An Effective Approach of Points-To Analysis

Zhang Yuping[1, a], Deng Zhaori [1], Zhang Xiaoning[1] and Ma Yan[1]
1College Of Information Mechanical And Electrical Engineering, Shanghai Normal University, Shanghai, China
[a]yp_zhang@shnu.edu.cn

Abstract.

Aim at the potential that the time efficiency of the technology of Cycle elimination for invocation graph-based context-sensitive pointer analysis can be improved. Through using wave and deep propagation method, which is the state-of-the-art techniques of Inclusion Based Pointer Analysis on-line optimization Technology, to optimize the technology of cycle elimination for invocation graph-based context-sensitive pointer analysis, and then puts forward a context-sensitive wave and deep propagation algorithm. First, introduces the definition and initialization of new constraint graph; secondly, through an example to describe the context-sensitive wave and deep propagation, which can accurately and efficiently realize context-sensitive pointer analysis; finally, use the CIL tool for experiment, the experimental results show that these new algorithms save about 9s or 10s when the original technology and the new algorithm are all analyzing the same large scale program.

Keywords: Optimization, Invocation, Context-sensitive process, Points-to analysis, Cycle elimination, Call graph

Introduction

Pointer analysis refers to the analysis of program to approximately calculate the point to the set of the pointer expression. Based on analysis method containing a pointer[1] is a kind of important context-free pointer analysis method (merge all invocations).Context-sensitive pointer analysis is used to distinguish between different invocation points of the same functions. In recent study, for example, Binary Decision Diagram[2] can well handle highly context-sensitive pointer analysis, but in the further scalability analysis[3] it did not perform predefined constraints. However, Woongsik Choi etc[4]. puts forward the technology of Cycle elimination for invocation graph-based context-sensitive pointer analysis which has both high context-sensitivity and good scalability, whose time analysis efficiency still has potential for improvement. Fortunately, there have appeared a lot of the optimization techniques[5] based on analysis method containing a pointer during the past 20 years. Following Fahndrichng[5], Pearce[8,9], Pereira[6] successively puts forward some cycle detection and elimination technology of constraint graph which have their respective advantages and disadvantages .Wave Propagation(referred to WP)and Deep Propagation(referred to DP) of Pereira[6]

are the most outstanding one, because it can greatly improve the time and space efficiency analysis. Therefore, in this paper, by using the WP and DP algorithm to optimize the cycle elimination technology to respectively proposed context-sensitive WP and DP algorithm, which can accurately express and realize the cycle elimination technique more effectively. Finally, in the CIL[7], using OCaml language implementation analysis, and analyzing the six program on the line of 20,000 ~ 290,000 prove that the time efficiency of the new algorithm is better than the Cycle elimination technology.

1 Constraint graph and its initialization

1.1 A new definition of constraint graph

Context-sensitivity is combined with the new constraint graph. First of all, each node has the attribute of cs (true: the corresponding variable of the node is context-sensitive, or is context-free) and ct (for the context-free variables, the value is the context set, stating that another variable point to the variable in the set). For example, a ρ{b}, if a is context-sensitive, b is context-free, there will be node a(cs is true, ct is null) and b (cs stand foe false, ct stand for ρ) in the graph, indicating that a point to b in the context of ρ. The circle of context-free variable is gray, otherwise is white.

In addition, side is divided into three categories: common sides (solid arrow); calling sides(solid arrow ,standing for function calls) namely, the side of arguments point to the formal parameters ;and return sides (dotted arrow, standing for call returns), namely, the return variables point to the side of the variables receiving its value. The identifier of side E(v,w,γ,ρ) is ρ(when ρ is ⊤,there will be no identifier) .v, w stand for respectively side head and tail node.γ stands for category; its value is 1,2,3,which respectively means common sides, calling sides and return sides. When the value of γ equals to 1, ρis the representative of context set, indicating variable w in the ρ includes when the value of γ equals to 2, ρis the numberm, indicating this side is calling the function of point m, and V, w respectively stand for arguments and formal parameters. When the value of γ equals to 3, ρis the number m, indicating call returns of point m, and V, w respectively stand for return variable and variable receiving value. For example , a ρb , refer to common side(b,a,1,ρ); a f(b)1 refer to calling side (b, ,2,ι) and return side (,a,3,ι)。

1.2 Initializes the constraint graph

If a is a local or global or stacks variable addressed, then it is context-free, otherwise it is context-sensitive. Assuming that the context-free pointer analysis has already obtained the function pointer and function calls graph[4],the variable has no duplication of name.

Both initial constraint concentration value and the context identifier of each variable constraint is ⊤ .Every calling point ι corresponds context ρι=(ι,⊤, ⊥).Point 1 and point 2 is context ρ1=(1,⊤, ⊥) and ρ2=(2,⊤, ⊥) respectively. Context-sensitive: a,b,c,d,e,s,t,v,w; context-free : p,q,x,y.

