{a'_1, \dots, a'_m\}$ and there is a permutation π on $\{1, \dots, m\}$ such that $a_i =_v a'_{\pi(i)}$ for $i = 1, \dots, m$, and *(ii)* if $ele(u) = [u_1, \dots, u_k]$, then $ele(v) = [v_1, \dots, v_k]$ and $u_i =_v v_i$ for $i = 1, \dots, k$. Note that the notion of value equality takes the document order of the XML tree into account. We remark that $=_v$ is an equivalence relation on the node set V of the XML tree.

2.2 Node Selection Queries

Regular paths have been widely used to express queries for selecting nodes in XML trees. In the sequel, we use the path language $PL^{\{.,\_,\_^*\}}$ consisting of expressions given by the following grammar:

$$Q \to \ell \mid \varepsilon \mid Q.Q \mid \_ \mid \_^*$$

Herein, $\ell \in \mathcal{L}$ 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* 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 $\mathcal{L}$. For example, *Canada.TSX.share* is a refinement of *Canada._.share*. Note that $\lesssim$ is a pre-order on $PL^{\{.,\_,\_^*\}}$. Let $\sim$ denote the congruence induced by the identity $\_^*.\_^* = \_^*$ on $PL^{\{.,\_,\_^*\}}$, and observe that $P \sim Q$ holds if and only if P and Q are refinements of each other. For example, *Canada._*.share* $\sim$ *Canada._*._*.share*.

Regular paths allow one to navigate in an XML tree. We briefly recall the semantics of expressions from $PL^{\{.,\_,\_^*\}}$ in the context of XML. 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 nodes $v, w \in V$, we write $T \models Q(v, w)$ if w is reachable from v following a Q-path in T.

For a node v in the XML tree T, let $v[\![Q]\!]$ denote the set of nodes in T that are reachable from v following any Q-path, that is, $v[\![Q]\!] = \{w \mid T \models Q(v, w)\}$. In particular, we use $[\![Q]\!]$ as an abbreviation for $r[\![Q]\!]$ where r is the root node.

For a subset $\mathcal{Z} \subseteq \{., \_, \_^*\}$, let $PL^{\mathcal{Z}}$ denote the subset of $PL^{\{.,\_,\_^*\}}$ with expressions restricted to the constructs in $\mathcal{Z}$. In particular, $PL^{\{.\}}$ is the set of simple path expression without wildcards.

Since attribute and text nodes in an XML tree T are always leaves, $Q \in PL^{\{.,\_,\_^*\}}$ is *valid* only if it has no labels $\ell \in \mathbf{A}$ or $\ell = S$ in a position other than the terminal one. Note that each prefix of a valid Q is valid, too.

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 $v[\![P]\!] \subseteq v[\![Q]\!]$. It follows immediately from the definition that $P \lesssim Q$ implies $P \subseteq Q$.

We work with the quotient set $PL^{\{.,\_,\_^*\}}_{/\sim}$ rather than with $PL^{\{.,\_,\_^*\}}$ directly: A word from $PL^{\{.,\_,\_^*\}}$ is in *normal form* if it has no consecutive variable-length wildcards, i.e., if it has no consecutive “_*” and no occurrence of “_*._”. Note that, each congruence class contains a unique word in normal form. Each word from $PL^{\{.,\_,\_^*\}}$ can be transformed into normal form in linear time, just by removing superfluous variable-length wildcards and replacing each occurrence of “_*._” by “_._*”. The *length* $|Q|$ of a $PL^{\{.,\_,\_^*\}}$ expression Q is the number of labels in Q plus the number of wildcards (counting both variable-length and single-label wildcards) in the normal form of Q.

The empty path expression ε has length 0. The natural homomorphism from $PL^{\{.,\_,\_^*\}}$ to $PL^{\{.,\_,\_^*\}}_{/\sim}$ is an isomorphism when restricted to words in normal form. By abuse of notation we use the words from $PL^{\{.,\_,\_^*\}}$ to denote their respective congruence class.

For nodes v and v' of an XML tree T, the *value intersection* of $v[\![Q]\!]$ and $v'[\![Q]\!]$ is given by $v[\![Q]\!] \cap_v v'[\![Q]\!] = \{(w, w') \mid w \in v[\![Q]\!], w' \in v'[\![Q]\!], w =_v w'\}$. That is, $v[\![Q]\!] \cap_v v'[\![Q]\!]$ consists of all those node pairs in T that are value equal and are reachable from v and v', respectively, by following Q-paths.

3 From Expressive Towards Precious Classes

The expressiveness of our class of cardinality constraints results from the ability to specify both lower and upper bounds, from the generality of our notion of value-equality and that of the path language. For more expressive path languages the containment problem becomes at least intractable [42]. Let $\mathbb{N}$ denote the positive integers, and let $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$.

Definition 1 *A* cardinality constraint φ *for XML is an expression of the form*

$$card(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$$

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, where* k *is a non-negative integer, and where* $\min \in \mathbb{N}$ *and* $\max \in \bar{\mathbb{N}}$ *with* $\min \leq \max$.

Herein, Q *is called the* context path*,* Q' *is called the* target path*,* $Q_1, \ldots, Q_k$ *are called the* field paths*,* min *is called the* lower bound*, and* max *the* upper bound *of* φ*. If* $Q = \epsilon$*, we call* φ absolute*; otherwise* φ *is called* relative. $\blacksquare$

For a cardinality constraint φ, let Q_φ denote its context path, Q'_φ its target path, $Q_1^\varphi, \ldots, Q_{k_\varphi}^\varphi$ its field paths, $\min_\varphi$ its lower bound, and $\max_\varphi$ its upper bound. Let $\sharp S$ denote the cardinality of a finite set S, i.e., the number of its elements.

Definition 2 *Let $\varphi = card(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$ be a cardinality constraint. An XML tree T* satisfies *φ, denoted by $T \models \varphi$, if and only if for all $q \in [\![Q]\!]$, for all $q'_0 \in q[\![Q']\!]$ such that for all $x_1, \ldots, x_k$ with $x_i \in q'_0[\![Q_i]\!]$ for $i = 1, \ldots, k$, it is true that*

$$\min \leq \sharp\{q' \in q[\![Q']\!] \mid \exists y_1, \ldots, y_k. \forall i = 1, \ldots, k.\, y_i \in q'[\![Q_i]\!] \wedge x_i =_v y_i\} \leq \max$$

holds. ∎

Note that $card(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$ enforces the cardinalities imposed by min and max only on those target nodes $q'_0 \in q[\![Q']\!]$ in T for which for all $i = 1, \ldots, k$, field nodes $x_i \in q'_0[\![Q_i]\!]$ exist in T. Hence, if no such target node q'_0 exists in T, then T automatically satisfies the constraint.

3.1 Some Examples

Recall the XML tree T depicted in Figure 2. We formalize the constraints discussed in the introduction. The constraint $card(\_^*.\text{Canada}, (\_.\_, \{\_.\text{S}\})) = (0, 2)$ says that every company that is listed in a Canadian stock market is listed in at most two of the stock markets that operate in the country. This constraint is satisfied by T.

Instead of making the constraint relative to the subtree rooted at *Canada*, we can make it absolute: $card(\varepsilon, (\_.\_.\_, \{\_.\text{S}\})) = (0, 2)$. This constraint, however, is violated by T since the company *Barrick Gold* has its shares listed in three stock markets.

The following constraint states that each company lists its shares in at least two and at most four stock markets: $card(\varepsilon, (\_.\_.\_., \{\_.\text{S}\})) = (2, 4)$. This constraint is also absolute and violated by T since the company *Air Canada* is only listed in the Toronto Stock Exchange.

The constraint $card(\_, (\_, \emptyset)) = (2, \infty)$ illustrates the case in which the set of field paths is empty. It states that each country has at least two stock markets or none at all. Again, this constraint is not satisfied by T since *Chile* has only one stock market.

It is noteworthy that XML keys as studied in [8, 13, 15, 23, 25] are a special case of cardinality constraints. In fact, a cardinality constraint φ is an XML key precisely when $\min_\varphi = \max_\varphi = 1$. An example of an XML key is $card(\_.\_, (\_, \{\_.\text{S}\})) = (1, 1)$ stating that a company cannot be listed more than once in the same stock market. The key $card(\_^*.\text{TSX}, (\_^*.\text{share}, \{\text{code})) = (1, 1)$ states that in the Toronto Stock Exchange a share is identified by its code; and $(card(\varepsilon, (\_.\_.\_.\_, \{\_.\text{S}\})) = (1, 1)$ states that a share is identified by the text values of its code and price. Note that the XML keys of [26] are not covered by cardinality constraints.

3.2 Intractability of the Implication Problem

In order to take advantage of XML applications effectively it becomes necessary to reason about constraints efficiently. Central to this task is the implication problem. Let $\Sigma \cup \{\varphi\}$ be a finite set of constraints in a class $\mathcal{C}$. We say that Σ *(finitely) implies* φ, denoted by $\Sigma \models_{(f)} \varphi$, if and only if every (finite) XML tree T that satisfies all $\sigma \in \Sigma$ also satisfies φ. The *(finite) implication problem* for the class $\mathcal{C}$ is to decide, given any finite set of constraints $\Sigma \cup \{\varphi\}$ in $\mathcal{C}$, whether $\Sigma \models_{(f)} \varphi$. If Σ is a finite set of constraints in $\mathcal{C}$ let

$\Sigma^*_{(f)}$ denote its *(finite) semantic closure*, i.e., the set of all constraints (finitely) implied by Σ. That is, $\Sigma^*_{(f)} = \{\varphi \in \mathcal{C} \mid \Sigma \models_{(f)} \varphi\}$.

Unfortunately, the price for the general notion of cardinality constraints results in the intractability of the finite implication problem. As the following result shows this is already true for some large subclasses [27].

Theorem 1 *The finite implication problem for each of the following classes*

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

is coNP-hard. ▮

These results are proven as follows. For each of the classes considered in Theorem 1, the 3-colorability problem over graphs polynomially transforms to the complement of the implication problem. That is, for each of the classes $\mathcal{C}_i$ in Theorem 1 and each graph G, there is a corresponding constraint set $\Sigma_i \cup \{\varphi_i\} \in \mathcal{C}_i$ such that $\Sigma_i \not\models \varphi_i$ iff G is 3-colorable. The values of the variables k, *min* and *max* are determined by the actual constraint sets used in each of these transformations.

Using the previous theorem we discover at least three different sources for the observed intractability: i) the simultaneous use of both lower and upper bounds as permitted in $\mathcal{C}_1$, ii) the complete absence of field paths as permitted in $\mathcal{C}_2$, and iii) the simultaneous use of arbitrary length wildcards in both target and field paths as permitted in $\mathcal{C}_3$.

3.3 Max-Constraints to the Rescue

Interestingly, the three sources of intractability mentioned above are the only sources. When looking for a subclass of cardinality constraints that is as expressive as possible but does not include any of $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, our attention is naturally directed to the following one:

$$\mathfrak{M}^{ext} = \{card(Q, (Q', \{Q_1, \dots, Q_k\})) = (1, \max) \mid \begin{array}{l} Q, Q', Q_1, \dots, Q_k \in PL^{\{.,\_,\_^*\}} \text{ but s.t.} \\ Q' \text{ or } Q_1. \cdots .Q_k \in PL^{\{.,\_\}}\} \end{array}$$

It is easy to see that $\mathfrak{M}^{ext}$ does not include any of $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$. As we have seen in Theorem 1 we are likely to experience computational difficulties if we allow cardinality constraints with an empty set of field paths. For the time being, we will therefore assume that we have $k \geq 1$ for all max-constraints in $\mathfrak{M}^{ext}$. In Section 5.4 we will come back to this issue. For $k_\varphi = 0$ we set $Q^\varphi_1. \cdots .Q^\varphi_{k_\varphi} = \_^*$. So, whenever we do allow a max-constraint φ to have an empty set of field paths, then φ will have no variable-length wildcard in its target path. As we will see in Section 5.4 all our results still hold true when we allow $k = 0$ with this restriction for $\mathfrak{M}^{ext}$.

