基于 XML Store 的程序代码查询匹配算法①

肖　奔　邓爱萍　(湖南人文科技学院计算机科学技术系 湖南 娄底 417000)

摘　要: 在研究程序代码相似性度量方法的基础上，提出一种基于 XML store 的程序代码查询匹配算法。由于 XML store 以树型结构保存 XML 文件，算法将通过查询 XML store 中 DVM 树来对判断程序之间是否具有相同结构的子树，进行相似度度量。最后,通过在原型系统上进行的一系列实验，进一步证明了提出的算法在程序代码相似度度量实际应用中的可行性和有效性。

关键词: 匹配度; XML 存储库; DVM 树; 匹配模型

Matching Algorithm for XML Store-Based Program Query

XIAO Ben, DENG Ai-Ping (Department of Computer Science, Hunan Institute of Humanities, Science and Technology, Loudi 417000, China)

Abstract: On the base of studying the program code similarity measure method, proposed matching algorithm for XMLstore-based programs query. Because XML store uses tree structure to save the XML document, the algorithm will query the DVM tree in XML store to determine whether have the same structure between subtrees, for similarity measure. Finally, a series of experiments that carried out on the prototype system proved that the algorithm is feasible and effective.

Keywords: matching degree; XML store; DVM tree; matching mode

程序代码相似性度量旨在判断一个程序的源代码是否是从另外一个程序的源代码拷贝而来。目前的程序代码相似性度量方法可分为两大类：属性计数法和结构度量法。属性计数法对程序的属性，如关键字数、操作符数、循环数等进行处理，不考虑程序的内部结构。早期的程序剽窃检测系统一般采用此类方法。结构度量法则根据程序的结构度量两个程序之间的相似度，它需要对程序的内部结构或控制流进行分析。结构度量技术所依赖的程序特征及度量元素有很多,如 McCabe 圈复杂度、代码嵌套深度等[1-4]。

常用的程序代码剽窃检测工具通过从每个需判断的文件中选择大量指纹，对这些指纹与其他文件的指纹进行两两比较来计算程序代码相似度的方式进行工作。

本文提出以 XML 的树查询匹配模式来实现程序代码查询匹配。XML store 是一个存储 XML 文档的工具，它将 XML 文档以树形结构保存，并允许节点共享。通过使用 XML store 作为副本检测工具，不需对源程序文件提取指纹及对指纹的比较。

1　XML store

1.1 XML Store 的工作原理

XML store 是一种分布式的面向值的 XML 文档存储库，以 XML 格式存储从各个数据源抽取来的数据，形成单一的数据访问层面[5]。XML 是国际公认的数据交换标准，使用 XML store 在进行数据处理时无需进行格式转换，从而可以达到更高的数据处理效率。基本的 XML store 以下述步骤工作[6]：

(1) 从外部源读取 XML 数据

(2) 序列化展平文件

(3) 将数据解析成一种内部表示形式

(4) 反解析并保存每个节点到数据流和磁盘

① 基金项目:湖南省教育厅科学研究项目(08C458)
收稿日期:2010-03-05;收到修改稿时间:2010-04-09

1.2 XML 描述文档的树型表示

```
<DOC1>
  <n1>
    <n3>
      L1
    </n3>
  </n1>
  <n2>
    <n4>
      L2
    </n4>
    <n5>
      L3
      <n7> L4 </n7>
    </n5>
  </n2>
</DOC1>
```

```
<DOC2>
  <n2>
    <n4>
      L2
    </n4>
    <n5>
      L3
      <n7> L4 </n7>
    </n5>
  </n2>
  <n6>
    L5
  </n6>
</DOC2>
```

图 1 两个简单的 XML 文档

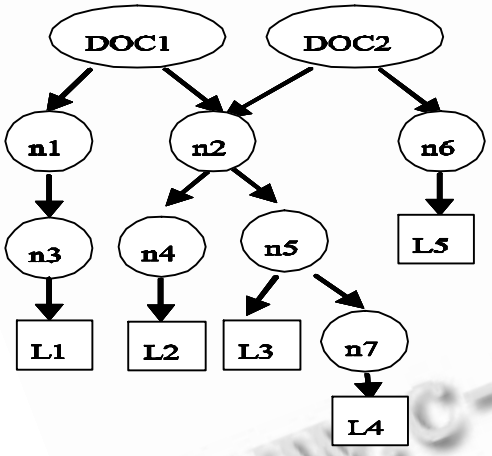


图 2 XML Store 共享数据

程序的 XML 描述文档可以映射为一棵无序树,采用 DVM(Document Value Mode)模型来描述。DVM 树是一个有向无环图，为 XML store 提供了一个用来表示 XML 文档的树型结构。DVM 树允许节点共享，使它非常适合副本检测中的应用。XML store 在保存两个或多个程序时，结构相同的部分被共享。例如：下图 1 所示为两篇 XML 文档,中有一部分结构是共享的，图 2 为 XML Store 中对这两篇文档的表示。

