departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Concurrency and Parallelism
# *(Concorrência e Paralelismo – CP 11158)*

## Lecture 5
— Atomicity Violations —

*João Lourenço <joao.lourenco@fct.unl.pt>*

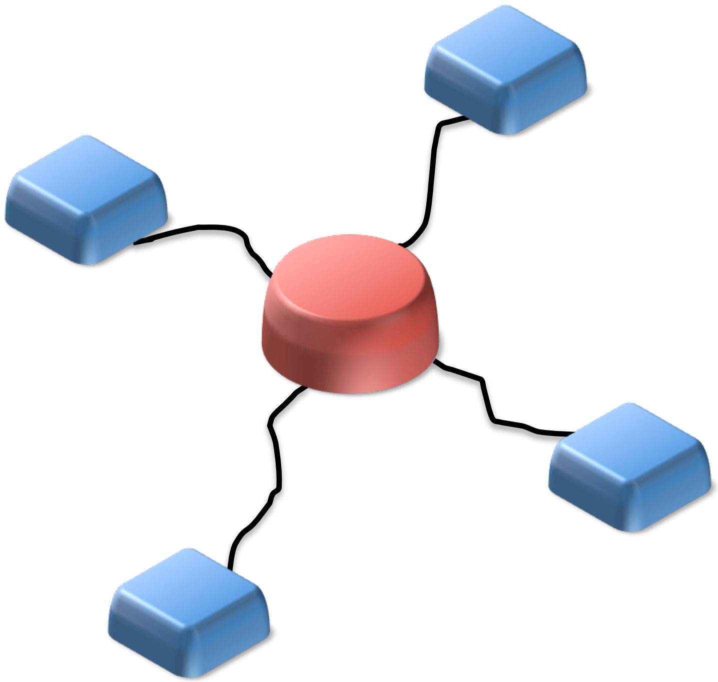
# Single-Threaded Process

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Multi-Threaded Process

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

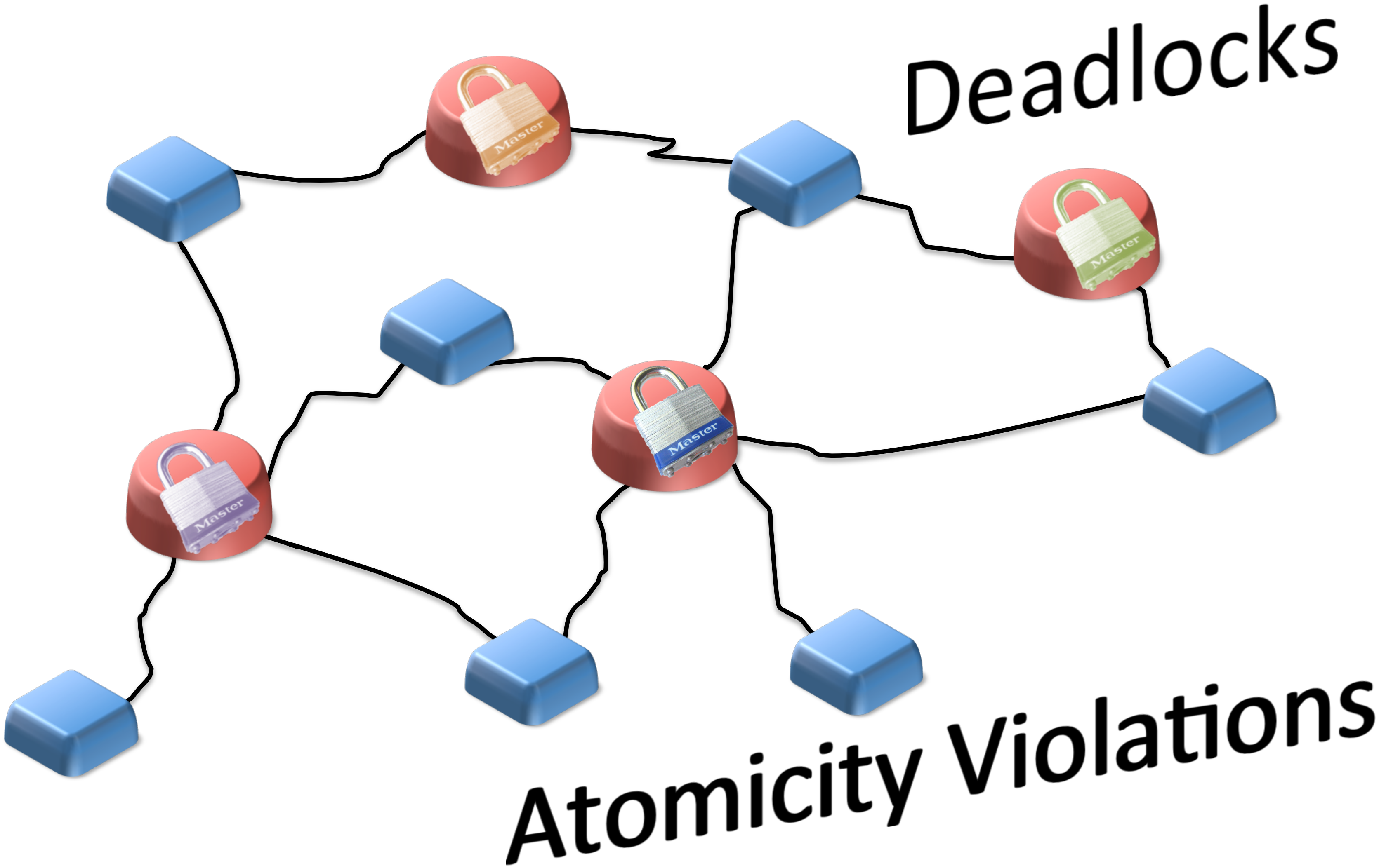
# Sharing Resources



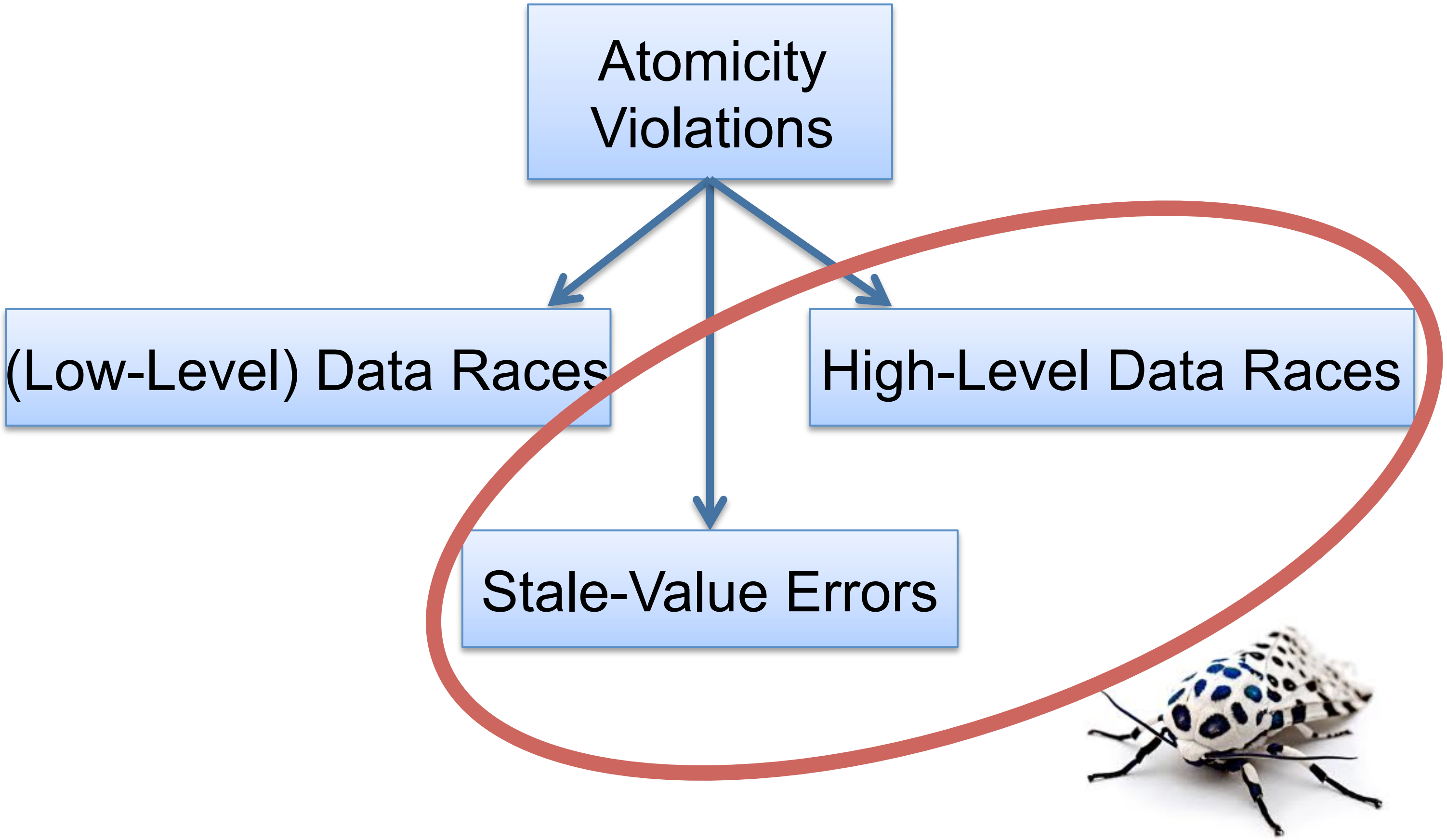
# Sharing Resources

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

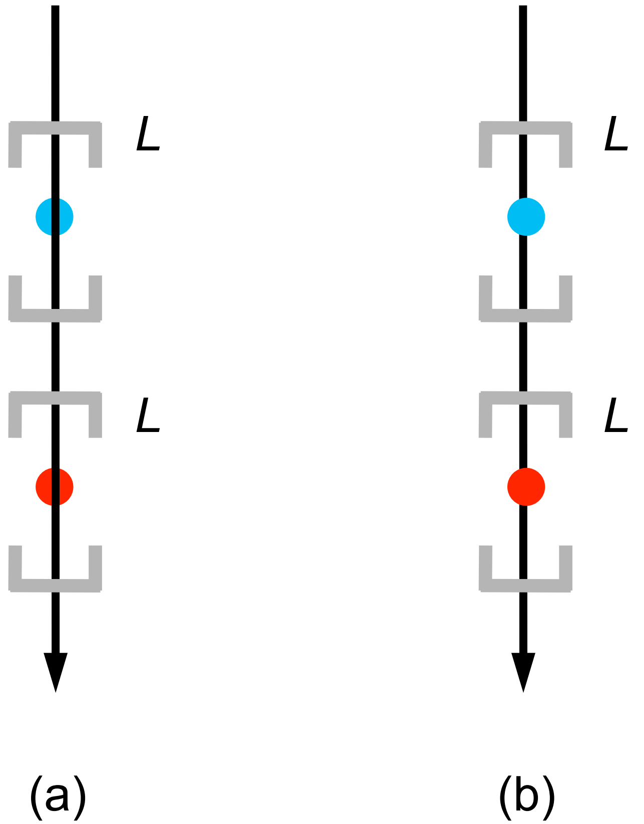
