Diagnosing Memory Leaks using Graph Mining on Heap Dumps

Evan K. Maxwell, Godmar Back, Naren Ramakrishnan

Virginia Tech, Department of Computer Science

July 27, 2010

KDD 2010 — Washington, D.C.

Memory Leaks

```
Exception in thread "main":  java.lang.OutOfMemoryError
```

Effects:

- Exhaustion of heap memory
- Application slowdowns and crashes

Detection difficulties:

- Leak source $\neq$ observed failure
- Heap reachability graph
- Large heap size

Memory Leaks

```
Exception in thread "main":  java.lang.OutOfMemoryError
```

Effects:

- Exhaustion of heap memory
- Application slowdowns and crashes

Detection difficulties:

- Leak source $\neq$ observed failure
- Heap reachability graph
- Large heap size

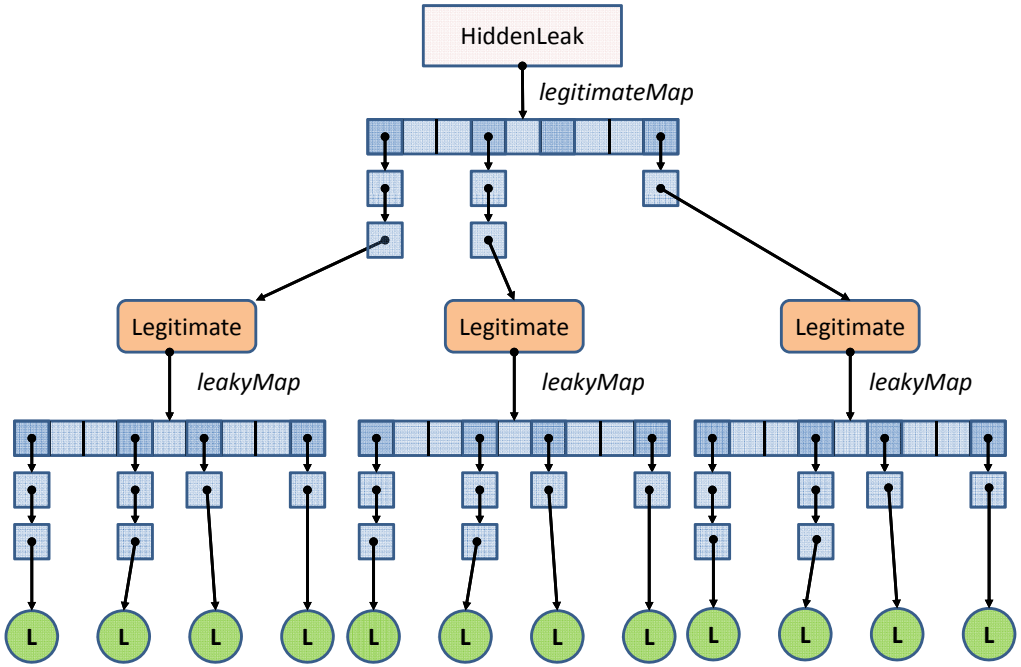
Existing methods:

- Tools:
  - Eclipse MAT[1]
  - IBM's LeakBot[2]
- Static (post-mortem)[1] vs. dynamic (runtime)[2] heap analysis
- Object size

