基于 JavaScript 切片的 AJAX 框架网络爬虫技术研究①

Web Crawler Technology of AJAX Frame Based on JavaScript Slicing

曾伟辉[1,2] 李 淼[1] (1.中国科学院合肥智能机械研究所 安徽 合肥 230031;
2.中国科学技术大学 信息科学技术学院自动化系 安徽 合肥 230027)

摘 要: 自 Jesse James Garrett 提出了 AJAX 概念以来，由于 AJAX 在提升用户交互体验的同时，又不需要在客户端安装插件。因此，一经提出就引起了互联网领域的广泛关注。但目前的网络爬虫技术在 AJAX 框架的 URL 解析过程中存在着不能够识别事件触发顺序等问题，导致大量数据不能被搜索引擎有效检索。本文针对此问题，通过研究基于对象的程序切片算法，以及脚本执行引擎与切片模块的互操作，最终解决 AJAX 框架中 URL 提取以及异步 JavaScript 网络爬虫系统的关键技术问题。

关键词: JavaScript 程序切片 网络爬虫 有限状态机 AJAX

1 引言

AJAX[1]是 2005 年 Jesse James Garrett 在一篇名为《Ajax: A New Approach to Web Applications》的文章中定义的一种客户端技术,它建立在 JavaScript 和 XML 基础上，真正实现了 RIA(Rich Internet Applications，丰富互联网应用程序)，克服了其它 RIA 技术需要在客户端安装插件才能够实现动态交互的弱点。同时，AJAX 极大地提高了用户的交互体验。因此，AJAX 一经提出，就被网站开发人员大规模采用，各种 AJAX 框架如 GWT，Atlas，Dojo 也应运而生。但 AJAX 站点中包含大量 JavaScript 代码，对于 RIA 中 JavaScript 脚本的处理，国内外的研究是采用标准浏览器 API 自动构建迷你浏览器替代 web 浏览器(如 IE)处理脚本执行代码[2]。中国科学院计算技术研究所在 Mozilla 开源的 JavaScript 引擎 SpiderMonkey 下，通过构造浏览器内置对象的方法提取动态网页的 URL[3]，但这些方法都只能够处理 JavaScript 脚本中 URL 字符串常量，而对于需要按照特定的顺序触发事件才能得出完整 URL 变量的情况无法处理。

程序切片 (Program slicing)即影响变量 v 在程序 P 中某一点状态的所有语句和断言的集合。程序切片实际上是得到了程序 P 的一个有效子集，而省略了其他不相关代码，降低了代码执行的时间和空间复杂度。程序切片技术[4,5]从 1979 年 Mark Weiser 提出以来在国内外已引起了人们的关注，目前已经有了许多切片工具。国外的研究有支持 C 语言的 Wisconsin 程序切片工具 version 1.1、Chopshop、Ghinsu，Menagerie。有支持 ANSIC 的 Unravel， The Oberon Slicing Tool (OST)，支持 Java 语言的 Indus；基于 Oberon 语言的 The Linz Oberon Slicing System，以及 Microsoft's slicing tool、Spyder、PSS/Ada 系统等等。国内研究比较多的是南京大学、东南大学等，他们组成的程序切片研究小组开发的基于分层切片模型的面向 Java 程序切片工具，目前已经用于 OOPQL 语言环境中提供相关程序的切片查询服务。然而 JavaScript 语言的程序切片工具目前尚未成熟，因此本文对 JavaScript 这类基于对象语言的切片技术进行了研究，并深入探讨了 JavaScript 切片技术在网络爬虫中技术的应用。

① 基金项目:中国科学院知识创新工程重要方向项目(KGCX2-SW-511)
收稿时间:2008-10-27

2　程序切片技术描述

程序切片[6]是根据控制流和数据流分析而引进的一种程序理解技术，是一种程序分析和逆向工程技术，它通过寻找程序内部的相关性来分解

程序，再通过对所得程序切片的分析达到对整个程序的分析和理解。满足以下两个条件的程序切片被称为 Mark Weiser 程序切片:(1)一个程序切片对应一个特定的切片准则((slicing criterion) (V, i)，其中 V 表示在 i 定义或者使用的变量集合，i 是程序中的某个程序点 (一般而言是指某条语句):(2)程序 P 的切片 S 可通过从 P 中删除 0 条或者多条语句得到，但要保证程序 P 和切片 S 关于切片准则 (V, i)具有相同的行为定义。

从一个程序行为的子集开始，把程序简化为一个较小的但是仍然具有原始行的形式，减小以后的程序就称为切片，是一个独立的程序，在指定的行为子集中等价于原始程序。Weiser 所定义的切片包含了程序中所有的可能影响 i 点 V 的变量值的程序语句。也就是说，一个程序切片包含了程序中所有可能影响在些程序点上我们感兴趣的变量值的程序语句。

