Concurrency

EECS 4315

www.cse.yorku.ca/course/4315/

Counter Class

Problem

Implement the class `Counter` with

- attribute `value`,
- initialized to zero, and
- the methods `increment` and `decrement`.

Counter Class

Problem

Implement the class `Counter` with

- attribute `value`,
- initialized to zero, and
- the methods `increment` and `decrement`.

Question

Can multiple threads share a `Counter` object and use methods such as `increment` and `decrement` concurrently?

Counter Class

Problem

Implement the class `Counter` with

- attribute `value`,
- initialized to zero, and
- the methods `increment` and `decrement`.

Question

Can multiple threads share a `Counter` object and use methods such as `increment` and `decrement` concurrently?

Answer

No, as before, if two threads invoke `increment` concurrently, the counter may only be incremented by one (rather than two).

Synchronized Methods

Methods such as `increment` should be executed atomically. This can be accomplished by declaring the method to be `synchronized`.

A lock is associated with every object. For threads to execute a synchronized method on such the object, first its lock needs to be acquired.

Synchronized Methods

Methods such as `increment` should be executed atomically. This can be accomplished by declaring the method to be `synchronized`.

A lock is associated with every object. For threads to execute a synchronized method on such the object, first its lock needs to be acquired.

```
public synchronized void increment()
{
  this.value++;
}
```

Resource Class

Problem

Implement the class **`Resource`** with

- attribute **`available`**,
- initialized to true, and
- the methods **`acquire`** and **`release`**.

Wait and Notify

The Object class contains the following three methods:

- `wait`: causes the current thread to wait until another thread wakes it up.
- `notify`: wakes up a single thread waiting on this object's lock; if there is more than one waiting, an arbitrary one is chosen; if there are none, nothing is done.
- `notifyAll`: wakes up all threads waiting on this objects lock.

