Checking the Satisfiability of XML-Specifications

Dr. Harald Hiss

University Freiburg, D-79110 Freiburg, Germany,
`hiss@informatik.uni-freiburg.de`,
WWW home page: `http://dbis.informatik.uni-freiburg.de/index.php?person=hiss`

Abstract. New developments in databases build on XML-technologies. Concepts of the relations model are transferred to XML. This is not unproblematic, transfer of integrity constraints causes a problem. Some XML-specifications are unsatisfiable.
A deductive checker is presented. An extensive formalization developed with Isabelle integrates circular XML-specifications with an inductive method. These XML-specifications are unsatisfiable. The checker generates a representation with F-Logic that is checked with Florid. The correctness is proven.

1 Checking the Satisfiability

New developments in databases build on XML [BMP+06] technologies. Concepts of the relational [AHV95] model are transferred to XML. This is not unproblematic, transfer of integrity constraints causes a problem. Some XML-specifications are unsatisfiable.

A deductive checker for XML-specifications is presented. The complexity of the satisfiability is proven in [FL02]. Implication of relational integrity that is undecidable [CV85] is represented with XML-specifications. A transformation for model checking XML-specifications is presented in [His07]. The transformation generates constraints. A model checker proves the satisfiability of the constraints. An extensive formalization developed with Isabelle [Pau94b] proves the correctness. Circular XML-specifications are integrated with an inductive method [Pau94a]. These XML-specifications are unsatisfiable. A deductive checker is presented based on this development. The checker generates a representation with F-Logic [KLW95] that is checked with Florid [HLS07]. The correctness of the checker is proven.

XML-specifications are introduced in the next section with a database of teachers. Section 3 presents a formalization of XML-specifications illustrated with the example. Then section 4 formalizes circular XML-specifications. Section 5 presents theorems for proving that circular XML-specifications are unsatisfiable in section 6. Then section 7 presents the deductive checker and section 8 concludes the contribution.

2 A Database of Teachers

A database of teachers is represented with an XML-specification. Elements (*teachers*, *research*, *subject*) and attributes (*name*, *instructor*) are defined with the structural schema in figure 1. Content models form a structure for XML-trees. The root labeled *teachers* stores content model $teacher^+$. XML-trees of the structural schema have a *teachers* root with *teacher* children. Figure 2 shows an instance, the next section presents details. Attribute *instructor* represents a teacher. Integrity represents dependencies of attributes. Keys and inclusion constraints formalize integrity. Key $teacher.name \rightarrow teacher$ represents *teacher* with *name*. Inclusion constraint $subject.instructor \subseteq teacher.name$ represents the dependency of *instructor* of *subject* on names.

$$teacher.name \rightarrow teacher$$
$$subject.instructor \rightarrow subject$$
$$subject.instructor \subseteq teacher.name$$

The XML-tree presented in figure 2 satisfies $teacher.name \rightarrow teacher$. The *teacher* nodes store *Dr. Brett* and *Prof. Crey*. The instructors are contained, $subject.instructor \subseteq teacher.name$ is satisfied. several subjects store *Dr. Brett*, the key $subject.instructor \rightarrow subject$ isn't satisfied. The example is unsatisfiable. The

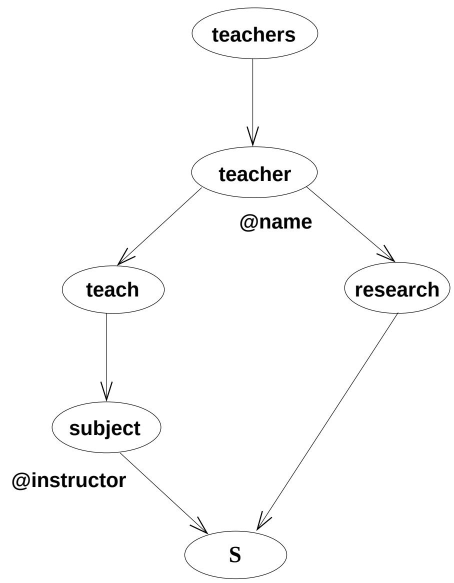

Fig. 1. A graph represents the structural schema of the example.

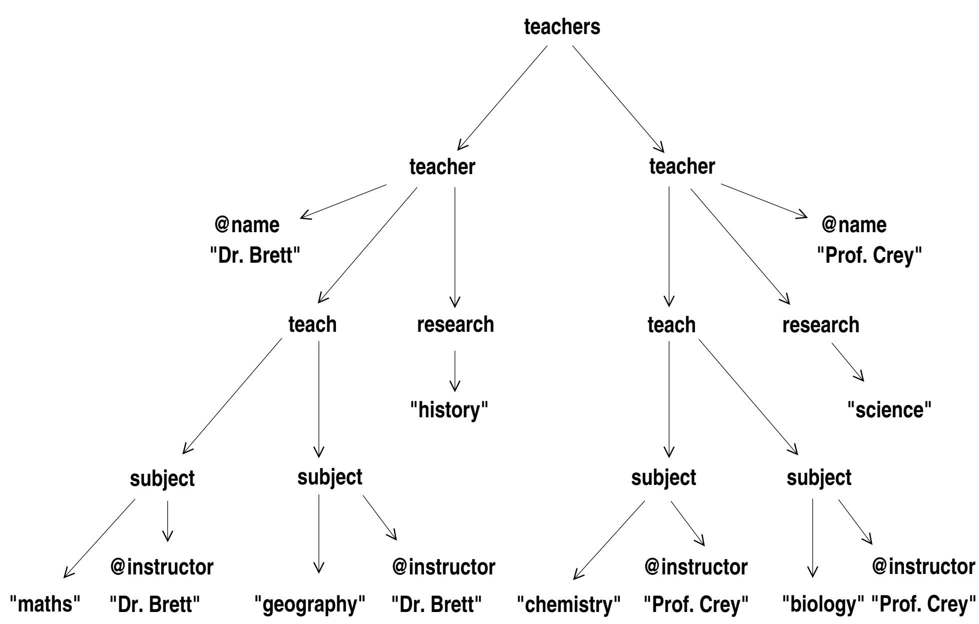

Fig. 2. An example XML-tree stores the teachers *Dr. Brett* and *Prof. Crey*.

structural schema contains a branch. The *teacher* nodes, formalized with $ext(teacher)$, have more *subject* descendants.

$$2|ext(teacher)| \leq |ext(subject)|$$

A document has at least a *teacher*.

$$|ext(teacher)| < |ext(subject)|$$

A contradiction is proven with $subject.instructor \rightarrow subject$ and $subject.instructor \subseteq teacher.name$. The next section presents the formalization of XML-specifications that forms the fundament for the checker presented in section 7.

