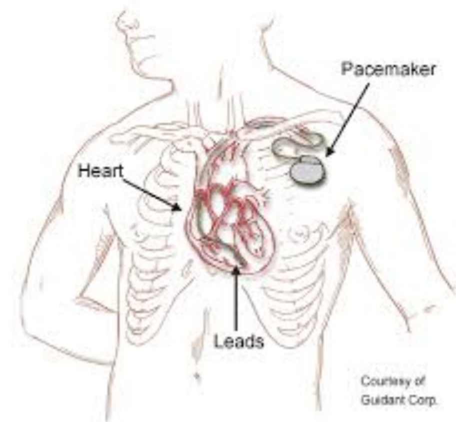
Flexible Intrusion Detection Systems for Memory-Constrained Embedded Systems

Farid Molazem, Karthik Pattabiraman

University of British Columbia

Embedded Systems



Attacks spreading

Attacks spreading

Need an Intrusion Detection System

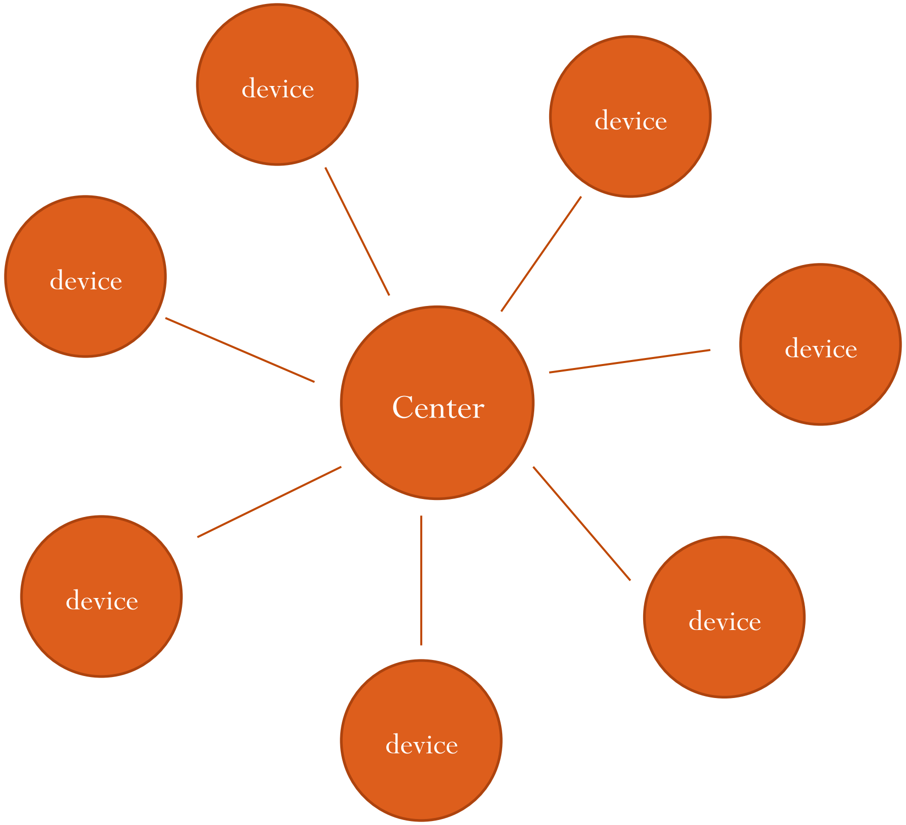
Existing solutions

- Statistical Techniques
  - Neural networks [Moradi et. al.]
  - Hidden Markov Models [Warrender et. al.]
  - Support Vector Machines [Wenjie et. al.]

- Static analysis
  - Call-graph, NDPDA [Wagner et. al.]
  - Dyck [Giffin et. al.]

Challenge

- False positives



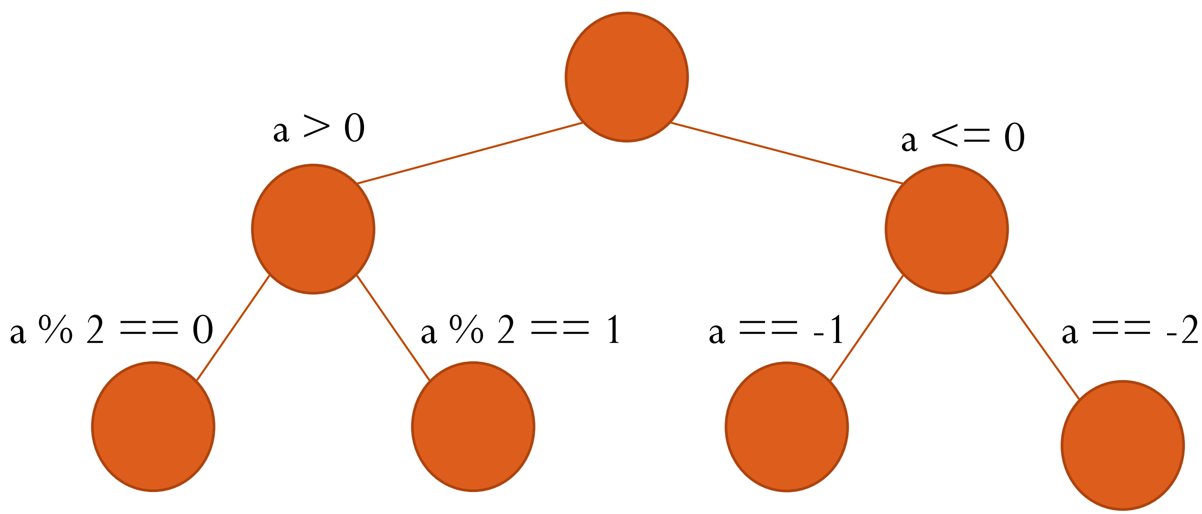
Challenge

- Memory

```
{
  a = receive();
  if (a > 0)
    foo(a);
  else
    bar(a);
}
```

```
void foo(int a) {
  if (a % 2 == 0)
    even(a);
  else
    odd(a);
}
```

```
void bar(int a) {
  if (a == -1)
    error1();
   else if (a == -2)
    error2();
}
```