# Sharing Resources



# Concurrency Anomalies

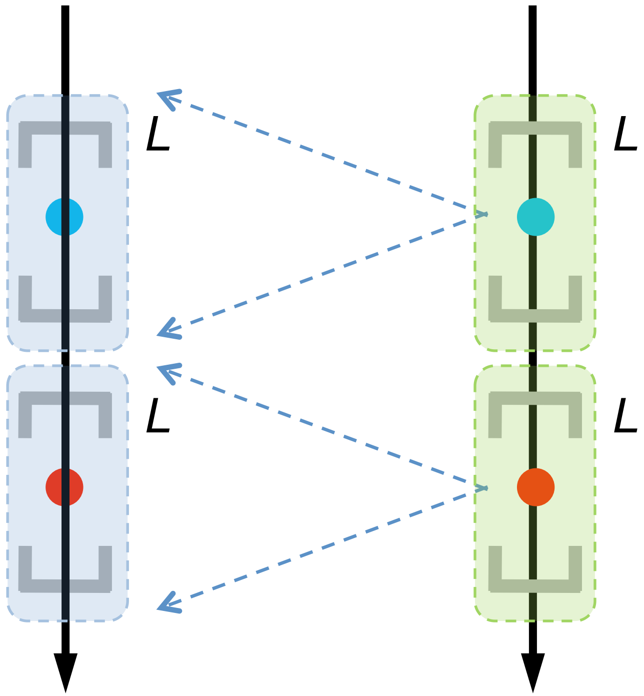


# A Simple Example




```
int counter;

Void increment(){
    int temp;

    lock (L)
    temp = counter;
    unlock (L)

    temp++;

    lock (L);
    counter = temp;
    unlock (L);
}
```

# A Simple Example of an Atomicity Violation

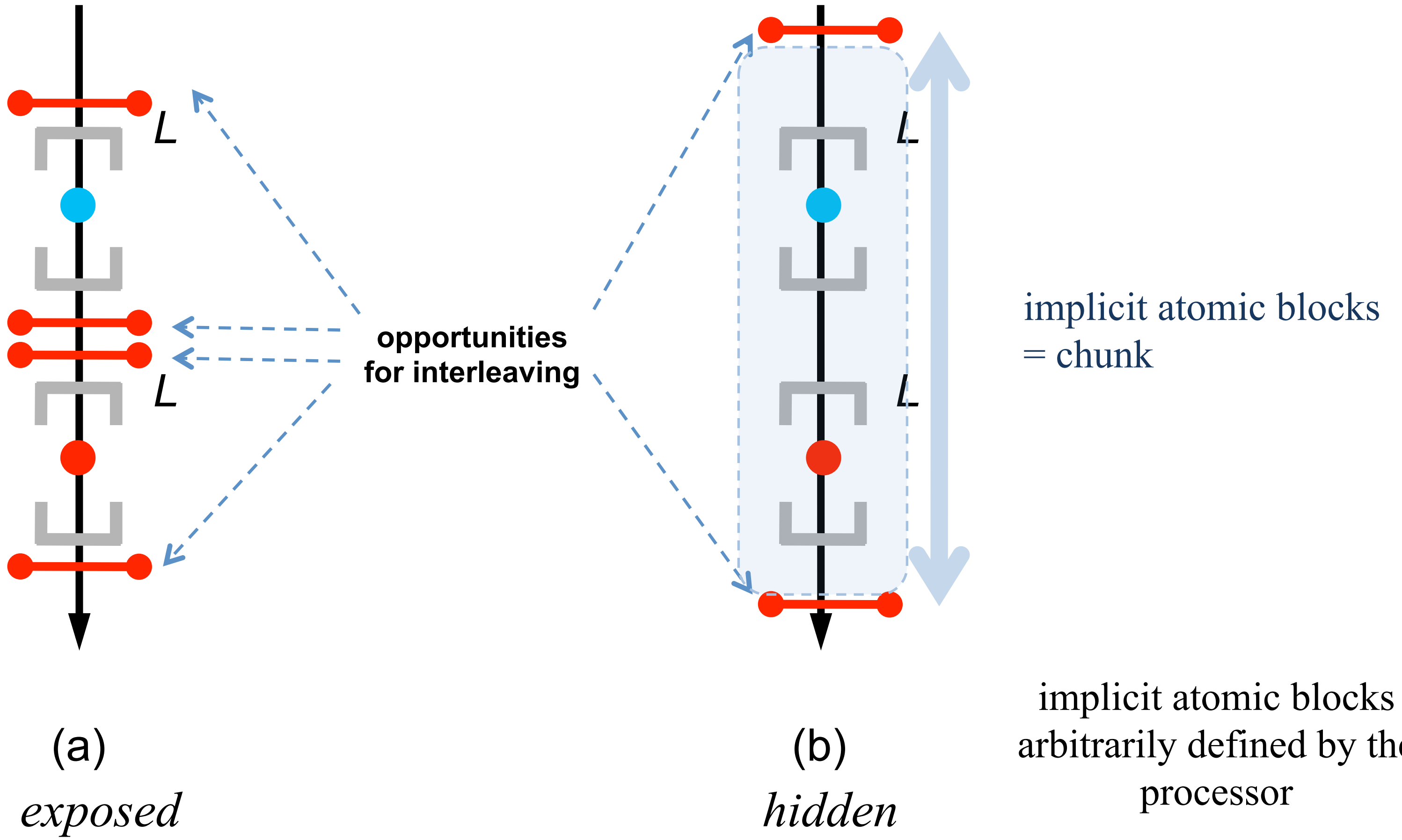



(a) (b)

bad interleavings of remote thread


**Possible Interleavings**

# Opportunities for interleaving




implicit atomic blocks arbitrarily defined by the processor

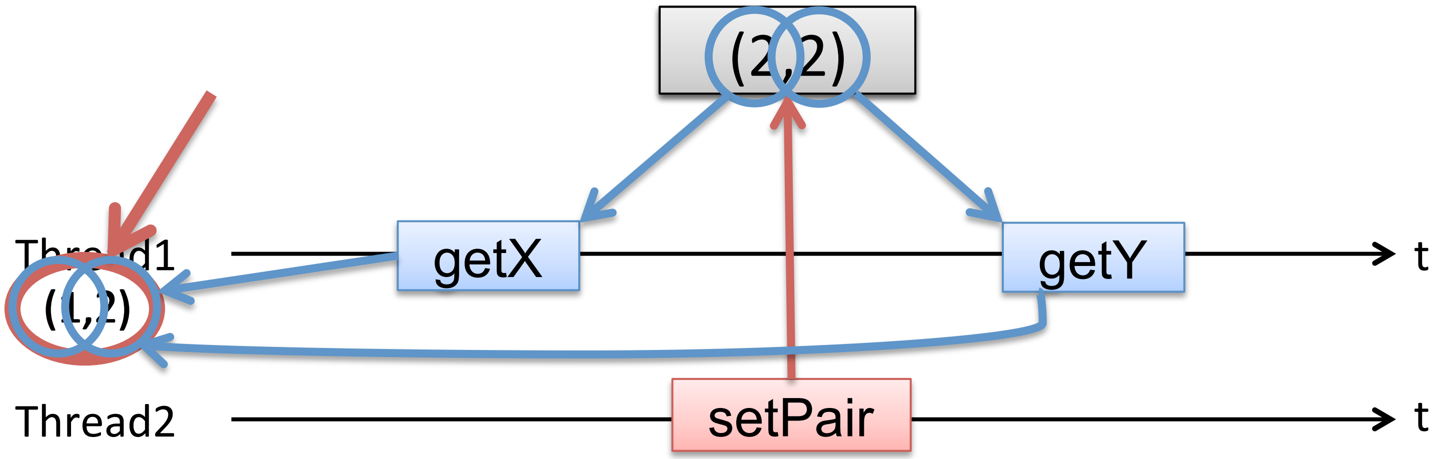
departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# High-Level Data Races