3 Formalization of the Database

The section presents a formalization of XML-specifications illustrated with the database of teachers. The structural schema and integrity are formalized. Attributes A (*name*, *instructor*) and elements E (*teachers*, *subject*) with root r are defined with a structural schema [BMP+06]. The example in figure 3 has the root

```
<!DOCTYPE teachers [
<!ELEMENT teachers (teacher+)>
<!ELEMENT teacher (teach, research)>
<!ELEMENT teach (subject, subject)>
<!ELEMENT research (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ATTLIST teacher name CDATA #REQUIRED>
<!ATTLIST subject instructor CDATA #REQUIRED> ]>
```

Fig. 3. The structural schema of figure 1 is defined.

teachers. The attributes of an element are stored with function R, *teacher* stores attribute *name*. Function P stores the content models. Element *teachers* stores $teacher^+$. Regular expressions [HMU06] are formalized with an inductive method [BW]. Concatenation, choice, star, plus and question mark form content models with labels $\tau \in E$, text $\mathbf{S}$ and empty content ϵ.

$$\alpha \ ::= \ \epsilon \mid \mathbf{S} \mid \tau \mid (\alpha, \alpha) \mid (\alpha|\alpha) \mid \alpha^\star \mid \alpha^+ \mid \alpha?$$

Wellformed structural schemas are formalized. The sets A and E are disjoint and don't include $\mathbf{S}$. The functions P and R are defined for E. Labels in $(E \cup \{\mathbf{S}\}) \setminus \{r\}$ form content models that connect r with the elements. XML-trees are formalized with the nodes V. The example XML-tree in figure 4 includes the nodes v_1, v_2, ..., v_9. Function *lab* stores labels in $A \cup E \cup \{\mathbf{S}\}$, *teacher* is stored for v_2 and *name* for v_4. Children are stored with *ele*. Parents are unique. Nodes $[v_2, v_3]$ are the children of v_1 that represents *root*, the only node with the label r. Attribute nodes are stored with *att*. The example defines *name* for *teacher*. For v_2 and this attribute *att* stores v_4. Function *val* stores text of nodes with a label in $A \cup \{\mathbf{S}\}$, v_4 stores *Dr. Brett*. Text nodes don't have children. Label *name* proves v_4 doesn't have children.

$$ext(\tau) = \{\, v \mid v \in V \ \wedge \ lab\ v = \tau \ \wedge \ \tau \in E \cup \{\mathbf{S}\}\}$$

Nodes labeled $\tau \in E \cup \{\mathbf{S}\}$ are formalized with $ext(\tau)$. For example, $ext(teacher)$ stores $\{v_2, v_3\}$. Paths are formalized with an inductive method [Pau94a].

$$\frac{v_1 \in ext(\tau)}{path(v_1, v_1)} \qquad \frac{path(v_1, v_3) \qquad v_2 \in ele\ v_3}{path(v_1, v_2)}$$

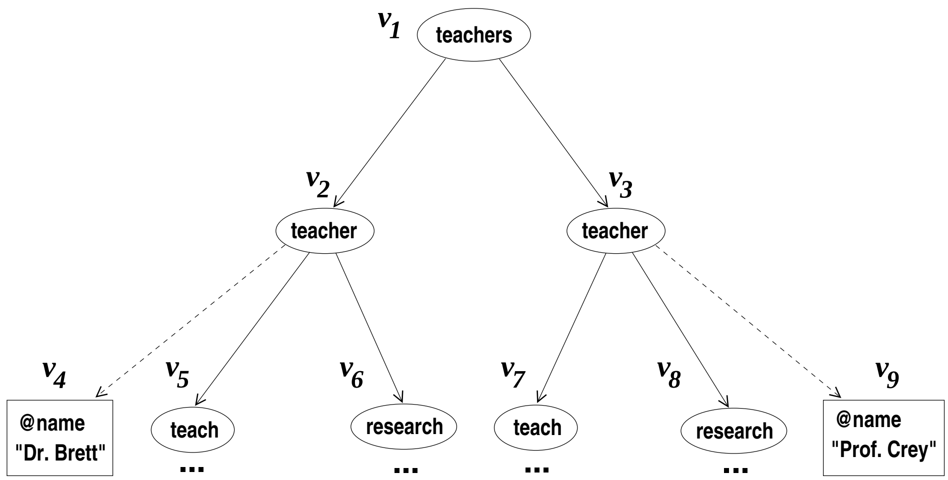

Fig. 4. A detailled view of the XML-tree in figure 2.

Paths are reflexive and $path(v_1, v_3)$ can be extended with children of v_3. The example satisfies $path(v_1, v_1)$, $path(v_1, v_2)$ and $path(v_1, v_5)$. XML-trees don't have cycles. Paths connect *root* with the element nodes. The section formalizes the validation of XML-trees. Children are labeled with the content models.

$$parse\ B\ (grammar(lab\ v)) \wedge getWord(B) = (\mathsf{map}\ lab\ (ele\ v))$$

An element node v has a parse tree B for the formal grammar [HMU06], formalized with $grammar(lab\ v)$. The labels of B computed with $getWord(B)$ are equal to the labels of the children. The nodes v_5, v_6, the children of v_2, have the following labels.

$$\mathsf{map}\ lab\ (ele\ v_2)\ =\ [teach, research]$$

The grammar for *teacher* accepts the labels. Details of the formalization are presented in [His07]. The section formalizes integrity. Attribute l of a τ node v is stored with $v.l = val(att(v, l))$. The example stores *Prof. Crey* with $v_9.name$. Attributes $L =< l_1, ..., l_n >$ are stored with $v[L] =< v.l_1, ..., v.l_n >$. The L tuples of τ nodes are formalized with $ext(\tau[L])$ and $ext(\tau.l)$ for one attribute. With $ext(teacher.name)$, the example stores $\{Dr.\ Brett, Prof.\ Crey\}$.

