Research on the Algorithm for Coarse-Grained Web Program Slicing

Lin Du[1, a], Yehong Han[2, b]

[1]School of Information Science and Engineering, University of Qilu Normal, Jinan 250014, China

School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou 221116, China

[2]School of Information Science and Engineering, University of Qilu Normal, Jinan 250014, China

[a]email:dul1028@163.com, [b]email:sdzzhyh@163.com

Keywords: Dependence Graph; Coarse-Grained; Algorithms Implement; Program Slicing

Abstract. A novel approach based on constructing coarse-grained system dependence is proposed to compute web program slicing. The method perfects web program semantics and reduces the computation complexity through expanding the signification of coarse-grained and analyzing the dependence among semantic units. Program semantic units are described in detail. The expression of dependence includes data dependence, control dependence and transfer dependence. At length, two algorithms for constructing system dependence graph and computing coarse-grained program slicing are designed.

Introduction

Program slicing is a technique for simplifying programs by focusing on selected aspects of semantics unit. The process of slicing deletes those parts of the program which can be determined to have no effect upon the semantics of interest. As a valid method to restrict the focus of a task to specific sub-components of a program, program slicing has extensive applications in software engineering [1]. The applications include program debugging, program testing, software metrics and software maintenance [2].Web program is a special kind of web application for releasing information and accepting input through web pages. With the rapid development of internet technology and www technology, Web application relates to different fields more widely. Recent years have witnessed the increasing of web applications with the advent of .net, J2EE and other new web integration framework [3]. Meanwhile, the size and complexity of web applications are also increasing. Web's rapid build features present the understanding, analysis and testing of the programs with difficulties.Program slicing technology is applied to web application analysis in this paper. Developers can focus on only those procedures associated with certain web pages or achieving a particular function, thus the reliability and validity of web program are improved.

This paper aims to compute coarse-grained web program slicing for obtaining interest points and hierarchy information of web application through analyzing the dependencies between web programs. The program slicing in the paper based on the dependency analysis of program's dataflow, control flow and interaction between the server application and client application.

Simplifying System Dependence Graph

In order to understand web program, analyzing the interaction relationships among multiple units is better than analyzing single statement. The definition of coarse-grained is enlarged in order to make the size of grain come up to the web program's semantic unit. Coarse-grained is defined as follows.

Definition.1 Coarse-grained: The graph G which meets the following characters is defined as coarse-grained. (1) Graph G contains the following units which include statement and predicate in the main, class, instance, member method and member variable. (2) In the member method M, if a statement belongs to G, then M also belongs to G. (3) If the member method, member variable and instance in the class A belongs to G, then class A also belongs to G.

System dependence graph is simplified on the basis of above definition of the coarse-grained. Take the following graph for example, fig.1 is a traditional system dependence graph, fig.2 shows the meaning of edges in the graph, and fig.3 is the simplified result.