目前程序切片主要针对面向过程和面向对象语言，而对于基于对象的语言如 JavaScript，还没有成熟的切片工具出现。目前切片算法主要有 Weiser 的基于数据流方程的算法[7]，K.7.Ottenstein 和 L.M.Ottenstein 以及 Horwitz 的基于程序依赖图的图形可达性算法[8]，基于波动图的算法[9]等。

3　JavaScript语言概述

JavaScript 是一种基于对象的编程语言，不同于其它面向对象的语言，它没有类的概念，只有对象。JavaScript 语言的对象有三个来源：JavaScript 内部对象、主机环境中对象、程序创建的对象。

任何对象都可以作为原型对象与另一个对象联系起来，允许后一个对象来共享前一个对象的所有属性。JavaScript 提供动态继承，继承可以根据单个对象的不同而不同。而且还支持不需要任何声明的函数，函数可以是对象的属性，作为宽松类型的方法被调用执行。

任一个对象都可以定义自己的属性，不管是在创建阶段还是在运行阶段。JavaScript 在运行时可以动态的增加或者删除任何对象的属性。如果为一组对象的原型对象增加一个属性的话，那么所有继承于这个原型对象的所有对象都可获取这个新增加的属性。

JavaScript 没有 Java 等面向对象语言所具有的静态类型，也没有严格的类型检查机制。但 JavaScript 支持大部分的 Java 语言的语法和控制流结构。Java 等面向对象语言的类是通过声明来创建并在编译阶段就固定好了的，而 JavaScript 支持基于以下几个基本数据类型的运行时类系统：数字类型，布尔类型和字符串类型。

4　Ajax框架网络爬虫系统实现

Ajax 框架网络爬虫系统主要包括规则集提取模块、网络爬虫模块、程序切片模块、脚本执行模块，其工作原理图如图 1 所示。以下主要介绍程序切片与脚本执行模块。

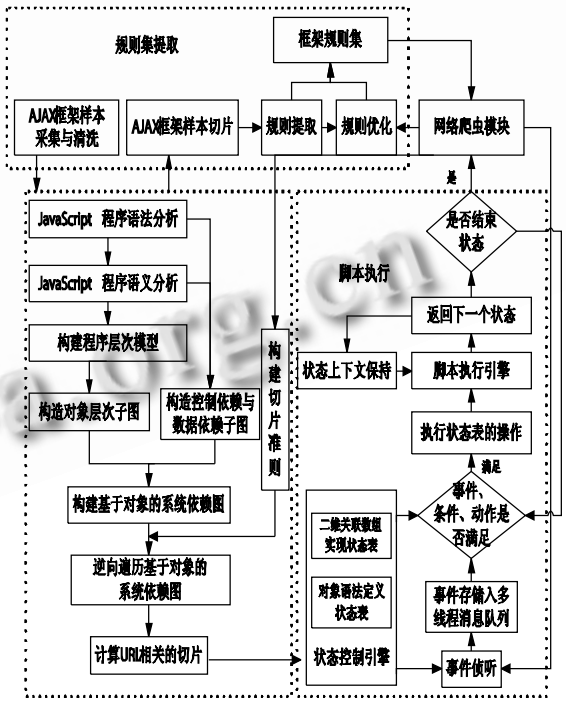

图 1　AJAX 框架爬虫系统工作原理图

4.1 JavaScript 切片算法实现

根据JavaScript 自身的特点建立JavaScript 语言的分层切片模型，采用对传统的系统依赖图进行基于对象扩充的方案进行 JavaScript 脚本切片，以获得 URL 相关的程序片段。将基于对象的程序切片计算分

成三部分：

(1) 构建出 JavaScript 程序的依赖关系

通过对 JavaScript 源程序进行基于扫描的语法分析，找出所有的对象、变量、方法的定义；按照逻辑结构将 JavaScript 分为对象层、方法层和语句层，逐层抽取程序语句间的数据依赖和控制依赖关系。通过对 JavaScript 源程序中赋值语句的左值右值，控制语句的谓词，函数调用语句对实参和全局变量的影响以及对象继承时的多态进行语义分析；针对 JavaScript 中动态定义对象的模式，实现对象中数据、方法的统一封装处理。

(2) 基于对象的系统依赖图构造方法研究

根据数据依赖和控制依赖关系构造出由对象层次子图(OHG)、控制依赖子图(CDG)、数据依赖子图(DDG)三个基本层次组成的基于对象的系统依赖图(BOSDG)。OHG 描述了基本对象的结构信息和对象层次信息，其中的顶点包括每个对象的对象首部顶点、定义在每个对象中的每个方法的方法首部顶点；边包括每个对象得对象首部顶点到与其有继承关系的对象的相应对象首部顶点的继承边，由方法首部表示的方法节点到定义该方法的对象的对象首部顶点的类成员边。当一个对象和另一个对象或者系统结合时，通过对象首部节点和对象成员边就能够方便的访问方法的信息。同时图中子对象没有重新表示从超对象中 CDG 中描述了函数方法的具体语句实现过程，采用继承的方法，因此消除了对继承方法的重复表示。用静态后向切片的方法，包含了方法的多态性表示等。DDG 中包括了对象的实现，消息动态绑定到对象中的特定方法表示，对象间的数据依赖关系等。

