Numerical constraints on XML data

Sven Hartmann a, Sebastian Link b, *

a Department of Informatics, Clausthal University of Technology, Germany
b School of Information Management, Victoria University of Wellington, New Zealand

ARTICLE INFO

Article history:
Received 23 November 2007
Revised 11 September 2008
Available online 6 December 2009

Keywords:
XML
Numerical constraint
Integrity
Axiomatisation
Implication
Complexity
Shortest path

ABSTRACT

Boundaries occur naturally in everyday life. This paper introduces numerical constraints into the framework of XML to take advantage of the benefits that result from the explicit specification of such boundaries. Roughly speaking, numerical constraints restrict the number of elements in an XML data fragment based on the data values of selected subelements. Efficient reasoning about numerical constraints provides effective means for predicting the number of answers to XQuery and XPath queries, the number of updates when using the XQuery update facility, and the number of encryptions or decryptions when using XML encryption. Moreover, numerical constraints can help to optimise XQuery and XPath queries, to exclude certain choices of indices from the index selection problem, and to generate views for efficient processing of common queries and updates.

We investigate decision problems associated with numerical constraints in order to capitalise on the range of applications in XML data processing. To begin with we demonstrate that the implication problem is strongly coNP-hard for several classes of numerical constraints. These sources of potential intractability direct our attention towards the class of numerical keys that permit the specification of positive upper bounds. Numerical keys are of interest as they are reminiscent of cardinality constraints that are widely used in conceptual data modelling. At the same time, they form a natural generalisation of XML keys that are popular in XML theory and practice. We show that numerical keys are finitely satisfiable and establish a finite axiomatisation for their implication problem. Finally, we propose an algorithm that decides numerical key implication in quadratic time using shortest path methods.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

The Extensible Markup Language (XML) [8] provides a high degree of syntactic flexibility and is therefore widely used for data exchange and data integration. On the other hand, XML shows significant shortcomings when it comes to specify the semantics of its data. Consequently, the study of integrity constraints has been recognised as one of the most important yet challenging areas of XML research [16,39,42,45]. The importance of XML constraints is due to a variety of applications ranging from schema design, query optimisation, efficient storing and updating, data exchange and integration, to data cleaning [16]. Several classes of integrity constraints have been defined for XML, including keys [9,28], path constraints [11], inclusion constraints [18] and functional dependencies [5,26,30,43]. The complex structure of XML data makes it challenging to balance the trade-off between the expressiveness of constraint classes and the efficient solution of their associated decision problems [4–18,39,42].

In this paper, we discuss numerical constraints as a natural and highly useful class of XML constraints. To make effective use of these constraints in various XML applications common decision problems such as satisfiability and implication need
to be studied. In this paper, we define numerical constraints on the basis of a tree model for XML as proposed by DOM [3] and XPath [14], but independently from schema specifications such as DTDs [8] or XSDs [41]. Fig. 1 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.

Numerical constraints are intended to restrict the number of nodes in an XML tree that have the same (complex) values on some selected descendant nodes. Such restrictions can be specified absolutely, i.e. for an entire document, or for parts thereof, i.e. relatively to selected context nodes. As usual in XML data processing, path expressions provide the mechanism for selecting the nodes of interest.

For an example, consider the XML tree \( T \) in Fig. 1. The following numerical constraint may have been specified for the subtrees rooted at semester-nodes: each student/@sid-value that occurs under some course-node actually occurs as a student/@sid-value under between two and four course-nodes in the same semester-subtree. In other words, each student who is studying in a certain semester must enrol in at least two and may enrol in up to four courses. This constraint may balance the workload of students appropriately. The XML tree \( T \) violates the constraint since the second semester-node (in document order) has a course-child with a student/@sid-descendent of value “007”, but there is no further course-child of this semester-node that possesses a student/@sid-descendent of value “007”, too.

Another numerical constraint that may have been specified for the subtrees rooted at year-nodes is the following: if there are semester-children then there must be precisely two. This time, the XML tree \( T \) satisfies the constraint.

We continue with some examples that may help to illustrate the potential usefulness of numerical constraints in XML data processing. Suppose students can query their university’s results processing system for their grades in a particular year. They without querying the database itself, the student can be informed of the maximal costs that will be charged for this service, and decide accordingly. Moreover, the service provider has minimised the costs for unpaid service.

Suppose the data designer has specified the additional numerical constraint that in each year a teacher can teach up to three different courses. While the XQuery query

\[
\text{for } s \text{ in doc("enrol.xml")/year[@calendar="2007"]/course/student}
\text{where } s/@sid="007"
\text{return (grade)(s/grade)/(grade)}
\]

(1)

Without querying the database itself, the student can be informed of the maximal costs that will be charged for this service, and decide accordingly. Moreover, the service provider has minimised the costs for unpaid service.

Suppose the data designer has specified the additional numerical constraint that in each year a teacher can teach up to three different courses. While the XQuery query

\[
\text{for } s \text{ in doc("enrol.xml")/year[@calendar="2007"]/semester/course}
\text{where } s/student/@sid="247" \text{ and } s/teacher="Principal Skinner"
\text{return (course)(s/name)/(course)}
\]

is equivalent to the query

\[
\text{for } s \text{ in doc("enrol.xml")/year[@calendar="2007"]/semester/course}
\text{where } s/teacher="Principal Skinner" \text{ and } s/student/@sid="247"
\text{return (course)(s/name)/(course)}
\]

one of the two queries is likely to perform better than the other as the query engine will evaluate the \text{where} conditions either from left to right or vice versa. If the \text{where} condition on the left is evaluated first, then the second query produces less intermediate results than the first query due to the smaller selectivity of course nodes based on teachers rather than students. However, such an optimisation of XQuery queries assumes that the database management system is capable of reasoning about numerical constraints.

The idea of numerical constraints is not new in database practice: similar notions known as cardinality or participation constraints are widely used in conceptual database design. The interested reader is referred to [40] for an overview. To the best of our knowledge numerical constraints have not yet been considered in XML. On the one hand, this is to some degree surprising since these constraints reflect interesting properties of XML data. On the other hand, the generalisation of constraints to XML is challenging as argued in [16,39,42], including the identification of classes that can be reasoned about efficiently.

1.1. Contributions and related work

We introduce numerical constraints into the context of XML. We believe that numerical constraints are naturally exhibited by XML data and a thorough study of these constraints can significantly improve the semantic capabilities of XML. Moreover, they generalise both the class of XML keys [9] and several classes of cardinality and participation constraints studied in the context of the Entity-Relationship model [22,24,33,34,40].

To take full advantage of these constraints in various XML applications, it is necessary to reason about them. Similar to many other classes of XML constraints reasoning about numerical constraints is likely to be intractable in general. We show that
the finite and unrestricted satisfiability problems are different in the presence of both lower and upper bounds, and pinpoint three potential sources of intractability for the finite implication problem of numerical constraints. A thorough analysis of these observations enables us to identify a large tractable subclass of numerical constraints. We call them numerical keys as they may be seen as a relaxation of the widely known XML keys. In fact, keys turn out to be numerical keys with a fixed upper bound of 1.

Unlike many other classes of XML constraints, numerical keys can be reasoned about efficiently. Specifically, any finite set of numerical keys is finitely satisfiable, and the (finite) implication of numerical keys is finitely axiomatisable. Furthermore, we propose a quadratic-time algorithm for deciding implication that uses a characterisation of numerical key implication in terms of shortest paths in a suitable directed graph.

In summary, we believe that numerical constraints and numerical keys form natural classes of constraints that can be utilised effectively by XML data designers and XML applications. Note that even incomplete sets of sound inference rules for their implication can be of great assistance for various XML applications. In this paper, we shall concentrate on numerical keys as the complexity of their associated decision problems indicates that they can be maintained efficiently by database systems.

Finally, we would like to emphasise that the numerical constraints we study in this paper are different from the occurrence constraints ($minOccurs, maxOccurs$) of XML Schema [41]. The latter ones impose restrictions on the number of children a node
can have, but do this independently from the data values that are actually occurring. In contrast numerical constraints impose restrictions on the number of nodes in subtrees based on the data values that can be found on selected descendant nodes.

1.2. Organisation of the paper

We introduce the underlying concepts required in this paper in Section 2. These include the XML tree model, the notion of value equality and path languages for node selections. Numerical constraints are defined in Section 3 based on the common XML tree model. It is shown that reasoning about numerical constraints is computationally intractable in general. Subsequently, we identify numerical keys as a tractable subclass. In Section 4 we propose a finite axiomatisation of numerical keys. We characterise the implication of numerical keys in terms of shortest paths in Section 5 and propose an algorithm that decides implication in time quadratic in the size of the constraints. We further illustrate the benefits of numerical keys for several XML applications in Section 6. Finally, Section 7 contains conclusions and suggestions for future work.

2. Preliminaries

In this section, we recall the basics of the XML tree model, the notion of value equality, and describe the path language used to locate sets of nodes within an XML tree. Throughout the paper we assume familiarity with basic terminology from graph theory, see, e.g. [32].

2.1. The XML tree model

It is common to represent XML data by ordered, node-labelled trees. We assume that there is a countably infinite set \( E \) denoting element tags, a countably infinite set \( A \) denoting attribute names, and a singleton set \( S \) denoting text (PCDATA). We further assume that these sets are pairwise disjoint, and put \( \mathcal{L} = E \cup A \cup S \). We refer to the elements of \( \mathcal{L} \) as labels.

An XML tree is a 6-tuple \( T = (V, \text{lab}, \text{ele}, \text{att}, \text{val}, r) \) where \( V \) denotes a set of nodes, and \( \text{lab} \) is a mapping \( V \rightarrow \mathcal{L} \) assigning a label to every node in \( V \). A node \( v \in V \) is called an element node if \( \text{lab}(v) \in E \), an attribute node if \( \text{lab}(v) \in A \), and a text node if \( \text{lab}(v) = S \). Moreover, \( \text{ele} \) and \( \text{att} \) are partial mappings defining the edge relation of \( T \): for any node \( v \in V \), if \( v \) is an element node, then \( \text{ele}(v) \) is a list of element and text nodes in \( V \) and \( \text{att}(v) \) is a set of attribute nodes in \( V \). If \( v \) is an attribute or text node, then \( \text{ele}(v) \) and \( \text{att}(v) \) are undefined. The partial mapping \( \text{val} \) assigns a string to each attribute and text node: for each node \( v \in V \), \( \text{val}(v) \) is a string if \( v \) is an attribute or text node, while \( \text{val}(v) \) is undefined otherwise. Finally, \( r \) is the unique and distinguished root node. \( T \) is said to be finite if \( V \) is finite, and is said to be empty if \( V \) consists of the root node only.

For a node \( v \in V \), each node \( w \) in \( \text{ele}(v) \) or \( \text{att}(v) \) 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 \( v_0, \ldots, v_m \) in \( V \) such that \((v_i, v_{i+1})\) is an edge of \( T \) for \( i = 1, \ldots, m \). We call \( p \) a path from \( v_0 \) to \( v_m \), and say that \( v_m \) is reachable from \( v_0 \) following the path \( p \). The path \( p \) determines a word \( \text{lab}(v_1) \ldots \text{lab}(v_m) \) over the alphabet \( \mathcal{L} \), denoted by \( \text{lab}(p) \). For a node \( v \in V \), each node \( w \) reachable from \( v \) is called a descendant of \( v \). Note that an XML tree has a tree structure: for each node \( v \in V \), there is a unique path from the root node \( r \) to \( v \).

2.2. Value equality of nodes in XML trees

We can now define value equality for pairs of nodes in an XML tree. Informally, two nodes \( u \) and \( v \) of an XML tree \( T \) are value equal if they have the same label and, in addition, either they have the same string value if they are text or attribute nodes, or their children are value equal if they are element nodes. More formally, two nodes \( u, v \in V \) are value equal, denoted by \( u \equiv_v v \), if and only if the subtrees rooted at \( u \) and \( v \) are isomorphic by an isomorphism that is the identity on string values. That is, two nodes \( u \) and \( v \) are value equal when the following conditions are satisfied:

(a) \( \text{lab}(u) = \text{lab}(v) \),
(b) if \( u, v \) are attribute or text nodes, then \( \text{val}(u) = \text{val}(v) \),
(c) if \( u, v \) are element nodes, then (i) if \( \text{att}(u) = \{a_1, \ldots, a_m\} \), then \( \text{att}(v) = \{a'_1, \ldots, a'_m\} \) and there is a permutation \( \pi \) on \( \{1, \ldots, m\} \) such that \( a_i = v_{\pi(i)} \) for \( i = 1, \ldots, m \), and (ii) if \( \text{ele}(u) = \{u_1, \ldots, u_k\} \), then \( \text{ele}(v) = \{v_1, \ldots, v_k\} \) and \( u_i = v_i \) for \( i = 1, \ldots, k \).

Note that the notion of value equality takes the document order of the XML tree into account. As an example, all teacher-nodes in Fig. 1 are value equal. We remark that \( \equiv_v \) is an equivalence relation on the node set \( V \) of the XML tree. This is easy to observe as value equality between nodes corresponds to isomorphism of the subtrees rooted at these nodes.

2.3. Path expressions for node selection in XML trees

In order to define numerical constraints, we need a mechanism for selecting nodes in an XML tree. Path expressions have been widely used for node selection in XML theory and practice, cf. [14,39]. We are interested in path languages that are expressive enough to be practical, yet sufficiently simple to be reasoned about efficiently. This is the case for the languages
PL and PLₜ that have been used in [9,10] for the definition of XML keys. For the sake of completeness, we will briefly introduce these languages here.

