一种调和序列生成及其在故障定位中的应用

叶俊民 谢 茜 姜 丽 李嵩嵩 许 磊
(华中师范大学计算机科学系 武汉 430079)

摘 要 为了提高软件的可靠性,在软件运行发生故障时,快速、准确地定位故障点成为非常有意义的研究课题。与以往方法不同,在将故障运行序列和最邻近运行序列进行差异对比前,为了避免选取第一条最近成功路径时引起的"盲区"使得后期搜索空间加大,引入了生物学基因序列比对原理,对最邻近运行序列不是通过编辑距离比较进行选取,而是通过一条调和序列进行处理。实验表明,方法的故障定位效果较好。

关键词 故障定位,序列比对,运行序列,调和序列
中图法分类号 TP311 **文献标识码** A

Generation of a Consensus Sequence and its Applications in the Fault Location

YE Jun-min XIE Qian JIANG Li LI Song-song XU Lei
(Department of Computer Science, Central China Normal University, Wuhan 430079, China)

Abstract In order to improve software reliability, software running in the event of the failure, rapid and accurate positioning of the point of failure has become a very meaningful study. Previous methods, Fault running sequence and the nearest running sequences to run the previous differences, in order to avoid selecting the first path caused by the recent success of recent blind-area exist, the impact of the increasing post-search space, the paper introduced gene biology the principle of alignment of the nearest sequence to run not by editors to select from the comparison, but to processed through a consensus sequence. Experiments show that the method of fault location better.

Keywords Fault location, Sequence alignment, Running sequence, Consensus sequence

1 引言

软件故障定位是目前软件故障诊断的一个热点研究课题,其目的是在软件运行失效或是故障时,可以快速、准确地对失效区域或是故障点进行定位。在目前众多的故障定位方法中,例如程序的切片/削片技术[1]、算法调试技术[2]、利用测试信息可视化[3]以及"近邻模型"思想[4]等,以"近邻模型"定位效果较好[5]。文献[6]提出了一种双轨迹差异分析法:按照一定的距离准则(Distance Criterion),选取一条和故障的运行序列相似的正确的运行序列,通过比较二者的差异,产生程序中的可疑部分。

然而,文献[6]采用基于执行路径之间的差异来辅助进行故障定位时,存在以下问题:在最近成功执行路径进行选取时,只是选择了成功运行轨迹库中搜索到的第一条成功路径,这会导致存在高扇入节点,增大了后期代码的搜索空间。而且故障节点不一定包含在将这第一条成功路径和故障序列进行差异比较后的结果中,我们将这种差异比较结果称为"盲区"。然而,事实上最近成功执行路径通常不止一条。文献[5]利用"替换无约束边一填补不连续"思想,通过对TCAS(飞行器碰撞侦测与避让决策系统)进行试验发现,产生的相似成功路径会存在多条。文献[9]在利用界限模型进行错误定位时也提到,反例路径的最近例证并不唯一,并且真正错误点也不一定从属于寻找到的第一个最近证例。文献[7]采用后期进行广度优先代码检查来避免"盲区"出现。对此,本文针对寻找到的最近的正确运行序列不止一条的问题,引入了生物学基因序列比对原理,提出了一种调和序列生成方法,并将其应用于故障定位中。

2 研究基础

2.1 序列距离度量

在通过一系列的测试用例提取生成相应的正确运行序列和一条对应的故障运行序列后,参考"近邻模型"思想,为了选取和故障运行序列最相近的正确运行序列,我们需要对运行序列进行相似性度量。通常用距离来表示两个序列之间的相似程度,距离越大相似性越差,距离越小相似性越好。为此,我们采用文献[15]中提到的一种基于离散性、交叉性、非完全性的字符串匹配方法。该方法首先定义了文本 T 和模式 P 之间的匹配关系,并分析了它们之间的匹配关系:

定义1 对给定的文本 T 与模式 P,它们的匹配关系存在3种映射:

到稿日期:2010-04-06 返修日期:2010-07-20 本文受武汉大学计算机软件工程国家重点实验室开放基金项目(SKLSE20080705),湖北省自然科学基金(2007ABA034,2008CDB349),华中师范大学中央高校基本科研业务费项目(CCNU09Y01009和CCNU09Y01013)资助。
叶俊民(1965-),男,博士,教授,主要研究方向为高可信软件工程、软件质量保障,E-mail:jmye@mail.ccnu.edu.cn;谢 茜 硕士生,主要研究方向为软件工程。

① $p_i \to t_j$ ($1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$)，表示 $p_i = t_j$ 并匹配；

② $p_i \to \Lambda$ ($1 \leqslant i \leqslant m$)，表示 p_i 未匹配；

③ $\Lambda \to t_j$ (min$\leqslant j \leqslant$max)，表示 t_j 未被匹配。

其中，p_i，t_j 最多匹配一次，Λ 表示为空，min 为文本中被匹配的最小位置，max 为文本中被匹配的最大位置。匹配关系反映了模式与文本的一次匹配中相关范围的字符映射关系。将模式 P 与文本 T 中满足上述 3 种映射关系的映射关系个数分别定义为匹配关系的离散数、非完全数和交叉数。

定义 2 给定模式 P 与文本 T，并设模式 P 的长度为 m，匹配关系 R 为模式 P 与文本 T 的一次匹配，其实际离散数为 d，实际交叉数为 c，实际非完全数为 n'，匹配关系 R 即模式 P 与文本 T 的相似度就定义为 Approximate(R)$=(2m-2n'-c)/(2m+d-n')$。其中，分母$(2m+d-n')$代表了匹配关系中涉及的字符总数，而分子$(2m-2n'-c)$代表了匹配关系 R 中，形如 $p_i \to t_j$ 的映射关系涉及的字符数，并减去交叉数。

定义 3 稳定序列段集 SS(Stable Set)是指在得到的集合 N 中满足以下条件之一的序列段集：

1)$N_i = N_j (i \neq j)$，即 N_i 在集合 N 中重复出现；

2)N_i 去掉首字母后的 $N_i' \subseteq SS$ 中某一序列段，即 N_i' 是 SS 中某序列段的子集；

3)N_i 去掉尾字母后的 $N_i' \subseteq SS$ 中某一序列段，即 N_i' 是 SS 中某序列段的子集；

4)N_i 去掉首尾字母后得到的序列段。

2.2 调和序列

在生物基因学中，一个家族的调和序列(consensus sequence)就是一个捕获家族多数成员的共同特征的序列[11]。通过调和序列可以很方便地确定一个基因家族的特征，也可以很方便地总结出一个序列家族中的共同元素。求解调和序列的方法有以下几种：第一种是在序列族多序列比对的基础上，根据每一列对比结果，通过概率权值挑出共同的元素[16]；第二种方法是渐进比对方法[16]；第三种方法是基于模拟退火算法找调和序列[12]。

3 基于多运行序列比对的调和序列生成

目前，生物学上的比对分为两类[11]：一类是双序列比对，是指通过一定算法对两个 DNA 或蛋白质序列进行比较，找出两者之间最大相似性匹配；另一类是多序列比对，有时用来区分一组序列之间的差异，但主要用于描述一组序列之间的相似性关系，揭示整个基因家族的特征，其比对结果的一种表现方法就为调和序列。该调和序列如本文 2.2 节描述，是一条捕获了该家族多数成员的共同特征的序列。

采用“近邻模型”思想进行故障定位中，通常选取的最近正确运行序列不止一条。如果选取其中某一条进行差异分析，在后期代码检查过程中，就会存在高扇入节点，极大地增大了后期代码搜索空间的情况出现[9]；而将多条都与故障序列进行差异分析，加大了后期代码分析难度。由于软件运行序列表示与生物基因序列之间存在着很大的相似性，借鉴生物学中的多序列比对后生成调和序列的方法，提取这 n 个最近的正确运行序列的共同特征，生成一个调和序列。该调和序列包含了正确运行序列的绝大多数成员的共同特征，再进行后续的故障定位，就减少了“盲区”的存在，降低了后期代码搜索空间。