Shared variables: x, y

```
// Thread 1
public boolean equals(){
   int loc_x = getX(); // Atomic
   int loc_y = getY(); // Atomic
   return loc_x == loc_y;
}
```

```
// Thread 2
@Atomic
public int setPair(int v1, int v2){
      x = v1;
      y = v2;
}
```



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Stale-Value Errors

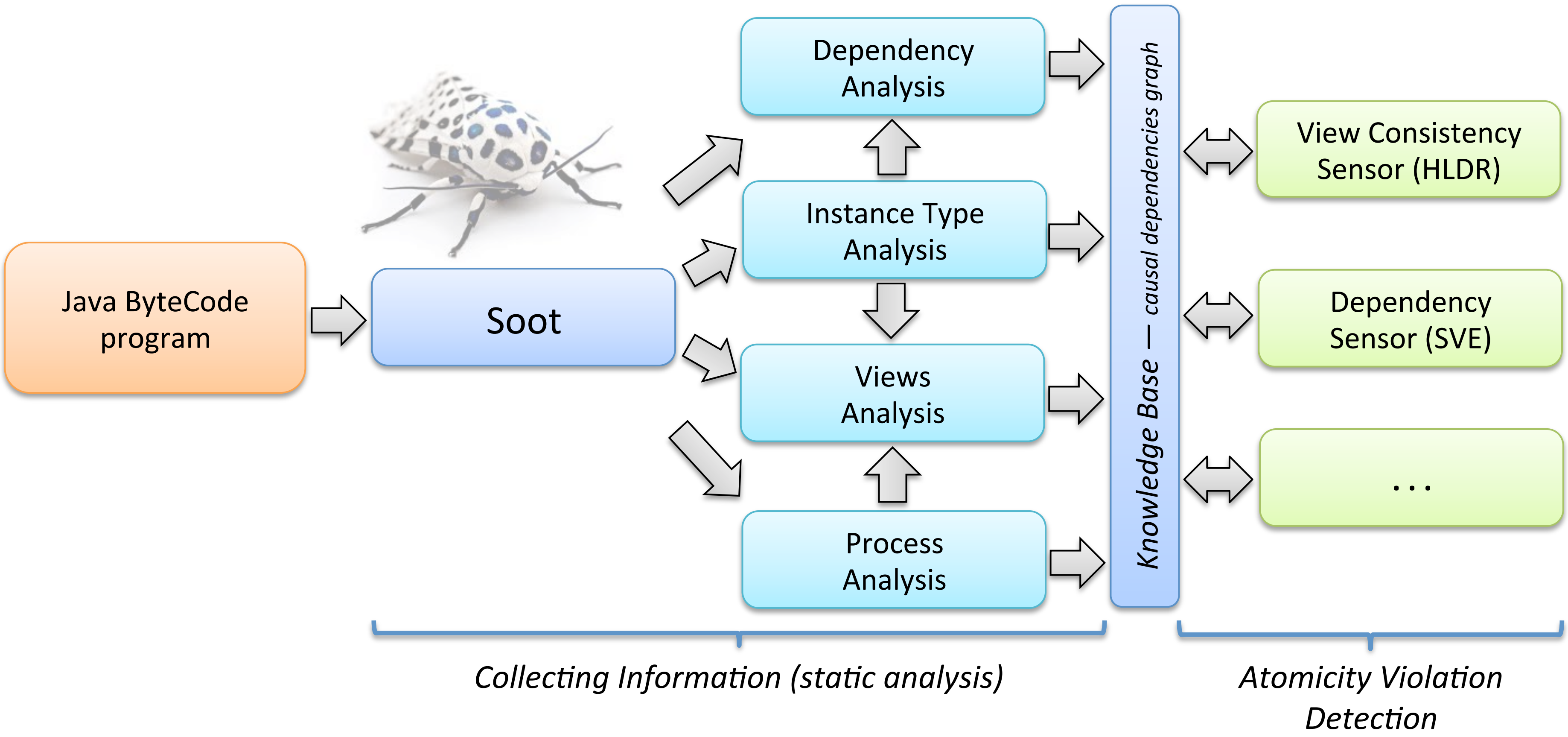
Shared variables: x, y

```
void yEqualsXTimesTwo () {
    int local = getX(); // Atomic
    // local may have a stale value
    setY(2 * local) ; // Atomic
}

@Atomic
private int getX() {
    return x ;
}

@Atomic
private void setY(int value) {
    y = value ;
}
```

x

write(x)?

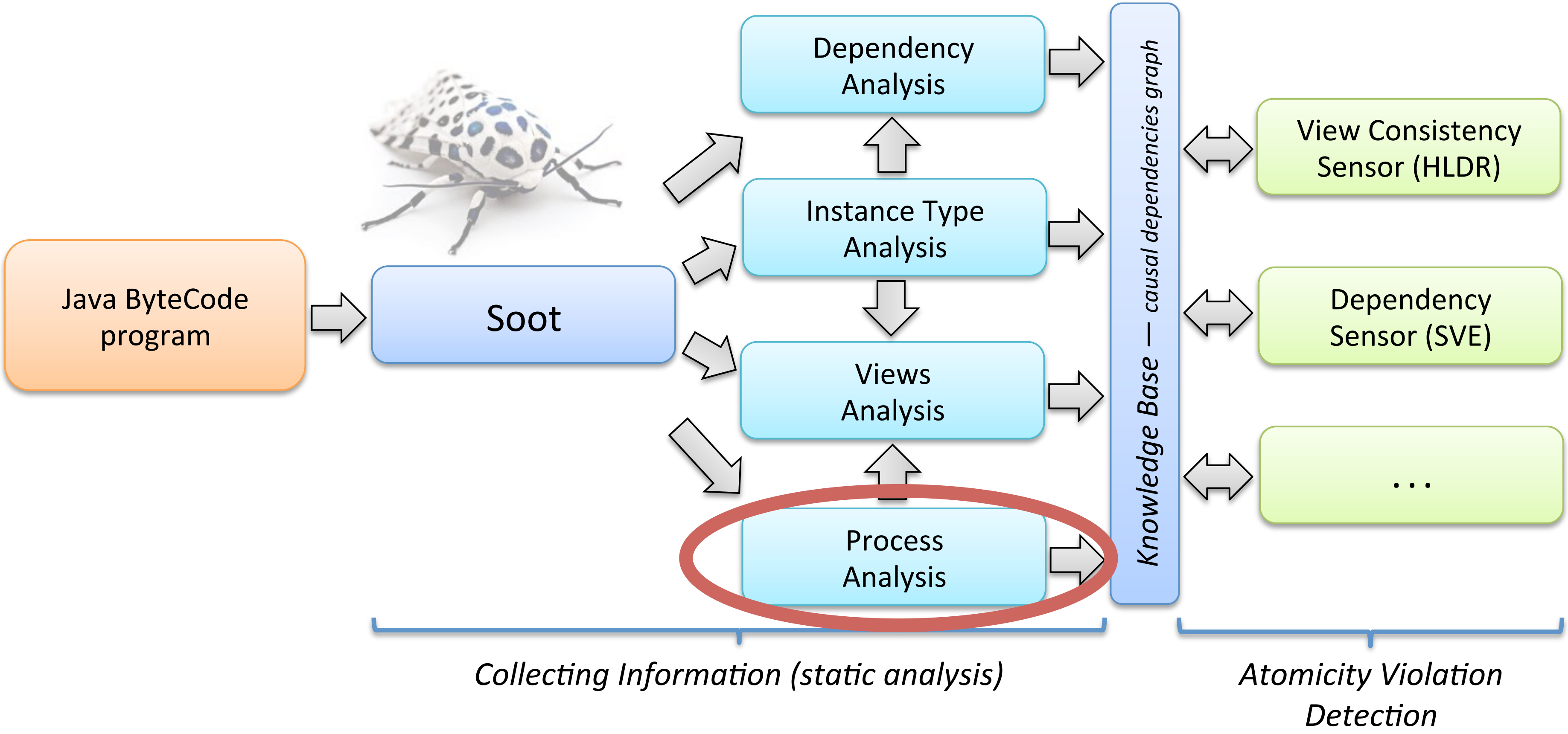
# Approach



Java ByteCode program
Soot
Dependency Analysis
Instance Type Analysis
Views Analysis
Process Analysis
Knowledge Base — causal dependencies graph
View Consistency Sensor (HLDR)
Dependency Sensor (SVE)
. . .
Collecting Information (static analysis)
Atomicity Violation Detection