$$ext(\tau[L])\ =\ \{v[L]\,|\,v \in V\ \wedge\ lab\ v = \tau\}$$

The section formalizes integrity. Key $\tau[L] \rightarrow \tau$ identifies τ nodes with attributes L. The formalization considers a function $f(v) = v[L]$.

$$\tau[L] \rightarrow \tau\ \Leftrightarrow\ \mathsf{inj\_on}\ f\ ext(\tau)$$

For $\tau \in E$ with attributes L, the key is satisfied provided f is injective for the τ nodes. The example satisfies $teacher.name \rightarrow teacher$. Inclusion constraints represent dependencies of attributes.

$$\tau_1[L_1] \subseteq \tau_2[L_2]\ \Leftrightarrow\ ext(\tau_1[L_1])\ \subseteq\ ext(\tau_2[L_2])$$

An XML-tree satisfies inclusion constraint $\tau_1[L_1] \subseteq \tau_2[L_2]$ whenever L_1 tuples of τ_1 nodes depend on L_2 tuples of τ_2. The formalization has been implemented with Isabelle [NP07]. The formalization is the fundament for the contribution. The next section formalizes circular XML-specifications.

4 Circular XML-Specifications

The section presents the formalization of circular XML-specifications. They are formalized with an inductive method [Pau94a] that proves the correctness of cryptographic protocols in [Pau98]. The section formalizes ways. A branch has two ways to a descendant. XML-specifications are circular when there is a branch without cycle and the descendant of the branch depends on the ancestor. The next section proves that circular XML-specifications are unsatisfiable. The branch proves a constraint and the dependency proves the opposite. Section 7 presents a deductive checker for XML-specifications based on circular XML-specifications. The formalization considers normalized content models [His07].

$$\forall \tau \in E. (P\ \tau = \epsilon) \vee (\exists \tau_1, \tau_2 \in E \cup \{\mathbf{S}\}.\ P\ \tau = \tau_1 \vee P\ \tau = (\tau_1, \tau_2) \vee (P\ \tau = (\tau_1|\tau_2) \wedge \tau_1 \neq \tau_2))$$

They have less or equal two labels and don't contain plus, question mark and star. The section formalizes ways formed with content models.

$$\frac{\tau_1 \in E}{\mathit{way}(\tau_1, \tau_1)} \qquad \frac{P\ \tau_1 = \tau_3 \qquad \mathit{way}(\tau_3, \tau_2)}{\mathit{way}(\tau_1, \tau_2)}$$

Ways are reflexive and content τ_3 of τ_1 with $\mathit{way}(\tau_3, \tau_2)$ implies $\mathit{way}(\tau_1, \tau_2)$.

$$\frac{P\ \tau_1 = (\tau_3, \tau_4) \quad \mathit{way}(\tau_3, \tau_2)}{\mathit{way}(\tau_1, \tau_2)} \qquad \frac{P\ \tau_1 = (\tau_4, \tau_3) \quad \mathit{way}(\tau_3, \tau_2)}{\mathit{way}(\tau_1, \tau_2)}$$

Ways can be extended with concatenated content when τ_3 proves $\mathit{way}(\tau_3, \tau_2)$.

$$\frac{P\ \tau_1 = (\tau_3|\tau_4) \quad \mathit{way}(\tau_3, \tau_2) \quad \mathit{way}(\tau_4, \tau_2)}{\mathit{way}(\tau_1, \tau_2)}$$

Content models $(\tau_3|\tau_4)$ extend ways where the elements τ_3, τ_4 have a way. The τ_1 nodes have τ_2 descendants when there is a way from τ_1 to τ_2. Section 6 proves an extensive library of theorems.

$$\mathit{branch}(\tau_1, \tau_2) \Leftrightarrow \exists \tau_3, \tau_4, \tau_5 \in E.\ \mathit{way}(\tau_1, \tau_3) \wedge P\ \tau_3 = (\tau_4, \tau_5) \wedge \mathit{way}(\tau_4, \tau_2) \wedge \mathit{way}(\tau_5, \tau_2)$$

A structural schema has a branch from τ_1 to τ_2 when there is a way from τ_1 to an element τ_3 such that the labels of the concatenated content model have a way to the descendant. The element τ_3 represents the branch. An XML-tree of a structural schema with the contents $\mathit{branch}(\tau_1, \tau_2)$ includes τ_2 descendants for τ_1 nodes. The example has a branch, *teacher* and *teach* have a way to *subject*. Element *teach* represents the branch with $(\mathit{subject}, \mathit{subject})$. The example satisfies $\mathit{branch}(\mathit{teach}, \mathit{subject})$ and $\mathit{branch}(\mathit{teacher}, \mathit{subject})$.

The section formalizes cycles. Circular XML-specifications have a branch without cycle. Elements that have a possible way are formalized.

$$\frac{\tau_2 \in P\ \tau_1}{\mathit{possibleWay}(\tau_1, \tau_2)} \qquad \frac{\mathit{possibleWay}(\tau_1, \tau_3) \qquad \tau_2 \in P\ \tau_3}{\mathit{possibleWay}(\tau_1, \tau_2)}$$

Possible ways are proven with content models. A possible way is obtained with an element $\tau_3 \in P\ \tau_1$ and a possible way from τ_3 to τ_2. XML-trees of a structural schema with $\mathit{possibleWay}(\tau_1, \tau_2)$ possibly have a τ_2 descendant for a τ_1 node. The example has a possible way from *teacher* to *subject*. Possible ways formalize cycles.

$$\mathit{cycle}(\tau) \Leftrightarrow \mathit{possibleWay}(\tau, \tau)$$

Structural schemas that satisfy $\mathit{possibleWay}(\tau, \tau)$ have a cycle with τ. The example doesn't have a cycle. The section formalizes integrity that bounds elements.

