Metal, Continued

Reading: *Checking System Rules Using System-Specific, Programmer-Written Compiler Extensions*

17-654/17-754
Analysis of Software Artifacts
Jonathan Aldrich

Assertion Side-Effects

```c
#include <assert.h>

assert call_insensitive {  
  match expressions of "any" type  
  decl ( any ) expr, x, y, z;  
  used in combination to match all  
  calls with any arguments  
  decl ( any_call ) any_fcall;  
  decl ( any_args ) args;  

  // Find all assert calls. Then apply  
  // EM to "expr" in state "ln_assert."  
  expr ( assert(expr); ) =>  
    { expr_expr_recursion(expr, ln_assert); }  
  // Find all side-effects  
  in_assert:  
    // Match all calls  
    { any_fcall(args) } =>  
      { expr("function call"); }  
    // Match any assignment (including  
    // the operators +=, -=, etc.)  
    | ( x = y ) => { expr("assignment"); }  
    // Match all increments and decrements  
    // => x omitted for brevity  
    | ( z++ ) => { expr("post-increment"); }  
    | ( z-- ) => { expr("post-decrement"); }  

  // Causes VM fault if assert is disabled
```
Assertion Failure

- Perform reaching definitions *path-sensitively*
  - Different results for each path
- At an assert statement
  - If reaching definition of each variable in assert is a constant assignment, evaluate the assert
  - Flag an error if it is false
- Results
  - 5 errors in FLASH
  - Well-tested code
  - Def to assignment paths long and complex
    - e.g. 300 lines, 20 if statements, 4 else clauses, 10 conditional compilation directives

---

Tainted Analysis

- Kernel shouldn’t trust data from user
  - Could pass null references
- Analysis
  - Assume all data from user initially in *tainted* state
  - Tainted data cannot be used except by functions that check its validity
  - 18 errors
  - 15 false positives
  - Example error:

```c
/* from sys/kern/disk.c */
int sys_disk_request (u_int sn, struct Xn_name
                    *xn_user, struct buf *reqbp, u_int k) {
    ...

    /* bypass for direct scsi commands */
    if (reqbp->b_flags & B_SCSICMD)
        return sys_disk_scsicmd (sn, k, reqbp);
```

---
Memory Management

- Similar to PREfix
  - Catch leaks, use after free, possible null dereferences
- Challenge: How to do this intra-procedurally?
  - It’s common for procedures to return newly allocated memory
- Solution
  - Check error return paths
    - OS: those returning a negative integer
  - Catches many (but not all) errors
    - PREfix can do better using interprocedural analysis

Interprocedural analysis

- First, perform local analysis
  - e.g. does this function block?
  - e.g. are interrupts enabled?
- Later, perform reachability analysis on call graph
  - e.g. is a blocking function transitively called?
  - If so, interrupts better be enabled
- Can find only simple interprocedural errors
  - local analysis + reachability
- Vs. PREfix
  - Perform local analysis
  - Compute summary
  - Use summary to analyze callers
  - Handle recursion by exploring up to a fixed call depth
  - Can only track (language-level) information
- Vs. Fluid
  - Perform local analysis only
  - Use annotations to determine what a callee does
  - Can track sophisticated predicates but requires user input
Two Kinds of Path Sensitivity

- **Metal**
  - Explores all paths separately
  - Trims paths that share states at a program point
  - Does not keep track of predicates
  - If (threads)
  lock(y);
doSomething();
if (threads)
unlock(y)

- **PREfix**
  - Explores all paths separately
  - Keep track of predicates
  - Only explores feasible paths (based on predicates)
  - Metal will report a double-unlock error
    - False positive!
  - PREfix will not

Comparison

<table>
<thead>
<tr>
<th></th>
<th>Focus</th>
<th>Inter-procedural</th>
<th>Sound?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREfix</td>
<td>Language errors</td>
<td>Summaries</td>
<td>No</td>
</tr>
<tr>
<td>Fluid</td>
<td>Concurrency errors</td>
<td>Annotations</td>
<td>Yes/Contingent</td>
</tr>
<tr>
<td>Metal</td>
<td>Rule violations</td>
<td>Post-pass</td>
<td>No</td>
</tr>
</tbody>
</table>
Fugue:
Annotations for Protocol Checking

