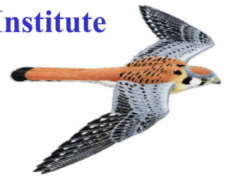
Synthesis of a Complex Software Vulnerability Analyzer (SVA)

Kestrel Institute

Jim McDonald

March 7, 2002

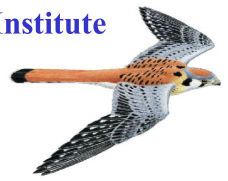
Kestrel
Institute



Outline

- Goals
- Project strategy and flow
- Initial success
- Implementation of tool
- Demo
- Taxonomies
- Current vision

Kestrel Institute

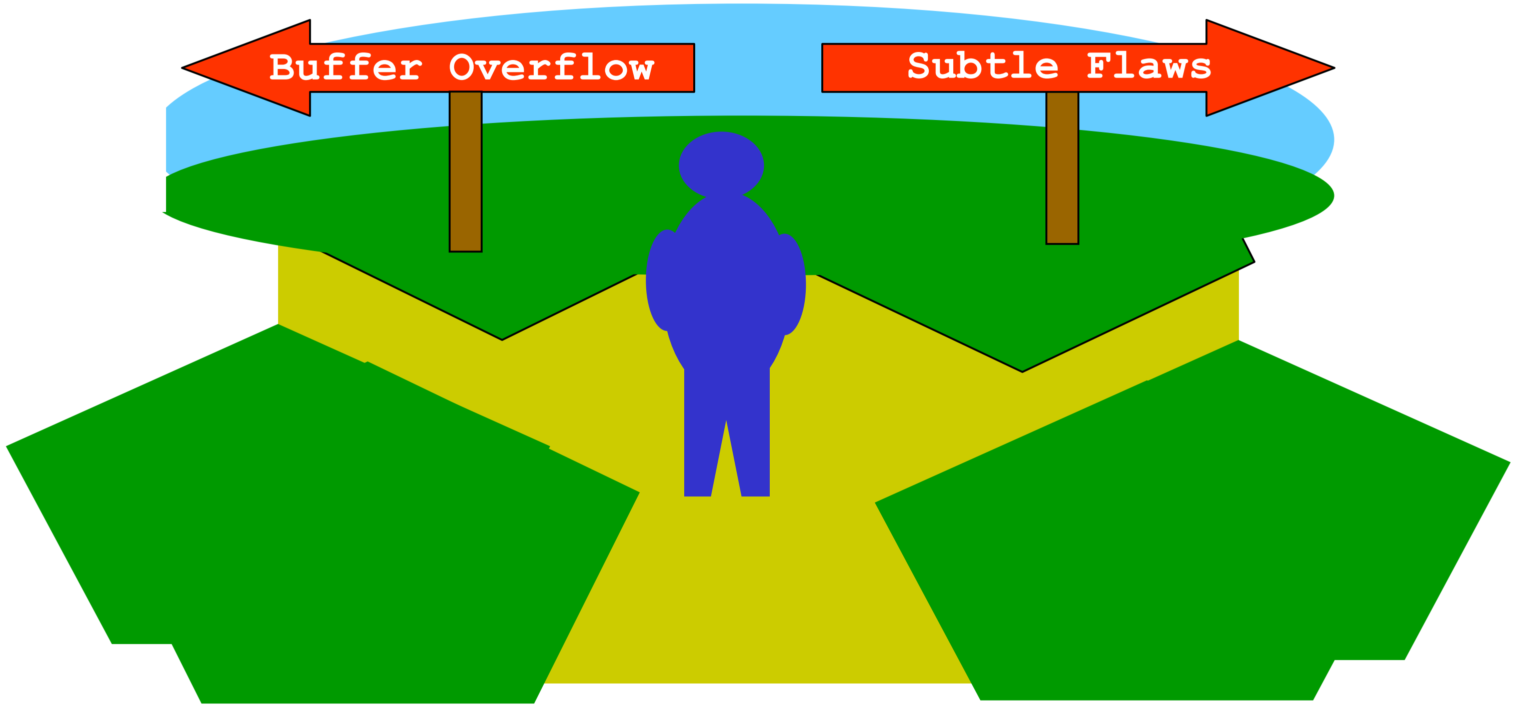


SVA Project Goals

- Build characterization of vulnerabilities to support automated analysis
  - Semantic rigor
  - Organized / Modular
  - Reusable
  - Extendable
- Build inference & analysis tools to detect vulnerabilities
  - Automation
  - Mixed initiative
- Demonstrate detection of real vulnerabilities

Kestrel Institute

SVA Project strategy

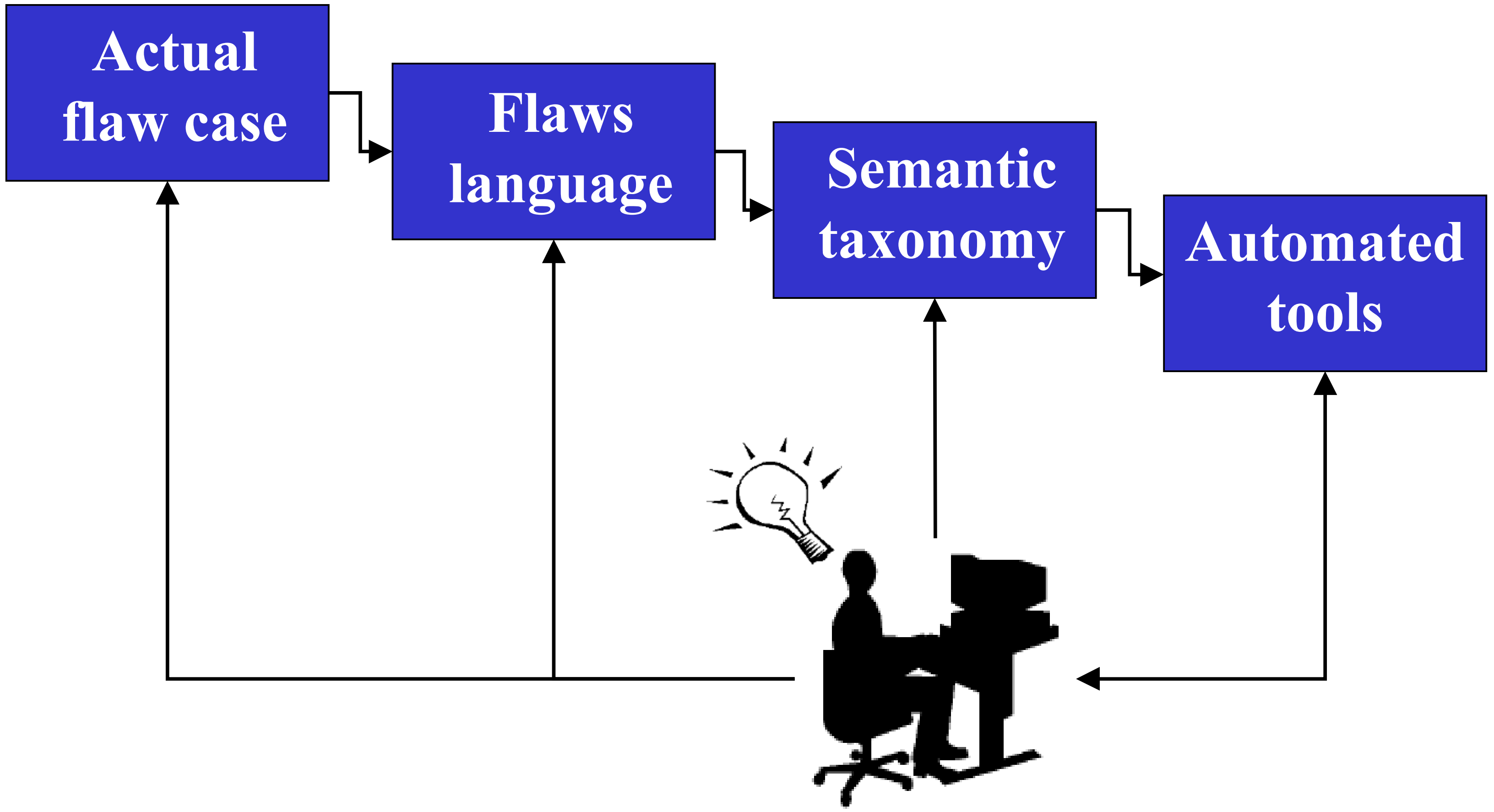


Subtle flaws

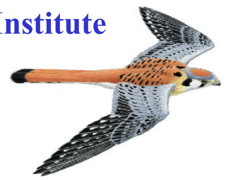
- Elude smart compiler – buffer overflow detection increasingly tractable
- Multiple element interactions – possibly great complexity
- Handle protocol implementations – optimization can cloud interactions
- Typically require human assessment & guided search to assess impact

Kestrel Institute

SVA Project flow



Kestrel Institute



August 8, 2000: real flaws

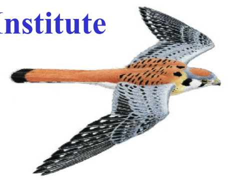
[ed note: text taken from Dan Brumleve's website]

2000.08.03, San Francisco

I've discovered a pair of new capabilities in Java, one residing in the Java core and the other in Netscape's Java distribution. The first (exploited in **BOServerSocket and BOSocket) allows Java to open a server which can be accessed by arbitrary clients.** The second (**BOURLConnection and BOURLInputStream) allows Java to access arbitrary URLs, including local files.**

As a demonstration, I've written **BOHTTPD** for Netscape Communicator. BOHTTPD is a browser-resident web server and file-sharing tool that demonstrates these two problems in Netscape Communicator. BOHTTPD will serve files from a directory of your choice, and will also act as an HTTP/FTP proxy server. **[ed note: "open door"]**

Kestrel Institute



Two days later

[ed note: text taken from Dan Brumleve's website]

2000.08.05

Right now I'm at the internet cafe (Club I) at 850 Folsom in San Francisco (between 4th and 5th street). I'll be here until 2:00 a.m. showing demos to anybody interested.