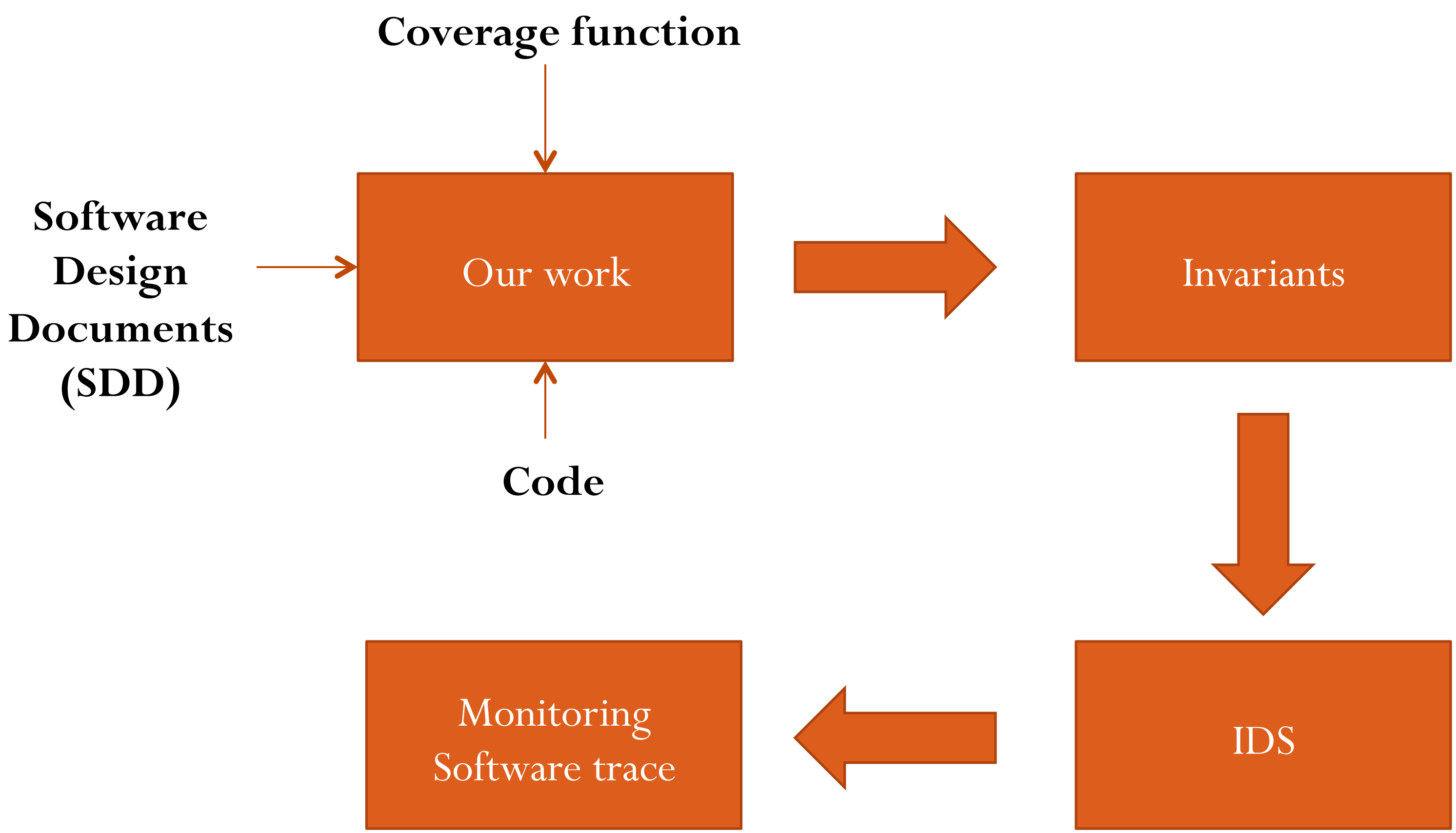
Idea

- Can't fit everything in memory

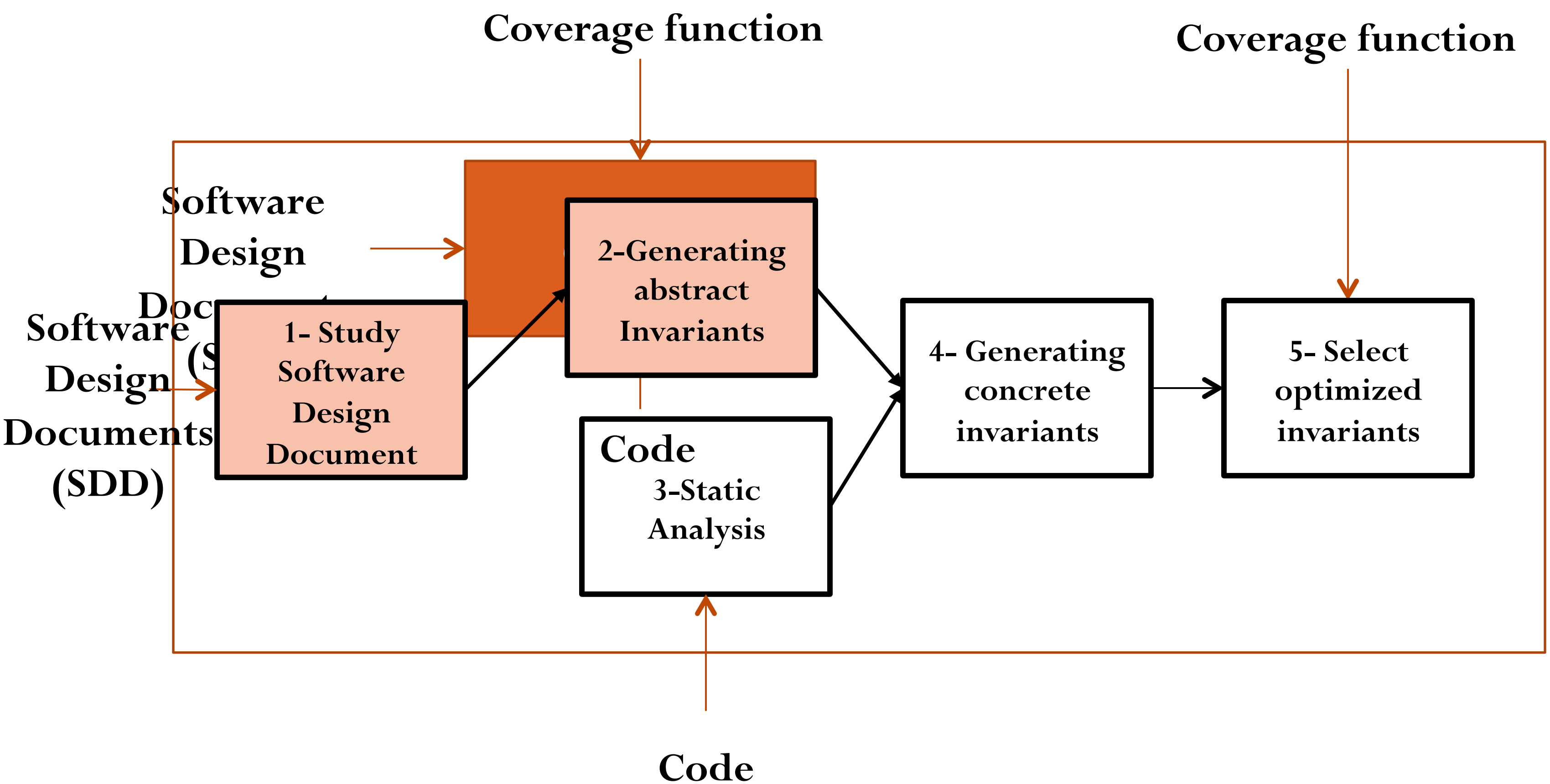
- Quantify security

- Optimize security for the memory we have

Overview

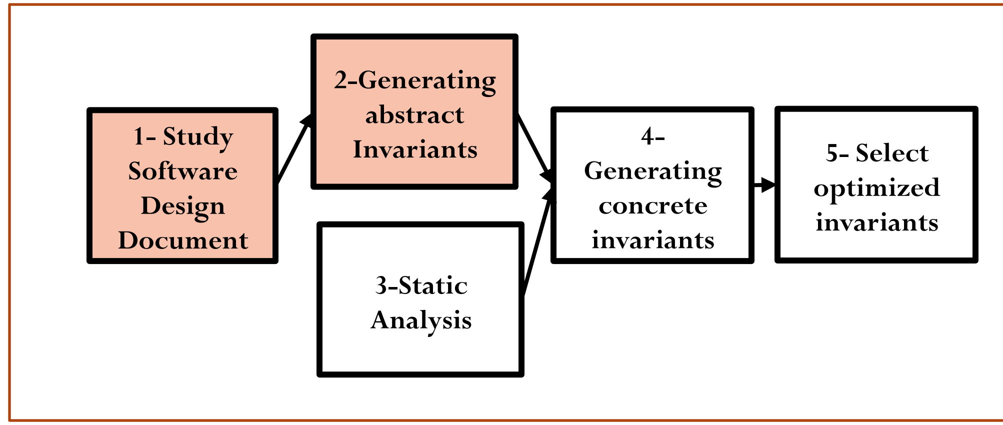

Coverage function
Software
Design
Documents
(SDD)
Our work
Code
Invariants
IDS
Monitoring
Software trace

What we do



Coverage function
Coverage function
Software Design
Software Design Documents (SDD)
1- Study Software Design Document
2-Generating abstract Invariants
Code
3-Static Analysis
4- Generating concrete invariants
5- Select optimized invariants
Code

