Typestate-guided Exploration of Interleavings Using SMT

Robert Frohardt and Jonathan Turner

(with help from Devin Coughlin, Bor-Yuh Evan Chang, and Sriram Sankaranarayanan)

Outline

- Motivation
- Example
- SMT Encoding
- Results
- Future Work
- Conclusion
Motivation

- Concurrent programs are difficult to test
- Many possible thread interleavings
Motivation

- Concurrent programs are difficult to test
- Many possible thread interleavings

Instead, use SMT solver to explore interleavings automatically

### Example

**Thread a**

*\(x = 42\)

- Thread Spawn b
- Monitor Enter o
- \(x = \text{Null}\)
- Monitor Exit o
- Thread Join b

**Thread b**

- Monitor Enter o
- print *\(x\)
- Monitor Exit o
Example

```
4/30/2010
Thread Spawn b
Monitor Enter o
x = Null
Monitor Exit o
Monitor Exit o
Thread Join b
```

Possible Race Condition

Typestates

```
Set not null
Null
Not Null
Set not null
Error
Deref
Set null
Deref
Set null
```

Relation: Set null
Encoding Synchronization

\[ a_0 < a_1 \land a_1 < a_2 \land a_2 < a_3 \]
\[ \land a_3 < a_4 \land a_4 < a_5 \]
\[ \land b_0 < b_1 \land b_1 < b_2 \]
\[ \land a_1 < b_0 \land b_2 < a_5 \]
\[ \land (a_2 < b_0 \Rightarrow a_4 < b_0) \]
\[ \land (b_0 < a_2 \Rightarrow b_2 < a_2) \]
Results

Error detected at depth 0! Here is an example of a serialization that leads to an error: Running type state model for object Main.heapObject:

```
48, 0: ThreadStart -> Null
3, 0: ThreadStart -> Null
3, 125: SetNotNull Main.heapObject -> NotNull
48, 3: ThreadExit -> NotNull
3, 189: ThreadSpawn 36 -> NotNull
3, 194: ThreadSpawn 43 -> NotNull
43, 0: ThreadStart -> NotNull
3, 197: MonitorEnter 37 -> NotNull
3, 198: SetNull Main.heapObject -> Null
3, 199: MonitorExit 37 -> Null
36, 0: ThreadStart -> Null
36, 1: MonitorEnter 37 -> Null
44, 0: ThreadStart -> Null
36, 2: Deref Main.heapObject -> Error
```

Future Work

- Complement dynamic analysis with static analysis of control flow
- Encode the typestate model for the SMT solver
- Prune long error witnesses
- Determine feasibility of interleaving in original program
Conclusion

- Difficult to discover bad interleavings with traditional testing
- **But**, we can explore interleavings using a trace, a typestate model, and an SMT solver
- Limitations to the current approach, but we hope to address these in future iterations

Questions?