Let ∗ be a distinguished symbol not in $\mathcal{L}$. It will serve as a \textit{variable length do not care} wildcard, that is, as a combination of a \textit{single symbol} wildcard (denoted by ∗) and the Kleene star (*). Let PL denote the set of all words over the alphabet $\mathcal{L} \cup \{\ast\}$. Further let PL₁ be the subset of PL containing all words over the alphabet $\mathcal{L}$. Both PL and PL₁ form free monoids with the binary operation of concatenation (denoted by \ast) and the empty word (denoted by $\epsilon$) as identity element.

Let P, Q be words from PL. P is a \textit{refinement} of Q, denoted by $P \preceq Q$, if P is obtained from Q by replacing wildcards in Q by words from PL. For example, $\text{year.semester.course}$ is a refinement of $\ast_{\ast}\ast_{\ast}\ast_{\ast}\ast_{\ast}\ast$ on PL. Observe that $P \sim Q$ holds if and only if P and Q are refinements of each other.

We now define the semantics of words 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 example, in the XML tree in Fig. 1, all teacher-nodes are reachable from the root node following a $\text{year.semester.course.teacher}$-path. Obviously, they are also reachable from the root node following a $\ast_{\ast}\ast_{\ast}\ast_{\ast}\ast_{\ast}$-path. For a node $v \in V$, let $v(Q)$ denote the set of nodes in T that are reachable from v following any Q-path, that is, $v(Q) = \{ v' \mid T \models Q(v, v') \}$. As an example consider the second semester-node v in the XML tree in Fig. 1. Then $v(\ast_{\ast}\ast\ast\ast\ast\ast\ast)$ is the set of all student-nodes that are descendants of the second semester node. We use $[Q]$ as an abbreviation for $r(Q)$ where r is the root node of T. Thus, $\ast_{\ast}\ast\ast\ast\ast\ast\ast$ is the set of all teacher-nodes in the entire XML tree.

Recall that each attribute or text node in an XML tree T is a leaf. Therefore, a word Q from PL is said to be valid if it does not have labels $\ell \in A$ or $\ell = S$ in a position other than the last 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 $P \subseteq Q$ implies $P \subseteq Q$.

The containment problem of PL is to decide, given valid P and Q from PL, whether $P \subseteq Q$ holds. In [10,35] it is shown that valid P, Q from PL satisfy $P \subseteq Q$ if and only if P is a refinement of Q and that the containment problem of PL can be decided in quadratic time.

In accordance with [9] we will work with the quotient set $PL_{\sim}$ rather than with PL directly: A word from PL is in \textit{normal form} if it has no consecutive wildcards. 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 wildcards. In particular, each word from PL₁ is in normal form. The natural homomorphism from PL to $PL_{\sim}$ is an isomorphism when restricted to words in normal form. By abuse of notation we will use the words from PL to denote their respective congruence class, cf. [9]. It is a straightforward exercise to extend the terminology introduced above for PL to $PL_{\sim}$.

We will call members of $PL_{\sim}$ (and $PL_{S_{\sim}}$) \textit{PL expressions} (or PL₁ expressions, respectively) in order to emphasise their use for node selection in XML. Note that there is an easy conversion of PL expressions to XPath expressions [14], just by replacing $\ast_{\ast}$ with “/,” ∗ with “.”

The choice of a path language for selecting nodes in XML trees is directly influenced by the complexity of its containment problem. Buneman et al. [9,10] argue that PL is simple yet expressive enough to be adopted by data designers and maintained by systems for XML applications. Note that Buneman et al. have included the wildcard ∗ in their definition of XML keys in [9], but not in their investigations on axiomatisability and the complexity of the implication problems [10]. Since we want to establish reasoning facilities for numerical constraints we utilise the same path languages as defined in [10]. However, partial answers to the axiomatisability and implication problems for XML keys permitting the use of the single symbol wildcard have recently been established [44].

To conclude this section we repeat the notion of value intersection from [10]: For nodes $v$ and $v'$ of an XML tree T, the \textit{value intersection} of $v(Q)$ and $v'(Q)$ is given by $v(Q) \cap v'(Q) = \{ (w, w') \mid w \in v(Q), w' \in v'(Q), w \equiv v w' \}$. That is, $v(Q) \cap v'(Q)$ 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. Numerical constraints for XML

In this section, we introduce numerical constraints for XML and show that reasoning about these constraints is likely to be computationally intractable. Furthermore, we identify numerical keys as an important tractable subclass of numerical constraints.

3.1. Defining numerical constraints

Let $\mathbb{N}$ denote the positive integers, and $\mathbb{N} = \mathbb{N} \cup \{\infty\}$ denote the positive integers together with $\infty$.

\textbf{Definition 1.} A \textit{numerical constraint} $\phi$ for XML is an expression $\text{card}(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$ where $Q, Q', Q_1, \ldots, Q_k$ are PL expressions such that $Q, Q', Q_1, \ldots, Q'_{Q_k}$ are valid, where $k$ is a non-negative integer, and where $\min \in \mathbb{N}$ and $\max \in \mathbb{N}$ with $\min \leq \max$. Herein, Q is called the \textit{context path}, Q' is called the \textit{target path}, $Q_1, \ldots, Q_k$ are called the \textit{key paths}, min is called the \textit{lower bound}, and max the \textit{upper bound} of $\phi$. If $Q = \epsilon$, we call $\phi$ \textit{absolute}; otherwise $\phi$ is called \textit{relative}.
For a numerical constraint $\varphi$, let $Q_\varphi$ denote its context path, $Q_\varphi^c$ its target path, $Q_\varphi^s_1, \ldots, Q_\varphi^s_k$ its key 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 $\varphi = \text{card}(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$ be a numerical constraint. An XML tree $T$ satisfies $\varphi$, denoted by $T \models \varphi$, if and only if for all $q \in Q$, for all $q_0^i \in q|Q'$ such that for all $x_1, \ldots, x_k$ with $x_i \in q_0^i|Q_i$ for $i = 1, \ldots, k$, it is true that

$$\min \leq \sharp \{q' \in q|Q' | \exists y_1, \ldots, y_k. \forall i = 1, \ldots, k. y_i \in q'|Q_i \land x_i =_\varphi y_i\} \leq \max$$

holds.

Note that a numerical constraint $\text{card}(Q, (Q', \{Q_1, \ldots, Q_k\})) = (\min, \max)$ enforces the cardinalities imposed by $\min$ and $\max$ only on those target nodes $q_0^i \in q|Q'$ in $T$ for which for all $i = 1, \ldots, k$ nodes $x_i \in q_0^i|Q_i$ exist in $T$. In other words, if there is no node $q_0^i$ in $T$ that meets all of these conditions, then $T$ automatically satisfies the constraint. It is future work to develop a visual language that assists the data designer in specifying numerical constraints. Such a specification language could be based on tree patterns [2] extended by $\min$ and $\max$ bounds on the target nodes.

To illustrate the definitions above, we formalise the constraints from Section 1. Let $T$ be the XML tree in Fig. 1. The constraint

$$\text{card}(_* \text{. semester}, (\text{course}, \{_* \text{. sid}\})) = (2, 4)$$

says that in every semester each student who decides to study must enrol in at least two and at most four courses. $T$ violates this constraint since Bart Simpson is enrolled in a course in the second semester of 2007 (in the "PE"-course), but Bart Simpson is not enrolled in any other course in the second semester of 2007.

However, $T$ does satisfy the second constraint

$$\text{card}(_* \text{. year}, (_* \text{. semester, } \emptyset)) = (2, 2)$$

that states that each year-node has either precisely two semester-children or none at all. The numerical constraint

$$\text{card}(_* \text{. year}, (_* \text{. course, } \{\text{teacher}\})) = (3, 6)$$

is satisfied while $\text{card}(_* \text{. semester, } (\text{course, } \{\text{teacher}\})) = (3, 3)$ is violated by $T$ since Principal Skinner only teaches maths and physics within semester 1 of 2007. An example of an absolute numerical constraint is

$$\text{card}(e, (_* \text{. course, } \{\name, \text{student.sid}\})) = (1, 3),$$

i.e., a student may attempt to pass the same course up to 3 times.

XML keys as introduced by Buneman et al. [9] and further studied in [10,28,29] are completely covered by numerical constraints. More specifically, the numerical constraint $\varphi$ is an XML key precisely when $\min_\varphi = \max_\varphi = 1$. Major observations that hold for the definition of XML keys [9,28] therefore also apply to numerical constraints. Some examples of XML keys are $\text{card}(e, (_* \text{. year, } )\text{[calendar-child,}] = (1, 1)$ stating that year-nodes can be identified by the value of their calendar-child, $\text{card}(_* \text{. year, } (_* \text{. semester, } \{\text{name}\})) = (1, 1)$ stating that semester-nodes can be identified by the value of their no-child relatively to the year, $\text{card}(_* \text{. semester, } (_* \text{. course, } \{\name\})) = (1, 1)$ stating that course-nodes can be identified by the value of their name-child relatively to the semester, and $\text{card}(_* \text{. course, } (_* \text{. student.sid})) = (1, 1)$ stating that student-nodes can be identified by the value of their sid-child relatively to the course.

### 3.2. Satisfiability and implication of numerical constraints

In this section, we define fundamental decision problems associated with various classes of constraints. Let $\Sigma$ be a finite set of constraints in a class $\mathcal{C}$ and $T$ be an XML tree. We say that $T$ satisfies $\Sigma$ if and only if $T \models \sigma$ for every $\sigma \in \Sigma$. The (finite) satisfiability problem for $\mathcal{C}$ is to determine, given any finite set $\Sigma$ of constraints in $\mathcal{C}$, whether there is a (finite) XML tree satisfying $\Sigma$.

Let $\Sigma \cup \{\varphi\}$ be a finite set of constraints in $\mathcal{C}$. We say that $\Sigma$ (finitely) implies $\varphi$, denoted by $\Sigma \models_{\text{f}} \varphi$, if and only if every (finite) XML tree $T$ that satisfies $\Sigma$ also satisfies $\varphi$. 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 $\Sigma \models_{\text{f}} \varphi$. If $\Sigma$ is a finite set of constraints in $\mathcal{C}$ let $\Sigma^*_{\text{f}}$ denote its (finite) semantic closure, i.e., the set of all constraints (finitely) implied by $\Sigma$. That is, $\Sigma^*_{\text{f}} = \{\varphi \in \mathcal{C} | \Sigma \models_{\text{f}} \varphi\}$.

For the class of XML keys, the finite and the unrestricted implication problem coincide [10]. For numerical constraints, however, the situation is already different for the satisfiability problem.
Theorem 3. The satisfiability problem and the finite satisfiability problem for the class of numerical constraints do not coincide.

Proof. Consider the numerical constraint \( \text{card}\left(\ldots, \left(\ldots, \emptyset\right)\right) = (2, \infty) \). In an XML tree satisfying this constraint, each node has at least 2 descendant nodes. As this applies in particular to the root node, there is no finite XML tree satisfying the constraint. On the other hand, an infinite path from the root node would satisfy the constraint. □

Despite Theorem 3, satisfiability for numerical constraints can be decided easily. This is mainly due to the fact that the empty XML tree satisfies every numerical constraint that contains at least one label from \( \mathcal{L} \). For the remaining numerical constraints satisfiability can be checked on a case-by-case basis. For example, \( \text{card}(e, \left(\ldots, \left(\ldots, \emptyset\right)\right)) = (\min, \max) \) is (finitely) satisfiable if and only if \( \min = 1 \) due to the uniqueness of the root node. We leave the straightforward, but tedious analysis of the remaining cases to the reader as it is not essential for our investigation here.

3.3. Computational intractability

Deciding implication, on the other hand, is not that easy. We will show that reasoning about numerical constraints is likely to be intractable already for very restricted subclasses. Our first result concerns simple absolute numerical constraints that have a non-empty set of key paths. Put

\[
\mathcal{F}_1 = \{\text{card}(e, (P', (P_1, \ldots, P_k))) = (\min, \max) \mid P', P_1, \ldots, P_k \in \mathcal{P}_A, k \geq 1, \max \leq 5\}.
\]

The 3-Colourability problem asks for a given graph \( G \) with vertex set \( V = \{1, \ldots, n\} \) and edge set \( E \) whether there is a node-colouring \( f : V \to \{1, 2, 3\} \) such that no edge in \( E \) has two nodes of the same colour. The problem is known to be NP-complete, cf. [32].

Theorem 4. The finite implication problem for the class \( \mathcal{F}_1 \) is coNP-hard.

Proof. We show that the 3-Colourability problem polynomially transforms to the complement of the problem under inspection. Let \( G = (V, E) \) be an instance of the 3-Colourability problem, where \( n \) denotes the cardinality of the vertex set \( V \). We shall find a constraint set \( \Sigma \cup \{\varphi\} \) in \( \mathcal{F}_1 \) such that \( \Sigma \) does not finitely imply \( \varphi \) if and only if \( G = (V, E) \) is a yes-instance. Let \( \ell_A, \ell_B, \ell_1, \ldots, \ell_n \) be mutually distinct labels from \( \mathcal{L} \). Let \( \Sigma \) consist of the following constraints:

\[
\begin{align*}
\text{card}(e, (\ell_A, [\ell_B])) & = (5, 5) \\
\text{card}(e, (\ell_A, [\ell_B], [\ell_1])) & = (3, 3) \quad \text{for all } i \in V \\
\text{card}(e, (\ell_A, [\ell_B], [\ell_1], [\ell_2])) & = (1, 2) \quad \text{for all } (i, j) \in E.
\end{align*}
\]

Further, let \( \varphi \) be the constraint \( \text{card}(e, (\ell_A, [\ell_B], [\ell_1], \ldots, [\ell_n])) = (1, 1) \).