3.1 调和序列问题定义

为此我们参照生物学多序列比对问题描述，将 n 个最近的正确运行序列生成调和序列(Generate Consensus Sequence，GCS)的过程描述如下：

$$GCS=(Q,S,C,O,F)$$

式中，1)$Q=Q_0 \cup \{\Lambda\}$，Q_0 为 n 个最近的正确运行序列的基本块字符表，“Λ”为空位符；$|Q|$ 为基本块字符表中不同字符的个数；2)S 为 n 个最近的正确运行序列的全集；$S=\{S_i \mid i=1,2,\cdots,n\}$，其中 $S_i=(S_{i1},S_{i2},\cdots,S_{im})$。$m$ 为序列 S_i 的长度，S_{ij} 为序列 S_i 中的第 j 个字符；3)$C=(S_{ij})$ 为调和序列，其中 $S_{ij} \in Q$，但不一定为 S 中的某个序列，它是一条与 n 个最近的正确运行序列集具有最大比对得分之和的序列；4)O 为基本比对操作集，在生物学上可以认为 $O=\{$insert，delete，replace$\}$，即插入、删除和替换操作；5)F 为寻找调和序列的算法。

GCS 过程与生物多序列比对过程(Process of Multiple Sequence Alignment，PMSA)的区别在于：(1)Q_0 的定义不同。在 PMSA 中其被定义为基因序列的不同生物分子；在 GCS 中我们将软件运行描述为“基本块”[10]的形式，这里 Q_0 就定义为寻找到的 n 个最近的正确运行序列的基本块组成字符表。(2)C 的对象不同。在 PMSA 中 C 为该生物比对家族的调和序列；在 GCS 中 C 为前期寻找到的 n 个最近的正确运行序列的调和序列，目的是为了避免故障定位后期，即代码的检查过程中，更多地对“盲区”进行检查，从而降低后期代码搜索空间，提高定位效率。(3)算法 F 不同。在 PMSA 中多是通过编辑距离比较后进行相关归并操作，而采用编辑距离进行字符串匹配具有多态性，会影响匹配质量；但在本算法中采用的序列距离度量，效率优于编辑距离度量，尽量减少了字符串匹配时的多态性。

3.2 算法描述

GCS 过程中算法 F，即寻找调和序列的基本流程可以表述为图 1。该过程大致可以分为 4 步进行。

第一步　从所有正确运行序列中选出 n 条最近正确运行序列。我们采用文献[15]中的基于离散数、交叉数和非完全数的度量方法，将每条成功运行序列分别与失效运行序列进行相似度度量，并计算相应的离散数、交叉数和非完全数，从而得出该成功运行序列与失效运行序列的相似度。根据相似度的取值大小来选取这多条最接近成功运行序列，相似度的取值越大，该成功运行序列与失效运行序列越近。

第二步　对多条最近正确运行序列进行比对，分别计算每条最近正确运行序列与余下的$n-1$条正确运行序列的相似度的总得分，选取 min$[Score(S_i),1 \leqslant i \leqslant n]$(即总得分最小)的序列为候选序列。

第三步　以寻找到的候选序列为固定序列，将余下的$n-1$条最近正确运行序列逐次地归并入候选序列。归并操作采用 N-W(Needleman-Wunsch) 算法。N-W 算法是基于动态规划思想的全局序列比对的基本算法，具体过程参见文献[14]。我们这里利用 N-W 算法，目的是在归并 $S_j \in S-\{S'\}$ 到 M 中时保持最多匹配。

