一类受限正则表达式的推断算法

冯晓强[1,2] 郑黎晓[3] 陈海明[1]
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[1]
(中国科学院大学 北京 100049)[2] (华侨大学计算机科学与技术学院 厦门 361021)[3]

摘 要 XML模式推断问题的主要任务可以归约为从一个句子集合中推断出对应的确定型正则表达式。提出了一类在XML模式中大量出现的受限正则表达式,给出了该类正则表达式的推断算法。该算法首先根据给定的句子集合构造自动机,然后根据自动机和句子集合推断出对应的正则表达式。该算法的时间复杂度为 $\max(O(|V|+|E|), O(L))$,其中 V 和 E 分别表示自动机的节点集合和边集合,L 表示句子集合中所有句子的长度之和。对算法的终止性和正确性进行了证明。

关键词 XML模式,模式推断,正则表达式,自动机,算法
中图法分类号 TP301 **文献标识码** A

Inferring Algorithm for a Subclass of Restricted Regular Expressions

FENG Xiao-qiang[1,2] ZHENG Li-xiao[3] CHEN Hai-ming[1]
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(University of Chinese Academy of Sciences, Beijing 100049, China)[2]
(College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China)[3]

Abstract The problem of inferring XML schemas reduces to inferring deterministic regular expressions from a set of sentences. A subclass of restricted regular expressions which commonly occur in practical XML schemas was proposed. An algorithm for inferring this kind of regular expressions was described. The algorithm first constructs the corresponding automata according to the sentence set, then infers the regular expression from the automata and the sentence set. The complexity of the algorithm is $\max(O(|V|+|E|), O(L))$ where V and E are the set of states and the set of edges of the constructed automata respectively, and L is the total length of sentences. The termination and correctness of the algorithm were proved.

Keywords XML schema, Schema inference, Regular expression, Automata, Algorithm

1 引言

XML(eXtensible Markup Language)是指可扩展标记语言,它是互联网中用于数据交换的通用的标准文档格式[1]。XML简单灵活的特点以及良好的可扩展性使其广泛应用于很多领域。在应用中,XML文档的结构和类型要遵循XML模式语言(简称XML模式)的定义。常用的XML模式定义语言有Document Type Definition (DTD)[2]和XML Schema Definition (XSD)[3]。

XML模式在XML数据处理中扮演着非常重要的角色,例如:可以用于自动验证XML文档结构的有效性;在XML数据处理的优化问题中[4],其起到了重要的作用;在通过模式匹配进行数据集成的应用中[5],其是必要的。然而在实际应用中,经常存在XML模式缺失的情况。例如,文献[6]的统计结果显示互联网中可用的XML文档中将近一半没有给定对应的模式。另外文献[7,8]的研究显示,从互联网中抓取的XSD中大约有2/3不满足W3C XML Schema规范[3]。关于DTD也有类似的统计结果[9]。

既然XML模式如此重要,而实际中的很多XML文档没有对应的模式或给定的模式不符合规范,那么由此产生的一个问题是:如何从XML文档集中推断出一个对应的模式(DTD或XSD)?

例1是一个电视节目表的DTD实例。

例1 电视节目表的DTD实例:

```
<!ELEMENT tvschedule (channel+)>
<!ELEMENT channel (banner,day+)>
<!ELEMENT day (date,(holiday | programslot+)+)>
<!ELEMENT programslot (time,title,description)>
```

到稿日期:2013-06-22 返修日期:2013-08-23 本文受国家自然科学基金项目(61070038),华侨大学人才引进科研启动基金项目(12BS215)资助。

冯晓强(1987—),男,博士生,CCF学生会员,主要研究方向为形式语言与自动机理论,E-mail:fengxq@ios.ac.cn;郑黎晓(1983—),女,博士,讲师,主要研究方向为软件测试、数据库系统与理论;陈海明(1966—),男,博士,研究员,CCF高级会员,主要研究方向为形式语言与自动机理论、计算模型、形式规约和程序分析。

DTD中的每一个元素都有一个正则表达式与之对应，该正则表达式称为元素的内容模型。例如元素 day 对应于正则表达式(date,(holiday | programslot$^+$)$^+$)，这里的“,”表示连接运算，“|”表示并运算。如果要推断出该 DTD，就必须推断出每个元素对应的正则表达式。W3C 规范要求 XML 模式中出现的表达式必须为确定型正则表达式(参见定义 2)。也就是说，模式推断问题的核心部分可以归约为：从给定的句子集合中推断出对应的确定型正则表达式。