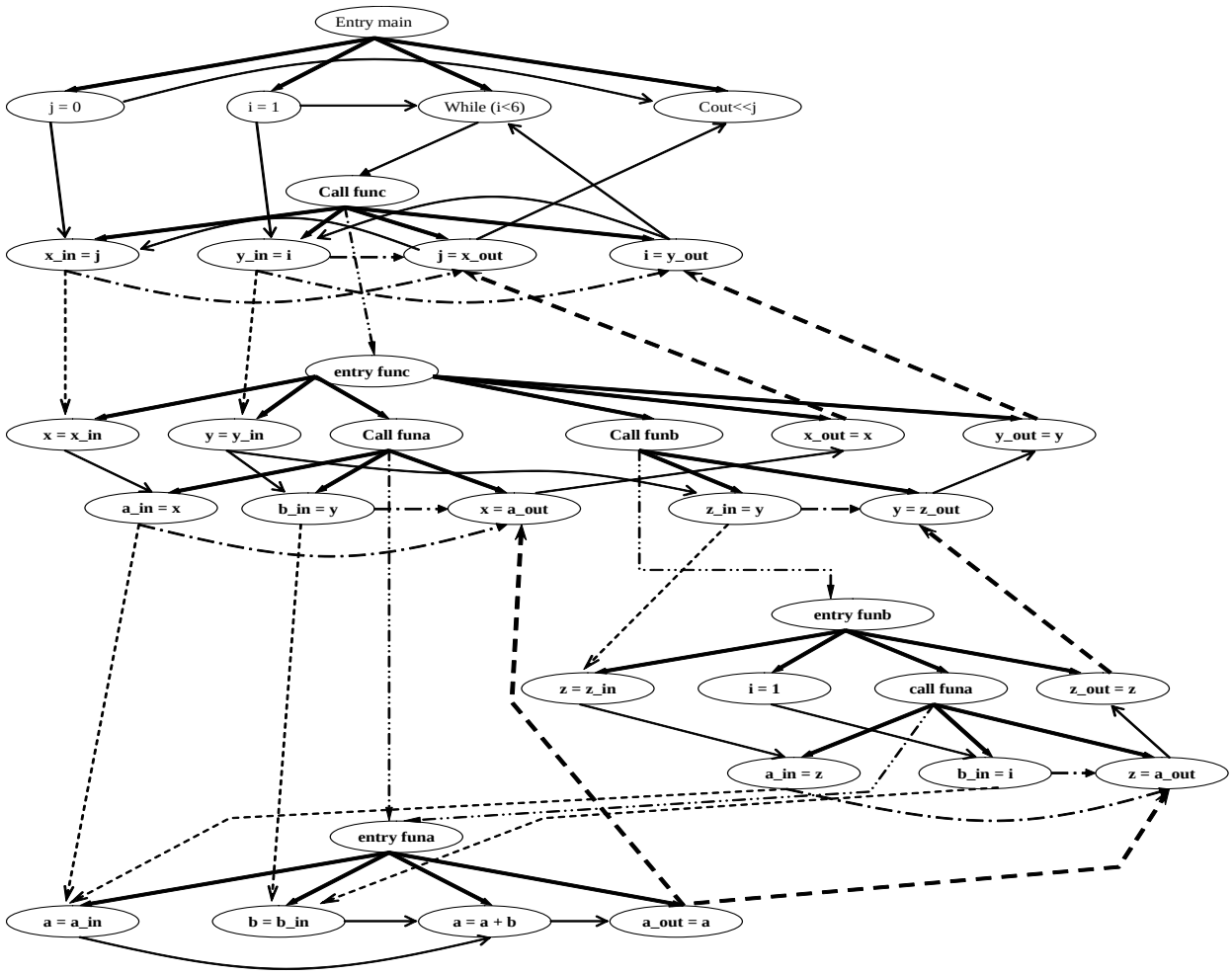

Fig.1 A case of traditional system dependence graph

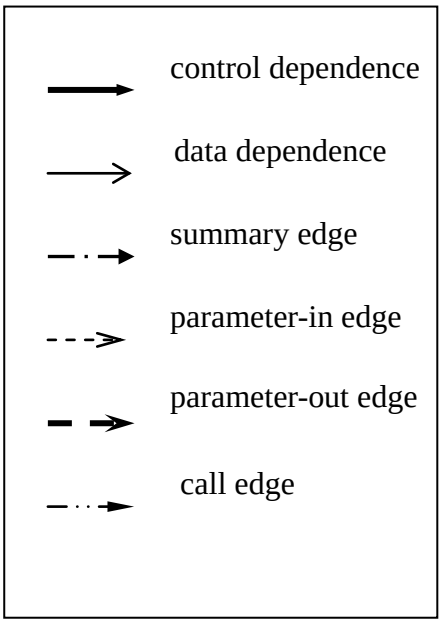

Fig.2 The representation of edges

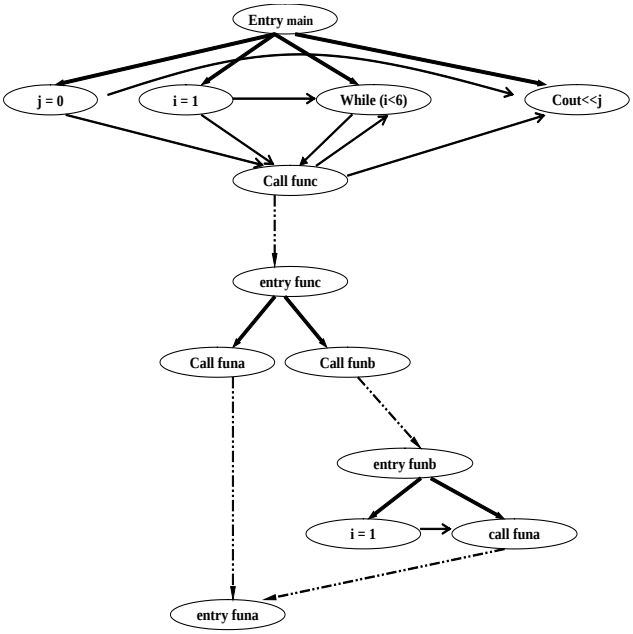

Fig.3 Simplified system dependence graph

The description of the process dependence doesn't enter the process inside but indicate process prelude node simply. The data dependence between parameter nodes of different methods is indicated by data dependence among multiple methods. It is achieved by data dependence edge which point to the call directly.

Semantic Description of System *Dependence Graph*

When the class, instance, member method and member variables are described, for the sake of expressing membership, instance node, member method prelude node and member variable node are connected to the accessory class prelude node. The method and process do not achieve internal processing but provide prelude node. The meaning of method prelude node is expanded through hiding data transference among multiple parameter nodes. Data dependence among parameter nodes of different methods relies on data dependence among methods. That is to say, three kinds of different nodes are increased in system dependence graph. Instance nodes are increased in order to express that member method refers to class instance node. Member variable nodes are increased in order to express that member method refers to member variable. Instance nodes which express message receiver object are increased in the method node in order to reflect the change of object's state. All of the class prelude nodes which have inheritance are connected for the sake of expressing inheritance. The class prelude node and class member edge are connected because of the interaction of different classes. When a virtual method in the child class which inherits from the parent class is modified, the method is described only in the child class. What's more, the