

**FAKULTI SISTEM KOMPUTER & KEJURUTERAAN PERISIAN**

CHESS MODEL CHECKER

**BCS2213 FORMAL METHODS**

**SEM II 2013/2014**

**LECTURER: PROF DR. VITALIY MEZHUYEV**

**ELVIRA ARIAPUTRA**

**CB13125**

**Section 01A  
11th March 2014**

**Abstract**

Formal method is defined as mathematical modelling techniques for specification, development and verification of software and hardware systems. The behavior of creating errors in the systems requires formal method or model checking tools to lower the percentage of bigger risk and damage. Model checking might prove that the system is error free by verify the correctness of a finite state concurrent system.

This paper presents an easy-to-understand introduction on one of the formal method model checking tools, **CHESS**. It is a software model checker that follows a systematic exploration of thread schedules in order to find errors or Heisenbugs in multithreaded software. Under Microsoft research, CHESS aims to find and reproduce Heisenbugs in concurrent programs.

In this paper also aims to brings a brief understanding towards another formal methods tools in model checking tool; CHESS. The method and implementation aspects has proven that CHESS is a brilliant useful software that acts like real-life chess game movements.

**Introduction**

To venture deeper into formal methods: model checking, I feel that CHESS would help me understand the testing process of bugs in a concurrent programs. There are two attractive features that make me choose CHESS as my topic;

1. It **finds errors**, such as data-races, deadlocks, livelocks, and data-corruption included access violations which are **extremely difficult to find** with the current testing tools.
2. When it detects an error, it **provides a full repeatable execution of the program directing or leading to the error**, meaning focusing on the roots of error by the whole program which is a big advantage aiding the debugging process.

CHESS, a Microsoft Research tool works best for concurrent programs that are mostly used in large systems programs. These concurrent programming is tough because of the possibility of unexpected interference in concurrently executing tasks. Interference like these results in “Heisenbugs” that are rare and extremely difficult to reproduce and debug.

**Literature review**

Regarding to formal method and system testing, it appears that most of the systems have hidden errors and bugs. Carson Jones (2009) investigates CHESS as a verification tool in finding and reproducing Heisenbugs which is a class of bugs that are rare and difficult to reproduce. He uses CHESS integrated environment with Microsoft Visual Studio.NET and comments that the interface is really nice and usable. As he tested the software, he noticed that CHESS needs a bit of suggestions on where to preempt threads because the software edition he tried only preempts a thread twice throughout an execution and uncertain on which variables are being shared among threads. He also gave a good feedback on how the system notifies (state the schedule string to reproduce the error without waiting for all previous schedules) to the user when there is an error. Not only that, Carson Jones feel that this system is very useful when the system detects deadlock or livelock errors on to his codes where he assume he coded correctly. Plus, CHESS explored all thread schedules and replay, a good feature for debugging and finding data races and deadlocks. Despite on all the good points, there were some weakness onto the system, he noticed that some test were inconclusive, some races were undetected, difficult visibility into deadlocks and long running-time. However, these studies were based on the older edition of CHESS without data race detection, there might be improvements on the system in future.

Meanwhile, Rahul and Boby (2008) examines CHESS as a concurrency testing tools. They mention that from the testing view point, user should begin running CHESS with an iteration context bound of 2. If there is no error or bugs found, increase the context bound to 3 and so on. This is because it is an easier way to detect the bugs sooner. CHESS runs a regular unit test repeatedly on a a specialized scheduler which is good as it chooses a different scheduling order on every repetition. They too gave a feedback that CHESS is beneficial for the developers and testers who had to solely rely on stress for better interleaved testing.

During an interview session with Scott Hanselman at PDC Conference, Shaz Qadeer (2008) states that CHESS makes it easier in finding and reproducing in systematic bug known as Heisenbugs. This is because they have identified the main cause of these bugs is concurrency that the multithreaded application running and they’re relative timing causes problems. CHESS works systematically generate all possible orders of thread events in the application and when it finds a particular execution order that causes a bug, it will remember it and the next time the application run, it can repro that. From the interview, Scott Hanselman understands that why this is called CHESS because they’re making many moves ahead to see what’s going to happen and then make the right move for the future. In short, CHESS is a brilliant software model checker to detect rare and common bugs in concurrent application.

**How it is works?**

The concurrency testing today is equivalent to stress testing but stress testing is not systematic, not predictable, and makes any error found hard to debug.

Concurrent Program

CHESS replace the OS scheduler with a demonic scheduler  
and systematically explore all scheduling choices.

Win32 API

It will also explore the thread in different interleaving by  
intercepting system calls in order to systematically explore the  
space of schedules.  
  