Reading: *The Fugue Protocol Checker: Is Your Software Baroque?*

17-654/17-754
Analysis of Software Artifacts
Jonathan Aldrich

Find the Bug!

```csharp
void CopyFile (string src, string dest)
{
    StreamReader fromFile = new StreamReader(src);
    StreamWriter toFile = new StreamWriter(dest);
    string line;
    while (((line = fromFile.ReadLine()) != null) {
        toFile.WriteLine(line);
    }
    fromFile.Close();
}
```
Find the Bug!

```csharp
static public string DoSocketGet (string server) {
    Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
    byte[] cmd = Encoding.ASCII.GetBytes("GET / HTTP/1.1\r\nHost: " + server + "\r\nConnection: Close\r\n\r\n");
    s.Send(cmd);
}
```

Specifications(1)

```csharp
class StreamWriter {
    [Creates]
    StreamWriter (string filename);

    [Disposes]
    void Close ();
}
```

- Invariants
  - No resource is referenced after its release
  - All resources are released or returned to caller
- Does this cover all uses in practice?
Specifications(2)

```java
class Socket {
    [Creates("raw")]
    public Socket (...);

    [ChangesState("raw", "bound")]
    public void Bind (EndPoint localEP);

    [ChangesState("raw", "connected"). ChangesState("bound", "connected")]
    public void Connect (EndPoint remoteEP);

    [InState("connected")]
    public int Send (...);

    [InState("connected")]
    public int Receive (...);

    [ChangesState("connected", "down")]
    public void Shutdown (SocketShutdown how);

    [Disposes(State.Any)]
    public void Close ();
}
```

Specifications(3)

```java
class WebPageFetcher {
    [Opens("")]
    class WebPageFetcher {
        [InState("connected"). WhenEnclosingState="open"). NotAliased(WhenEnclosingState="open")]
        private Socket socket;

        [Creates("closed")]
        public WebPageFetcher () {
        }

        [ChangesState("closed", "open")]
        public void Open (string server) {
            Socket newSock = new Socket (AddressFamily.Internet, SocketType.Stream, ProtocolType.Tcp);
            this.socket = newSock;
            IPAddress host = Dns.Resolve(server).AddressList[0];
            socket.Connect(new IPEndPoint(host, 80));
        }

        [InState("open")]
        public string GetPage (string url) {
            this.socket.Send (Encoding.ASCII.GetBytes("GET / HTTP/1.1\r\nHost: * + server + ";
            //...
        }

        [ChangesState("open", "closed")]
        public void Close () {
            this.socket.Send(Encoding.ASCII.GetBytes("QUIT\r\n"));
            this.socket.Close();
        }
    }
```
Verification

private Socket socket;

public void Open (string server) {
    Socket newSock = new Socket(...);
    this.socket = newSock;
    IPAddress host = ...;
    socket.connect(...);
}

Verification

public string GetPage (string url) {
    this.socket.Send(...);
    ...
}
Aliasing Challenges

a.Open(); b.Open();

• Legal only if a != b

Fugue Alias Analysis

• Annotations
  • NotAliased
    • Field or param is unique pointer to an object
    • Local variables may temporarily alias
    • Allows type system to track state changes
    • Warning (lost track of object) if assigned to Escaping parameter
  • MaybeAliased
    • May have aliases
    • May not call state-changing functions
    • If not escaping, error if assigned to field or passed to Escaping parameter
  • Escaping
    • A MaybeAliased parameter that may be (transitively) assigned to a field
Fugue Alias Analysis

- Analysis information
  - Environment env: var \to addr
  -Capabilities: addr \to aliasInfo
  -aliasInfo: one of NotAliased, MayBeAliased, MayBeAliased/Escaping