Remark 1 *Note that in the conference version [14] of this article we were investigating the proper subset $\mathfrak{M}$ of $\mathfrak{M}^{ext}$ where the field paths $Q_1, \dots, Q_k$ are all required to be expressions in $PL^{\{.,\_\}}$. However, the proof arguments from [14] can be carefully generalized to show that even the implication problem for the more expressive class $\mathfrak{M}^{ext}$ can be decided efficiently.*

We give some examples to illustrate the difference between both classes: the constraint $card(\varepsilon, (\_^*.\mathrm{account}, \{\mathrm{transaction}.\_.\mathrm{date.S}\})) \leq 100$ belongs to the class $\mathfrak{M}$ and thus also to the class $\mathfrak{M}^{ext}$. On the other hand, the constraint $card(\varepsilon, (\_.\mathrm{account}, \{\mathrm{transaction}.\_^*.\mathrm{date.S}\})) \leq 100$ belongs to the class $\mathfrak{M}^{ext}$ but not to $\mathfrak{M}$. The constraint $card(\varepsilon, (\_^*.\mathrm{account}, \{\mathrm{transaction}.\_^*.\mathrm{date.S}\})) \leq 100$ belongs to neither $\mathfrak{M}$ nor to $\mathfrak{M}^{ext}$ as it contains variable-length wildcards in the target path *and* in a field path.

Definition 3 *We call $\mathfrak{M}^{ext}$ the class of* max-constraints for XML *and use the abbreviation*

$$card(Q, (Q', \{Q_1, \dots, Q_k\})) \leq \max$$

to denote these constraints. $\blacksquare$

The classes $\mathfrak{M}$ and $\mathfrak{M}^{ext}$ are quite expressive. In particular, they each subsume the class of XML keys [13] for the special case where $\max_\varphi = 1$. One of the major contributions of the conference version [14] was the inclusion of single-label wildcards in the definition of max-constraints. We would like to emphasize the significance of this extension. In fact, this feature adds expressiveness to the language that has important applications in data integration.

Example 5 *We will illustrate this point by the example from the introduction. Let us consider the constraints $\varphi_1 = card(\_^*.Canada, (\_.\_, \{\_.S\})) \leq 2$ and $\varphi_2 = card(\varepsilon, (\_.\_.\_, \{\_.S\})) \leq 4$. That is, every company is listed in at most two of the stock markets that operate in Canada, and every company lists its shares in at most four stock markets. Clearly, these constraints cannot be expressed without the single-label wildcard when we consider trees with the structure of T in Figure 2.*

We could represent the same information in a tree T' different from T. For example, replacing the element nodes *Canada* and *Chile* by element nodes *country* with attribute children *name* where $val(name) = Canada$ and $= Chile$, respectively; replacing the element nodes *TSX*, *MX* and *SSE* by element nodes *market* with attribute children *date* and *name* where $val(name) = TSX$, $= MX$, and $= SSE$, respectively; and replacing all non-English labels of the remaining element nodes by their corresponding English translation, i.e., replacing *empresa* by *company*, *papel* by *share* and so on.

Over trees with the structure of T', it is easy to check that the constraint $card(\varepsilon, (\mathrm{country.market.company}, \{name.\mathrm{S}\})) \leq 4$ (without single-label wildcards) is equivalent to φ_2. However, φ_1 is not meaningful in T' and there is no cardinality constraint φ_1' such that for every tree T_i with the structure of T and every corresponding equivalent tree T_i' with the structure of T', $T_i \models \varphi_1$ iff $T_i' \models \varphi_1'$. We can replace the *country*-nodes in T' with the labels they have in T, but then no constraint without a single-label wildcard is equivalent to φ_2.

Remark 2 *In the current article, we now also allow variable-length wildcards in the field paths (as long as there is no variable-length wildcard in the target path of the same constraint). That is, we can now take advantage of the combined expressiveness of max-constraints with single-label wildcards in all parts, and with variable-length wildcards in the context path and in either the target path or the field paths. Relaxing this restriction any further is likely to cause computational problems by virtue of Theorem 1, thus giving rise to a class of constraints that is no longer precious. For the class $\mathfrak{M}^{ext}$, however, we can guarantee that it is precious, as we will prove in the current article. Clearly, our results on $\mathfrak{M}^{ext}$ here subsume the results on the proper subclass $\mathfrak{M}$ studied in the conference version [14].*

3.4 A Plan for Verifying the Tractability of Max-Constraints

The class $\mathfrak{M}^{ext}$ of max-constraints provides XML engineers with an enhanced ability to capture interesting properties of XML data. These are useful for several tasks in XML practice, including data integration and cardinality estimation. In the remainder of the article we will establish that $\mathfrak{M}^{ext}$ is indeed a precious class of cardinality constraints. That is, despite its expressiveness the class $\mathfrak{M}^{ext}$ can be reasoned about efficiently.

We will proceed as follows: First we will characterize the implication problem associated with $\mathfrak{M}^{ext}$ in terms of a finite axiomatization. We can speak of *the* implication problem as the finite and unrestricted implication problems coincide for the class $\mathfrak{M}^{ext}$. Note that this is different for the general class of cardinality constraints. The axiomatization provides complete insight into the interaction of max-constraints. This insight allows us to characterize the implication problem by constructive graph properties. Eventually, this characterization will enable us to establish a compact, low-degree polynomial-time algorithm for deciding implication.

4 Sound Inference Rules for Max-Constraints

Our goal is to establish a finite axiomatization for the implication of max-constraints. To begin with we assemble a set of inference rules that allow us to derive new max-constraints from given ones. Derivability with respect to a set $\mathfrak{R}$ of inference rules, denoted by the binary relation $\vdash_{\mathfrak{R}}$ between a set of max-constraints and a single max-constraint, can be defined analogously to the notion in the relational data model [1, pp. 164-168].

We aim to find a set of inference rules which is *sound* and *complete* for the implication of max-constraints. A set $\mathfrak{R}$ of inference rules is sound (respectively, complete) for the implication of max-constraints if for all finite sets Σ of max-constraints we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*$ (respectively, $\Sigma^* \subseteq \Sigma^+_{\mathfrak{R}}$). Herein, $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ denotes the *syntactic closure* of Σ under derivation by $\mathfrak{R}$.

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

$$\frac{}{card(Q,(Q',S)) \leq \infty}\; Q' \in PL^{\{\cdot,\text{-}\}} \text{ or } \emptyset \neq S \subseteq PL^{\{\cdot,\text{-}\}}$$
(infinity)

$$\frac{}{card(Q,(\epsilon,S)) \leq 1}$$
(epsilon)

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

$$\frac{card(Q,(Q',S)) \leq \max}{card(Q,(Q',S)) \leq \max +1}$$
(weakening)

$$\frac{card(Q,(Q',S)) \leq \max}{card(Q,(Q',S \cup \{P\})) \leq \max}\; Q' \text{ or } P \in PL^{\{\cdot,\text{-}\}}$$
(superfield)

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

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

$$\frac{card(Q,(Q',S)) \leq \max}{card(Q'',(Q',S)) \leq \max}\; Q'' \subseteq Q$$
(context-path-containment)

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

$$\frac{card(Q,(Q',S)) \leq \max}{card(Q,(Q'',S)) \leq \max}\; Q'' \subseteq Q'$$
(target-path-containment)

$$\frac{\begin{array}{c} card(Q,(Q',\{P.P_1,\ldots,P.P_k\})) \leq \max, \\ card(Q.Q',(P,\{P_1,\ldots,P_k\})) \leq \max' \end{array}}{card(Q,(Q'.P,\{P_1,\ldots,P_k\})) \leq \max \cdot \max'}$$
(multiplication)

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

Table 1: A Finite Axiomatization for Max-constraints in $\mathfrak{M}^{ext}$.

We prove below the soundness of the *field-path containment* rule and the *subnodes* rule. The soundness of the remaining rules can be shown using similar arguments. Therefore we omit those lengthy, but not very difficult proofs. For comparison, we also refer to our soundness proofs in [27] for the special case where no single-label wildcards are permitted, and variable-length wildcards are permitted to occur only in the context and target paths.

Lemma 1 *The* field-path containment *rule is sound for the implication of max-constraints in the class $\mathfrak{M}^{ext}$.*

Proof Suppose an XML tree T violates $card(Q,(Q',S \cup \{P'\})) \leq \max$. Let $S \cup \{P'\} = \{P_1,\ldots,P_k\}$ such that $P_k = P'$ with $k \geq 1$. Then there is some node $q \in [\![Q]\!]$ and some node $q'_0 \in q[\![Q']\!]$ such that for some $x_1,\ldots,x_k$ with $x_i \in q'_0[\![P_i]\!]$ for $i = 1,\ldots,k$, it holds that

$$\sharp\{q' \in q[\![Q']\!] \mid \exists y_1,\ldots,y_k.\forall i = 1,\ldots,k.\ y_i \in q'[\![P_i]\!] \wedge x_i =_v y_i\} > \max .$$

However, since $P' = P_k \subseteq P$, for every $q' \in q[\![Q']\!]$ it holds that there is a $y_k \in q'[\![P_k]\!]$ with $x_k =_v y_k$ if it also holds that $y_k \in q'[\![P]\!]$. But then it is easy to see that there are

$x_1, \dots, x_k$ with $x_i \in q'_0[\![P_i]\!]$ for $i = 1, \dots, k-1$ and $x_k \in q'_0[\![P]\!]$ such that

$$\begin{aligned}\sharp\{q' \in q[\![Q']\!] \mid\ & \exists y_1, \dots, y_k. \forall i = 1, \dots, k-1. y_i \in q'[\![P_i]\!] \wedge \\ & y_k \in q'[\![P]\!] \wedge x_i =_v y_i \wedge x_k =_v y_k\} > \max .\end{aligned}$$

This shows that T also violates $\sigma = card(Q, (Q', S \cup \{P\})) \leq \max$.

Lemma 2 *The* subnodes *rule is sound for the implication of max-constraints in the class $\mathfrak{M}^{ext}$.*

Proof Suppose an XML tree T violates $card(Q, (Q', \{P.P'\})) \leq \max$. Then there is some node $q \in [\![Q]\!]$ and some node $q'_0 \in q[\![Q']\!]$ such that for some $x \in q'_0[\![P.P']\!]$ it holds that

$$\sharp\{q' \in q[\![Q']\!] \mid \exists y. y \in q'[\![P.P']\!] \wedge x =_v y\} > \max .$$

For every pair of distinct nodes q'_i and q'_j in the previous set, it also holds that q'_i is neither an ancestor nor a descendant of q'_j. This holds true, since T is a tree and due to the condition of the *subnodes* rule Q' or $P.P'$ is a $PL^{\{.,\_\}}$ expression, that is, has no variable-length wild cards. Consequently, we have

$$\sharp\{p \in q[\![Q'.P]\!] \mid \exists y. y \in p[\![P']\!] \wedge x =_v y\} > \max .$$

This shows that T also violates $\sigma = card(Q, (Q'.P, \{P'\})) \leq \max$.

Remark 3 *In the proof of the previous lemma we have made an interesting observation: If we allow the use of variable-length wildcards in the target and field paths at the same time, then we could have a pair of distinct target nodes in the set $\{q' \in q[\![Q']\!] \mid \exists y. y \in q'[\![P.P']\!] \wedge x =_v y\}$ so that one is an ancestor of the other one. This would allow us to build a counterexample tree to demonstrate that the* subnodes *rule is not sound for such an extended class of max-constraints. However, we do not study this case any further since the implication problem becomes intractable as seen in Theorem 1.*

5 Axiomatic Characterization of Implication

Our goal is to demonstrate that the set $\mathfrak{R}$ of inference rules is also complete for the implication of max-constraints in the class $\mathfrak{M}^{ext}$. Completeness means we need to show that for an arbitrary finite set $\Sigma \cup \{\varphi\}$ of max-constraints in the class $\mathfrak{M}^{ext}$ with $\varphi \notin \Sigma^+_{\mathfrak{R}}$ 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.

The general proof strategy is as follows: For T to be a counter-example we i) require a context node q_φ with more than $\max_\varphi$ target nodes q'_φ that all have value-equal field nodes $q^\varphi_1, \dots, q^\varphi_{k_\varphi}$, and ii) must for each context node q_σ not have more than $\max_\sigma$ target nodes q'_σ that all have value-equal field nodes $q^\sigma_1, \dots, q^\sigma_{k_\sigma}$, for each member σ of Σ. Basically, such a counter-example tree exists if and only if these two conditions can be satisfied simultaneously. In a first step, we represent φ as a finite node-labeled tree $T_{\Sigma,\varphi}$, which we call the *mini-tree*.

Then, we reverse the edges of the mini-tree and add to the resulting tree downward edges for certain members of Σ. Finally, each upward edge receives a label of 1 and each downward edge resulting from $\sigma \in \Sigma$ a label of $\max_\sigma$. This final digraph $G_{\Sigma,\varphi}$ is called the *cardinality network*. A downward edge resulting from σ tells us that under each source node there can be at most $\max_\sigma$ target nodes. Now, if we can reach the target node of φ from its context node along a dipath of weight (the product of its labels) at most $\max_\varphi$, then there is no counter-example tree T. In other words, if we satisfy condition ii) above, then we cannot satisfy condition i). Otherwise, we can construct a counter-example tree T.