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Process Analysis



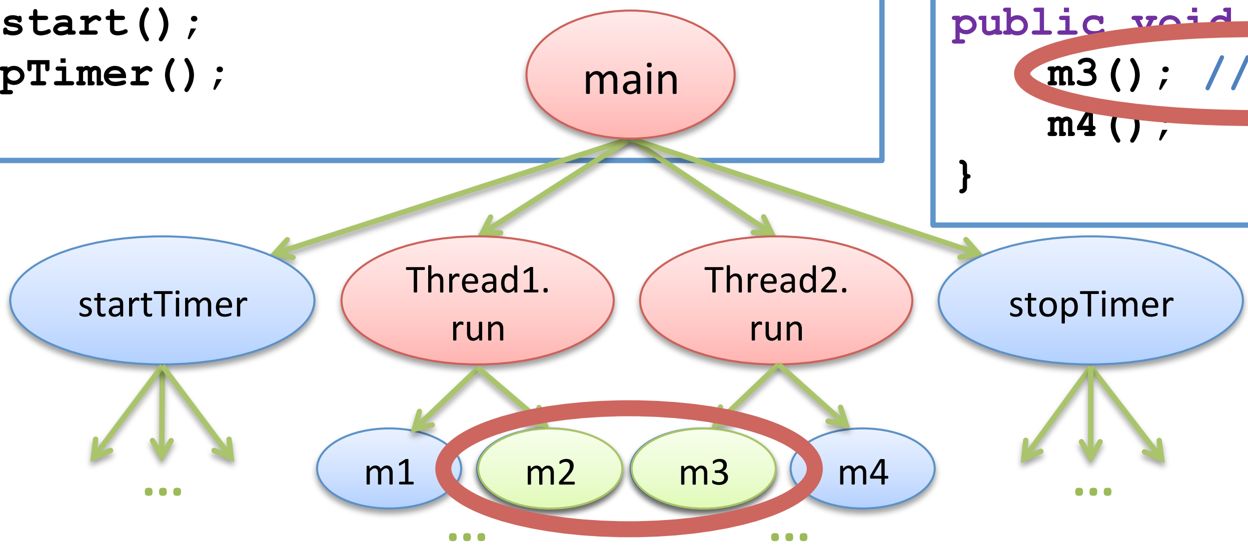
departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Process Analysis

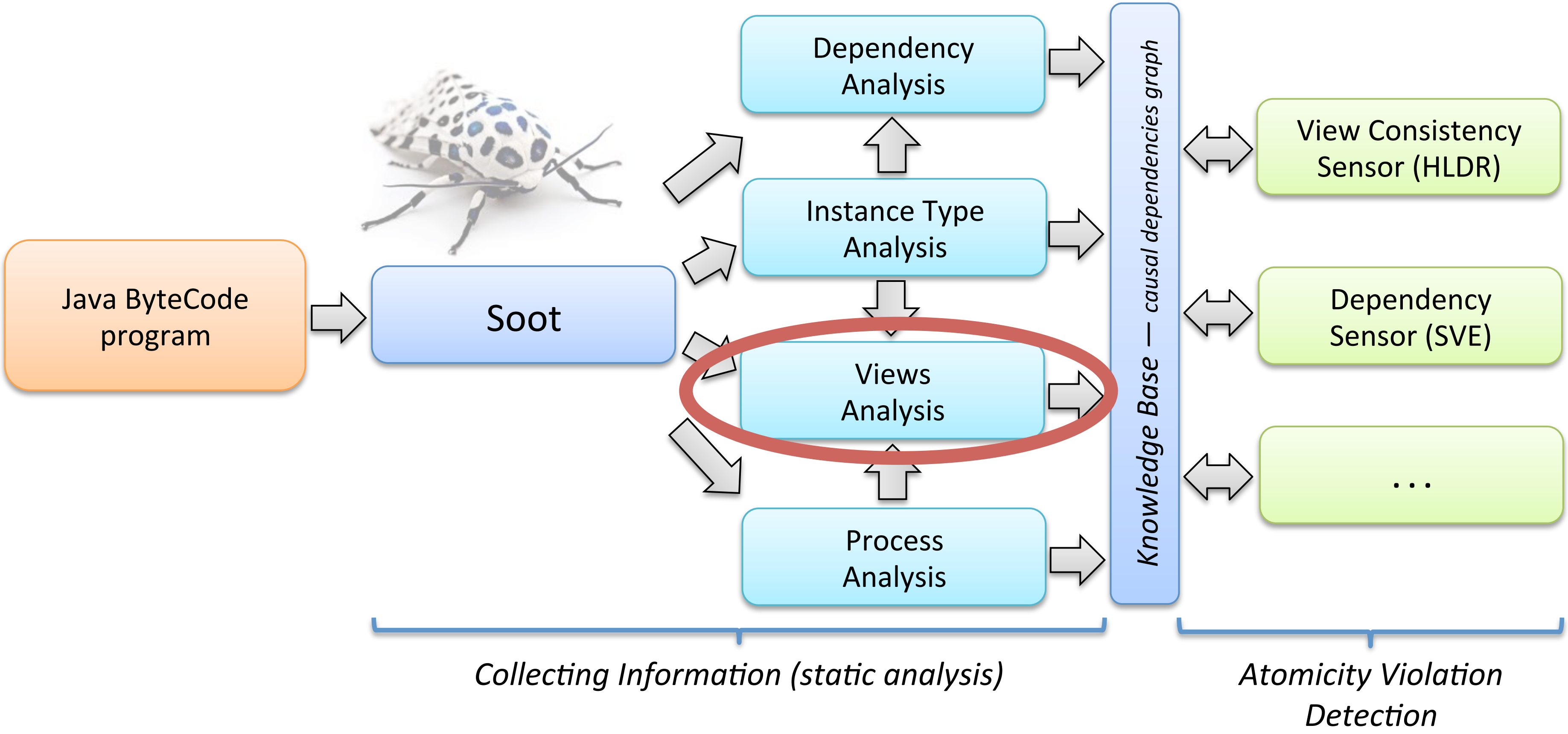
- Identify threads and atomic regions

```java
// Main Class
public static void main(String[] args){
    Thread1 t1 = new Thread1();
    Thread2 t2 = new Thread2();
    startTimer();
    t1.start();
    t2.start();
    stopTimer();
}
```

```java
// Thread1 Class
public void run(){
    m1();
    m2(); // Atomic
}
// Thread2 Class
public void run(){
    m3(); // Atomic
    m4();
}
```



# Views Analysis



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Views Analysis [Artho et al.]

- Views: sets of shared variables accessed atomically

```
@Atomic
public void incX(){
    int local = x;
    setX(local + 1)
}

public void setX(int aux){
    x = aux;
}
```

$$V(incX) = \{read(x)\} \cup V(setX) = \{read(x), write(x)\}$$

# Views Analysis (Vr / Vw)

- Views: sets of shared variables accessed atomically

```
@Atomic
public void incX(){
    int local = x;
    setX(local + 1)
}

public void setX(int aux){
    x = aux;
}
```