2 程序代码查询匹配算法

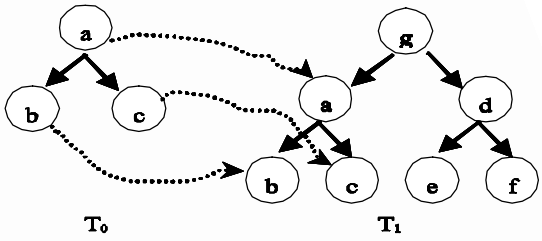
2.1 查询匹配模型

XML 的树查询匹配模型主要有嵌入匹配、树的包含匹配、包涵匹配，如图 3 所示[7-9]。

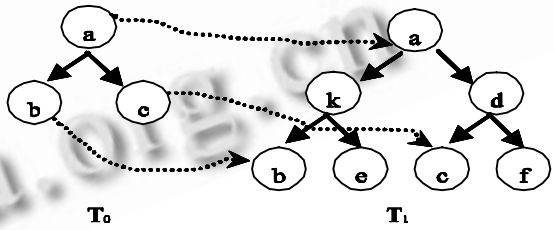
(1) 树的嵌入匹配：要求保持查询结点对的父子关系，同时不容许出现多余的结点元素。所以，嵌入匹配可以看作精确匹配，使用嵌入匹配时查准率较高，但相应的查全率就会很低。

(2) 树的包含匹配:要求保持结点对的祖先与后代关系，不用严格保持父子关系，匹配时允许包含树中出现了多余结点。查询包含匹配时查全率有所提高。

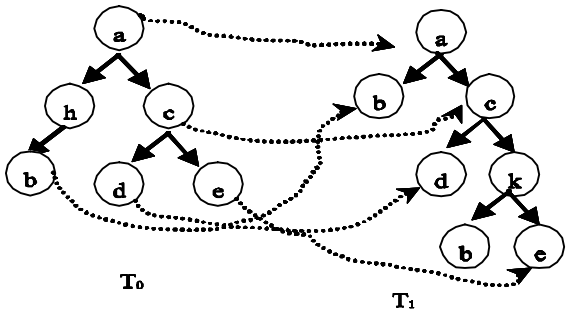
(3) 树的包涵匹配:既允许被查询树中出现无用结点，又允许被查询树缺少有用结点，要求匹配的结点对具有近似关系即可，这一改进提高了精确匹配的基础上模糊查询的能力，更进一步提高了查全率。



(a)树的嵌入匹配



(b)树的包含匹配



(c)树的包涵匹配

图 3 三种树匹配模型示意图

2.2 程序代码查询匹配算法描述

如前所述由 XML store 存储的树实际上是有向无环图，相同的部分只存储一次。如果某些节点有相同的子树，则共享部分可能是复制而来。程序代码查询匹配算法主要是通过对文档树的结点进行遍历，计算出两棵树的结构匹配度，再根据用户给出的最小结构匹配度来判断它们是否同源。首先导入两棵树，然后从树的最底层开始比较，计算出这一层节点的匹配度，如果其匹配度大于给定的最小结构相似度，则继续往上一层进行匹配。这样逐层往上，直到树的根节点。如果最终结构匹配度大于阈值则生成并输出它们的集成模式文件。匹配算法如下：

输入：查询树 T0，数据源树 T1，最小匹配度 MinDegree

输出：匹配度 degree

```
Matche(Tree T0, Tree T1, double MinDegree)
{ double Degree;
 // 层次遍历 T0 和 T1，并将扁历结果存放于链表中
  List0 = layerorder(T0);
  List1 = layerorder(T1);
  L0=list0;
  L1=list1;
  while (L0->next!=null){
    while (L1>next!=null){
        计算链表中树结点的匹配度 Degree;
        If(Degree>MinDegree) //继续匹配
          { L0=L0->next;L1=L1->next}
        else exit      //退出匹配
       }
   }
  }
```