Steps 1-2



- Storage/Retrieval integrity

Sensor data must eventually be stored on flash memory

$$\square(\mathit{getting\ sensorData} \Longrightarrow (\Diamond \mathit{store\ on\ flash}))$$

Receive sensor data

Temporal Logic

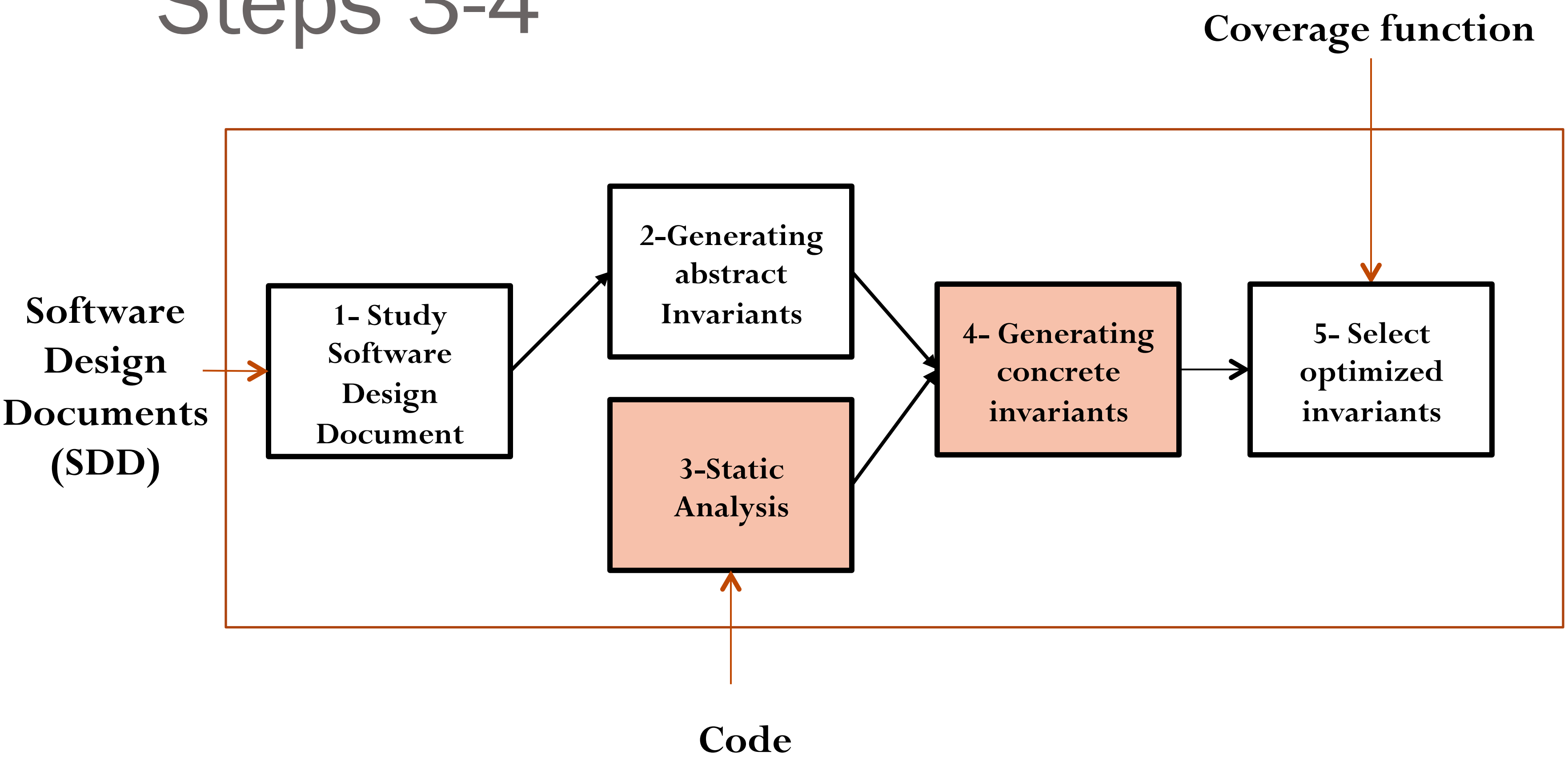
$\Diamond$: Something eventually happens

$\square$: Something always happens

$\bigcirc$: Something happens the next time

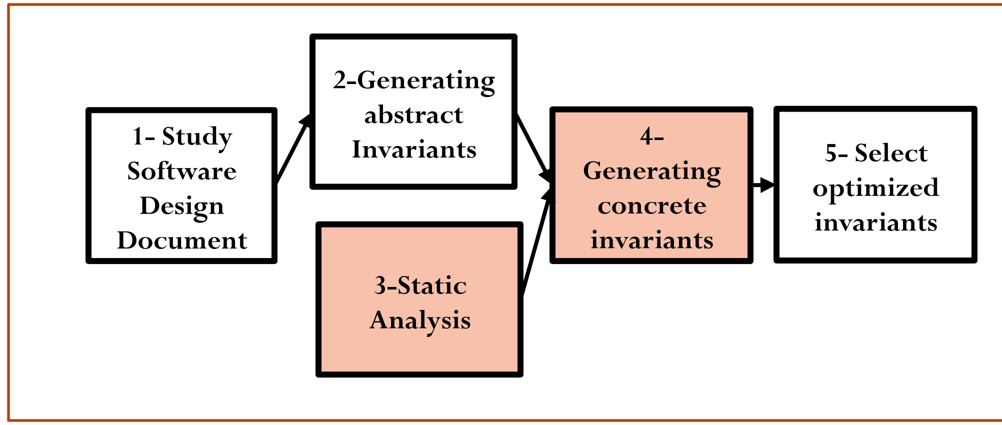
Store on flash memory

Steps 3-4



Abstract invariants → Concrete invariants (contain system calls)