First of all, focus on building a node having the same name for every variable, initialize the cs attribute of every node, then process value constraints, assign the variable of the right side corresponding value. For example, a⊤b, the ct of b equal to ⊤.Then, add one side to graph for every variable constrains to initial the attribute of this side. For example, c ⊤ e, add (e,c,1,⊤).At last, add the function calling constraints. For example, v f(a,b)1, add side (a,s,2,1), (b,t,2,1) and (t,v,3,1).

Focus point, for example, xρ, the right subscript is the cs attribute of x. If it is ⊤, node x will be represented as x.

2 Wave propagation considering the context sensitivity

2.1 The new frame of wave propagation

The new WP is the while loop from Algorithm1, with its condition always true. The input is initial constraint graph G=(V,E), the output is the mapping of every node to the point set. First of all, calling Algorithm2 in the graph to explore and merge ring; then calling the Algorithm4 for different transmission[5]. In the end, calling Algorithm5 to process complex constraints to add new side.

2.2 Cycle detection and elimination

Algorithm3 combined detection loop method in[4]: Detect cycle of common side, only when the value of ρ equal to ⊤ (condition of 2nd row). Detect the cycle consisting of (e,c,1,⊤)(c,e,1,⊤), and merge c, e into c, then, the point set of c, e is q⊤. There is at least one context-free node in the cycle, then set all the nodes in the cycle to false. All of the point set of the nodes are the combination of point set of the junction point source. Chose one node as the representation.

2.3 Difference propagation

Algorithm4 by combining the context-sensitive constraint solve [trans1], [trans2], [param1], [param2], [ret1], [ret 2] and [ret3] of the rules to the algorithm 4 of literature[6].

2.4 deal with complex constraints

Algorithm5 (deal with complex constraints except function call) combines load1, load2, ret1, ret2, ret3[4].

3 Deep propagation considering context sensitivity

3.1 The new frame of deep propagation

Just like Algorithm6, the input of the new DP is a initialized original constraint graph; the output is the mapping of each node to the point set. Explore and merge the cycle (Algorithm2) at first, then execute difference propagation again. In the

end, execute repeated circulation to deal with complex constraints until the point set of the entire node does not change after calling Algorithm7.

3.2 Cycle detection and elimination

The method of the cycle detected and eliminated is the same as the context-sensitive wave propagation. The result of the cycle detected and eliminated is shown in Figure1(b).

3.3 Difference propagation

The method of difference propagation is the same as the wave propagation. As shown in Algorithm4, Figure1(c) can be made through difference propagation.

The new Algorithm6 can be made by combing [load1], [load2], [ret1], [ret2], [ret3][4] to literature [6], used to deal with complex constrains. Note: there is no need dealing with functions to call a f(b)1 .

For example, after dealing with complex constraints, Figure1(c) will turn into Figure1(d). According to Figure1(c) , the point of s is { pρ1, qρ2}. As for pρ1, because the cs value of t is true and

Algorithm7 The Deep Propagation Routine. Input: the point-to set Pdif that must be propagated, the n node v with the edge (t,v, γ, ρ) and that is been visited and the stop point s. Output: true if stop point s is reachable from v, and false otherwise. Require: Pcur(r) Pcur(w) if w is reachable from v.Ensure: Pcur(r) Pcur(w) if w is reachable from v.

side(t,p,1,ρ1) is not in the Figure1(c), deal with it, and add this side to the graph (it is no need to execute difference propagation on this new side). Then, take side (t,p,1, ρ1) as parameter from p , and take r as the stop node for deep propagation (call Algorithm7). As a result, the point set of p is {x$\top$} and be marked black. Namely, p cannot reach to r (the ρ attribute of all of the sides is $\top$ in the path reached to).