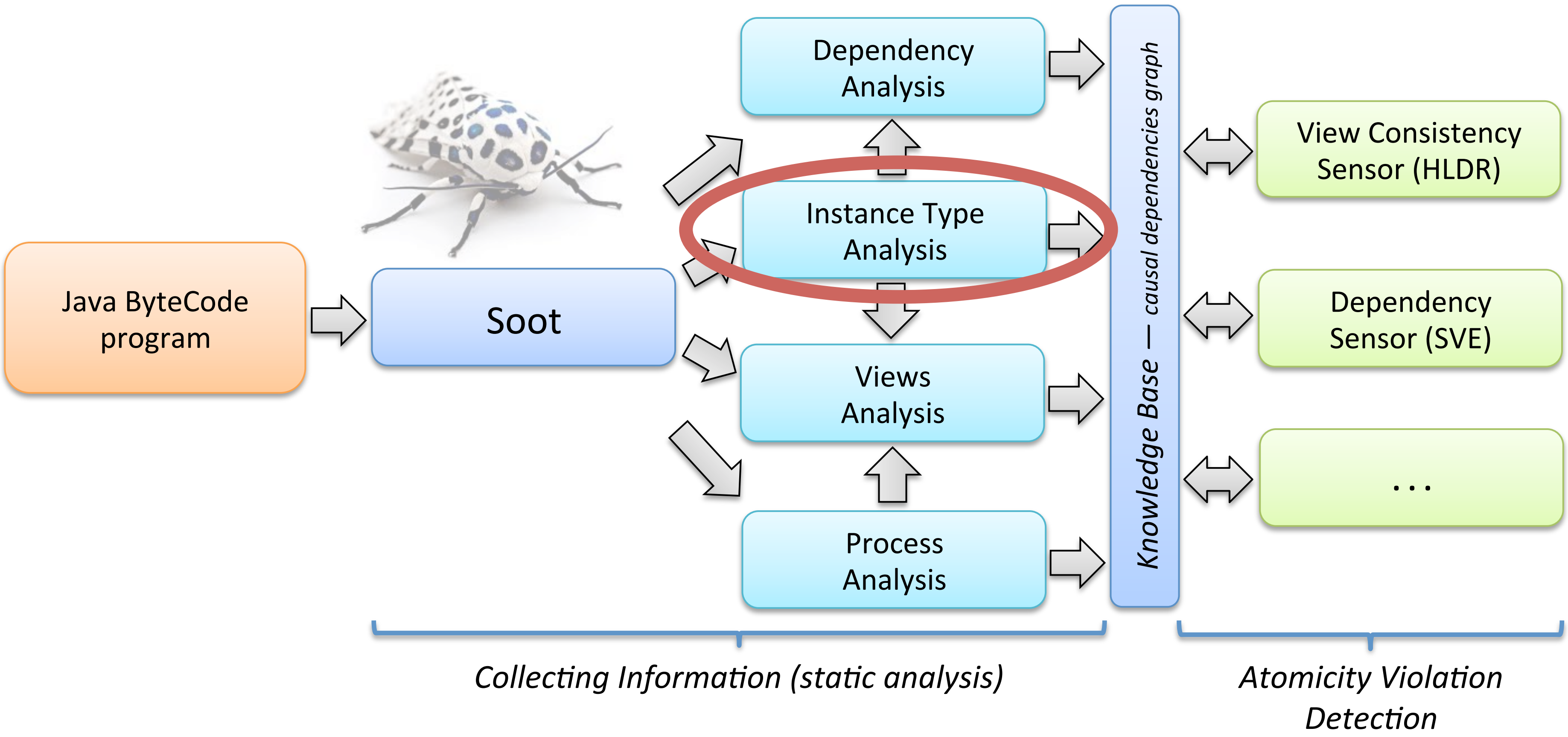
$$V(incX) = \{read(x)\} \cup \{write(x)\} = \{read(x), write(x)\}$$

$$\rightarrow V_r(incX) = \{read(x)\}$$

$$\rightarrow V_w(incX) = \{write(x)\}$$

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Instance Type Analysis



# Instance Type Analysis

- How to deal with interfaces with multiple implementations?
  - E.g. In Java:

```java
protected static List <Integer> list;

public void func(boolean cond){
    if(cond)
        list = new ArrayList <Integer>();
    else
        list = new LinkedList <Integer>();

    list.add(1);
}
```

- How to deal with native methods?

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Instance Type Analysis

- How to deal with dynamic dispatching?

```
protected static List <Integer> list;

public void func(boolean cond){
    if(cond)
        list = new ArrayList <Integer>();
    else
        list = new LinkedList <Integer>();

    list.add(1);  ---> ???
}
```

(list, ArrayList)

(list, LinkedList)

// ArrayList, LinkedList

V(list.add) = V(list.add)@ArrayList U V(list.add)@LinkedList

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

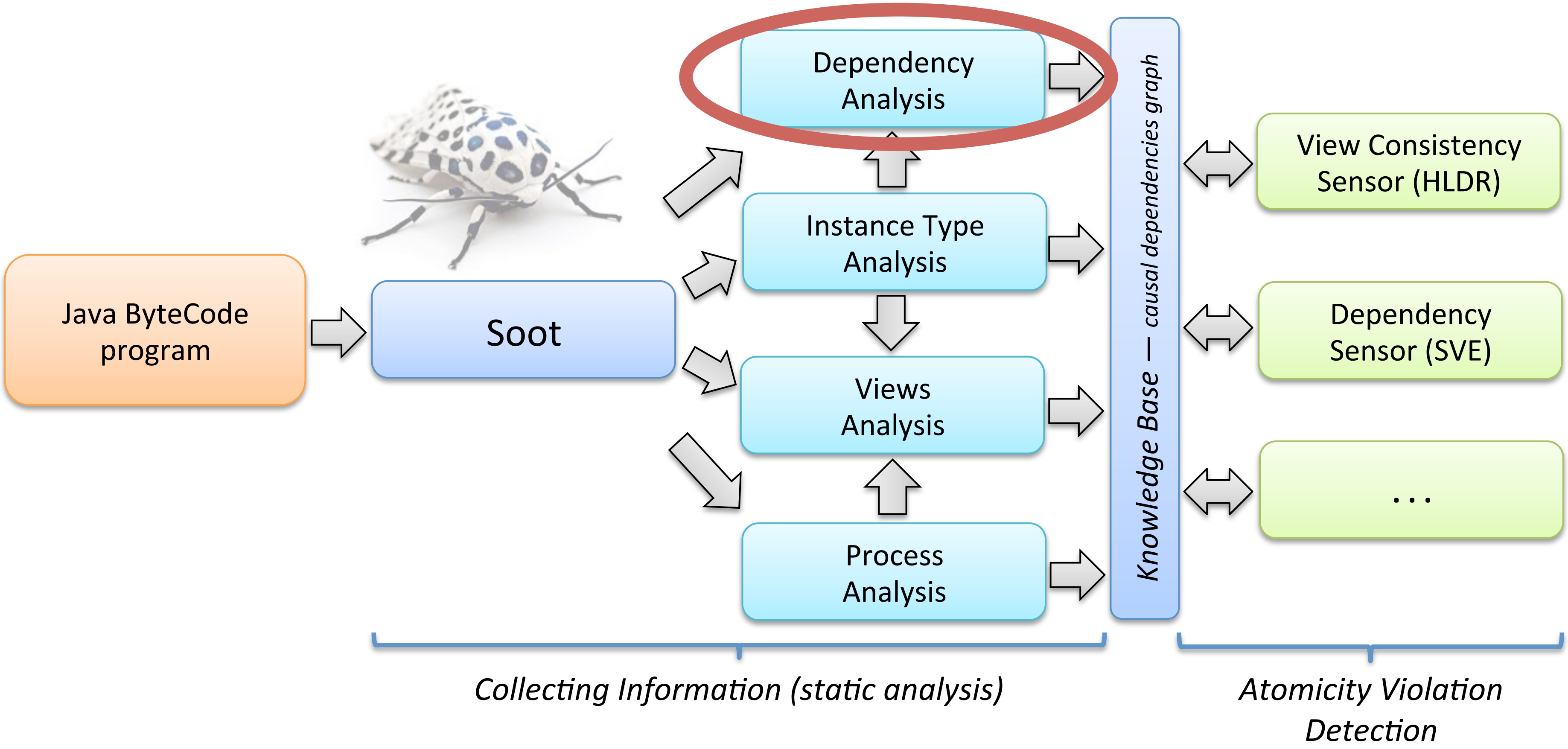
# Annotation Mechanism

```
class OurList{
    /**
     * This method adds a new element to the list
     * @param i, new element to add
     */
    private native void put(Integer i);
}
```

XML Annotations

```
<class id="OurList">
    <method id="put(Integer i)"/>
        <reads>this</reads>
        <writes>this</writes>
        <reads>p0</reads>
        <writes>p0</writes>
    </method>
</class>
```