5.1 Cardinality Networks

Let $\Sigma \cup \{\varphi\}$ be a finite set of max-constraints in the class $\mathfrak{M}^{ext}$. 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 $\ell_0 \in \mathbf{E} - \mathcal{L}_{\Sigma,\varphi}$. 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 constraint in $\Sigma \cup \{\varphi\}$. This transformation turns Q_φ into O_φ, Q'_φ into O'_φ, and each Q^φ_i into O^φ_1 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).

The proper choice of the integer l is essential for the later construction. In particular, if there are no occurrences of single-label wildcards in the max-constraints under consideration, then $l = 0$ and we just replace each variable-length wildcard "_*" by one ℓ_0.

To continue with our construction, 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^φ_i-path from a node r^φ_i to a node x^φ_i, 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 *mini-tree* $T_{\Sigma,\varphi}$ by identifying the node r'_φ with q_φ, and by identifying each of the nodes r^φ_i with q'_φ.

The *marking* of the mini-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 $Q^\varphi_i \neq \varepsilon$, then $\mathcal{M}$ consists of the leaves of $T_{\Sigma,\varphi}$, and otherwise $\mathcal{M}$ consists of all descendant nodes of q'_φ in $T_{\Sigma,\varphi}$.

We use mini-trees to calculate the impact of max-constraints in Σ on a possible counter-example tree for the implication of φ by Σ. To distinguish max-constraints that have an impact from those that do not, we introduce the notion of *applicability*. Intuitively, when a max-constraint is not applicable, then we do not need to satisfy its upper bound in a counter-example tree as it does not require all its field paths. Let $T_{\Sigma,\varphi}$ be the mini-tree of the max-constraint φ with respect to Σ, and let $\mathcal{M}$ be its marking. A 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 $w'_\sigma[\![P^\sigma_i]\!] \cap \mathcal{M} \neq \emptyset$ for all $i = 1, \ldots, k_\sigma$. We say that w_σ and w'_σ *witness* the applicability of σ to φ.

Example 6 *Let us consider an XML database for projects of a company. A year is divided into quarters and each quarter contains a sequence of projects. Each project*

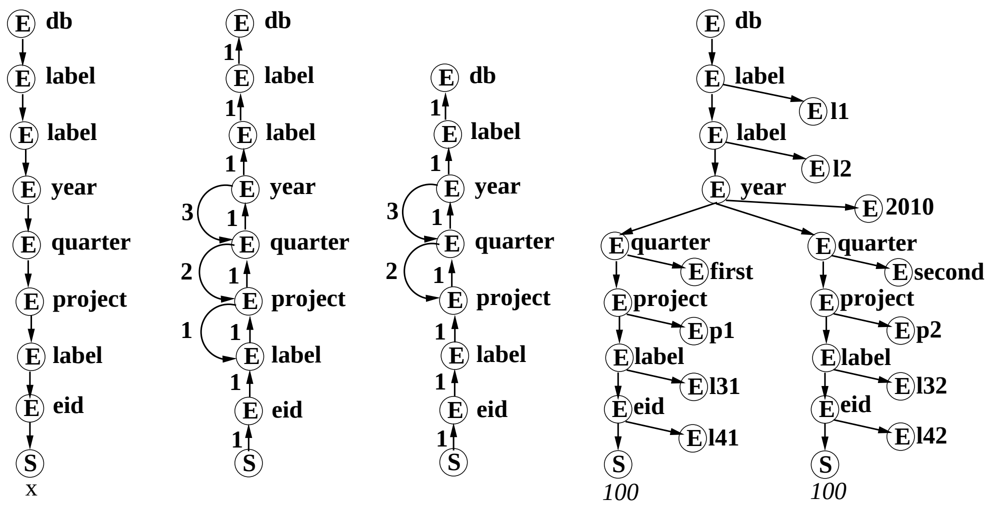

Figure 4: A mini-tree, two cardinality networks and a counter-example tree

has a list of employees labeled by their role in the project (e.g. manager or consultant). Employees are identified by their employee id. In this context, the constraint $\varphi = card(\_^*.year, (quarter.project, \{\_.eid.S\})) \leq 5$ *indicates that in a given year an employee cannot be involved in more than* 5 *projects. Now, suppose that we have a set* $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ *of max-constraints, where*

$$\sigma_1 = card(\_^*.year, (quarter, \{project.\_.eid.S\})) \leq 3, \quad \textit{and}$$

$$\sigma_2 = card(\_^*.year.quarter, (project, \{\_.eid.S\})) \leq 2, \quad \textit{and}$$

$$\sigma_3 = card(\_^*.project, (\_, \{eid.S\})) \leq 1.$$

The first constraint σ_1 *says that the same employee can be involved in projects in up to three quarters of each year. The constraint* σ_2 *says that the same employee can work on at most two different projects per quarter, and* σ_3 *says that no employee can take on more than one role in each project. The left picture of Figure 4 shows the mini-tree of* φ *and its marking (note that leaves are marked by* $\times$*). All max-constraints* σ_1*,* σ_2 *and* σ_3 *are applicable to* φ*. On the other hand,* $\sigma_4 = card(\_.year.\_, (project.\_, \{eid.S\})) \leq 2$ *is not applicable to* φ*.* ∎

We define the *cardinality network* $G_{\Sigma,\varphi}$ of φ and Σ as the node-labeled digraph 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 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 $\omega(u, v) = \min\{\max_\sigma \mid (u, v)$ witnesses the applicability of some $\sigma \in \Sigma$ to $\varphi\}$.

We need to use some graph terminology, cf. [31]. Consider some digraph G. A path t is a sequence $v_0, \ldots, v_m$ of nodes with an edge (v_{i-1}, v_i) for each $i = 1, \ldots, m$. We call t a path of length m from node v_0 to node v_m containing the edges (v_{i-1}, v_i), $i = 1, \ldots, m$. A simple path is just a path whose nodes are pairwise distinct. Note that for every path from u to v there is also a simple path from u to v in G containing only edges of the path. In the cardinality network $G_{\Sigma,\varphi}$ the weight of a path t 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.

Example 7 *Let Σ and φ be as in Example 6. The cardinality network of Σ and φ is illustrated in the second picture from the left in Figure 4. Let v denote the unique* year*-node, w the unique* project*-node, and u the unique* eid*-node in the second picture from the left in Figure 4. Then $d(v, w) = 6$ and $d(v, u) = \infty$.* ▮

In the following section we will prove the following crucial fact. If the distance $d(q_\varphi, q'_\varphi)$ from q_φ to q'_φ in $G_{\Sigma,\varphi}$ is at most $\max_\varphi$, then $\varphi \in \Sigma^+_{\mathfrak{R}}$. In other words, if φ is not derivable from Σ, then every path from q_φ to q'_φ in $G_{\Sigma,\varphi}$ has distance at least $\max_\varphi + 1$.

Example 8 *Let $\varphi' = card(\_^*.year, (quarter.project, \{\_.eid.S\})) \leq 6$ and Σ be as in Example 6. The corresponding mini-tree and cardinality network are shown as first and second picture from the left in Figure 4, respectively. Since $d(q_\varphi, q'_\varphi) \leq 6$, it follows by Lemma 3 that φ' is derivable from Σ. In fact, φ' is clearly derivable by a single application of the* multiplication *rule to σ_1 and σ_2.* ▮

5.2 Encoding Inferences as Shortest Paths

Our next result states that we can inspect the cardinality network constructed above to derive new max-constraints from the ones given in Σ. This makes the cardinality network a useful tool for computing new max-constraints that is more convenient to handle than the inference rules in Table 1.

Lemma 3 *Let $\Sigma \cup \{\varphi\}$, where $\varphi = card(Q_\varphi, (Q'_\varphi, \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_\varphi$, be a finite set of max-constraints in the class $\mathfrak{M}^{ext}$. If the distance $d(q_\varphi, q'_\varphi) \leq \max_\varphi$ in the cardinality network $G_{\Sigma,\varphi}$, then $card(Q_\varphi, (Q'_\varphi, \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_\varphi \in \Sigma^+_{\mathfrak{R}}$.* ▮

The strategy to prove this lemma is to encode an inference by $\mathfrak{R}$ by witness edges of the cardinality network. We use the following example to illustrate the main steps of the proof.

Example 9 *Let us consider an XML database for bank transactions. Among other information, the date of each transaction is registered in this database. Each transaction* *belongs to an account which belongs to a branch of a given bank. In this context, the max-constraint*

$$\varphi = card(bank, (branch.\_^*.account.transaction, \{\_.date.S\})) \leq 300,000$$

indicates that each bank can handle up to $300,000$ *transactions per day. Suppose that we have a set of max-constraints* $\Sigma = \{\sigma_1, \sigma_2\}$*, where*

$$\sigma_1 = card(\varepsilon, (\_^*.account, \{transaction.\_.date.S\})) \leq 100, \quad and$$

$$\sigma_2 = card(\_^*.branch, (\_^*.transaction.\_.date, \{\_\})) \leq 3,000$$

The first constraint σ_1 *says that in any given day, an individual account can be involved in at most* 100 *transaction. The constraint* σ_2 *says that any given bank branch can process up to a maximum of* $3,000$ *transactions per day.*

We will formally prove Lemma 3 following a top-down approach. Thus, we start by laying out the strategy of the proof and leave the technical details for later.

Proof Due to the *infinity* rule there is nothing to show if $\max_\varphi = \infty$. Assume $\max_\varphi < \infty$. If $d(q_\varphi, q'_\varphi) \leq \max_\varphi$, then let D denote the simple path in $G_{\Sigma,\varphi}$ from q_φ to q'_φ with $\omega(D) = d(q_\varphi, q'_\varphi)$. According to the definition of the cardinality network we can assume without loss of generality that D consists of a sequence $\pi_1, \dots, \pi_{n+1}$, $n \geq 1$, where for each $i = 1, \dots, n$, π_i starts with a possibly empty sequence of upward edges each of weight 1 followed by a single witness edge $(w_{\sigma_i}, w'_{\sigma_i})$ labeled with $\max_{\sigma_i}$ where w_{σ_i} and w'_{σ_i} witness the applicability of σ_i to φ, and π_{n+1} is a possibly empty sequence of upward edges labeled with 1. Moreover, we can assume that $q_\varphi, w'_{\sigma_1}, \dots, w'_{\sigma_n}$ form a proper descendant chain, q'_φ is a proper descendant of $w'_{\sigma_{n-1}}$ and w'_{σ_n} is a descendant node of q'_φ in $T_{\Sigma,\varphi}$. This situation is illustrated by the cardinality network $G_{\Sigma,\varphi}$ corresponding to Example 9 which is depicted in Figure 5. Note that $d(q_\varphi, q'_\varphi) = 300,000$ and that the thick arrows show the path D from q_φ to q'_φ with $\omega(D) = d(q_\varphi, q'_\varphi)$.

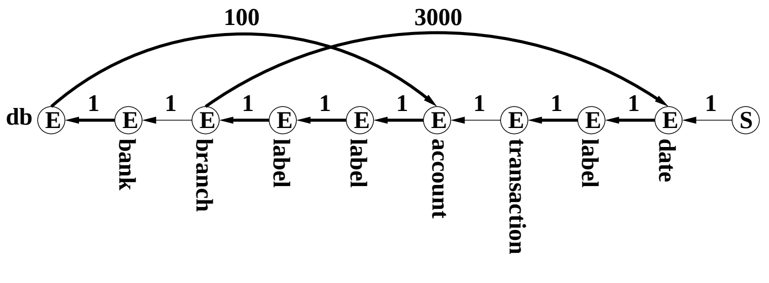

Figure 5: Example 9:Cardinality network

We now describe a series of assumptions which we use to show that the existence of the witness edges in D implies the existence of a witness edge (q_φ, q'_φ) whose weight is that of the original path D and which results from a max-constraint σ in Σ^+. We formally prove that each of these assumptions hold in A afterwards.

1. The final witness edge in D can be replaced by a witness edge that ends in q'_φ. That is, we can assume without loss of generality that π_{n+1} is indeed an empty sequence and $w'_{\sigma_n} = q'_\varphi$ where the set of field paths of σ_n is $\{Q_1^\varphi, \dots, Q_{k_\varphi}^\varphi\}$. In our example, this means that we can assume the existence of an implied max-constraint that determines the witness edge denoted with a dashed arrow in Figure 6.

