Type Inference Extras

CSCI 334
Stephen Freund

Type Inference Applications
- Compilers
  - are values used consistently with some type?
- C++ template expansion
  - must we generate a new template version?
- JVM Safety Checking
- Race condition analysis

Programs on the Web

Intel 4004 (1971) 2,300 transistors
Intel 8086 (1978) 50,000 transistors
Intel Pentium 4 (2000) 50,000,000 transistors

Processor Clock Speeds

1000 MHz
100 MHz
10 MHz
1 MHz

Processor Clock Speeds

Minimum Clock Speed (MHz)
Intel Core i7 (2010)

2,000,000,000 transistors

Multi-Core Chips

Concurrent Programming With Threads

Concurrent Programming With Threads

Multithreaded Program Execution

Bank Server

Thread A

\[ t_1 = bal; \]
\[ bal = t_1 + 100; \]

Thread B

\[ t_2 = bal; \]
\[ bal = t_2 - 100; \]
Multithreaded Program Execution

Thread A
...  
\( t1 = bal; \)
\( bal = t1 + 100; \)  
...  
Thread B
...  
\( t2 = bal; \)
\( bal = t2 - 100; \)  
...  
\( bal = 500 \)  
\( bal = 400 \)

Race Condition

Thread A
...  
\( t1 = bal; \)
\( bal = t1 + 100; \)  
...  
Thread B
...  
\( t2 = bal; \)
\( bal = t2 - 100; \)  
...  
\( bal = 500 \)  
\( bal = 400 \)

Avoiding Race Conditions

Thread A
acquire(m);
\( t1 = bal; \)
\( bal = t1 + 100; \)
release(m);
...  
Thread B
acquire(m);
\( t2 = bal; \)
\( bal = t2 - 100; \)
release(m);
...  
\( bal = 500 \)  
\( bal = 400 \)

Avoiding Race Conditions

Thread A
acquire(m);
\( t1 = bal; \)
\( bal = t1 + 100; \)
release(m);
...  
Thread B
acquire(m);
\( t2 = bal; \)
\( bal = t2 - 100; \)
release(m);
...  
\( bal = 500 \)  
\( bal = 400 \)

• Common, Hard to Detect, Costly to Fix

Thread A
acquire(m);
\( t1 = bal; \)
\( bal = t1 + 100; \)
release(m);
...  
Thread B
acquire(m);
\( t2 = bal; \)
\( bal = t2 - 100; \)
release(m);
...  
\( bal = 500 \)  
\( bal = 400 \)

Thread A
synchronized(m){
\( t1 = bal; \)
\( bal = t1 + 100; \)
}
...  
Thread B
synchronized(m){
\( t2 = bal; \)
\( bal = t2 - 100; \)
}
...  
\( bal = 500 \)  
\( bal = 500 \)
Type Inference to Identify Races

Thread 1

synchronized(l) {
    x := 10;
}
synchronized(m) {
    synchronized(l) {
        x := !y + 1;
    }
    y := 2;
}

Thread 2

synchronized(m) {
    print !y;
}
synchronized(m) {
    print !x;
}