CHESS has an innovative search order called *iterative context bounding* where the tool explores executions with at most k context-switches, where k is a parameter that is iteratively increased. The purpose for this search ordering is that many bugs are inspected in long executions containing just a few unexpected context-switches.

Demonic

Scheduler

For a multithreaded application, CHESS will run system unit test in a loop and CHESS inserts the user mode scheduler between application and the underlying runtime. The user mode scheduler will intercepts calls made to the underlying threading and this allows CHESS to hijack the scheduling and control it. The next time CHESS runs the application in a loop it repro the bug.

CHESS also prioritizes among the schedules that if the ones with the errors generates, those schedules that are more likely to yield bugs.

If CHESS has a parameter that is okay in preemptions of maybe two, then CHESS will systematically generate schedules in which there are no more preemptions than two. So it investigates all possible place for preempting except places of not more than two preemptions and this is very useful because a lot of concurrency bugs that found, it just require one or two preemptions and it have found bugs in large applications using this method.

**Sample/Implementation**

For implementation on own user experience, install CHESS onto your Visual Studio team test and set a whole site attribute called CHESS to your unit test.

http://d.adroll.com/cm/w/outCHESS has been tested out in large concurrent program, the results many unknown bugs is found in the system that stress testing could not detect for months. For example, CHESS discovered a bug in PLINQ (Parallel LINQ) which caused by improper use of LiteEvents, a concurrency primitive implemented in the library.

Another example, Singularity operating system was successfully booted under the control of CHESS.

**Conclusion**

Software testing and finding bugs can be difficult as errors and mistakes are not visible till it is tested. The subject formal method is essential to all software and hardware as it is proven on a mathematical modeling techniques. In summary, these paper discussed CHESS, a Microsoft research project as a software model checker in checking concurrency-related errors in multithreaded system. The capabilities of CHESS to detect rare and reproduce bugs such as Heisenbugs in concurrent programs is a big achievement for a model checking tools. With a systematic scheduling way, CHESS are able to track bugs in a multithreading by iterative context bounding. Over the years, CHESS has matured as an efficient platform for building analysis tools for concurrency. With this checking tool usage, CHESS increase the confidence level of performed system after being tested. In sum, there is no need of worries on stress testing or other testing when there is CHESS testing.

**References**

[1] Carson Jones. C# Threading Verification with CHESS, CS6966 Project. 5th July 2009. Article. <http://www.cs.utah.edu/~carsonj/cs6966/Project/html/finalmain.html>

[2] Rahul V.Patil, Boby George. Concurrency: Tools and Techniques to Identify Concurrency Issues. MSDN Magazine. June 2008. Article. <http://msdn.microsoft.com/en-us/magazine/cc546569.aspx>

[3] An interview session with Shaz Qadeer, one of the project member of CHESS by Scott Hanselman at PDC Conference. 30 October 2008. Audio. <http://www.hanselminutes.com/136/msrpdc-microsoft-research-at-the-professional-developers-conference>

[4] ACM SIGPLAN 2009 Conference on Programming Language Design and Implementation (PLDI'09). Tentative. <http://www.cs.virginia.edu/kim/publicity/pldi09tutorials/#A1>

[5] Madanlal Musuvathi. CHESS: Systematic Concurrency. Powerpoint. [www.research.ibm.com/haifa/Workshops/.../musuvathi.ppt](http://www.research.ibm.com/haifa/Workshops/.../musuvathi.ppt)

[6] Madanlal Musuvathi, Shaz Qadeer. Fair Stateless Model Checking - Microsoft Research. PDF. <http://research.microsoft.com/en-us/projects/chess/pldi08-FairStatelessModelChecking.pdf>.

[7] Ranjit Jhala, Rupak Majumdar. Software Model Checking. October 2009. PDF. <http://www.cs.colostate.edu/~france/CS614/Readings/Readings2011/Model-checking-Survey-a21-jhala.pdf>.

[8] CHESS model checker. Wikipedia. <http://en.wikipedia.org/wiki/CHESS_model_checker>

[9] CHESS: Find and Reproduce Heisenbugs in Concurrent Programs. Microsoft Research. <http://research.microsoft.com/en-us/projects/CHESS/>

[10] Micheal Bentley. CHESS Index. 2009. <http://www.projectbentley.com/work/chess/#microsoft>

[11] CHESS: A Systematic Testing Tool for Concurrent Software (2007). CiteSeerx. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.168.2329>

[12] Stateless model checking (2008). CiteSeerx. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.153.3680>

[13] Microsoft Research Releases Testing Tool for Multithreaded Apps. Dr Dobb’s M-Dev. 12th November 2008. Article. <http://www.drdobbs.com/windows/microsoft-research-releases-testing-tool/212002025>