Bex 等人[10]证明了从句子集合中推断确定型正则表达式类是不可行的，因此可以考虑推断受限的确定型正则表达式。文献[7]中的统计结果显示，XML 模式中出现的正则表达式绝大部分为 single occurrence regular expression(Sore，参见定义 3)，Sore 中大部分为 chain regular expression(Chare，参见定义 4)。根据定义可知 Sore 和 Chare 是确定型正则表达式的子类，Bex 等人[11]给出了这两类表达式的推断算法。但我们发现 Sore 中有一部分正则表达式的形式与 Chare 类似，但不满足 Chare 的定义，例 1 中的(date,(holiday | programslot$^+$)$^+$)就不是一个 Chare。为此本文对 Chare 进行扩展，提出了 Extended chain regular expression(Echare，参见定义 5)，并给出了 Echare 的推断算法。该算法的时间复杂度为 $\max(O(|V|+|E|),O(L))$，其中 V 和 E 分别表示 single occurrence automaton(Soa，参见定义 6)的节点集合和边集合，L 表示句子集合中所有句子的长度之和。

2 基本概念

2.1 正则表达式

定义 1 Σ 表示一个字母表(终极符的集合)，Σ 上的正则表达式[11]的形式如下：

$r,s=\emptyset|\varepsilon|a|r\cdot s|r+s|r?|r^+$

具体的正则表达式中可以用圆括号表示优先级关系。上式中，$\emptyset$ 表示空集，ε 表示空字符，$a\in\Sigma$。对于两个正则表达式 r 和 s，$r\cdot s$ 表示连接运算，$r+s$ 表示并运算，r^+ 表示 r 出现一次或多次，$r?$表示 r 出现零次或一次。上述形式中没有出现闭包操作符 $*$，r^* 表示 r 出现零次或多次，等价于$(r?)^+$或$(r^+)?$。$L(r)$表示正则表达式 r 所代表的语言，$L(r)$的定义如下：

$L(\emptyset)=\emptyset$

$L(\varepsilon)=\{\varepsilon\}$

$L(a)=\{a\}$

$L(r\cdot s)=\{vw|v\in L(r),w\in L(s)\}$

$L(r+s)=L(r)\cup L(s)$

$L(r^+)=\{v_1\cdots v_n|n\geqslant 1$ 且 $v_1,\cdots,v_n\in L(r)\}$

$L(r?)=L(r)\cup\{\varepsilon\}$

对于一个正则表达式 r，可以给 r 中每个字符添加不同的下标来表示字符的出现位置。添加了下标的正则表达式记为 r'，称 r'为 r 的标记表达式[12]。我们用符号“#”表示对字符进行添加或去掉下标的运算。

例如，$r=ab^+(c?+d)$，$r'=a_1b_2^+(c_3?+d_4)$。其中 $a^\#=a_1$，$b_2{}^\#=b$，以此类推(为了方便表示，本文所有例子中均省去了连接运算符)。

定义 2 一个正则表达式 r 是确定型正则表达式[12]，当且仅当对 $L(r')$中所有的句子 uxv,uyw，其中 $|x|=|y|=1$，如果 $x\neq y$，那么 $x^\#\neq y^\#$。

定义 3 一个正则表达式中，如果每个字符最多只出现 1 次，那么称该表达式为 single occurrence regular expression[11]，简称 Sore。

例如，$(((a^+(b+c)?)^+)d)^+$是一个 Sore，$a?(ba+c)$不是。

定义 4 如果一个 Sore 满足如下形式：$f_1\cdot\cdots\cdot f_n(n\geqslant 0)$，其中每个 f_i 是一个链式因子，而且每个链式因子的形式为$(a_1+\cdots+a_k)$、$(a_1+\cdots+a_k)?$、$(a_1+\cdots+a_k)^+$或$(a_1+\cdots+a_k)^+?$，其中 $k\geqslant 1$，$a_j(1\leqslant j\leqslant k)$为终极符，那么称该正则表达式为 chain regular expression[11]，简称 Chare。

例如，$(a+b)^+?c?(d+e+f)^+$是一个 Chare，$(ab^++c+d)?(e?+f+g^+)^+$不是。

定义 5 如果一个 Sore 满足如下形式：$f_1\cdot\cdots\cdot f_n(n\geqslant 0)$，其中每个 f_i 是一个链式因子，每个链式因子 f_i 的形式为$(b_1+\cdots+b_k)$、$(b_1+\cdots+b_k)?$、$(c_1+\cdots+c_k)^+$或$(c_1+\cdots+c_k)^+?$，其中 $k\geqslant 1$，$b_j(1\leqslant j\leqslant k)$为 a 或 a^+，$c_j(1\leqslant j\leqslant k)$为 a、$a?$、a^+或 $a^+?$，a 为终极符，那么称该正则表达式为 Extended chain regular expression，简称 Echare。

例如，$(a+b^+)(c^++d?+e)^+?(f^++g)?$是一个 Echare，但不是一个 Chare。

2.2 自动机

定义 6 Σ 表示一个字母表，*src* 和 *sink* 是两个特殊的符号，*src* 节点表示开始状态，*sink* 节点表示接受状态。Σ 上的 single occurrence automaton[11](简称 Soa)是一个满足以下两个条件的有向图 $A=(V,E)$：