Steps 3-4




```
{
    ....
    data = socket.receive();
    ....
}
```

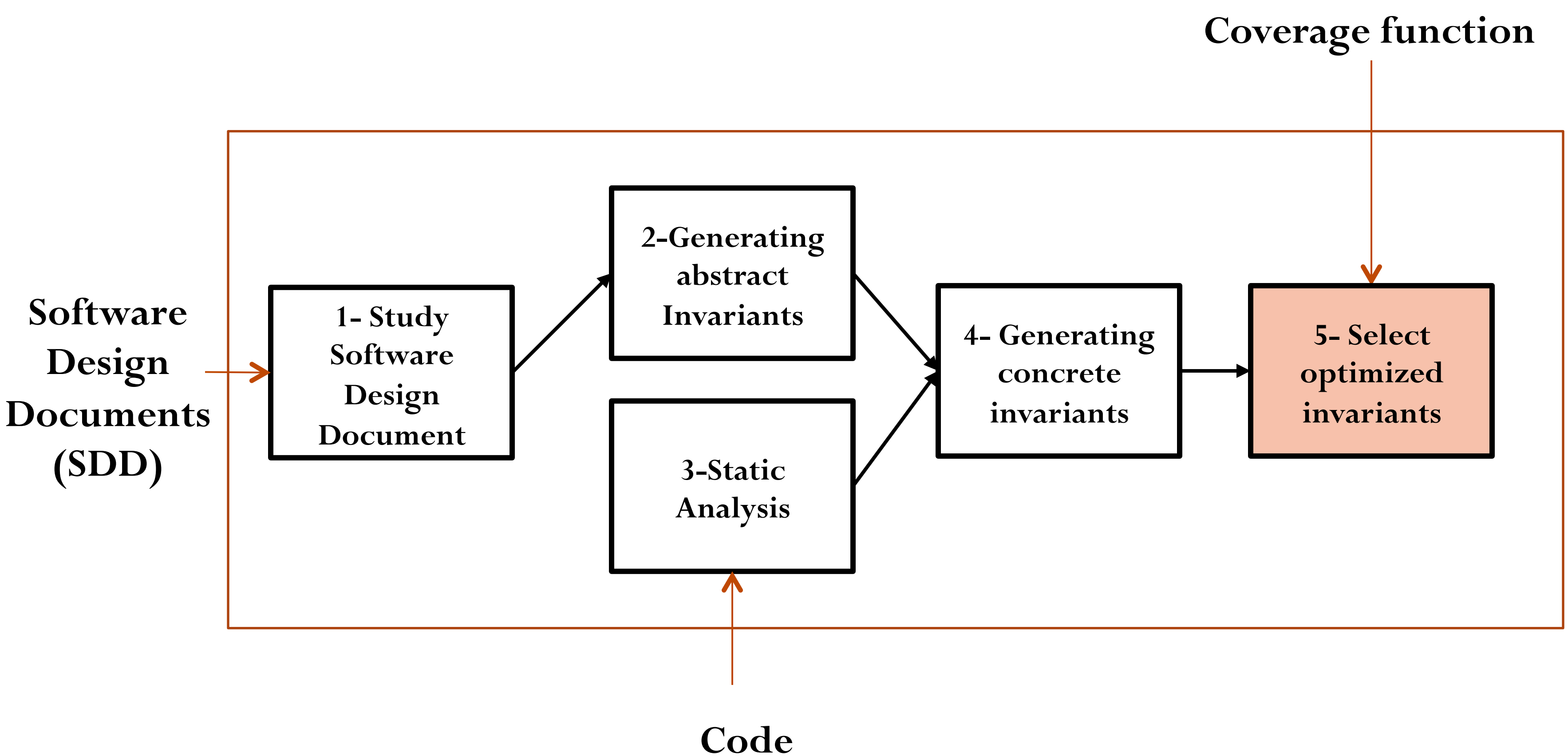
```
{
    ....
    write(f, data);
    ....
}
```

```
...
recv(4, 0x47cf68, 8192, 0)
...
write(1, 0x47cf68, 4) = 4
...
```

$$\square(getting\ sensorData(data) \Longrightarrow (\Diamond store\ on\ flash(data)))$$

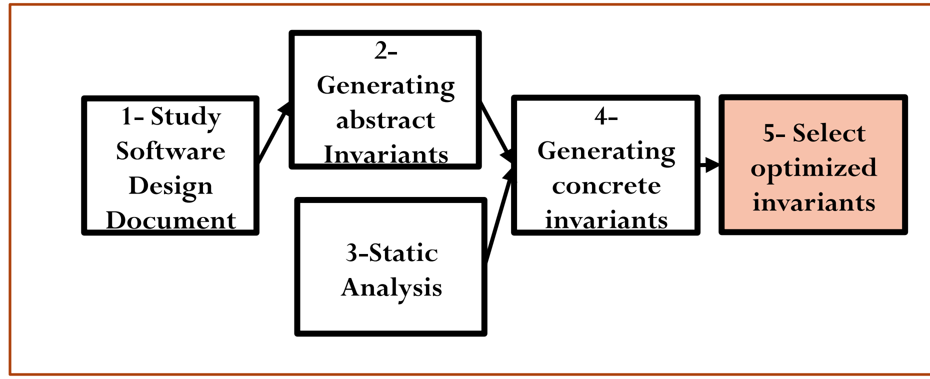
$$\square(receive(d) \Longrightarrow (\Diamond write(d)))$$

Step 5

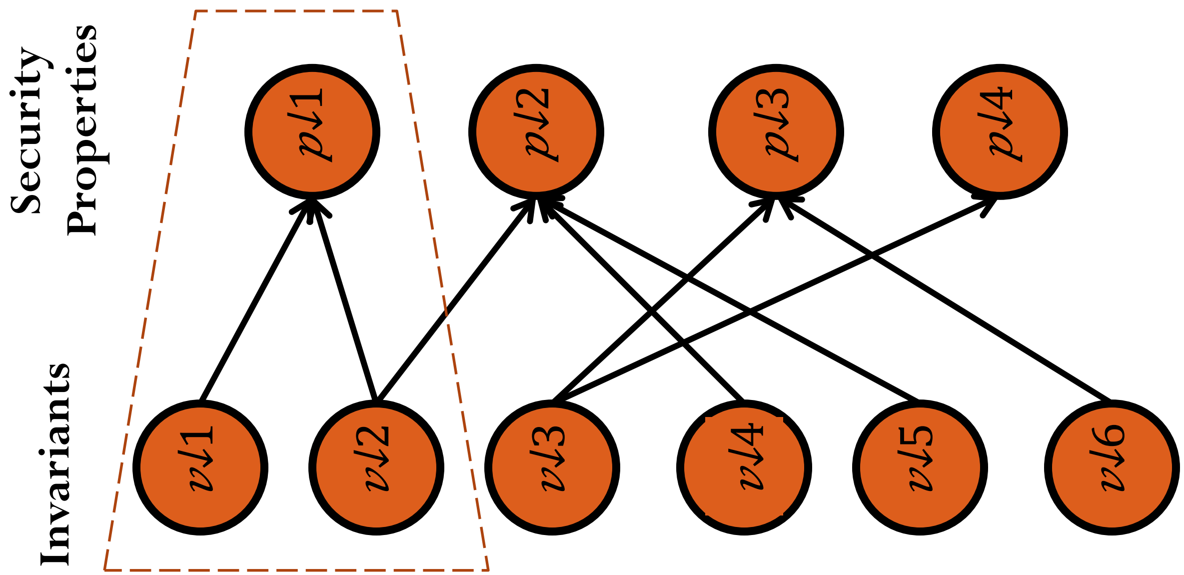


Coverage function
Software Design Documents (SDD)
1- Study Software Design Document
2-Generating abstract Invariants
3-Static Analysis
4- Generating concrete invariants
5- Select optimized invariants
Code

