**CS6910: Testing/Verification of Concurrent Programs**

All computers are parallel,  
The only question is whether your program is.

---

**Need for concurrent programs**

- Hardware $\rightarrow$ multi-core architecture  
- Software $\rightarrow$ parallel multi-threaded codes  
- Serious challenge for software developers!

---

**Challenge to Write Concurrent Programs**

- Problem: Let $f$ be a function from integers to integers with a zero. Write a concurrent program Zero that finds such zero.  
- Idea: Solve the problem by splitting it into two subproblems that can be solved independently

---

**Solution 1**

```plaintext
found = false;  
x = 0;  
while (!found)  
    x = x + 1;  
    found = (f(x) == 0);  

found = false;  
y = 1;  
while (!found)  
    y = y - 1;  
    found = (f(y) == 0);  

ZERO-1 : [S1||S2]
```

---

**Solution 2**

```plaintext
x = 0;  
while (!found)  
    x = x + 1;  
    found = (f(x) == 0);  

y = 1;  
while (!found)  
    y = y - 1;  
    found = (f(y) == 0);  

ZERO-2 :  
found = false;  
[S1||S2];
```

---

**Solution 3**

```plaintext
x = 0;  
while (!found)  
    x = x + 1;  
    if (f(x) == 0)  
        found = true;  

y = 1;  
while (!found)  
    y = y - 1;  
    if (f(y) == 0)  
        found = true;  

ZERO-3 :  
found = false;  
[S1||S2];
```
Solution 4

```c
x=0;
while (!found)
    await turn=1
    then turn = 2;
    x = x+1;
if (f(x)==0)
    found = true;
ZERO-4 :
    turn=1; found=false;
[S1||S2];
```

Solution 5

```c
x=0;
while (!found)
    await turn=1
    then turn = 2;
    x = x+1;
if (f(x)==0)
    found = true;
ZERO-5 :
    turn=1; found=false;
[S1||S2];
```

Solution 6

```c
x=0;
while (!found)
    await turn=1
    then turn = 2;
    x = x+1;
if (f(x)==0)
    found = true;
ZERO-6 :
    turn=1; found=false;
[S1||S2];
```

Why to make sure software is correct?

- After all engineering is not science, always room for improvement
- What is the big deal if my windows application stops responding?
- Plenty of examples

Ariane 5 (1996)

- Ariane 5 used software used prior in Ariane 4
- 64-bit floating-point to 16-bit integer generated conversion an overflow
- Error was caught, sub-system shut down
- Error was caught, sub-system shut down
- Back-up systems failed for same reason
- Rocket veered off course
- Control system decided to abort
- Result: Rocket self-destructed
- Cost: $400 million payload
Pentium (1994)
- First release of Intel Pentium chip
- Mistakes when dividing floating-point numbers that occur within a specific range
- Estimated 3 million to 5 million defective chips
- Cost: $475 million

Mars Climate Orbiter (1999)
- One development team used pound/second in their code while the other used Newton/second
- Values passed from one module to another without conversion
- Result: Loss of the craft
- Cost: $125 million

Failing Computer System Costs Lives
- Potential problems are obvious:
  - Software used to control nuclear power plants
  - Air-traffic control systems
  - Spacecraft launch vehicle control
  - ...

Testing Seems Easy
- Just do the following:
  - designing test inputs
  - producing test values
  - running test scripts
  - analyzing results

Reality Seems Different
- Cost of software bugs to U.S. economy in 2002: $60 Billion
- 80% of software development cost is in debugging
- Testing
  - Effective in discovering bugs in early stages
  - Expensive and not exhaustive!
  - Testing does not scale

Testing Concurrent Programs Even More Difficult
- Testing multithreaded program is difficult:
  - Non-deterministic scheduler
  - $10^{69}$ interleavings for 3 threads each with 50 lines of code
  - Non-reproducible results
Scheduler in Testing and Reality

- Consider a program with two threads: the main thread forks off a child thread and then proceeds to write out ten instances of the string "X," with separate calls to Console.Write. Then the main thread waits for the child thread, which is concurrently writing out ten instances of the string "Y."

```csharp
foo (int a){
  1. y = a + 1;
  2. if (y < 2)
  3. complexA();
  4. else
  5. assert(y >= 2);
}
```

```csharp
bar (int b){
  6. if (b >= 0)
  7. y = b + 1;
  8. else
  9. complexB();
}
```

An Example

Input: \( a=1, b=0 \)

<table>
<thead>
<tr>
<th>Step</th>
<th>( n_1 )</th>
<th>( n_2 )</th>
<th>( n_3 )</th>
<th>( n_4 )</th>
<th>( n_5 )</th>
<th>( n_6 )</th>
<th>( n_7 )</th>
<th>( n_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Assertion: \( \checkmark \times \checkmark \times \checkmark \times \checkmark \times \checkmark \times \checkmark \times \checkmark \times \checkmark \)

Still an Immature Research Area

- No textbook, We will read research papers
- Ideal for
  - M.S. students who want to pursue Ph.D.
  - Ph.D. students who look for research projects
- Not ideal if you
  - want to be an expert in concurrent programming
  - Want to learn mature development/testing/verification tools

No textbook, We will read research papers

Ideal for
- M.S. students who want to pursue Ph.D.
- Ph.D. students who look for research projects

Not ideal if you
- want to be an expert in concurrent programming
- Want to learn mature development/testing/verification tools