第四步　在完成余下的 $n-1$ 条最近正确运行序列归并

后，结合本文定义的选择规则，选出每一列的候选字符，进而组成调和序列。

每一列的候选字符选择规则如下。

规则 1　如果该列只出现过一个字符 A，概率为 100%，即该列对应的每一行都是该字符 A，那么候选字符就为此字符。

规则 2　如果该列出现两个以上字符，并且无“∧”，选择出现概率大的字符；如果概率相同，随机选择一个作为候选字符。

规则 3　如果该列出现两个以上字符，且其中存在“∧”，选择“∧”为候选字符。此时要保证尽可能降低误选出错。

该流程的算法描述如图 2 所示。

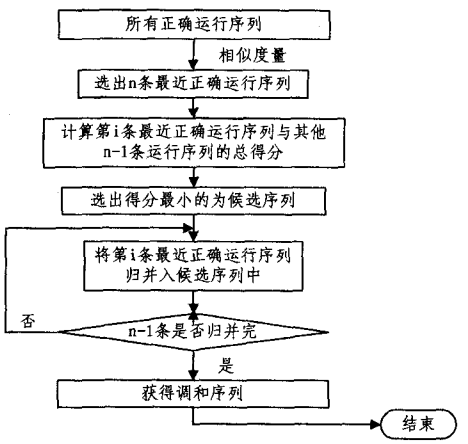

图 1　寻找调和序列的基本流程

生成调和序列算法 *F*

输入：正确运行序列集合 A 和故障序列 H

输出：调和序列 C

Begin

Step1　通过 2.1 节中的序列距离度量，从集合 A 中寻找 n 个最近的正确运行序列的全集 S；

Step2　判断最近的正确运行序列的条数，if($n=2$) 跳转执行 Step4，否则依次执行；

Step3　对最近正确运行序列集 S 中的每条序列 S_i 分别与 S 中其他运行序列 S_j ($i\neq j$) 进行相似度度量 similarity(S_i, S_j)；

Step4　计算序列 S_i 的相似度总分 $\text{Score}(S_i)=\sum_{j=1,j\neq i}^{n}\text{similarity}(S_i,S_j)$；

Step5　if(($n=2$)选取 S_1 为候选序列，即 $S'=S_1$；否则选取总分最小的序列为候选序列，即 $S'=\min[\text{Score}(S_i),1\leqslant i\leqslant n]$；

Step6　初始化比对列表 $M=\{S',\}$ 以 S' 为候选序列，将其他序列通过算法 $N-W$ 归并到 M 中，进行比对；

Step7　计算比对列表 M 中每个字符出现的概率，结合选择规则，选出每一列的候选字符，组成调和序列 C；

Step8　输出调和序列 C；

End

图 2　生成调和序列算法 F

现分析寻找调和序列算法 F 的复杂度。F 算法的运行时间主要取决于 Step2 中序列进行相似度度量和 Step5 中 N-W 算法的计算时间。如果最近的正确运行序列的全集 S 中有 n 条序列，长度平均值为 m，那么相似度度量的时间复杂度为 $O(m*n)$，归并操作需要进行 $n-1$ 次，时间复杂度为 $O(m^2)$。所以，算法 F 总的运行时间为 $O(m*n+m^2)$。

通过上述方法生成的调和序列包含了正确运行序列的绝大多数成员的共同特征，相比只在成功运行轨迹库中搜索到的第一条成功路径，所包括的信息量更多。再进行后续的故障定位，就减少了“盲区”，降低了后期代码搜索空间。

4　基于调和序列的故障定位

软件运行的形式化我们可以采用“基本块”进行描述。基本块”[10]可以是程序模块、代码行或是分支语句。具体采取什么形式，视不同的情况而定。因此软件的一次执行过程可以看成是这些“基本块”按照不同的执行顺序完成的一次执行序列。软件的运行过程也就是对这些基本块的调用过程。不妨设一个软件系统可以划分为 k 个“基本块”($k=\{k_i\mid i=1,2,\cdots,n\}$)，那么软件从相应的输入到对应的输出的一次执行就可以表示为

$$k_i\rightarrow k_{i+1}\rightarrow k_{i+2}\rightarrow\cdots\rightarrow k_{i+m}\ (i,m\in N)$$