Suppose \( G = (V, E) \) is a yes-instance of the 3-Colourability problem, that is, there is a node-colouring \( f : V \to \{1, 2, 3\} \) such that no edge in \( E \) has two nodes of the same colour. For \( m = 1, \ldots, 3 \), let \( V_m \) denote the set of all nodes in \( V \) with colour \( m \). In addition, we put \( V_4 = V_5 = V \). We construct a finite XML tree whose root node \( r \) has five children \( u_1, \ldots, u_5 \), all with label \( \ell_A \). For \( m = 1, \ldots, 5 \), let \( u_m \) have a child \( v_m \) with label \( \ell_B \). Moreover, for \( m = 1, \ldots, 5 \) and \( i \in V_m \), let \( u_m \) have a child \( w_{mi} \) with label \( \ell_C \). It is easy to check that the resulting XML tree satisfies \( \Sigma \), but violates \( \varphi \) due to the existence of \( u_4, u_5 \). Hence, \( \Sigma \) does not imply \( \varphi \).

Conversely, suppose \( \Sigma \) does not finitely imply \( \varphi \), that is, there exists some finite XML tree \( T \) that satisfies \( \Sigma \), but violates \( \varphi \). Since \( T \) violates \( \text{card}(e, (\ell_A, [\ell_B], [\ell_1], \ldots, [\ell_n])) = (1, 1) \), its root node \( r \) has two children \( u_4 \) and \( u_5 \) with label \( \ell_A \), both \( u_m \) (\( m = 4, 5 \)) have value equal children \( v_{m1} \) with label \( \ell_B \), and for each \( i \in V \) both \( u_m \) (\( m = 4, 5 \)) have value equal children \( w_{mi} \) with label \( \ell_C \). By \( \text{card}(e, (\ell_A, [\ell_B])) = (5, 5) \), the root node \( r \) of \( T \) has another three children \( u_1, u_2, u_3 \) with label \( \ell_A \), each having a child \( v_{m1} \) (\( m = 1, 2, 3 \)) with a label \( \ell_B \) that is value equal to \( v_4 \) and \( v_5 \). Note that \( r \) may have further children with label \( \ell_A \) that again may have children with label \( \ell_B \), but those grand-children must not be value equal to \( v_4 \) and \( v_5 \). Therefore, by \( \text{card}(e, (\ell_A, [\ell_B], [\ell_C])) = (3, 3) \), for each \( i \in V \), exactly one of \( u_1, u_2, u_3 \) has a child \( w_{mi} \) with label \( \ell_C \) that is value equal to \( w_{4i} \) and \( w_{5i} \). This defines a partition of \( V \) into three subsets \( V_1, V_2, V_3 \) where \( V_m \) consists of all those \( i \in V \) for which the child \( w_{mi} \) exists. We claim that this partition of \( V \) induces a node-colouring \( f : V \to \{1, 2, 3\} \) where no edge in \( E \) has two nodes of the same colour. Assume, on the contrary, that there is an edge \( [i, j] \) where \( i \) and \( j \) belong to the same subset \( V_m \). Then, however, \( \text{card}(e, (\ell_A, [\ell_B], [\ell_C])) = (1, 2) \) would be violated due to the existence of \( u_m, u_4 \) and \( u_5 \), which contradicts the choice of \( T \). Hence, the partition of \( V \) into \( V_1, V_2, V_3 \) indeed induces a node-colouring \( f : V \to \{1, 2, 3\} \) where no edge in \( E \) has two nodes of the same colour, that is, \( G = (V, E) \) is a yes-instance of the 3-Colourability problem. □

The previous result suggests that computational intractability may result from the specification of both lower and upper bounds.
The next result suggests that empty sets of key paths may form another source of computational intractability. We are concerned with simple absolute numerical constraints where the lower bound is fixed to 1, i.e., these constraints only permit the specification of upper bounds. Put

$$\mathcal{F}_2 = \{\text{card}(e, (P', (P_1, \ldots, P_k))) = (1, \max) \mid P', P_1, \ldots, P_k \in \mathcal{P}_k, k \geq 0, \max \leq 6\}.$$

**Theorem 5.** The finite implication problem for the class $\mathcal{F}_2$ is coNP-hard.

**Proof.** We show that the 3-Colourability problem polynomially transforms to the complement of the problem under inspection. Let $G = (V, E)$ be an instance of the 3-Colourability problem, where $n$ denotes the cardinality of the vertex set $V$.

We shall find a constraint set $\Sigma \cup \{\varphi\}$ in $\mathcal{F}_2$ such that $\Sigma$ does not finitely imply $\varphi$ if and only if $G = (V, E)$ is a yes-instance. Let $\ell_A, \ell_B, \ell_C_1, \ldots, \ell_C_n$ be mutually distinct labels from $\mathcal{L}$. Let $\Sigma$ consist of the following constraints:

$$\text{card}(e, (\ell_A, \ell_B, \ell_C_1)) = \begin{cases} (1, 6) & \text{for all } (i, j) \in E. \end{cases}$$

Further, let $\varphi$ be the constraint $\text{card}(e, (\ell_A, \ell_B, \ell_C_1, \ldots, \ell_C_n)) = (1, 1)$.

Suppose $G = (V, E)$ is a yes-instance of the 3-Colourability problem, that is, there is a node-colouring $f : V \to \{1, 2, 3\}$ such that no edge in $E$ has two nodes of the same colour.

For $m = 1, \ldots, 3$, let $V_m$ denote the set of all nodes in $V$ with colour $m$.

We construct a finite XML tree whose root node $r$ has two value equal children $u_1, u_2$, both with label $\ell_A$. For $h = 1, 2$, let $u_h$ have three children $v_{h1}, v_{h2}$ and $v_{h3}$, all with label $\ell_B$. Moreover, for $h = 1, 2$ and $i \in V$, let $w_{hi}$ be a node with label $\ell_C$. If $i \in V_m$ we put $w_{hi}$ as a child of $v_{hi}$ (and thus as a grandchild of $u_h$). It is easy to check that the resulting XML tree satisfies $\Sigma$, but violates $\varphi$ due to the existence of $u_1, u_2$. Hence, $\Sigma$ does not imply $\varphi$.

Conversely, suppose $\Sigma$ does not absolutely imply $\varphi$, that is, there exists some finite XML tree $T$ that satisfies $\Sigma$, but violates $\varphi$. Since $T$ violates $\text{card}(e, (\ell_A, \ell_B, \ell_C_1, \ldots, \ell_C_n)) = (1, 1)$, its root node $r$ has two value equal children $u_1, u_2$ with label $\ell_A$ and with $\ell_B, \ell_C$-paths ($i \in V$) starting from both of them. Note that there may be several ways to choose the nodes $w_{hi}$ ($h = 1, 2$ and $i \in V$) be the terminal nodes of these paths. Then, for $h = 1, 2$, let $w_{hi}$ be a node with label $\ell_C$. If $i \in V_m$ we put $w_{hi}$ as a child of $v_{hi}$ (and thus as a grandchild of $u_h$). Note that if some of the nodes $v_{1m}$ ($m = 1, 2, 3$) do not exist, we keep the respective subsets $V_m$ empty. We claim that this partition of $V$ induces a node-colouring $f : V \to \{1, 2, 3\}$ where no edge in $E$ has two nodes of the same colour. Assume, on the contrary, that there is an edge $(i, j)$ where $i$ and $j$ belong to the same subset $V_m$. Then, however, $\text{card}(e, (\ell_A, \ell_B, (\ell_C_1, \ell_C_2))) = (1, 1)$ would be violated due to the existence of $v_{1m}$ and $v_{2m}$. This contradicts the choice of $T$.

Next, we identify another source of intractability: the permission to have arbitrary $\mathcal{P}_L$ expressions in both target- and key paths. Our result concerns absolute numerical constraints that have a non-empty set of key paths and where the lower bound is fixed to 1, i.e., these constraints only permit the specification of upper bounds. Put

$$\mathcal{F}_3 = \{\text{card}(e, (Q', (Q_1, \ldots, Q_k))) = (1, \max) \mid k \geq 1, \max \leq 4\}.$$

**Theorem 6.** The finite implication problem for the class $\mathcal{F}_3$ is coNP-hard.

**Proof.** We show that the 3-Colourability problem polynomially transforms to the complement of the problem under inspection. Let $G = (V, E)$ be an instance of the 3-Colourability problem, where $n$ denotes the cardinality of the vertex set $V$.

We shall find a constraint set $\Sigma \cup \{\varphi\}$ in $\mathcal{F}_3$ such that $\Sigma$ does not finitely imply $\varphi$ if and only if $G = (V, E)$ is a yes-instance. Let $\ell_A, \ell_B, \ell_C_1, \ldots, \ell_C_n$ be mutually distinct labels from $\mathcal{L}$. Let $\Sigma$ consist of the following constraints:

$$\text{card}(e, (\ell_A, (\ell_C_1))) = \begin{cases} (1, 2) & \text{for all } i \in V. \end{cases}$$

Further, let $\varphi$ be the constraint $\text{card}(e, (\ell_A, (\ell_C_1, \ldots, \ell_C_n))) = (1, 2)$. □
3.4. Numerical keys

In order to take advantage of XML applications effectively it becomes necessary to reason about constraints efficiently. The results from the last section motivate the study of a large subclass of numerical constraints that, as we shall show in this paper, turns out to be computationally tractable.

Definition 7. A numerical key for XML is a numerical constraint $\text{card}(Q, (Q', (P_1, \ldots, P_k))) = (1, \text{max})$ such that $P_1, \ldots, P_k$ are $PL_s$ expressions and $k$ is a positive integer. We will use $\text{card}(Q, (Q', (P_1, \ldots, P_k))) \leq \text{max}$ to denote the numerical key. Let $\mathcal{N}$ denote the class of all numerical keys.

Note that numerical keys are still far more expressive than XML keys. More specifically, a numerical key $\varphi$ becomes a key precisely when $\text{max}^\varphi = 1$. The next result states that every finite set of numerical keys can be satisfied by some finite XML tree: we can just choose the empty XML tree. This extends a corresponding result for XML keys [10].

Proposition 8. Every finite set of numerical keys is finitely satisfiable.

Furthermore, the coincidence of finite and unrestricted implication carries over from keys to numerical keys. In what follows we will therefore speak of the implication problem for numerical keys.

Theorem 9. The implication and finite implication problems for numerical keys coincide.

Proof. Let $\Sigma \cup \{\varphi\}$ be a finite set of numerical keys in $\mathcal{N}$. We have $\Sigma \models \varphi$ if and only if there is no XML tree $T$ that satisfies $\Sigma$ and does not satisfy $\varphi$. It is therefore sufficient to show that the existence of an XML tree $T$ that satisfies $\Sigma$ and does not satisfy $\varphi$ implies the existence of some finite XML tree $T'$ that satisfies $\Sigma$ and does not satisfy $\varphi$. Indeed, if $T$ does not satisfy $\varphi$, then there is some $q \in \{Q_x\}$, there are pairwise distinct $q_1, \ldots, q_{\text{max}^\varphi + 1} \in q\{Q_x\}$ and there are $x_1^j \in q\{P_{i_j}^\varphi\}$ for all $i = 1, \ldots, k_\varphi$ and $j = 1, \ldots, \text{max}^\varphi + 1$ such that $x_1^j = v x_2^j$ for all $i = 1, \ldots, k_\varphi$ and $j = 1, \ldots, \text{max}^\varphi + 1$. Let $T'$ be the finite XML subtree of $T$ consisting of nodes in paths from the root node $r$ to $x_j^i$ for $i = 1, \ldots, k_\varphi$ and $j = 1, \ldots, \text{max}^\varphi + 1$. If $x_j^i$ is an element node for some $i \in \{1, \ldots, k_\varphi\}$ and $j \in \{1, \ldots, \text{max}^\varphi + 1\}$, then we add a text node as a new child of $x_j^i$ to $T'$. Then we choose string values for these child nodes in such a way that a node pair $x, y$ in $T'$ is value equal if and only if it is value equal in $T$. This ensures that $T'$ satisfies $\Sigma$. Moreover, $T'$ does not satisfy $\varphi$ by construction. \(\square\)

4. Axiomatization of numerical keys

The notion of derivability ($\vdash_{\mathcal{R}}$) with respect to a set $\mathcal{R}$ of inference rules can be defined analogously to the notion in the relational data model [1, pp. 164–168]. The aim in this section is to find a set $\mathcal{R}$ of inference rules which is sound and complete for the implication of numerical keys. A set $\mathcal{R}$ of inference rules is sound (complete) for the implication of numerical keys, if for all finite sets $\Sigma$ of numerical keys we have $\Sigma_{\mathcal{R}} \subseteq \Sigma^*$ ($\Sigma^* \subseteq \Sigma_{\mathcal{R}}$) where $\Sigma_{\mathcal{R}} = \{\varphi \mid \Sigma \vdash_{\mathcal{R}} \varphi\}$ denotes the syntactic closure of $\Sigma$ under inference using $\mathcal{R}$.