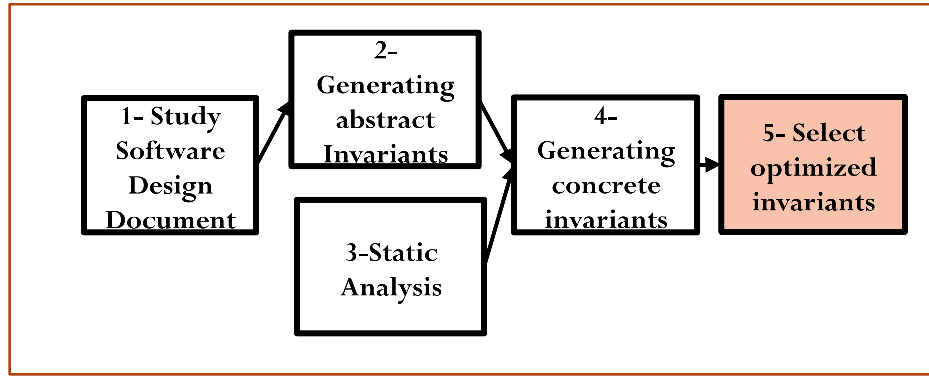
Coverage, Example



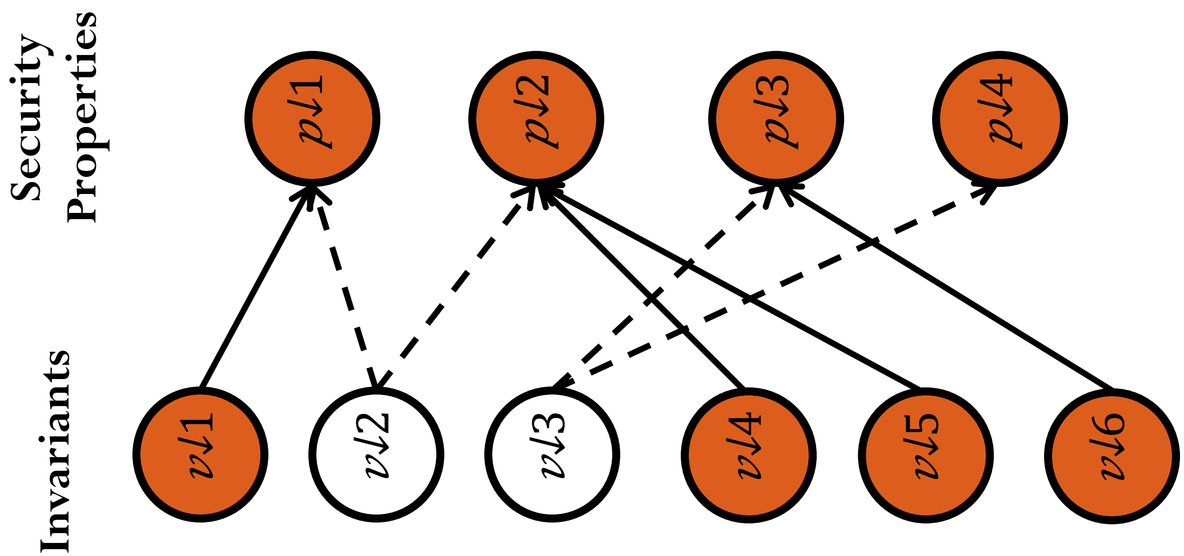
Define a coverage function on the graph and maximize it



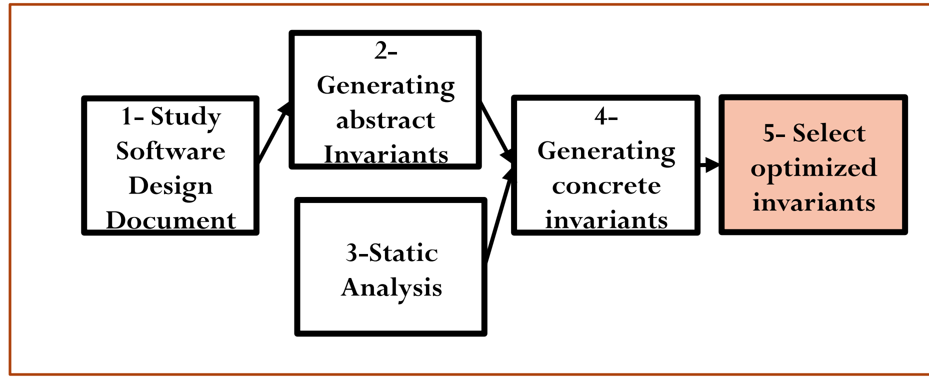
Coverage, Example



Define a coverage function on the graph and maximize it

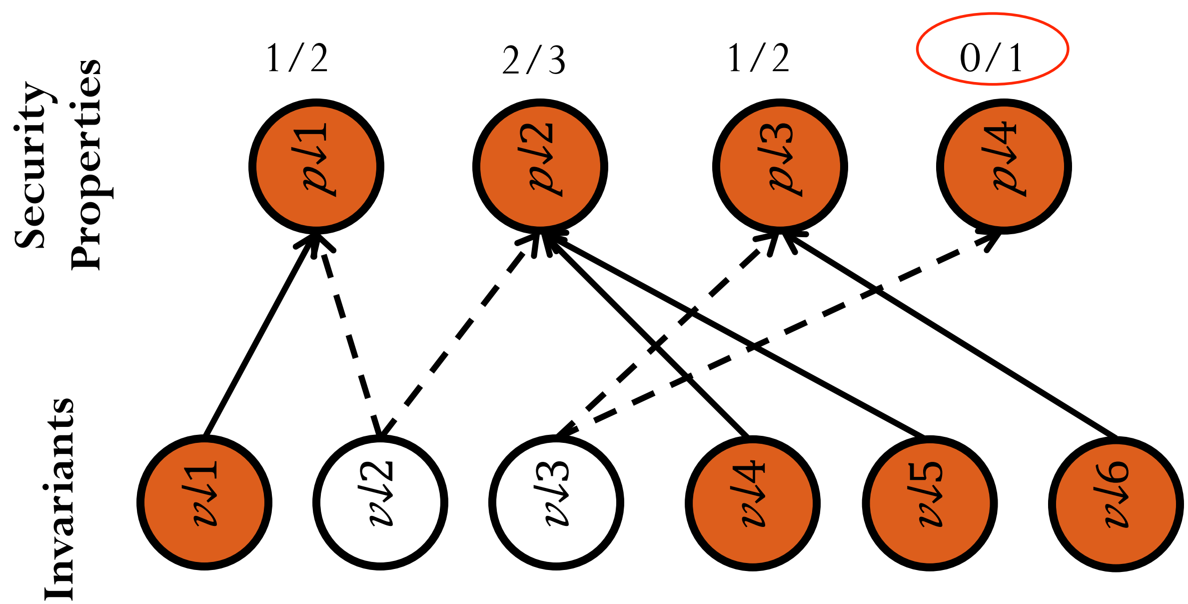


Coverage, Example

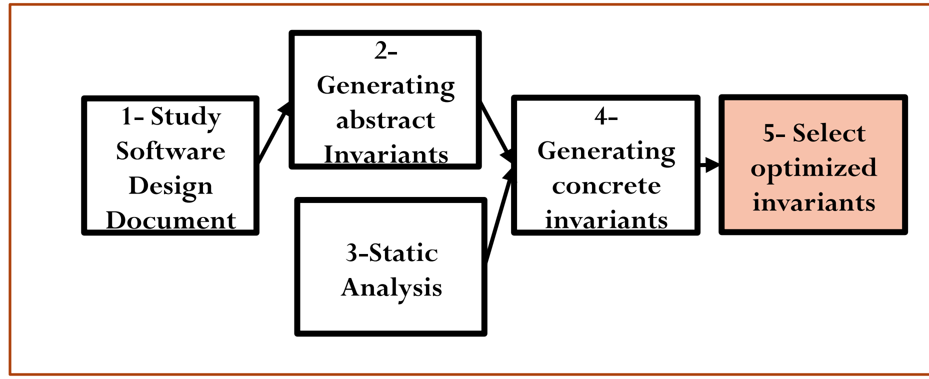


MaxMin Coverage IDS:

Intuition: Make the weakest coverage as strong as possible

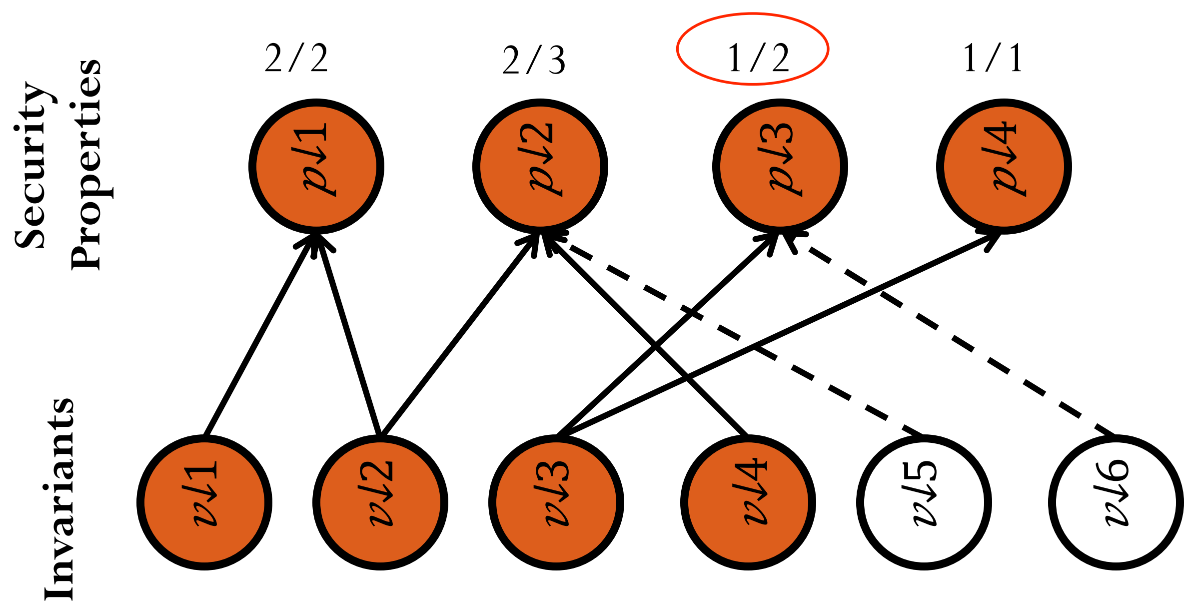


Coverage, Example



MaxMin Coverage IDS:

Intuition: Make the weakest coverage as strong as possible



Building the IDS

Select the invariants from the graph

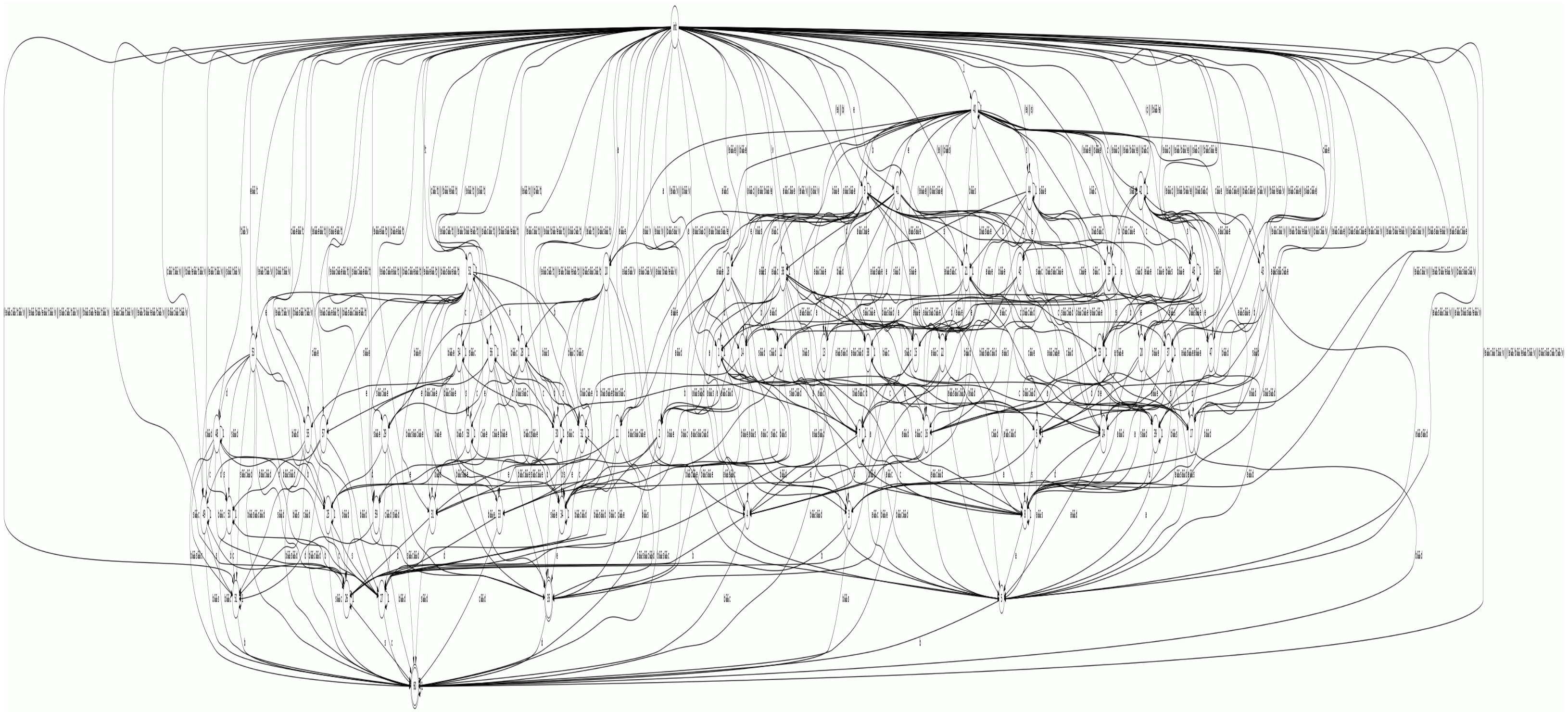
↓

Automatically convert it to Buchi Automaton

Iterative process, optimize for memory.

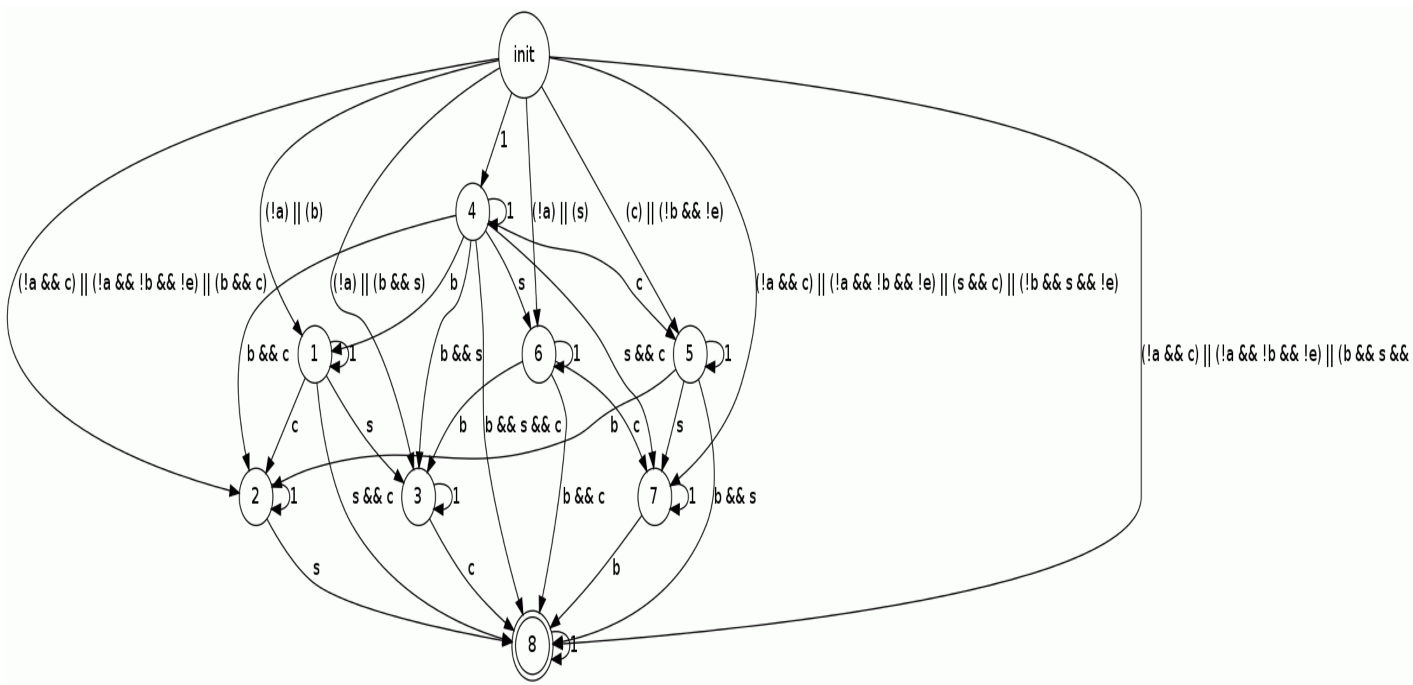
Building the IDS

(a -> F b) && (a -> F s) && (b -> F c) && (e -> F c) && (t -> F (a && X b)) && (v -> F (e && X d)) && (s -> F d) && (s -> F e) && (s -> F w)



