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

```
{ #include <assert.h> }
// Apply SM ignoring control flow
sm Assert flow_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
  // SM to "expr" in state "in_assert."
  start: { assert(expr); } ==>
       { mgk_expr_recurse(expr, in_assert); } ;
  // Find all side-effects
  in_assert:
     // Match all calls
     { any_fcall(args) } ==>
          { err("function call"); }
      // Match any assignment (including
      // the operators +=, -=, etc.)
    | { x = y }  ==> { err("assignment"); }
      // Match all increments and decrements
      // --z and ++z ommited for brevity
    | { z++ } ==> { err("post-increment"); }
    | { z-- } ==> { err("post-decrement"); } ;
}
```

- Flags error if assert() has side effects
- Illustrates matching on sub-expressions
- 14 errors, 2 false positives in ExOS
- Example bug:
  - ExOS, mmap

```
/* libexos/os/mmap.c:mmap_fault_handler:410 */
assert(_exos_self_insert_pte(0, PG_P|
     PG_U|PG_W, PGROUNDDOWN(va), 0, NULL) == 0);
```

  - Causes VM fault if assert is disabled

2

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:

```
/* 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
- PREfix
  - Explores all paths separately
  - Keep track of predicates
  - Only explores feasible paths (based on predicates)

```
if (threads)
    lock(y);
do_something();
if (threads)
    unlock(y)
```

- Metal will report a double-unlock error
  - False positive!
- PREfix will not

Comparison

	Focus	Inter-procedural	Sound?
PREfix	Language errors	Summaries	No
Fluid	Concurrency errors	Annotations	Yes/ Contingent
Metal	Rule violations	Post-pass	No

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!

```
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!

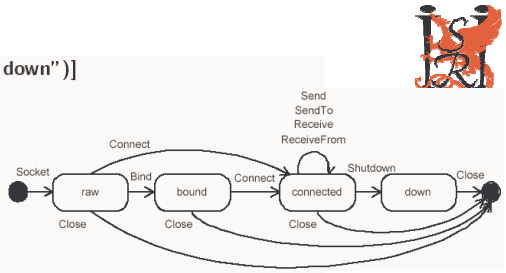
```
static public string DoSocketGet (string server)
{
  Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
  byte[] cmd = Encoding.ASCII.GetBytes("GET / HTTP/1.1\r\nHost: " +
     server + "\r\nConnection: Close\r\n\r\n");
  s.Send(cmd);


}
```


Specifications(1)

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

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

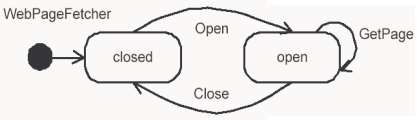
- Invariants
  - No resource is referenced after its release
  - All resources are released or returned to caller
- Does this cover all uses in practice?

Specifications(2)




```
[WithProtocol("raw","bound","connected","down")]
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)




```
[WithProtocol("open", "closed")]
class WegPageFetcher
{
  [InState("connected", WhenEnclosingState="open"), NotAliased(WhenEnclosingState="open")]
  [Unavailable(WhenEnclosingState="closed")]
  private Socket socket;

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

  [ChangesState("closed","open")]
  public void Open (string server)
  {
    Socket newSock = new Socket( AddressFamily.InterNetwork, 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 + "\r\nConnection: Close\r\n\r\n"));
    //...
  }

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

Verification

```
[InState("connected",
      WhenEnclosingState="open")]
[Unavailable(WhenEnclosingState="
      closed")]
private Socket socket;


[ChangesState("closed", "open")]
public void Open (string server) {
      Socket newSock
                  = new Socket(…);
      this.socket = newSock;
      IPAddress host = …;
      socket.connect(…);
}
```

- Initial assumption
  - WebServer closed
  - socket unavailable
- After new Socket(…)
  - newSock is raw
- After assignment
  - socket is raw
- Before Connect(…)
  - verify socket is raw or bound
- After Connect(…)
  - socket is connected
- End of method
  - Verify Webserver open
  - Ok because socket is connected

Verification

```
[InState("connected",
      WhenEnclosingState="open")]
[Unavailable(WhenEnclosingState="
      closed")]
private Socket socket;


[InState("open")]
public string GetPage (string url) {
      this.socket.Send(…);
      …
}
```

- Initial assumption
  - WebServer open
  - socket connected
- Before Send(…)
  - verify socket is connected
- After Send(…)
  - socket is still connected
- End of method
  - Verify Webserver open
  - Ok because socket is connected

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 → addr
  - Capabilities: addr → aliasInfo
  - aliasInfo: one of NotAliased, MayBeAliased, MayBeAliased/Escaping

