基于XML路径和类型的决策问题的研究

沈 洁 印桂生 王向辉
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

摘 要 提出了一种算法来分析用正则树表示的XML数据中对XPath的决策问题,并用该方法检查XPath静态类型的数据。此外根据对有限顺序树的带逆操作的逻辑的判定性定理,证明了算法时间复杂度为简单的一个表达式的指数级大小。提出了一套实用的、有效的数学模型来解决XPath表达式中的可满足性问题。并通过对一些决策问题,例如带或者不带类型约束的XPath的空、包含、重叠和覆盖问题的实验对算法进行了证实,实验证明该系统能够有效用于对操作XPath表达式和XML类型注释的程序语言的静态分析器中。

关键词 XML,决策问题,可满足性检测
中图法分类号 TP393 **文献标识码** A

Research on the Decision Problem of XML Path and Type

SHEN Jie YIN Gui-sheng WANG Xiang-hui
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract In this paper, a new algorithm was proposed to analyze the decision problem to XPath in XML data which is represented as regular tree. By this way we could also check the static data's type of XPath. Based on the decidability of a logic with converse for finite ordered tree, we proved that its' time complexity is a simple exponential of the size of a formula. Then a practical and effective mathematical model was built to slove the satisfiability problem of a formula. Through some examples of decision problems, such as XPath emptiness, containment, overlap, and coverage, with or without type constraints, the algorithm was confirmed by these experiments that our system can be effectively used in static analyzers for both XPath expressions and XML type annotations.

Keywords XML, Decision problem, Satisfiability-testing

随着XML数据应用的日益广泛,对基于XML的程序语言中存在的XML数据类型和XPath查询进行有效的检查已经成为一个重要的研究热点。在这些问题当中,对XML数据的类型,DTDs或者XML Schema中的关于XPath决策问题的研究日益增多,例如包括空集检测(一个表达式曾经是否选择了节点)、包容性(一个表达式的结果是否总是包含于另一个表达式的结果中)、重叠(两个表达式是否选择了相同的节点)、覆盖(一个表达式所选出来的节点是否总是包含于若干个其他表达式所选择的节点的集合中)。本文主要讨论XPath决策问题中的两个议题,首先我们需要识别出可以用来描述正则树类型和XPath的逻辑表达式;其次,我们需要有效地解决可满足性问题,检测一个给定的逻辑表达式能否描述符合可满足性检测算法的有限树。

本文提出了一种可满足性检测算法来分析用正则树表示的XML数据中XPath的某些决策问题,并用该方法检查XPath静态类型的数据。然后,提供了对有限顺序树的带逆操作的逻辑表达式的判定性定理,证明了该算法的时间复杂度是表达式的指数级大小,这里涉及到的表达式是无环的(cycle-free)有限树,满足不带最多驻点的可供选择的自由模态 μ-calculus 的逻辑限制。

本文第1节介绍了算法所涉及到聚焦树、逻辑表达式以及嵌入式的正则树的定义;第2节定义了XPath和逻辑表达式之间的转化规则;第3节提出了一种可满足性检测算法用来解决一些XPath的决策问题并在理论上证明了它的正确性、完备性以及算法的复杂度;最后通过在实际的应用中的一些决策问题、类型检查方面的应用以及实验结果证明了算法的可行性。

1 相关技术

1.1 焦点树(Focused Tree)

为了在数据库导航中更好地表示XML树,我们引入焦点树[1]的概念。焦点树不仅描述了树还包括它的上下文关系:它的前兄弟节点和它的父节点。这样的一种结构可以有效地保存所有的信息,这对于某些特定的语言,例如XPath导航中的向前和向后轴的查询是非常有用的。首先假定一个标签的字母表 Σ,定义焦点树如下。