associated edge should be increased between class prelude node of the parent class and method prelude node of the child class. Above method makes the expression of inheritance mechanism and virtual method doesn't require increasing associated edge between method of the parent class and method of the child class. The associated edge is only increased between class prelude node and method prelude node. The virtual method prelude node and polymorphism call edge are increased to represent polymorphism. The call nodes are connected to each method node which is called by object possibly by multiple call edge. The multiple polymorphism nodes which have the same protocol represent dynamic selection. The description of the process dependence doesn't enter the process inside but indicate process prelude node [4]. The expression of the dependence which belongs to parameter nodes of different methods is indicated by data dependence among multiple methods. Meanwhile, the data dependence edge which point to the call directly is constructed. Fig.4 shows the result of describing the dependence of case codes [5].

```
L1:   D2Vector * vp;
L2:   D3Vector v3(10, 10, 10);
L3:   int v3sum;
L4:   int dim_z;
L5:   if (argc > 1)
L6:   vp = new D3Vector(1, 1, 1);
L7:   vp = new D2Vector(1, 1);
L8:   vp ->scale(10);
L9:   v3sum = sum(v3);
L10:  dim_z = extend_z(v3, 10);
L11:  cout<<v3sum<<endl;
L12:  cout<<dim_z<<endl; }
```

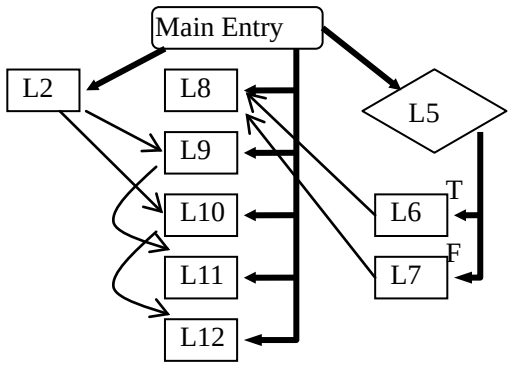

Fig.4 Case codes and the description of dependence

The system dependence graph describes three kinds of dependence. The data dependence is aroused by variable relationships. The control branch conditions give rise to the control dependence. The interactions between client programs and server programs lead to transfer dependence.

Algorithm Design

Afterwards, two algorithms are designed in order to construct system dependence graph and compute coarse-grained program slicing.