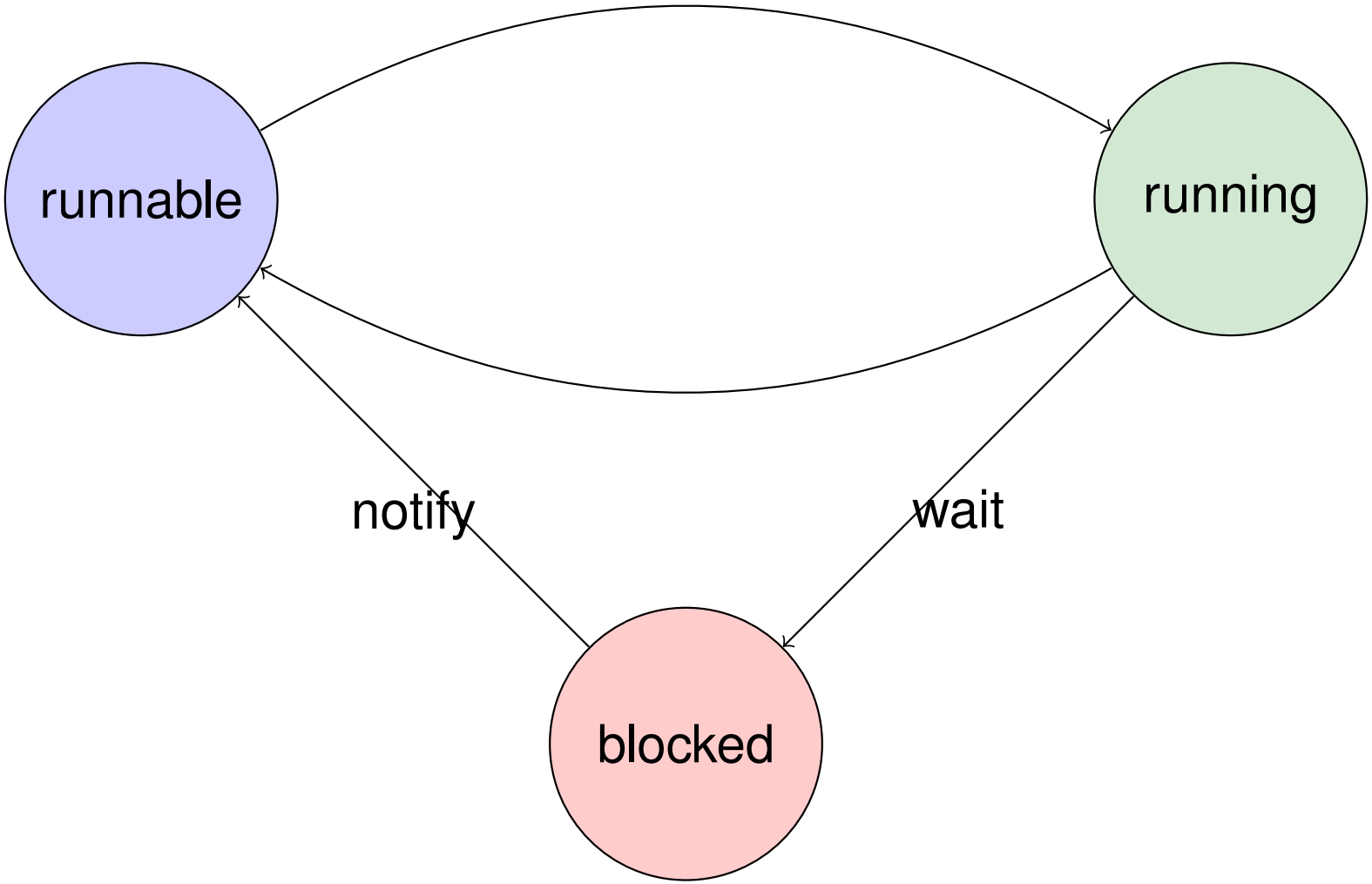
Wait and Notify

The Object class contains the following three methods:

- `wait`: causes the current thread to wait until another thread wakes it up.
- `notify`: wakes up a single thread waiting on this object's lock; if there is more than one waiting, an arbitrary one is chosen; if there are none, nothing is done.
- `notifyAll`: wakes up all threads waiting on this objects lock.

Since every class extends the class Object, these methods are available to every object.

States of a Thread



Counter Class

```
public class Counter extends Thread
{
  private int value;

  public Counter()
  {
    this.value = 0;
  }

  ...

}
```

Counter Class

```
public void run()
{
  this.value++;
}
```

```
0: aload_0
1: dup
2: getfield
5: iconst_1
6: iadd
7: putfield
10: return
```

Main Class

```
public class Main
{
  public static void main(String[] args)
  {
    Counter one = new Counter();
        Counter two = new Counter();
        one.start();
        two.start();
  }
}
```

```
0: new
3: dup
4: invokespecial
7: astore_1
8: new
```

```
11: dup
12: invokespecial
15: astore_2
16: aload_1
17: invokevirtual
```

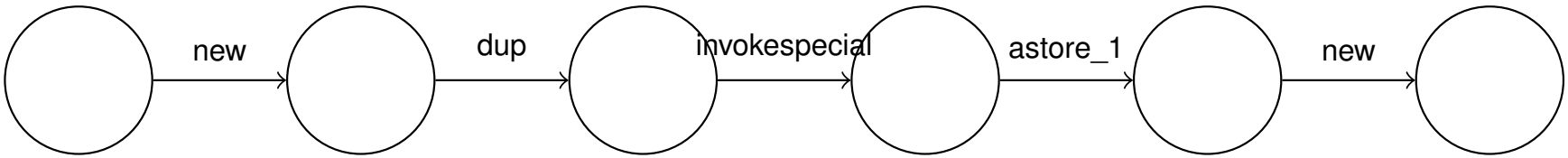
```
20: aload_2
21: invokevirt
24: return
```

State-Transition Diagram

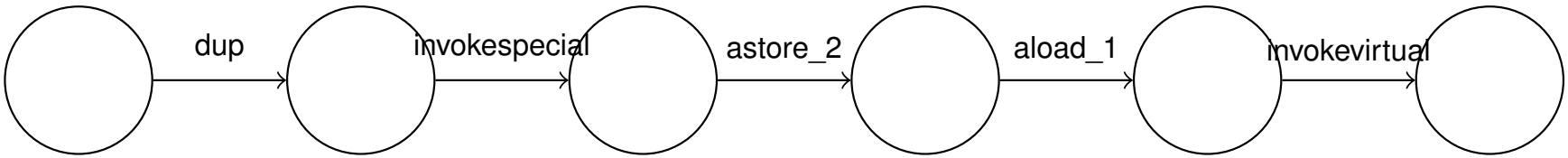
Question

Draw the corresponding state-transition diagram.

