基于语义的中文 Deep Web 查询接口集成*)

洪 辉[1,2] 李石君[1,2] 余 伟[1] 田建伟[1]

(武汉大学计算机学院 武汉 430072)[1] (中国科学院计算机科学重点实验室 北京 100080)[2]

摘 要 现在网上信息正越来越被在线数据库深化,而传统的搜索引擎对此类信息源却没有很好的获取办法,加剧了人们想得到有用知识而搜索结果并不理想的形势。本文针对这种情况,简要论述了中文 Deep Web 研究工作的必要性及其发展前景,探讨了中文 Deep Web 技术的关键问题,并基于中文语义,提出了中文 Deep Web 中的查询接口集成方案。实验表明该方法能使得接口之间属性匹配的正确率达到 98%以上。

关键词 深网,接口集成,模式匹配

Semantic-based Chinese Deep Web Interface Integration

HONG Hui[1,2] LI Shi-Jun[1,2] YU Wei[1] TIAN Jian-Wei[1]

(School of Computer, Wuhan University, Wuhan 430072)[1]

(Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)[2]

Abstract Nowadays the Web is being increasingly deepened by a large group of online database, while traditional search engines are not able to make good use of such kind of information, which deteriorates the situation that people want to get more and more valuable gold of knowledge but unfortunately with bad search results. In this paper, we firstly emphasize the research in Chinese Deep Web as well as discuss its key issues aiming at the problem raised above, and refer to interface integration scheme based on Chinese semantic information, with a series of experiment results in the end that show good performance of our scheme and reach a as high accuracy as 98%.

Keywords Deep Web, Interface integration, Schema matching

1 引言

深层 Web 即 Deep Web 信息总量为浅层 Web 的 450～550 倍[1],通过传统搜索引擎所能获取的信息总量约为总资源的 0.2%,设法利用 Deep Web(以下简称深网)成为当前研究的热点。另外,中国互联网络信息中心[13]显示中文在线数据库数量在 2005 年达到 30 余万,截至 2010 年其数量可能达到 60 万以上;中国人口众多,潜在信息需求很大,中文深网前景乐观;而现在深网研究范围一般限定在英文数据库上,因此对中文深网技术的研究应当引起关注。

如果能为学生张三提供一个能同时查询多个书店数据库的统一查询接口,将能快速准确地返回所需书目信息,张三就不用分别查看当当网和卓越网等书店来确定最合适的交易。实现该目标可以分为两个步骤:首先将同领域的查询表单进行属性匹配,然后选取最具有表现力、最容易为用户接受的属性组成统一接口,本文主要讨论前者的解决办法。

本文所做贡献如下:

ⅰ)目前深网研究主要关注英文网站及其在线数据库,没涉及中文领域,本文较先地讨论中文深网技术,并给出了表单及其集成的形式化定义;

ⅱ)指出接口集成为中文深网关键问题,并基于中文语义,提出了中文深网中的查询接口集成方案;

ⅲ)实验结果表明该方法能使得接口之间属性匹配的正确率达到 98%以上,为实施一整套中文深网系统解决了最大难题。

2 相关工作

深网可分为 6 个方面问题:ⅰ)发现数据源;ⅱ)数据源聚类;ⅲ)将同一领域查询接口集成为统一查询接口;ⅳ)将用户提交给统一接口的查询映射到与查询相关的在线数据库;ⅴ)提取在线数据库返回的结果;ⅵ)集成与查询相关的在线数据库返回的结果,并以统一的形式展现给用户。其中ⅲ与语言的相关性最大,不同的语言所具有的特点,必将导致接口集成[12]方法各异。

B. He[11]等人从同一领域的词汇集合中发现隐藏的模式模型,然后利用该模型来帮助寻找同义词,进而实现接口集成。HeCH[8]先将属性匹配看作相关性挖掘,然后把挖掘得到的相关性知识应用于辅助属性匹配。接口模式在一定程度上反映数据库内部模式,WangWL[7]向数据库递交查询请求,通过分析返回结果,来弄清数据库的真实结构,来辅助属性匹配工作。WISE-Integrator[10] positive 匹配基于聚簇方法,属于准确匹配,主要任务是将匹配的属性聚到同一簇中,并选出每一簇中的代表属性,predictive 匹配基于权重,属于近似匹配,主要任务是构造最终的集成接口。W. Wu[6,4]将属性匹配问题转化成一个以树为背景的优化问题,比如树 A 和 B 合并生成树 C,不仅要包括 A 和 B 中所有叶子节点,而且要在最大程度上保留 A 和 B 中叶子节点和内部节点的结构特征。大规模属性匹配[6,8,10,11]并不需要所有模式信息,HeC[3]采