Heap Reachability Graph

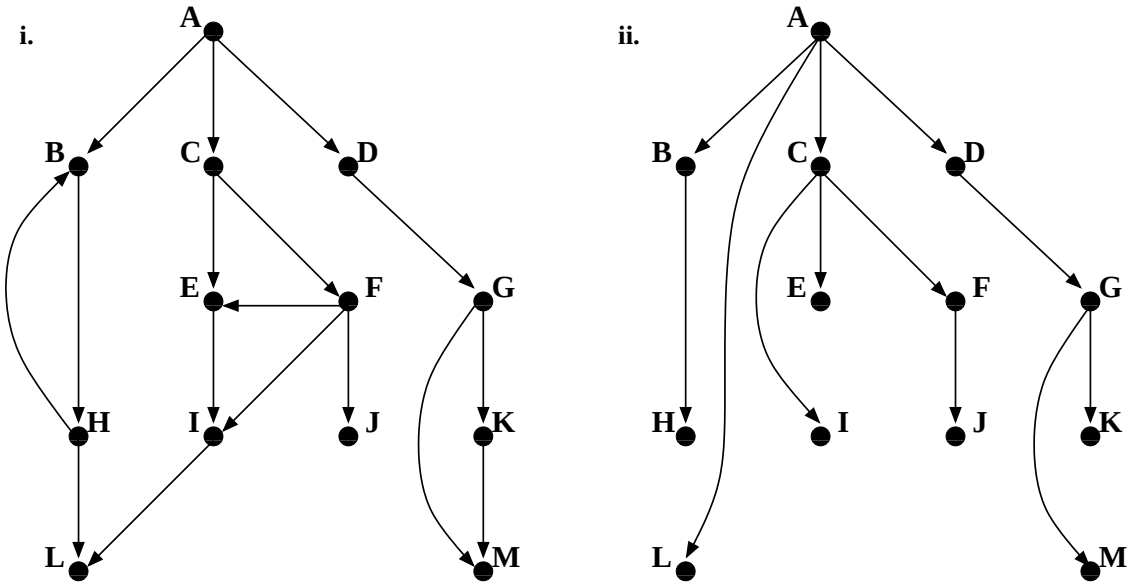
```
public class HiddenLeak
{
  static HashMap legitimateMap;

  static class Legitimate
  {
    HashMap leakyMap = new HashMap();

    static class Leak
    {
      // This object is leaked
    }

    void leak()
    {
      // insert Leak instances
      // into leakyMap
    }
  }

  public static void main(String []av)
  {
     // insert Legitimate instances
     // into legitimateMap
  }
```

HiddenLeak.java



Compute Dominator Tree

- Edges indicate object ownership/liveness
- 44-61% edge reduction



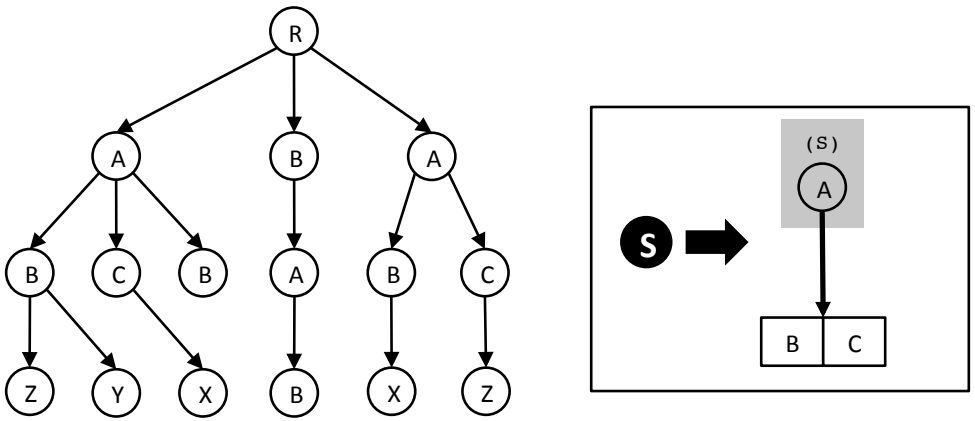
(i) An example graph and (ii) its dominator tree.

Our Approach

1. Leaks manifest as frequent subgraphs.
2. Dominator tree shows object liveness.
3. Frequent subgraphs equivalent from graph to dominator tree.
4. Root path summarization shows objects' location.