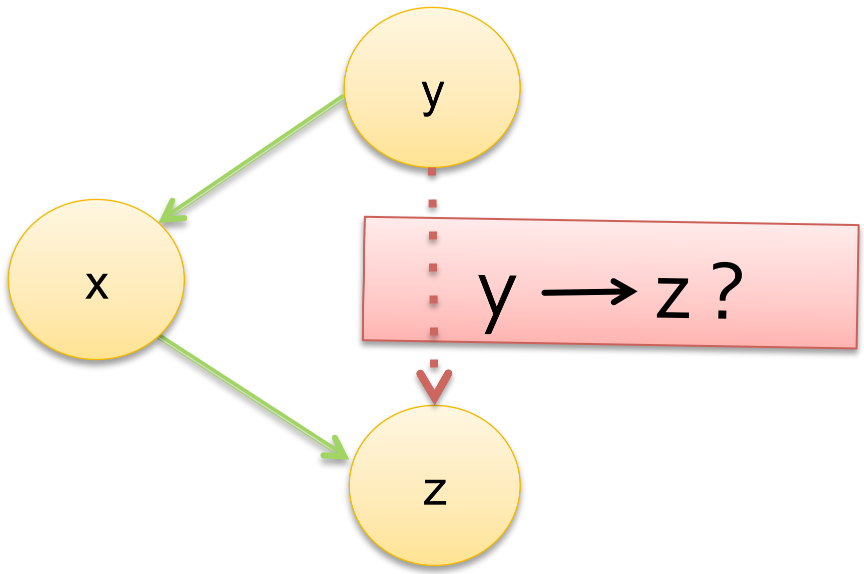
# Dependency Analysis



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Data Dependency Analysis

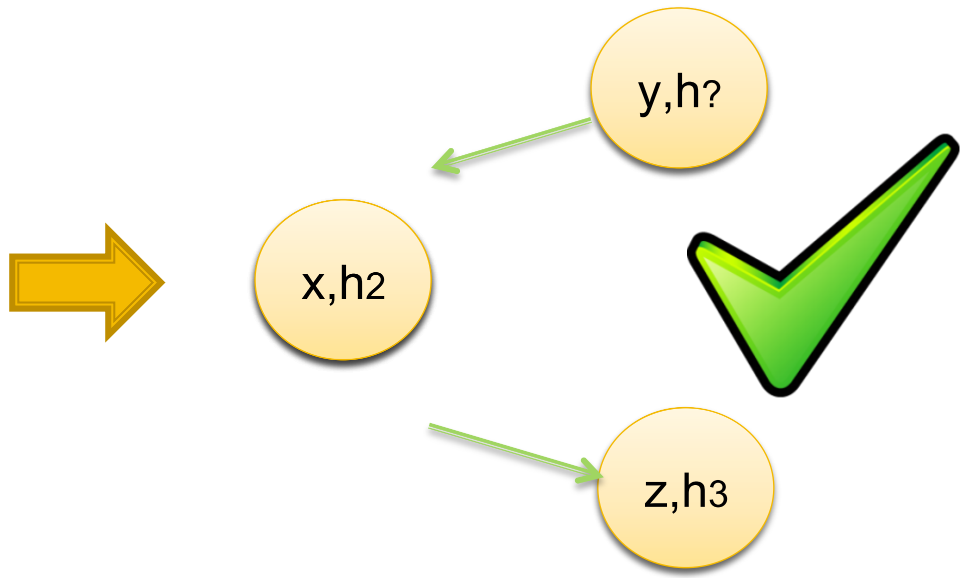
```
h1: x = y;
h2: x = 2;
h3: z = x;
```



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Data Dependency Analysis

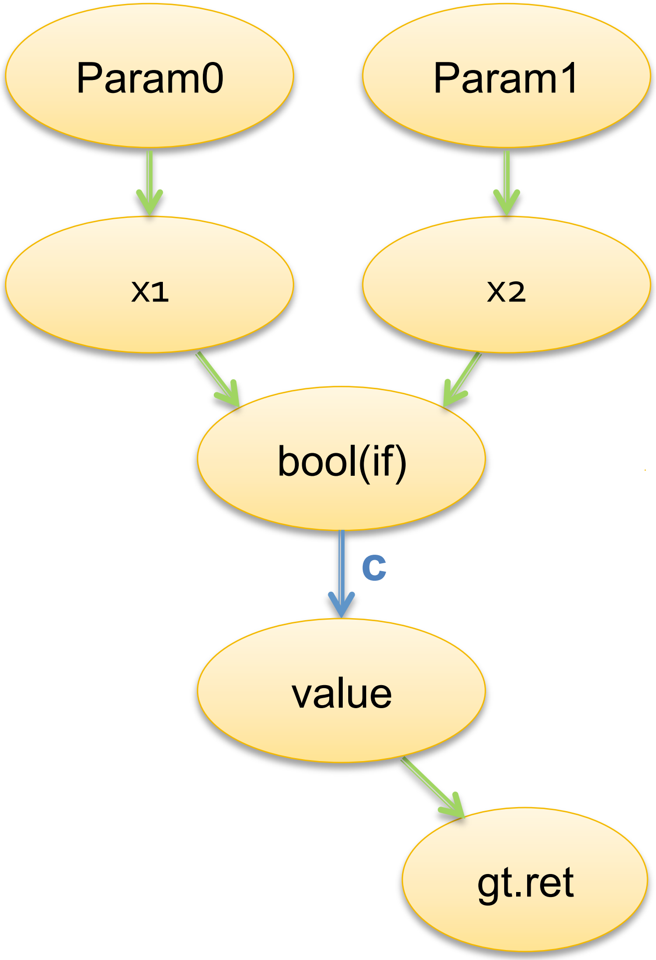
```
h1:  x = y;
h2:  x = 2;
h3:  z = x;
```



# Control Dependency Analysis

- Data Dependencies are not enough!

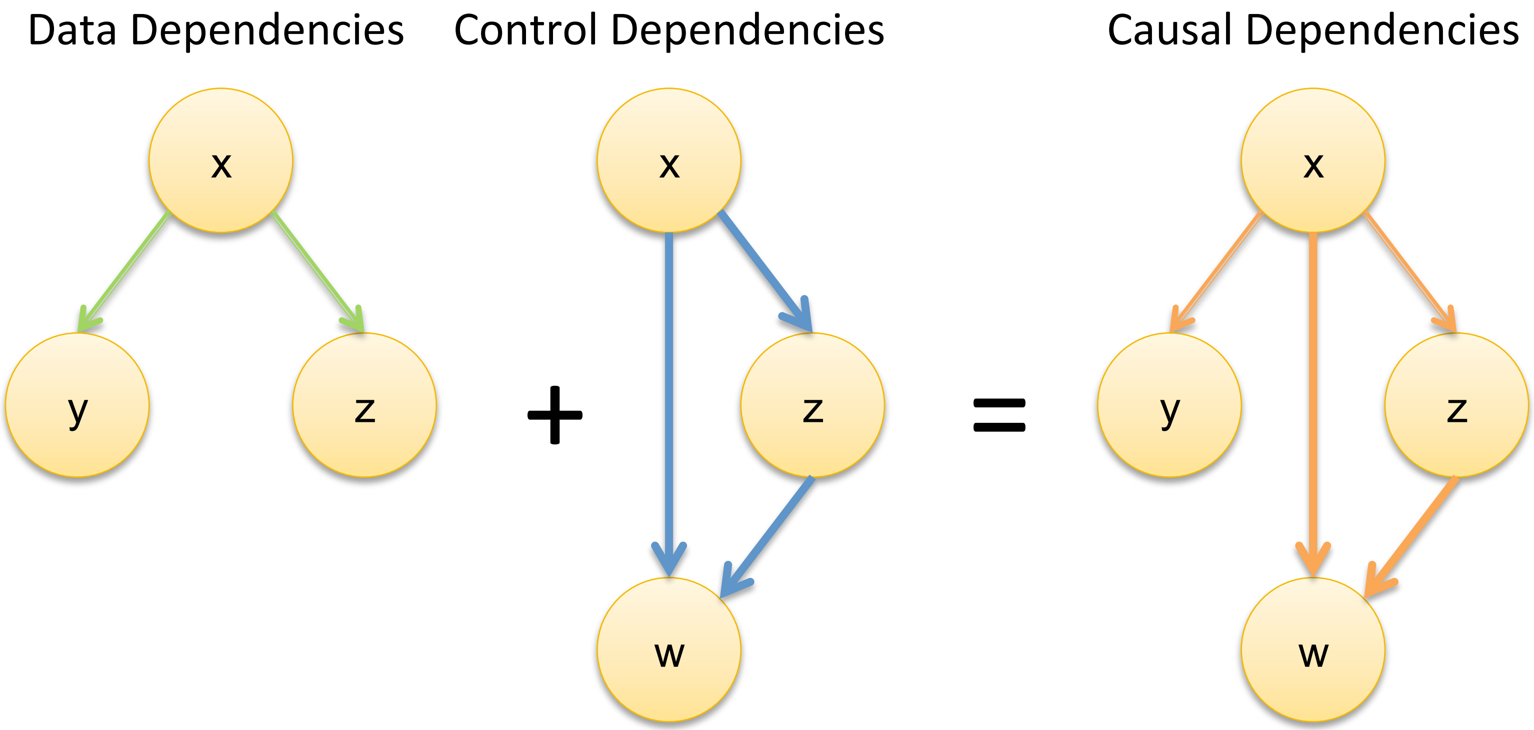
```
     boolean gt(int x1, int x2) {
         boolean value;
h1:      if(x1>x2){
h2:              value = true;
         }else{
h3:              value = false;
         }
h4:      return value;
     }
```



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Causal Dependencies Graph

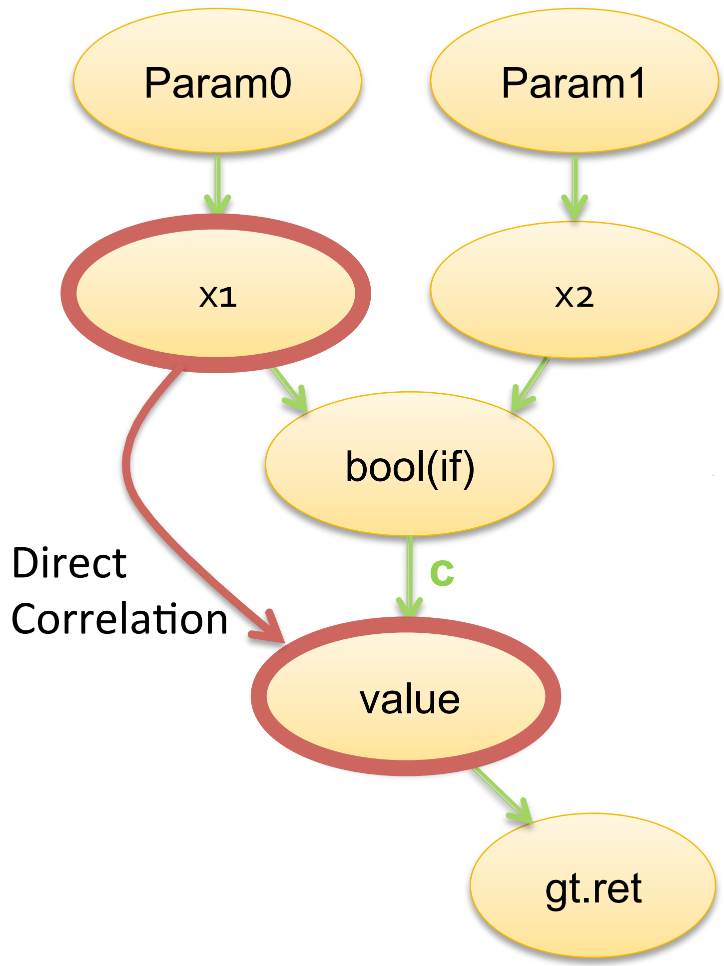
- Merge Data and Control Flow Dependencies in the Causal Dependencies Graph