4.1. Inference rules for numerical keys

Table 1 shows a set of inference rules for the implication of numerical keys. The majority of these rules are extensions of the inference rules for keys [10,28]. Moreover, $\text{infinity}$, $\text{weakening}$, and $\text{superkey}$ rule form an axiomatisation for max-cardinality constraints in the Entity-Relationship model [24].

The $\text{infinity}$ rule introduces a numerical key that is satisfied by any XML tree since $\infty$ does not put any restriction on the tree. The $\text{weakening}$ rule indicates that a numerical key with upper bound $\text{max} < \infty$ implies $\text{infinitely} many numerical keys with less restrictive upper bounds $\text{max'} > \text{max}$. Therefore, $\Sigma^*$ is infinite in general.

A significant rule is the $\text{multiplication}$ rule. The $\text{interaction}$ rule for keys in [10,28] appears as the special case of the $\text{multiplication}$ rule where $\text{max} = 1 = \text{max'}$. We will illustrate the $\text{multiplication}$ rule by the following example.

Example 10. Imagine a sports league competition in which teams compete against each other during a season. The numerical key

$$\sigma_1 = \text{card} (\_+.\text{season}, (\text{month}, \{\text{match.home}.S, \text{match.away}.S\})) \leq 3$$

indicates that during a season there are at most 3 months in which the same home team and the same away team can both play a match. Furthermore, the same home team can match up against the same away team twice during each month of the season, i.e.,
Table 1
An axiomatisation for numerical keys.

<table>
<thead>
<tr>
<th>card(Q, (Q', S)) ≤</th>
<th>card(Q, (ε, S)) ≤ 1</th>
<th>card(Q, (Q', S)) ≤ max + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(infinity)</td>
<td>(epsilon)</td>
<td>(weakening)</td>
</tr>
<tr>
<td>card(Q, (Q', S)) ≤ max</td>
<td>card(Q, (Q', {P'}) ≤ max</td>
<td>card(Q, (Q', Q'', S)) ≤ max</td>
</tr>
<tr>
<td>(superkey)</td>
<td>(subnodes)</td>
<td>(target-to-context)</td>
</tr>
<tr>
<td>card(Q'', (Q', S)) ≤ Q'' ≤ Q</td>
<td>card(Q, (Q', S)) ≤ max</td>
<td>card(Q, (Q', S U {ε, P})) ≤ max</td>
</tr>
<tr>
<td>(context-path-containment)</td>
<td>(target-path-containment)</td>
<td>(prefix-epsilon)</td>
</tr>
<tr>
<td>card(Q, (Q', P, {ε, P})) ≤ max</td>
<td>card(Q, (Q', P1, ..., Pk)) ≤ max, card(Q, Q', (P, {P1, ..., Pk}) ≤ max'</td>
<td>card(Q, (Q', P, {P1, ..., Pk})) ≤ max · max'</td>
</tr>
<tr>
<td>(subnodes-epsilon)</td>
<td></td>
<td>(multiplication)</td>
</tr>
</tbody>
</table>


The soundness of the multiplication rule tells us now that

\[ \text{card}(_*, \text{season.month}, (\text{match}, [\text{home.S, away.S}])) \leq 6 \]

is implied by these two numerical keys. That is, during every season the same home team can host the same away team up to 6 times. Suppose that the rules of the league suggest that the same home team plays the same away team at most 5 times during a season. This constraint ϕ*, i.e.,

\[ \text{card}(_*, \text{season}, (\text{month.match}, [\text{home.S, away.S}])) \leq 5 \]

is not implied by σ1 and σ2 as the XML tree in Fig. 9 shows. Therefore, this numerical key must be specified in addition to σ1 and σ2.

4.2. Soundness of the inference rules

Next we show that inferences using the rules in Table 1 always lead to numerical keys that are implied.

Lemma 11. The inference rules in Table 1 are sound for the implication of numerical keys in \( \mathcal{N}^\prime \).

Proof. We will show the following: if a finite XML tree T satisfies all the premises of an inference rule in Table 1, then T also satisfies the conclusion of the inference rule. Based on Theorem 9, the soundness of the set of inference rules in Table 1 for the implication of numerical keys in \( \mathcal{N}^\prime \) follows then by induction on the length of an inference of an arbitrary numerical key \( \varphi \) from an arbitrary set \( \Sigma \) of numerical keys in \( \mathcal{N}^\prime \).

For the soundness of the infinity rule let \( \varphi = \text{card}(Q, (Q', (P_1, ..., P_k))) \leq \infty \). Then every finite XML tree T satisfies \( \varphi \) since for all \( q \in [Q] \), for all \( q_i \in q[Q'] \) such that for all \( x_1, ..., x_k \) with \( x_i \in q'[Q] \) for \( i = 1, ..., k \), the set

\[ \{ q' \in q[Q'] | \exists y_1, ..., y_k, \forall i = 1, ..., k, y_i \in q'[Q] \land x_i =_T y_i \} \]

has only finitely many elements.

For the soundness of the epsilon rule let \( \varphi = \text{card}(Q, (\varepsilon, (P_1, ..., P_k))) \leq \text{max} \). Then every finite XML tree T satisfies \( \varphi \) since for all \( q \in [Q] \) the set \( q[\varepsilon] \) is a singleton.

For the soundness of the weakening rule let \( \sigma = \text{card}(Q, (Q', (P_1, ..., P_k))) \leq \text{max} \), and let T be an arbitrary finite XML tree that satisfies \( \sigma \). Let \( q \in [Q] \), and \( q_i \in q[Q'] \) such that for all \( x_1, ..., x_k \) we have \( x_i \in q'[P_i] \) for \( i = 1, ..., k \). Since T satisfies \( \sigma \) it follows that

\[ \exists q' \in q[Q'] | \exists y_1, ..., y_k, \forall i = 1, ..., k, y_i \in q'[P_i] \land x_i =_T y_i \leq \text{max} \]

However, since \( \text{max} \leq \text{max} + 1 \) holds we conclude that T also satisfies \( \varphi = \text{card}(Q, (Q', (P_1, ..., P_k))) \leq \text{max} + 1 \).
For the soundness of the superkey rule let \( \varphi = \text{card}(Q, (Q', (P_1, \ldots, P_k) \cup \{P_{k+1}\})) \leq \text{max} \), and let \( T \) be an arbitrary finite XML tree that violates \( \varphi \). Consequently, there are \( q \in [Q], q'_0 \in q[Q'] \) and \( x_1, \ldots, x_{k+1} \) with \( x_j \in q[P_j] \) for \( i = 1, \ldots, k+1 \) such that

\[
\exists q' \in q[Q'] \mid \exists y_1, \ldots, y_{k+1} . \forall i = 1, \ldots, k+1 . y_i \in q'[P_i] \wedge x_i =_v y_i > \text{max} .
\]

However, it follows immediately from the definition of satisfaction that \( T \) also violates \( \sigma = \text{card}(Q, (Q', (P_1, \ldots, P_k))) \leq \text{max} \).

For the soundness of the subnodes rule let \( \varphi = \text{card}(Q, (Q', (P.P'))) \leq \text{max} \), and let \( T \) be an arbitrary finite XML tree that violates \( \varphi \). Consequently, there are \( q \in [Q], q'_0 \in q[Q'] \) and \( p' \in q'_0[P.P'] \) such that

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

Note that since \( T \) is a tree and \( P.P' \) a PL-expression it follows that no pair of distinct nodes \( q' \) in the previous set is in an ancestor/descendant-relationship. By definition, there is some \( p_0 \in q[Q'.P] \) such that \( p_0 \in q'_0[P] \) and \( p' \in p_0[P.P'] \). Moreover, it follows that

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

However, by the definition of satisfaction this shows that \( T \) also violates \( \sigma = \text{card}(Q, (Q'.P', (P')) \leq \text{max} \). The situation is depicted in the left of Fig. 2.

For the soundness of the target-to-context rule let \( \varphi = \text{card}(Q, Q', (Q'', (P_1, \ldots, P_k))) \leq \text{max} \), and let \( T \) be an arbitrary finite XML tree that violates \( \varphi \). Consequently, there are \( q' \in [Q.Q'], q''_0 \in q[Q''] \) such that \( x_1, \ldots, x_k \) with \( x_i \in q''_0[P_i] \) for \( i = 1, \ldots, k \) such that

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

By definition there is some \( q \in [Q] \) such that \( q' \in q[Q'] \) and \( q''_0 \in q[Q''] \) hold. Consequently, \( T \) also violates \( \sigma = \text{card}((Q, (Q'.Q'', (P_1, \ldots, P_k))) \leq \text{max} \).

For the soundness of the context-path-containment rule let \( \varphi = \text{card}(Q''.Q', (P_1, \ldots, P_k))) \leq \text{max} \), and let \( T \) be an arbitrary finite XML tree that violates \( \varphi \). Consequently, there are \( q \in [Q'], q'_0 \in q[Q'] \) such that \( x_1, \ldots, x_k \) with \( x_i \in q'_0[P_i] \) for \( i = 1, \ldots, k \) such that

\[
\exists q' \in q[Q'] \mid \exists y_1, \ldots, y_k . \forall i = 1, \ldots, k . y_i \in q'_0[P_i] \wedge x_i =_v y_i > \text{max} .
\]

However, since \( q \in [Q'' \subseteq Q \text{ holds by assumption it follows that } q \in [Q] \text{ holds as well. This shows that } T \text{ also violates } \sigma = \text{card}(Q, (Q'.Q'', (P_1, \ldots, P_k))) \leq \text{max} \).

For the soundness of the target-path-containment rule let \( \varphi = \text{card}(Q, (Q''.Q', (P_1, \ldots, P_k))) \leq \text{max} \), and let \( T \) be an arbitrary finite XML tree that violates \( \varphi \). Consequently, there are \( q \in [Q], q'_0 \in q[Q'] \) such that \( x_1, \ldots, x_k \) with \( x_i \in q'_0[P_i] \) for \( i = 1, \ldots, k \) such that

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

However, since \( q'_0 \in q[Q'] \) and \( Q'' \subseteq Q \) holds by assumption it follows that \( q'_0 \in q[Q'] \) holds as well. Moreover, as \( Q'' \subseteq Q' \) holds by assumption it follows that there are \( x_1, \ldots, x_k \) with \( x_i \in q'_0[P_i] \) for \( i = 1, \ldots, k \) such that

\[
\exists q' \in q[Q'] \mid \exists y_1, \ldots, y_k . \forall i = 1, \ldots, k . y_i \in q'[P_i] \wedge x_i =_v y_i > \text{max} .
\]
This shows that $T$ also violates $σ = \text{card}(Q, (Q', (P_1, \ldots, P_k))) \leq \text{max}$.

For the soundness of the prefix-epsilon rule let $ϕ = \text{card}(Q, (Q', (P_1, \ldots, P_k, ε, P.P'))) \leq \text{max}$, and let $T$ be an arbitrary finite XML tree that violates $ϕ$. Consequently, there are $q ∈ [Q]. q'_0 ∈ q[Q']$ such that there are $x_1, \ldots, x_k$ with $x_i ∈ q'_0[P_i]$ for $i = 1, \ldots, k$ and $x_i^{'} ∈ q'_0[P.P']$ such that

$$z(q' ∈ q[Q'] | q'_0 = v q' ∧ ∃y_1, \ldots, y_k. y'_i = 1, \ldots, k. y_i ∈ q'[P_i] ∧ x_i = v y'_i ∧ x = v y) > \text{max}.$$ 

It follows that $q ∈ [Q], q'_0 ∈ q[Q']$ such that there are $x_1, \ldots, x_k$ with $x_i ∈ q'_0[P_i]$ for $i = 1, \ldots, k$ and $x ∈ q'_0[P]$ such that

$$z(q' ∈ q[Q'] | q'_0 = v q' ∧ ∃y_1, \ldots, y_k. y'_i = 1, \ldots, k. y_i ∈ q'[P_i] ∧ x_i = v y'_i ∧ x = v y) > \text{max}.$$ 

In particular, the value equality of $x ∈ q'_0[P]$ and $y ∈ q'[P]$ follows from the value equality of $q'_0$ and $q'$. Consequently, $T$ violates $σ = \text{card}(Q, (Q', (P_1, \ldots, P_k))) \leq \text{max}. The situation is depicted in the right of Fig. 2.

For the soundness of the subnodes-epsilon rule let $ϕ = \text{card}(Q, (Q', (ε, P.P'))) \leq \text{max}$, and let $T$ be an arbitrary finite XML tree that violates $ϕ$. Consequently, there are $q ∈ [Q], q'_0 ∈ q[Q']$ such that there is some $x' ∈ q'_0[P.P']$ such that

$$z(q' ∈ q[Q'] | q'_0 = v q' ∧ ∃y_1, \ldots, y_k. y'_i = 1, \ldots, k. y_i ∈ q'[P_i] ∧ x_i = v y'_i ∧ x = v y) > \text{max}.$$ 

It follows that $q ∈ [Q]$, and there is some $p_0 ∈ q[Q'.P]$ with $p_0 ∈ q'_0[P]$ such that $x' = v y') > \text{max}$. 