Algorithm.1 Construct system dependence graph

```
Input: the abstract syntax tree of P = (M, C)
Output: the SDG of P
void Construct SDG ( )
{ for (class Ci of C)
{ for (method m defined in Ci)
{if (m is “marked”)
make Ci and the“marked” method of base class connected as membership;
else{
calculate the PrDG of m;
make m as“marked”;
}}}
connect( );
}//end Construct SDG
```

Algorithm.2 Compute coarse-grained program slicing

```
Input: the SDG of P
Output: coarse-grained slicing
void SliceNodeObject (Node node, EdgeSet includeEdges, NodeSet visitedNodes, BOOL
        back)
{if (node is not marked){
mark node as visited;
insert node into visitedNodes;
if ( back ){
for (all edges e leading from other node n to node)
{
if (kind of e is in includeEdges)
SliceNodeObject(n, includeEdges, visitedNodes, back);
}//end for1}
else
for (all edges e leading from node to other node n )
{
if (kind of e is in includeEdges )
SliceNodeObject(n, includeEdges, visitedNodes, back);
}//end for2}//end if2}//end if1}//end SliceNodeObject
void ComputeSlice (Node node)
{
// phase 1
SliceNodeObject (node, {control dependency edge, data dependency edge, polymorphic call
edge ,call edge},visitedNodes, FALSE);
// phase 2
for ( all nodes n in visitedNodes )
SliceNodeObject (n, {call edge,instance edge }, visitedNodes, TRUE);
} //end ComputeSlice
```

The description of the system dependence graph is reduced to the statement, the predicate in the main and class, instance, member method and member variable achieved by above semantic units. The program is represented as a two-tuples. The two-tuples is expressed (M, C) in which M is the main and C is a collection of classes [6]. The algorithm calls the function connect which connect call node of process dependence graph and method prelude node. Meanwhile class graph and the main program are connected.

Web programs include two kinds of data dependence. The first is caused by member methods of JavaScript program and servlet program. The second is caused by html program and java program embedded in JSP pages. The control predicate which includes condition control and circulation control lead to control dependence [7,8]. The interaction between JSP program in client unit and servlet program in server unit give rise to transfer dependence. The browser sends the client's data to the web server. Servlet container in the server calls the JSP program and compiler automatically. The script is executed and generates some output. The servlet container sends the output to the browser in the client.

Conclusion

In this paper, a novel method to compute web program slicing is proposed. The definition of coarse-grained is extended for the sake of making the size of grain come up to program's semantic units which include class, instance, member method and member variable. Two algorithms for constructing system dependence graph and compute coarse-grained program slicing are designed. Above algorithms correctly reflect the semantic of programs and improve the precision of program slicing. Our method provides a solid foundation for the further analysis of reengineering of legacy software. We strongly believe that, in the near future, this research field will promote the fundamental theory research in the related software engineering fields.

Acknowledgement

In this paper, the research was sponsored by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2013FL010).

References

[1] B.Korel. Slicing of state based models[C]. Proceedings of the IEEE International Conference on Software Maintenance.2003 34-43.

[2] Mary Jean Harrold. Regression Test Selection for Java Software [C].OOPSLA. 2001 313-326.

[3] Xu BW. Comments on a cohesion measure for object-oriented classes [J]. Software--Practice and Experience.2001 31(14) 1381-1388.

[4] Chen ZQ. A novel approach to measure class cohesion based on dependence analysis[C]. IEEE International Conference on Software Maintenance.2002 377-383.

[5] Chen ZQ. Slicing object-oriented Java programs [J]. ACM SIGPLAN Notices.2001 36(4) 33-40

[6] Xu BW, Chen ZQ, Zhou XY. Slicing object-oriented Ada95 programs based on dependence analysis [J]. Journal of Software.2001 12(12) 208-213.

[7] Horwitz, S.Liblit, B.Polishchuk. Better Debugging via Output Tracing and Call -stack Slicing [J].Software Engineering.2010 36(1) 7-19.

[8] Rupak Majumdar, Ru-Gang Xu. Reducing Test Inputs Using Information Partitions [J]. Lecture Notes in Computer Science.2009 56(4) 555-569.