这样，在软件运行时，“基本块”的调用序列就构成了软件的运行序列，它准确地反映了软件实际的运行过程。如果用顺序编号对每个基本块进行唯一标识，那么软件运行序列从形式上表现为一个由编号排列而成的字符串或是字符序列，从而程序运行的相似性可以转换为字符串序列间的相似性度量问题。

针对文献[6]采用基于执行路径之间的差异来辅助定位故障时，只选择了成功运行轨迹库中搜索到的第一条最近正确运行序列，导致了“盲区”现象，本文引入了生物学基因序列比对原理，亦即将多条最近正确运行序列进行多序列比对，生成调和序列，然后进行差异比较。此方法的具体流程如图 3 所示。

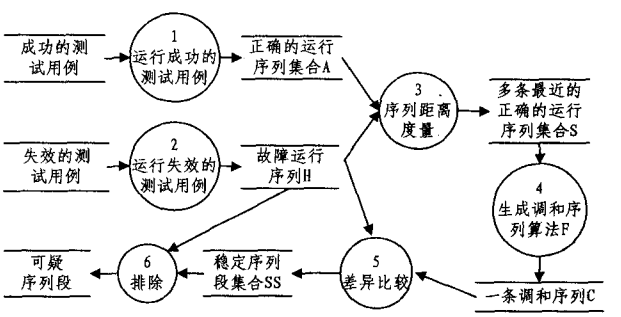

图 3　多运行序列比对故障定位流程

过程的具体描述：软件运行序列获取分为 1 和 2 两个步骤，通过软件的正确运行和错误运行，分别设计对应的成功测试用例和失效测试用例。在步骤 1，运行多个成功测试用例，从而得到相应的正确运行序列集合 A；步骤 2，运行失效测试用例，从而得到相应的故障运行序列 H，通常针对某类故障，只生成一条故障序列；步骤 3，在得到正确运行序列集合和故障序列的情况下，采用本文 2.2 节中提到的基于离散性、交叉性、非完全性的字符串匹配方法进行序列之间的差异度量，即序列距离度量，寻找到多条最近正确运行序列集合 S；步骤 4，按照本文第 3 节中提到的将多条最近正确运行序列集合提取共同特征生成调和序列的 GCS 过程，调用其中的算法 F，生成一条调和序列 C；步骤 5，调和序列和故障运行序列采用 N-

W算法进行二次差异比对后，以“∧”为标记断开得到的序列段集合为 N，其中 $N=\{N_i \mid i=1,2,\cdots,m\}$，$m$ 为序列段个数，按照本文2.1节中稳定序列段集的定义，生成稳定序列段集SS；步骤6，采用排除法求解可疑序列段 SSS，其中 SSS=H−SS，即从故障序列中除去稳定序列段集中的相关内容，得到的就是故障序列中的可疑序列段。

5 实验验证及其分析

本文以比较两个有理分数的大小程序进行实验。首先将源程序进行“基本块”的形式抽象，按照一定的序列度量准则，选取和故障运行序列最接近的正确运行序列，以例后续通过差异比对来定位程序中的可疑故障位置。其中，故障运行序列和正确运行序列可以通过设计对应的失效测试用例和成功测试用例，再将其返回软件程序源代码运行来获得，然后按照本文第4节中的基于调和序列的故障定位方法进行试验。为了验证本方法的有效性，我们将本实验结果与文献[6]的双轨迹差异分析法实验结果进行比较。实验结果采用最终获取的可疑序列段长度和包含故障点的概率(故障定位精度)两个方面作为评价指标。

实验1　通过设计相应的成功测试用例和失效测试用例来获取长度不一的相对应的正确运行序列和故障运行序列，序列的长度可由测试用例的输入数据量决定。运用文献[6]和本文方法，分别运行了5次，最终获取可疑序列段长度的比较情况。

实验2　分析最终包含故障点的概率。这里我们根据故障定位结果中包含故障点的“基本块”个数 N 与可疑序列段长度 Num 之间的关系定义评价函数 $\text{Accuracy}=\frac{N}{Num}\times 100\%$，来评价故障定位精度。评价函数 Accuracy 对即包含了故障点“基本块”，同时可疑序列段长度短的故障定位结果将给出较高分数。