*)湖北省自然科学基金(No. 2005ABA238)资助。洪 辉 硕士研究生,主要研究方向:Web 搜索与挖掘;李石君 教授,博士生导师,主要研究方向:Web 搜索与挖掘。

用有效的采样技术和投票机制，找到了尽可能正确的子集并得到了尽可能准确的匹配结果。MKB [5] 充分利用了匹配历史，训练精准分类器，进行后续匹配：如果待匹配接口中的属性分别与训练集中某已知接口中属性相似，则认为两接口也相似。WebIQ[2] 利用现有庞大 Web 资料库（浅层 Web 和能够探明的深层 Web），根据标签或者实例同时出现得越频繁，认为它们相似的可能性越大这个特点，有效地解决了属性实例不足的问题。

上述方法很多都基于对英语语法的分析，而且准确度不高，很难有效地解决中文属性匹配问题。英语与汉语最大区别在于，汉语是一种表意型语言，十分依赖特定的语境，而英语则相对准确些。如果照搬上述方法简单地从字面分析会引起较大的歧义，加入语义信息能在很大程度上提高中文属性匹配准确度。本文基于两点假设：1. 相关数据源已经聚类；2. 相关表单及其属性已经被抽取，借助知网技术[2]，结合中文语义，提出了中文深网接口集成解决方案。

3 形式化描述及集成方案

3.1 定义表单

定义 1 表单用下面一个五元组表示：

$FORM=\{ID, REQUEST, METHOD, N, NAME\_VALUE\}$，可用 $FORM$ A 来定义表单 A。其中：

ID —表单标识符，由系统自动赋予并用来唯一表示表单；

$REQUEST$—表示提交表单对应的 URL 请求，针对该 URL 请求，后台服务器有与之对应的响应程序；

$METHOD$ —提交表单数据所采用的方式 Method，值域为{POST，GET}，POST 表示隐式提交表单数据，而 GET 则表示显式提交；

N —正整数，表示提交表单中属性的个数；

$NAME\_VALUE$—属性名/值对集合，用来表示表单中属性的详细信息，其个数为 N。

3.2 定义表单集成

为了方便论述，本节只讨论两个表单集成的简单情况。

定义 2 表单 A 和 B 集成后，生成表单 C，可以表示为：

$\{ID^A, REQUEST^A, METHOD^A, N^A, NAME\_VALUE^A\}$

INT

$\{ID^B, REQUEST^B, METHOD^B, N^B, NAME\_VALUE^B\}$

$=\{ID^C, REQUEST^C, METHOD^C, N^C, NAME\_VALUE^C\}$。

简写为：A INT B=C（其中，符号“INT”表示集成）。

定义 2 给出了表单集成的形式描述，规则 1 将阐明集成符号“INT”真正的涵义，即集成规则。

规则 1 “INT”集成操作包括以下 5 个方面：

ⅰ）$ID^A\ INT\ ID^B=ID^C$；

ID^C 由系统自动分配，比如采用 Oracle 中的 Sequence 机制。

ⅱ）$REQUEST^A\ INT\ REQUEST^B=REQUEST^C$；

$REQUEST^C=REQUEST^A \cup REQUEST^B=\{URL|URL \in REQUEST^A$ 或 $URL \in REQUEST^B\}$。表单 C 的提交请求的 URL 是 A 和 B 的简单并集，但记录下了表单和 URL 之间的映射关系，即 URL_A 对应 A，URL_B 对应 B，这是为了保证查询派发的正确性。

ⅲ）$METHOD^A\ INT\ METHOD^B=METHOD^C$；

类似 ⅱ。

ⅳ）$N^A\ INT\ N^B=N^C$；

此处 N^C 并不是 N^A 和 N^B 简单的加法运算，根据定义 1 中的“N 表示提交表单中数据的个数，也即属性的个数”，得 $N^C=|NAME\_VALUE^C|$。