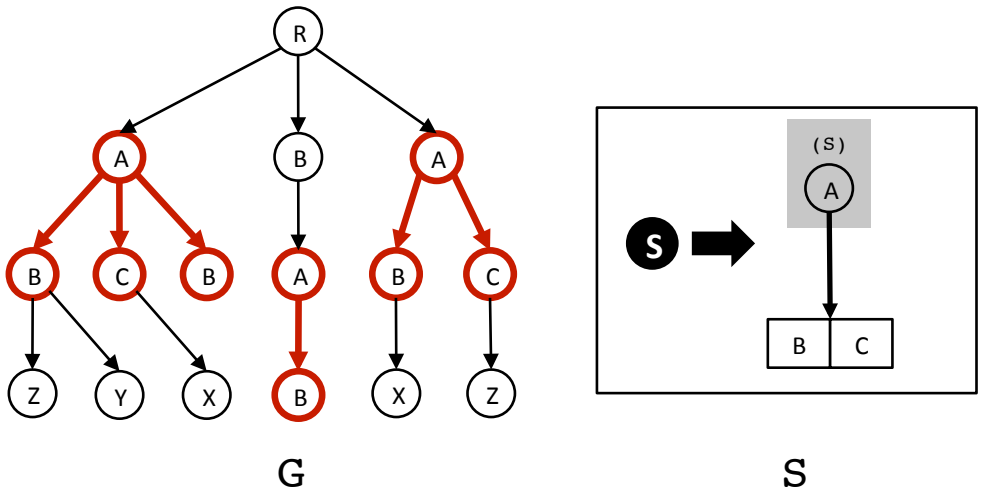
Graph Grammar Mining

- Recursive constructs → data structure usage.
- Variable labels → inexact pattern matching.



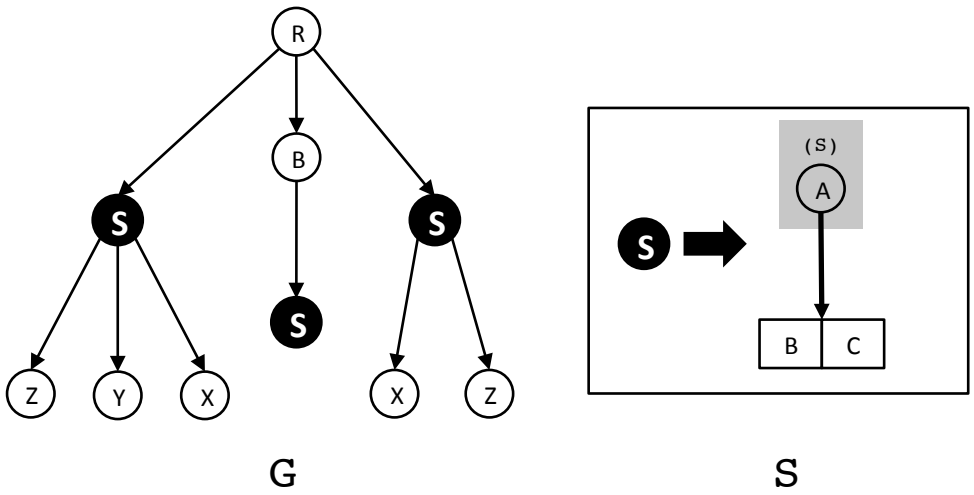
Graph Grammar Mining

- Recursive constructs → data structure usage.
- Variable labels → inexact pattern matching.