到稿日期:2009-12-31 返修日期:2010-03-11 本文受863基金(No. 2007AA012401),哈尔滨市基金(RC2009XK010003)资助。
沈 洁(1981—),女,博士生,主要研究方向为数据库与知识库、XML数据库;印桂生(1964—),男,教授,博士生导师,主要研究方向为数据库与知识库、虚拟现实,E-mail: yinguisheng@hrbeu.edu.cn。

$$\begin{cases} t::=\sigma[tl] & \text{tree;} \\ tl::= & \text{list of tree;} \\ \quad \begin{cases} \epsilon & \text{empty list;} \\ t::tl & \text{cons cell;} \end{cases} & \\ c::= & \text{context;} \\ \quad \begin{cases} (tl,Top,tl) & \text{root of the tree;} \\ (tl,c[\sigma],tl) & \text{context node;} \end{cases} & \\ f::=(t,c) & \text{focused tree;} \end{cases}$$

1.2 逻辑表达式

下面介绍一下 XPath 表达式和 XML 正则树类型的转换逻辑表达式，一个带逆操作的可供选择的自由模态 μ-calculus 的子逻辑，还要给出 XML 表达式的一些定义和规范表示，以及给出关于这些表达式的解释，例如有限焦点树的集合。最后给出该模型中仅有一个简单驻点的逻辑以及它们在导航中的封闭性。

$L_\mu \ni \varphi,\psi::=$formula

$$\begin{cases} T & \text{ture;} \\ X & \text{variable;} \\ \langle a\rangle\varphi \mid \neg\langle a\rangle T & \text{existential;} \\ \sigma \mid \neg\sigma & \text{atomic prop(negated);} \\ \varphi\vee\psi & \text{disjunction;} \\ \text{Ⓡ} \mid \neg\text{Ⓡ} & \text{start prop(negated);} \\ \varphi\wedge\psi & \text{conjunction;} \\ \mu\overline{X_i\cdot\varphi_i} \text{ in } \psi & \text{least n-ary fixpoint;} \\ \nu\overline{X_i\cdot\varphi_i} \text{ in } \psi & \text{greatest n-ary fixpoint;} \end{cases}$$

下面我们给出关于这些逻辑表达式的一些解释。一般定义 $a\in\{1,2,\bar{1},\bar{2}\}$，原子命题 σ 和 Σ 中的标签相关。并且我们假设 $\bar{\bar{a}}=a$。上面所定义的表达式包含了真谓词，原子命题（表示焦点树的名称），开始命题，表达式的与、或形式，存在行条件下的表达式以及最少和最多的 n 元驻点。下面我们一般都习惯性地用 $\mu X\cdot\varphi$ 来表示 $\mu X\cdot\varphi$ in φ。

1.3 嵌入式的正则树语言

用以描述 XML 文档的类型方式有很多，例如 DTD，XML Schema，Relax NG 等等。而本文中我们要提出一种嵌入式的正则树语言 L_μ。L_μ 为 unranked 正则树表示的类型和二进制的正则树表示的类型的同构。假定一个关于类型变量 X 的可数的有限集，用二进制正则树表达的类型定义如下所示：

$L_{BT}\ni T::=$ tree type expression

$$\begin{cases} \Phi & \text{empty set;} \\ \epsilon & \text{leaf;} \\ T_1 \mid T_2 & \text{union;} \\ \sigma(X_1,X_2) & \text{label;} \\ let\ \overline{X_i\cdot T_i} \text{ in } T & \text{binder;} \end{cases}$$

我们根据正则树语言的指称语义[2]可以将一个二进制正则树转化为 L_μ，具体操作如下：

$$\|\cdot\|:L_{BT}\to L_\mu$$

$$\|T\| = \sigma\wedge\neg\sigma \text{ for } T=\phi,\epsilon$$

$$\|T_1\mid T_2\| = \|T_1\|\vee\|T_2\|$$

$$\|\sigma(X_1,X_2)\| = \sigma\wedge succ_1(X_1)\wedge succ_2(X_2)$$

$$\|let\ \overline{X_iT_i} \text{ in } T\| = \mu\overline{X_i\|T_i\|} \text{ in } \|T\|$$

式中，函数 $succ(\cdot)$ 的定义如下，其中谓词 unllable(X)用以确定是否与 X 绑定的类型 $T\neq\epsilon$ 包含空的树。

$$succ_a(X)=\begin{cases} \neg\langle\alpha\rangle T & \text{if } X \text{ is bounded to } \epsilon? \\ \neg\langle\alpha\rangle T\vee\langle\alpha\rangle X & \text{if unllable}(X) \\ \langle\alpha.X\rangle & \text{if not unllable}(X) \end{cases}$$