(1)$\{src,sink\}\subseteq V$，$V\subseteq\Sigma\cup\{src,sink\}$，$V$ 中的每个节点都有节点标记，对应于 Σ 中的一个终极符。

(2)*src* 节点只有出边，*sink* 节点只有入边，每一个节点 $v\in V$都位于 *src* 到 *sink* 的路径上。

一个句子 $w=a_1\cdots a_n(n\geqslant 0)$被 Soa 接受，当且仅当在 Soa 中存在这样一条路径：$src\rightarrow v_1\rightarrow\cdots\rightarrow v_n\rightarrow sink$，其中节点 v_i 的标记为 $a_i(1\leqslant i\leqslant n)$。$L(A)$表示所有被 Soa $A=(V,E)$接受的句子的集合，即被 A 接受的语言。

Soa 的定义和通用的自动机定义有些区别，Soa 中的边没有迁移标记，每个节点都对应于一个终极符，称为节点标记。不难看出 Soa 是确定型有穷自动机(Deterministic Finite Automaton，简称 DFA)的子类，且每个 Soa 都存在一个等价的 DFA 与之对应。如果在 DFA 中存在一条迁移标记为 $a\in\Sigma$ 的边指向状态 q，那么相应地在 Soa 中存在一个标记为 a 的节点有一条入边。DFA 中的开始状态对应于 Soa 中的 *src* 节点，DFA 中的接受状态对应于 Soa 中所有到 *sink* 节点存在迁移边的节点。

例 2 如图 1 所示的自动机是一个 Soa，图 2 是对应的 DFA。

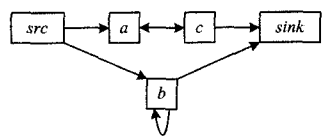

图 1 Soa

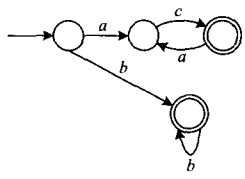

图 2 DFA

定义 7 一个 generalized Soa[11] 是一个满足以下两个条件的有向图 $A=(V,E)$：

(1) $\{src,sink\}\subseteq V$，$V-\{src,sink\}$ 中的所有节点的节点标记是一个 Sore，且任意两个节点的标记中不会出现相同的终极符。

(2) *src* 节点只有出边，*sink* 节点只有入边，每一个节点 $v\in V$ 都位于 *src* 到 *sink* 的路径上。

一个句子 $w=w_1\cdots w_n(n\geqslant 0)$ 被 Soa 接受，当且仅当在 Soa 中存在这样一条路径：$src\rightarrow v_1\rightarrow\cdots\rightarrow v_n\rightarrow sink$，其中节点 v_i 的标记为 $r_i(1\leqslant i\leqslant n)$，$w_i\in L(r_i)$。由定义可知，Soa 属于 generalized Soa。$L(A)$ 表示所有被 generalized Soa $A=(V,E)$ 接受的句子的集合，即被 A 接受的语言。

定义 8 Σ 表示一个字母表，对每一个 Σ 上的正则表达式 r 定义如下集合：

$first(r)=\{a\mid aw\in L(r),a\in\Sigma,w\in\Sigma^*\}$

$last(r)=\{a\mid wa\in L(r),w\in\Sigma^*,a\in\Sigma\}$

$follow(r,a)=\{b\mid uabv\in L(r),u,v\in\Sigma^*,b\in\Sigma\},a\in\Sigma$

例如，给定正则表达式 $r=a(b+c^+)^+$，$first(r)=\{a\}$，$last(r)=\{b,c\}$，$follow(r,b)=\{b,c\}$。

定义 9 一个正则表达式 r 对应的 Glushkov 自动机[13] $M_r(Q,\Sigma,\delta,q_I,F)$ 定义如下：

(1) $Q=\Sigma_{r'}\cup\{q_I\}$，状态集合为 r' 的位置集合加上初始状态 q_I。

(2) 对每个 $a\in\Sigma$，$\delta(q_I,a)=\{x\mid x\in first(r'),x^{\#}=a\}$。

(3) 对每个 $a\in\Sigma$ 和 $x\in\Sigma_{r'}$，$\delta(x,a)=\{y\mid y\in follow(r',x),y^{\#}=a\}$。

(4) 如果 $\varepsilon\in L(r)$，那么 $F=last(r')\cup\{q_I\}$；否则，$F=last(r')$。

定理 1[13] 对于正则表达式 r 和其对应的 Glushkov 自动机 M_r，有：$L(M_r)=L(r)$。$L(M_r)$ 表示 M_r 接受的语言。

对于一个 Echare r，其对应的 Glushkov 自动机 M_r 显然是一个 DFA，并且存在一个等价的 Soa $A=(V,E)$ 与 M_r 对应，即 $L(A)=L(M_r)$。