ⅴ）$NAME\_VALUE^A\ INT\ NAME\_VALUE^B=NAME\_VALUE^C$。

$NAME\_VALUE^C=$ Integration（$NAME\_VALUE^A$，$NAME\_VALUE^B$）。

3.3 Integration 方法

定义 3 对于给定的一个阈值 ξ，如果 Sim(Attribute A，Attribute B)$\geq\xi$（Sim 为相似度方法，其计算方法参见第 4 节），则认为属性 A 与 B 相匹配，记为 AttributeMatch(A，B)=1；否则不相匹配，AttributeMatch(A，B)=0。

以下介绍如何找到 A 和 B 中相互对应的属性。假设 A 和 B 中的属性采用链表组织，表头分别为 HeadA，HeadB，且 A 表长度较小。寻找匹配属性算法如下：

```
/*MAX存储相似度最大值;
r,s分别指向A,B表中当前比较节点;
Rmax,Smax分别A,B表中匹配节点;
Delete(node)能将节点node从链表中删除;*/
    p:=HeadA;
    while p→next<>NULL do
    begin
      r:=p→next; MAX:=0;q:=HeadB;
        while q→next<>NULL do
          begin
          s:=q→next;
        if MAX<Sim(r,s)
            then
              begin
                MAX:=Sim(r,s);Rmax:=r;Smax:=s;
              end
          q:=s→next;
        end
end
p:=r→next;Delete(Rmax);Delete(Smax);
```

找到 A 和 B 中匹配属性后，假设相匹配属性为各自的前 m 个：A 的属性 A_i 与 B 的属性 B_i 相匹配（$1\leq i\leq m$）。基于这个假设，下面给出 Integration 方法。

function Integration(NAME_VALUE^A，NAME_VALUE^B)

```
var i,j,k: integer
begin
  //将相匹配属性组装入表单C中
  for i:=0 to m-1 do NAME_VALUE^C.add(A_i);
  //将A余下属性组装入表单C中
  for j:=m to N^A-1 do NAME_VALUE^C.add(A_j);
  // B余下属性组装入表单C中
  for k:=m to N^B-1 do NAME_VALUE^C.add(B_k);
end
```

推论 1 $N^C=|NAME\_VALUE^C|=N^A+N^B-m$。

4 相似度计算

定义 4（属性相似度，Attribute Similarity） 对于属性 Att^A，Att^B，其中 Att^A 有 m 个词语 $W_i^A(1\leq i\leq m)$，Att^B 有 n 个词语 $W_j^B(1\leq j\leq n)$，则 Att^A 与 Att^B 之间的相似度为两属性间词语相似度的最大值。即：

$$\mathrm{Sim}(Att^A, Att^B)=\max Sim(W_i^A, W_j^B)(1\leq i\leq m, 1\leq j\leq n)$$

定义 5（词语相似度，Word Similarity） 对于词语 W^A，W^B，其中 W^A 有 m 个义项 $W_i^A(1\leq i\leq m)$，W^B 有 n 个义项 $W_j^B(1\leq j\leq n)$，则 W^A 与 W^B 之间的相似度为两词间义项相

似度的最大值。即：

$Sim(W^A, W^A) = \max \mathrm{Sim}(W_i^A, W_j^B)(1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n)$。

定义 6(义项相似度，Concept Similarity)　义项 C^A，C^B 之间的相似度为：

$$Sim(C^A, C^B) = \sum_{i=1}^{4} \beta_i \prod_{j=1}^{i} Sim_j(C^A, C^B)$$

其中：$\beta_i (1 \leqslant i \leqslant 4)$ —可调节参数，且 $\sum_{i=1}^{4} \beta_i = 1$，$\beta_1 \leqslant \beta_2 \leqslant \beta_3 \leqslant \beta_4$；$Sim_i(C^A, C^B)$ 为义原相似度。

定义 7(义原相似度，Primitive Similarity)　义原 P^A，P^B 之间的相似度随两者距离增大而减小。即：

$Sim(P^A, P^B) = \varphi/(\varphi + Dis(P^A, P^B))$。

其中：φ—可调节参数。

$Dis(P^A, P^B)$ —义原 P^A，P^B 在知网[14]义原层次树中的距离。

假定 $\xi = 0.5$；$\varphi = 1.6$；$\beta_1 = 0.5$；$\beta_2 = 0.2$；$\beta_3 = 0.17$；$\beta_4 = 0.13$；试计算属性 A(书名)和属性 B(商品名称)的相似度。