$$\mathit{anchor}(\tau_1, \tau_2) \Leftrightarrow \exists L_1 \subseteq (R\ \tau_1).\ \exists L_2 \subseteq (R\ \tau_2).\ \tau_1[L_1] \rightarrow \tau_1 \wedge \tau_1[L_1] \subseteq \tau_2[L_2]$$

A key $\tau_1[L_1] \rightarrow \tau_1$ and an inclusion constraint form an anchor when $\tau_1[L_1]$ depends on $\tau_2[L_2]$. The example has an anchor from *subject* to *teacher*.

$$\mathit{onceOccurs}(\tau_1, \tau_2) \Leftrightarrow P\ \tau_2 = \tau_1 \vee \exists \tau_3.\ \tau_3 \neq \tau_1 \wedge (P\ \tau_2 = (\tau_1, \tau_3) \vee P\ \tau_2 = (\tau_3, \tau_1))$$

Single and concatenated content models that contain a particular label once are formalized with *onceOccurs*. The example satisfies $onceOccurs(teacher, teach)$.

$$moreOccurs(\tau_1) \Leftrightarrow \exists \tau_2, \tau_4 \in E.\ \tau_1 \in P\ \tau_2\ \wedge\ \tau_1 \in P\ \tau_4\ \wedge\ \tau_2 \neq \tau_4$$

A structural schema satisfies $moreOccurs(\tau_1)$ when τ_1 occurs in the content models of some elements. The example satisfies $moreOccurs(\mathbf{S})$ because *research* and *subject* store text. The section presents the formalization of bounds.

$$\frac{anchor(\tau_1, \tau_2)}{bounds(\tau_1, \tau_2)} \qquad \frac{onceOccurs(\tau_1, \tau_2) \quad \neg moreOccurs(\tau_1)}{bounds(\tau_1, \tau_2)}$$

Element τ_2 bounds τ_1 when integrity forms an anchor. The example bounds *subject* with *teacher*. Element τ_2 that stores τ_1 once bounds τ_1.

$$\frac{way(\tau_1, \tau_2) \qquad \neg cycle(\tau_1)}{bounds(\tau_1, \tau_2)} \qquad \frac{bounds(\tau_1, \tau_3) \quad bounds(\tau_3, \tau_2)}{bounds(\tau_1, \tau_2)}$$

A way without cycle satisfies $bounds(\tau_1, \tau_2)$. The relation is transitive.

$$circular \Leftrightarrow way(r, \tau_1)\ \wedge\ branch(\tau_1, \tau_2)\ \wedge\ \neg cycle(\tau_1)\ \wedge\ bounds(\tau_2, \tau_1)$$

An XML-specification is circular, when the structural schema has a way from r to a branch that doesn't have a cycle at the ancestor of the branch and the descendant bounds the ancestor. The example is circular. There is a way from r to *teacher* and a branch from *teacher* to *subject*. The ancestor *teacher* doesn't have a cycle. The descendant *subject* is bounded with *teacher*. The section has presented the formalization of circular XML-specifications. The next section proves theorems for proving the correctness of the checker presented in section 7.

5 Paths in XML-Trees

The previous section has formalized circular XML-specifications based on the formalization in section 3. This section formalizes descendants and proves theorems of paths and ways. The next section proves that circular XML-specifications are unsatisfiable. The theorems prove the correctness of the checker presented in section 7.

Descendants are formalized with an inductive method [Pau94a] that formalizes circular XML-specifications in section 4. Then the section presents theorems for proving that the descendants of a branch of an XML-tree are disjoint in the next section. The τ_2 descendants of τ_1 nodes are formalized with $descendant(\tau_1, \tau_2)$.

$$\frac{v_1 \in ext(\tau_1)}{v_1 \in descendant(\tau_1, \tau_1)} \qquad \frac{v_1 \in ext(\tau_2) \quad v_1 \in ele\ v_2 \quad v_2 \in descendant(\tau_1, \tau_3)}{v_1 \in descendant(\tau_1, \tau_2)}$$

The descendants are reflexive for τ_1 nodes. Node v is a descendant of a τ_1 node when the parent $v_3 \in ext(\tau_3)$ is a descendant. The *subject* nodes of the example in figure 2 are descendants of teachers. They are children of a *teach* node that is a child of a *teacher* node of $descendant(teacher, teacher)$. Next, paths without label τ are formalized with *differentLabel*.

$$\frac{v_1 \in ext(\tau_1) \qquad \tau_1 \neq \tau}{differentLabel(v_1, v_1, \tau)} \qquad \frac{lab\ v_2 \neq \tau \quad v_2 \in ele\ v_3 \quad differentLabel(v_1, v_3, \tau)}{differentLabel(v_1, v_2, \tau)}$$

Nodes v_1 with a label τ_1 not equal τ satisfy $differentLabel(v_1, v_1, \tau)$. Such paths are extended with nodes having a label unequal τ. The section formalizes next nodes of a specified label.

$$same(v_1, v_2, \tau) \Leftrightarrow v_1 = v_2\ \wedge\ v_1 \in ext(\tau)$$

The τ nodes satisfy *same*.

$$nextDifferent(v_1, v_2, \tau) \Leftrightarrow \exists v_3.\ v_2 \in ext(\tau) \land v_2 \in ele\ v_3 \land differentLabel(v_1, v_3, \tau)$$

The section formalizes paths to τ nodes that don't contain τ.

$$next(v_1, v_2, \tau) \Leftrightarrow same(v_1, v_2, \tau) \lor nextDifferent(v_1, v_2, \tau)$$

Nodes that are next have a path. The section formalizes descendants of the branch.

$$v \in descendant1(\tau_1, \tau_2, \tau_3) \Leftrightarrow \exists v_1, v_2, v_3, v_4.\ v_1 \in ext(\tau_1) \land next(v_1, v_2, \tau_2) \land ele\ v_2 = [v_3, v_4] \land next(v_3, v, \tau_3)$$

A node $v \in descendant1(\tau_1, \tau_2, \tau_3)$ takes the first way at the next τ_2 descendant of a τ_1 node. Node v is the next τ_3 descendant of the first child of the τ_2 descendant. Maths and chemistry are stored with $descendant1(teacher, teach, subject)$.