A guy showed up here and made BOHTTPD multithreaded. This new functionality is live right now…

WHOA! I just saw a Windows 2000 system that was still running BOHTTPD even after Netscape had been apparently terminated. Even the "Task Manager" showed no trace. **[ed note: "door stays open"]**

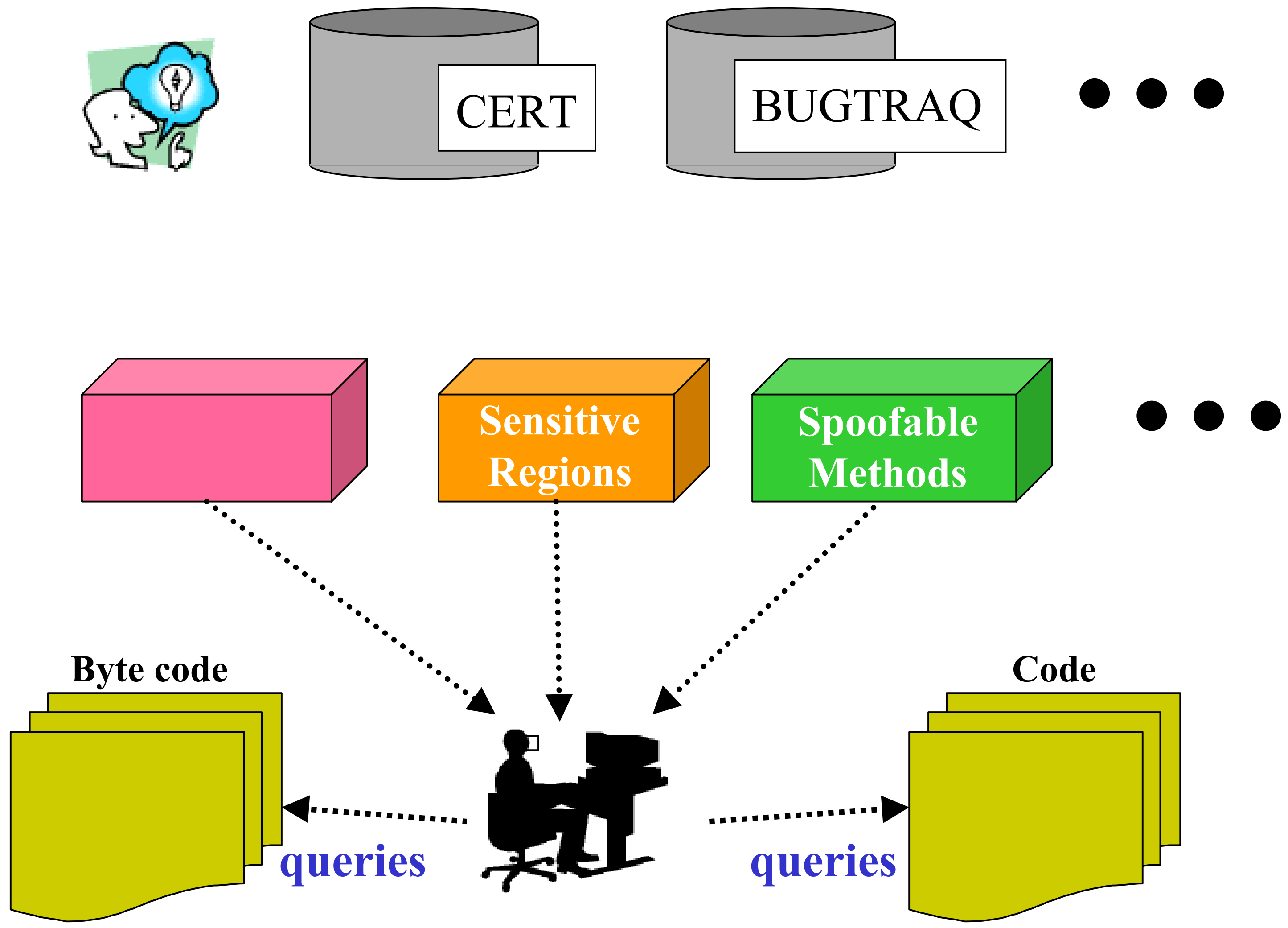
Kestrel Institute

Connecting flaw concepts to code

Mine rich sources of flaws

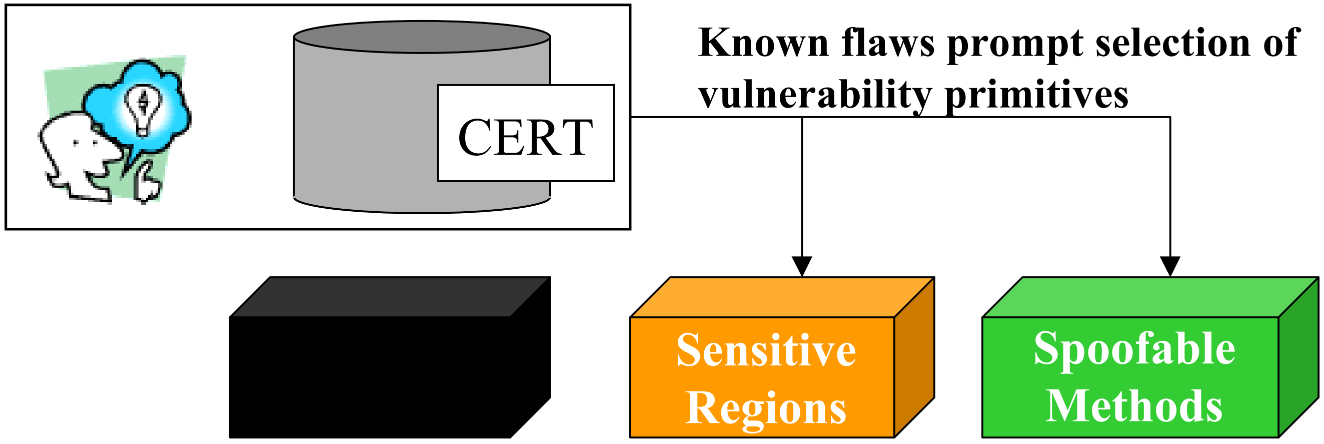
Select flaw primitives for a language

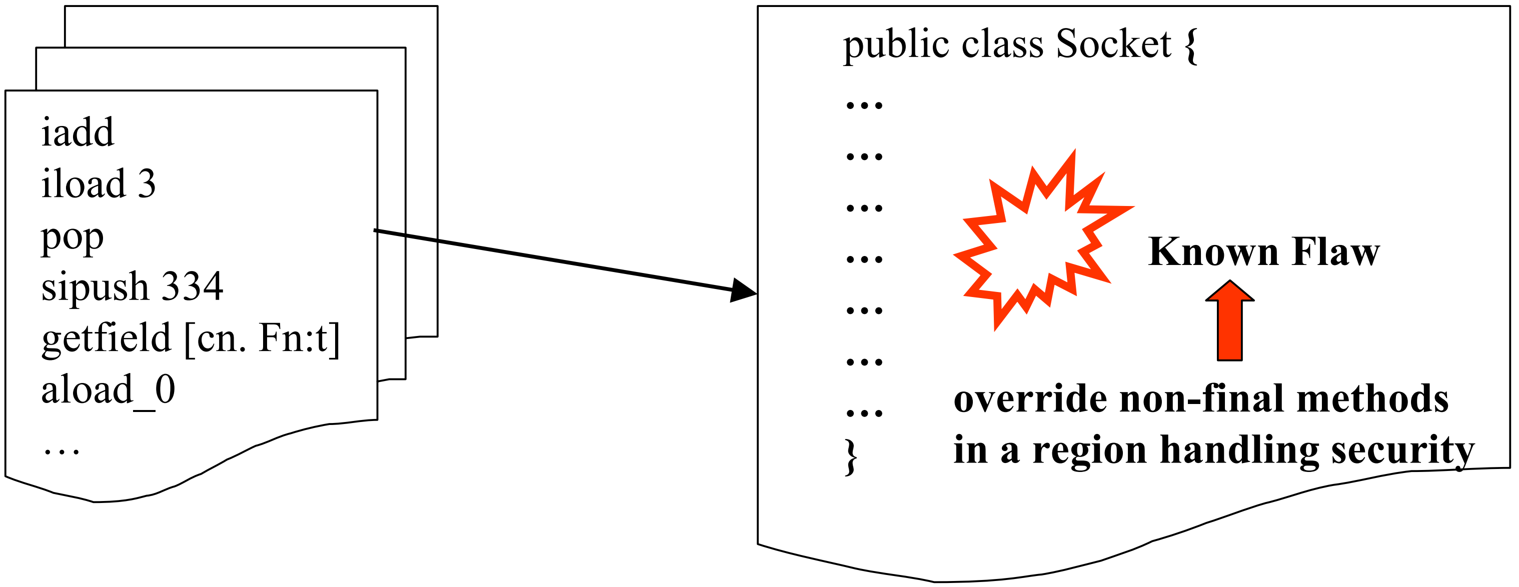
Create tools for making queries on (byte)code