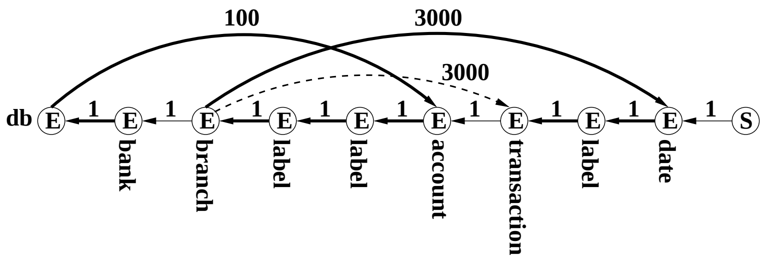

Figure 6: Example 9: Implied witness edge derived from Assumption 1

2. If there is a witness edge (w_σ, w'_σ) in the cardinality network $G_{\Sigma,\varphi}$ that corresponds to the applicability of some $\sigma \in \Sigma_{\mathfrak{R}}^+$ to φ, then for each node w between w_σ and w'_σ in $T_{\Sigma,\varphi}$ there is also a witness edge (w, w'_σ) in $G_{\Sigma,\varphi}$ with $\omega(w_\sigma, w'_\sigma) = \omega(w, w'_\sigma)$ which corresponds to the applicability of some $\sigma' \in \Sigma_{\mathfrak{R}}^+$ to φ. In particular, the two witness edges denoted with dashed arrows in Figure 7, follow from this assumption.

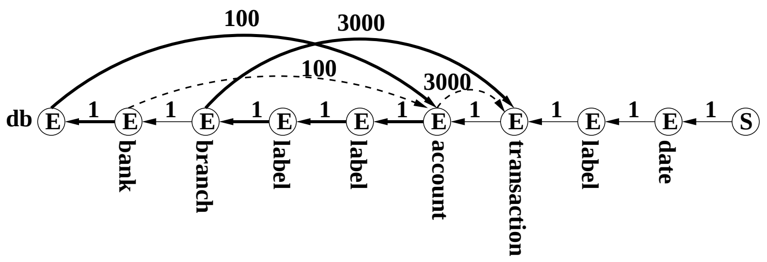

Figure 7: Example 9: Implied witness edges derived from Assumption 2

3. If there is a witness edge $(w_{\sigma_1}, w'_{\sigma_1})$ with weight $\max_{\sigma_1}$ and another witness edge $(w'_{\sigma_1}, q'_\varphi)$ with weight $\max_{\sigma_2}$, then there is also a witness edge $(w_{\sigma_1}, q'_\varphi)$ with weight $\max_{\sigma_1} \cdot \max_{\sigma_2}$. An implied witness edge that can be derived by applying this assumption to our example is shown in Figure 8.

From Assumption 1 and 2, we can conclude that there is a simple path D' in $G_{\Sigma,\varphi}$ from q_φ to q'_φ and $\omega(D) = \omega(D')$. In fact, D' consists of the sequence $\pi'_1, \dots, \pi'_n$ where each π'_i with $1 \le i \le n$ consists of a single witness edge $(w_{\sigma_i}, w'_{\sigma_i})$ labeled with $\max_{\sigma_i}$ and where $w'_{\sigma_i} = w_{\sigma_{i+1}}$ for $i = 1, \dots, n-1$ and $w_{\sigma_1} = q_\varphi$ and $w'_{\sigma_n} = q'_\varphi$. Again, $q_\varphi, w'_{\sigma_1}, \dots, w'_{\sigma_n}$ form a proper descendant chain.

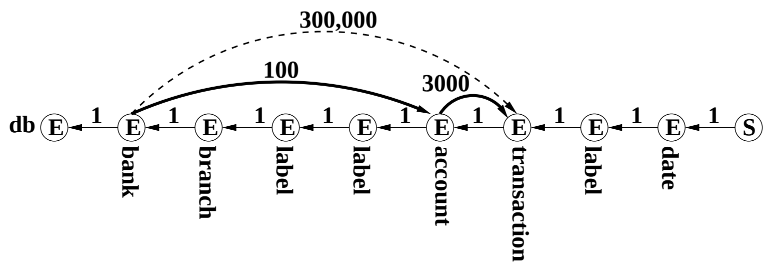

Figure 8: Example 9: Implied witness edge derived from Assumption 3

At this stage we can use Assumption 3 repeatedly to conclude that there is a single witness edge $D_0 = (q_\varphi, q'_\varphi)$ in $G_{\Sigma,\varphi}$ resulting from the max-constraint

$$\sigma = card(Q_\sigma, (Q'_\sigma, \{Q_1^\varphi, \dots, Q_{k_\varphi}^\varphi\})) \leq \prod_{i=1}^{n} \max_{\sigma_i} \in \Sigma_{\mathfrak{R}}^+$$

that is applicable to φ. Due to the applicability of σ to φ we conclude that $Q_\varphi \subseteq Q_\sigma$ and $Q'_\varphi \subseteq Q'_\sigma$. We can now apply the *context-path-containment* and *target-path-containment* rule to obtain

$$card(Q_\varphi, (Q'_\varphi, \{Q_1^\varphi, \dots, Q_{k_\varphi}^\varphi\})) \leq \prod_{i=1}^{n} \max_{\sigma_i} \in \Sigma_{\mathfrak{R}}^+.$$

Since

$$w(D_0) = \prod_{i=1}^{n} \max_{\sigma_i} = \omega(D) = d(q_\varphi, q'_\varphi) \leq \max_\varphi$$

holds, applications of the *weakening* rule show that also

$$card(Q_\varphi, (Q'_\varphi, \{Q_1^\varphi, \dots, Q_{k_\varphi}^\varphi\})) \leq \max_\varphi \in \Sigma_{\mathfrak{R}}^+$$

holds which proves the lemma.

5.3 Establishing Completeness

We have now the tools to prove the completeness of our set of inference rules.

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

Proof Let $\Sigma \cup \{\varphi\}$ be a finite set of max-constraints in the class $\mathfrak{M}^{ext}$ such that $\varphi \notin \Sigma_{\mathfrak{R}}^+$. We construct a finite XML tree T which satisfies all max-constraints in Σ but does not satisfy φ. Since $\varphi \notin \Sigma_{\mathfrak{R}}^+$ every existing path from q_φ to q'_φ in $G_{\Sigma,\varphi}$ has weight at least $\max_\varphi + 1$. For each node n in $G_{\Sigma,\varphi}$ let $\omega'(n) = \omega(D)$ where D denotes the shortest path from q_φ to n in $G_{\Sigma,\varphi}$, or $\omega'(n) = \max_\varphi + 1$ if there is no such path. In particular, we have $\omega'(q_\varphi) = 1$ and $\omega'(q'_\varphi) > \max_\varphi$. Let T_0 be a copy of the path from the root node r to q_φ in $T_{\Sigma,\varphi}$. We extend T_0 as follows: for each node n on the path from q_φ to q'_φ in $T_{\Sigma,\varphi}$ we introduce $\omega'(n)$ copies $n_1, \dots, n_{\omega'(n)}$ into T_0. Suppose T_0 has been constructed down

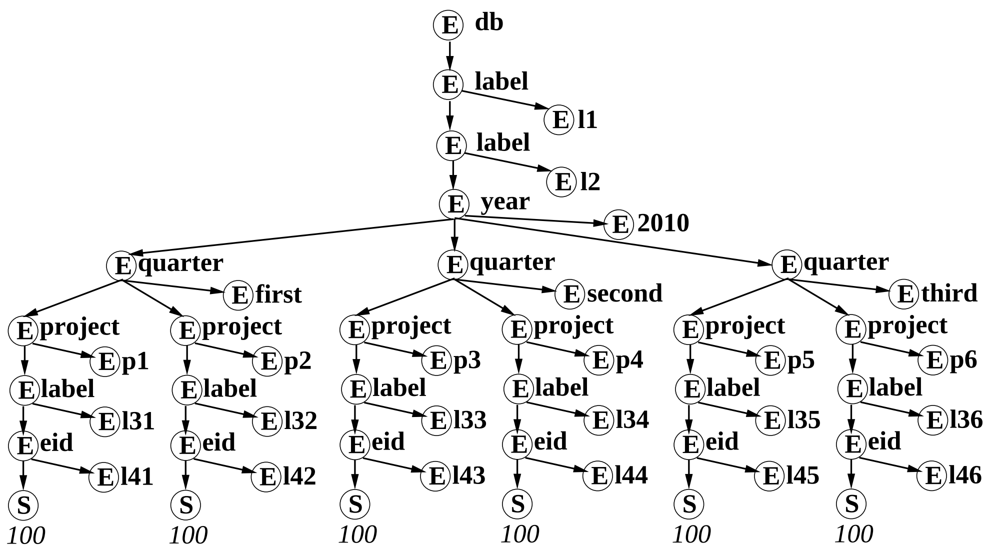

Figure 9: Counter-example tree for the implication of φ by Σ from Example 10.

to the level of $u_1, \dots, u_{\omega'(u)}$ corresponding to node u in $T_{\Sigma,\varphi}$, and let v be the unique successor of u in $T_{\Sigma,\varphi}$. Then $\omega'(u) \leq \omega'(v)$ due to the upward edges in $G_{\Sigma,\varphi}$. For all $i = 1, \dots, \omega'(u)$ and all $j = 1, \dots, \omega'(v)$ we introduce a new edge (u_i, v_j) in T if and only if j is congruent to i modulo $\omega'(u)$. Eventually, T_0 has $\omega'(q'_\varphi) > \max_\varphi$ leaves.

For $i = 1, \dots, \omega'(q'_\varphi)$ let T_i be a node-disjoint copy of the subtree of $T_{\Sigma,\varphi}$ rooted at q'_φ. We want that for any two distinct copies T_i and T_j a node of T_i and a node of T_j become value equal precisely when they are copies of the same marked node in $T_{\Sigma,\varphi}$. For attribute and text nodes this is achieved by choosing string values accordingly, while for element nodes we can adjoin a new child node with a label from $\mathcal{L} - (\mathcal{L}_{\Sigma,\varphi} \cup \{\ell_0\})$ to achieve this. The counter-example tree T is obtained from $T_0, T_1, \dots, T_{\omega'(q'_\varphi)}$ by identifying the leaf node q'_i of T_0 with the root node of T_i for all $i = 1, \dots, \omega'(q'_\varphi)$. We conclude that T violates φ since $\omega'(q') > \max_\varphi$, and our construction guarantees that T satisfies Σ.

The construction of the counter example tree T in the proof of Theorem 2 is illustrated by the following example.

Example 10 *Let Σ and φ be as in Example 6. A counter-example tree T for the implication of φ by Σ is shown in Figure 9. In particular, φ is violated since the unique* year-*node has six distinct* project-*descendants whose corresponding* eid-*grandchildren nodes have the same text content.* ▮

5.4 Dealing with Empty Sets of Field Paths

Cardinality constraints with empty sets of field paths are useful, too. In a sense they complement cardinality constraints with field paths. The former ones give bounds on the total number of elements that can be reached from a context node using the context path as a query. The latter ones bound the number of these elements that share the same information on their field paths.

Example 11 *Consider the XML tree in Figure 9. We could use a constraint* $\varphi = card(\_^*.year, (quarter, \emptyset)) \leq 4$ *to express that every year has at most four quarters.*

Suppose now that we allow $k = 0$ in the definition of $\mathfrak{M}^{ext}$. That is, we consider also max-constraints of the form $\alpha = (Q, (Q', \emptyset))$ whose target path Q' may contain single-label wildcards, but no variable-length wildcards. It is not hard to see that the inference rules in Table 1 still hold for this case. Using the *superfield* rule we can derive $\alpha' = (Q, (Q', \{\_^*\}))$ from α.

So when constructing the cardinality network $G_{\Sigma,\varphi}$ for some Σ and φ in $\mathfrak{M}^{ext}$ such that α belongs to Σ then we will replace α by α'. As we have demonstrated above the counter-example tree T constructed for φ and the resultant Σ' would satisfy Σ' but violate φ. However, it is easy to validate that by our construction T satisfies not only α' but even α. So T would actually satisfy Σ but violate φ, thus showing that Σ does not imply φ.

If φ has an empty set of field paths, then we will construct the cardinality network $G_{\Sigma,\varphi'}$ where $\varphi' = (Q_\varphi, (Q'_\varphi, \{\_^*\})))$. As we have demonstrated above the counter-example tree T constructed for φ' and Σ would satisfy Σ but violate φ'. Consequently, it would also violate φ (as φ implies φ'), thus showing that Σ does not imply φ.

Finally, note that $\varphi' = (Q_\varphi, (Q'_\varphi, \{\_^*\})))$ does not generally imply φ. A counter-example tree T can be easily constructed from merging two copies of $T_{\Sigma,\varphi'}$ on all nodes other than $q'_{\varphi'}$ and then assigning two different string values to the two copies of $q'_{\varphi'}$. Then T would satisfy φ' but violate φ.