$$v \in descendant2(\tau_1, \tau_2, \tau_3) \Leftrightarrow \exists v_1, v_2, v_3, v_4.\ v_1 \in ext(\tau_1) \land next(v_1, v_2, \tau_2) \land ele\ v_2 = [v_3, v_4] \land next(v_4, v, \tau_3)$$

Nodes in $descendant2(\tau_1, \tau_2, \tau_3)$ take the second way to τ_3. The section presents theorems of paths and ways. Then it is proven that circular XML-specifications are unsatisfiable.

$$\frac{v_2 \in ele\ v_1 \qquad v_3 \in ele\ v_1 \qquad v_2 \neq v_3}{\neg path(v_2, v_3)}\ T_1$$

Theorem T_1 proves that children aren't connected. The proof assumes the opposite, an induction with $path(v_2, v_3)$ proves the following.

$$P_1(v_2, v_3)\ \Leftrightarrow\ v_2 \in ele\ v_1 \land v_3 \in ele\ v_1 \land v_2 \neq v_3 \rightarrow \mathsf{false}$$

The base case proves a contradiction with $v_2 \neq v_3$. The induction step considers a node v_4 with the child v_3 and $path(v_2, v_4)$, $P_1(v_2, v_3)$ is proven. Parents are unique, v_1 is equal to v_4. XML-trees don't have cycles. A contradiction is proven with $v_2 \in ele\ v_1$ and $path(v_2, v_1)$.

$$\frac{path(v_1, v_3) \qquad path(v_2, v_3)}{path(v_1, v_2) \lor path(v_2, v_1)}\ T_2$$

Nodes with a path to the same node are connected (T_2). The contrapositive is proven. An induction with $path(v_1, v_3)$ proves that v_2 doesn't have a path to v_3.

$$P_2(v_1, v_3)\ \Leftrightarrow\ \neg path(v_1, v_2) \land \neg path(v_2, v_1) \rightarrow \neg path(v_2, v_3)$$

The hypothesis with one node is a tautology. The induction step considers a node v_4 with $path(v_1, v_4)$ that satisfies $P_2(v_1, v_4)$. Node v_4 has the child v_3 and $P_2(v_1, v_3)$ is proven. When v_2 is equal to v_3, it is a child of v_4 and the path from v_1 to v_4 gives a contradiction with $\neg path(v_1, v_2)$. Otherwise, $path(v_2, v_3)$ proves a path from v_2 to v_4 with child v_3. Then $P_2(v_1, v_4)$ proves a contradiction.

$$\frac{path(v_1, v_2) \qquad lab\ v_1 = \tau_1 \qquad lab\ v_2 = \tau_2 \qquad v_1 \neq v_2}{possibleWay(\tau_1, \tau_2)}\ T_3$$

Paths prove a possible way (T_3). An induction with $path(v_1, v_2)$ proves this.

$$P_3(v_1, v_2)\ \Leftrightarrow\ \forall \tau_1, \tau_2.\ lab\ v_1 = \tau_1 \land lab\ v_2 = \tau_2 \land v_1 \neq v_2 \rightarrow possibleWay(\tau_1, \tau_2)$$

The base case proves a contradiction. The induction step satisfies $P_3(v_1, v_3)$ and considers a child v_2 of v_3. The section proves that τ_1 has a possible way to the label of v_2. When v_1 equals v_3 the content model of τ_1 includes τ_2, a possible way is proven. Otherwise, the induction hypothesis proves $\mathit{possibleWay}(\tau_1, (\mathit{lab}\ v_3))$. The content model of the label of v_3 includes τ_2.

$$\frac{\mathit{next}(v_1, v_2, \tau) \qquad \mathit{next}(v_1, v_3, \tau) \qquad \mathit{path}(v_2, v_3)}{v_2 = v_3}\ T_4$$

Theorem T_4 proves that nodes are equal when they are connected next descendants of the same node. Nodes that are equal can satisfy *same*. Otherwise, v_2 and v_3 have a path without τ from v_1. Then v_1 has a *differentLabel* path to the parent of v_3. The path contains the τ node v_2.

$$\frac{v_1 \in \mathit{ext}(\tau_1) \qquad \mathit{way}(\tau_1, \tau_2)}{\exists v_2 \in \mathit{ext}(\tau_2).\ \mathit{path}(v_1, v_2)}\ T_5$$

With T_5, a path to a τ_2 node is proven for a τ_1 node when there is a way from τ_1 to τ_2. An induction with $\mathit{way}(\tau_1, \tau_2)$ uses hypothesis $P_4(\tau_1, \tau_2)$.

$$P_4(\tau_1, \tau_2) \ \Leftrightarrow\ \forall v_1 \in \mathit{ext}(\tau_1).\ \exists v_2 \in \mathit{ext}(\tau_2).\ \mathit{path}(v_1, v_2)$$

The base case is proven with one node. The induction step considers the content models of τ_1. Concatenated content models prove a τ_3 child that extends a path to v_2 proven with hypothesis $P_4(\tau_3, \tau_2)$. Otherwise $P\ \tau_1 = (\tau_3|\tau_4)$, a child in $\mathit{ext}(\tau_3) \cup \mathit{ext}(\tau_4)$ is proven. Then $P_4(\tau_3, \tau_2)$ and $P_4(\tau_4, \tau_2)$ prove a path. Theorems have been presented for proving that circular XML-specifications are unsatisfiable in the next section. Section 7 presents a deductive checker.

6 Unsatisfiable Circular XML-Specifications

The section proves that circular XML-specifications are unsatisfiable with the theorems of the previous section. A branch proves more nodes of the descendant. The branch is bounded, a contradiction is proven.

$$\frac{\mathit{branch}(\tau_1, \tau_2) \qquad \mathit{ext}(\tau_1) \neq \emptyset \qquad \neg \mathit{cycle}(\tau_1)}{|\mathit{ext}(\tau_1)| \ < \ |\mathit{ext}(\tau_2)|}$$

An XML-tree with τ_1 nodes contains more descendants of a branch when the structural schema doesn't have a cycle with τ_1. The section proves the first and second descendants of the XML-tree are disjoint. Then a cycle with τ_1 is proven.

