4.3 有效性威胁

本文实验有如下有效性威胁.

- 实验采用了 JSlice 的动态切片工具.然而,由于切片技术与 JSlice 工具局限性,与系统库相关的依赖无法提取.这时,错误语句很有可能不包含在切片结果中,如 Tot_info 的版本 11.动态切片技术的另一个缺点是对某些类型的错误无法实施切片,例如,遗漏类型的错误语句无法产生执行信息,动态切片无法切到该类型的错误语句.这些缺点是由动态切片技术自身特点所引起的.
- 实验有一个潜在假设:当一个失败的测试用例运行时,错误的语句应该被执行.这个假设通常成立,实验结果也证实了假设的合理性.然而在某些情况下,如程序中出现多重错误时,则假设可能不成立.但是对于单一错误进行研究是必要的,因为它们是对多个错误进行定位的研究基础.这就是为什么大多数现有的研究集中在单一错误情景的原因.
- 实验对象也是有效性威胁之一.实验选择广泛应用于错误定位领域的代表性程序.然而现实调试中存在许多未知因素,说明它们不能覆盖并适用于现实中的所有情况.因此,未来工作使用更多的真实大型程序将会是本文研究的重点.

5 相关工作

有许多研究人员根据覆盖信息或切片信息研究错误定位技术.本节简要介绍这些研究.更多的错误定位工作可参考 Wong 等人近期发表的综述[1].

基于覆盖的错误定位技术将程序谱数据从测试执行转换为程序实体的可疑值,并使用诸如基于程序谱的错误定位(SFL)等统计公式对其进行排序.当使用这些技术时,本文不需要知道程序的详细信息,只需运行通过和失败的测试用例.Chen 等人[29]提出了 Jaccard 技术,这是一种统计错误定位算法.Jones 等人[30]提出了 Tarantula 技术,计算每个语句的可疑值,并根据他们的可疑值进行排名.Tarantula 在后续研究中是广泛使用和比较的技术.Abreu 等人[31]应用 Ochiai 定位单一错误.他们指出:他们的技术 Ochiai 一直优于 Tarantula,并且在 EXAM 得分方面平均提高了 5%.Abreu 等人[32]在后来的研究中评估了 Tarantula 的有效性以及其他技术,并指出,Ochiai 对测试用例进行了最佳的评估.Wong 等人[33]利用数据和控制流程,并提出了几个指标,如 Wong1-3,Wong3′.它们都是计算具有失败测试用例的程序实体的可疑值的统计公式.虽然 Wong 等人[8]提出了一种基于代码覆盖的方法,可以通过测试执行自动调整可疑语句的权重,还提供了一些减少搜索域的方法.Wong 等人[34,35]还提出了一种基于交叉表的方法,利用语句的覆盖和执行信息,并提出了一种名为 DStar(D*)的技术.该方法计算了修改语句的可疑值.最近,Xie 等人[14]在理论上研究了 GP 进化公式.

程序切片算法(静态和动态)的应用也被许多研究人员广泛研究用于程序调试.Lei 等人[27,28]采用静态切片对每个测试用例的输出进行切片,以此构建输入矩阵,并基于这个矩阵重新定义可疑值计算方法.本文方法仅对失败测试用例的输出进行动态切片,其针对性强,避免了成功测试用例可能带来的不确定影响.同时,动态切片体积小于静态切片体积,搜索范围更小.本文实验结果也表明,Context-FL 优于 Lei 等人的方法.Zhang 等人[36,37]研究了动态切片在定位错误中的有效性,并开发了一种采用动态切片的策略,通过计算从 0 到 1 的置信度值,来识别可能影响输出以产生不正确值的语句的子集.Zhang 等人[38]还提出了一种类似切片的技术,以便从许多事件中剪除不相关的事件.Jones 等人[30]使用动态切片和执行切片来减少搜索域.Alves 等人[39]将变化影响分析纳入动态切片,以进一步优化动态切片.Wen 等人[40]提出了利用混合切片谱的统计方法,以提高错误定位的有效性.与这些方法不同,本文方法使用的是动态切片,对程序的动态执行轨迹进行了分析,显示了如何捕获动态数据和控制依赖关系.Wong 等人[41]提出了一种基于执行切片和块之间的数据依赖关系,在小范围代码内实现高效准确的错误定位的方法.我们的方法与它不同.我们利用动态切片来构建语句的上下文关系.