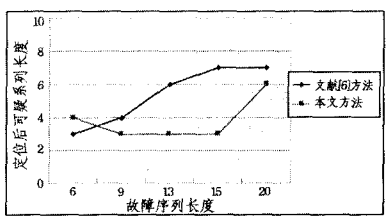

图4　两种方法定位后获得的可疑序列段长度比较

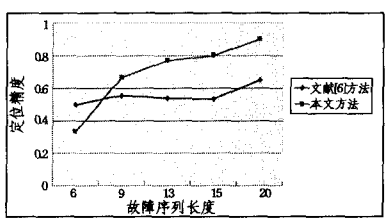

图5　两种方法的故障定位精度比较

从图4和图5可以看出，在故障序列长度一定的情况下，本文的调和序列进行故障定位方法与文献[6]相比，在定位后的可疑序列段的获取长度上和定位精度上均较好，可疑序列段的程度越短，后期的代码搜索空间就越小；在较短的可疑序列段的基础上，定位精度也较高。

结束语　本文引入生物学基因序列比对原理，在进行故障序列和成功序列的差异比对前，生成调和序列后进行差异比对，保留了序列的多数特征，再进行后续的故障定位，减少了“盲区”存在，降低了后期代码搜索空间。通过实验比较，本文方法排除了更多不产生故障的部分，缩小了源代码审查，提高了故障定位的效率。

参考文献

[1] Hiralal A, Joseph R H, Saul L, et al. Fault localization using execution slices and dataflow tests[C]//Proceedings of 6th International Symposium on Software Reliability Engineering. Toulouse: IEEE Computer Society, 1997: 143-171

[2] Ehud Y S. Algorithmic program debugging[M]. The ACM Distinguished Dissertation Series. Cambridge: The MIT Press, 1983

[3] James A J. Fault localization using visualization of test information[C]//Proceedings of the 26th International Conference on Software Engineering (ICSE 2004). Edinburgh, Scotland: IEEE Computer Society, 2004: 74-76

[4] Manos R, Steven P R. Fault localization with nearest neighbor queries[C]//Proceedings of 18th IEEE International Conference on Automated Software Engineering (ASE'03). Montreal: IEEE Computer Society, 2003: 30-39

[5] 朱凯. 一种基于相似路径集生成的程序故障定位方法[D]. 武汉：华中师范大学，2007：3

[6] 刘彦斌，朱小冬. 基于双轨迹差异分析法的软件故障定位[J]. 计算机工程，2007，9(33)

[7] Groce A. Error Explanation with Distance Metrics[C]//Tenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04). Barcelona, Spain, Lecture Notes in Computer Science, vol 2988. 2004: 108-122

[8] Mount D W. Bioinformatics: sequence and genome analysis[M]. USA: Cold Spring Harbor Laboratory Press, 2002: 53-54

[9] 沈胜宇. 模型检验的反例解释[D]. 长沙：国防科学技术大学，2005：17

[10] 王毅刚，朱小冬. 基于运行序列的软件故障诊断方法[J]. 微计算机信息，2006，7(22)

[11] 明华. 基于模拟退火的多序列比对算法的研究[D]. 西安：西安电子科技大学，2006

[12] Keith J M, Adams P, Bryant D, et al. A simulated annealing algorithm for finding consensus sequences[J]. Bioinformatics, 2002, 18(11): 1494-1499

[13] Sankoff D. The early introduction of dynamic programming into computational biology[J]. Bioinfromatics, 2000, 16(1): 41-47

[14] Hirschberg D S. A linear space algorithm for computing maximal common subsequences[J]. Communications of the ACM, 1975, 18(6): 341-343

[15] 丁光耀. 基于离散性、交叉性、非完全性的字符串匹配方法[P]. 中国专利，申请号：200610021787.5，2006-09