(3) URL 相关的程序片段切分与计算方法

利用两阶段图形可达性算法逆向遍历 BOSDG(基于对象的系统依赖图)。首先(1)在 BOSDG 上找出从节点 n 出发，沿(正向或逆向)数据依赖边或者控制依赖边可以到达的节点进行标记，构成程序关于节点 n(语句 n)的程序切片。(2)标记在 BOSDG 中与 n 相连的节点，然后标记跟这些节点相连的节点，依次计算到不能找到新的节点为止。通过上述遍历过程中的节点标记，计算出 URL 相关的程序片段。

4.2 JavaScript 切片模块，爬虫模块，脚本执行模块之间的互操作协议研究

研究三种模块之间的互操作协议，以实现三者之间的数据传递，状态控制等。同时保持了模块最大限度的独立，且能互相协作，提高了模块的高内聚性和模块间的低耦合性，降低了程序的复杂度。

(1) 构建 JavaScript 切片的有限状态机

通过切片后得到的 JavaScript 代码来构造有限状态机的状态表，以表示所有情况下响应每个事件所需的操作以及状态变量，为状态间的事件和转移组织动作。将 JavaScript 函数存储于关联数组中，将状态表实现为一个二维的函数关联数组，使用状态名称和事件名称作为索引。利用 JavaScript 中关联数组是对象的特性，使用对象语法定义状态表，将状态表直接转换成代码。

(2) 程序切片模块、脚本执行模块之间的互操作

启动事件侦听端口，用多线程消息队列存储事件，对于状态表中事件条件动作、事件动作、事件条件、条件动作、条件、动作的转换，程序切片模块判断转换条件、动作，条件满足时，脚本执行模块执行状态表中的操作，返回下一个状态，并保留状态上下文。动态跟踪程序执行过程，截获所需的中间结果以及最终结果。

(3) 程序切片模块，爬虫模块，脚本执行模块之间的互操作

动态跟踪程序执行过程，将最终结果返回给爬虫模块处理。爬虫模块继续利用 URL 参数构建切片准则，调用程序切片模块得到该参数的代码片断，返回给脚本执行模块，反复执行上述操作，直至 URL 完整构造出来。其中 JavaScript 脚本的解释处理大体可分为五大模块：JavaScript 编译模块，JavaScript 字节码解释执行模块，atom 管理模块，垃圾收集模块以及标准类管理模块。其体系结构和工作原理可参考 NetScape 公司开发的 JavaScript 解释器引擎 SpiderMonkey。

5 结束语

本文针对 AJAX 框架中大量数据不能被有效检索

(下转第 137 页)

(上接第 171 页)

的问题，研究了基于对象的程序切片算法，构建程序层次模型，解决了 URL 关联信息提取问题；研究了脚本执行引擎与切片模块的互操作协议，构造状态控制引擎，解决了切片代码的有序执行问题。从而最终解决了 AJAX 框架中 URL 提取以及网络爬虫系统实现的关键技术问题。

参考文献

1 Jesse James Garrett. Ajax: A New Approach to Web Applications.http://www.adaptivepath.com/ideas/essays/archives/000385.php2005.

2 Alvarez M, Pan A, Raposo J, Vina A. Client-Side Deep Web Data Extraction ext ended paper. http://www.tic.udc.es/~mad/publications/csdeepweb_extended.pdf.

3 王映,于满泉,李盛韬,王斌,余智华. JavaScript 引擎在动态网页采集技术中的应用.计算机应用, 2004,24(2):33－36.

4 张勇翔,李必信,郑国梁.程序切片技术的研究与应用.计算机科学, 2000,27(1):31－35.

5 Steindl C. Program slicing for object-oriented programmming languages [PhD Thesis]. Johannes Kepler University Linz.1999.

6 陆波.程序切片技术在程序理解中的应用研究[硕士学位论文].济南:山东大学, 2004.

7 Weiser M. Program slicing. IEEE Transactions on Software Engineering, July1984.

8 Ottenstein KJ, Ottenstei LM. The program dependence graph in a softwaredevelopment environment. Proceedings of the ACM SIGSOFT/ SIGP LAN software Engineering Sysposium on Practical Software Development Environments, ACM SIGPLAN Notices. 1984,19(5).

9 董志宏.面向对象程序的波动分析及其在程序切片中的应用[硕士学位论文].南京:南京大学, 2001.