Kestrel Institute

Work with Brumleve's "BO" attack




Data:
1031 Netscape class files
Known Flaw
override non-final methods in a region handling security

```
iadd
iload 3
pop
sipush 334
getfield [cn. Fn:t]
aload_0
…
```

```
public class Socket {
…
…
…
…
…
…
…
}
```

Kestrel
Institute

Anatomy of the “BO” attack

```
public class BOHTTP extends Applet {
  …
  public void init () {
  …
  ess = new BOServerSocket(port);
  …
  }
  …
  public void run () {
    BOSocket client;
    …
    client = ess.accept.any();
    BOHTTPConnection ff = new BOHTTPConnection();
    …
   (new Thread(ff)).start();
  }
 …
}
```

Kestrel Institute

Anatomy of the “BO” attack

```
public class BOServerSocket extends ServerSocket {
  …
  public BOSocket accept_any () throws IOException {
    BOSocket s = new BOSocket();
    try { implAccept(s); }
    catch (SecurityException se) { }  ←  Does Nothing!
    return s;
  }
}

public class BOSocket extends Socket {
  public void close_real () throws IOException {
    super.close();
  }
  public void close () { }  ←  Does Nothing!
}
```

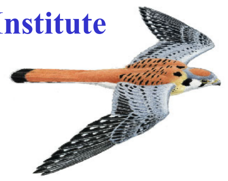
Kestrel Institute

Anatomy of the “BO” attack

```
protected final void implAccept (Socket socket) throws IOException
{ try
  {  socket.impl.address = new InetAddress();
     socket.impl.fd = new FileDescriptor();
     impl.accept(socket.impl);
     SecurityManager securitymanager = System.getSecurityManager();
     if (securitymanager != null)
     { securitymanager.checkAccept(socket.getInetAddress().getHostAddress(),
                                   socket.getPort());
       return; }
  …
  catch (SecurityException securityexception)
  {
      socket.close();                    ← Could be close from BOSocket!

      throw securityexception;           ← accept_any from BOServerSocket can thwart!
  }
}
public void close () throws IOException
{ impl.close }
```

Kestrel Institute



Anatomy of the “BO” attack

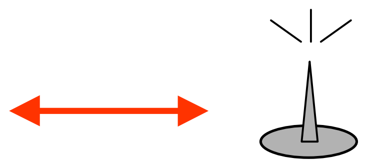
```
Class BOURLConnection extends URLConnection {
  …
  public BOURLConnection (URL u) {
    super(u);
    connected = true;
  }
}

Class BOURLInputStream extends URLInputStream {
  …
  public BOURLInputStream (URLConnection uc)
      throws IOException {
    super(uc);
    open();
   }
}
```

Kestrel Institute

Anatomy of the “BO” attack

```
class BOHTTPDConnection implements Runnable {
    …
    euc = new BOURLConnection(uu);
    euis = new BOURLInputStream(euc);
    while ((b = euis.read()) >= 0) os.write(b);
    …
}
```



Files exposed across the net

Kestrel Institute

Concepts lead to queries

Find all spoofable methods

Non-final methods that can be overridden

Compute their traces

Leverage from bytecode verifier

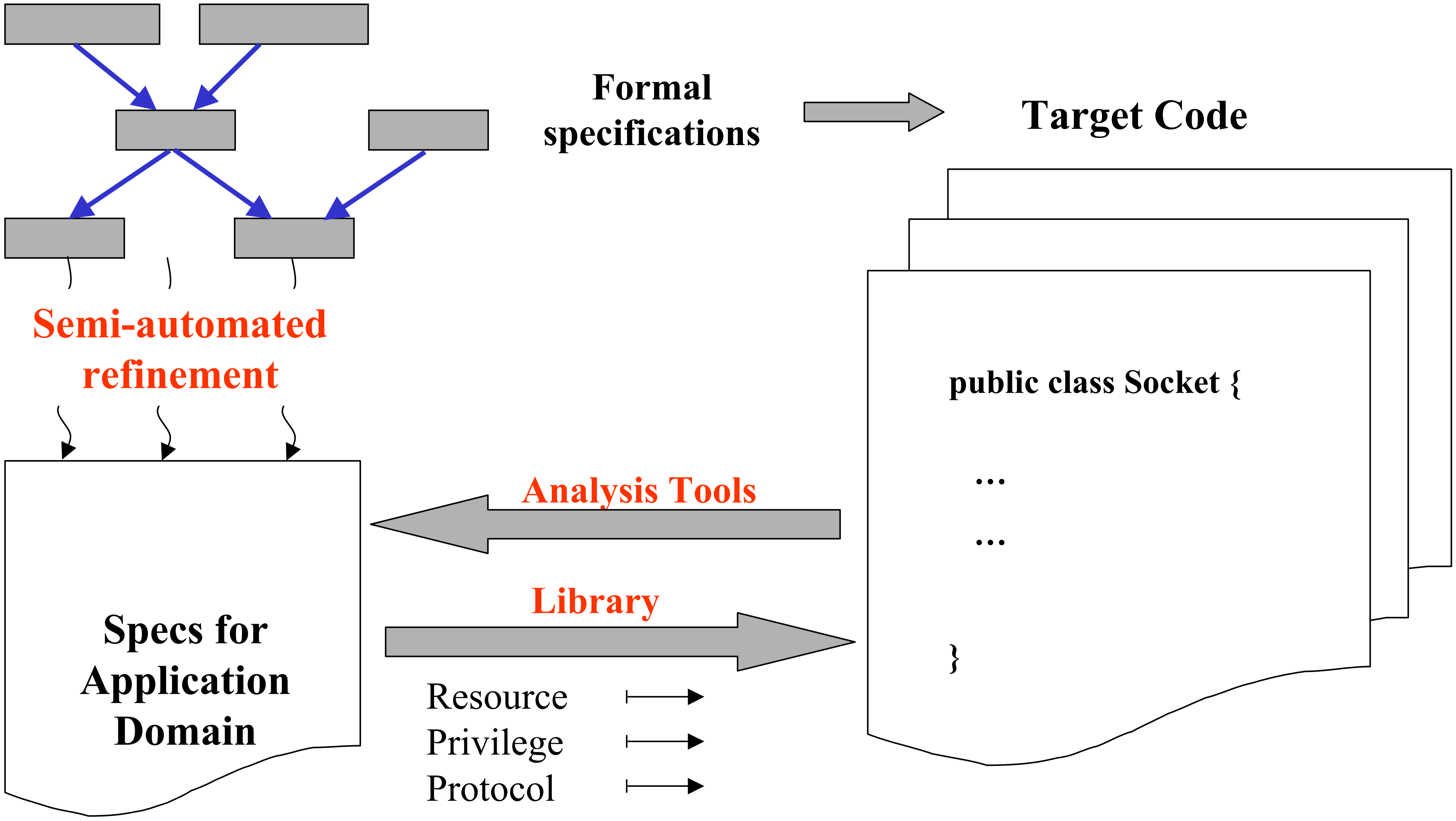
Find all sensitive regions

In particular, those handling security mechanisms

Look for invocations of spoofable methods that pass through sensitive regions

Kestrel Institute

Code ~~synthesis~~ analysis



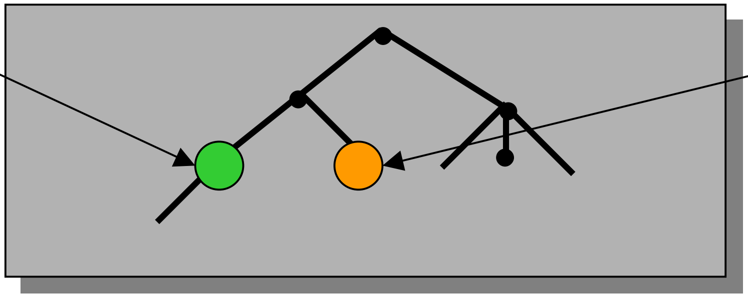
```
public class Socket {
    …
    …
}
```

Kestrel Institute

Formalizing the semantics

Spoofable invocations

Sensitive regions

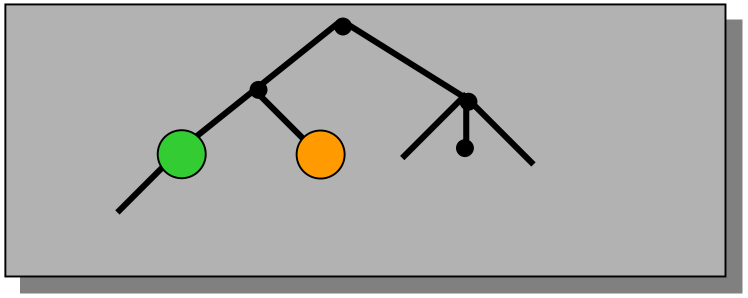


```
spec Spoofable_Invocation is
  op final?        : method     → Boolean
  op virtual?      : invocation → Boolean
  op spoofable? : invocation → Boolean
  …
  end-spec
```