Remark 4 *In* $\mathfrak{M}^{ext}$ *we may also allow max-constraints without any field paths, say of the form* $\varphi = card(Q, (Q', \emptyset))$. *However, by definition of* $\mathfrak{M}^{ext}$ *the target path* Q' *must then be in* $PL^{\{.,\_\}}$, *that is, may contain single-label wildcards, but no variable-length wildcards. By virtue of Theorem 1 a further relaxation of the restriction for max-constraints would turn* $\mathfrak{M}^{ext}$ *into a class which is no longer precious.*

6 Algorithmic Characterization of Implication

We will now design a low-degree polynomial time algorithm for deciding the implication problem of max-constraints in $\mathfrak{M}^{ext}$. It is based on the following characterization of the implication problem in terms of the shortest path problem between two suitable nodes of the cardinality network.

Theorem 3 *Let* $\Sigma \cup \{\varphi\}$ *be a finite set of max-constraints in* $\mathfrak{M}^{ext}$. *We have that* $\Sigma \models \varphi$ *if and only if* $d(q_\varphi, q'_\varphi) \leq \max_\varphi$ *in the cardinality network* $G_{\Sigma,\varphi}$. ▮

Theorem 3 suggests to decide implication by constructing the cardinality network and applying well-known shortest paths techniques. This establishes a surprisingly compact method.

The presentation of Algorithm 1 to decide the implication of max-constraints remains the same as Algorithm 1 in [27] to decide the implication of the strictly less expressive class of numerical keys. However, the construction of the cardinality network $G_{\Sigma,\varphi}$, which is

Algorithm 1 Max-constraint implication

```
1: procedure Decide-Implication($\Sigma \cup \{\varphi\}$)
2:     Construct $G_{\Sigma,\varphi}$ for $\Sigma$ and $\varphi$;
3:     Find the shortest path $P$ from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$;
4:     if $\omega(P) \leq \max_\varphi$ then
5:         return(yes);
6:     else
7:         return(no);
8:     end if
9: end procedure
```

central to both algorithms, requires considerably more effort for the more expressive class of max-constraints. This effort results in an increase in the worst-case time complexity of the algorithm compared to numerical keys. Nevertheless, the simplicity of Algorithm 1 enables us to conclude that the implication of max-constraints in $\mathfrak{M}^{ext}$ can be decided in low-degree polynomial time in the worst case.

Theorem 4 *If $\Sigma \cup \{\varphi\}$ is a finite set of max-constraints, 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 [25, 27], then the worst-case time complexity of Algorithm 1 is that of the algorithm dedicated to (numerical) keys only [25].

Remark 5 *If we simply replace each variable-length wildcard "$\_^*$" by the single label ℓ_0 and not by a sequence of $l+1$ labels ℓ_0, then Theorem 3 does not hold.*

To see this, consider $\varphi = card(\_^*.year, (quarter.project, \{\_.eid.S\})) \leq 1$ and $\Sigma = \{\sigma_1, \sigma_2\}$, where $\sigma_1 = card(\_.year, (quarter, \{project.\_.eid.S\})) \leq 3$, and $\sigma_2 = card(\_^*.year.quarter, (project, \{\_.eid.S\})) \leq 2$. A simple replacement of "$\_^*$" by ℓ_0 results in the cardinality network shown on the third picture in Figure 4. But then by Lemma 3, Σ would imply φ, which is clearly incorrect as shown by the counter-example tree on the fourth picture in Figure 4. ▮

7 Soft Max-Constraints

Integrity checking is an important task where database constraints are often used. To ensure data integrity it is common to enforce integrity constraints whenever the data kept in the database is modified. Integrity checking itself can benefit a lot from the ability to decide implication efficiently. Clearly, if Σ implies φ and we have already checked that an XML data tree satisfies Σ then there is no need to test φ anymore.

When treating database constraints as integrity constraints then any violation indicates that the current state of the database is faulty, that is, does not reflect any plausible state of the underlying universe of discourse and should therefore not occur during the runtime of the database.

In addition, however, there are database constraints that describe what is desirable or normally satisfied, but violations may occur. These properties do not yield integrity constraints. Nevertheless, they represent valuable information that should be documented at design time of the database. Database constraints that express desirable properties of the data are sometimes called deontic constraints or soft constraints. They may be used for example to describe ideal states of the underlying universe of discourse or preferences.

Data integration is an important area where database constraints are frequently treated as soft constraints rather than integrity constraints. To provide a concise customer service [18] it is useful to integrate not only the data itself but also the knowledge that we have about the data, such as database constraints. However, meta data from different origins often exhibits different quality levels, e.g., in terms of accuracy and reliability.

Example 12 *Recall the XML database for bank transactions. The max-constraint*

$$\sigma_1 = card(\varepsilon, (\_^*.account, \{transaction.\_.date.S\})) \leq 100$$

states that in any given day, an individual account can be involved in at most 100 *transactions. This might be understood as a hard integrity constraints that should be enforced. Alternatively, it might be understood as a soft constraint that describes normal behavior, but violations may occur.*

When treating database constraints as soft constraints one might still expect the constraint to be satisfied in some sense. As an example we will interpret normal behavior as average behavior. For cardinality constraints this interpretation is evident for tasks such as cardinality estimates for query optimization or resource planning.

Consider a max-constraint $\varphi = card(Q, (Q', \{Q_1, \ldots, Q_k\})) \leq \max$, a context node $q \in [\![Q]\!]$ and a target node $q_0' \in q[\![Q']\!]$. As usual we are interested in the maximum number of target nodes q' that share with q_0' the same information on their field paths. Given an XML tree T we set $f_T^{\varphi}(q, q_0')$ as the maximum of $\sharp\{q' \in q[\![Q']\!] \mid \exists y_1, \ldots, y_k.\forall i = 1, \ldots, k.\, y_i \in q'[\![Q_i]\!] \wedge x_i =_v y_i\}$ where $x_1, \ldots, x_k$ ranges through all $x_i \in q_0'[\![Q_i]\!]$ (with $i = 1, \ldots, k$). The max-constraint φ states that $f_T^{\varphi}(q, q_0') \leq max_{\varphi}$ for all choices of q and q_0'.

By Definition 2, T satisfies φ as a hard integrity constraint if and only if $f_T^{\varphi}(q, q_0') \leq max_{\varphi}$ hold for all choices of q and q_0'. Alternatively, T satisfies φ *as a soft constraint* if

$$\frac{1}{\sharp\, q[\![Q']\!]} \sum_{q_0' \in q[Q']} f_T^{\varphi}(q, q_0') \leq \max{}_{\varphi}$$

holds for every context node $q \in [\![Q]\!]$. In our bank example above, the max-constraint $\sigma_1 = card(\varepsilon, (\_^*.\text{account}, \{\text{transaction}.\_.\text{date.S}\})) \leq 100$ would be satisfied as a soft constraint if for every account the average daily workload is at most 100 transaction.

It is not hard to see that the inference rules given in Table 1 are also sound for the implication of max-constraints as soft constraints.

Theorem 5 *The set $\mathfrak{R}$ of inference rules in Table 1 is complete for the implication of max-constraints in $\mathfrak{M}^{ext}$ as soft constraints.*

Proof Let $\Sigma \cup \{\varphi\}$ be a finite set of max-constraints from $\mathfrak{M}^{ext}$ such that $\varphi \notin \Sigma^+_{\mathfrak{R}}$. In the proof of Theorem 2 we have constructed an XML tree T that satisfies all max-constraints in Σ (as hard constraints), but violates φ (as a hard constraint). By definition, T also satisfies all max-constraints in Σ as soft constraints. By our construction T has a single context node q_φ and $max_\varphi + 1$ target nodes that all share the same information on their field paths. Hence T violates φ also as a soft constraint.

8 Related Work

Cardinality constraints are one of the most influential contributions conceptual modeling has made to the study of database constraints. They were already present in Chen's seminal paper [10] on conceptual database design. All major languages currently used for conceptual database design (say, in particular, the ER model and its extensions as well as UML and ORM) come with means for specifying cardinality constraints. Cardinality constraints have been extensively studied in database design [6, 9, 12, 19, 34, 35, 41, 46, 49, 52]. For a recent survey, see [53].

Cardinality estimation has attracted considerable attention in the XML community as a means for predicting the size of query results. Most research has focussed on XPath queries. Recent work includes [17, 37, 45, 48, 47, 50, 57]. Cardinality estimates use statistical or combinatorial summaries that keep track of the number of data items stored in an XML database. While cardinality constraints often reflect semantic information gathered from the universe of discourse at design time, cardinality estimation monitors the behavior of the database at run time. Both tasks are complementary, but have some commonalities such as the efficient use of regular path languages.

There has been some effort to use cardinality constraints during transformations of conceptual models to XML [16, 38, 43, 44]. This work, however, does not study the expressiveness and tractability of cardinality constraints on XML data. In [38] it is shown that occurrence constraints and foreign keys together cannot express cardinality constraints defined over n-ary relationship types. The cardinality constraints studied in this article, however, suffice to express cardinality constraints over n-ary relationship types.

The current work is an extended version of the conference paper [14]. The extensions are manifold. They include several examples that illustrate and motivate our concepts. Proofs formally validate our results, and provide insight into the techniques developed. The interpretation of cardinality constraints as soft constraints opens up their application to real-world database practice, where exceptions to common rules frequently occur.

Moreover, the class $\mathfrak{M}^{ext}$ of max-constraints studied in this article is even more expressive than the class of cardinality constraints presented in the conference paper [14], without additional penalties on the worst-case efficiency of deciding implication. The results presented here generalize most of our own previous work on this subject [13, 15, 23, 24, 25, 26, 27]. In particular, the class $\mathfrak{M}^{ext}$ of cardinality constraints subsumes all the tractable classes of key and numerical keys explored in our previous work.

9 Conclusion

Cardinality constraints are naturally exhibited by XML data since they represent restrictions that occur in everyday life. They cover XML keys where the upper bound is fixed to 1, occurrence constraints from XML Schema, and also generalized participation constraints as known from database design and conceptual modeling. XML applications such as consistency management, data integration, query optimization, view maintenance and cardinality estimation can therefore benefit from the specification of cardinality constraints.

We have proposed the class of max-constraints that is sufficiently flexible to advance XML data processing. The flexibility results from the right balance between expressiveness 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 cardinality constraints that can be utilized effectively by data engineers. Indeed, the complexity of its associated implication problem indicates that it can be maintained efficiently by database systems for XML applications. Finally, the ability of our class to be exploited as soft constraints makes it particularly interesting in practice where exceptions to the common rule occur frequently.

Future work in this area can go into various directions. XML practice may well warrant the study of other classes of cardinality constraints that require different paradigms to select and compare nodes, or specify restrictions. It would be interesting to investigate the interaction of cardinality constraints with schema specification languages and other classes of database constraints, including functional, multivalued and inclusion dependencies [5, 21, 22, 28, 29, 30, 32, 33, 36, 39, 54, 55]. The broad areas in which cardinality constraints can be applied, as indicated in several parts of this article, warrant further studies. It is also desirable to include cardinality constraints as first-class citizens in mainstream database design tools.

10 Acknowledgement

This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand, and a research grant of the Alfried Krupp von Bohlen und Halbach foundation, administered by the German Scholars organization.

References

[1] S. Abiteboul, R. Hull, V. Vianu, *Foundations of databases*, Addison-Wesley, 1995.

[2] V. Apparao et al., Document Object Model (DOM) Level 1 Specification, W3C Recommendation, Oct. 1998. `http://www.w3.org/TR/REC-DOM-Level-1-19981001/`.

[3] M. Arenas, W. Fan, L. Libkin, What's hard about XML schema constraints?, in: *DEXA*, volume 2453 of Lecture Notes in Computer Science, Springer, 2002, pp 269–278.

[4] M. Arenas, W. Fan, L. Libkin, On the complexity of verifying consistency of XML specifications, *SIAM J. Comput.* 38(3) (2008) 841–880.

[5] M. Arenas, L. Libkin, An information-theoretic approach to normal forms for relational and XML data, *J. ACM* 52(2) (2005) 246–283.

[6] A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, M. Zakharyaschev, Reasoning over extended ER models, in: *ER*, volume 4801 of Lecture Notes in Computer Science, Springer, 2007, pp. 277–292.

[7] T. Bray et al., Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation, Feb. 2004. `http://www.w3.org/TR/2004/REC-xml-20040204/`.

[8] P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, Keys for XML, *Computer Networks* 39(5) (2002), 473–487.