The proof considers τ_3 that represents the branch. It is proven that the intersection of $\mathit{descendant1}(\tau_1, \tau_3, \tau_2)$ and $\mathit{descendant2}(\tau_1, \tau_3, \tau_2)$ is empty. A node v of the intersection is presumed. Function f_1 (f_2) chooses the ancestor of the first (second) descendant of a branch of the XML-tree. The function $f_1(v) = v_1$ is defined with nodes v_2, v_3 and v_4 that satisfy $\mathit{next}(v_1, v_2, \tau_3)$, $\mathit{ele}\ v_2 = [v_3, v_4]$ and $\mathit{next}(v_3, v, \tau_2)$. In this way, $f_2(v) = v_5$ is defined with nodes v_6, v_7 and v_8 that satisfy $\mathit{next}(v_5, v_6, \tau_3)$, $\mathit{ele}\ v_6 = [v_7, v_8]$ and $\mathit{next}(v_8, v, \tau_2)$. When $v_1 \neq v_5$, T_3 proves $\mathit{possibleWay}(\tau_1, \tau_1)$ with the path of v_1 and v_5 proven with T_2. This is a contradiction with $\mathit{cycle}(\tau_1)$. Thus, v_1 and v_5 are equal. The proof considers node v_2 (v_6) that represents the first (second) branch. They have a path to v. Theorem T_2 proves a path connects them. The nodes are next τ_3 nodes of v_1, T_4 proves they are equal. Moreover, the children are equal. They aren't connected (T_1) and have the descendant v, T_2 proves a contradiction. The first and second descendants are disjoint.

An XML-tree has more τ_2 descendants than descendants of the first branch.

$$|\mathit{descendant}(\tau_1, \tau_2)| \geq |\mathit{descendant1}(\tau_1, \tau_3, \tau_2) \cup \mathit{descendant2}(\tau_1, \tau_3, \tau_2)|$$

The τ_2 descendants contain the descendants of a branch.

$$|\mathit{descendant1}(\tau_1, \tau_3, \tau_2)| + |\mathit{descendant2}(\tau_1, \tau_3, \tau_2)| > |\mathit{descendant1}(\tau_1, \tau_3, \tau_2)|$$

They are disjoint, the sum is considered. The XML-tree has τ_1 nodes, T_5 proves descendants of the branch.

The proof presumes less or equal τ_2 descendants than τ_1 nodes. The previous inequation proves more τ_1 nodes than τ_2 descendants of the first branch. A function f_3 chooses a first descendant of a branch. The function is defined with $f_3(v_1) = v_2$ and nodes v_3, v_4 and v_5 such that $next(v_1, v_3, \tau_3)$, $ele\ v_3 = [v_4, v_5]$ and $next(v_4, v_2, \tau_2)$ are satisfied. The range equals the first descendants, T_5 proves f_3 is defined for τ_1 nodes. The domain is greater, there are nodes $v_1, v_2 \in ext(\tau_1)$ with the descendant $v_3 = f_3(v_1) = f_3(v_2)$. The nodes have a path to v_3, T_2 proves v_1 and v_2 are connected. Then T_3 proves $possibleWay(\tau_1, \tau_1)$. This is a contradiction with $\neg cycle(\tau_1)$. The XML-tree satisfies $|ext(\tau_1)| < |descendant(\tau_1, \tau_2)|$. The descendants are contained in $ext(\tau_2)$, the theorem is proven. Theorem T_5 proves τ_1 nodes with $way(r, \tau_1)$. Circular XML-specifications satisfy $|ext(\tau_1)| < |ext(\tau_2)|$. The next theorem proves the opposite with $bounds(\tau_2, \tau_1)$. Circular XML-specifications are unsatisfiable.

$$\frac{bounds(\tau_1, \tau_2)}{|ext(\tau_1)| \leq |ext(\tau_2)|}$$

An induction with $bounds(\tau_1, \tau_2)$ proves less or equal τ_1 than τ_2 nodes. Anchors prove the inequation with integrity. A constraint of the transformation for model checking XML-specifications presented in [His07] proves an equation for elements that occur once. An injective function chooses a descendant for ways without cycle. Finally, the induction hypothesis proves the theorem. The proof defines the induction hypothesis $|ext(\tau_1)| \leq |ext(\tau_2)|$. Integrity implies inequations proven in [His07]. An anchor is defined with a key $\tau_1[L_1] \rightarrow \tau_1$ and an inclusion constraint $\tau_1[L_1] \subseteq \tau_2[L_2]$.

$$|ext(\tau_1)| = |ext(\tau_1[L_1])| \leq |ext(\tau_2[L_2])| \leq |ext(\tau_2)|$$

The key proves the number of τ_1 nodes and L_1 tuples is equal. The inclusion constraint proves that they are less or equal than the L_2 tuples of τ_2. Then the proof considers elements that occur once. The transformation proves a structured representation of nodes.

$$|ext(\tau_1)| = \sum_{\substack{\tau_1 \in P\ \tau_2 \\ i \in \{1,2\}}} |children(\tau_2, \tau_1, i)|$$

The constraint represents the τ_1 nodes with the first and second children.

$$|ext(\tau_1)| = |children(\tau_2, \tau_1, i)| \leq |ext(\tau_2)|$$

Then $onceOccurs(\tau_1, \tau_2)$ and $\neg moreOccurs(\tau_1)$ prove that τ_1 has the parent τ_2. An XML-tree has less or equal children than τ_2 nodes. The proof considers ways without cycle.

$$\frac{way(\tau_1, \tau_2) \qquad \neg cycle(\tau_1)}{|ext(\tau_1)| \leq |ext(\tau_2)|}$$

An injective function proves the theorem choosing the next τ_2 node of a τ_1 node. The way proves with T_5 that a path exists. The function f_4 is proven injective. Otherwise there are nodes v_1 and v_2 with $v_3 = f_4(v_1) = f_4(v_2)$. With the paths to v_3 T_2 proves v_1 and v_2 are connected. Moreover, with T_3 a possible way from τ_1 to itself is proven. The contradiction with $\neg cycle(\tau_1)$ proves the inequation.