3 生成 Echare

本文提出的 GenEchare 算法是由给定的句子集合 S 以及 Soa 生成 Echare。在介绍该算法之前，先介绍如何由给定的句子集合生成 Soa。

3.1 Soa 的生成

我们可以运用 2T-INF 算法[10,13] 来解决这个问题。该算法的详细过程如下：

输入：一个有限句子集合 S

输出：满足 $S\subseteq L(A)$ 的 Soa $A=(V,E)$

步骤 1 计算在 S 中出现的所有字符，设该字符集合为 C，$V=C+\{src,sink\}$，其中 *src* 为开始状态，*sink* 为接受状态；初始化 $E=\varnothing$。

步骤 2 对于 S 中的每个句子 $a_1\cdots a_n(n\geqslant 1)$，将边 $(src,a_1),(a_1,a_2),\cdots,(a_n,sink)$ 加入到 E 中。

步骤 3 返回 $A=(V,E)$。

下面用一个简单的例子来说明算法的过程。

例 3 给定句子集合 $S=\{adadf,beggg,becbeg\}$，节点集合 $V=\{src,a,b,c,d,e,f,g,sink\}$，边的集合 $E=\{(src,a),(a,d),(d,a),(d,f),(f,sink),(src,b),(b,e),(e,g),(g,g),(g,sink),(e,c),(c,b)\}$。由 2T-INF 生成的 Soa 如图 3 所示。

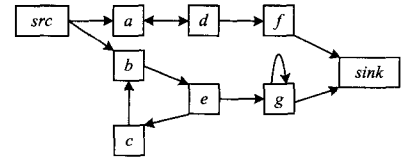

图 3 Soa $A=(V,E)$

定理 2[14] 对于给定句子集合 S 和由 2T-INF 算法生成的 Soa $A=(V,E)$，有 $S\subseteq L(A)$ 成立。

3.2 GenEchare 算法

在描述算法细节之前先介绍一些算法中用到的概念。

定义 10 设 $B=(V',E')$ 是 Soa $A=(V,E)$ 的一个非平凡强连通分量，那么 B 也是 A 的一个极大强连通子图，且 B 至少包含两个节点。对于 B 中任意两个节点 a、$b\in V'$，既存在一条从 a 到 b 的路径，也存在一条从 b 到 a 的路径。

例如在图 3 所示的 Soa $A=(V,E)$ 中，根据上述定义可知，A 共有两个非平凡强连通分量，分别为 $C_1=\{a,d\}$，$C_2=\{b,c,e\}$。

定义 11 在 Soa $A=(V,E)$ 中，一个节点 $v\in V$ 的级数[15] 是指从节点 *src* 到节点 v 的最长路径的长度，显然这里的 A 为有向无环图。根据定义可知，*src* 节点的级数为 0，对于其他任意一个节点 v，设 v 的所有前驱节点中最大的级数为 *level*，则 v 的级数为 *level*+1。

定义 12 在 Soa $A=(V,E)$ 中，称级数 L 为跳级[15]，当且仅当下述条件成立：存在两个节点 u、$v\in V$，从 u 到 v 有一条边，使得 $level_u<L$ 和 $level_v>L$ 均成立。其中，$level_u$、$level_v$ 分别为 u 和 v 的级数。

例 4 给定如图 4 所示的 Soa $A=(V,E)$。根据上述定义，从 *src* 节点开始依次给每个节点赋值对应的级数。级数为 0 的节点集合为 $\{src\}$，级数为 1 的节点集合为 $\{a,b,c\}$，级数为 2 的节点集合为 $\{d,e\}$，级数为 3 的节点集合为 $\{f,g\}$，级数为 4 的节点集合为 $\{h\}$，级数为 5 的节点集合为 $\{sink\}$。其中，节点 d 到节点 h 有一条边，所以级数 3 为跳级。同理可知级数 4 也为跳级。

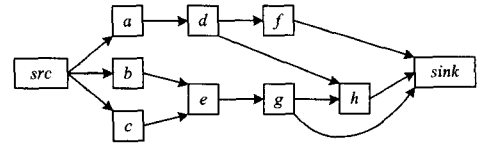

图 4 Soa $A=(V,E)$

GenEchare 算法的基本思想是：首先将 Soa 中非平凡强连通分量进行合并，从而将 Soa 简化成有向无环图；然后给每个节点赋值对应的级数，每个级数对应的节点代表了 Echare 的一个或几个链式因子；最后将所有的链式因子按级数顺序进行连接运算生成 Echare。算法的详细步骤如下：

GenEchare

输入：句子集合 S，Soa $A=(V,E)$

输出：Echare *echare*

步骤 1　计算出 A 中所有带有自环的节点（即到自身有迁移边的节点），设该节点集合为$\{v_1,\cdots,v_m\}$，将集合内 v_i 节点的标记变为 $v_i^+(1\leqslant i\leqslant m)$，并将自环去掉。该算法中认为节点名称和节点标记是相同的概念，此时 A 是一个 generalized Soa，记为 $A'=(V',E')$。