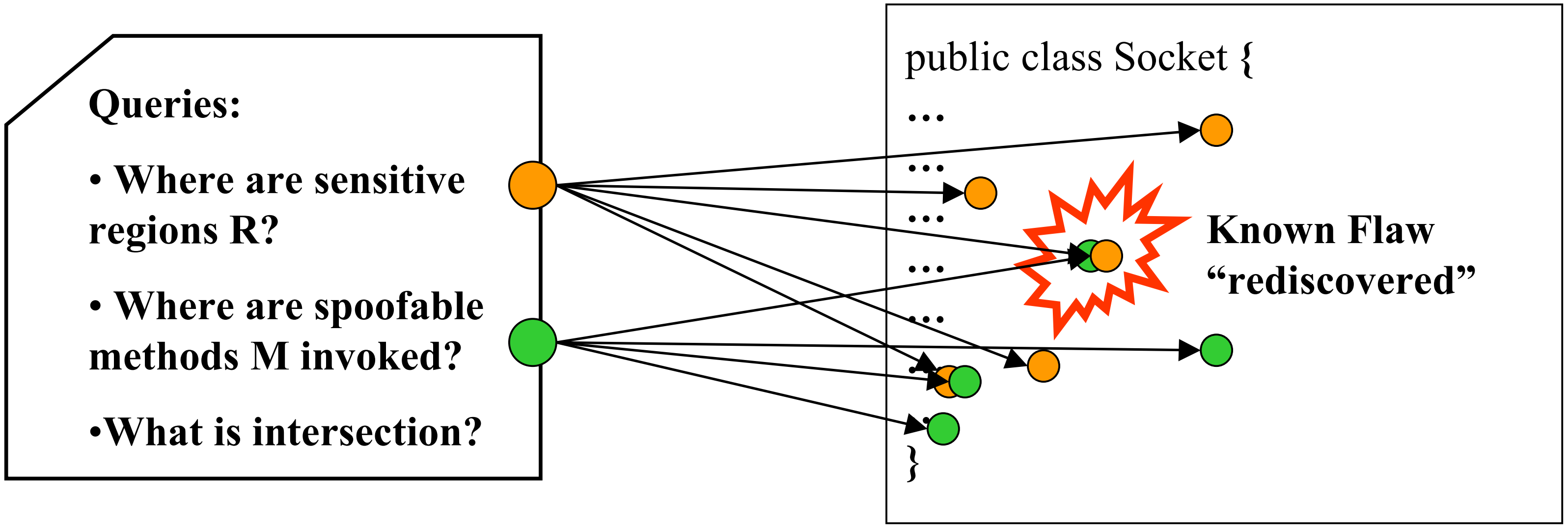
```
spec Sensitive_Region is
  sort Code_Region =
          {context      : method,
           start        : pc,
           end          : pc,
           attributes : set CR_Attribute}
  sort CR_Attribute = | privileged
                      | …
  …
  end-spec
```

Kestrel Institute

Initial queries on Brumleve's code



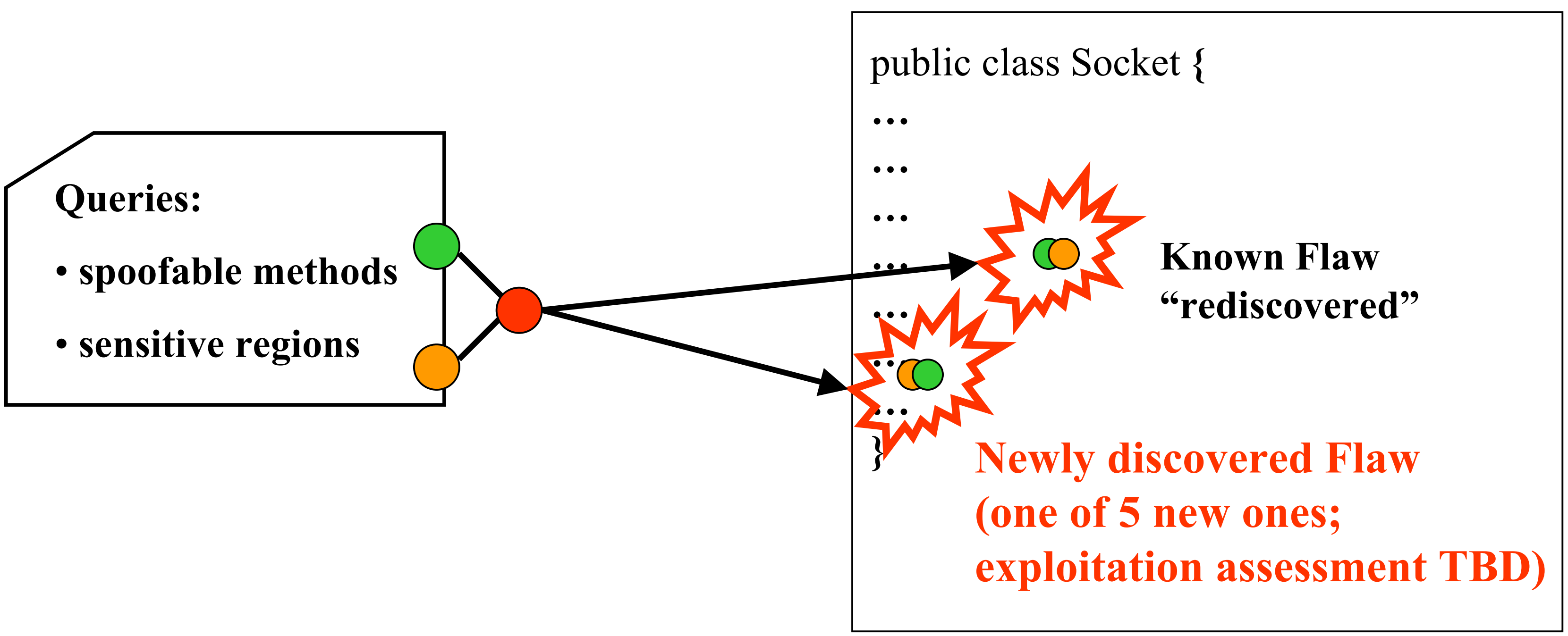
New entries for the semantic taxonomy



Queries:
• Where are sensitive regions R?
• Where are spoofable methods M invoked?
•What is intersection?
public class Socket {
…
}
Known Flaw "rediscovered"

Kestrel
Institute

Finding more than expected

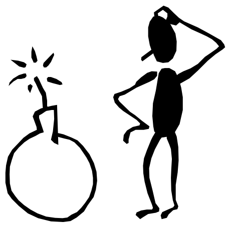


Kestrel Institute

Finding more than expected

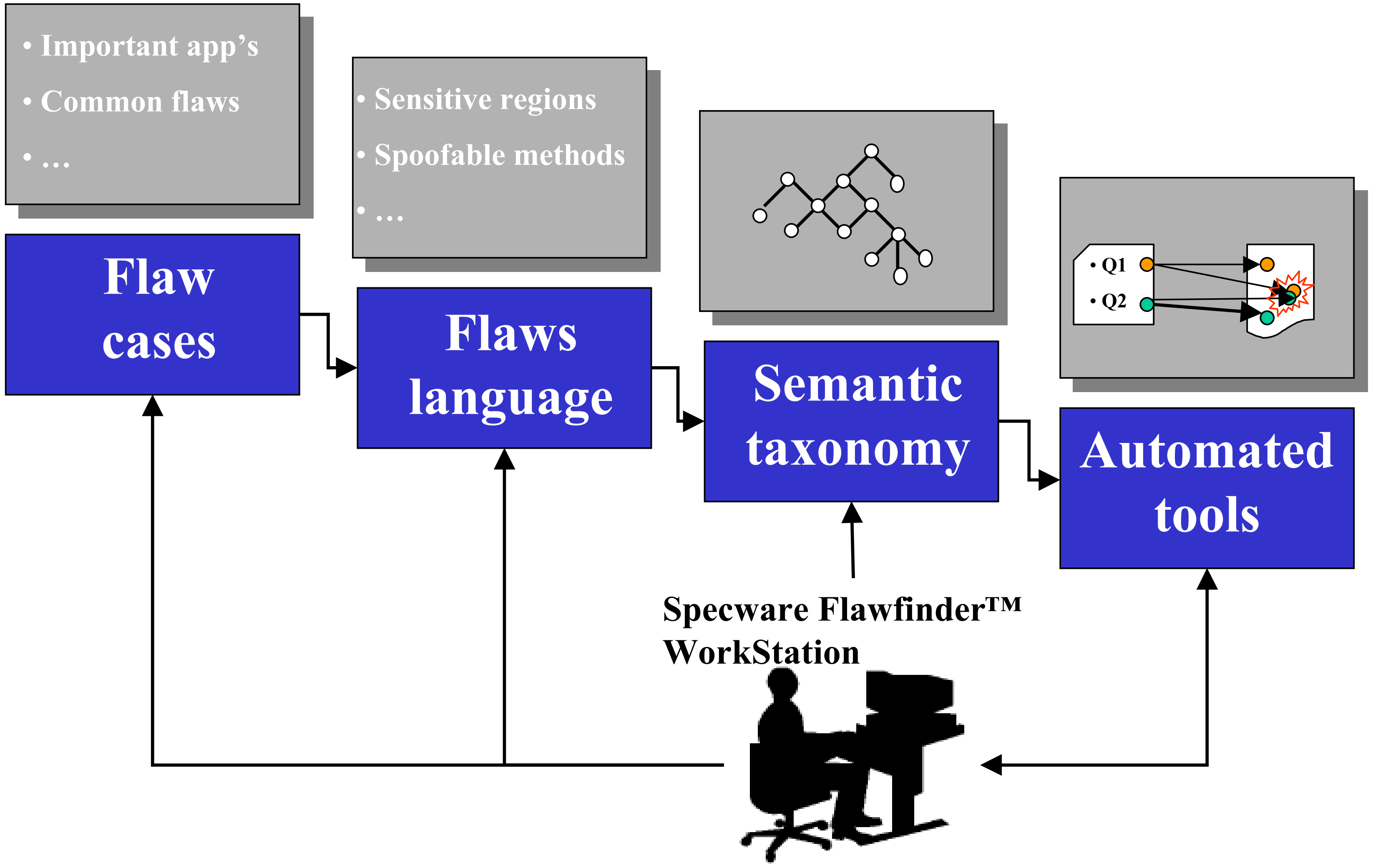
From java.net.DatagramSocket :

```
public synchronized void receive (DatagramPacket datagrampacket)
    throws IOException
{
   SecurityManager securitymanager = System.getSecurityMaganager();
   synchronized(datagrampacket)
   { if (securitymanager != null) do
     {  InetAddress inetaddress = new InetAddress();
        int I = impl.peek(inetaddress);
        try
         { securitymanager.checkConnect(inetaddress.getHostAddress(), I);
           break; }
        catch (SecurityException _ex)
         { DatagramPacket datagrampacket2 = new DatagramPacket (new byte[1], 1);
           impl.receive(datagrampacket2); }
      } while (true);
    impl.receive(datagrampacket);
   }
}
```