2 可满足性检测算法

在这一节中将展示我们的算法，并验证其正确性和完备性，以及该算法的时间复杂度。对于任意一个表达式 φ 的检查，我们的算法首先将可满足性检测表达式从其子表达式（包括他们的否定形式）中抽取出来，然后检查是否有 φ 的产生。首先介绍一下如何在 φ 中抽取它的子表达式。

2.1 初步定义

对于给定的 $\varphi=\mu\overline{X_i\cdot\varphi_i}$ in ψ，我们定义了 $\exp(\varphi)=\psi\{\mu\overline{X_i\cdot\varphi_i} \text{ in } X_i/X_i\}$ 用来表示表达式 ψ，并且其中的每一个 X_i 都由 $\mu\overline{X_i\cdot\varphi_i}$ in X_i 所替代。定义表达式 ψ 的闭包 $cl(\psi)$ 为所有的 ψ 的子表达式的集合，并且所有的驻点表达式仅能被使用一次。接下来需要定义关系 $\to_e L_\mu\times L_\mu$ 为满足下列条件的最小关系：

- $\varphi_1\wedge\varphi_2\to_e\varphi_1,\varphi_1\wedge\varphi_2\to_e\varphi_2$
- $\varphi_1\vee\varphi_2\to_e\varphi_1,\varphi_1\vee\varphi_2\to_e\varphi_2$
- $\langle a\rangle\varphi'\to_e\varphi'$
- $\mu\overline{X_i\cdot\varphi_i} \text{ in } \psi\to_e \exp(\mu\overline{X_i\cdot\varphi_i} \text{ in } \psi)$

闭包 $cl(\psi)$ 是包含 ψ 并且对于关系 $\to_e$ 闭合的最小集 R。例如，如果 $\varphi_1\in R$ 且 $\varphi_1\to_e\varphi_2$，则 $\varphi_2\in R$。$\Sigma(\psi)$ 为应用于 ψ 的原子命题 σ 的集合。我们定义 $cl^*(\psi)=cl(\psi)\cup\{\neg\varphi\mid\varphi\in cl(\varphi)\}$。每个表达式 $\varphi\in cl^*(\psi)$ 都可以被视为表达式集合的布尔合并，称之为 Lean(ψ)[3]，我们用形式化的定义来描述集合：

$$\text{Lean}(\psi)=\{\langle a\rangle T\mid a\in\{1,2,\bar{1},\bar{2}\}\}\cup\Sigma(\psi)\cup\{\text{Ⓡ}\}\{\langle a\rangle\varphi\mid\langle a\rangle\varphi\in cl(\psi)\}$$

我们定义 ψ 类型为一个满足下列条件的集合 $t\subset Lean(\psi)$：

- $\forall\langle a\rangle\varphi\in Lean(\psi),\langle a\rangle\varphi\in t\Rightarrow\langle a\rangle T\in t$ 模型的一致性。
- $\langle\bar{1}\rangle T\notin t\vee\langle\bar{2}\rangle T\notin t$ 树中的一个节点不可能既是第一个孩子又是第二个孩子。
- 当一个原子命题 $\sigma\in t$ 是，则用函数 $\sigma(t)$ 来返回类型 t 的原子命题。
- Ⓡ有可能属于 t。

满足以上条件的 ψ 类型的集合称为 Type(ψ)。对于一个 ψ 类型 t，其补操作为集合 Lean(ψ)\t。表达式的类型决定了 $cl^*(\psi)$ 中表达式的真值配置。

2.2 可满足性检测算法

我们的算法以三元组的形式 (t,w_1,w_2) 来表示，其中，t 为类型，w_1 和 w_2 为用来表示每个依据关系 $\Delta_1(t,\cdot)$ 和 $\Delta_2(t,\cdot)$ 的 t 的证明集合。算法的处理过程采用自底向上的方式，重复地增加新的三元组直到一个满意度模型被发现（例如，某个三元组的第一部分的类型包含了此表达式），或者直到更多的三元组被增加。算法的每次递归都检测类型表达的深一层树结构中是否存在未知的反向模态。不带有反向模态的类型就是满意的类型，如果该类型中包含了待检测的表达式，那么算法就结束。具体的算法表达式如图 1 所示。