In particular, the value equality of $p_0 ∈ q'_0[Q'.P]$ and $p ∈ q[Q'.P]$ follows from the value equality of $q'_0$ and the ancestor $q' ∈ q[Q']$ of $p$ in $q'[P]$. Consequently, $T$ violates $σ = \text{card}(Q, (Q', (P_1, \ldots, P_k))) \leq \text{max}.$

It remains to verify the soundness of the multiplication rule. Let $T$ be an arbitrary finite XML tree that satisfies $σ_1 = \text{card}(Q, (Q', (P.P,P_1, \ldots, P.P_k))) \leq \text{max}$ and $σ_2 = \text{card}(Q, Q', (P_1, \ldots, P_k)) \leq \text{max}$. We want to show that $T$ also satisfies $ϕ = \text{card}(Q, Q', (P_1, \ldots, P_k)) \leq \text{max}$. Let $q ∈ [Q], p_0 ∈ q[Q'.P]$ where $q'_0 ∈ q[Q']$ and $p_0 ∈ q'_0[P]$ such that there are $x_1, \ldots, x_k$ with $x_i \in p_0[P_i]$ for $i = 1, \ldots, k$. Since $T$ satisfies $σ_1$ we have

$$z(q' ∈ q[Q'] | ∃y_1, \ldots, y_k. y_i = 1, \ldots, k. y_i ∈ q'[P.P_i] ∧ x_i = v y_i) \leq \text{max}.$$ 

On the other hand, since $T$ satisfies $σ_2$ we have

$$z[p ∈ q'_0[P] | ∃y_1, \ldots, y_k. y_i = 1, \ldots, k. y_i ∈ p[P_i] ∧ x_i = v y_i)] \leq \text{max}.$$ 

Consequently, we conclude that

$$z[p ∈ q[Q'.P] | ∃y_1, \ldots, y_k. y_i = 1, \ldots, k. y_i ∈ p[P_i] ∧ x_i = v y_i)] \leq \text{max} \cdot \text{max}'.$$

Since $q ∈ [Q]$ was chosen arbitrarily it follows that $T$ satisfies $ϕ$. The situation is depicted in Fig. 3. □

4.3. Completeness of the inference rules

In this section, we are concerned with proving the completeness of the inference rules in Table 1. That is, every implied numerical key can be inferred by using only the inference rules from Table 1. The argument to be used extends an earlier
technique applied to XML keys [28]. We start by introducing two further notions that will play a central role in the completeness proof later on, namely the mini-tree and the cardinality graph.

Let \( \Sigma \cup \{w\} \) be a finite set of numerical keys in \( \mathcal{N} \). Let \( \mathcal{L}_{\Sigma,w} \) denote the set of all labels \( \ell \in \mathcal{L} \) that occur in PL expressions of numerical keys in \( \Sigma \cup \{w\} \), and fix a label \( \ell_0 \in E - \mathcal{L}_{\Sigma,w} \). Further, let \( O_w \) and \( O_w' \) be the PL \( \ell \) expressions obtained from the PL expressions \( Q_w \) and \( Q_w' \), respectively, when replacing each \( \ell \) by \( \ell_0 \).

Let \( p \) be an \( O_w \)-path from a node \( r_p \) to a node \( q_p \), let \( p' \) be an \( O_w' \)-path from a node \( r_p' \) to a node \( q_p' \), and, for each \( i = 1, \ldots, k_w \), let \( p_i \) be a \( P_i \)-path from a node \( r_i \) to a node \( x_i \), such that the paths \( p, p', p_1, \ldots, p_{k_w} \) are mutually node-disjoint. From the paths \( p, p', p_1, \ldots, p_{k_w} \) we obtain the mini-tree \( T_{\Sigma,w} \) by identifying the node \( r_q \) with \( q_p \) and by identifying each of the nodes \( r_i \) with \( q_i \). Note that \( q_p \) is the unique node in \( T_{\Sigma,w} \) that satisfies \( q_p \in [O_w] \), and \( q_i \) is the unique node in \( T_{\Sigma,w} \) that satisfies \( q_i \in [O_w'] \).

In the sequel, we will discuss how to construct an XML tree from \( T_{\Sigma,w} \) that could serve as a counter-example for the implication of \( \varphi \) by \( \Sigma \). A major step in this construction is the duplication of certain nodes of \( T_{\Sigma,w} \). To begin with, we determine those nodes of \( T_{\Sigma,w} \) for which we will generate sufficiently many value equal copies in a possible counter-example tree. The marking of the mini-tree \( T_{\Sigma,w} \) is a subset \( \mathcal{M} \) of the node set of \( T_{\Sigma,w} \); if for all \( i = 1, \ldots, k_w \), we have \( P_i \neq \emptyset \), then \( \mathcal{M} \) consists of the leaves of \( T_{\Sigma,w} \), and otherwise \( \mathcal{M} \) consists of all descendant nodes of \( q_p \) in \( T_{\Sigma,w} \). The nodes in \( \mathcal{M} \) are said to be marked.

**Example 12.** The left picture of Fig. 4 shows the mini-tree of \( card(=*\text{ season}, (\text{month.match, \{home.S,away.S\}})) \leq 5 \) and its marking (leaves are marked by \( \times \)).

We use mini-trees to calculate the impact of a numerical key in \( \Sigma \) on a possible counter-example tree for the implication of \( \varphi \) by \( \Sigma \). To distinguish numerical keys that have an impact from those that do not, we introduce the notion of applicability, cf. [28]. Let \( T_{\Sigma,w} \) be the mini-tree of the numerical key \( \varphi \) with respect to \( \Sigma \), and let \( \mathcal{M} \) be its marking. A numerical key \( \sigma \) is said to be applicable to \( \varphi \) if there are nodes \( w_\sigma \in [Q_w] \) and \( w'_\sigma \in [Q_w'] \) in \( T_{\Sigma,w} \) such that \( w'_\sigma [P_i] \cap \mathcal{M} \neq \emptyset \) for all \( i = 1, \ldots, k_w \). We say that \( w_\sigma \) and \( w'_\sigma \) witness the applicability of \( \sigma \) to \( \varphi \).

**Example 13.** Both of the numerical keys \( \sigma_1 \) and \( \sigma_2 \) from Example 10 are applicable to \( \varphi = card(=*\text{ season}, (\text{month.match, \{home.S,away.S\}})) \leq 5 \). On the other hand, \( card(=*\text{ season.month}, (\text{match, \{e\}})) \leq 3 \) is not applicable to \( \varphi \). Indeed, if \( v \) denotes the unique node in \([*\text{ month.match}]\) (left of Fig. 4), then \( \forall \emptyset \subseteq \{v\} \) but \( \emptyset \) is not marked, i.e., \( \emptyset \subseteq \mathcal{M} \).

We define the cardinality graph \( G_{\Sigma,w} \) of \( \varphi \) and \( \Sigma \) as the node-labelled digraph obtained from \( T_{\Sigma,w} \) as follows: the nodes and node-labels of \( G_{\Sigma,w} \) are exactly the nodes and node-labels of \( T_{\Sigma,w} \), respectively. The edges of \( G_{\Sigma,w} \) consist of the reversed edges from \( T_{\Sigma,w} \). Furthermore, for each numerical key \( \sigma \) in \( \Sigma \) that is applicable to \( \varphi \) and for each pair of nodes \( w_\sigma \in [Q_w] \) and \( w'_\sigma \in [Q_w'] \) that witness the applicability of \( \sigma \) to \( \varphi \) we add a directed edge \( (w_\sigma, w'_\sigma) \) to \( G_{\Sigma,w} \). Subsequently, we refer to these additional edges as witness edges while the reversed edges from \( T_{\Sigma,w} \) are referred to as upward edges of \( G_{\Sigma,w} \). This is motivated by the fact that for every witness \( w_\sigma \) and \( w'_\sigma \) the node \( w'_\sigma \) is a descendant node of the node \( w_\sigma \) in \( T_{\Sigma,w} \), and thus the witness edge \( (w_\sigma, w'_\sigma) \) is a downward edge or loop in \( G_{\Sigma,w} \).

So far, the cardinality graph is similar to the witness graph as introduced in [28]. The opposite orientation of edges, however, results from our objective to yield a characterisation of numerical key implication in terms of shortest paths. As a novelty we now introduce weights as edge-labels: every upward edge \( e \) of \( G_{\Sigma,w} \) has weight \( \omega(e) = 1 \), and every witness edge \( (u, v) \) of \( G_{\Sigma,w} \) has weight \( \omega(u, v) = \min(\max|\ell, |(u, v)|) \) witnesses the applicability of some \( \sigma \in \Sigma \) to \( \varphi \).

For convenience we recall some graph terminology, cf. [32]. Consider a 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.
Lemma 16. Let \( \{(v_{i-1}, v_i)\}, i = 1, \ldots, m \). 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 graph \( G_{\Sigma,\varphi} \) 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 \( \infty \) if no such path exists.

Example 14. Let \( \Sigma = \{\sigma_1, \sigma_2\} \) from Example 10, and let \( \varphi \) be \( \text{card}(\_'.season, \langle\text{month.match,}\langle\text{home.S,away.S}\rangle\rangle) \leq 5 \). The cardinality graph of \( \Sigma \) and \( \varphi \) is illustrated in the right picture of Fig. 4. Let \( v \) denote the unique season-node, \( w \) the unique match-node, and \( u \) the unique home-node in the right picture of Fig. 4. Then \( d(v, w) = 6 \) and \( d(v, u) = \infty \).

The following observation is now crucial. If the distance \( d(q_\varphi, q'_\varphi) \) from \( q_\varphi \) to \( q'_\varphi \) in \( G_{\Sigma,\varphi} \) is at most \( \max_{\varphi} \), then \( \varphi \in \Sigma^+ \). In other words, if \( \varphi \) is not derivable from \( \Sigma \), then every path from \( q_\varphi \) to \( q'_\varphi \) in \( G_{\Sigma,\varphi} \) has distance at least \( \max_{\varphi} + 1 \). For the remainder of this section, we will be concerned with proving this observation, i.e., Lemma 18. More specifically we want to show the following: if \( d(q_\varphi, q'_\varphi) \leq \max_{\varphi} \) in \( G_{\Sigma,\varphi} \), then \( \varphi \in \Sigma^+ \). Since \( G_{\Sigma^+,\varphi} \) contains all the edges of \( G_{\Sigma,\varphi} \) the following holds: if \( d(q_\varphi, q'_\varphi) \leq \max_{\varphi} \) in \( G_{\Sigma,\varphi} \), then \( d(q_\varphi, q'_\varphi) \leq \max_{\varphi} \) in \( G_{\Sigma^+,\varphi} \). In fact, we will show that if \( d(q_\varphi, q'_\varphi) \leq \max_{\varphi} \) in \( G_{\Sigma^+,\varphi} \), then \( \varphi \in \Sigma^+ \). That is, we may assume from now on that \( G_{\Sigma^+,\varphi} = G_{\Sigma,\varphi} \), i.e., \( \Sigma \) is closed under inferences.

Our first lemma says that if there is a witness edge \( (w_\sigma, w'_\sigma) \) in \( G_{\Sigma,\varphi} \) that results from the applicability of some \( \sigma \in \Sigma^+ \) to \( \varphi \), then for each node \( w \) between \( w_\sigma \) and \( w'_\sigma \) in \( T_{\Sigma,\varphi} \) there is also a witness edge \( (w, w'_\sigma) \) in \( G_{\Sigma,\varphi} \) with \( \omega(w_\sigma, w'_\sigma) = \omega(w, w'_\sigma) \) that results from the applicability of some \( \sigma' \in \Sigma^+ \) to \( \varphi \). This is illustrated in Fig. 5.

Lemma 15. Let \( \Sigma \cup \{\varphi\} \) be a finite set of numerical keys in \( \mathcal{N} \), and let \( \sigma \in \Sigma^+ \). Suppose further that \( w_\sigma \) and \( w'_\sigma \) witness the applicability of \( \sigma \) to \( \varphi \). For each descendant node \( w \) of \( w_\sigma \) in \( T_{\Sigma,\varphi} \) that is also an ancestor of \( w'_\sigma \) in \( T_{\Sigma,\varphi} \) there is some \( \sigma' \in \Sigma^+ \) such that \( w \) and \( w'_\sigma \) witness the applicability of \( \sigma' \) to \( \varphi \), and where \( \max_{\sigma'} = \max_{\sigma^\prime} \).

Proof. The lemma is an immediate consequence of the target-to-context rule. \( \square \)

The next lemma shows that a witness edge that ends in a descendant node of \( q'_\varphi \) implies the existence of another witness edge that ends in \( q'_\varphi \). The latter witness edge results from a numerical key \( \sigma' \in \Sigma^+ \) whose set of key paths coincides with the set of key paths of \( \varphi \). The situation is depicted in Fig. 6.

Lemma 16. Let \( \Sigma \cup \{\varphi\} \) be a finite set of numerical keys in \( \mathcal{N} \), and let \( \sigma \in \Sigma^+ \) be applicable to \( \varphi \). Suppose further that \( w_\sigma \) and \( w'_\sigma \) witness the applicability of \( \sigma \) to \( \varphi \), and that there is a PL expression \( O \) and a PLs expression \( P \) such that \( O.P \subseteq Q'_\varphi \) with \( q'_\varphi \in w_\sigma [O] \) and \( w'_\sigma \in q'_\varphi [P] \). Then \( \text{card} (Q_\sigma, (O, \left\{ P_1, \ldots, P_k \right\})) \leq \max_{\sigma} \in \Sigma^+ \).

Proof. First we note that the case where \( Q'_\varphi = \varepsilon \) is trivial because of the epsilon and the weakening rules. In the following, we distinguish two different cases.
Case 1. Suppose that all key paths of \( \varphi \) are different from \( \varepsilon \), i.e., for all \( i = 1, \ldots, k_{\varphi} \) we have \( P_{i}^{\varphi} \neq \varepsilon \). Then, just the leaves of \( T_{\Sigma, \varphi} \) are marked. We discuss two subcases.