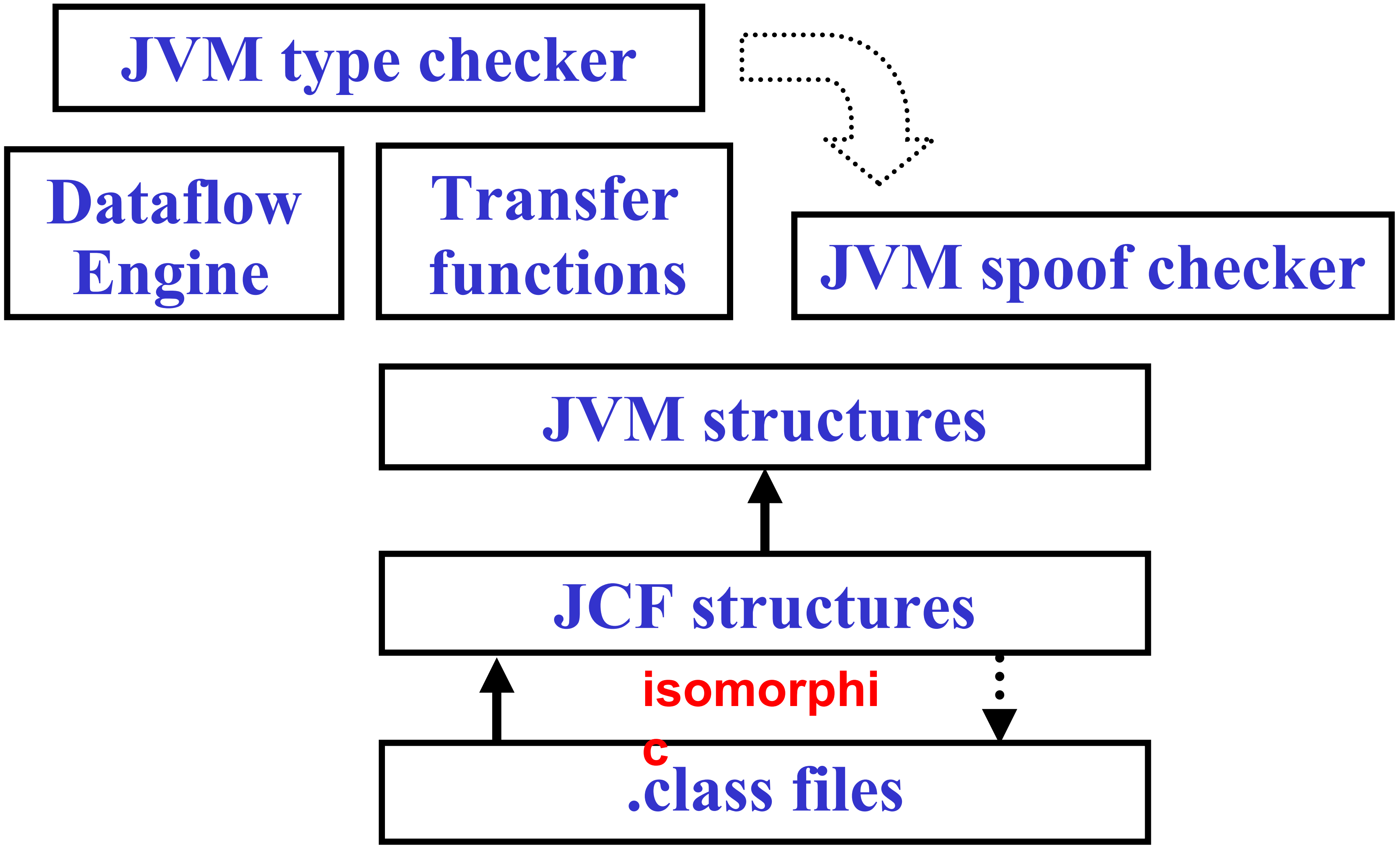
Kestrel Institute

Current vision



Kestrel Institute

Infrastructure



Kestrel
Institute

Performance

- Several Enhancements
  - Multiple entries for curried functions
  - Extensive use of hash codes
  - Canonical print routines
  - Various algorithmic improvements
  - Multiple refinements of maps, sequences, etc.

Kestrel Institute

Multiple Refinements

Many ways to implement maps

	update	access
Lists	O(1)	O(n)
Arrays	O(N)	O(1)
Trees	O(logN)	O(logN)

Kestrel Institute

Which Refinement?

Assume N updates followed by N accesses:

Map	$\longrightarrow$	List	O(N**2)	access
Map	$\longrightarrow$	Array	O(N**2)	update
Map	$\longrightarrow$	Tree	O(N log N)	access/update

Kestrel Institute

Multiple Refinements!

$$\begin{tikzcd} & \text{List} \arrow[d, leftrightarrow] \\ \text{Map} \arrow[r] & \left\{ \text{Array} \right. \arrow[d, leftrightarrow] \\ & \text{Tree} \end{tikzcd}$$

Kestrel Institute

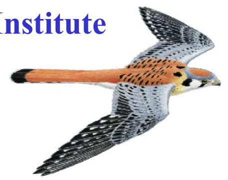
Multiple Refinements!

O(N) updates

O(N) accesses

$$\begin{tikzcd}
\text{List} & \bullet \arrow[r, "O(1)"] \arrow[d, <->, "O(N)"] & \bullet \arrow[r, "O(N)"] \arrow[d, <->, "O(N)"] & \bullet \arrow[d, <->, "O(N)"] \\
\text{Array} & \bullet \arrow[r, "O(N)"] \arrow[d, <->, "O(N)"] & \bullet \arrow[r, "O(1)"] \arrow[d, <->, "O(N)"] & \bullet \arrow[d, <->, "O(N)"] \\
\text{Tree} & \bullet \arrow[r, "O(\log N)"] & \bullet \arrow[r, "O(\log N)"] & \bullet
\end{tikzcd}$$

Kestrel Institute



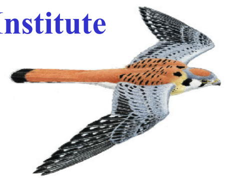
Description of Demonstration

- Background:
  - Show infrastructure for analyzing Java byte code
- Ideas:
  - spoofable invocation – virtual invocation of non-final method
  - sensitive region – try/catch/throw involving security, etc.
  - Intersection is a vulnerability
- Demo:
  - Write specs to instantiate these ideas
  - Generate code to find and report vulnerabilities

Demo

Start Demo!

Kestrel Institute



Taxonomies

- Semantically rich connections

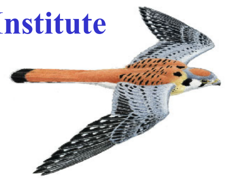
  Arrows embed one theory into another

- Exploited in semi-automated ways

  Results for theories propagate

- Morphisms from one taxonomy node into a domain theory provide leverage for constructing the embedding of children or sibling nodes

Kestrel Institute



Taxonomies of Vulnerabilities

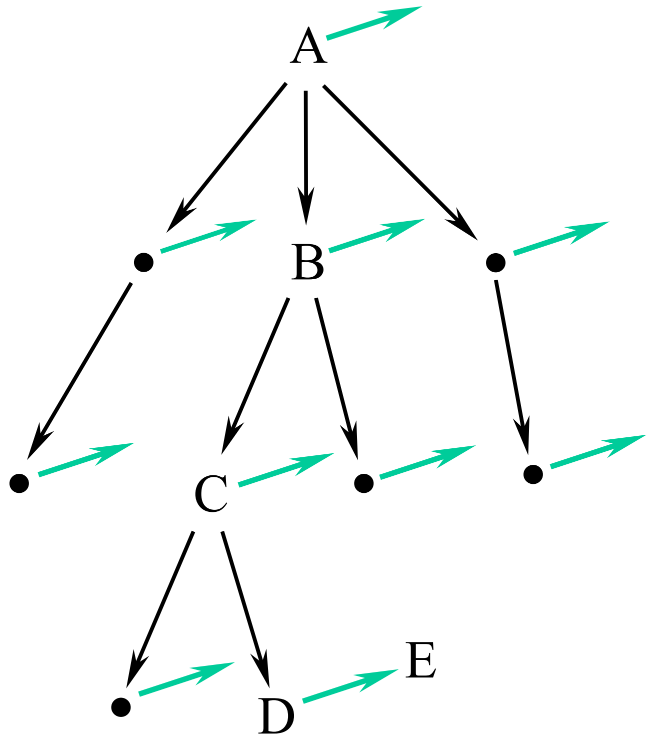
Developing a useful taxonomy of vulnerabilities requires:

- Languages for describing flaws
- Theories to express properties of flaws
- Morphisms to relate those theories
- Power tools to exploit morphisms

Kestrel Institute

Design by Classification

Refinements (green arrows) are organized into a taxonomy



Refinements are accessed and applied incrementally via a ladder construction

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & S_0 \arrow[d] \\
B \arrow[r] \arrow[d] & S_0^{+} \arrow[d] \\
C \arrow[r] \arrow[d] & S_0^{++} \arrow[d] \\
D \arrow[r] \arrow[d] & S_0^{+++} \arrow[d, "po"'] \\
E \arrow[r] & S_1
\end{tikzcd}$$

classifying the structure of S_0

applying the refinement $D \rightarrow E$

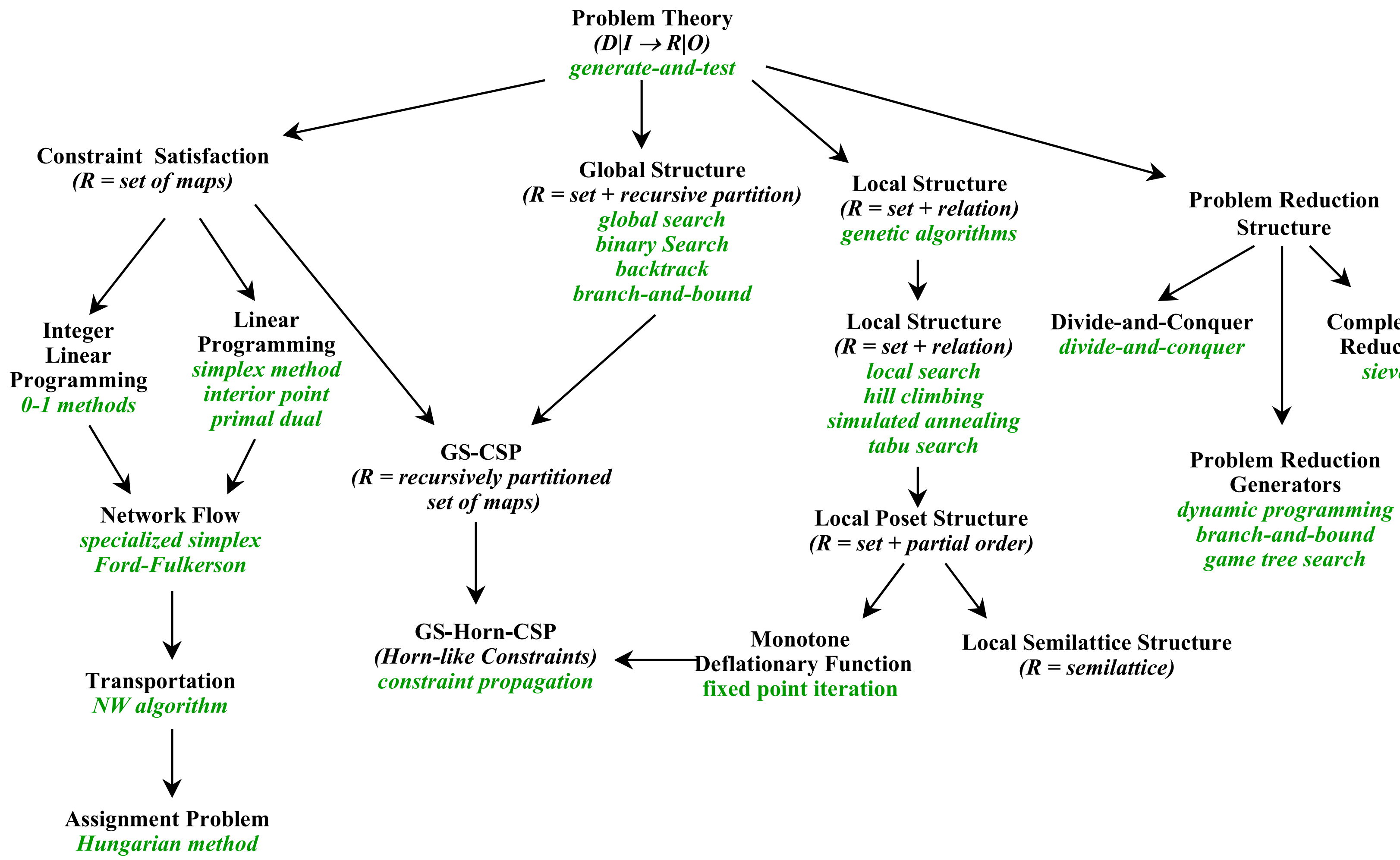
Kestrel Institute

Taxonomy of Collection Datatypes

$$\begin{tikzcd}
 & & & \text{PROTO-COLLECTION} \arrow[d] & & \\
 & \text{LIST} & & \text{PROTO-SEQ} \arrow[d] \arrow[lld] & & \\
 & \text{SEQ} \arrow[u, Rightarrow] \arrow[d] & & \text{PROTO-BAG} \arrow[d] \arrow[rd] & & \text{SEQ} \\
\text{ARRAY} & \text{BOUNDED-SEQ} \arrow[l, Rightarrow] & & \text{PROTO-SET} \arrow[d] & \text{BAG} \arrow[ru, Rightarrow] & \\
 & & \text{BAG} & \text{SET} \arrow[l, Rightarrow] \arrow[ld] \arrow[dd] \arrow[rd] & & \\
\text{INDEXED-PARTITION} & & \text{SET(TUPLE)} \arrow[ll, Rightarrow] & & \text{BOUNDED-SET} \arrow[r, Rightarrow] & \text{BIT-VECTOR} \\
\text{ORDERED-SEQ} & & & \text{SET-OVER-LINEAR-ORDER} \arrow[lll, Rightarrow] & &
\end{tikzcd}$$

Kestrel Institute

Taxonomy of Algorithm Theories



Kestrel Institute

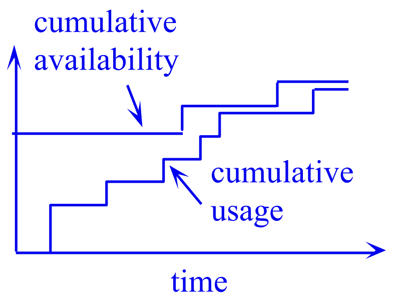
Taxonomy of Resource Theories

Resource *(Start-time, Resource-type, Instantaneous demand, Precedes…)*

Consumable

examples: fuel, crew time

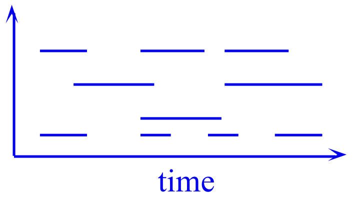
constraint: cum. use ≤ cum. avail



Reusable *(Duration, Finish-time, max/min-capacity,…)*

examples: parking lots, ramp space, parallel processors, power

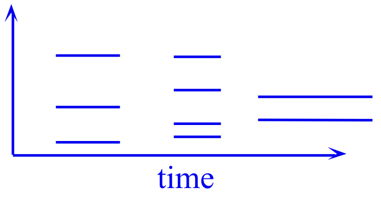
constraints: upper bound on capacity
finite usage intervals



Synchronously Reusable *(Separation)*

examples: transportation, washing machine

constraints: synchronized blocks of reservations
min separation between blocks



Transportation Resource

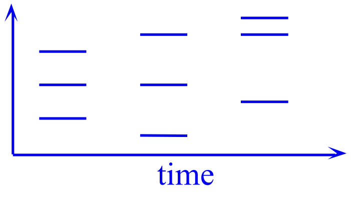
examples: ship, aircraft, truck

(Origin, Destination, speed, Duration = distance/speed)

Exact Capacity

example: wafer oven

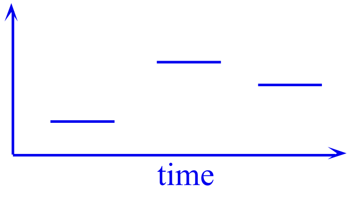
constraint: lb = ub on capacity



Nonsharable

examples: berth, runway, crew

constraint: capacity = 1



Kestrel Institute

Constructing Refinements

1. Library of Refinements

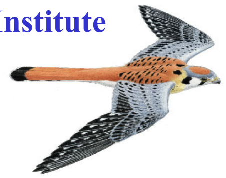
$$\begin{tikzcd}
\text{Global Search} \arrow[d] & \text{Set} \arrow[d] & \text{Resource} \arrow[d] \\
\text{Global Search Algorithm} & \text{Sequence} & \text{Transportation Resource}
\end{tikzcd}$$

2. Library of Refinement Generators

- Rewrite Simplification
- Context-dependent Simplification
- Finite Differencing
- Case Analysis
- Partial Evaluation

$$\begin{tikzcd}
\text{Global Search} \arrow[r] \arrow[d] & \text{Scheduling}_{0} \arrow[d, "\text{po}"'] & \\
\text{Global Search Algorithm} \arrow[r] & \text{Scheduling}_{1} \arrow[d, "\text{po}"] & \text{Set} \arrow[l] \arrow[d] \\
 & \text{Scheduling}_{2} \arrow[d, "\text{Context-dependent Simplification}"'] & \text{Sequence} \arrow[l] \\
 & \text{Scheduling}_{3} \arrow[d, "\text{Finite Differencing}"] & \\
 & \text{Scheduling}_{4} &
\end{tikzcd}$$