1. 计算所有义项相似度，并求得词语相似度。

属性 A 含两个词语——“书”，“名”；

其中词语“书”有 6 个义项：

1〉N attribute|属性，form|形状，&character|文字；

2〉N attribute|属性，style|形状，&character|文字；

3〉N document|文书；

4〉N letter|信件；

5〉N publications|书刊；

6〉V write|写。

词语“名”有 2 个义项：

1〉ADJ aValue|属性值，reputation|名声，glorious|荣，desired|良；

2〉N attribute|属性，name|姓名，&entity|实体。

属性 B 含两个词语——“商品”，“名称”。

其中词语“商品”有 1 个义项：N artifact|人工物，commercial|商，generic|统称；词语“名称”有 1 个义项：N attribute|属性，name|姓名，&entity|实体。

根据定义 6 求得词语“书”和“商品”的所有义项相似度(共 5 * 1=5 个)，然后按定义 5 其中最大值 0.266670 为词语相似度。同理求得“书”和“名称”，“名”和“商品”，“名”和“名称”的词语相似度(见表 1)。

表 1　属性相似度计算示例

词语 A	词语 B	相似度
书	商品	0.266670
书	名称	0.614719
名	商品	0.045908
名	名称	1.000000

2. 求属性相似度

由定义 8 可知，属性相似度：$Sim(Att^A, Att^B) = \max\{0.266670, 0.614719, 0.045908, 1.000000\} = 1.000000 > \xi = 0.5$，故，可以认为属性 A(书名)和属性 B(商品名称) 相匹配。

5　实验

为了保证结果的一般性，该实验对象为随机抽取的 250 个查询表单，均匀覆盖书店，笔记本，手机，房地产，航班等 5 个领域，其统计信息如表 2 所示。

表 2　样本信息

类别	表单数量	表单中属性平均数量
书店	50	8.4
笔记本	50	6.6
手机	50	18.6
房地产	50	6.8
航班	50	5.2
总计	250	9.12

在本文实验中，前面提到的参数取值为：

$\varphi = 1.6$；$\beta_1 = 0.5$；$\beta_2 = 0.2$；$\beta_3 = 0.17$；$\beta_4 = 0.13$ 时，表单属性相似度计算参照上例进行。图 1 为书，笔记本，手机，房地产，航班等领域的属性匹配准确率。

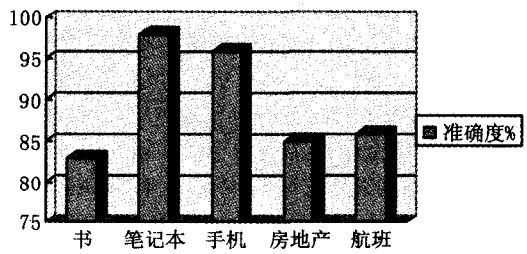

图 1　属性匹配准确度

上述 5 种领域属性匹配准确率分别为 83%，98%，96%，85%，86%。其中，领域“书”的匹配准确率较低，这是因为该领域同一查询表单中，包括“书名”，“从书名”，“原从书名”，作者，译者等不同属性，但根据本文方法计算的相似度可以认为它们相似，这样待匹配属性中可能出现一配多，多配多的情况，所以很可能出现匹配差错。航班含有起始站，终点站等易判为相似的属性，结果较差。相比而言，房地产属性相关性也较低，匹配结果较理想，笔记本，手机表单是按部件设计的，基本上不存在前述“自包含”异常情况，因而匹配结果较为理想。

结束语　本文较先地探讨了中文深网技术，宏观上提出了整体解决方案，并分析指出其关键技术在于解决接口集成问题；文中形式化地给出了表单及其集成概念的准确定义；针对后者，本文给出了一种基于中文语义信息的接口集成方法；实验结果表明该方法能使得接口之间属性匹配的正确率高达 98%，效果比较理想。

本文是对中文深网技术的一次初探，主要研究其接口属性匹配问题。下一步不仅应该深入研究接口集成效果的改进，还当研究如何准确理解用户的请求并找到相关数据源，我们还会致力于做出全自动的属性匹配原型系统。另外针对中文深网一整套解决方案实现原型系统也是未来工作的重点。

参 考 文 献

1　Chang K C C, He B, Li C, Patel M, Zhang Z. Structured Databases on the Web: Observations and Implications. SIGMOD Record, 2004,33(3):61～70