步骤 2　计算 A' 中所有的非平凡强连通分量，分别记为 $C_1,\cdots,C_n$。对于任意一个非平凡强连通分量 $C_i(1\leqslant i\leqslant n)$，设其分量内部节点集合为$\{v_1,\cdots,v_m\}$，依次遍历 S 中的每个句子，计算出每个句子对应的包含$\{v_1,\cdots,v_m\}$中节点的最长子串。若这些子串中存在某个子串不包含节点 $v_j(1\leqslant j\leqslant m)$，那么将 v_j 的标记变为 $v_j?$。然后将该集合中的所有节点合并成为一个节点，并将该节点的标记记为$(v_1+\cdots+v_m)^+$。对所有的非平凡强连通分量按上述方法处理，记此时的generalized Soa为 $A''=(V'',E'')$。

步骤 3　经过步骤 2 之后，A''为有向无环图。给 A''中的每个节点赋值对应的级数，并计算出哪些级数为跳级。

步骤 4　将 *echare* 的初始值赋为 ε。设 *level* 为节点的级数，将 *level* 的初始值赋为 1，依次执行以下步骤：

(1)设级数为 *level* 且节点标记中只含有一个终极符的节点集合为 V_s，设级数为 *level* 且节点标记中含有多于一个终极符的节点集合为 V_t。

(2)如果 V_t 不为空集，设 $V_t=\{v_1,\cdots,v_k\}$，那么当级数 *level* 为跳级或者 V_s 和 V_t 的节点总数大于 1 时：

$echare=echare\cdot v_1?\cdot\cdots\cdot v_k?$

否则，当级数 *level* 不为跳级且 V_s 和 V_t 的节点总数等于 1 时：

$echare=echare\cdot v_1$

因为 V_t 不为空集，所以此时 V_s 为空集且 V_t 中只有一个节点。

(3)如果 V_s 不为空集，设 $V_s=\{v_1,\cdots,v_m\}$，那么当级数 *level* 为跳级或者 V_t 不为空集时：

$echare=echare\cdot(v_1+\cdots+v_m)?$

否则，当级数 *level* 不为跳级且 V_t 为空集时：

$echare=echare\cdot(v_1+\cdots+v_m)$

(4)将级数 *level* 加 1，跳至(1)继续执行，直到 *level* 等于 *sink* 节点的级数为止。

步骤 5　返回 *echare*。

接下来用一个实例描述该算法的执行过程。

例 5　给定句子集合 $S=\{aadfhh,adadfi,begggi,becbeg\}$。该句子集合对应的 Soa $A=(V,E)$如图 5 所示。

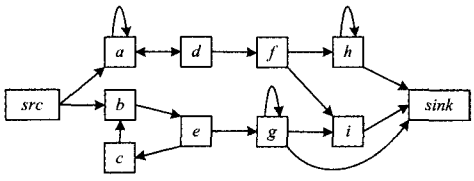

图 5　Soa $A=(V,E)$

首先计算 A 中的带自环的节点，该节点集合为$\{a,g,h\}$，将其节点标记变为 a^+,g^+,h^+。此时 A 为一个 generalized Soa $A'=(V',E')$，如图 6 所示。

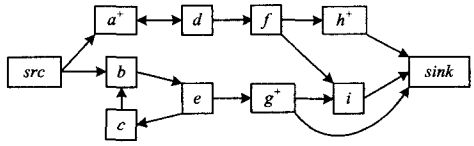

图 6　Soa $A'=(V',E')$

然后计算非平凡强连通分量，结果为 $C_1=\{a^+,d\}$，$C_2=\{b,e,c\}$。对于分量 C_2，句子 *begggi* 包含 C_2 中节点的最长子串为 *be*，该子串不包含节点 *c*，所以将节点 *c* 的标记变为 *c*?。然后将每个分量各自合并生成新的 Soa $A''=(V'',E'')$，如图 7 所示。

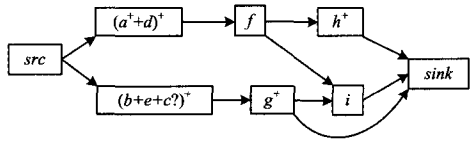

图 7　Soa $A''=(V'',E'')$

然后给每个节点赋值对应的级数。级数为 1 的节点集合为$\{(a^++d)^+,(b+e+c?)^+\}$，级数为 2 的节点集合为$\{f,g^+\}$，级数为 3 的节点集合为$\{h^+,i\}$，级数为 4 的节点集合为$\{sink\}$。其中节点 g^+到节点 *sink* 有一条边，所以级数 3 为跳级。根据算法步骤 4 可得，级数 1、2、3 对应的链式因子分别为：$(a^++d)^+?$、$(b+e+c?)^+?$、$(f+g^+)$、$(h^++i)?$，将所有的链式因子按级数大小依次进行连接运算后得到的 *echare* 为$(a^++d)^+?(b+e+c?)^+?(f+g^+)(h^++i)?$。