算法 1 可满足性检测算法

(Satisfiability-Testing Algorithm)

$X \leftarrow \phi$

Repeat

$X' \leftarrow X$

$X \leftarrow Upd(X')$

If FinalCheck(ψ,X) then

Return "ψ is satisfiable"

Until $X = X'$

Return "ψ is satisfiable"

图 1 可满足性检测算法

在该算法中,$X \subseteq Types(\psi) \times 2^{Types(\psi)} \times 2^{Types(\psi)}$,算法中使用到的操作符 $Upd(\cdot)$和函数 $FinalCheck(\cdot)$的定义如下:

$$Upd(X) = X \cup \{(t, w_1(t, X^o), w_2(t, X^o)) \mid ® \notin t \subseteq Types(\psi) \wedge \langle 1\rangle T \in t \Rightarrow w_1(t, X^o) \neq \phi \wedge \langle 2\rangle T \in t \Rightarrow w_2(t, X^o) \neq \phi\}$$

在这个操作符的定义中用到的函数定义如下:

$$w_a(t, X) = \{type(x) \mid x \in X \wedge \langle \bar{a}\rangle^T \in type(x) \wedge \Delta_a(t, type(x))\}$$

$$X^{®} = \{x \in X \mid x = (\_, \_, \_)^{®}\};$$

$$X^{o} = \{x \in X \mid x = (\_, \_, \_)\};$$

$$FinalCheck(\psi, X) = \exists x \in X^{®}, dsat(x, \psi) \wedge \forall a \in \{\bar{1}, \bar{2}\}, \langle a\rangle T \notin type(x);$$

$$dsat((t, w_1, w_2), \psi) = \psi \in t \vee \exists', dsat(x', \psi) \wedge (x' \in w_1 \vee x' \in w_2);$$

$$type((t, w_1, w_2)) = t。$$

2.3 算法的正确性证明和复杂度计算

在本小节中,我们从理论上来证明可满足性检测算法的正确性及完备性,并且证明其时间复杂度为 $2^{O(|Lean(\psi)|)}$。

我们的证明方法是基于以下两项已知结论。第一,XML 正则树类型和 XPath 表达式可以线性地转换为无环的表达式。第二,对于有限树最少和最多的不动点是相等的,因此在否定的情况下逻辑是闭合的。

2.3.1 正确性证明

在证明可满足性检测算法的正确性之前必须先证明算法是可终止的。

证明:对 $\psi \in L_\mu$,由于 $cl(\psi)$为有限集合,$Lean(\psi)$和 $2^{Lean(\psi)}$也为有限集合,并且 $Upd(\cdot)$为单调函数,每个 X^i 都包含于有限集 $Types(\psi) \times 2^{Types(\psi)} \times 2^{Types(\psi)}$,因此算法是可终止的。

在证明了算法的可终止性后,我们可以证明算法的正确性。在证明正确性之前首先介绍表达式的部分满足性的概念,反向模式被用来检查给定的层次。一个表达式 φ 为部分满足的当且仅当 $\| \varphi \|^{\eta}_{V} \neq \phi$(详细定义见图 2)。

$$\| T \|^{\eta}_{V} = F \qquad \| X \|^{\eta}_{V} = V(X)$$

$$\| \varphi \vee \psi \|^{\eta}_{V} = \| \varphi \|^{\eta}_{V} \cup \| \psi \|^{\eta}_{V}$$

$$\| p \|^{\eta}_{V} = \{f \mid nm(f) = p\}$$

$$\| \varphi \wedge \psi \|^{\eta}_{V} = \| \varphi \|^{\eta}_{V} \cap \| \psi \|^{\eta}_{V}$$

$$\| \neg p \|^{\eta}_{V} = \{f \mid nm(f) \neq p\}$$

$$\| \langle \bar{a}\rangle \varphi \|^{\eta}_{V} = F$$

$$\| ® \|^{\eta}_{V} = \{f \mid f = (\sigma^{®}[tl], c)\}$$

$$\| \langle \bar{a}\rangle \varphi \|^{\eta > 0}_{V} = \{f\langle a\rangle \mid f \in \| \varphi \|^{\eta - 1}_{V} \wedge f\langle a\rangle defined\}$$