偶然正确性(coincidental correctness)问题是错误定位研究的一个热点.当测试用例执行了错误语句,却并没有导致程序失效时,就产生了偶然正确性问题.偶然正确性问题存在于成功测试用例中,对错误定位性能有负效应.为了解决偶然正确性问题,如何识别存在偶然正确性问题的成功测试用例成是关键.Wang 等人[42]提出了上

下文模型(context pattern),通过是否匹配上下文模型来识别存在偶然正确性问题的成功测试用例,并以此优化覆盖矩阵,从而提升错误定位性能.Masri 等人[43]定义了基于缺陷的失效,以此描述具有偶然正确性问题的成功测试用例,从而更有利于缓解偶然正确性问题.随后,Masri 等人[44]对偶然正确性问题进一步研究,采用 Euclidean 标准度量出成功测试用例与失败测试用例之间的相似度,以此移除具有偶然正确性问题的成功测试用例.Miao 等人[45]基于偶然正确性的成功测试用例与失败测试用例有相似行为的思想,采用聚类方法对测试用例进行分类.当成功测试用例被划分到包含失败测试用例的类别时,该成功测试用例存在偶然正确性问题的可能性较大.Bandyopadhyay[46]基于成功测试用例与失败测试用例之间执行语句的相似度来定义权重,并以此预测可能具有偶然正确性问题的成功测试用例,最后,根据预测来优化错误定位性能.Lei 等人[47]系统分析和总结了测试用例集的错误定位效能,从理论和实验证明了偶然正确性问题是成功测试用例对错误定位效能影响最大的因素.他们的研究还明确失败测试用例对错误定位效能具有正效应作用.与成功测试用例的偶然正确性问题相比,失败测试用例的信息更加直接有效.因此,本文方法关注于失败测试用例,通过动态切片提取更精确信息,大幅度缩小错误搜索范围,以此为基础来实现错误定位性能提升,也印证了 Lei 等人[47]对失败测试用例的结论.虽然本文方法不关注优化具有偶然正确性问题的成功测试用例,且采用的动态切片不能根除偶然正确性,但是通过动态切片对失败测试用例的优化,大幅度缩小错误搜索范围,抑制了具有偶然正确性问题的成功测试用例发挥负效应的空间,间接地缓解了偶然正确性问题.

6 结 论

本文提出了一种基于上下文的错误定位方法 Context-FL.该方法结合动态切片和可疑性度量方法,构建可疑值标记的上下文.实验结果表明,与 5 种典型定位方法对比,Context-FL 显著优于这些方法,有效缩减了代码检查的范围,大幅度提升了错误定位性能.未来的工作包括方法优化,以提高其准确性.此外,我们还会研究将 Context-FL 扩展到多错误场景下的定位.

References:

[1] Wong WE, Gao R, Li Y, Rui A. A survey on software fault localization. IEEE Trans. on Software Engineering, 2016,42(8): 707−740.

[2] Xie X, Kuo FC, Chen TY, Yoo S, Harman M. Provably optimal and human-competitive results in SBSE for spectrum based fault localisation. In: Proc. of the 5th Symp. on Search-based Software Engineering. St. Petersburg: Springer-Verlag, 2013. 224−238.

[3] Acharya M, Robinson B. Practical change impact analysis based on static program slicing for industrial software systems. In: Proc. of the Int'l Conf. on Software Engineering. 2012. 746−765.

[4] Pearson S, Campos J, Just R, *et al*. Evaluating and improving fault localization. In: Proc. of the Int'l Conf. on Software Engineering. Buenos Aires: IEEE, 2017. 609−620.

[5] Naish L, Lee HJ, Ramamohanarao K. A model for spectra-based software diagnosis. ACM Trans. on Software Engineering and Methodology, 2011,20(3):1−32.

[6] Yoo S. Evolving human competitive spectra-based fault localisation techniques. Int'l Conf. on Search Based Software Engineering, 2012,7515:244−258.

[7] Abreu R, Zoeteweij P, van Gemund AJC. On the accuracy of spectrum-based fault localization. In: Proc. of the Testing: Academic and Industrial Conf. on Practice and Research Techniques (MUTATION). Windsor: IEEE, 2007. 89−98.