4　算法分析与证明

4.1　算法复杂度分析

设句子集合 S 中所有句子的长度之和为 L，Soa 的节点集合为 V，边的集合为 E。步骤 1 中计算带自环的节点，直接遍历每个节点即可，复杂度为 $O(|V|)$。步骤 2 中非平凡强连通分量的计算可以运用深度优先搜索算法[16]，复杂度为 $O(|V|+|E|)$；依次遍历每个句子计算最长子串的复杂度为 $O(L)$。步骤 3 中级数的相关计算可以运用广度优先搜索算法[16]和拓扑排序算法[16]，复杂度均为 $O(|V|+|E|)$。步骤 4 中链式因子的连接计算的复杂度为 $O(|V|)$。这样整个算法的时间复杂度为 $\max(O(|V|+|E|),O(L))$。

4.2　算法的终止性证明

由算法描述可知，只有在步骤 1、步骤 2 和步骤 4 中存在循环。任给一个 Soa $A=(V,E)$，A 中的节点个数以及非平凡强连通分量的个数是有限的，所以步骤 1 和步骤 2 中的循环只执行有限步。对于一个有向无环图，级数的范围是有界的，所以步骤 4 中的循环也只执行有限步。综上所述，该算法在执行有限步之后是终止的。

4.3　算法的正确性证明

下面的定理 3 形式化地描述了该算法的正确性。

定理 3　给定一个句子集合 S 和对应的 Soa $A=(V,E)$，运用 GenEchare 算法可以求得对应的 Echare r，并且有 $S\subseteq L(A)\subseteq L(r)$。

证明：由算法描述可知，经过步骤 1 之后，每个节点的标记有两种形式：a 或 a^+，a 是终极符；经过步骤 2 之后，每个节点的标记有 3 种形式：a、a^+或$(c_1+\cdots+c_k)^+$，其中 $c_j(1\leqslant j\leqslant k)$为 a、$a?$、a^+或 $a^+?$，a 为终极符；经过步骤 4 之后，每个链式

因子的形式为$(b_1+\cdots+b_k)$、$(b_1+\cdots+b_k)?$、$(c_1+\cdots+c_k)^+$或$(c_1+\cdots+c_k)^+?$，其中$b_j(1\leqslant j\leqslant k)$为$a$或$a^+$，$c_j(1\leqslant j\leqslant k)$为$a$、$a?$、$a^+$或$a^+?$，$a$为终极符，显然该算法的输出符合Echare的定义。

给定一个句子集合S和对应的Soa $A=(V,E)$，运用GenEchare算法可以求得对应的Echare r。根据r可以构造出相应的Glushkov自动机，记为$M_r(Q,\Sigma,\delta,q_I,F)$。根据定理1可知，$L(M_r)=L(r)$。在$M_r$中添加一个*sink*节点作为唯一的接受状态，并在*sink*节点与F中每个节点之间添加一条边，将q_I看作M_r的*src*节点，那么添加*sink*节点之后的M_r可以转换为一个等价的Soa，记为$M_s=(V_s,E_s)$，显然有$L(M_s)=L(M_r)=L(r)$。经过上述处理之后可知M_s和A的节点集合是相同的，即$V_s=V$。

在$A=(V,E)$中有两类边，一类是属于非平凡强连通分量内部的边，记为E_c，另一类是不属于非平凡强连通分量的边，记为E_d，即$E=E_c+E_d$。

对于$A=(V,E)$中任意一个非平凡强连通分量C，设C的节点集合为$\{v_1,\cdots,v_m\}$，由GenEchare算法可知r的链式因子中必定包含$(v_1+\cdots+v_m)^+$或$(v_1+\cdots+v_m)^+?$。对于C中的任意两个节点v_i和$v_j(1\leqslant i\leqslant m,1\leqslant j\leqslant m)$有$v_j\in follow(r,v_i)$和$v_i\in follow(r,v_j)$成立，由Glushkov自动机的定义可知，在$M_r$中从$v_i$到$v_j$以及从$v_j$到$v_i$均有边存在，显然在$M_s$中也存在相同的边。那么在$M_s$中，集合$\{v_1,\cdots,v_m\}$中的任意一个节点到自身以及该集合内其他所有节点均有边存在，所以对于$A=(V,E)$中任意的边$e\in E_c\subseteq E$，都有$e\in E_s$。

对于$A=(V,E)$中任意的边$e\in E_d\subseteq E$，不妨设$e=(v_1,v_2)$，那么有$v_2\in follow(r,v_1)$成立。由Glushkov自动机的定义可知，在M_r中必然存在一条v_1到v_2的边，同样在M_s中也存在这条边，即$e\in E_s$。

综上所述，对于$A=(V,E)$中任意一条边$e\in E$，都有$e\in E_s$成立，同时又因$V=V_s$，所以有$L(A)\subseteq L(M_s)$成立，即$L(A)\subseteq L(r)$。结合定理2可知$S\subseteq L(A)\subseteq L(r)$。证毕。