$$\| \langle a\rangle \varphi \|^{\eta}_{V} = \{f\langle \bar{a}\rangle \mid f \in \| \varphi \|^{\eta + 1}_{V} \wedge f\langle \bar{a}\rangle defined\}$$

$$\| \neg\langle a\rangle T \|^{\eta}_{V} = \{f \mid f\langle a\rangle undefined\}$$

$$\| \mu \overline{X_i \varphi_i} \text{ in } \psi \|^{\eta}_{V} \equiv let\ T_i = (\bigcap \{\overline{T_i} \subseteq \overline{F} \mid \| \overline{\varphi_i} \|^{\eta}_{V_{\overline{T_i/X_i}}} \subseteq \overline{T_i}\})_i \text{ in } \| \psi \|^{\eta}_{V_{\overline{T_i/X_i}}}$$

图 2 部分可满足性

对于一个类型 t,我们定义 $\varphi_c(t)$为其最大约束表达式,其中的原子来自 $Lean(\psi)$。在下面的公式中,o 代表®,如果®$\in t$,反之则代表$\neg$®。

$$\varphi_c(t) = \sigma(t) \wedge \bigcap_{\sigma \in \Xi, \sigma \notin t} \neg\sigma \wedge o \wedge \bigcap_{\langle a\rangle \sigma \in t} \langle a\rangle \varphi \wedge \bigcap_{\langle a\rangle \sigma \notin t} \neg\langle a\rangle \varphi$$

接下来定义模态的串联路径 ρ,若路径为空则定义为ε,路径串联则定义为 ρa。每条路径都被给定一个深度 depth:

$\text{depth}(\varepsilon) = 0$

$\text{depth}(\rho a) = \text{depth}(\rho) + 1, \text{if } a \in \{1, 2\}$

$\text{depth}(\rho a) = \text{depth}(\rho) - 1, \text{if } a \in \{\bar{1}, \bar{2}\}$

一个正向传输路径为只与正向形态相关的路径。我们定义一个类型树 T,其节点为类型 $T(\cdot) = t$,每个节点至多有两个孩子,$T\langle 1\rangle$和 $T\langle 2\rangle$。一个类型树是连贯的(consistent)当且仅当对于每个正向通路 ρ 及 $T\langle\rho\rangle$每个孩子 a,有 $T\langle\rho\rangle(\cdot) = t$,$T\langle\rho a\rangle(\cdot) = t'$,即$\langle a\rangle T \in t$,$\langle \bar{a}\rangle T \in t'$,$\Delta_a(t, t')$。而对于给定的连贯的类型树 T,我们定义其依赖图的节点是正向通路 ρ 的两端节点和 $t = T\langle\rho\rangle(\cdot)$中的表达式或者是 t 的补集中表达式的否。图的定向边上的标签表示是连贯性的类型树的模态。由于每一次的叠代都要覆盖更远的正向通路,因此这个图与算法的最终结果相关。

定理 1 一个无环的的表达式的连贯的类型树的依赖图是无圈的[3](cycle-free)。

算法的正确性的证明基于以上的定义和定理,假定 T 是算法的结果集合 $t \in T$ 以及任意的 $\varphi \in t$,都存在 $\| \varphi \|^{0}_{\eta} \neq \phi$,则算法是正确的,证明的思路如下。

我们采用归纳法对算法进行证明。对每个 T^n 中的 t 和每个以 t 为根节点依据 X^n 构造的基点标志树 Γ,我们知道 Γ 为一个连续的树类型,据此构建有限的焦点树 f,即为第 0 层的符合表达式 $\varphi_c(t)$,然后依此方式对下一层的依赖树进行归纳。

2.3.2 完备性证明

对于一个闭合的且无环的的表达式 $\varphi \in L_\mu$ 如果 $\| \varphi \|^{0}_{\phi} \neq \phi$,则算法终止于一个三元组集合 $FinalCheck(\varphi, X)$。