Kestrel Institute



Languages for Vulnerabilities

- **Ontology:**
  - Resource, Agent, Action, Manager, …
  - Privilege, Authorization, Friend, Enemy, …
  - Message, Channel, Send, Receive, Request, …
  - File, Owner, Read, Write, Modify, …
  - Process, Thread, Exception, Interrupt, …
- **Modal, Meta**, or **Higher-Order** Concepts
  - Time, Knowledge, Necessity, Desirability, …
  - Race, Deadlock, Cost, …
- **Objectives**
  - Security, Reliability, Availability, Efficiency, …

Typical Expressions

- Requests(x, y, action) □ trusts(y, x) □
  Executes(y, action)

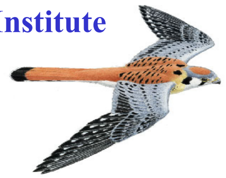
- Receives(x, msg) □
  Believes(x, sent(author(msg), msg))

Kestrel
Institute

Theory of a Flaw

- Receives(x,request) □ Validates(x,request) □ Executes(x,request)
- Send(x,y,request) □ author(request) = x
- Validates(x,request) ⇔
  Friend(author(request),x) □ ¬ Dangerous(request)
- ¬ Dangerous(send(x,y,z))
- Send(Intruder, Dupe, ‘Send(Dupe,Victim,bomb)’)

Kestrel Institute

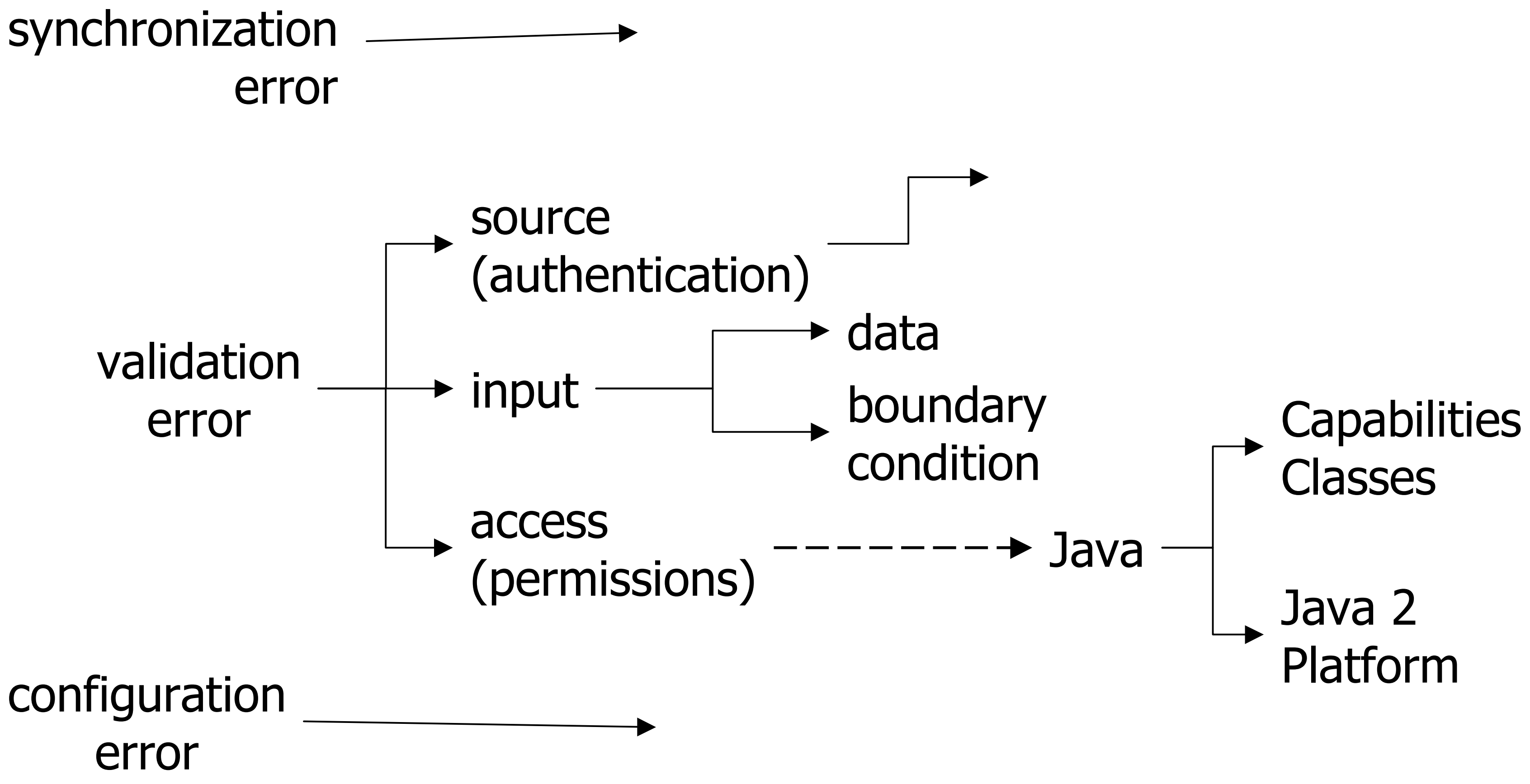


Morphisms

- Resource => Space, Processor, Data, …
- File => Unix-File, NT-File, …
- Privilege => Read, Write, Execute, …
- Read => fread, mmap, …