4.4 算法效果实例分析

本文算法是在已有Chare算法[11,15]的基础上进行的扩展。下面我们通过一个简单的例子来对比本文算法和已有算法的结果。

例6 给定句子集合$\{aab,aaaab,cddd,cdd\}$，对应的Soa如图8所示：

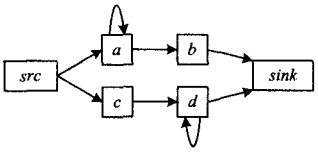

图8 Soa

CRX算法[11]的结果为：$a^+?c?b?d^+?$，Soa2Chare算法[15]的结果为$a^+?c?d^+?b?$，GenEchare算法的结果为：$(a^++c)(d^++b)$。CRX和Soa2Chare算法推断出的表达式都可以生成a、c或b、d同时出现的句子，而本文算法推断出的表达式不会生成这样的句子。从匹配句子集合的角度看，本文算法的结果更加合理准确。

5 相关工作

XML模式推断对XML的相关应用有非常重要的意义。XML模式推断的核心问题可以归约为：从一个句子集合中推断对应的确定型正则表达式。Garofalakis等人[17]于2003年提出了一个推断DTD的工具XTRACT。XTRACT首先为集合中的每个句子生成一个或多个对应的正则表达式，然后将具有公共子表达式的正则表达式进行合并，最后运用最小描述长度原则(Minimum Description Length)[18]选出最佳的正则表达式。文献[11]的研究表明，XTRACT存在两个不足之处：(1)容易生成冗长的、难以解释的正则表达式；(2)不能处理大数据集的情形。第二个不足是因为XTRACT运用最小描述长度原则的相关算法是NP难的。Min等人[19]采用XTRACT的思路提出了推断一类受限正则表达式的算法。根据文献[19]中的实验结果可知，该算法在准确度和效率上都有提升，但XTRACT的两个不足之处还是没有得到解决。

Bex等人[7,8]通过统计观察实际应用中的XML模式数据，提出了两类正则表达式：Sore和Chare。同时文献[11]也给出了这两类正则表达式的推断算法：RWR和CRX。其中RWR为推断Sore的算法，其核心思想是先由句子集合生成Soa，然后在Soa上运用重写规则推断出对应的Sore，如果某时刻所有的重写规则都不可用，那么对Soa运用对应的修复规则。限于篇幅，具体规则不在这里赘述。RWR算法的复杂度为$O(n^5)$，其中n为字母表的大小。CRX是推断Chare的算法，该算法首先从句子集合中计算链式因子，然后对链式因子进行排序，最后推断出对应的Chare。CRX算法的复杂度为$O(m+n^3)$，其中m为句子集合的大小，n为字母表的大小。通过实验[11]对比，RWR算法和CRX算法的效果要明显优于XTRACT，而且也不存在XTRACT的两个不足。

Freydenberger等人[15]针对Sore和Chare给出了新的算法：Soa2Sore和Soa2Chare。这两个算法的思想都是先由句子集合生成Soa，然后在Soa上进行计算并推断出对应的表达式。其中Soa2Sore算法的复杂度为$O(nm)$，Soa2Chare算法的复杂度为$O(m)$，n为Soa中节点的数目，m为Soa中边的数目。而且这两个算法推断出的正则表达式都是最小的：即给定句子集合，生成的表达式所表示的语言是该句子集合的最小超级。例如，给定一个句子集合S，运用Soa2Sore算法生成了Sore r，那么不存在另外一个Sore δ使得$S\subseteq L(\delta)\subset L(r)$成立。

本文针对实际数据对Chare进行扩展，提出了Echare并给出了推断算法。4.4节通过一个实例对比本文算法与已有Chare算法的推断效果。从匹配句子集合的角度看，本文算法的推断结果更加合理准确。

结束语 XML模式推断的核心问题可以归约为确定型正则表达式的推断问题。本文提出了一类在XML模式中大量出现的受限正则表达式Echare，并给出了Echare的推断算法。该算法首先合并Soa的非平凡强连通分量，将Soa转化成一个有向无环图；然后计算该有向无环图中每个节点对应的级数，按照级数顺序计算相应的链式因子；最后将所有的链式因子按照级数顺序进行连接运算生成Echare。该算法的时间复杂度为$\max(O(|V|+|E|),O(L))$。此外本文证明了该算法的正确性，表明该算法是可行的并且可以应用到XML模式推断问题中。下一步的工作是对该算法进行相关的实验。

参 考 文 献

[1] Abiteboul S, Buneman P, Suciu D. Data on the Web: From Relations to Semistructured Data and XML [M]. Burlington, USA: Morgan Kaufmann, 1999

[2] Bray T, Paoli J, Sperberg-McQueen C M, et al. Extensible Markup Language (XML) 1.0 (Fifth Edition) [EB/OL]. http://www.w3.org/TR/2008/REC-xml-20081126/, 2008

[3] Gao Shu-di, Sperberg-McQueen C M, Thompson H S. XML Schema Definition Language (XSD) 1.1 Part 1: Structures [EB/OL]. http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/, 2012