2　Wu W, Doan A, Yu C T. WebIQ: Learning from the Web to Match Deep-Web Query Interfaces. In ICDE, 2006. 44

3　He B, Chang K C C. Making holistic schema matching robust: an ensemble approach. In KDD, 2005, 429～438

4　Wu W, Doan A, Yu C T. Merging Interface Schemas on the Deep Web via Clustering Aggregation. In ICDM, 2005, 801～804

5　Madhavan J, Bernstein P A, Doan A. Corpus-based Schema Matching. In ICDE, 2005. 57～68

6　Wu W, Yu C T, Doan A. An Interactive Clustering-based Approach to Integrating Source Query interfaces on the Deep Web. In: SIGMOD Conference, 2004. 95～106

7 Wang J, Wen J R, Lochovsky F H. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In VLDB, 2004. 408～419
8 He B, Chang K C C, Han J. Discovering complex matchings across web query interfaces: a correlation mining approach. In KDD, 2004. 148～157
9 He B. A Holistic Paradigm for Large Scale Schema Matching. SIGMOD Record, 2004, 33(4):20～25
10 He H, Meng W, Yu C T, Wu Z. WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce. In VLDB, 2003. 357～368
11 He B, Chang K C C. Statistical Schema Matching across Web Query Interfaces. In:SIGMOD Conference, 2003. 217～228
12 Rahm E, Bernstein P A. A survey of approaches to automatic schema matching. VLDB J., 2001,10(4):334～350
13 http://www.cnnic.net.cn/download/2005/2005041401.pdf
14 Liu Q, Li S J. Word Similarity Computing Based on How-net. In: The 6th Chinese Lexical Semantics Workshop,2002

(上接第 60 页)

1. 计算树模式 q 的所有自同态及其目标结点集；

2. 计算包含 1 中每个结点集的最小子查询的结点集，同时找到其中规模最小的子查询的结点集；

3. 根据 2 的结点集构造树模式，即是最终输出。

在第 1 步中，我们记树模式 q 的自同态 h_a 的目标结点集为 S_i，$S_i = NODES(h_a(q))$，记树模式 q 的所有自同态目标结点集的集合为 $tarset(q)$，$tarset(q)=\{S_i\}$。在图 3 的例子中，树模式 q 存在 2 个自同态，一个是图中所示的自同态，记作 h_{e1}，另一个是每个结点都映射到其自身的自同态，记作 h_{e2}，它们的目标结点集分别是 $S_1=\{$node-1, node-4, node-5, node-6$\}$, $S_2=\{$node-1, node-2, node-3, node-4, node-5, node-6$\}$，所以 $tarset(q)=\{S_1, S_2\}$。

在第 2 步中，我们计算包含 S_i 的最小子查询的结点集，记为 $minisub(S_i)$，即 $minisub(S_i)=\{x|x\in S_i\}\cup\{y|(y,x)\in EDGES^+(q)\}$。同时找到所有 $minisub(S_i)$ 中规模最小的结点集，记为 $miniSet$，即 $miniSet=\min\{minisub(S_i)\}$。在图 3 的例子中，$minisub(S_1)=S_1$，$minisub(S_2)=S_2$，$miniSet=\min\{S_1, S_2\}=S_1=\{$node-1, node-4, node-5, node-6$\}$。

在第 3 步中，我们通过如下方法构造规模最小的子查询：按深度优先顺序遍历原树模式查询 q，如果当前结点不在 $miniSet$ 中，那么删除以该结点为根的子树。最终获得原树模式查询 q 的最小等价子查询。在图 3 的例子中，q_0 是 q 的最小等价子查询。

```
算法：树模式最小化算法
输入：树模式 q
输出：q 的最小等价树模式
(1)   计算 q 的所有自同态{h_ei};
(2)   计算每个自同态的目标结点集 tarset(q) = {NODES(h_ei(q))};
(3)   int miniSize = |q|;
(4)   Set miniSet = ∅;
(5)   for each S ∈ tarset(q){
(6)     minisub(S) = ∅;
(7)     for each k ∈ S{
(8)       if ( k ∉ minisub(S) )
(9)         minisub(S) = minisub(S) ∪ {k};
(10)      for each pk ∈ {pk | (pk, k) ∈ EDGES+(q)}
(11)        if (pk ∉ minisub(S))
(12)          minisub(S) = minisub(S) ∪ {pk};
(13)      if (|minisub(S)| < miniSize){
(14)        miniSize = |minisub(S)|;
(15)        miniSet = minisub(S);}}}
(16)  if (miniSize = |q|)
(17)    return q;
(18)  else{
(19)    for each x ∈ NODES(q) //深度优先顺序
(20)    if ( x ∉ miniSet)
(21)      删除以 x 为根的子树;
(22)    return q;}
```