[9] D. Calvanese, M. Lenzerini, On the interaction between ISA and cardinality constraints, in: *ICDE*, IEEE Computer Society, 1994, pp. 204–213.

[10] P. P. Chen, The entity-relationship model: towards a unified view of data. *ACM Trans. Database Systems* 1 (1976), 9–36.

[11] J. Clark, S. DeRose, XML path language (XPath) version 1.0, W3C Recommendation, 1999. `http://www.w3.org/TR/REC-xpath-19991116/`.

[12] F. Currim, N. Neidig, A. Kampoowale, G. Mhatre, The CARD system, in: *ER*, volume 6412 of Lecture Notes in Computer Science, Springer, 2010, pp. 433–437.

[13] F. Ferrarotti, S. Hartmann, S. Link, J. Wang, Promoting the semantic capability of XML keys, in: *XSym*, volume 6309 of Lecture Notes in Computer Science, Springer, 2010, pp. 144–153.

[14] F. Ferrarotti, S. Hartmann, S. Link, A precious class of cardinality constraints for flexible XML data processing, in: *ER*, volume 6998 of Lecture Notes in Computer Science, Springer, 2011, pp. 175–188.

[15] F. Ferrarotti, S. Hartmann, S. Link, M. Marín, E. Muñoz, Performance Analysis of Algorithms to Reason about XML Keys, in: *DEXA*, volume 7446 of Lecture Notes in Computer Science, Springer, 2012, pp. 101–115.

[16] M. Franceschet, D. Gubiani, A. Montanari, C. Piazza, From entity relationship to XML schema: A graph-theoretic approach, in: *Database and XML Technologies*, volume 5679 of Lecture Notes in Computer Science, Springer, 2009, pp. 165–179.

[17] J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, J. Siméon, StatiX: Making XML count, in: *SIGMOD*, ACM, 2002, pp. 181–191.

[18] A. Fuxman, R. Miller, Towards inconsistency management in data integration systems, in: *IIWeb*, 2003, pp. 143–148.

[19] S. Hartmann, On the implication problem for cardinality constraints and functional dependencies, *Ann. Math. Art. Intell.* 33 (2001) 253–307.

[20] S. Hartmann, M. Kirchberg, S. Link, A subgraph-based approach towards functional dependencies for XML, in: *SCI*, volume IX of Computer Science and Engineering II, International Institute of Informatics and Systemics, 2003, pp 200–2005.

[21] S. Hartmann, M. Kirchberg, S. Link, Design by example for SQL table definitions with functional dependencies, *VLDB J.* 21(1) (2012) 121–144.

[22] S. Hartmann, H. Köhler, S. Link, B. Thalheim, Armstrong databases and reasoning for functional dependencies and cardinality constraints over partial bags, in: *FoIKS*, volume 7153 of Lecture Notes in Computer Science, Springer, 2012, pp 164–183.

[23] S. Hartmann, S. Link, Unlocking Keys for XML Trees, in: *ICDT*, volume 4353 of Lecture Notes in Computer Science, Springer, 2007, pp 104–118.

[24] S. Hartmann, S. Link, Numerical Constraints for XML, in: *WoLLIC*, volume 4576 of Lecture Notes in Computer Science, Springer, 2007, pp 203–217.

[25] S. Hartmann, S. Link, Efficient reasoning about a robust XML key fragment, *ACM Trans. Database Syst.* 34(2) (2009) Article 10.

[26] S. Hartmann, S. Link, Expressive, yet tractable XML keys, in: *EDBT*, volume 360 of the ACM conference proceedings series, ACM, 2009, pp. 357–367.

[27] S. Hartmann, S. Link, Numerical constraints on XML data, *Inf. Comput.* 208(5) (2010) 521–544.

[28] S. Hartmann, S. Link, The implication problem of data dependencies over SQL table definitions: Axiomatic, algorithmic and logical characterizations, *ACM Trans. Database Syst.* 37(2) (2012) 13.

[29] S. Hartmann, S. Link, T. Trinh, Constraint acquisition for entity-relationship models, *Data Knowl. Eng.* 68(10) (2009) 1128–1155.

[30] S. Hartmann, S. Link, T. Trinh, Solving the implication problem for XML functional dependencies with properties, in *WoLLIC*, volume 6188 of Lecture Notes in Computer Science, Springer, 2010, pp. 161–175.

[31] D. Jungnickel, *Graphs, networks and algorithms*, Springer, 1999.

[32] M. Karlinger, M.W. Vincent, M. Schrefl, Inclusion dependencies in XML: Extending relational semantics, in: *DEXA*, volume 5690 of Lecture Notes in Computer Science, Springer, 2009, pp. 23–37.

[33] M. Karlinger, M.W. Vincent, M. Schrefl, Keys in XML: Capturing identification and uniqueness, in: *WISE*, volume 5802 of Lecture Notes in Computer Science, Springer, 2009, pp. 563–571.

[34] M. Lenzerini, P. Nobili, On the satisfiability of dependency constraints in entity-relationship schemata, *Information Science* 15 (1990) 453–461.

[35] S. Liddle, D. Embley, S. Woodfield, Cardinality constraints in semantic data models, *Data Knowl. Eng.* 11(3) (1993) 235–270.

[36] W. Langeveldt, S. Link, Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies, *Inf. Syst.* 35(3) (2010) 352–374.

[37] H. Li, M. L. Lee, W. Hsu, G. Cong, An estimation system for XPath expressions, in: *ICDE*, IEEE Computer Society, 2006, pp. 54.

[38] S. Link, T. Trinh, Know your limits: Enhanced XML modeling with cardinality constraints, in: *ER tutorials*, volume 83 of *CRPIT*, Australian Computer Society, 2007, pp. 19–30.

[39] S. Link, Characterisations of multivalued dependency implication over undetermined universes, *J. Comput. Syst. Sci.* 78(4) (2012) 1026-1044.

[40] C. Luo, Z. Jiang, W. C. Hou, F. Yu, Q. Zhu, A sampling approach for XML query selectivity estimation, in: *EDBT*, ACM, 2009, pp. 335–344.

[41] A. McAllister, Complete rules for n-ary relationship cardinality constraints, *Data Knowl. Eng.* 27 (1998) 255–288.

[42] G. Miklau, D. Suciu, Containment and equivalence for a fragment of XPath, *J. ACM*, 51(1) (2004) 2–45.

[43] M. Necaský, XSEM: A conceptual model for XML, in: *APCCM*, volume 67 of *CRPIT*, Australian Computer Society, 2007, pp. 37–48.

[44] M. Necaský, I. Mlýnková, J. Klímek, J. Malý, When conceptual model meets grammar: A dual approach to XML data modeling, *Data Knowl. Eng.* 72 (2012) 1–30.

[45] S. Obermeier, S. Boettcher, T. Wycisk, XPath selectivity estimation for a mobile auction application, in: *IDEAS*, IEEE Computer Society, 2007, pp. 251–262.

[46] A. Queralt, A. Artale, D. Calvanese, E. Teniente, OCL-lite: Finite reasoning on UML/OCL conceptual schemas, *Data Knowl. Eng.* 73 (2012) 1–22.

[47] C. Sartiani, A general framework for estimating XML query cardinality, in: *Database Programming Languages*, volume 2921 of Lecture Notes in Computer Science, Springer, 2004, pp. 117–118.

[48] S. Sakr, XSelMark: A micro-benchmark for selectivity estimation approaches of XML queries, in: *DEXA*, volume 5181 of Lecture Notes in Computer Science, Springer, 2008, pp. 735–744.

[49] I.-Y. Song, M. Evans, E. Park, A comprehensive analysis of entity-relationship diagrams, *J. Computer Software Eng.* 3 (1995) 427–459.

[50] J. Teubner, T. Grust, S. Maneth, S. Sakr, Dependable cardinality forecasts for XQuery, *Proc. VLDB* 1 (2008) 463–477.

[51] B. Thalheim, Fundamentals of cardinality constraints, in: *ER*, volume 645 of Lecture Notes in Computer Science, Springer, 1992, pp. 7–23.

[52] B. Thalheim, *Foundations of entity-relationship modeling*, Springer, 2000.

[53] B. Thalheim, Integrity constraints in (conceptual) database models. in: *The Evolution of Conceptual Modeling*, volume 6520 of Lecture Notes in Computer Science, Springer, pp. 42–67.

[54] M.W. Vincent, J. Liu, C. Liu, Strong functional dependencies and their application to normal forms in XML, *ACM Trans. Database Syst.* 29(3) (2004) 445–462.

[55] M.W. Vincent, J. Liu, M.K. Mohania, The implication problem for 'closest node' functional dependencies in complete XML documents, *J. Comput. Syst. Sci.* 78(4) (2012) 1045–1098.

[56] H. Thompson, D. Beech, M. Maloney, N. Mendelsohn, XML Schema part 1: Structures second edition, W3C Recommendation, `http://www.w3.org/TR/REC-xmlschema-1-20041028/`.

[57] N. Zhang, M. Ozsu, A. Aboulnaga, I. Ilyas, XSEED: Accurate and fast cardinality estimation for XPath queries, in: *ICDE*, IEEE Computer Society, pp. 61.

A Verifying our Assumptions in the Proof of Lemma 3

We show next that the three assumptions used to prove Lemma 3 are indeed correct.

Remark 6 *The proofs in this section are rather technical. This is partly due to the extension of our study from the class $\mathfrak{M}$ used in the conference version [14] to the more expressive class $\mathfrak{M}^{ext}$. If we assume that variable-length wildcards never occur in field paths, then the proofs can be shortened considerably.*

The following result provides evidence that our first assumption holds.

Lemma 4 *Let $\Sigma \cup \{\varphi\}$ be a finite set of max-constraints in the class $\mathfrak{M}^{ext}$, and let $\sigma \in \Sigma^+_{\mathfrak{R}}$ be applicable to φ. Suppose that w_σ and w'_σ witness the applicability of σ to φ. Suppose further that* $\text{prefix}(Q'_\sigma)$ *and* $\text{suffix}(Q'_\sigma)$ *are $PL^{\{\cdot,\_,\_^*\}}$ expressions such that $Q'_\sigma \sim \text{prefix}(Q'_\sigma).\text{suffix}(Q'_\sigma)$, $q'_\varphi \in w_\sigma[\![\text{prefix}(Q'_\sigma)]\!]$ and $w'_\sigma \in q'_\varphi[\![\text{suffix}(Q'_\sigma)]\!]$. Then there is a $PL^{\{\cdot,\_,\_^*\}}$ expression $F \subseteq \text{prefix}(Q'_\sigma)$ such that $card(Q_\sigma, (F, \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_\sigma \in \Sigma^+_{\mathfrak{R}}$.*

Proof First we note that the case where $Q'_\sigma \sim \varepsilon$ is trivial because of the *epsilon* and the *weakening* rules. In the following we distinguish two different cases.

Case 1. Suppose that all field paths of φ are different from ε, i.e., for all $i = 1, \ldots, k_\varphi$ we have $Q^\varphi_i \not\sim \varepsilon$. Then, just the leaves of $T_{\Sigma,\varphi}$ are marked. We discuss two subcases.

Case 1.a) Suppose that $\text{suffix}(Q'_\sigma) \sim \varepsilon$. That is, $\text{prefix}(Q'_\sigma) \sim Q'_\sigma$ and $w'_\sigma = q'_\varphi$. Since w_σ and w'_σ witness the applicability of σ to φ and only the leaves of $T_{\Sigma,\varphi}$ are marked we know that for all $j = 1, \ldots, k_\sigma$ there is some i_j with $1 \leq i_j \leq k_\varphi$ such that $O^\varphi_{i_j} \lesssim Q^\sigma_j$. Thus, by construction of $O^\varphi_{i_j}$ from $Q^\varphi_{i_j}$, it follows that $Q^\varphi_{i_j} \lesssim Q^\sigma_j$, which implies that $Q^\varphi_{i_j} \subseteq Q^\sigma_j$. Choosing F properly with $F \lesssim Q'_{\sigma'}$, we therefore obtain the following inference:

$$\cfrac{\cfrac{\cfrac{card(Q_\sigma, (Q'_\sigma, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}{card(Q_\sigma, (F, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{Q^\varphi_{i_1}, \ldots, Q^\varphi_{i_{k_\sigma}}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_\sigma}$$

in which we first apply *target-path-containment* rule, followed by k_σ applications of the *field-path containment* rule, and then the *superfield* rule.

Case 1.b) Suppose that $\text{suffix}(Q'_\sigma) \not\sim \varepsilon$.