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Direct Correlation

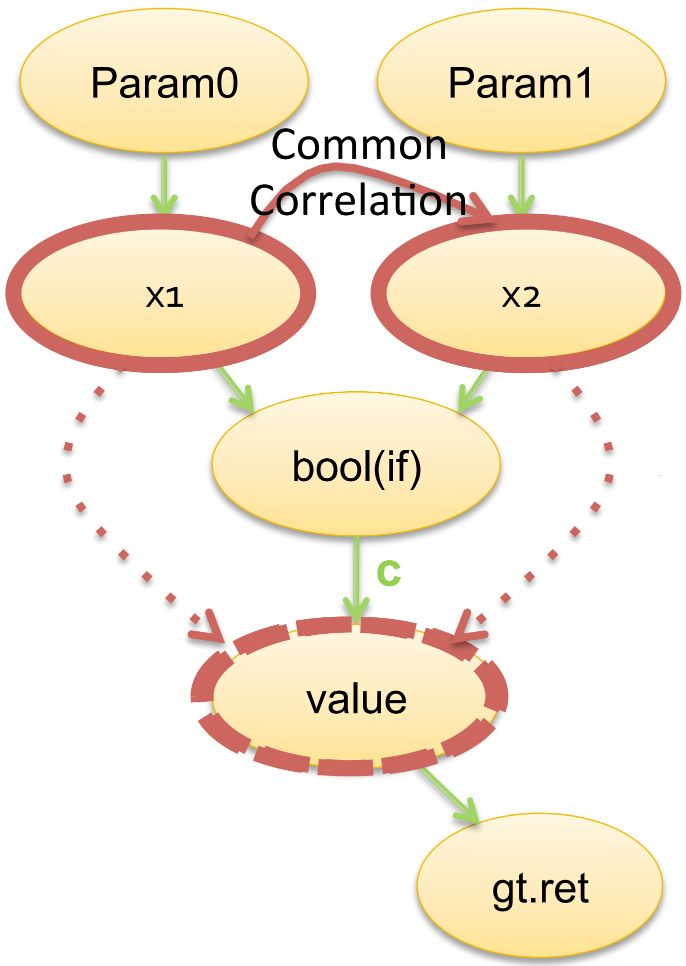
```
    boolean gt(int x1, int x2) {
        boolean value;
h1:     if(x1>x2){
h2:             value = true;
        }else{
h3:             value = false;
        }
h4:     return value;
    }
```



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Common Correlation

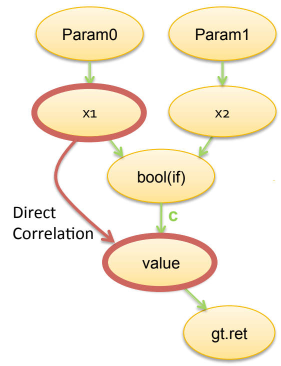
```
    boolean gt(int x1, int x2) {
        boolean value;
h1:     if(x1>x2){
h2:             value = true;
        }else{
h3:             value = false;
        }
h4:     return value;
    }
```



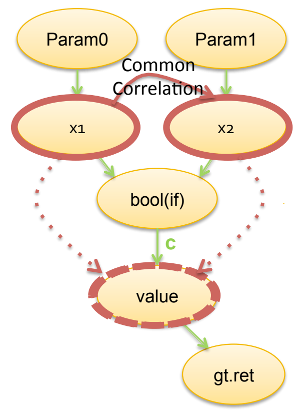
departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# Variable's Correlation

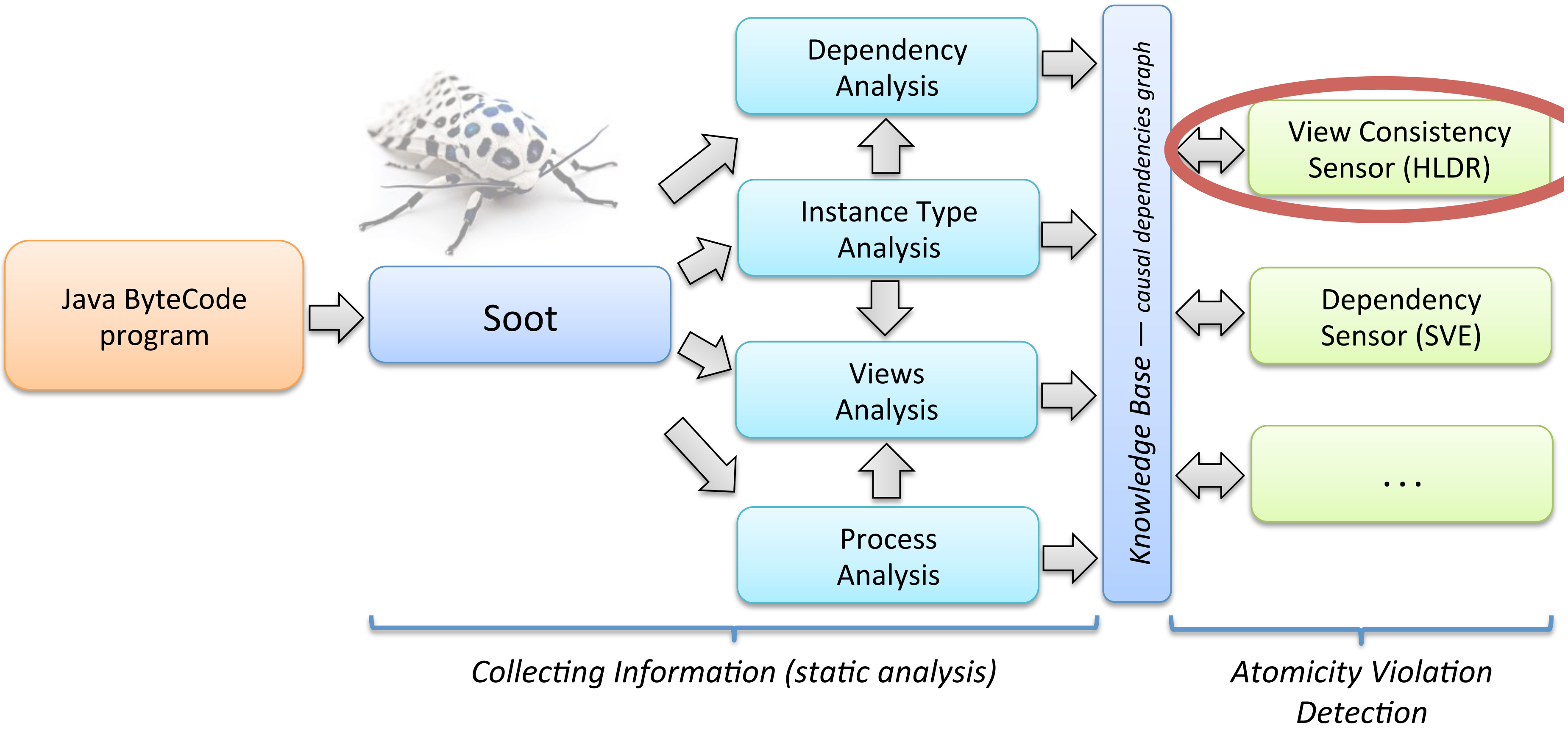
- Direct Correlation (x, y):
  There is a direct correlation between a read variable x and a written variable y if there is a path from x to y, in a dependency graph D.




- Common Correlation (x, y):
  There is a common correlation between a read variable x and a read variable y if there is a written variable z, where z ≠ x and z ≠ y, for which there is a path from x to z and another path from y to z, in a dependency graph D.



