Synchronisation Synthesis for Concurrent Programs

Thorsten Tarrach (joint work with Pavol Černý, Ashutosh Gupta, Thomas A. Henzinger, Arjun Radhakrishna, Leonid Ryzhyk and Roopsha Samanta)

Introduction: Concurrency bugs

- Concurrency bugs are hard to find and fix
- We attempt to fix them automatically using synthesis

**Specification:** add atomic sections, wait-notifies; reorder commands

**Correct program for concurrency**

Our synthesis

**Assertions pass in all scheduling**

Atomic sections example

- This example requires two atomic sections to be fixed
- With a linear trace we cannot infer where to place atomic sections

```plaintext
init: x = 0; t1 = F
thread1
A: 11 = x 1: 12 = x
B: 11++ 2: 12++
C: x = 11 3: x = 12
D: t1 = T 4: assert(¬(t1|x=2))
```

Using a happens-before relationship we can infer atomic sections after two iterations

- An atomic section is denoted by a loop inside a thread (it is created by adding an edge)

```plaintext
l1 = x
l1++
x = l1
l1 = x
x = l2
l2++
l2 = x
```

Reordering example

```plaintext
init: IntrMask=0; ready=0; handled=0
init_thread
M: IntrMask=1 R: handled=ready

thread
A: x=1
B: IM=1
C: IM=1
D: t1 = T 1: await(IM==1)
2: await(IM==1)
3: await(IM==1)
```

Reordering can cause regressions

- By analysing a good trace we can identify possible regressions before reordering instructions

```plaintext
1: await(IM==1)
A: IM=1
B: IM=1
C: IM=1
```

**Possible fixes:**

- Swapping B ↔ C or swap A ↔ C
- Changing the order of assertions and p failing

Conclusion

- We consider reorderings as fixes
- We generalise the counter-example trace to capture the cause of the error
- We prevent regressions by analysing good traces

Recent: Better trace generalisation

- Trace generalisation is crucial to the success of the synthesis
- Trace generalisation should capture the core of the bug

**Idea:** Represent traces as a Boolean formula over happens-before constraints

Let

- \( \pi \)
- \( \langle \text{thread withdraw} \rangle \)
- \( \langle \text{thread deposit} \rangle \)
- \( \langle \text{thread checkresult} \rangle \)

Representation of good interleavings of \( \pi \)

```plaintext
\begin{align*}
\langle \text{thread withdraw} \rangle &= \langle \text{localvars: temp} \rangle \\
\langle \text{thread deposit} \rangle &= \langle \text{localvars: temp} \rangle \\
\langle \text{thread checkresult} \rangle &= \langle \text{localvars: temp} \rangle 
\end{align*}
```

We introduce rewrite rules on \( X^n \) for synthesis, e.g.

```plaintext
\begin{align*}
\langle \text{lock} \rangle &= \langle \text{lock} \rangle \\
\langle \text{ADD.LOCK} \rangle &= \langle \text{ADD.LOCK} \rangle 
\end{align*}
```

References