Case 1(a). Suppose that \( P = \varepsilon \). That is, \( O \subseteq Q'_{\varphi} \) and \( w_{\sigma} = q'_{\varphi} \). Since \( w_{\sigma} \) and \( w_{\sigma}' \) witness the applicability of \( \sigma \) to \( \varphi \) we know that for all \( j = 1, \ldots, k_{\sigma} \) there is some \( i \) with \( 1 \leq i \leq k_{\varphi} \) such that \( P_{j}^{\sigma} = P_{i}^{\varphi} \). Therefore, we obtain the following inference:

\[
\begin{align*}
\text{card} \left( Q_{\sigma} \left( Q'_{\varphi}, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ P_{1}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma}
\end{align*}
\]

in which we first apply the target-path-containment rule, and then the superkey rule.

Case 1(b). Suppose that \( P \neq \varepsilon \), i.e., \( w_{\sigma} = q'_{\varphi} \) where \( P \) is the prefix of \( P_{i}^{\varphi} \) for some \( i \) with \( 1 \leq i \leq k_{\varphi} \). Due to the definition of applicability it follows that for all \( j = 1, \ldots, k_{\sigma} \) we have \( P_{j}^{\sigma} = P_{i}^{\varphi} \). Since \( P_{i}^{\varphi} \) and \( P_{j}^{\sigma} \) are PLs expressions for all \( j = 1, \ldots, k_{\sigma} \), it follows that all \( P_{i}^{\sigma} \) coincide with one another, i.e., \( k_{\sigma} = 1 \). Therefore, we obtain the following inference:

\[
\begin{align*}
\text{card} \left( Q_{\sigma} \left( Q'_{\varphi}, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ P_{1}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma}
\end{align*}
\]

in which we first use the fact that \( k_{\sigma} = 1 \), and then apply the target-path-containment rule, followed by the subnodes and then the superkey rule.

Case 2. Suppose now that there is a key path of \( \varphi \) which is \( \varepsilon \), say \( P_{l}^{\varphi} = \varepsilon \) for some \( l \) with \( 1 \leq l \leq k_{\varphi} \). Then, all descendant nodes of \( q'_{\varphi} \) in \( T_{\Sigma, \varphi} \) are marked. We discuss two subcases.

Case 2(a). Suppose that \( P = \varepsilon \). That is, \( O \subseteq Q'_{\varphi} \) and \( w_{\sigma} = q'_{\varphi} \). Since \( w_{\sigma} \) and \( w_{\sigma}' \) witness the applicability of \( \sigma \) to \( \varphi \) we know that for all \( j = 1, \ldots, k_{\sigma} \) there is some PLs expression \( P_{j}^{\sigma} \) and some \( i \) with \( 1 \leq i \leq k_{\varphi} \) such that \( P_{j}^{\sigma} = P_{i}^{\varphi} \). Therefore, we obtain the following inference:

\[
\begin{align*}
\text{card} \left( Q_{\sigma} \left( Q'_{\varphi}, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ \varepsilon, P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma}
\end{align*}
\]

in which we first apply target-path-containment, followed by superkey, \( k_{\sigma} \) applications of prefix-epsilon and then superkey.

Case 2(b). Suppose that \( P \neq \varepsilon \), i.e., \( w_{\sigma} = q'_{\varphi} \) where \( P \) is the prefix of \( P_{i}^{\varphi} \) for some \( i \) with \( 1 \leq i \leq k_{\varphi} \). Due to the definition of applicability it follows first that there is some \( j \) with \( 1 \leq j \leq k_{\sigma} \) such that for all \( m = 1, \ldots, k_{\sigma} \), \( P_{m}^{\sigma} \) is a prefix of \( P_{i}^{\varphi} \). Without loss of generality we assume that \( j = 1 \). Moreover, there is some PLs expression \( P_{1}^{\sigma} \) such that \( P_{1}^{\sigma} = P_{1}^{\varphi} \). Therefore, we obtain the following inference:

\[
\begin{align*}
\text{card} \left( Q_{\sigma} \left( Q'_{\varphi}, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma} \\
\text{card} \left( Q_{\sigma} \left( O, \{ \varepsilon, P_{1}^{\sigma}, \ldots, P_{k_{\sigma}}^{\sigma} \} \right) \right) & \leq \text{max}_{\sigma}
\end{align*}
\]

in which we first use the fact that \( k_{\sigma} = 1 \), and then apply the target-path-containment rule, followed by the subnodes and then the superkey rule.
in which we first apply the target-path-containment rule, then the superkey rule to introduce \( \varepsilon \), followed by the prefix-epsilon rule until all \( P_j^e \) are extended to equal \( P_j^1 \), followed by the subnodes-epsilon, the prefix-epsilon rule again, and then the superkey rule. \( \square \)

Before proving Lemma 18 we show another technical lemma. If there is a witness edge \((w_1, w_2')\) with weight \( \max_{\sigma_1} \) and another witness edge \((w_2', q_\phi)\) with weight \( \max_{\sigma_2} \), then there is also a witness edge \((w_1, q_\phi)\) with weight \( \max_{\sigma_1} \cdot \max_{\sigma_2} \).

The situation is depicted in Fig. 7.

**Lemma 17.** Let \( \Sigma \cup \{ \phi \} \) be a finite set of numerical keys in \( \mathcal{N} \), and let \( \sigma_1, \sigma_2 \in \Sigma \) where \( \sigma_2 = \text{card} \left( Q_{\sigma_2}, \left( Q_{\sigma_2}', \left\{ P_1^e, \ldots, P_k^e \right\} \right) \right) \leq \max_{\sigma_2} \). Suppose further that \((w_1, w_2')\) witnesses the applicability of \( \sigma_1 \) to \( \phi \), and \((w_2', q_\phi)\) witnesses the applicability of \( \sigma_2 \) to \( \phi \). Then there is some \( PL_\phi \) expression \( P \) such that \( \sigma' = \text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \left\{ P_1^e, \ldots, P_k^e \right\} \right) \right) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+ \) and \((w_1, q_\phi)\) witnesses the applicability of \( \sigma' \) to \( \phi \).

**Proof.** Since \( w_1 \) is an ancestor node of \( q_\phi \), and all the key paths are \( PL_\phi \) expressions, the applicability of \( \sigma_1 \) to \( \phi \) implies that the key paths of \( \sigma_1 \) have a common prefix \( P \), i.e., there is some \( PL_\phi \) expression \( P \) such that (i) for all \( j = 1, \ldots, k_\phi \) there is some \( PL_\phi \) expression \( P_j^{e,1} \) such that \( P_j^{e,1} \) is a key path of \( \sigma_1 \), (ii) \( P \subseteq Q_{\sigma_2} \) and (iii) \( q_\phi \in w_1 \cdot I_P \). That is, \( \sigma_1 = \text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \left\{ P_1^e, \ldots, P_k^e \right\} \right) \right) \leq \max_{\sigma_1} \in \Sigma^+ \).

An application of the superkey rule results in

\[
\text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', \mathcal{S} \right) \right) \leq \max_{\sigma_1} \in \Sigma^+ \tag{2}
\]

where \( \mathcal{S} = \left\{ P, P_1^e, \ldots, P_k^e, P_1 P_1^{e,1}, \ldots, P_k P_k^{e,1} \right\} \).

According to applicability we have \( Q_{\sigma_1}, Q_{\sigma_1}' \subseteq Q_{\sigma_2} \) and by (ii) applications of the context-path-containment, target-path-containment and superkey rule to \( \sigma_2 \) give

\[
\text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \mathcal{F} \right) \right) \leq \max_{\sigma_2} \in \Sigma^+ \tag{3}
\]

where \( \mathcal{F} = \left\{ P_1 P_1^{e,1}, \ldots, P_k P_k^{e,1} \right\} \). From (2) and (3) we infer

\[
\text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \mathcal{F}' \right) \right) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+ \tag{4}
\]

by means of the multiplication rule. We now distinguish between two different cases.

**Case 1.** Assume first that for all \( i = 1, \ldots, k_\phi \) we have \( P_i^e \neq \varepsilon \). It follows, by applicability of \( \sigma_1 \) to \( \phi \), that for all \( j = 1, \ldots, k_\phi \) there is some \( i \) with \( 1 \leq i \leq k_\phi \) such that \( P_i^e = P_j^{e,1} \). Therefore, (4) reduces to

\[
\text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \left\{ P_1^{e,1}, \ldots, P_k^{e,1} \right\} \right) \right) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+ .
\]

**Case 2.** For the remaining case we suppose that there is some \( i \) with \( 1 \leq i \leq k_\phi \) such that \( P_i^e = \varepsilon \). Applicability of \( \sigma_1 \) to \( \phi \) means that for all \( j = 1, \ldots, k_\phi \) there is some \( PL_\phi \) expression \( P_j \) and some \( i \) with \( 1 \leq i \leq k_\phi \) such that \( P_i^e = P_j^{e,1} P_j^1 \). A repeated application of the prefix-epsilon rule to (4) results in \( \text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \mathcal{F}' \right) \right) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+ \) where \( \mathcal{F}' = \left\{ P_1^{e,1}, \ldots, P_k^{e,1}, P_1 P_1^{e,1}, \ldots, P_k P_k^{e,1} \right\} \). According to the key path equalities this reduces to

\[
\text{card} \left( Q_{\sigma_1}, \left( Q_{\sigma_1}', P, \left\{ P_1^{e,1}, \ldots, P_k^{e,1} \right\} \right) \right) \leq \max_{\sigma_1} \cdot \max_{\sigma_2} \in \Sigma^+ .
\]

Hence, in both cases we derive that \( \sigma' \in \Sigma^+ \). It is immediate that \( \sigma' \) is applicable to \( \phi \) as witnessed by \( w_{\sigma_1} \) and \( q_\phi' \). \( \square \)
We will now use the previous lemmata to prove Lemma 18. The individual proof steps are illustrated in Fig. 8. If there is a simple path $D$ from $q_1$ to $q_n$ in $G_{\Sigma^+}$, then it has the form as illustrated in the leftmost picture. This is due to the definition of the cardinality graph and its upward and witness edges. The existence of the witness edges in $D$ implies the existence of other witness edges in $G_{\Sigma^+}$. In a first step, we conclude by Lemma 16 that the final witness edge of $D$ can be replaced by a witness edge that ends in $q_n$, resulting in a new simple path $D'$ illustrated in the second picture from the left. Subsequently, we apply Lemma 15 to show the existence of a simple path $D''$ in $G_{\Sigma^+}$ as depicted in the third picture from the left. Finally, Lemma 17 is applied to show the existence of a single witness edge $(q_1, q_n)$ whose weight is that of the original path $D$ and which results from a numerical key $\sigma$ in $\Sigma^+$. This is illustrated in the right picture. It follows then by applicability of $\sigma$ to $\varphi$ that $\varphi$ is indeed derivable from $\Sigma$.

**Lemma 18.** Let $\Sigma \cup \{\varphi\}$ be a finite set of numerical keys in $\Lambda^*$. If $d(q_1, q_n) \leq \max_{\varphi}$ in the cardinality graph $G_{\Sigma^+}$, then \( \text{card}\left(Q_0, (Q_0, \{P_{n_1}^\varphi, \ldots, P_{n_k}^\varphi\})\right) \leq \max_{\varphi} \in \Sigma^+ \).

**Proof.** As discussed at the beginning of this section, we assume without loss of generality that $G_{\Sigma^+} = G_{\Sigma^+}^{\Sigma^+}$, i.e., $G_{\Sigma^+}$ already contains witness edges and weights that result from applicable numerical keys that can be inferred from $\Sigma$.

Due to the infinity rule there is nothing to show if $\max_{\varphi} = \infty$. Assume $\max_{\varphi} < \infty$. If $d(q_1, q_n) \leq \max_{\varphi}$, then let $D$ denote the simple path in $G_{\Sigma^+}$ from $q_1$ to $q_n$ with $\omega(D) = d(q_1, q_n)$. According to the definition of the cardinality graph we can assume without loss of generality that $D$ consists of a sequence $\pi_1, \ldots, \pi_{n+1}$, $n \geq 1$, where for each $i = 1, \ldots, n$, $\pi_i$ starts with a possibly empty sequence of upward edges each of weight 1 followed by a single witness edge $(w_{\pi_i}, w'_{\pi_i})$ labelled with $\max_{\pi_i}$ where $w_{\pi_i}$ and $w'_{\pi_i}$ witness the applicability of $\pi_i$ to $\varphi$, and $\pi_{n+1}$ is a possibly empty sequence of upward edges labelled with 1. Moreover, we can assume that $q_1, w'_{\pi_1}, \ldots, w'_{\pi_{n+1}}$ form a proper descendant chain, $q_n$ is a proper descendant of $w'_{\pi_{n+1}}$ and $w'_{\pi_{n+1}}$ is a descendant node of $q_n$ in $T_{\Sigma^+}$. This situation is depicted in the top left of Fig. 8.

Next we note that $\sigma_n$ satisfies the assumptions of Lemma 16. We can therefore assume that $D = D'$ as illustrated in the second picture from the left of Fig. 8. That is, we can assume without loss of generality that $\sigma_{n+1}$ is indeed an empty sequence and $w'_{\pi_n} = q_n$ where the set of key paths of $\sigma_n$ is $\{P_{n_1}^\varphi, \ldots, P_{n_k}^\varphi\}$.

