Scope and Memory Management

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

Running Programs in a Browser

www.nasa.gov

Satellite Tracking Page

files
monitor
printer
network

Running Programs in a Browser

www.nasa.gov

files
monitor
printer
network
Running Programs in a Browser

Sandbox Security Model

Enforcing Sandbox Boundaries

• Problem: Prevent direct access to resources

• Enforcement through type safety
  - permit library calls, but no "unsafe" operations
  - example:
    ```
    char *s = "moo";
    s = s - 1000; \to BAD
    print s;
    ```
  - another example:
    ```
    byte b[] = { 0x12, 0xa3, 0x05, ... };
    ((function)b)(); \to REALLY BAD
    ```

Using Type Safety for Security

• Compiler rejects programs with type errors:

• Why not sufficient for the Web?
JVM Architecture (incomplete)

Java Compiler → A.java

class A {
...
}

Java Virtual Machine
- loader
- linker
- bytecode interpreter
and libraries

Network
- loader
- linker
- bytecode interpreter
and libraries

(Compare to JavaScript)

Java vs. Java Bytecodes

class A extends Object {
    int i;
    void f(int val) { i = val + 1; }
}

Method void f(int)
0 aload 0
1 iload 1
2 2
3 3
4 putfield #4 <Field int i>
5 return

A obj. 100 Var 0
Var 1

Stack

Java vs. Java Bytecodes

class A extends Object {
    int i;
    void f(int val) { i = val + 1; }
}

Method void f(int)
0 aload 0
1 iload 1
2 2
3 3
4 putfield #4 <Field int i>
5 return

A obj. 100 Var 0
Var 1

Stack
Java vs. Java Bytecodes

class A extends Object {
    int i;
    void f(int val) { i = val + 1; }
}

Method void f(int)
0 aload 0
1 iload i
2 iconst 1
3 iadd
4 putfield #4 <Field i>
5 return

A obj. Var 0
Var 1
Stack

Java vs. Java Bytecodes

class A extends Object {
    int i;
    void f(int val) { i = val + 1; }
}

Method void f(int)
0 aload 0
1 iload i
2 iconst 1
3 iadd
4 putfield #4 <Field i>
5 return

A obj. Var 0
Var 1
Stack

Java vs. Java Bytecodes

class A extends Object {
    int i;
    void f(int val) { i = val + 1; }
}

Method void f(int)
0 aload 0
1 iload i
2 iconst 1
3 iadd
4 putfield #4 <Field i>
5 return

A obj. Var 0
Var 1
Stack

Variable and Stack Types

Method void f(int)
0 aload 0
1 iload i
2 iconst 1
3 iadd
4 putfield #4 <Field i>
5 return

A V int
W
int
S
A

Does stack top have two integers?
Stack  |  Var 0 Type | Var 1 Type | Method void f(int) |  
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>nil</td>
<td>int</td>
<td>0 aload 0</td>
</tr>
<tr>
<td>S1</td>
<td>V0</td>
<td>W0</td>
<td>1 iload 1</td>
</tr>
<tr>
<td>S2</td>
<td>V1</td>
<td>W1</td>
<td>2 iconst 1</td>
</tr>
<tr>
<td>S3</td>
<td>V2</td>
<td>W2</td>
<td>3 iadd</td>
</tr>
<tr>
<td>S4</td>
<td>V3</td>
<td>W3</td>
<td>4 putfield #4 &lt;Field int i&gt;</td>
</tr>
<tr>
<td>S5</td>
<td>V4</td>
<td>W4</td>
<td>5 return</td>
</tr>
</tbody>
</table>

```
S0 = nil
S1 = V0::S0
S2 = W1::S1
S3 = int::S2
S4 = int::'a
S5 == int::'b
S5 = 'b
V0 = A
W0 = int
V1 = V0
W1 = W0
V2 = V2
W2 = W2
V3 = V3
W3 = W4
```
Multi-Core Chips

Concurrent Programming With Threads

Concurrent Programming With Threads

Multithreaded Program Execution

Multithreaded Program Execution
### Race Condition

**Thread A**
- ... 
- \( t_1 = \text{bal}; \) 
- \( \text{bal} = t_1 + 100; \) 
- ... 

**Thread B**
- ... 
- \( t_2 = \text{bal}; \) 
- \( \text{bal} = t_2 - 100; \) 
- ... 

**Thread A**
- \( t_1 = \text{bal} \)
- \( \text{bal} = t_1 + 100 \)

**Thread B**
- \( t_2 = \text{bal} \)
- \( \text{bal} = t_2 - 100 \)

### Avoiding Race Conditions

**Thread A**
- acquire(m); 
- \( t_1 = \text{bal} \) 
- \( \text{bal} = t_1 + 100 \) 
- release(m); 

**Thread B**
- acquire(m); 
- \( t_2 = \text{bal} \) 
- \( \text{bal} = t_2 - 100 \) 
- release(m); 

- \( \text{bal is 500} \) 

- **Thread A**
- \( t_1 = \text{bal} \) 
- \( \text{bal} = t_1 + 100 \) 

- **Thread B**
- \( t_2 = \text{bal} \) 
- \( \text{bal} = t_2 - 100 \) 

- \( \text{bal is 400} \)

- Common, Hard to Detect, Costly to Fix

### Type Inference to Identify Races

**Thread 1**

\[ \text{synchronized(m)} \{ \]
\[ t_1 = \text{bal}; \]
\[ \text{bal} = t_1 + 100; \]
\[ \} \]

**Thread 2**

\[ \text{synchronized(m)} \{ \]
\[ x := 10; \]
\[ \} \]
\[ \text{synchronized(m)} \{ \]
\[ y := t_1 + 1; \]
\[ \} \]
\[ y := 2; \]
\[ \} \]