证明思路:由于表达式是满意的,我们认为必定存在一个最小的焦点树 f 也是满意的。因此可以推导出包含 f 的有限的关系 φ,然后就可以依据这个关系构建算法的一次运行来产生一个包含表达式的不带反向模态的类型。

2.3.3 复杂度计算

对于 $\psi \in L_\mu$,可满足性检测问题 $\| \varphi \|_{\phi} \neq \phi$ 的执行时间为 $2^{o(n)}$,其中 $n = |Lean(\psi)|$。

3 算法的分类应用及实验结果

对于一组 XPath 表达式 $e_1, \cdots, e_n$,我们可以形式化地描述 XML 类型表达式 $T_1, \cdots, T_n$ 中的许多决策问题。

• XPath 中的包含决策问题

$E^{\leftarrow}||e_1||_{||T_1||} \wedge E^{\leftarrow}||e_2||_{||T_2||}$(如果表达式不符合可满

(下转第 211 页)

LNCS 2393, Borovets, Bulgaria, July 2002: 15-19
[15] 袁洋,李善平. 基于语义 Web 的本体映射方法综述[J]. 计算机科学,2004,31(5):5-8
[16] 曹泽文,钱杰,张维明,等. 一种综合的概念相似度计算方法[J]. 计算机科学,2007,34(03):174-191
[17] Castano S, Ferrara A, Montanelli S. Matching Ontologies in Open Networked Systems: Techniques and Applications [EB/OL]. http://islab.dico.unimi.it/hmatch/downloads.php?cat_id=2,2007-9-20
[18] Joost C, van Rijsbergen K. Information retrieval[EB/OL]. http://www.dcs.gla.ac.uk/Keith/Preface.html,2007-9-21
[19] Hong-Hai D, Sergey M, Erhard R. Comparison of Schema Matching Evaluations [J]. Lecture Notes in Computer Science, 2003 (2593): 221-237
[20] Avesani P, Giunchiglia F, Yatskevich M. A large scale taxonomy mapping evaluation[C]// Proceedings of the International Semantic Web Conference (ISWC). 2005: 67-81
[21] Hirst G, St-Onge D. Lexical Chains as representations of context for the detection and correction of malapropisms[M]. Fellbaum, 1998: 305-332

(上接第 174 页)

足性检测算法,则节点的选择由在约束类型表达式 T_1 下对 e_1 的节点选择和在约束类型表达式 $\cdot T_2$ 下对 e_2 的节点选择共同决定)

• XPath 中的空集决策问题:$E^{\rightarrow}||e_1||_{||T_1||}$

• XPath 中的重叠决策问题:$E^{\rightarrow}||e_1||_{||T_1||} \wedge E^{\rightarrow}||e_2||_{||T_2||}$

• XPath 中的覆盖决策问题:$E^{\rightarrow}||e_1||_{||T_1||} \wedge \wedge_{2\leqslant i\leqslant n} \neg E^{\rightarrow}||e_i||_{||T_i||}$

可满足性检测算法还可以用来对 XML 类型检查中的两类问题进行处理。

• 带注释的 XPath 查询的静态类型检查

$E^{\rightarrow}||e_1||_{||T_1||} \wedge \neg||T_2||$(如果表达式不满足可满足性检测算法,则所有的节点都来自于包含于约束类型表达式 T_2 下的 T_1 相对于对 e_1 的节点选择)

• 类型约束下的 XPath 等价转化

$E^{\rightarrow}||e_1||_{||T_1||} \wedge \neg E^{\rightarrow}||e_2||_{||T_2||}$ 和 $E^{\rightarrow}||e_1||_{||T_1||} \wedge E^{\rightarrow}||e_2||_{||T_2||}$(这个测试可以用于动态地检查那些经过 T_2 修改后的 T_1 以及用同样的方式经过 e_2 修改过的 e_1 得到的节点,在查询过程中经常会遇到当一个输入类型发生改变时相应的 XPath 查询也必须同时改变的这种情况)。