# View Consistency Sensor



departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# View Consistency Sensor

Thread 1

```
@Atomic
public int setPair(int v1, int v2){
    x = v1;
    y = v2;
}
```

Thread 2

```
public boolean equals(){
  int loc_x = getX(); // Atomic
  int loc_y = getY(); // Atomic
  return loc_x == loc_y;
}
```

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# View Consistency Sensor

Thread 1

```
@Atomic
public int setPair(int v1, int v2){
    x = v1;
    y = v2;
}
```

Thread 2

```
public boolean equals(){
  int loc_x = getX()   // Atomic
  int loc_y = getY()   // Atomic
  return loc_x == loc_y;
}
```

| Thread 1 | Thread 2 |
|---|---|
| $V_w$(setPair) = {x, y} | $V_w$(getX) = { } |
| $V_r$(setPair) = { } | $V_r$(getX) = {x} |
| | $V_w$(getY) = { } |
| | $V_r$(getY) = {y} |

departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

# View Consistency Sensor

Thread 1

```
@Atomic
public int setPair(int v1, int v2){
    x = v1;
    y = v2;
}
```

Thread 2

```
public boolean equals(){
  int loc_x = getX(); // Atomic
  int loc_y = getY(); // Atomic
  return loc_x == loc_y;
}
```

Maximal View$_W$ of T1

| Thread 1 | Thread 2 |
|---|---|
| $V_w$(setPair) = {x, y} | { } |
| $V_r$(setPair) = { | (getX) = {x} |
| | $V_w$(getY) = { } |
| | $V_r$(getY) = {y} |

View$_R$ of T2

View$_R$ of T2

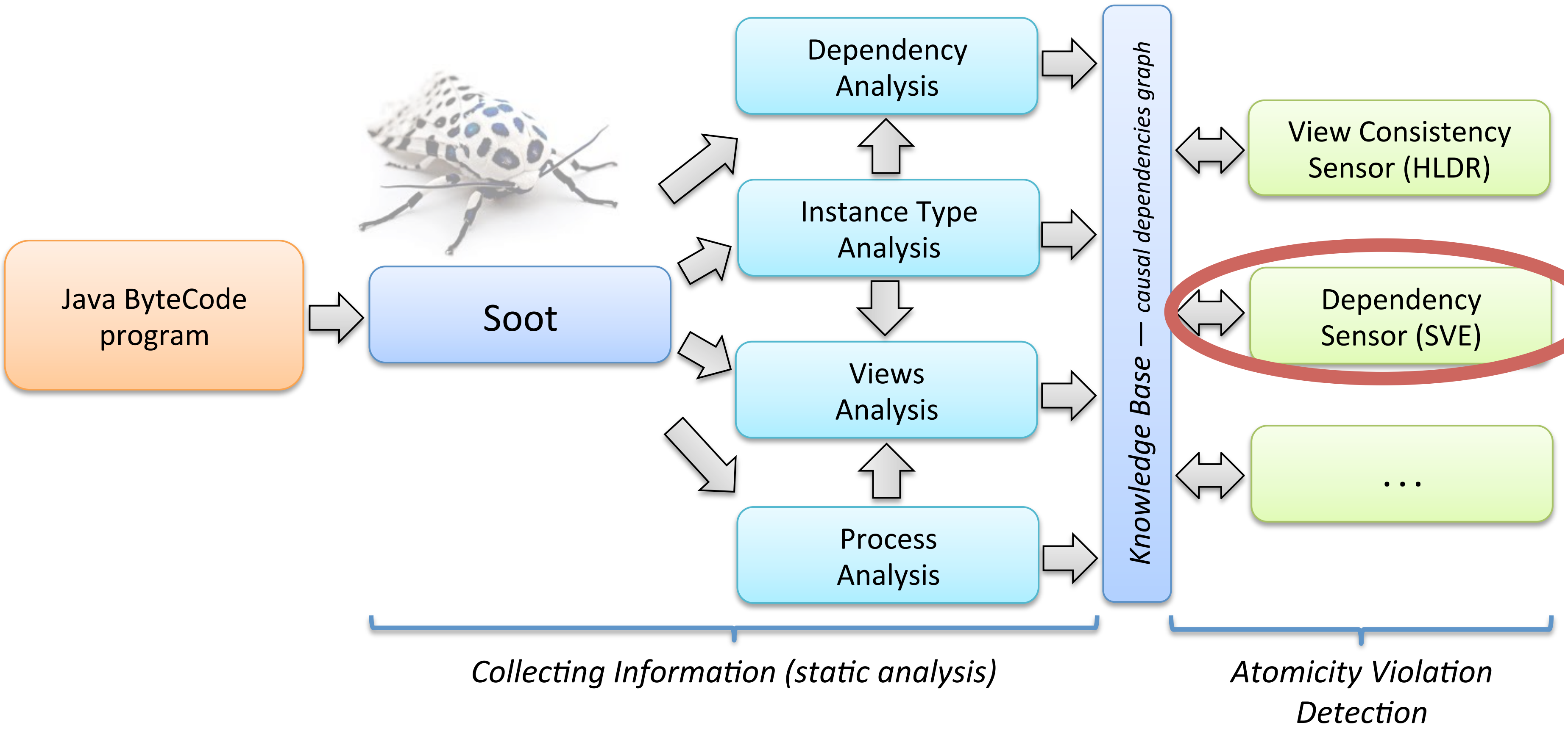
Data Race!

$\{x, y\} \cap \{x\} = \{x\}$

$\{x, y\} \cap \{y\} = \{y\}$

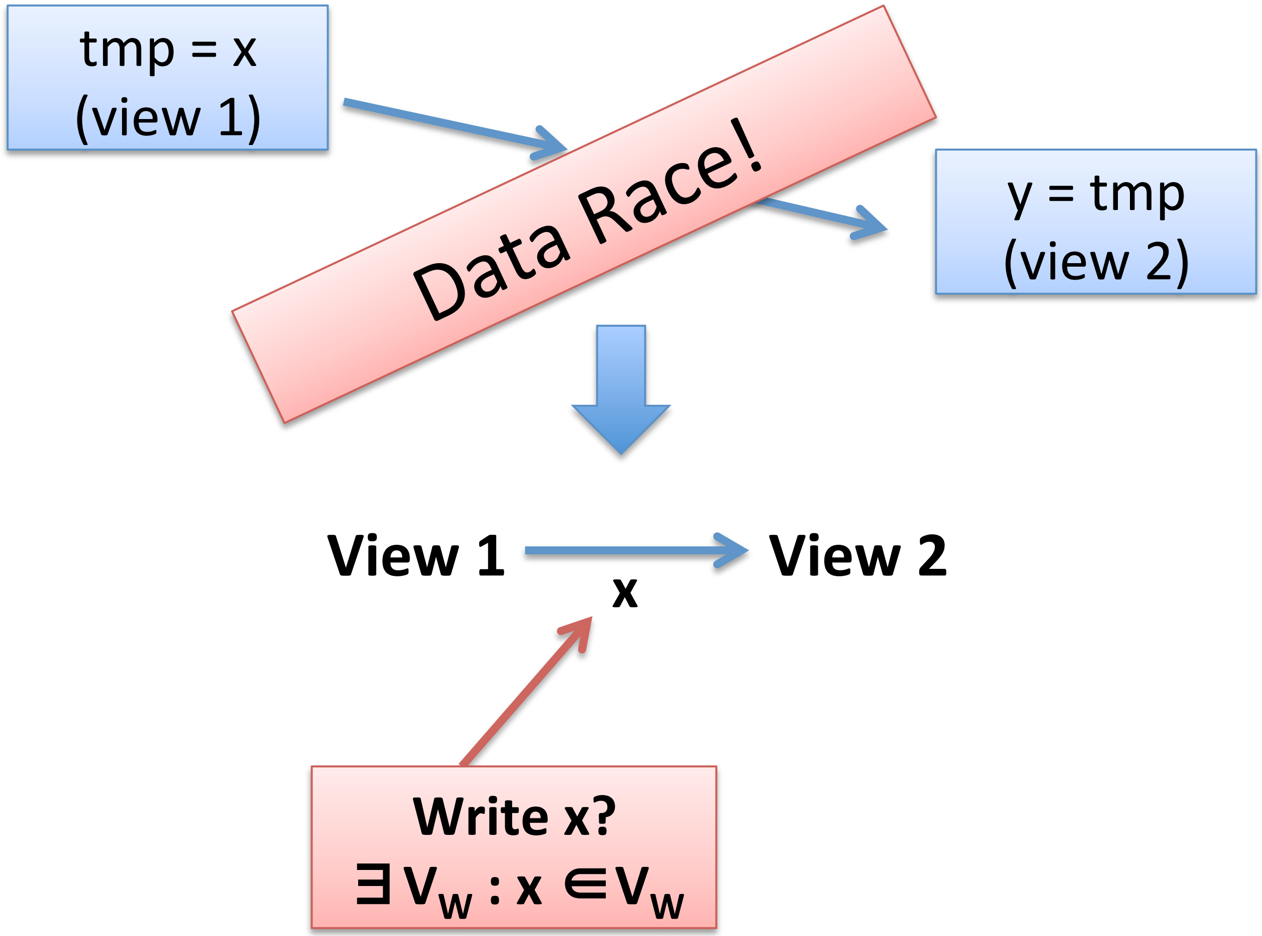
$(\{x\} \nsubseteq \{y\} \wedge \{y\} \nsubseteq \{x\}) \wedge$ Common Correlation ({x}, {y})

# Dependency Sensor



# Dependency Sensor

- Detecting Stale-Value Errors...



# The End…

- Thank you for listening! ☺