[8] Wong WE, Debroy V, Choi B. A family of code coverage-based heuristics for effective fault localization. Journal of Systems and Software, 2010,83(2):188−208.

[9] Parnin C, Orso A. Are automated debugging techniques actually helping programmers. In: Proc. of the 2011 Int'l Symp. on Software Testing and Analysis. Toronto: ACM Press, 2011. 199−209.

[10] Zhang Z, Mao XG, Lei Y, Zhang P. Enriching contextual information for fault localization. IEICE Trans. on Information and Systems, 2014,E97.D(6):1652−1655.

[11] Weiser M. Program slicing. IEEE Trans. on Software Engineering, 1984,SE-10(4):352−357.

[12] Korel B, Laski J. Dynamic program slicing. Information Processing Letters, 1988,29(3):155−163.

[13] Zhang X, Gupta R, Zhang Y. Efficient forward computation of dynamic slices using reduced ordered binary decision diagrams. In: Proc. of the 26th Int'l Conf. on Software Engineering. Edinburgh: IEEE, 2004. 502−511.

[14] Xie X, Chen TY, Kuo FC, Xu B. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. ACM Trans. on Software Engineering and Methodology, 2013,22(4):1−40.

[15] Gallagher K, Lyle J. Using program slicing in software maintenance. IEEE Trans. on Software Engineering, 1991,17(17):751−761.

[16] Gupta R, Harrold M, Soffa M. An approach to regression testing using slicing. In: Proc. of the Conf. on Software Maintenance. Orlando: IEEE, 1992. 299−308.

[17] Horwitz S, Reps T, Binkley D. Interprocedural slicing using dependence graphs. ACM Sigplan Conf. on Programming Language Design and Implementation, 1990,12(1):26−60.

[18] Weiser M. Programmers use slices when debugging. Communications of the ACM, 1982,25(7):446−452.

[19] Nanda MG. Slicing concurrent programs. In: Proc. of the ISSTA 2000. 1993. 223−240.

[20] Duesterwald E, Gupta R, Soffa ML. Distributed slicing and partial re-execution for distributed programs. In: Proc. of the Int'l Workshop on Languages and Compilers for Parallel Computing. New Haven: Springer-Verlag, 1992. 329−337.

[21] EMMA. http://emma.sourceforge.net/

[22] Jslice. http://jslice.sourceforge.net/

[23] Wang T, Roychoudhury A. Using compressed bytecode traces for slicing Java programs. In: Proc. of the ACM/IEEE Int'l Conf. on Software Engineering. Edinburgh: IEEE, 2004. 512−521.

[24] SIR. http://sir.unl.edu/portal/index.php

[25] defects4j. http://defects4j.org

[26] Debroy V, Wong WE, Xu X, Choi B. A grouping-based strategy to improve the effectiveness of fault localization techniques. In: Proc. of the 10th Int'l Conf. on Quality Software. Zhangjiajie: IEEE Computer Society, 2010. 13−22.

[27] Lei Y, Mao XG, Dai ZY, Wang CS. Effective statistical fault localization using program slices. In: Proc. of the 36th Annual Int'l Computer Software and Applications Conf. Izmir: IEEE Computer Society, 2012. 1−10.

[28] Lei Y, Mao X, Dai Z, Qi Y, Wang C. Slice-based statistical fault localization. Journal of Systems and Software, 2014,89(1):51−56.

[29] Chen M, Kiciman E, Fratkin E, Fox A, Brewer E. Pinpoint: Problem determination in large, dynamic Internet services. In: Proc. of the Int'l Conf. on Dependable Systems and Networks. Bethesda: IEEE, 2002. 595−604.

[30] Jones JA. Fault localization using visualization of test information. In: Proc. of the 26th Int'l Conf. on Software Engineering. Edinburgh: IEEE, 2004. 54−56.

[31] Abreu R, Zoeteweij P, van Gemund AJC. An evaluation of similarity coeffcients for software fault localization. In: Proc. of the 12th Pacific Rim Int'l Symp. on Dependable Computing. Riverside: IEEE Computer Society, 2006. 39−46.