Example: Alias Analysis

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

void h([NotAliased] y) {

    z = y;

    v = new T();

    g(z);

    f(v);
}
```

Environment	Capabilities
y → *a*	*a* → NA
y → *a*, z → *a*	*a* → NA
y→*a*, z→*a*, v→*b*	*a*→NA, *b*→NA
y→*a*, z→*a*, v→*b* *a still NotAliased*	*a*→NA, *b*→NA
y→*a*, z→*a*, v→*b* *Warning: lost track of b*	*a*→NA, *b*→MBA

Typestate Analysis Lattice

Adapted from MSR TR and ECOOP '04 paper to match dataflow theory

- Lattice element σ
  - (**Var** → **Addr**, **Addr** → **ObjDesc**)
  - ObjDesc: (**Type**, **Alias**, ***StateSet***, **FieldMap**)
    - State used for typestate analysis
  - **Alias**: NA, MA, MA/E
  - **FieldMap**: **Field** → **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(α_1, α_2, L_1) $\sqsubseteq L_2$
    - merge substitutes α_1 for α_2 in L_1; joins their statesand fieldmaps, and joins the both alias infos together with MA
    - Intuitively, allows more aliasing than was present before
- artificial $\perp$
- $\top$ = ({x → α_T},{α_T→(T,MA/E,states(T), f → $\alpha_{type(f)}$})
  - T = type(x)
- Join
  - Least upper bound of $\sqsubseteq$
  - If NA becomes MA or MA/E, warn "lost track of object"

Typestate Analysis Flow Functions

- $f_{TA}(\sigma, [\text{new T}]_k)$
  $= [t_k \mapsto \alpha][\alpha \mapsto (T, NA, initState(T), \varnothing)]\ \sigma$
  - $\alpha \notin domain(\sigma)$
- $f_{TA}(\sigma, [[\ldots]_n.f]_k) = [t_k \mapsto \beta]\ \sigma$
  - $\sigma(t_n)=\alpha$, $\sigma(\alpha).f=\beta$
- $f_{TA}(\sigma, [[\ldots]_n.f]_k) = [t_k \mapsto \beta][\beta \mapsto annot(f)]\ \sigma$
  - $\sigma(t_n)=\alpha$, $f \notin domain(\sigma(\alpha))$, $\beta \notin domain(\sigma)$, T=type(f)
  - *annot(f)* denotes the state annotated on f
- $f_{TA}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$
- $f_{TA}(\sigma, [x := [\ldots]_n]_k) = [x \mapsto \sigma(t_n)]\ \sigma$
- $f_{TA}(\sigma,$ /* *any other* */$) = \sigma$

Typestate Analysis Flow Functions

- $f_{TA}(\sigma, [[\ldots]_n.f := [\ldots]_m]_k) = [\alpha.f \mapsto \sigma(t_m)]\ \sigma$
  - $\sigma(t_n)=\alpha$, alias($\sigma(\alpha)$)=NA
- $f_{TA}(\sigma, [[\ldots]_n.f := [\ldots]_m]_k) = \sigma$
  - $\sigma(t_n)=\alpha$, alias($\sigma(\alpha)$)≠NA, alias($\sigma(\sigma(t_m))$)=MA/E
  - check that *pack*($\sigma(\sigma(t_m))$, S_{ann}) ⊑ *annot*(f, state($\sigma(\alpha)$))
  - S_{ann} = state(*annot*(f, state($\sigma(\alpha)$)))
- $f_{TA}(\sigma, [\text{fn}([\ldots]_n)]_k) = [\alpha \mapsto \textit{annot}(\text{fn}_{out})]\ \sigma$
  - $\sigma(t_n)=\alpha$
  - check that *pack*($\sigma(\alpha)$, S_{ann}) ⊑ *annot*(fn_{in})
  - if alias($\sigma(\alpha)$)=NA and alias(*annot*(fn_{in}))=MA/E
    - lost track of t_n warning
- σ_ι=({x → α_x},{α_x → *annot*(x))
- end of function
  - check for argument x that *pack*($\sigma(\sigma(x))$, S_{ann}) ⊑ *annot*(fn_{out})
- *pack*((T,alias,S, {f_i → α_i}), S') = (T,alias,S',∅)
  - check that *pack*($\sigma(\alpha_i)$, S_{ann}) ⊑ *annot*(f_i, 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",
      WhenEnclosingState="open"),
      NotAliased(WhenEnclosingState="open")]
      private Socket socket;
      ...
      [ChangesState("open", "closed")]
      public void Close() {
            Socket sock = this.socket;
            sock.Send(...);
            sock.Close();
      } ...
```

Analysis Information

- Entry of Close()
  - [this → a_0,
    a_0 → (WebPageFetcher, NA, "open", {socket → a_1}),
    a_1 → (Socket, NA, "connected", ∅)]
- Socket sock = this.socket;
  - [this → a_0, socket → a_1,
    a_0 → (WebPageFetcher, NA, "open", {socket → a_1}),
    a_1 → (Socket, NA, "connected", ∅)]
- sock.Send(…);
  - verify: sock in "connected" state (yes)
- sock.Close();
  - verify: sock ∈ State.Any
  - verify: alias($\sigma(\sigma(\text{sock}))$) = NA
  - [this → a_0, socket → $\boldsymbol{a_1}$,
    a_0 → (WebPageFetcher, NA, "open", {socket → $\boldsymbol{a_1}$})]
  - sock and this.socket become dangling
- Exit of Close()
  - verify: σ(sock) ∉ *domain*(σ)

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