Since $w'_\sigma \in q'_\varphi[\![\text{suffix}(Q'_\sigma)]\!]$ it follows from the definition of the mini-tree $T_{\Sigma,\varphi}$ and the applicability of σ to φ that, for some $1 \leq i \leq k_\varphi$, it holds that $O_\varphi.O'_\varphi.O^\varphi_i \lesssim Q_\sigma.Q'_\sigma.Q^\sigma_j$ for every $j = 1, \ldots k_\sigma$. Since there is *no* sequence of more than l consecutive single-label wildcards in σ, we have that $Q_\varphi.Q'_\varphi.Q^\varphi_i \lesssim Q_\sigma.Q'_\sigma.Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.

We next divide each of Q'_σ, Q'_φ and Q^φ_i into prefix and suffix so that $q'_\varphi \in [\![Q_\sigma.\text{prefix}(Q'_\sigma)]\!]$, $w_\sigma \in [\![Q_\varphi.\text{prefix}(Q'_\varphi)]\!]$ and $w'_\sigma \in [\![Q_\varphi.Q'_\varphi.\text{prefix}(Q^\varphi_i)]\!]$. As before it still holds that

$$Q_\varphi.\text{prefix}(Q'_\varphi).\text{suffix}(Q'_\varphi).\text{prefix}(Q^\varphi_i).\text{suffix}(Q^\varphi_i) \lesssim$$
$$Q_\sigma.\text{prefix}(Q'_\sigma).\text{suffix}(Q'_\sigma).Q^\sigma_j \text{ for every } j = 1, \ldots, k_\sigma.$$

From this we obtain the following statements:

1. $Q_\varphi.Q'_\varphi \lesssim Q_\sigma.\text{prefix}(Q'_\sigma)$.
2. $Q^\varphi_i \lesssim \text{suffix}(Q'_\sigma).Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.
3. $\text{suffix}(Q'_\varphi).\text{prefix}(Q^\varphi_i) \lesssim Q'_\sigma$.
4. $\text{suffix}(Q^\varphi_i) \lesssim Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.
5. $Q^\varphi_i \lesssim \text{suffix}(Q'_\sigma).\text{suffix}(Q^\varphi_i)$.

Furthermore, there is a max-constraint $\sigma' \in \Sigma^+_{\mathfrak{N}'}$ with $Q_{\sigma'} \sim Q_\sigma$, prefix$(Q'_{\sigma'}) \subseteq$ prefix(Q'_σ), suffix$(Q'_{\sigma'}) \subseteq$ suffix(Q'_σ) and $Q^{\sigma'}_j \sim Q^\sigma_j$ for all $j = 1, \ldots, k_\sigma$, such that σ' is applicable to φ, w_σ coincides with $w_{\sigma'}$, w'_σ coincides with $w'_{\sigma'}$ and the following conditions are met:

a. If a variable-length wildcard appears in suffix(Q^φ_i), then it does not appear in prefix$(Q'_{\sigma'})$.suffix$(Q'_{\sigma'})$.

b. If a variable-length wildcard appears in prefix(Q'_σ).suffix(Q'_σ), then a variable-length wildcard appears in prefix$(Q'_{\sigma'})$ or suffix$(Q'_{\sigma'})$, but not in both.

Note that if a variable-length wildcard appears in suffix(Q^φ_i), then, by statement (4.) above, it also appears in Q^σ_j for every $j = 1, \ldots, k_\sigma$. Since $\sigma \in \mathfrak{M}^{ext}$, we conclude that the variable-length wildcard does not appear in prefix(Q'_σ).suffix(Q'_σ). This shows that conditions (a.) and (b.) are both satisfied for prefix$(Q'_{\sigma'}) \sim$ prefix(Q'_σ) and suffix$(Q'_{\sigma'}) \sim$ suffix(Q'_σ).

On the other hand, suppose that the variable-length wildcard does not appears in suffix(Q^φ_i). If the variable-length wildcard does not appears in both prefix(Q'_σ) and suffix(Q'_σ), then again the conditions (a) and (b) are both satisfied for prefix$(Q'_{\sigma'}) \sim$ prefix(Q'_σ) and suffix$(Q'_{\sigma'}) \sim$ suffix(Q'_σ). Otherwise, we consider two cases. In the first case we assume that the variable-length wildcard appears in prefix(Q^φ_i). Since $\varphi \in \mathfrak{M}^{ext}$, it follows that the variable-length wildcard is not in Q'_φ. But then, since suffix$(Q'_\varphi) \subseteq$ prefix(Q'_σ), a suitable prefix$(Q'_{\sigma'})$ and suffix$(Q'_{\sigma'})$ can be obtained by replacing each variable-length wildcard in prefix(Q'_σ) by an appropriate sequence of single-label wildcards of length $\leq l$ and setting suffix$(Q'_{\sigma'}) \sim$ suffix(Q'_σ). In the second case, we assume that the variable-length wildcard does not appear in prefix(Q^φ_i). Since prefix$(Q^\varphi_i) \subseteq$ suffix(Q'_σ), a suitable prefix$(Q'_{\sigma'})$ and suffix$(Q'_{\sigma'})$ can be obtained by replacing each variable-length wildcard in suffix(Q'_σ) by an appropriate sequence of single-label wildcards of length $\leq l$ and setting prefix$(Q'_{\sigma'}) \sim$ prefix(Q'_σ).

Choosing F properly with $F \lesssim$ prefix$(Q'_{\sigma'})$, we obtain the following inference:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{card(Q_\sigma, (Q'_\sigma, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{\text{suffix}(Q^\varphi_i)\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\text{suffix}(Q'_{\sigma'}).\text{suffix}(Q^\varphi_i)\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{Q^\varphi_i\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_\sigma}$$

in which we first apply the *target-path-containment* rule, followed by k_σ applications of the *field-path containment* rule, then the *subnodes* rule, then again the *field-path containment* rule and finally the *superfield* rule. Note that (a) and (b) ensure that the conditions for the application of the *subnodes* rule are met.

Case 2. Suppose now that there is a field path of φ which is ε, say $Q^\varphi_l \sim \varepsilon$ for some l with $1 \leq l \leq k_\varphi$. Then, all descendant nodes of q'_φ in $T_{\Sigma,\varphi}$ are marked. We consider two subcases.

Case 2.a) Suppose that $Q \sim \varepsilon$. Hence, $w'_\sigma = q'_\varphi$. Since w_σ and w'_σ witness the applicability of σ to φ we know that for all $j = 1, \ldots, k_\sigma$ there is some Q'_j and some

i_j with $1 \leq i_j \leq k_\varphi$ such that $O^\varphi_{i_j} \lesssim Q^\sigma_j.Q'_j$. Thus, by construction of $O^\varphi_{i_j}$ from $Q^\varphi_{i_j}$, it follows that $Q^\varphi_{i_j} \lesssim Q^\sigma_j.Q'_j$, which implies that $Q^\varphi_{i_j} \subseteq Q^\sigma_j.P'_j$. Choosing F properly with $F \lesssim Q'_\sigma$, we therefore obtain the following inference:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{card(Q_\sigma, (Q'_\sigma, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}{card(Q_\sigma, (F, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\varepsilon, Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\varepsilon, Q^\sigma_1.Q'_1, \ldots, Q^\sigma_{k_\sigma}.Q'_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\varepsilon, Q^\varphi_{i_1}, \ldots, Q^\varphi_{i_{k_\sigma}}\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \underbrace{\{\varepsilon, Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\}}_{=\{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\}})) \leq \max_\sigma}$$

in which we first apply the *target-path-containment* rule then the *superfield* rule to introduce ε, followed by k_σ applications of the *prefix-epsilon* rule and k_σ applications of the *field-path containment*, and again the *superfield* rule.

Case 2.b) Suppose that $Q \not\sim \varepsilon$. Since $w'_\sigma \in q'_\varphi[\![\text{suffix}(Q'_\sigma)]\!]$ it follows from the definition of the mini-tree $T_{\Sigma,\varphi}$ and the applicability of σ to φ that, for some $1 \leq i \leq k_\varphi$, there is a prefix(O^φ_i) such that prefix$(O^\varphi_i) \lesssim$ suffix(Q'_σ). Let $O^\varphi_i \sim$ prefix(O^φ_i).suffix(O^φ_i). Due to the definition of applicability and the fact that all descendant nodes of q'_φ in $T_{\Sigma,\varphi}$ are marked, it follows that, for every $j = 1, \ldots, k_\sigma$, there is a prefix O_j of suffix(O^φ_i) such that $O_j \lesssim Q^\sigma_j$, and also that $O_\varphi.O'_\varphi$.prefix$(O^\varphi_i).O_j \lesssim Q_\sigma.Q'_\sigma.Q^\sigma_j$. Let $Q^\varphi_i \sim$ prefix(Q^φ_i).suffix(Q^φ_i) with $w'_\sigma \in [\![Q_\varphi.Q'_\varphi.\text{prefix}(Q^\varphi_i)]\!]$. Since there is *no* sequence of single-label wildcards in σ of length $> l$ and for every $j = 1, \ldots, k_\sigma$ there is a prefix Q_j of suffix(Q^φ_i) such that $O_j \lesssim Q_j$, we have that $Q_\varphi.Q'_\varphi$.prefix$(Q^\varphi_i).Q_j \lesssim Q_\sigma.Q'_\sigma.Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.

We next divide Q'_σ, Q'_φ and Q^φ_i into prefix and suffix so that $q'_\varphi \in [\![Q_\sigma.\text{prefix}(Q'_\sigma)]\!]$, $w_\sigma \in [\![Q_\varphi.\text{prefix}(Q'_\varphi)]\!]$ and $w'_\sigma \in [\![Q_\varphi.Q'_\varphi.\text{prefix}(Q^\varphi_i)]\!]$. As before we still have Q_φ.prefix(Q'_φ).suffix(Q'_φ).prefix$(Q^\varphi_i).Q_j \lesssim Q_\sigma$.prefix$(Q'_\sigma)$.suffix$(Q'_\sigma).Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$,

The following statements hold.

1. $Q_\varphi.Q'_\varphi \lesssim Q_\sigma$.prefix$(Q'_\sigma)$.
2. prefix$(Q^\varphi_i).Q_j \lesssim$ suffix$(Q'_\sigma).Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.
3. suffix(Q'_φ).prefix$(Q^\varphi_i) \lesssim Q'_\sigma$.
4. $Q_j \lesssim Q^\sigma_j$ for every $j = 1, \ldots, k_\sigma$.

It also holds that there is a max-constraint $\sigma' \in \Sigma^+_{\mathfrak{R}'}$ with $Q_{\sigma'} \sim Q_\sigma$, prefix$(Q'_{\sigma'}) \subseteq$ prefix(Q'_σ), suffix$(Q'_{\sigma'}) \subseteq$ suffix(Q'_σ) and $Q^{\sigma'}_j = Q^\sigma_j$ for all $j = 1, \ldots, k_\sigma$, such that σ' is applicable to φ, w_σ coincides $w_{\sigma'}$, w'_σ coincides $w'_{\sigma'}$ and the following conditions are met:

a. If there is a Q_j for $j = 1, \ldots, k_\sigma$ such that the variable-length wildcard appears in Q_j, then it does not appear in prefix$(Q'_{\sigma'})$.suffix$(Q'_{\sigma'})$.

b. If the variable-length wildcard appears in prefix(Q'_σ).suffix(Q'_σ), then the variable-length wildcard either appears in prefix($Q'_{\sigma'}$) or suffix($Q'_{\sigma'}$), but not in both.

c. If the variable-length wildcard appears in suffix(Q^φ_i), then it does not appear in prefix($Q'_{\sigma'}$).

Note that if there is a Q_j for $j = 1, \ldots, k_\sigma$ (recall that Q_j is a prefix of suffix(Q^φ_i)) such that the variable-length wildcard appears in Q_j, then by Equation (4), it also appears in Q^σ_j. Since $\sigma \in \mathfrak{M}^{ext}$, we get that the variable-length wildcard does not appear in prefix(Q'_σ).suffix(Q'_σ). This shows that conditions (a), (b) and (c) are satisfied for prefix($Q'_{\sigma'}$) $\sim$ prefix(Q'_σ) and suffix($Q'_{\sigma'}$) $\sim$ suffix(Q'_σ).