图 4 XPath 查询最小化算法

算法如图 4 所示，第一步(行 1～2)的时间杂度主要由计算同态映射的复杂度($O(|q|^2)$)[11]决定，除此，计算目标结点集发生在已知同态映射之后，因此不增加新的时间复杂度。第二步内层循环(行 7～15)最坏情况是遍历 q 的所有结点，时间复杂度为 $|q|$；外层循环(行 5～15)的时间复杂度是 $|tarset(q)|$，这通常是一个比 $|q|$ 小的数。因此第二步的时间复杂度是 $|q|^2$。第三步(行 16～22)遍历 q 的所有结点，故时间复杂度是 $|q|$。综上，算法的时间复杂度为 $O(2|q|^2+|q|)$。

结论和展望 本文研究树模式查询最小化问题，提出了一个可扩展的最小化的框架，并给出一个有效的算法。对于不同类型的查询片段，只要有相应的计算同态映射的算法，都可以嵌入本框架，形成新的算法。本文提出的算法是高效率的，但由于存在同态映射是查询包含的充分但非必要条件，因此对于某些查询片段，该算法不能获得最小查询。考虑到查询最小化问题是 coNP 完备问题，故寻求完备性和高效性的折中是不可避免的，即使对于这些特殊的片段，我们的算法也是实用的。

参 考 文 献

1 Bray T, Paoli J, Sperberg-McQueen C M, Maler E, Microsystems S, Yergeau F, Cowan J. Extensible Markup Language (XML) 1. 1. http://www.w3.org/TR/2004/REC-xml11-20040204/
2 Berglund A, Boag S, Chamberlin D, Fernandez M F, kay M, Robie J, Simeon J. XML Path Language(XPath) 2. 0. http://www.w3.org/TR/xpath20/
3 Boag S, Chamberlin D, Fernández M F, Florescu D, Robie J, Siméon J. XQuery 1. 0: An XML Query Language. http://www.w3.org/TR/xquery
4 Amer-Yahia S, Cho S R, Lakshmanan L V S, Srivastava D. Minimization of Tree Pattern Queries. In: Proc. of the 2001 ACM SIGMOD Conf on Management of Data. Santa Barbara, California, USA, May 2001. 497～508
5 Miklau G, Suciu D. Containment and equivalence for an XPath fragment. In: Proc. 21th Symposium on Principles of Database Systems (PODS 2002), 2002. 65～76
6 Milo T, Suciu D. Index structures for path expressions. In: Proc. Database Theory - ICDT '99, 7th International Conference 1999. 277～295
7 Amer-Yahia S, Cho S R, Lakshmanan L V S, Srivastava D. Tree pattern query minimization. The VLDB Journal, 2002, 11(4): 315～331
8 Schwentick T. XPath query containment. In: ACM SIGMOD Database principles Column, ACM Press, 2004, 33(1):101～109
9 Miklau G, Suciu D. Containment and equivalence for a fragment of XPath. Journal of the ACM, 2004, 51(1): 2～45
10 Chandra A K, Merlin P M. Optimal Implementation of Conjunctive Queries in Relational Databases. In: Proc. of the 9th ACM Syrup, Boulder, Colorado, United States: Theory of Computing, 1977. 77～90
11 Ramanan P. Efficient Algorithms for Minimizing Tree Pattern Queries. In: Proc. of the 2002 ACM SIGMOD Conf. on Management of Data, Madison, Wisconsin, June 2002. 299～309
12 Flesca S, Furfaro F, Masciari E. On the minimization of XPath queries, In: Proc. of the 29th Int Conf. on Very Large Data Base, Berlin, Germany, September, 2003
13 Liao Y, Feng J, Zhang Y, Zhou L. Hidden Conditioned Homomorphism for XPath Fragment Containment. In: Proc. of the 11th International Conference on Database Systems for Advanced Applications (DASFAA'2006), Springer-Verlag, April 2006. 454～467