ConSeq

夏亦谦

一、背景：

Concurrency bugs 通常都是通过分析源码来发现潜在的 Bugs，主要的方法是发现代码中的 Bug Patterns（指根据实验发现的 Bug Pattern）。但是这一类方法有如下局限性：

1) 漏报（False negatives），由于这类方法是利用已知的 Bug Patterns 来寻找 Concurrency Bug，因此是否能找的更多的 Bug 取决于 Bug Patterns 是否更多；
2) 误报（False positives），传统的方法只是根据 Bug Patterns 找到了可能的 Concurrency Bug，至于这些 Bug 是否会造成系统的 Crash 并不能给出确定的结论（2-10%的 Bug Patterns 会造成 failure），因此会存在误报的可能性；
3) 用户不友好 。

另外，文章提出一个 Bug 的生命周期如下：

1) triggering，某个操作出发 bug，使得某个数据错了；
2) propogation，错误的数据被传播出去；
3) failure，当某个操作使用这个数据时，导致系统 Crash

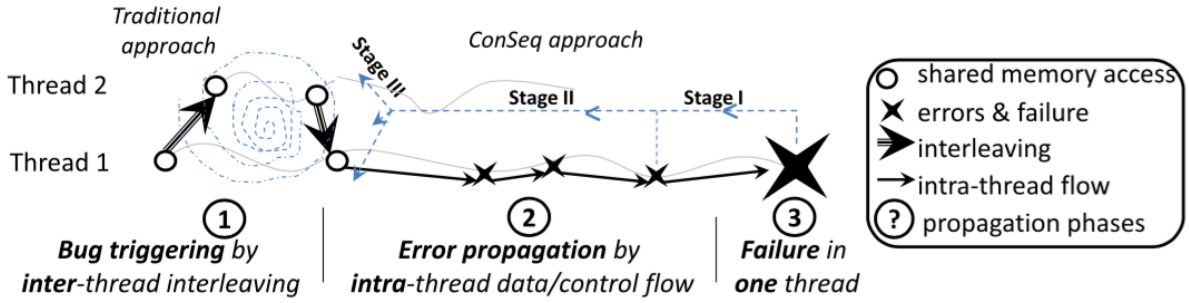

Figure 2. The common three-phase error-propagation process for most concurrency bugs.

基于以上背景，ConSeq 提出了从发现系统错误（failure）入手的一种反向分析的方法，failure->propogation->triggering，来发现 Concurrency bugs。这一方法的依据在于作者对于 Concurrency bugs 的传播特点做了分析，结论是：

1) 59/70 的非死锁 Bugs 的传播距离是很短的（后文中作者用了 4）；
2) 66/70 的 bugs 导致的 crash 是出现在一个线程中的（也就是说系统 crash 基本是 Sequential error，这也是标题 detecting concurrency bugs through sequential errors 的依据）；

3) failure patterns of concurrency bugs 主要有五种：assertion-violation，error-massage（eg: printf、log 函数），incorrect outputs，infinite loop，memory-bugs，占了 65/70；并且和 Sequential failure 非常像。

二、ConSeq 的介绍：

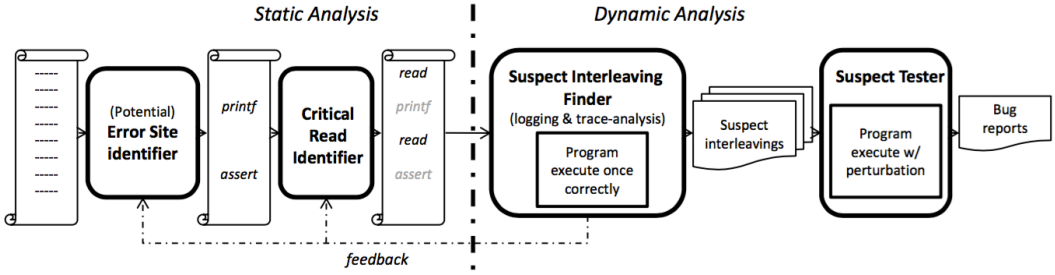

Figure 4. An overview of the ConSeq architecture.

ConSeq 主要通过以下几个步骤来发现 concurrency bugs：

1) Error-site identifier，通过对 binary 的静态分析，找出五类可能的 failure 的位置，通过一些特征值来发现，比如 assert，printf，fprintf，stderr 等，还有一些工具，SCC，Daikon（用于发现 implicit errors）；
2) Critical-read identifier，用静态分析找到影响 error site 的 critical-read（short propogration distance）；
3) Suspicious-interleaving finder，动态分析，运行一次 program，得到 trace 以及 control/data 依赖图，通过 trace 和依赖图来找到可能的 correlation variables；再对它们进行分析，排除掉不可能的，剩下的就是可能性较大的 bug；

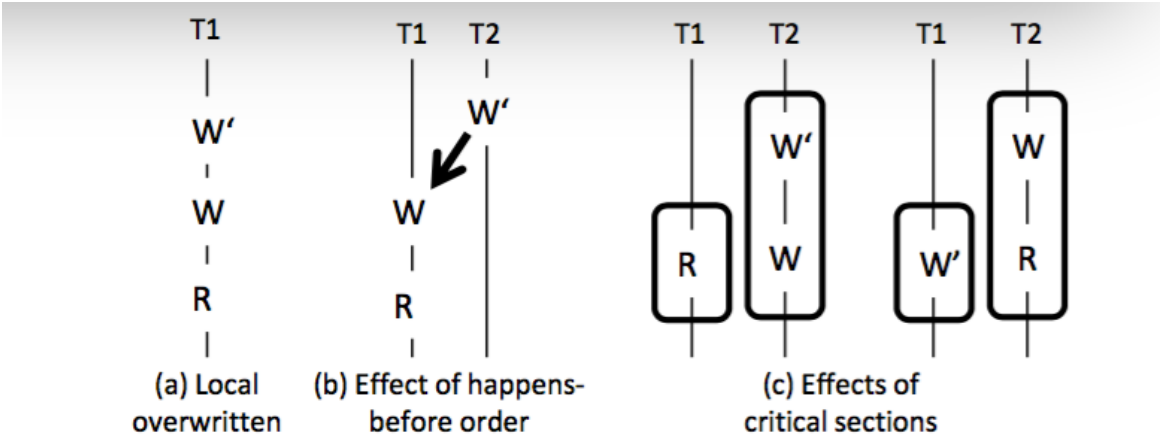

Figure 6. A value written by W' may never reach R

4) 4. Suspicious-interleaving tester，动态分析，找到那些真正会导致的 failure 的 bug。

三、ConSeq VS MUVI

1) MUVI 基于源码分析，ConSeq 基于 Binaries；
2) MUVI 正向，找到源码中可能的 bugs，但并不确定这些 bug 会导致 failure，同时漏报的可能性大；ConSeq 反向，从 failure 入手，找到的 Bugs 肯定会导致 failure，同时漏报的可能性小（原因在于 failure 的覆盖度更大）；

ConSeq 的不足之处：

1) 由于存在动态分析，因此需要依赖输入，来保证高的代码覆盖率，以保证找到更多的 bugs;
2) 目前对于 failure 只是关注五种，虽然基本覆盖了 90%的 failure，但还是有提高的空间；
3) 由于在方法中用到了传播路径短这一前提条件，因此对那些有较长传播路径的 bugs 是无法发现的；
4) 另外 ConSeq 主要对那些 failure 和 critical-read 发生在同一个线程的情况有效。