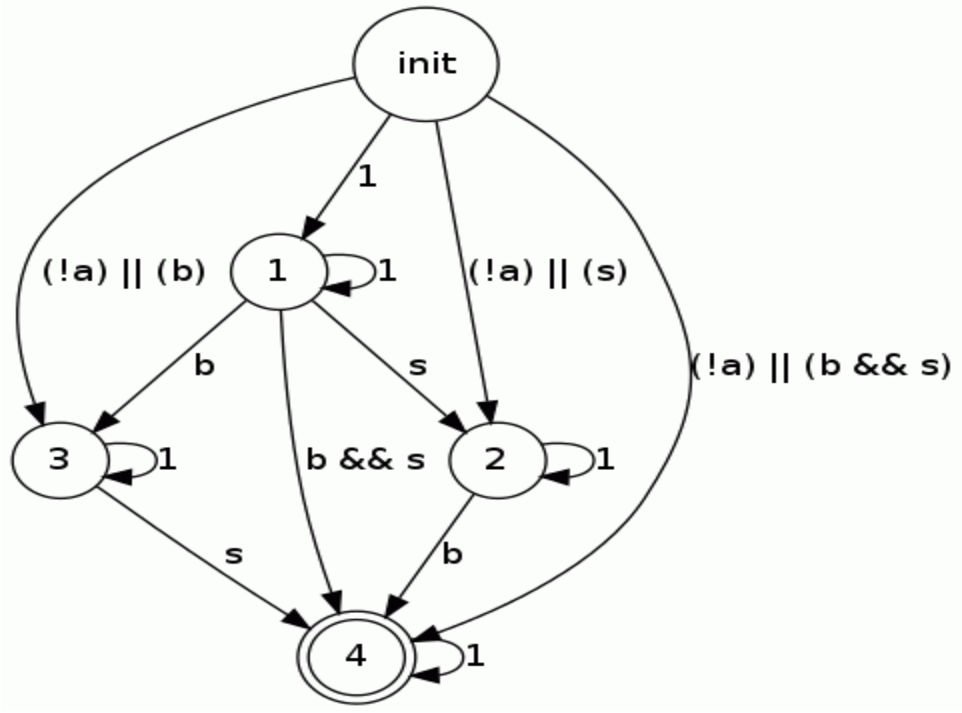
Building the IDS

(a -> F b) && (a -> F s) && (b -> F c) && (e -> F c) && (t -> F (a && X b))



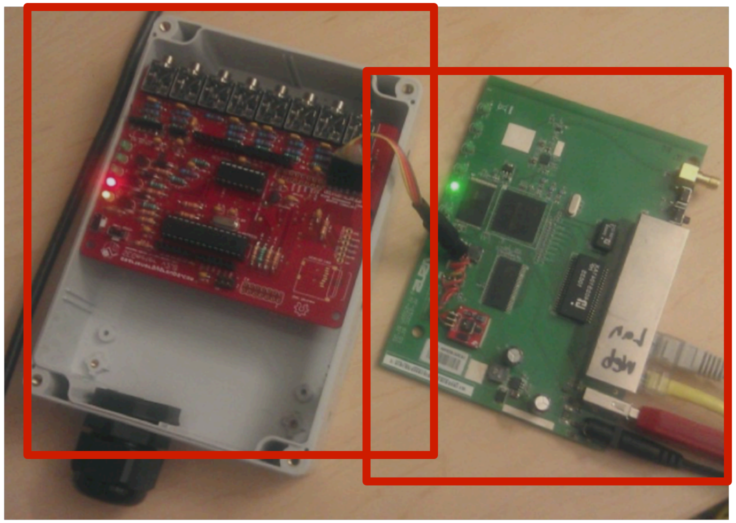
Building the IDS

(a -> F b) && (a -> F s)



Evaluation

- SegMeter: Smart meter, an important device used in smart homes
- Meter:
  - Arduino board
    - ATMEGA 32x series microcontroller
    - Sensors
  - Gateway board
    - Broadcom BCM 3302 240MHz CPU
    - 16 MB RAM
    - **4 MB available for IDS**
    - OpenWRT Linux
  - IDS runs on the Gateway board
  - No attack database available => We use fault injection to simulate attacks

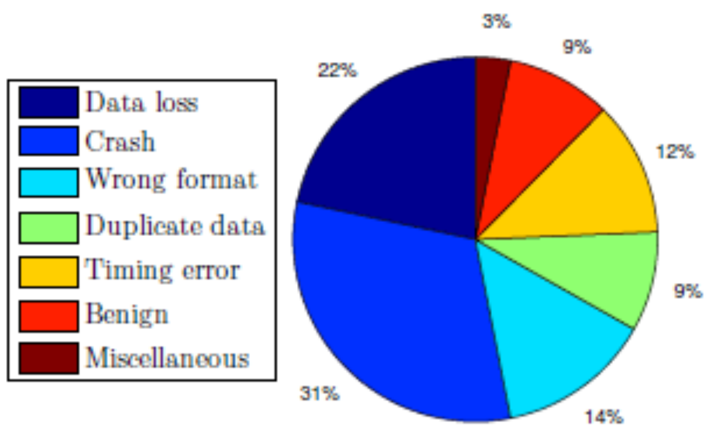


Fault injection

- Flipping branches

```
if (data_file ~= nil) then
   big_string = data_file:read("*all")
   ...
end
```

```
if (data_file == nil) then
   big_string = data_file:read("*all")
   ...
end
```



Research questions

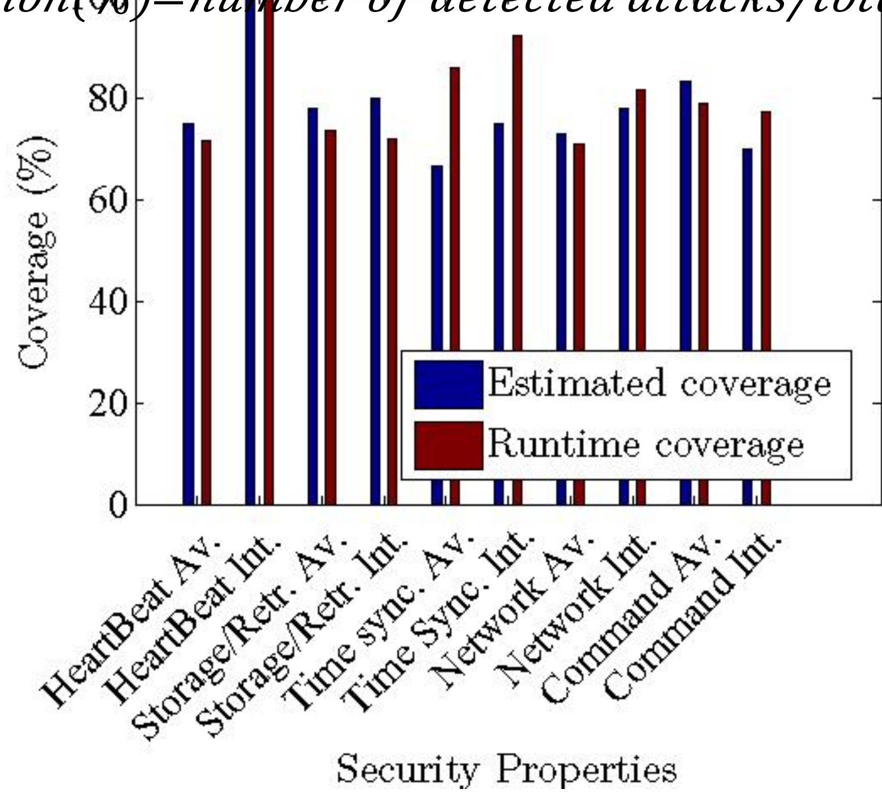
- How close is the estimated coverage at design time to the coverage at run-time?
  - Shows whether the theoretical optimization is useful