2.3 匹配度计算

对于查询树 TO=(VO,EO,root(TO))和数据源树 T1=(V1,E1,root(T1))，匹配度计算见式(1)[9]。

$$Degree\,[T_0,T_1]=\frac{|V_1|}{|V_0|}\cdot\sum a_{uv}\frac{Min(l_{uv},L_{ue})}{Max(l_{ue},L_{uv})} \quad (1)$$

式中，u,v 为树 TO 与 T1 的结点,且结点 u、v 在树中连通；luv 为树 T1 中 u、v 结点之间的路径所包含的边数；Luv 为树 T0 中 u、v 结点之间的路径所包含的边数；auv 为在树 T0 中 u、v 结点之间的路径权值。

查询结果从两个方面考虑：$\frac{|V_1|}{|V_0|}$ 表示有效结点析出的多少，即查询出的有用信息量；$\sum a_{uv}\frac{Min(l_{uv},L_{ue})}{Max(l_{ue},L_{uv})}$ 表示有效结点间的结构关系，即查询结果树的深度和分支。

3 实验结果分析

为了测试提出的方法，实现了一个简单的测试平台对 XML 程序集进行复制检测，系统结构如图 4 所示。

测试数据集为一个简单的已知文件 file1.xml 以及以此为副本作出了一些相应修改后的文件，将生成的文件与原始文件进行匹配度测试，检测结果如表 1 所示。从表 1 可见，复制原始程序并对其进行变量重命名、代码段颠倒秩序、增加冗余语句、修改选择语句等修改后，都能够比较正确地得出结论。

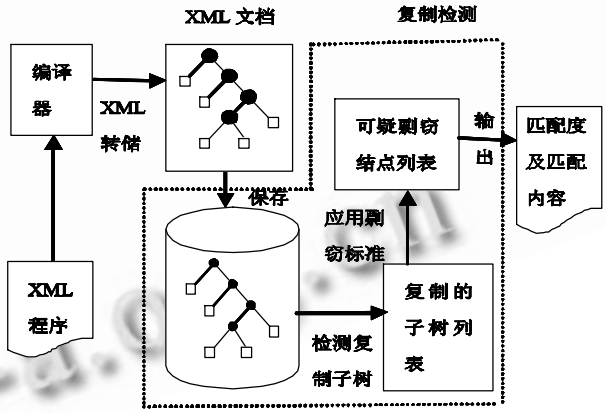


图 4　系统结构图

表 1　检测结果

修改对象	检测结论(%)
变量重命名	98
代码段颠倒秩序	98
冗余语句	89
选择语句结构	65
数据类型	22

4 结语

本文研究了 XML store 在程序代码相似性度量方

(下转第 256 页)

(上接第 228 页)

面的应用，由于 XML store 以树型结构保存 XML 文件，它将识别出 XML 程序文档相同的部分，并对这些共享的部分仅保存一次。使用该方法我们可通过程序的语法而不只是词法来检测程序代码抄袭的可能性。

参考文献

1 Granville A. Detecting Plagiarism in Java Code. Supervisor: Yorick Wilks, 2002.

2 Clough P. Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies. Research Memoranda CS200205, Department of Computer Science, University of Sheffield, 2000.

3 Prechelt L, Malpohl G, Philippsen M. Finding Plagiarisms Among a Set of Programs with JPlag. Journal of Universal Computer Science, 2000,8(11): 1016—1038.

4 Vamplew P, Dermoudy J. An anti-plagiarism editor for software development courses. Australian Computer Society, Inc. 2005,42:83—90.

5 陈和平,高丽,杨玲贤.基于面向值的映像方法在 XML 数据存储中的应用.武汉科技大学学报(自然科学版),2005,28(2):95—98.

6 Jesper Tejlgaard Pedersen and Kasper B.gebjerg Pedersen. Value-oriented XML Store. ITU and DTU, 2002.

7 Li Q，Moon B. Indexing and querying XML data for regular path expressions. Proeeedings of the 27th International Conference on Very Large Data Bases, Roma, Italy, 2001.

8 徐如志,钱乐秋,程建平,等.基于 XML 的软件构件查询匹配算法研究.软件学报, 2003,14(7):1195—1202.

9 范通让,王奕,赵永斌,等.匹配预处理对 XML 查询的优化.计算机工程与应用. 2009,45(19):125—127.