Finally, the induction proves $|ext(\tau_1)| \leq |ext(\tau_2)|$ with labels τ_1, τ_2 and τ_3 that satisfy $bounds(\tau_1, \tau_3)$ and $bounds(\tau_3, \tau_2)$. The induction hypothesis proves $|ext(\tau_1)| \leq |ext(\tau_3)|$ and $|ext(\tau_3)| \leq |ext(\tau_2)|$. The next section presents a checker based on circular XML-specifications.

7 Deductive Checker

The previous section has proven that circular XML-specifications are unsatisfiable. This section presents a checker based on circular XML-specifications. The checker generates a representation with F-Logic [KLW95].

Objects represent elements and attributes, a class hierarchy represents the structural schema. The section presents a deductive checker based on circular XML-specifications. Section 6 proves the correctness of the checker that has been implemented with the DEAXS [His07] project. The checker generates F-Logic facts that are checked with Florid [HLS07].

The section presents the formalization of XML-specifications with F-Logic. Objects of class *Element* represent elements, the subclasses represent the normalized structural schemas [His07]. The checker is presented with the example defined in figure 3 and root *teacher*. The section formalizes the structural schema. Class

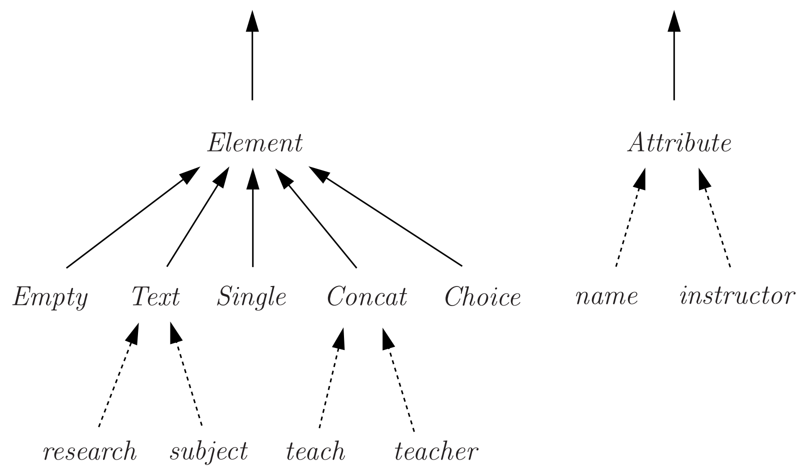

Fig. 5. A class hierarchy represents the example in figure 3.

Element has the subclasses *Empty*, *Single*, *Text*, *Concat* and *Choice* that represent the content models. A signature defines the class hierarchy and provides the method declaration.

$$\begin{array}{l} Element[attributes \Rrightarrow Attribute]. \\ Empty :: Element. \\ Single :: Element[contents \Rightarrow Element]. \\ Text :: Element[contents \Rightarrow Empty]. \\ Concat :: Element[contents@(integer) \Rightarrow Element]. \\ Choice :: Element[contents@(integer) \Rightarrow Element]. \end{array}$$

For example, *teacher* stores content model $(teach, research)$. Object *teacher* is an instance of *Concat*. Attributes are stored with method *attributes*. Element *teacher* stores *name*, an instance of *Attribute*. The hierarchy is shown in Figure 5. Fact $\tau : Empty$ represents ϵ. Contents $P\ \tau = \tau_1$ is represented with $\tau : Single[contents \rightarrow \tau_1]$, fact $\tau : Concat[contents@(1) \rightarrow \tau_1; contents@(2) \rightarrow \tau_2]$ represents (τ_1, τ_2).
The structural schema of the example is represented.

$$\begin{array}{l} name : Attribute. \\ instructor : Attribute. \\ teacher : Concat[contents@(1) \rightarrow teach; contents@(2) \rightarrow research; attributes \twoheadrightarrow name]. \\ teach : Concat[contents@(1) \rightarrow subject; contents@(2) \rightarrow subject; attributes \twoheadrightarrow \{\}]. \\ research : Text[contents \rightarrow S; attributes \twoheadrightarrow \{\}]. \\ subject : Text[contents \rightarrow S; attributes \twoheadrightarrow instructor]. \\ \mathbf{S}[attributes \twoheadrightarrow \{\}]. \end{array}$$

Rules define relation *way* for formalizing circular XML-specifications in section 4.

$$
\begin{aligned}
way(X_1, X_1) &\longleftarrow X_1 : Element.\\
way(X_1, X_2) &\longleftarrow X_1 : Single[contents{\rightarrow}X_3] \wedge way(X_3, X_2).\\
way(X_1, X_2) &\longleftarrow X_1 : Concat[contents@(\_){\rightarrow}X_3] \wedge way(X_3, X_2).\\
way(X_1, X_2) &\longleftarrow X_1 : Choice[contents@(1){\rightarrow}X_3; contents@(2){\rightarrow}X_4] \wedge way(X_3, X_2) \wedge way(X_4, X_2).
\end{aligned}
$$

The example satisfies $way(subject, subject)$, $way(teach, subject)$ and the way from *teacher* to *subject*. Next, a rule proves a branch.

$$
\begin{aligned}
branch(X_1, X_2) \longleftarrow\ & way(X_1, X_3) \wedge X_3 : Concat[contents@(1){\rightarrow}X_4; contents@(2){\rightarrow}X_5] \wedge\\
& way(X_4, X_2) \wedge way(X_5, X_2).
\end{aligned}
$$

Relation $branch(X_1, X_2)$ is defined with a way from X_1 to X_3 that represents the branch with ways to the descendant. Elements *teacher* and *teach* have a branch to *subject*. The example satisfies $branch(teach, subject)$ and a branch from *teacher* to *subject*. The section formalizes cycles.

$$
\begin{aligned}
X_1[occurs{\twoheadrightarrow}X_2] &\longleftarrow X_2 : Single[contents{\rightarrow}X_1].\\
X_1[occurs{\twoheadrightarrow}X_2] &\longleftarrow X_2 : Element[contents@(\_){\rightarrow}X_1].
\end{aligned}
$$

Attribute *occurs* ($Element[occurs{\Rrightarrow}Element]$) is defined with content models. Element *subject* satisfies $subject[occurs{\twoheadrightarrow}teach]$. Possible ways are formalized.