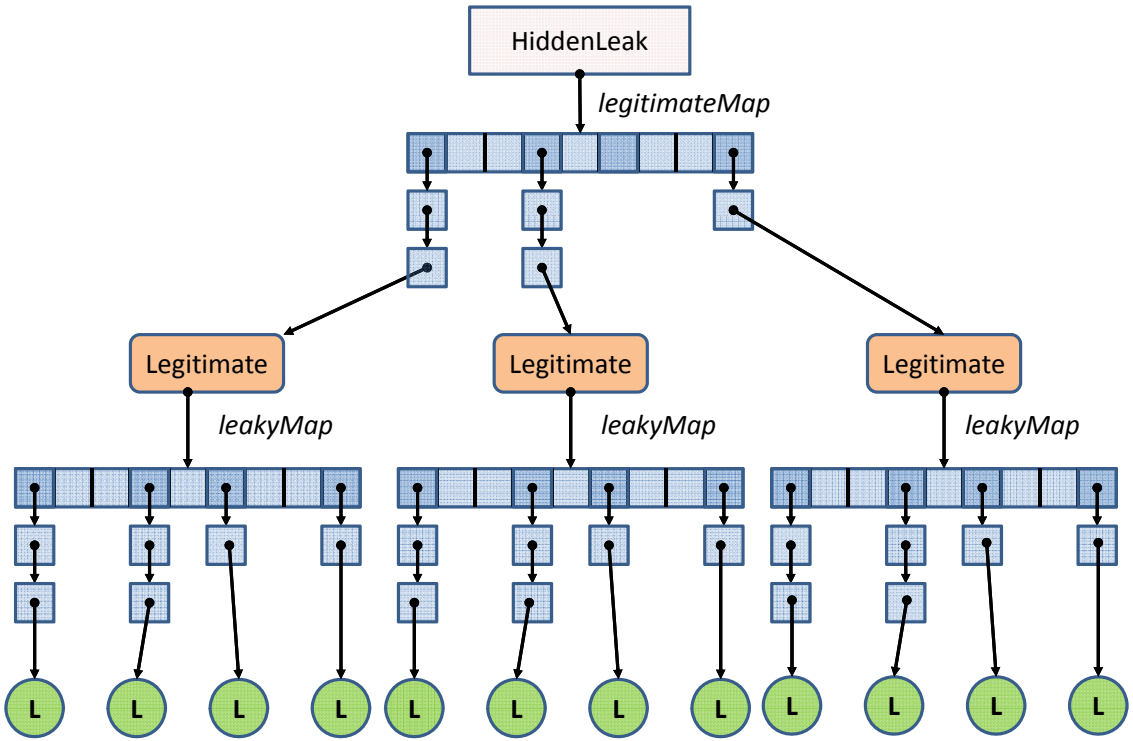
Graph Grammar Mining

- Recursive constructs → data structure usage.
- Variable labels → inexact pattern matching.

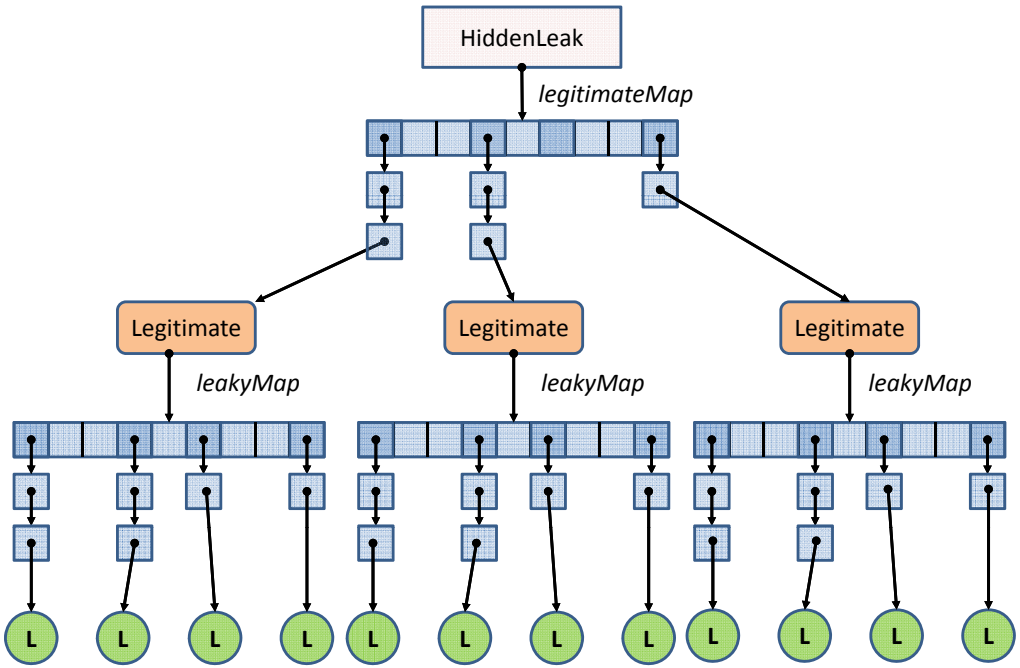


Root Path Analysis

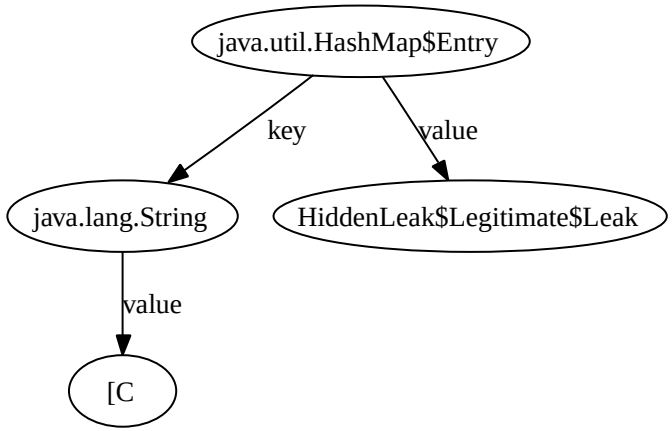
- Summarization of object ownership
- Provides context