We now apply Lemma 15 to conclude that there is a simple path $D''$ in $G_{\Sigma^+}$ from $q_1$ to $q_n$ and $\omega(D) = \omega(D'')$. In fact, $D''$ consists of the sequence $\pi_1, \ldots, \pi_n$ where each $\pi_i$ with $1 \leq i \leq n$ consists of a single witness edge $(w_{\pi_i}, w'_{\pi_i})$ labelled with $\max_{\pi_i}$ and where $w'_{\pi_i} = w_{\pi_{i+1}}$ for $i = 1, \ldots, n - 1$ and $w_{\pi_1} = q_1$ and $w_{\pi_n} = q_n$. Again, $q_1, w'_{\pi_1}, \ldots, w'_{\pi_{n+1}}$ form a proper descendant chain. $D''$ is illustrated in the third picture from the left of Fig. 8.

At this stage we apply Lemma 17 repeatedly to conclude that there is a single witness edge $D_0 = (q_1, q_n)$ in $G_{\Sigma^+}$ resulting from the numerical key

$$
\sigma = \text{card}\left(Q_0, (Q_0, \{P_{n_1}^\varphi, \ldots, P_{n_k}^\varphi\})\right) \leq \prod_{i=1}^n \max_{\pi_i} \in \Sigma^+
$$

that is applicable to $\varphi$. Due to the applicability of $\sigma$ to $\varphi$ we conclude that $Q_0 \subseteq Q_0'$ and $Q_0' \subseteq Q_0'$. We can now apply the context-path-containment and target-path-containment rule to obtain
Fig. 9. A counter-example tree for the implication of \( \varphi \) by \( \Sigma = \{ \sigma_1, \sigma_2 \} \) from Example 10.

\[
\text{card} \left( Q_{\varphi}, \left( Q'_{\varphi}, \left[ P^1_{\varphi}, \ldots, P^n_{\varphi} \right] \right) \right) \leq \prod_{i=1}^{n} \max_{\alpha_i} \in \Sigma^+.
\]

Since

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

holds applications of the weakening rule show that also

\[
\text{card} \left( Q_{\varphi}, \left( Q'_{\varphi}, \left[ P^1_{\varphi}, \ldots, P^n_{\varphi} \right] \right) \right) \leq \max_{\varphi} \in \Sigma^+
\]

holds which proves the lemma. \( \square \)

**Example 19.** Let \( \Sigma = \{ \sigma_1, \sigma_2, \varphi, \psi \} \) and \( w \) be as in Example 14, and recall that \( d(\psi, w) = 6 \) in \( G_{\Sigma, \varphi} \). Note that \( \text{card}(\psi.\text{season}, (\text{month}.\text{match}, (\text{home}.S, \text{away}.S))) \leq 6 \) is derivable from \( \Sigma \) (by a single application of the multiplication rule to \( \sigma_1 \) and \( \sigma_2 \)). Moreover, Fig. 9 shows that \( \varphi \) is not implied by \( \Sigma \) and thus not derivable from \( \Sigma \) according to the soundness of our inference rules. Note that \( d(\psi, w) \) is equal to \( \max_{\psi} + 1 \).

We are now ready to prove the completeness of the inference rules.

**Theorem 20.** The inference rules in Table 1 are complete for the implication of numerical keys in \( \mathcal{N}^\ast \).

**Proof.** Let \( \Sigma \cup \{ \varphi \} \) be a finite set of numerical keys in \( \mathcal{N}^\ast \) such that \( \varphi \notin \Sigma^+ \). We construct a finite XML tree \( T \) which satisfies all numerical keys in \( \Sigma \) but does not satisfy \( \varphi \). Since \( \varphi \notin \Sigma^+ \) every existing path from \( q_0 \) to \( q_\varphi \) 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 \( q_0 \) 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 \( \omega'(q_0) > \max_{\varphi} \). Let \( T_0 \) be a copy of the path from the root node \( r \) to \( q_\varphi \) in \( T_{\Sigma, \varphi} \). We extend \( T_0 \) as follows: for each node \( n \) on the path from \( q_0 \) to \( q_\varphi \) in \( T_{\Sigma, \varphi} \) we introduce \( \omega'(n) \) copies \( n_1, \ldots, n_{\omega'(n)} \) into \( T_0 \). Suppose \( T_0 \) has been constructed down to the level of \( u_1, \ldots, u_{\omega'(u)} \) 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 \( i = 1, \ldots, \omega'(u) \) and all \( j = 1, \ldots, \omega'(v) \) 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 \( i = 1, \ldots, \omega'(q_\varphi) \) let \( T_i \) be a node-disjoint copy of the subtree of \( T_{\Sigma, \varphi} \) rooted at \( q_\varphi \). 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 \( \mathcal{L} - (\mathcal{L} - \Sigma, \emptyset) \) to achieve this. The counterexample tree \( T \) is obtained from \( T_0, T_1, \ldots, T_{\omega'(q_\varphi)} \) by identifying the leaf node \( q_\varphi \) of \( T_0 \) with the root node of \( T_i \) for all \( i = 1, \ldots, \omega'(q_\varphi) \). We conclude that \( T \) violates \( \varphi \) since \( \omega'(q_\varphi) > \max_{\varphi} \), and our construction guarantees that \( T \) satisfies \( \Sigma \). \( \square \)

The construction of \( T \) in the proof of Theorem 20 is illustrated in Fig. 9 where \( m \) denotes the unique \( \text{month} \)-node, and \( q' \) denotes the unique \( \text{match} \)-node in \( T_{\Sigma, \varphi} \), respectively.
5. Deciding implication of numerical keys

We may use the axiomatisation established in Theorem 20 to enumerate all implicitly specified numerical keys. This can assist the data administrator in validating explicitly specified knowledge, or searching for implicit numerical keys to optimise queries or generate views for a more efficient way of processing common types of queries or updates. In practice, it also occurs quite often that the administrator is interested whether a specific numerical key is implied by a set of explicitly specified numerical keys. For this particular purpose the enumeration method is not suitable since one needs efficient means for deciding the implication problem of numerical keys.

In this section, we extend our technique of proving completeness to obtain an algorithm for deciding numerical key implication in time quadratic in the size of the constraints. This significantly generalises the algorithm proposed for deciding the implication problem of numerical keys.

In this section, we discuss an efficient implementation of Algorithm 1 and analyse its time complexity. The

Algorithm 1 is correct.

5.2. Time complexity of the algorithm

In this section, we discuss an efficient implementation of Algorithm 1 and analyse its time complexity. The length $|Q|$ of a PL expression $Q$ is the number of symbols (labels or wildcards) in the normal form of $Q$, cf. [9]. Further, let the length $|\varphi|$ of a numerical constraint $\varphi$ be the sum of the lengths of all PL expressions in $\varphi$, i.e., $|\varphi| = |Q_1| + \sum_{i=1}^{k} |Q^*_i|$. The shortest path starting from a fixed node in a digraph $G$ can be found in time quadratic in the number of nodes of $G$ using Dijkstra's algorithm, cf. [32]. Since the number of nodes in $G_{\Sigma,\varphi}$ is just the length of $\varphi$ plus 1, step (2) of Algorithm 1 can be executed in time $O(|\varphi|^2)$. It therefore remains to investigate the time complexity for generating the cardinality graph $G_{\Sigma,\varphi}$ from $\Sigma$ and $\varphi$.

To construct the cardinality graph $G_{\Sigma,\varphi}$ one needs to determine its witness edges. The naive approach for this is the following: for each $\sigma \in \Sigma$ find all witness pairs in $T_{\Sigma,\varphi}$ that arise from $\sigma$; for each of these witness pairs check whether it already occurs as a witness edge in $G_{\Sigma,\varphi}$; if not then insert it with weight $\max_\sigma$; otherwise check whether its current weight is less or equal to $\max_\sigma$, and update its weight to $\max_\sigma$ if not.
As outlined above, $\Sigma$ implies the numerical key $\varphi$ if and only if the distance between $q_\varphi$ and $q'_\varphi$ in the cardinality graph $G_{\Sigma,\varphi}$ is at most $\max_k$. We shall now show that for computing the distance from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$, we do not actually need to consider all witness pairs.

Firstly, we do not need to consider witness pairs $(w, w')$ where the node $w$ is a descendant node of $q'_\varphi$ in $T_{\Sigma,\varphi}$:

**Lemma 24.** Let $p$ be a simple path from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$ that contains a witness edge $(w, w')$ such that $w$ is a descendant node of $q'_\varphi$ in $T_{\Sigma,\varphi}$. Then there exists a simple path from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$ that has at most the weight of $p$ and that does neither contain $(w, w')$ nor any witness edge not in $p$.

Note that a witness pair may well arise from more than just one numerical key in $\Sigma$. When inspecting the witness pairs for $\sigma \in \Sigma$, it suffices to consider those ones whose weight in $G_{\Sigma,\varphi}$ is actually determined by $\sigma$, that is, equals $\max_k$. Moreover, if there are two witness pairs $(v, w')$ and $(w, w')$ arising from the same $\sigma \in \Sigma$ such that $v$ is a proper ancestor of $w$ in $T_{\Sigma,\varphi}$, we do not need to consider $(w, w')$:

**Lemma 25.** Let $p$ be a simple path from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$ that contains a witness edge $(w, w')$, and let $v$ be a proper ancestor of $w$ in $T_{\Sigma,\varphi}$ such that $(v, w')$ is a witness edge arising from the same numerical key $\sigma$ as $(w, w')$. Then the weight of $(w, w')$ in $G_{\Sigma,\varphi}$ is strictly less than $\max_k$, or there exists a simple path from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$ that has at most the weight of $p$ and that does neither contain $(w, w')$ nor any witness edge not in $p$, except for possibly $(w', v)$.

The observations above give rise to a slightly more efficient approach for determining the witness edges $(w, w')$ in $G_{\Sigma,\varphi}$ that are eventually needed to compute the distance from $q_\varphi$ to $q'_\varphi$. Consider a numerical key $\varphi \in \Sigma$. Let $W'_\varphi$ be the set of all nodes $w'$ in $T_{\Sigma,\varphi}$ for which there exists some node $w$ in $T_{\Sigma,\varphi}$ such that $w$ and $w'$ witness the applicability of $\varphi$. Further, for each $w' \in W'_\varphi$, let $W_\varphi(w')$ be the set of all nodes $w$ in $T_{\Sigma,\varphi}$ such that $w$ and $w'$ witness the applicability of $\varphi$. The witness edges in $G_{\Sigma,\varphi}$ are just the pairs $(w, w')$ with $w' \in W'_\varphi$ and $w \in W_\varphi(w')$. Due to Lemmas 24 and 25, we do not need all witness edges, and therefore it is not actually necessary to determine the entire set $W_\varphi(w')$ for each $w' \in W'_\varphi$. Rather we can restrict ourselves to the top-most ancestor of $q'_\varphi$ in $T_{\Sigma,\varphi}$ that belongs to $W_\varphi(w')$, which we denote by $w^{\text{top}}_\varphi(w')$ (if it exists).

We proceed in two steps. First we determine $W'_\varphi$, and afterwards, for each $w' \in W'_\varphi$, we determine $w^{\text{top}}_\varphi(w')$ (if it exists). By definition, $W'_\varphi$ consists of all nodes $w' \in \{Q_\varphi, Q'_\varphi\}$ in $T_{\Sigma,\varphi}$ such that, for each $i = 1, \ldots, k$, there is a marked node in $w'[P_i^\varphi]$. Note that $\{Q_\varphi, Q'_\varphi\}$ is a Core XPath [19] query, and recall that a Core XPath query $\psi[P]$ can be evaluated on a node-labelled tree $T$ in $\mathcal{O}(|T| \times |P|)$ time. Hence, $\{Q_\varphi, Q'_\varphi\}$ can be determined in $\mathcal{O}(\max_k \times |Q_\varphi| \times |Q'_\varphi|)$ time.

Now, fix some $i = 1, \ldots, k$. Let $v$ be a marked node in $T_{\Sigma,\varphi}$, and let $u$ denote the ancestor of $v$ that resides $|P_i^\varphi|$ levels atop of $v$ in $T_{\Sigma,\varphi}$ (if it exists). Recall that the level of a node in a tree is the length of the unique path from the root node of the tree to the node. We can then check whether $v \in u[P_i^\varphi]$, that is, whether the unique path from $u$ to $v$ is a $P_i^\varphi$-path. This can be done in $\mathcal{O}(\min(|P_i^\varphi|, |v|))$ time, since $P_i^\varphi$ is a $P_k$ expression. By inspecting all nodes $v \in \mathcal{N}$, we obtain the set $U_i^\varphi$ of all nodes $u$ in $T_{\Sigma,\varphi}$ for which $u[P_i^\varphi] \cap \mathcal{N} \neq \emptyset$. Overall, this takes $\mathcal{O}(\max_k \times |P_i^\varphi|)$ time.

By definition, $W_\varphi$ is the intersection of $\{Q_\varphi, Q'_\varphi\}$ with the sets $U_i^\varphi$, $i = 1, \ldots, k$. Hence, $W_\varphi$ can be determined in $\mathcal{O}(\max_k \times |P_i^\varphi|)$ time, and thus in $\mathcal{O}(\max_k \times |\sigma|)$ time.

It remains to determine $w^{\text{top}}_\varphi(w')$ for each $w' \in W'_\varphi$ (if it exists). If $Q'_\varphi$ is a $P_k$ expression, then $w^{\text{top}}_\varphi(w')$ is the node $|Q'_\varphi|$ levels atop of $w'$ in $T_{\Sigma,\varphi}$ and we are done. Otherwise $Q'_\varphi$ contains a $\ast$, and thus has the form $A_-^\ast B$ where $A$ is a $P_k$ expression and $B$ is a $P_l$ expression.