---

Example: Alias Analysis

```c
void f([MayBeAliased][Escaping] x);
void g([MayBeAliased] x);

void h([NotAliased] y) {
    Environment   Capabilities
    y \to a        a \to NA
    z = y;
    y \to a, z \to a a \to NA
    v = new T();
    y \to a, z \to a, v \to b a \to NA, b \to NA
    g(z);
    y \to a, z \to a, v \to b a \to NA, b \to NA
    f(v);
    y \to a, z \to a, v \to b a \to NA, b \to MBA
    a still NotAliased
    Warning: lost track of b
}
```

---
Typestate Analysis Lattice
Adapted from MSR TR and ECOOP '04 paper to match dataflow theory

- Lattice element $\sigma$
  - $(\text{Var} \rightarrow \text{Addr}, \text{Addr} \rightarrow \text{ObjDesc})$
  - ObjDesc: $(\text{Type, Alias, StateSet, FieldMap})$
    - State used for typestate analysis
  - Alias: NA, MA, MA/E
  - FieldMap: Field $\rightarrow$ Addr
  - Lattices are equivalent up to renaming of addresses

- $\sqsubseteq$
  - NA $\sqsubseteq$ MA $\sqsubseteq$ MA/E
  - $\sqsubseteq$ is $\subseteq$ for states
  - $L_1 \sqsubseteq L_2$ if merge($\alpha_1$, $\alpha_2$, $L_1$) $\sqsubseteq$ $L_2$
    - merge substitutes $\alpha_1$ for $\alpha_2$ in $L_1$; joins their states and fieldmaps, and
    - joins the both alias infos together with MA
    - Intuitively, allows more aliasing than was present before

- artificial $\perp$
- $\top = \{[x \rightarrow \alpha_T], (\alpha_T \rightarrow (T, MA/E, states(T), f \rightarrow \alpha_{type(f)}))$
- $\top = type(x)$
- Join
  - Least upper bound of $\sqsubseteq$
  - If NA becomes MA or MA/E, warn "lost track of object"

Typestate Analysis Flow Functions

- $f_{TA}\left(\sigma, \left[\text{new } T\right]_k\right) = [t_k \rightarrow \alpha][\alpha \rightarrow (T, NA, initState(T), \emptyset)] \sigma$
  - $\alpha \in \text{domain}(\sigma)$
- $f_{TA}\left(\sigma, \left[\ldots, n, f\right]_k\right) = [t_k \rightarrow \beta] \sigma$
  - $\sigma(t_n) = \alpha, \sigma(\alpha), f = \beta$
- $f_{TA}\left(\sigma, \left[\ldots, n, f\right]_k\right) = [t_k \rightarrow \beta][\beta \rightarrow \text{annot}(f)] \sigma$
  - $\sigma(t_n) = \alpha, f \in \text{domain}(\sigma(\alpha)), \beta \in \text{domain}(\sigma), T = \text{type}(f)$
  - $\text{annot}(f)$ denotes the state annotated on f
- $f_{TA}\left(\sigma, [x]_k\right) = [t_k \rightarrow \sigma(x)] \sigma$
- $f_{TA}\left(\sigma, [x := \ldots]_n\right) = [x \rightarrow \sigma(t_n)] \sigma$
- $f_{TA}\left(\sigma, */\text{any other */}\right) = \sigma$
Typestate Analysis Flow Functions