HiddenLeak.java



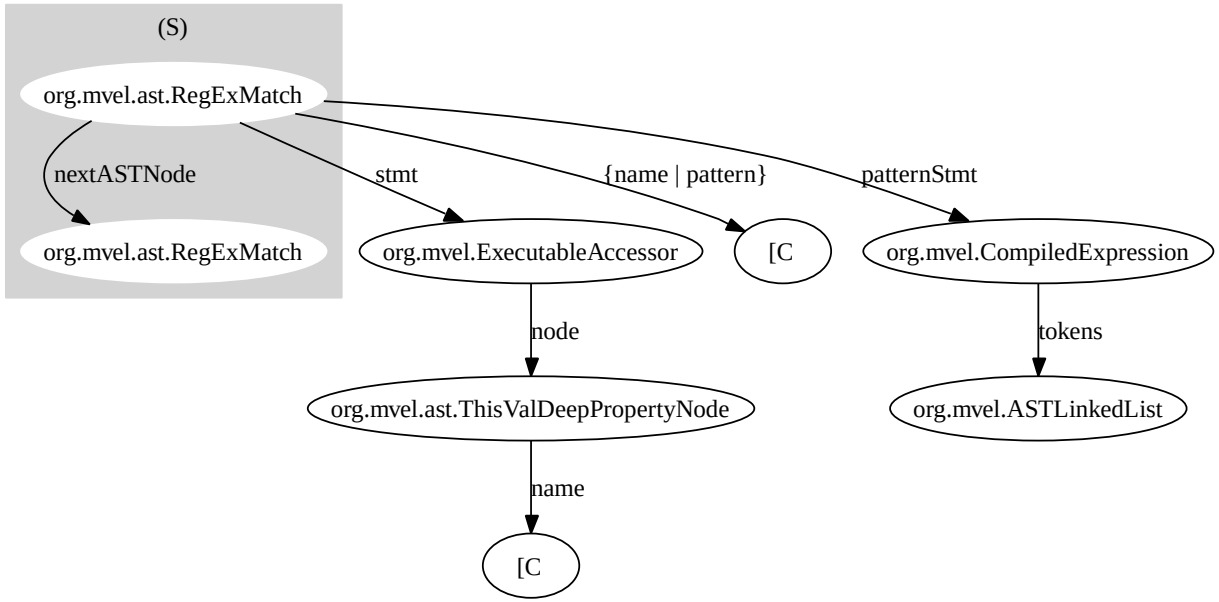

```
java.lang.Class
| (legitimateMap)
+-+-> java.util.HashMap
| (table)
+-+-> [Ljava.util.HashMap$Entry;
| ($array$)
+-+-> java.util.HashMap$Entry
| (value)
+-+-> HiddenLeak$Legitimate
| (leakyMap)
+-+-> java.util.HashMap
| (table)
+-+-> [Ljava.util.HashMap$Entry;
| ($array$)
+-+-> java.util.HashMap$Entry
```



MVEL Parser Leak

```
+-+-> Java_Local
| (??)
+-+-> org.mvel.ASTLinkedList
| (firstASTNode)
{
   +-+-> org.mvel.ast.RegExMatch
   | (nextASTNode)
   +-+-> org.mvel.ast.RegExMatch
}*
```

- ∼1.7 million objects.
- ∼4 million references.
- Manual inspection is impractical.



Contributions

Graph Mining:

- Dominator tree for graph reduction.

Memory Leak Detection:

- First static anaylsis method using graph mining.
- Graph grammars indicate data structures.
- Subgraph frequency + root path summarization = context.

Thank You