[32] Abreu R, Zoeteweij P, Golsteijn R, van Gemund AJC. A practical evaluation of spectrum-based fault localization. Journal of Systems and Software, 2009,82(11):1780−1792.

[33] Wong WE, Qi Y, Zhao L, Cai KY. Effective fault localization using code coverage. In: Proc. of the 31st Annual Int'l Computer Software and Applications Conf. Beijing: IEEE Computer Society, 2007. 449−456.

[34] Wong WE, Wei T, Qi Y, Zhao L. A crosstab-based statistical method for effective fault localization. In: Proc. of the 1st Int'l Conf. on Software Testing, Verification and Validation. Lillehammer: IEEE, 2008. 42−51.

[35] Wong WE, Debroy V, Li YH, Gao RZ. Software fault localization using dstar (*D**). In: Proc. of the 6th IEEE Int'l Conf. on Software Security and Reliability. Gaithersburg: IEEE Computer Society, 2012. 21−30.

[36] Zhang X, Tallam S, Gupta N, Gupta R. Towards locating execution omission errors. ACM SIGPLAN Conf. on Programming Language Design and Implementation, 2007,42(6):415−424.

[37] Zhang XY, Gupta N, Gupta R. Pruning dynamic slices with confidence. ACM SIGPLAN Conf. on Programming Language Design and Implementation, 2006,41(6):169−180.

[38] Zhang X, Tallam S, Gupta R. Dynamic slicing long running programs through execution fast forwarding. In: Proc. of the 14th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. Portland: DBLP, 2006. 81−91.

[39] Alves EC, Gligoric M, Jagannath V, Damorim M. Fault-localization using dynamic slicing and change impact analysis. In: Proc. of the 26th Int'l Conf. on Automated Software Engineering. Lawrence: IEEE, 2011. 520−523.

[40] Wen W, Li B, Sun X, Li J. Program slicing spectrum-based software fault localization. In: Proc. of the 23rd Int'l Conf. on Software Engineering and Knowledge Engineering. Miami Beach: IEEE Press, 2011. 213−218.

[41] Wong WE, Qi Y. An execution slice and inter-block data dependency-based approach for fault localization. In: Proc. of the Asia-Pacific Software Engineering Conf. Busan: IEEE Computer Society, 2004. 366−373.

[42] Wang X, Cheung SC, Chan WK. Taming coincidental correctness: Coverage refinement with context patterns to improve fault localization. In: Proc. of the Int'l Conf. on Software Engineering. Vancouver: IEEE Computer Society, 2009. 45−55.

[43] Masri W, Assi RA. Cleansing test suites from coincidental correctness to enhance fault-localization. In: Proc. of the Int'l Conf. on Software Testing. Washington: IEEE Computer Society, 2010. 165−174.

[44] Masri W, Assi RA. Prevalence of coincidental correctness and mitigation of its impact on fault localization. ACM Trans. on Software Engineering and Methodology, 2014,23(1):1−28.

[45] Miao Y, Chen ZY, Li SH, Zhao ZH, Zhou YM. A clustering-based strategy to identify coincidental correctness in fault localization. Int'l Journal of Software Engineering and Knowledge Engineering, 2013,23(5):721−741.

[46] Bandyopadhyay A. Mitigating the effect of coincidental correctness in spectrum based fault localization. In: Proc. of the 5th Int'l Conf. on Software Testing, Verification and Validation. Kolkata: IEEE, 2012. 479−482.

[47] Lei Y, Sun CN, Mao XG, Su ZD. How test suites impact fault localisation starting from the size. IET Software, 2018,12(3): 190−205.

张卓(1984－),男,山东蓬莱人,博士生,主要研究领域为程序切片,程序测试,深度学习.

雷晏(1985－),男,博士,副教授,CCF 专业会员,主要研究领域为软件错误定位,软件自动修复.

谭庆平(1965－),男,博士,教授,博士生导师,主要研究领域为软件工程,软件容错技术,系统软件.

常曦(1979－),女,博士,副教授,CCF 专业会员,主要研究领域为程序测试,程序分析.

毛晓光(1970－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为可信软件,软件维护与演化.

薛建新(1980－),男,博士,讲师,CCF 专业会员,主要研究领域为并发理论,程序分析.