[4] Benedikt M, Fan Wen-fei, Geerts F. XPath satisfiability in the presence of DTDs [J]. Journal of the ACM, 2008, 55(2): 1-79

[5] Rahm E, Bernstein P A. A survey of approaches to automatic schema matching [J]. The VLDB Journal, 2001, 10(4): 334-350

[6] Barbosa D, Mignet L, Veltri P. Studying the XML Web: gathering statistics from an XML sample [J]. World Wide Web: Internet and Web Information Systems, 2005, 8(4): 413-438

[7] Bex G J, Neven F, Van den Bussche J. DTDs versus XML Schema: a practical study [C]// WebDB. 2004: 79-84

[8] Martens W, Neven F, Schwentick T, et al. Expressiveness and Complexity of XML Schema [J]. ACM Transactions on Database Systems, 2006, 31(3): 770-813

[9] Sahuguet A. Everything you ever wanted to know about DTDs, but were afraid to ask [C]// WebDB. 2000: 171-183

[10] Bex G J, Gelade W, Neven F, et al. Learning deterministic regular expressions for the inference of schemas from XML data [J]. ACM Transactions on The Web, 2010, 4(4): 1-32

[11] Bex G J, Neven F, Schwentick T, et al. Inference of concise regular expresssions and DTDs [J]. ACM Transactions on Database Systems, 2010, 35(2): 1-47

[12] Chen Hai-ming, Lu Ping. Assisting the design of XML Schema: diagnosing nondeterministic content models [C] // APWeb. 2011: 301-312

[13] Brüggemann-Klein A, Wood D. One-unambiguous regular language [J]. Information and Computation, 1998, 142(2): 182-206

[14] García P J, Vidal E. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(9): 920-925

[15] Freydenberger D D, Kötzing T. Fast learning of restricted regular expressions and DTDs [C]// ICDT. 2013: 45-56

[16] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms [M]. Cambridge, USA: The MIT Press, 2001

[17] Garofalakis M, Gionis A, Rastogi R. XTRACT: Learning document type descriptors from XML document collections [J]. Data Mining and Knowledge Discovery, 2003, 7(1): 23-56

[18] Rissanen J. Modeling by shortest data description [J]. Automatica, 1978, 14(5): 465-471

[19] Min J-K, Ahn J-Y, Chung C-W. Efficient extraction of schemas for XML documents [J]. Information Processing Letters, 2003, 85(1): 7-12

(上接第 158 页)

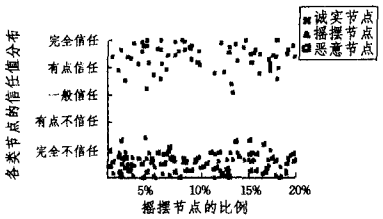

图 4 5000 次交易后 3 类节点的信任等级分布

结束语 本文在基于模糊逻辑的信任评判模型的基础上增加了对交易记录的聚合预测算法。算法基于多因素模糊逻辑推理规则对每次交易评价进行模糊评判，通过马尔科夫链记录用户的历史交易，使用转移分布和平稳分布来表示用户的信任预测矩阵。实验表明该算法具有较高的信任预测准确性，同时考虑交易的时间衰减和惩罚激励机制动态的修改转移矩阵，使得该模型对摇摆节点具有更好的抑制作用。

参 考 文 献

[1] 李勇军，代亚非. 对等网络信任机制研究[J]. 计算机学报，2010，33(3)：390-405

[2] Sepandar D K, Mario T S, Hector G M. The EigenTrust algorithm for reputation management in P2P networks[C]// Proceedings of the 12th Intertational Conference on World Wide Web. Budapest Hungary, 2003: 640-651

[3] Wang Yao, Julita V. Bayesian Network-Based Trust Model [C]// IEEE/WIC Int. Conf. on Web Intelligence. 2003: 2-7

[4] Song S, Hwang K, Zhou R, et al. Trusted P2P transactions with *fuzzy reputation aggregation* [J]. *Internet Computing*, IEEE, 2005, 9(6): 24-34

[5] 唐文，陈钟. 基于模糊集合理论的主观信任管理模型研究[J]. 软件学报，2003，14(8)：1401-1408

[6] Chen H W, Ye Z W. *Reaearch of P2P trust based on fuzzy decision-making*[C] // Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference. 2008: 793-796

[7] 张书钦，杨永田. 对等网络中基于模糊集的信任和声望模型[J]. 哈尔滨工程大学学报，2005，26(6)：764-766

[8] 王旭方，吴昊. 一种基于模糊理论的 P2P 电子商务信任模型[J]. 微计算机信息，2012(1)：122-123

[9] Hussain, Khadeer F, Chang E J, et al. Markov model for modeling and Managing dynamic trust 3rd IEEE Int. Conf. on Industrial Informatics. 2005: 725-733

[10] 叶尔骅，张德平. 概率论与随机过程[M]. 北京：科学出版社，2005：209-212