On the other hand, suppose that for every $j = 1, \ldots, k_\sigma$ the variable-length wildcard does not appears in Q_j. If the variable-length wildcard does not appears in both prefix(Q'_σ) and suffix(Q'_σ), then again the conditions (a), (b) and (c) are satisfied for prefix($Q'_{\sigma'}$) $\sim$ prefix(Q'_σ) and suffix($Q'_{\sigma'}$) $\sim$ suffix(Q'_σ). Otherwise, we consider two cases. In the first case we assume that the variable-length wildcard appears in prefix(Q^φ_i). Since $\varphi \in \mathfrak{M}^{ext}$, it follows that the variable-length wildcard is not in Q'_φ. But then, since suffix(Q'_φ) $\subseteq$ prefix(Q'_σ), a suitable prefix($Q'_{\sigma'}$) and suffix($Q'_{\sigma'}$) can be obtained by replacing each variable-length wildcard in prefix(Q'_σ) by an appropriate sequence of single-label wildcards of length $\leq l$ and setting suffix($Q'_{\sigma'}$) $\sim$ suffix(Q'_σ). In the second case, we assume that the variable-length wildcard does not appear in prefix(Q^φ_i). Since prefix(Q^φ_i) $\subseteq$ suffix(Q'_σ), a suitable suffix($Q'_{\sigma'}$) can be obtained by replacing each variable-length wildcard in suffix(Q'_σ) by an appropriate sequence of single-label wildcards of length $\leq l$. Regarding prefix($Q'_{\sigma'}$), if the variable-length wildcard appears in suffix(Q^φ_i), then it does not appear in suffix(Q'_φ) and thus a suitable prefix($Q'_{\sigma'}$) can be obtained by replacing each variable-length wildcard in prefix(Q'_σ) by an appropriate sequence of single-label wildcards of length $\leq l$. Otherwise, we just set prefix($Q'_{\sigma'}$) $\sim$ prefix(Q'_σ).

Note that, due to the definition of applicability and the fact that every Q_j is a prefix of suffix(Q^φ_i), there is some m with $1 \leq m \leq k_\sigma$ such that for all $j = 1, \ldots, k_\sigma$, it holds that Q_j is a prefix of Q_m. We choose F properly with $F \lesssim$ prefix($Q'_{\sigma'}$). For suffix(Q^φ_m) $\sim Q_m.Q'_m$, we get the following inference:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{card(Q_\sigma, (Q'_\sigma, \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{\epsilon, Q^\sigma_1, \ldots, Q^\sigma_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{\epsilon, Q_1, \ldots, Q_{k_\sigma}\})) \leq \max_\sigma}}{card(Q_\sigma, (F.\text{suffix}(Q'_{\sigma'}), \{\epsilon, Q_m\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\epsilon, \text{suffix}(Q'_{\sigma'}).Q_m\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \{\epsilon, \text{suffix}(Q'_{\sigma'}).Q_m.Q'_m\})) \leq \max_\sigma}}{card(Q_\sigma, (F, \underbrace{\{\varepsilon, Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\}}_{=\{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\}})) \leq \max_\sigma}$$

in which we first apply the *target-path-containment* rule, then the *superfield* rule to introduce ε, followed by k_σ applications of the *field-path containment* rule, further k_σ applications of the *field-path containment* rule, then the *subnodes-epsilon* rule, followed by the *prefix-epsilon* rule, and finally the *superfield* rule.

The previous lemma establishes the correctness of the first assumption in the proof of Lemma 3. The second assumption in the proof of Lemma 3 holds as we can see immediately from applying the *target-to-context* rule. Finally, to verify the third assumption we prove the following lemma.

Lemma 5 *Let $\Sigma \cup \{\varphi\}$ be a finite set of max-constraints in the class $\mathfrak{M}^{ext}$, and let $\sigma_1, \sigma_2 \in \Sigma^+_{\mathfrak{R}'}$ where $\sigma_2 = card(Q_{\sigma_2}, (Q'_{\sigma_2}, \{Q^\varphi_1, \dots, Q^\varphi_{k_\varphi}\})) \leq \max_{\sigma_2}$. Suppose that $(w_{\sigma_1}, w'_{\sigma_1})$ witnesses the applicability of σ_1 to φ where w_{σ_1} either coincides with q_φ or is an descendant node of it in $T_{\Sigma,\varphi}$, and $(w'_{\sigma_1}, q'_\varphi)$ witnesses the applicability of σ_2 to φ. Then there is a $PL^{\{\cdot,\_,\_^*\}}$ expression Q' and a suffix of Q'_φ, denoted* suffix(Q'_φ)*, such that*

$$\sigma' = card(Q_{\sigma_1}, (Q'.\text{suffix}(Q'_\varphi), \{Q^\varphi_1, \dots, Q^\varphi_{k_\varphi}\})) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'}$$

and $(w_{\sigma_1}, q'_\varphi)$ witnesses the applicability of σ' to φ.

Proof Since w'_{σ_1} is an ancestor node of q'_φ and w_{σ_1} and w'_{σ_1} witness the applicability of σ_1 to φ, it follows that there is a suffix suffix(O'_φ) of O'_φ such that suffix$(O'_\varphi) \lesssim Q'_{\sigma_2}$. Since there is *no* sequence of single-label wildcards of length $> l$ in σ_2, we have that suffix$(Q'_\varphi) \lesssim Q'_{\sigma_2}$ for $Q'_\varphi \sim$ prefix(Q'_φ).suffix(Q'_φ). We also have that, $q'_\varphi \in w'_{\sigma_1}[\![\text{suffix}(Q'_\varphi)]\!]$ and, for all $j = 1, \dots, k_{\sigma_1}$, it holds that suffix$(Q'_\varphi)$.suffix$(Q^{\sigma_1}_j) \lesssim Q^{\sigma_1}_j$ for $Q^{\sigma_1}_j \sim$ prefix$(Q^{\sigma_1}_j)$.suffix$(Q^{\sigma_1}_j)$. Thus, we can use k_{σ_1} applications of the *field-path containment* rule to derive the following max-constraint from σ_1.

$$card(Q_{\sigma_1}, (Q'_{\sigma_1}, \{\mathcal{R}\})) \leq \max_{\sigma_1} \in \Sigma^+_{\mathfrak{R}'} \tag{2}$$

where $\mathcal{R} = \{\text{suffix}(Q'_\varphi).\text{suffix}(Q^{\sigma_1}_1), \dots, \text{suffix}(Q'_\varphi).\text{suffix}(Q^{\sigma_1}_{k_{\sigma_1}})\}$. Note that, if there is a variable-length wildcard in suffix(Q'_φ), then there is also a variable-length wildcard in $Q^{\sigma_1}_j$ for every $j = 1, \dots, k_{\sigma_1}$ and thus no variable-length wildcard can appear in Q'_{σ_1}. Therefore, the max-constraint in (2) is also in the class $\mathfrak{M}^{ext}$.

Next, we note that if the variable-length wildcard appears in Q^φ_i for some $i = 1, \dots, k_\varphi$, then it cannot appear in Q'_φ. Otherwise φ would not be in $\mathfrak{M}^{ext}$. Thus, we can replace any variable-length wildcard that appears in Q'_{σ_1} by an appropriate sequence of single-label wildcards of length $\leq l$. If variable-length wildcard appears in Q^φ_i, then we set Q' to be the the expression in $PL^{\{\cdot,\_\}}$ obtained by this process. Otherwise, if the variable-length wildcard does not appear in Q^φ_i, we just set $Q' \sim Q'_{\sigma_1}$. By applying the *target-path containment* rule to (2) we get the following max-constraint which is still applicable to φ.

$$card(Q_{\sigma_1}, (Q', \{\mathcal{R}\})) \leq \max_{\sigma_1} \in \Sigma^+_{\mathfrak{R}'} \tag{3}$$

Then we can apply the *superfield* rule to obtain

$$card(Q_{\sigma_1}, (Q', \mathcal{S})) \leq \max_{\sigma_1} \in \Sigma^+_{\mathfrak{R}'} \tag{4}$$

where $\mathcal{S} = \{\text{suffix}(Q'_\varphi).Q^\varphi_1, \dots, \text{suffix}(Q'_\varphi).Q^\varphi_{k_\varphi}, \text{suffix}(Q'_\varphi).\text{suffix}(Q^{\sigma_1}_1), \dots,$

$\text{suffix}(Q'_\varphi).\text{suffix}(Q^{\sigma_1}_{k_{\sigma_1}})\}$.

According to applicability we have $Q_{\sigma_1}.Q'_{\sigma_1} \subseteq Q_{\sigma_2}$ and since $Q' \subseteq Q'_{\sigma_1}$, we also have that $Q_{\sigma_1}.Q' \subseteq Q_{\sigma_2}$. Recall that $\text{suffix}(Q'_\varphi) \subseteq Q'_{\sigma_2}$. Thus, applications of the *context-path-containment* rule, the *target-path-containment* rule and the *superfield* rule to σ_2 produce

$$card(Q_{\sigma_1}.Q', (\text{suffix}(Q'_\varphi), \mathcal{T})) \leq \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'} \tag{5}$$

where $\mathcal{T} = \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}, \text{suffix}(Q^{\sigma_1}_1), \ldots, \text{suffix}(Q^{\sigma_1}_{k_{\sigma_1}})\}$. Note that if the variable-length wildcard appears in $\text{suffix}(Q'_\varphi)$, then for every $i = 1, \ldots, k_\varphi$, it does not appear in Q^φ_i. So, if that is the case, we assume that for every $j = 1, \ldots, k_{\sigma_1}$, there is no single-label wildcard in $Q^{\sigma_1}_j$, since otherwise we can replace each of those variable-length wildcards by an appropriate sequence of single-label wildcards of length $\leq l$.

From (4) and (5) we infer

$$card(Q_{\sigma_1}, (Q'.\text{suffix}(Q'_\varphi), \mathcal{T})) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'} \tag{6}$$

by means of the *multiplication* rule.

We now distinguish between two different cases.

Case 1. Assume first that for all $i = 1, \ldots, k_\varphi$ we have $Q^\varphi_i \not\sim \varepsilon$. That is, only the leaf nodes of $T_{\Sigma,\varphi}$ are marked. It follows, by applicability of σ_1 to φ, that for all $j = 1, \ldots, k_{\sigma_1}$ there is some i_j with $1 \leq i_j \leq k_\varphi$ such that $O^\varphi_{i_j} \lesssim Q^{\sigma_1}_j$. Thus, by construction of $O^\varphi_{i_j}$ from $Q^\varphi_{i_j}$, it follows that $Q^\varphi_{i_j} \lesssim Q^{\sigma_1}_j$, which implies that $Q^\varphi_{i_j} \subseteq Q^{\sigma_1}_j$. Therefore, by k_σ applications of the *field-path containment* rule, (6) reduces to

$$card(Q_{\sigma_1}, (Q'.\text{suffix}(Q'_\varphi), \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'}.$$

Case 2. For the remaining case we suppose that there is some l with $1 \leq l \leq k_\varphi$ such that $Q^\varphi_l \sim \varepsilon$. That is, all descendant nodes of q'_φ in $T_{\Sigma,\varphi}$ are marked. Applicability of σ_1 to φ means that for all $j = 1, \ldots, k_{\sigma_1}$ there is some i_j with $1 \leq i_j \leq k_\varphi$ such that $O^\varphi_{i_j} \lesssim \text{suffix}(Q^{\sigma_1}_j).\text{suffix}(Q^\varphi_{i_j})$. Recall that $q'_\varphi \in w'_{\sigma_1}[\![\text{suffix}(Q'_\varphi)]\!]$ and $\text{suffix}(Q'_\varphi).\text{suffix}(Q^{\sigma_1}_j) \subseteq Q^{\sigma_1}_j$. Thus, by construction of $O^\varphi_{i_j}$ from $Q^\varphi_{i_j}$, it follows that $Q^\varphi_{i_j} \subseteq \text{suffix}(Q^{\sigma_1}_j).\text{suffix}(Q^\varphi_{i_j})$. A repeated application of the *prefix-epsilon* rule to (6) results in

$$card(Q_{\sigma_1}, (Q'.\text{suffix}(Q'_\varphi), \mathcal{T}')) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'},$$

where $\mathcal{T}' \sim \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}, \text{suffix}(Q^{\sigma_1}_1).\text{suffix}(Q^\varphi_{i_1}), \ldots, \text{suffix}(Q^{\sigma_1}_{k_{\sigma_1}}).\text{suffix}(Q^\varphi_{i_{k_{\sigma_1}}})\}$. Finally, by repeated applications of the *field-path containment* rule, this reduces to

$$card(Q_{\sigma_1}, (Q'.\text{suffix}(Q'_\varphi), \{Q^\varphi_1, \ldots, Q^\varphi_{k_\varphi}\})) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+_{\mathfrak{R}'}.$$

Hence, in both cases we derive that $\sigma' \in \Sigma^+_{\mathfrak{R}'}$. It is immediate that σ' is applicable to φ as witnessed by w_{σ_1} and q'_φ.

This concludes our proof of the assumptions made when verifying Lemma 3.