- What is the performance overhead
  - Shows whether it is practical to implement and use

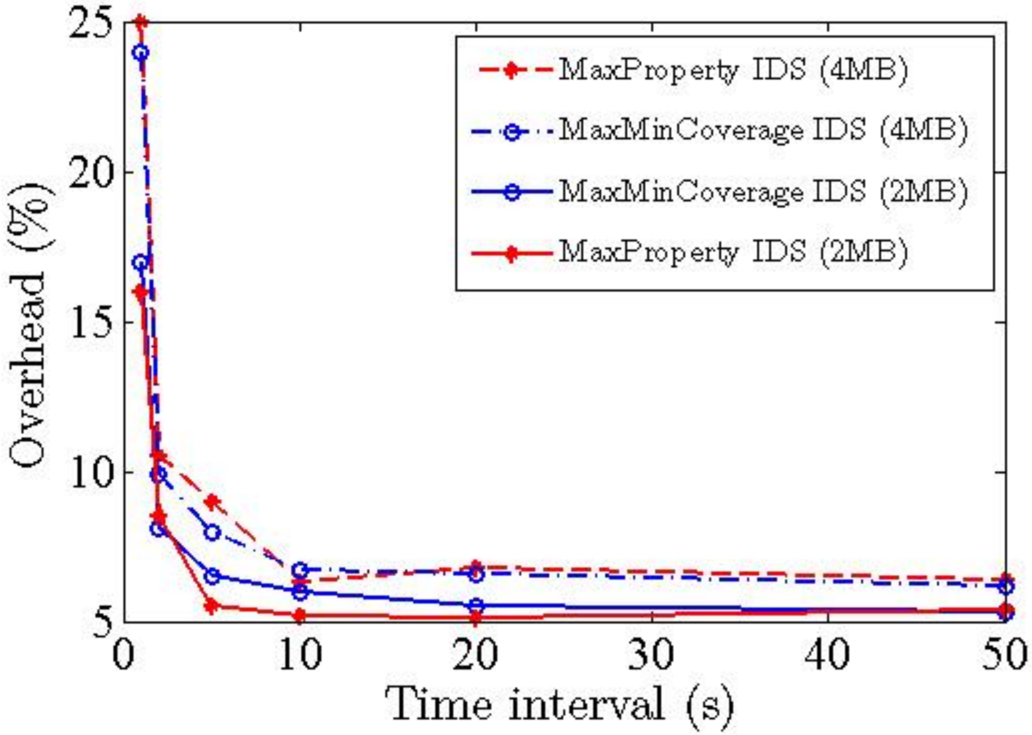
Results (MaxMin IDS)

- How good is the coverage of the IDS?
- How good the graph-based optimization is reflected at run-time ?

detection(%)=number of detected attacks/total umber of injected atta



Performance overhead

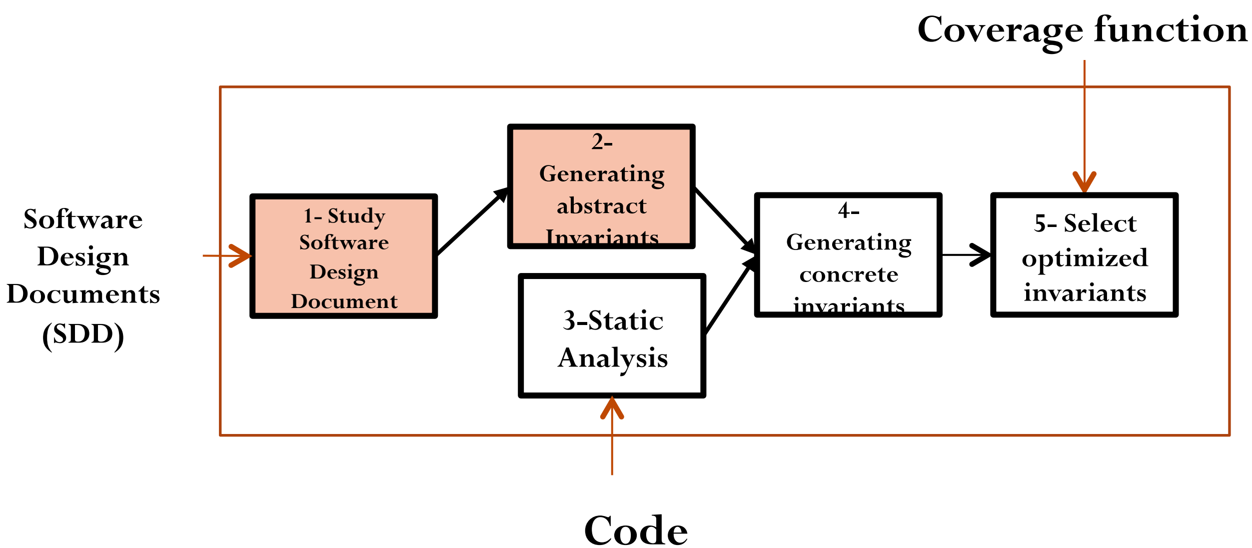


Discussions

- Quantifiable coverage provides flexibility
- How to pick a coverage function?
- Having a complete software engineering life cycle can help producing automated security solutions

Conclusions

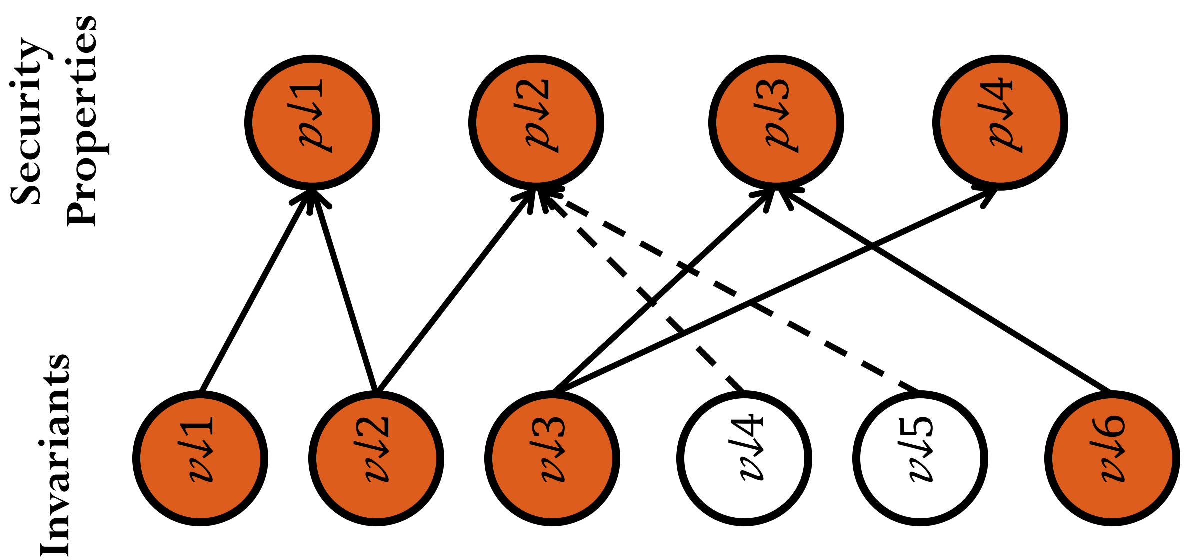
- Traditional solutions don't work
- **We can quantify security**
- We can use different security measurement functions



faridm@ece.ubc.ca

Coverage, Example 2

MaxProperty IDS:
Maxmize security properties that are fully covered



Results (MaxProperty IDS)

- How good is the coverage of the IDS?
- How good the graph-based optimization is reflected at run-time?