$$
\begin{aligned}
possibleWay(X_1, X_2) &\longleftarrow X_2[occurs{\twoheadrightarrow}X_1].\\
possibleWay(X_1, X_2) &\longleftarrow X_2[occurs{\twoheadrightarrow}X_3] \wedge possibleWay(X_1, X_3).
\end{aligned}
$$

The example satisfies $possibleWay(teacher, X)$ for $X \in \{research, subject, teach\}$.

$$cycle(X_1) \longleftarrow possibleWay(X_1, X_1).$$

A cycle with X_1 is proven when X_1 has a possible way to itself. The example doesn't have cycles. The section formalizes elements that occur with more content models.

$$moreOccurs(X_1) \longleftarrow X_1[occurs{\twoheadrightarrow}X_2] \wedge X_1[occurs{\twoheadrightarrow}X_3] \wedge X_2 \neq X_3.$$

A label X_1 that occurs in two content models satisfies $moreOccurs(X_1)$.

$$
\begin{aligned}
onceOccurs(X_1, X_2) &\longleftarrow X_2 : Single[contents{\rightarrow}X_1].\\
onceOccurs(X_1, X_2) &\longleftarrow X_2 : Concat[contents@(1){\rightarrow}X_1; contents@(2){\rightarrow}X_3] \wedge X_1 \neq X_3.\\
onceOccurs(X_1, X_2) &\longleftarrow X_2 : Concat[contents@(1){\rightarrow}X_3; contents@(2){\rightarrow}X_1] \wedge X_1 \neq X_3.
\end{aligned}
$$

Single content models τ_1 and concatenations with τ_3 ($\tau_3 \neq \tau_1$) that are stored for τ_2 prove $onceOccurs(\tau_1, \tau_2)$. The example satisfies $onceOccurs(research, teacher)$. Next, *bounds* are defined.

$$
\begin{aligned}
bounds(X_1, X_2) &\longleftarrow anchor(X_1, X_2).\\
bounds(X_1, X_2) &\longleftarrow bounds(X_1, X_3) \wedge bounds(X_3, X_2).\\
bounds(X_1, X_2) &\longleftarrow onceOccurs(X_1, X_2) \wedge \neg moreOccurs(X_1).\\
bounds(X_1, X_2) &\longleftarrow way(X_1, X_2) \wedge \neg cycle(X_1).
\end{aligned}
$$

An anchor bounds elements with integrity. XML-specifications with $\tau_1[L_1] \rightarrow \tau_1$ and $\tau_1[L_1] \subseteq \tau_2[L_2]$ satisfy $anchor(\tau_1, \tau_2)$. The example has an anchor. With the integrity constraints $subject.instructor \rightarrow subject$ and $subject.instructor \subseteq teacher.name$ the example proves $anchor(subject, teacher)$. Element X_2 that bounds X_3 that bounds X_1 satisfies $bounds(X_1, X_2)$. Elements that occur once satisfy *bounds*. Ways from X_1 to X_2 without cycle satisfy $bounds(X_1, X_2)$. The example bounds *subject* with *teacher*. Florid [HLS07] proves that the example is circular.

$$
\begin{aligned}
&way(teacher, teacher).\\
&branch(teacher, subject).\\
&\neg cycle(teacher).\\
&bounds(subject, teacher).
\end{aligned}
$$

The rules prove a branch. The root element *teacher* has a way to *teach* with a branch to *subject*. The example doesn't have a cycle with *teacher* that is bounded with *subject*.

$$?\!- \; way(r, X_1) \wedge branch(X_1, X_2) \wedge \neg cycle(X_1) \wedge bounds(X_2, X_1).$$

The example is proven circular with *teacher* for X_1, *subject* for X_2 and the root *teacher*. The section has presented a checker for XML-specifications. Section 6 has proven the correctness of the checker. The checker has been implemented with the DEAXS [His07] project.

8 Conclusion

The previous section has presented a deductive checker. The contribution concludes with an overview.

An extensive formalization is developed with Isabelle [Pau94b]. Details are presented in [His07]. Circular XML-specifications are formalized with an inductive method [Pau94a]. Section 6 proves that circular XML-specifications are unsatisfiable. Section 7 presents a checker based on circular XML-specifications. XML-specifications are represented with F-Logic [KLW95]. The correctness of the checker is proven. The checker is implemented with the DEAXS [His07] project. The checker normalizes structural schemas , generates graphs and the representation of XML-specifications with F-Logic [KLW95] that is checked with Florid [HLS07].

References

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[BMP+06] Tim Bray, Eve Maler, Jean Paoli, Michael C. Sperberg-McQueen, and François Yergeau. XML. http://www.w3.org/TR/REC-xml/, 2006.

[BW] Stefan Berghofer and Markus Wenzel. Inductive Datatypes in HOL—Lessons Learned in Formal- Logic Engineering. In *TPHOLs 99*, pages 19–36.

[CV85] Ashok K. Chandra and Moshe Y. Vardi. The Implication Problem for Functional and Inclusion Dependencies is Undecidable. *SIAM Journal on Computing*, 14(3):671–677, 1985.

[FL02] Wenfei Fan and Leonid Libkin. On XML Integrity Constraints in the Presence of DTDs. *Journal of the ACM*, 49(3):368–406, 2002.

[His07] Harald Hiss. Werkzeuge zur Entwicklung von XML-Spezifikationen. Technical report, Institut für Informatik, Universität Freiburg, 2007.

[HLS07] Thomas Hornung, Georg Lausen, and Florian Schmedding. Florid. http://dbis.informatik.uni-freiburg.de/index.php? project=Florid, 2007.

[HMU06] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley, 2006.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.

[NP07] Tobias Nipkow and Lawrence C. Paulson. Isabelle. http://www.cl.cam.ac.uk/Research/HVG/Isabelle/, 2007.

[Pau94a] Lawrence C. Paulson. A Fixedpoint Approach to Implementing (Co)Inductive Definitions. In *CADE-12*, pages 148–161, London, UK, 1994. Springer-Verlag.

[Pau94b] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS 828. Springer, 1994.

[Pau98] Lawrence C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.