现有的研究中没有出现过应用定位逆向轴进行递归操作的实例,因此我们简单地提供证据来证明我们的方法是有效的。我们进行的扩展性实验[7],所展示的仅仅是具有代表性的示例,包括许多复杂的语言特征,例如递归的前向和后向轴、交集和带字母表的递归形式。实验使用到的 XML 数据的类型表示如表 1 所列,XPath 表达式如表 2 所列(其中"//"是"/desc-or-self::*/"是缩写),表 3 描述了一些决策问题以及相关的实验结果。运行时间以毫秒为单位。

表 1 实验中使用到的数据类型

DTD	Symbols	Binary Type Variable
SMIL 1.0	19	11
XHTML1.0 Strict	77	325

第一个包含实例的 XPath 首先在文献[8]中作为示例来计算,假定树模式的同构技术并不完全。e_8 的示例说明官方的 XHTML DTD 并没有禁止语义上的嵌套。对于一个 XHTML 的示例,我们观察到所需的时间是非常重要的,但是仍与实际相关,尤其对于那些在编译时间执行的静态分析操作。

表 2 实验中使用到的 XPath 表达式

e1: /a[.//b[c/*//d]/b[c//d]/b[c/d]]
e2: /a[.//b[c/*//d]/b[c/d]]
e3: a/b//c/foll-sibling::d/e
e4: a/b//d[prec-sibling::c]/e
e5: a/c/following::d/e
e6: a/b[//c]/following::d/e∩a/d[prec-sibling::video]
e7: *//switch[ancestor::head]//seq//audio[pre-sibling::video]
e8: descendant::a[ancestor::a]
e9: / descendant::*
e10: html/(head|body)
e11: html/head/descendant::*
e12: html/body/descendant::*

表 3 一些决策问题以及相关的实验结果

XPath 决策问题	XML 类型	时间(ms)
$e_1 \subseteq e_2$ 且 $e_2 \not\subseteq e_1$	none	353
$e_3 \subseteq e_4$ 且 $e_4 \subseteq e_3$	none	45
$e_6 \subseteq e_5$ 且 $e_5 \not\subseteq e_6$	none	41
e_7 是可满意的	SMIL1.0	157
e_8 是可满意的	XHTML1.0	2630
$e_9 \subseteq (e_{10} \cup e_{11} \cup e_{12})$	XHTML1.0	2872

结束语 本文的主要工作是提出了一个正确的、完备的涉及正则树类型的决策问题的可满足性检测算法。我们的算法以有限树的子逻辑为基础,算法的证明方法显示了这些逻辑规则和 XPath 决策问题之间的联系。首先,XML 正则树类型和 XPath 片段之间的转化是无环、线性表达式大小的;其次,在一个有限树中,在否定情况下带有一个驻点的操作符的逻辑是闭合的,这就使得我们可以解决关键的 XPath 决策问题。如何将对于 XPath 类型的决策问题的研究扩展到对于数值比较的决策问题的研究将是下一步的研究方向。

参 考 文 献

[1] Huet G P. Functional Pearl the Zipper[J]. J. Functional rogramming, 1997, 7(5): 549-554
[2] Hosoya H, Vouillon J, Pierce B C. Regular expression types for XML[J]. ACM Trans. Program. Lang. Syst., 2005, 27(1): 46-90
[3] Pan G, Sattler U, Vardi M Y. BDD-based decision proceduresfor the modal logic K[J]. Journal of Applied Non-classical Logics, 2006, 16(1/2): 169-208
[4] Benedikt M, Fan W, Geerts F. XPath satisfiability in the presence of DTDs[C]// PODS '05: Proceedings of the twenty-fourth ACM Symposium on Principles of Database Systems. 2005: 25-36
[5] Genevès P, Layaïda N. Deciding XPath containement with MSO [J]. Data & Knowledge Engineering, 2007, 63(1): 108-136
[6] Mø ller A, Schwartzbach M I. The design space of type checkers for XML transformation languages[C]// Proc. Tenth International Conference on Database Theory, ICDT '05, volume 3363 of LNCS. 2005(1): 17-36
[7] Genevès P, Layaïda N, Schmitt A. A satisfiability solver for XML and XPath[J]. Journal of the ACM, 2007, 6(42): 342-351
[8] Miklau G, Suciu D. Containment and equivalence for a fragment of XPath[J]. Journal of the ACM, 2004, 51(1): 2-45