Towards validating software transactional memory libraries

Martin Nowack (martin@se.inf.tu-dresden.de);
Christof Fetzer
TU Dresden, Germany

WTM 2014 @ Eurosyst
Motivation

• **Goal:** Provide easy-to-use synchronisation mechanism for Joe Programmer
  ➔ Ease reasoning by providing illusion of atomic behaviour of code sections

• Development of (S)TM algorithms and libraries are a daunting task

```c
void parallel_executed_function() {
    atomic {
        // shared memory access I
        ...
    }
    atomic {
        // shared memory access II
        ...
    }
}
```
How to validate?

Model checking [1,2]

→ Abstract but systematic

Software Testing
→ unit test [3]
→ stress test [3]

→ Real Code

How to Validate Implementations?

• How to test TM?
Test different combinations of that protocol

• Two types of bugs [1]:

  • Bohr: sequentially executed code trigger bugs in the general case
  
  • Heisen: hard to trigger, e.g. due to concurrency

unsigned char \( x = 5; \)

1. unsigned char \( \text{foo}(\text{unsigned char } i) \) {
2. if \((i > x)\) {
3. // true branch
4. \(i++;\)
5. } else {
6. // false branch
7. \(i = i + 2;\)
8. }
9. \}
10. assert\((i != 0)\);
11. return \(i\);
12. }

Symbolic Execution\[1,2\]

1. \(\text{foo}(i)\) with \(i = 9\)
2. \((i > x)\) with \(i = 9\)
3. \((i > x)\) with \(i = 255\)
4. \((i > x)\) with \(i = 2\)
5. \((i != 0)\) with \(i = 255\)
6. \((i != 0)\) with no solution
7. \((?+2)\) with \(i = 2\)
8. \((?+2)\) with \(i = 9\)
9. \((?+2)\) with \(i = 255\)
10. \((?+2)\) with \(i = 2\)

Symbolic Execution for Multithreaded Application

- Generalized Symbolic Execution [1]:
  - Symbolic Execution
  - Extended with Model Checker
  - Check all possible interleavings
    - interleaving only on lock operation
  - Partial-order reduction [2]

Symbolic Execution for Multithreaded Application
Our Approach I

- Insight: We just want to test the library calls and their invocation
  - library calls contain small and finite number of steps
  - library calls are typically symmetric across threads
- Goals:
  First, we go for coverage;
  Second, we go for TM correctness
Symbolic Execution for Multithreaded Application

Our Approach II

• start with all interleavings of sequential execution of threads without symbols
  • only one path is taken through the code
  • register read and write operations to shared data structures per thread

• re-execute function under observation with values symbolised restricted to the concrete values observed
  • new code paths are taken
  • interferences between threads are tract and incorporated (e.g. breaking busy loops)
Current Prototype

• Extension of Symbolic Execution Engine KLEE [1] (which is based on LLVM [2] Compiler framework 3.4)

• exercises TM ABI used by TM compiler (GCC, ICC, DTMC)

• partial support for inline assembler (e.g. support libatomic_ops on x86)

• multithreading extension

[1] “KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs” Cadar et al., OSDI, 2008
Results for TM TinySTM++

- Bugs we found:
  - Timing issue in Txn-Initialisation per thread -> wrongly protected access might lead to lost update operation
  - Bug in read operation returning wrong value with write operation by other thread

- Modifications we made to TinySTM++ [1]
  - replace assembler-based setjmp/longjmp version with C-code based solution

WIP/ Future Work

- Exercising different protocol operations more thoroughly - in an automated way
- Independent execution allows to test on cluster of machines
Summary & Outlook

• Developed tool based on symbolic execution which allows to exercise TM libraries in a systematic way.
Error: memory error: out of bound pointer
File: /home/martin/tinystmplusplus/src/transaction.h: 82
Stack:
#0 00007638 in tinystm::TransactionBase::getPublicState (this=0) at src/transaction.h:82
#1 00009090 in tinystm::TransactionBaseSTMBound<tinystm::wt::STMTraits>::isActive (this=0) at src/transaction.h:181
#2 00009020 in tinystm::TransactionBaseSTMBound<tinystm::wt::STMTraits>::begin (this=0, abiCodeProps=1) at src/transaction.h:157
#3 0007251 in tinystm::TransactionBaseABI<tinystm::wt::STMTraits, tinystm::TxnMngmnt<tinystm::wt::STMTraits, tinystm::wt::STMTraits>, true>::begin (this=0, flags=1, jmpbuf=105699145786208) at /home/martin/tinystmplusplus/src/transaction.h:227
#4 00005067 in GTM_begin_transaction (flags=1, jmpbuf=105699145786208) at /home/martin/tinystmplusplus/src/c-interface.h:69
#5 00005419 in _ITM_beginTransaction (nr=1) at /home/martin/tinystmplusplus/src/tm-wt.cpp:263
#6 00000184 in simple_read () at /home/martin/tinystmplusplus/test/simple/simple.c:20
Info:
  address: 104 (0x68)
  next: object at 47974457809056 (0x2ba448e2418a0) of size 4
  M030[4] (no allocation info)