**Lemma 26.** Let $(w, w')$ be a witness edge arising from $\sigma$, and let $v \in \{Q_\varphi, Q'_\varphi\}$ be a proper ancestor of $w$ in $T_{\Sigma,\varphi}$ such that $\psi[A]$ is non-empty. Then $(v, w')$ is a witness edge arising from $\sigma$, too.

By Lemma 26, we derive that $w^{\text{top}}_\varphi(w')$ is the top-most ancestor $w$ of $q'_\varphi$ in $T_{\Sigma,\varphi}$, that belongs to $\{Q_\varphi, Q'_\varphi\}$ and for which $\psi[A]$ is non-empty. In particular, $w^{\text{top}}_\varphi(w')$ is independent from the choice of $w'$ in $W'_\varphi$. We will denote this node by $w^{\text{top}}_\varphi$ (if it exists).

To determine $w^{\text{top}}_\varphi$, we first determine the set $\{Q_\varphi, A\}$ of nodes in $T_{\Sigma,\varphi}$. Note that $\{Q_\varphi, A\}$ is a Core XPath query [19], and thus can be evaluated in $\mathcal{O}(\max_k \times |Q_\varphi, A|)$ time. If $\{Q_\varphi, A\}$ is non-empty, we choose a top-most node $v$, and consider the node $w$ that resides $|A|$ levels atop of $v$ in $T_{\Sigma,\varphi}$. If $w$ is a proper ancestor of $q'_\varphi$, then $w$ is the node $w^{\text{top}}_\varphi$ we are looking for. Otherwise, $w^{\text{top}}_\varphi$ does not exist, and thus $w^{\text{top}}_\varphi(w')$ does not exist for any $w' \in W'_\varphi$. Overall, the determination of $w^{\text{top}}_\varphi$ takes $\mathcal{O}(\max_k \times |Q_\varphi, A|)$ time, and thus $\mathcal{O}(\max_k \times |\sigma|)$ time.

In conclusion, both steps need $\mathcal{O}(\max_k \times |\sigma|)$ time each. Hence, it takes us $\mathcal{O}(\max_k \times |\sigma|)$ time to determine all the witness edges arising from $\sigma$ that are needed for computing the distance from $q_\varphi$ to $q'_\varphi$ in $G_{\Sigma,\varphi}$. The length $\max_k$ of a finite set $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ of numerical keys in $\mathcal{N}$ is defined as $\sum_{i=1}^n |\sigma_i|$.

**Theorem 27.** The implication problem $\Sigma \models \varphi$ for the class $\mathcal{N}$ of numerical keys can be decided in $\mathcal{O}(\max_k \times (|\Sigma| + |\varphi|))$ time.
6. Some applications of numerical constraints

The introduction has already illustrated how several XML-related W3C standards can benefit from the specification of numerical keys. In particular, we have seen examples for predicting the number of query answers for XPath and XQuery queries, and optimising such queries. In this section, we will show some further potential applications.

6.1. Approximating update and encryption costs

Instead of predicting the number of query answers we could also predict the number of updates, using for instance the XQuery update facility [12]. For example, the XQuery query

\[
\text{for } s \text{ in doc("enrol.xml")/year[@calendar="2007"]/semester[@no="2"]/student[@sid="247"]
return do replace value of $s/last with "Milhouse"
}\]

will update the lastname of the student having sid-value “247” to the new text value “Milhouse” in all course enrolments of this student in semester two of 2007. If the database management system is able to conclude that \(\text{card(_*_.semester, (_*_.student, \{sid\}))} \leq 4\) is implied by the set \(\Sigma\) of numerical keys specified by the data designer, then the maximal number of updates this query causes is 4.

When XML data is exchanged over the Web it is very common that sensitive information is encrypted, e.g., by XML encryption [31]. In order to evaluate queries on encrypted XML data it may become necessary to decrypt certain data element nodes in order to return the relevant information in the answer. Recall the XQuery query (1) from the introduction. Assume that the sid-attributes (among others) have been encrypted in order to hide what grade students received. Note that by not encrypting the grade-element students can still be informed about the distribution of the grades for a course. A database management system capable of inferring that both \(\text{card(_*_.year, (_*_.student, \{sid\}))} \leq 8\) and \(\text{card(_*_.student, \{grade, \emptyset\})} \leq 1\) are implied by the constraints specified can also predict that the number of necessary decryptions to answer this query is at most 8, and thus also predict the time necessary to deliver the information requested.

6.2. Indices

As a further application we look at indices that are commonly used to accelerate specific queries. The selection of indexes is an important task when tuning a database. Although already NP-hard for relational databases [37] XML queries pose additional challenges to the index selection problem since both structure and content need to be covered simultaneously, cf. [21]. Suppose the next type of XPath queries are of interest to our application:

\[
/year[@calendar="2007"]/course/teacher="Principal Skinner"
and student/@sid="007"/name
\]

That is, course names are selected according to a specific year in which the course is taught, a specific teacher who delivers the course and a specific student enrolled in that course. Such a query would call for a multi-key index where the first index is built on the values on \(/_year/@calendar\). The problem in this scenario is whether the second index will be built on teacher-values or on student/@sid-values. Reasoning about the numerical keys specified by the data designer may result in the information that for each year there are at most 100 teachers delivering courses in this year and for each of these teachers there are up to 500 students enrolled in courses this teacher delivers. On the other hand one might be able to infer that for each year there are up to 5000 students enrolled in that year’s courses and each of these students may be taught by up to 8 different teachers. In this case, the second index should be based on teacher-values leaving the student/@sid-values as the third index. This example illustrates how the specification of numerical keys and the ability to reason about them can potentially reduce the number of choices for the index selection problem.

6.3. Views and query rewriting

As a last application we demonstrate the potential of numerical constraints for generating XML views to efficiently process common types of queries and updates. Recall the numerical constraint that each year-node subsumes up to eight course-nodes that contain student/@sid-descendants with the same value. It could be rather expensive to query the original XML tree for course information based on a specific year-value and a specific sid-value since for all course-nodes (in the worst case) it has to be decided whether they have an sid-descendant with that particular value. Querying becomes even more inefficient when some of this information is encrypted.

Instead, one may create the XML view in Fig. 10 (showing the fragment for Bart Simpson only), rewrite query (1) and use the XML view to evaluate the resulting query.

On the XML view the constraint above translates into the following condition: under each student-node there are up to eight course-nodes independently from any data. Since students can be identified by their sid-value relatively to the year in
this XML view it is therefore better to pose queries on course information based on a specific year and a specific sid against this XML view. Then query (1) is rewritten into the following query:

\[
\text{for } \$c \text{ in } \text{doc("view.xml")/ /year[@calendar="2007"]/student[@sid="007"]/course}
\]

\[
\text{return}\langle \text{grade}\rangle \$c/\text{grade}\langle /\text{grade}\rangle
\]

The selection of student-elements based on their sid in early location steps achieves a better performance. Thus, the creation of XML views based on numerical constraints may lead to simplified integrity checking, and more efficient processing of common queries and updates.

7. Conclusion and future work

We have introduced the class of numerical constraints for XML. These constraints are naturally exhibited by XML data since they represent restrictions that occur in everyday life. Numerical constraints can specify such bounds either absolutely, i.e., for the entire application domain, or relatively to a certain context in the domain of interest. Numerical constraints can express more properties than other classes of constraints, e.g., XML keys where the upper bound is fixed to 1 [10,28,29,25], and generalised participation constraints in conceptual databases [24]. Moreover, we have illustrated that many XML applications can benefit from the specification of numerical constraints. In order to unlock these application domains effectively we have investigated decision problems associated with this class of constraints. While reasoning about numerical constraints is intractable in general we have identified the large subclass of numerical keys that are finitely satisfiable, finitely axiomatisable, and whose implication problem can be decided in time quadratic in the input. We have established that the implication problem of numerical keys can be characterised as a shortest path problem in a suitable digraph. Thus, numerical keys form a very natural and robust class of XML constraints that can be utilised effectively by data designers, and the complexity of their associated decision problems indicates that they can be maintained efficiently by database systems for XML applications.

In particular, our axiomatisation may provide the basis for an algorithm that mechanically enumerates all implicitly specified numerical keys that are logical consequences of those ones that have been specified explicitly. Moreover, our algorithm for deciding numerical key implication provides an efficient and simple tool to assist the data administrator in making informed choices about implicitly derived knowledge. For instance, XQuery queries may be optimised due to some implicitly specified numerical key, or XML views may be generated to process common types of queries and updates more efficiently.

Clearly, the discussion above gives rise to a variety of topics for future research. First of all, one might want to study (numerical) keys in the presence of a schema specification such as a DTD or an XSD provided by the data designer. This is likely to be a challenging task as already observed and illustrated by examples for keys [10]: (numerical) keys can interact with content models and thus behave completely differently under such specifications.

Furthermore, the tree model for XML adopted from [9] for our investigation here leaves considerable freedom to data designers. To some extent this flexibility has been exploited when constructing XML trees in the proofs. For certain applications one might want to incorporate additional features as specified by the W3C standard of XML. As an example we mention...
the uniqueness of attributes: no element may possess two distinct attribute children with the same label. This requirement cannot be captured by (numerical) keys. However, it may be expressed by numerical constraints \( \text{card}(\ell_a, (\ell_a, \emptyset)) = (1, 1) \) with \( \ell_a \in A \).

Suppose we require the uniqueness of attributes as part of the XML tree model. Then the numerical keys \( \text{card}(\ell_a, (\ell_a, \emptyset)) \leq 1 \) with \( \ell_a \in A \) hold trivially. When adding a corresponding axiom to the inference rules in Table 1 we may obtain a result similar to Theorem 20. Indeed the counter-example trees constructed in the completeness proof respect the uniqueness of attributes: to see this consider such an XML tree \( T \) constructed for a set \( \Sigma \) of numerical keys and some numerical key \( \varphi \notin \Sigma^+ \).

Assume \( T \) contains an element node \( x' \) labelled \( \ell_{a} \) with two distinct attribute children \( y', z' \) labelled \( \ell_{a} \). Let \( x, y, \) and \( z \) be the nodes in the mini-tree \( T_{\Sigma, e} \) from which \( x', y', z' \) were copied. Let \( p \) be the path from the root node to \( x \) in \( T_{\Sigma, e} \). From \( \lambda(b(p)) \) we obtain the PL expression \( R \) when replacing every occurrence of the special label \( \ell_{a} \) by a wildcard. We know that \( R, \ell_{a} \) is one of \( Q, Q', P_1, \ldots, Q, Q', P_k \), say \( Q, Q', P_1 \). Further we know that \( R \) is a prefix of \( Q, Q' \) as otherwise \( x' \) cannot have two children with the same attribute label. We have two cases: (1) \( Q, Q' = R \) and \( P_1 = \ell_{a} \), and (2) \( Q, Q' = R \ell_{a} \) and \( P_1 = \emptyset \). In case (1) none of the other key paths \( P_1 \) starts with \( \ell_{a} \). Hence \( y \) and \( z \) denote the same node in the mini-tree, and by construction of \( T, y \) and \( z \) denote the same node of \( T \). In case (2) we have \( k = 1 \) and 1 \( P_1 = \emptyset \) since all \( Q, Q', P_k \) are valid PL expressions. Again we observe that \( y \) and \( z \) denote the same node of the mini-tree. The trivial numerical key \( \text{card}(R, (\ell_{a}, \emptyset))) \leq 1 \) is applicable to \( \varphi \), hence there is an edge \((x, y)\) of weight 1 in the cardinality graph \( \mathcal{G}_{\Sigma, e} \). Therefore, the number of \( x \)-copies and of \( y \)-copies in \( T \) are equal, and every \( x \)-copy has exactly one \( y \)-child. In particular, \( y \) and \( z \) the same node in \( T \). In both cases we find that the counter-example we constructed respects the uniqueness of attributes as desired.

Another area that warrants future research is the study of (numerical) keys that are defined on the basis of more expressive path languages. It should be noted that efficient reasoning about numerical constraints relies on the computational tractability of the containment problem for such path languages. For recent results on the containment problem the interested reader is referred to [6,15,35,36,46].

We would like to extend numerical dependencies (a generalisation of functional dependencies) from relational databases to XML. However, since the implication problem for relational numerical dependencies is computationally infeasible [20], future work will focus on identifying useful restricted classes of such XML constraints.

As already indicated one should further explore the impact of numerical constraints on various XML applications, in particular on query optimisation, query rewriting, numbering schemes and indexing techniques. These applications can already benefit greatly from incomplete sets of sound inference rules for the implication of numerical constraints. While reasoning may be intractable in general one may develop efficient decision algorithms for those subclasses syntactically defined by some set of sound inference rules.

In practice, violations to the specified constraints may occur, e.g., when the stored XML data or the specified constraints deviate from reality. Therefore, one needs approaches adequate for handling XML data fragments that are inconsistent with the numerical constraints specified. Such approaches include the specification of soft constraints in which violations are permitted but reported to the data designer [23], or consistent query answering [13] in which only those query answers are returned that are present in all repairs of the database.

Finally, the logical characterisation of dependency implication in relational and complex-value data models has unified seemingly disparate areas of interest [27,38]. It would be rewarding if similar equivalences could be established for classes of numerical constraints in XML.

Acknowledgments

This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

The first author is supported by a research grant of the Alfried Krupp von Bohlen and Halbach Foundation, administered by the German Scholars Organisation.

References