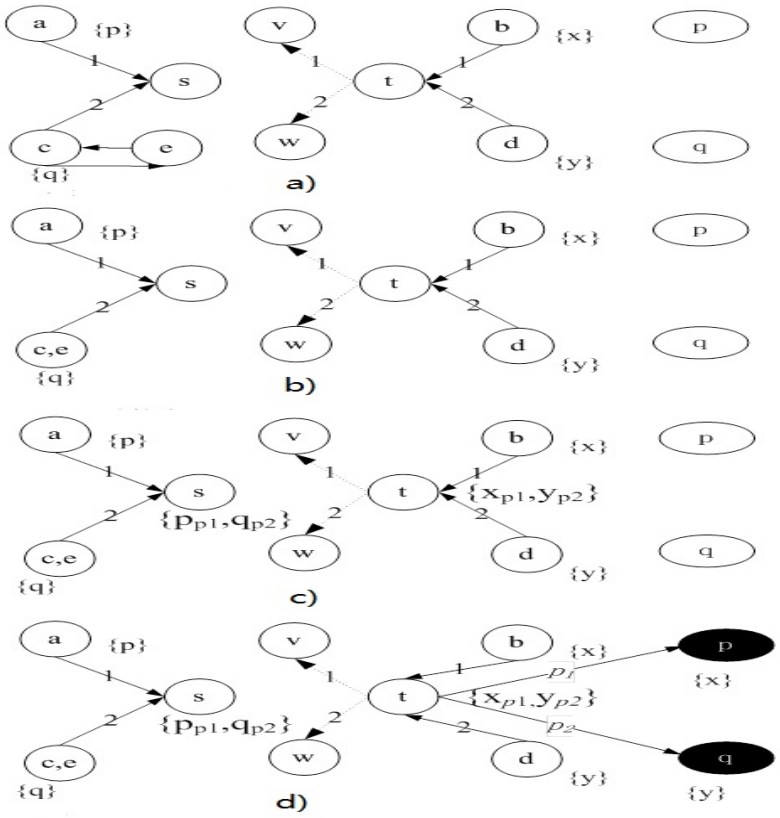

Figure 1 the example of analysis

Similarly, as for qρ2 , the new side (t,q,1, ρ2) is added to the graph, the point set of q is {y⊤}, marked as black, as shown in Figure1(d). Because there is no new side added to the latest figure after dealing with *s t and executing the repeat loop of Algorithm6, Algorithm1 terminates, which means the analysis finished.

4 Experiments

The new algorithm makes use of WP and DP to optimize the Cycle elimination technology, but the former is aimed at improving the efficiency of analysis and has no effect on the accuracy of the technology. So the time efficiency is focused on this experiment. The experiment adopts the experimental methods of literature[3],

implemented the analysis with Ocaml language in CIL[7] , borrowing the hash-consing of BuBDDy, conducting an experiment with a computer (CPU: Inter i7,3.91GHz,Memory: 12GB).The six programs in table 1 are analyzed six times to obtain the average time of the analysis, verifying the time and the Cycle elimination efficiency of the two new algorithm. The result of the experiment is shown in table 2, listing the average value and range of six analyses.

The list of row in table 1 is the preprocessing lines of code. The list of variable 1 is the number of the context-sensitive variable. The list of variable 2 is all possible contexts in the program.

Program	Version	Linage	Variable 1	Variable 2	Context
make	3.81	24,889	10,499	2028	5×10^{6}
tar	1.20	56,105	15,209	624	3×10^{7}
sqlite	3.6.16	65,235	38,074	640	6×10^{9}
povray	3.1g	81,941	26,896	891	2×10^{10}
python	2.0.1	163,824	52,092	206	1×10^{11}
named	9.4.3	295,766	87,476	14,979	2×10^{8}

Table 1 the list of experimental source program

Program	Cycle elimination technique(s)	Average value (s)		
		The new WP algorithm	The new DP algorithm	T a
make	0.2	0.10	0.19	
tar	1.3	0.88	0.8	
sqlite	33.4	32.4	31.7	
povray	9.3	8.32	8.2	
python	233.9	225.4	224.6	
named	484.2	475.4	474.3	4

Table 2 the time of analysis

What is listed in table 2 is the time comparison that the two new algorithms spend on the analysis of the same source program. According to table 2, because Cycle elimination technique adopts the wave propagation and deep propagation of online optimal technique, and our new algorithm performs more excellently in the terms of time efficiency.

5 Conclusion

This article puts forward two new algorithms by combing the method of WP and DP: the former optimizes the latter. The result of the experiment states that the new algorithm improves the time efficiency and Cycle elimination efficiency of Cycle elimination technique. However, this new algorithm is not applied to the specific problems, such as the detection of null pointer and so on. In the future, the application of this algorithm in the static analysis of the program will be realized.

Acknowledgements

This work was financially supported by Research and Innovation Fund Project of Shanghai Normal University (SK201413) and the National Natural Science Foundation (61373004).

Reference

[1] Andersen L O, Program analysis and specialization for the C programming language[D]. Copenhagen, University of Copenhagen, 1994

[2] Bryant R.E, Graph-based algorithms for boolean function manipulation[J]. IEEE Transactions on Computers, 1986, 35 (8):677–691

[3] Avots D, Dalton M, Livshits V.B, Lam M.S, Improving software security with a C pointer analysis[C]// Proc . of the International Conference on Software Engineering, St. Louis, MO, USA, 2005:332–341

[4] Woongsik Choi, Kwang-Moo Choe, Cycle elimination for invocation graph-based context-sensitive pointer analysis[J]. Information and Software Technology, 2011, 53(8):818-833

[5] Fahndrich M, Foster J S, Su Z, Aiken A, Partial online cycle elimination in inclusion constraint graphs[J]. ACM SIGPLAN Notices, 1998, 33(5): 85-96

[6] Pereira F M Q, Berlin D, Wave Propagation and Deep Propagation for Pointer Analysis analysis[C]//Proc . of the 2009 International Symposium on Code Generation and Optimization. Washington, DC, USA, 2009: 126-135

[7] Necula G C, McPeak S, Rahul S P, Weimer W, CIL: Intermediate language and tools for analysis and transformation of C programs[J]. Lecture Notes in Computer Science, 2002, 2304:209-265