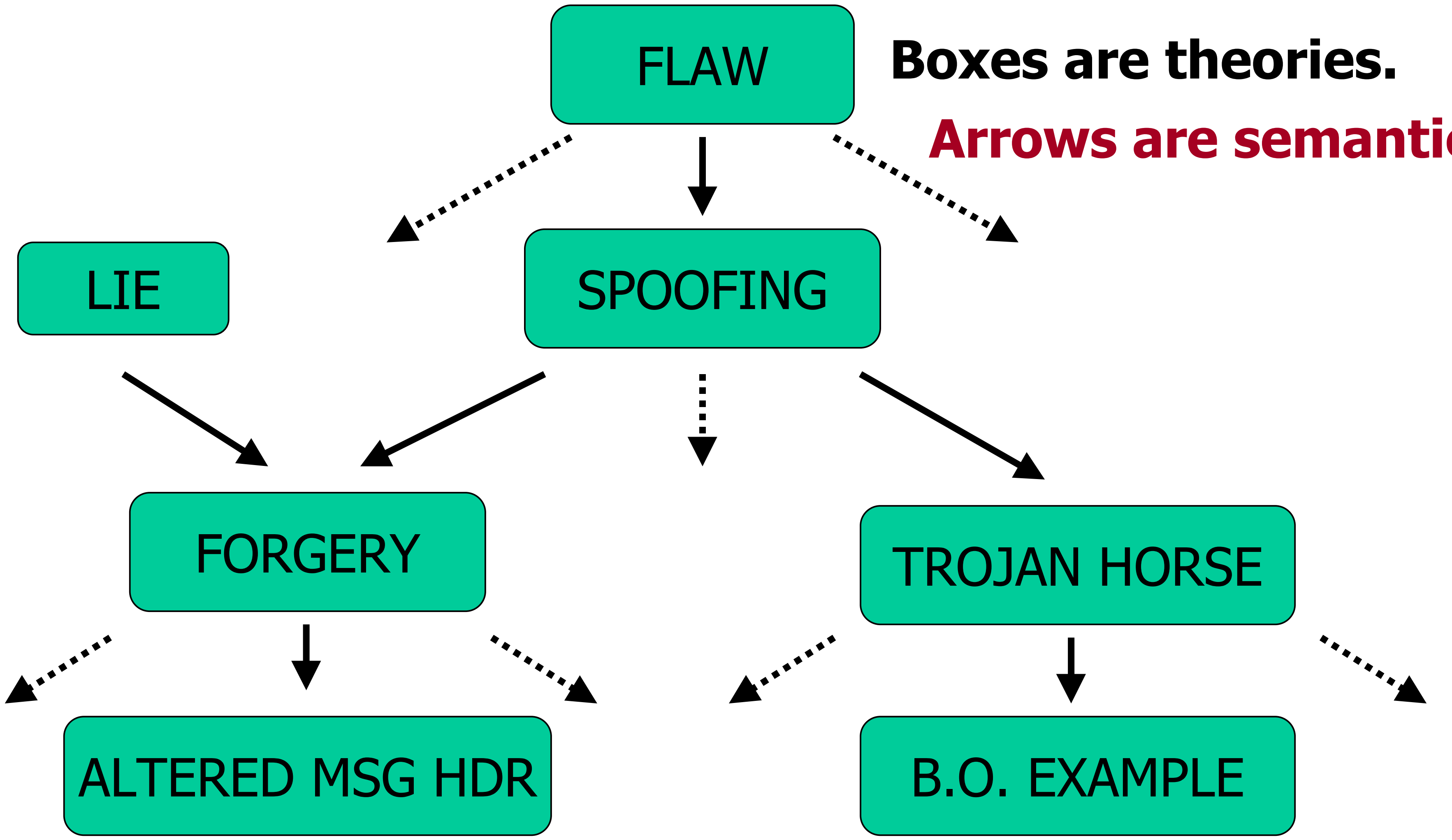
Kestrel Institute

Towards a taxonomy



Kestrel Institute

Semantic Taxonomy of Flaws

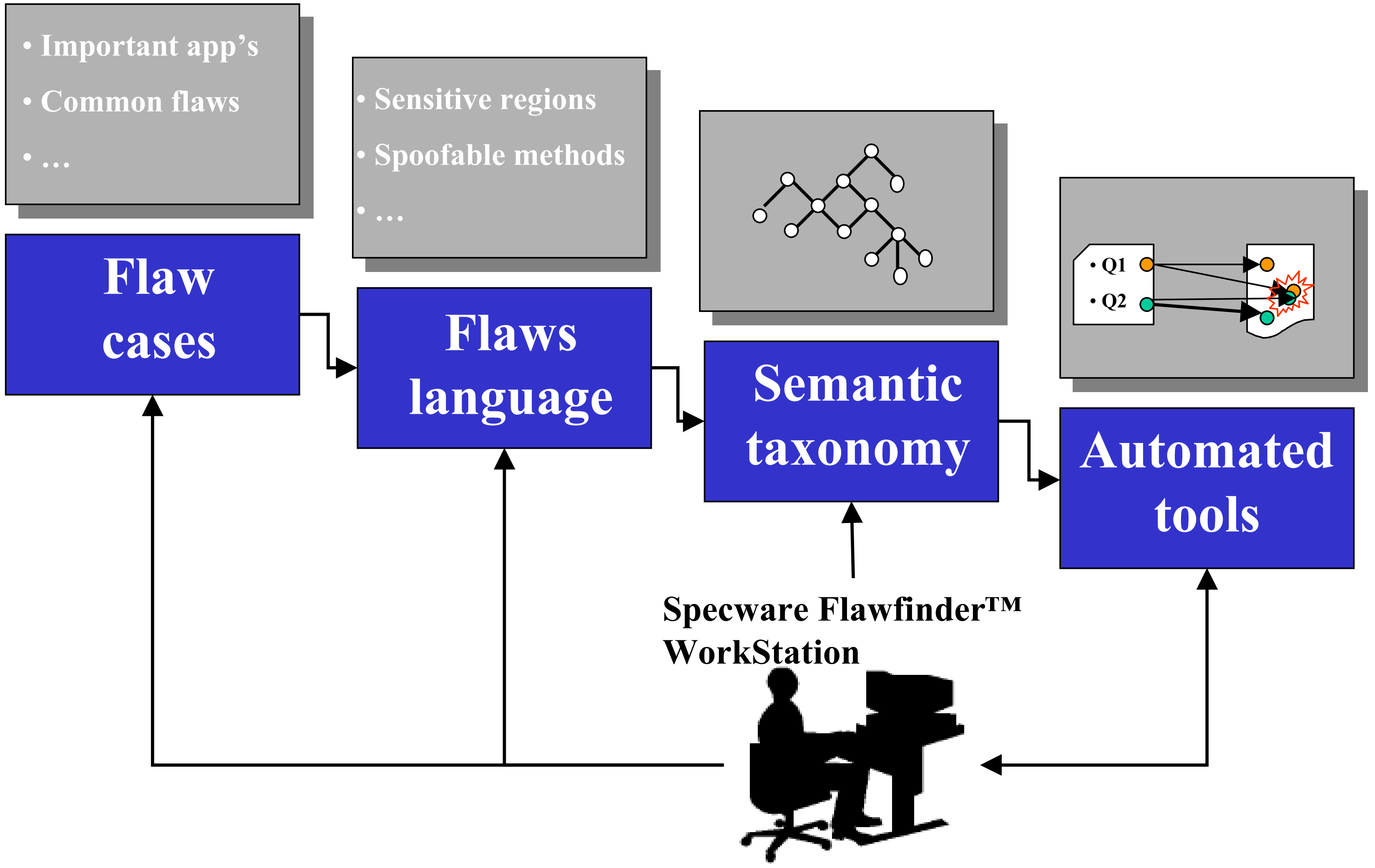


Boxes are theories.

Arrows are semantic!

Kestrel Institute

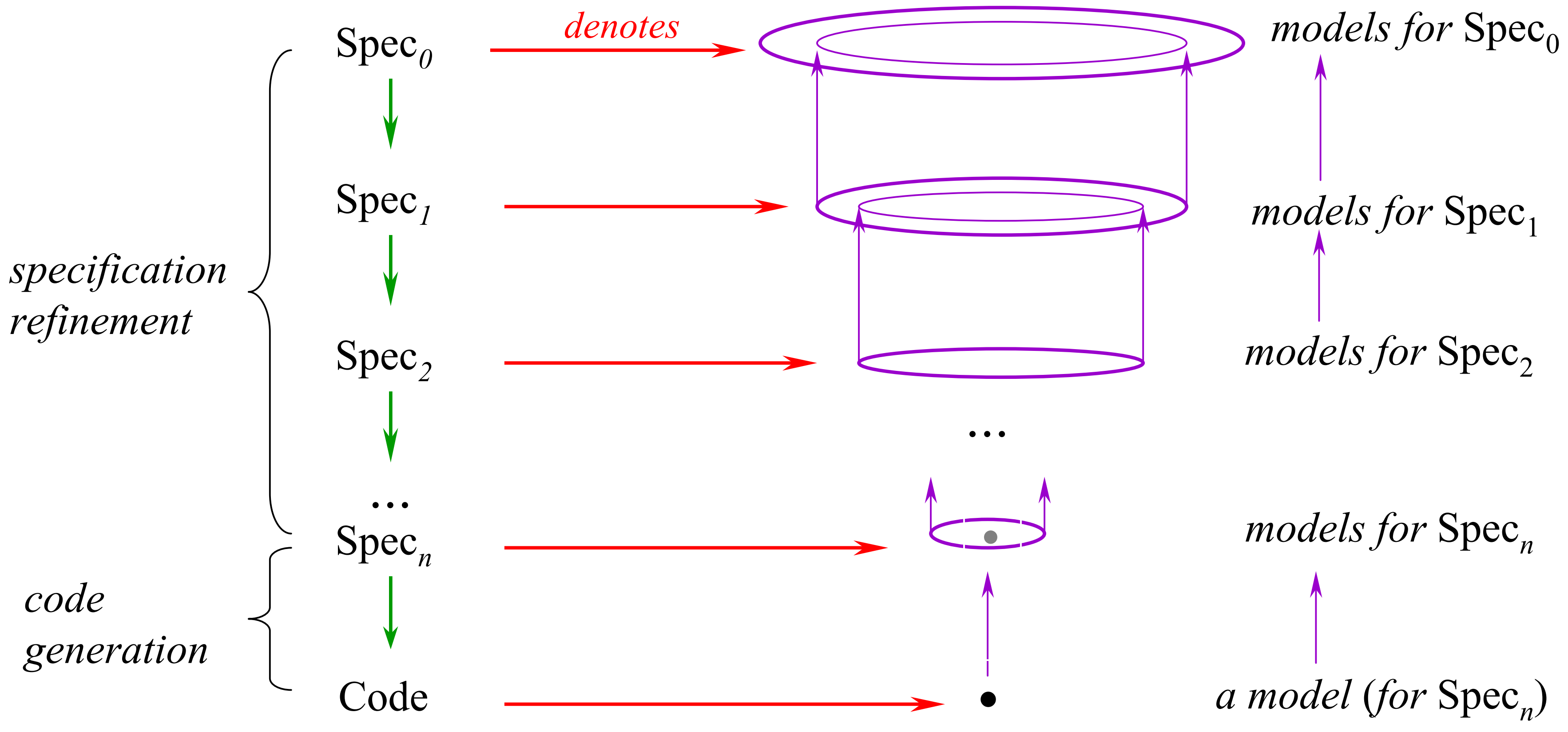
Current vision



Questions

Kestrel Institute

Software Development by Refinement



Code generation is accomplished via a logic morphism from **SPEC** *to the logic of a programming language*

Kestrel Institute

Planware Refinements

$$\begin{tikzcd}
\text{Resource} \arrow[r] \arrow[d] & \text{Abstract Scheduling} \arrow[d] & \\
\text{Transportation Resource} \arrow[r, "po"] & \text{Transportation Scheduling 0} \arrow[d] & \text{Task} \arrow[l] \arrow[d] \\
\text{Semilattice Attribute of Task} \arrow[r] \arrow[d] & \text{TS 1} \arrow[d] & \text{Transportation Tasks} \arrow[l, "po"'] \\
\text{Definite Constraint} \arrow[r, "po"] & \text{TS2} \arrow[d] & \mathit{Set}(A\times B\times C) \arrow[l] \arrow[d] \\
 & \text{TS3} & \mathit{Indexed\text{-}Partition}\ \mathit{map}(A,\ \mathit{Set}(A\times B\times C)) \arrow[l, "po"']
\end{tikzcd}$$

user chooses from the Resource Taxonomy

user edits the Spreadsheet

the rest is automatic!

Kestrel Institute

Planware Refinements (cont'd)

algorithm design
and
program optimizations

$$\begin{tikzcd}
\text{DRO} \arrow[r] \arrow[d] & \text{TS4} \arrow[d] & \\
\text{Global Search with CP} \arrow[r] \arrow[d] & \text{TS5} \arrow[d] & \\
\text{Global Search program} \arrow[r, "\text{po}"] & \text{TS6} \arrow[d] & \text{Definite Constraints} \arrow[l] \arrow[d] \\
& \text{TS7} \arrow[d, "\text{Context-Dependent Simplification}"'] & \text{Constraint Propagation algorithm} \arrow[l, "\text{po}"'] \\
& \text{TS8} \arrow[d] & \text{Sort} + n\text{-attributes} \arrow[l] \arrow[d] \\
& \text{TS9} \arrow[d] & n\text{-tuple} \arrow[l, "\text{po}"'] \\
& \text{CommonLisp code} &
\end{tikzcd}$$