- \( f_{TA}(\sigma, [[...]]_n.f := [[...]]_m)_k = [\alpha.f \mapsto \sigma(t_m)] \sigma \)
- \( \sigma(t_n) = \alpha, \text{alias}(\sigma(\alpha)) = \text{NA} \)
- \( f_{TA}(\sigma, [[...]]_n.f := [[...]]_m)_k = \sigma \)
  - \( \sigma(t_n) = \alpha, \text{alias}(\sigma(\alpha)) = \text{NA}, \text{alias}(\sigma(t_m)) = \text{MA/E} \)
  - check that \( \text{pack}(\sigma(t_m), S_{\text{ann}}) \in \text{annot}(f, \text{state}(\sigma(\alpha))) \)
  - \( S_{\text{ann}} = \text{state}(\text{annot}(f, \text{state}(\sigma(\alpha)))) \)
- \( f_{TA}(\sigma, \text{fn}([[...]]_n)_k) = [\omega \mapsto \text{annot}(\text{fn}_{\text{out}})] \sigma \)
  - \( \sigma(t_n) = \alpha \)
  - check that \( \text{pack}(\sigma(\alpha), S_{\text{ann}}) \in \text{annot}(\text{fn}_{\text{in}}) \)
  - if alias(\sigma(\alpha)) = NA and alias(\text{annot}(\text{fn}_{\text{in}})) = \text{MA/E} 
    - lost track of \( t_n \) warning
- \( \sigma = \{ x \rightarrow \alpha_x \}, \{ \alpha_x \rightarrow \text{annot}(x) \} \)
- end of function
  - check for argument \( x \) that \( \text{pack}(\sigma(\alpha(x)), S_{\text{ann}}) \in \text{annot}(\text{fn}_{\text{out}}) \)
- \( \text{pack}((T, \text{alias}, S, \{ f_j \rightarrow \alpha_j \}), S') = (T, \text{alias}, S', \emptyset) \)
  - check that \( \text{pack}(\sigma(\alpha(x)), S_{\text{ann}}) \in \text{annot}(f, S') \)

Example: Type State Analysis

[WithProtocol("raw", "bound", "connected", "down")] class Socket {
  ... [InState("connected")]
  public int Send(...);
  [Disposes(State.Any)]
  public void Close();
}

[WithProtocol("open", "closed")] class WebPageFetcher {
  [InState("connected"),
   WhenEnclosingStates="open",
   NotAliased(WhenEnclosingStates="open")]
  private Socket socket;
  ... [ChangesState("open", "closed")]
  public void Close() {
    Socket sock = this.socket;
    sock.Send(...);
    sock.Close();
  }
}

Analysis Information

- Entry of Close() 
  - [this \rightarrow a_0, 
    \( a_0 \rightarrow \) (WebPageFetcher, NA, "open", 
    [socket \rightarrow a_1]),
    \( a_1 \rightarrow \) (Socket, NA, "connected", \( \emptyset \))]
  - Socket sock = this.socket;
  - [this \rightarrow a_0, sock \rightarrow a_1,
    \( a_0 \rightarrow \) (WebPageFetcher, NA, "open", 
    [socket \rightarrow a_1]),
    \( a_1 \rightarrow \) (Socket, NA, "connected", \( \emptyset \))]
  - sock.Send(...);
  - verify: sock in "connected" state (yes)
  - sock.Close();
- Exit of Close() 
  - verify: sock \in State.Any
  - verify: alias(\sigma(\delta(sock))) = NA
  - [this \rightarrow a_0, sock \rightarrow a_1,
    \( a_0 \rightarrow \) (WebPageFetcher, NA, "open", 
    [socket \rightarrow a_1])]
    - sock and this.socket become dangling
  - verify: \sigma(sock) \notin \text{domain}(\sigma)
Experience

- Web server application
  - 16,000 lines of code
  - Well tested, deployed
  - Checked DB library usage
- Errors
  - Disposing command object (17 times)
  - Closing DB connections (9 times)
    - Could cause out of resources error
- Observations
  - Added states to objects to track initialization
  - Annotated 24 methods and 6 fields
    - 3 more methods used library only intra-procedurally
- How would Metal have done?

Fugue vs. Metal, PREfix

- Fugue
  - Manual annotations
  - Can find inter-procedural errors
  - Tracks aliases for soundness
- Metal
  - Fully automatic (once protocol specified)
  - Finds only intra-procedural errors
  - Unsound
- PREfix
  - Fully automatic
  - Finds only language errors
  - Unsound