State-Transition Diagram



State-Transition Diagram

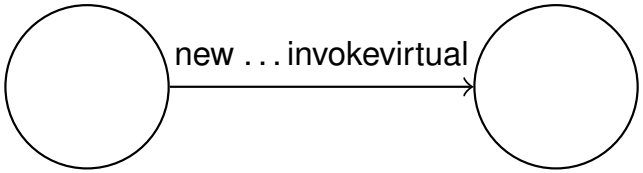


A Smaller Model

Combine the first ten transitions into one.

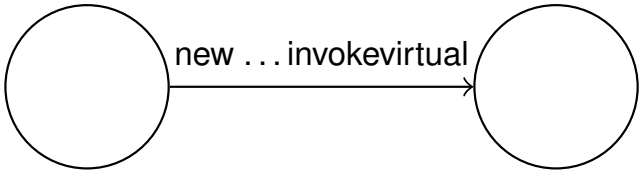
A Smaller Model

Combine the first ten transitions into one.



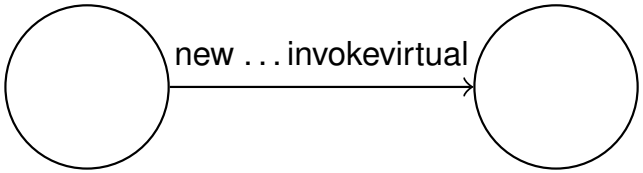
A Smaller Model

Combine the first ten transitions into one.



The actions of the labelled transition system are sequences of bytecode instructions.

State-Transition Diagram



Next instructions for the main thread:

```
20: aload_2
21: invokevirtual
24: return
```

Next instructions for the thread `one`:

```
0: aload_0
1: dup
2: getfield
5: iconst_1
6: iadd
7: putfield
```

State-Transition Diagram

Question

Can the bytecode instructions corresponding to the `run` invocation be modelled as a single transition?

State-Transition Diagram

Question

Can the bytecode instructions corresponding to the `run` invocation be modelled as a single transition?

Answer

Yes.

State-Transition Diagram

Question

Can the bytecode instructions corresponding to the `run` invocation be modelled as a single transition?

Answer

Yes.

Question

Why?

State-Transition Diagram

Question

Can the bytecode instructions corresponding to the `run` invocation be modelled as a single transition?

Answer

Yes.

Question

Why?

Answer

Because the execution of this method does not impact the other threads.

Combining Bytecode Instructions

- We combine the first ten bytecode instructions since there is only one thread.
- We combine the bytecode instructions corresponding to the `run` invocation because those do not impact the other threads.

Combining Bytecode Instructions

- We combine the first ten bytecode instructions since there is only one thread.
- We combine the bytecode instructions corresponding to the `run` invocation because those do not impact the other threads.

General idea

Combine those bytecode instructions that do not impact other threads.

Combining Bytecode Instructions

Problem

Given all the bytecode instructions, determine for a specific instruction whether it impacts other threads.

Combining Bytecode Instructions

Problem

Given all the bytecode instructions, determine for a specific instruction whether it impacts other threads.

Question

Give an algorithm that solves the problem.

Combining Bytecode Instructions

Problem

Given all the bytecode instructions, determine for a specific instruction whether it impacts other threads.

Question

Give an algorithm that solves the problem.

Question

Impossible!

Proving Impossibility

Question

Which other problems cannot be solved?

Proving Impossibility

Question

Which other problems cannot be solved?

Answer

The halting problem: given code and input for that code, determine whether the code terminates.

Proving Impossibility

Problem

Given all the bytecode instructions, determine for a specific instruction whether it impacts other threads.

Question

Prove that the problem cannot be solved.

Combining Bytecode Instructions

General idea

Combine those bytecode instructions for which we can prove that they do not impact other threads.

Combining Bytecode Instructions

General idea

Combine those bytecode instructions for which we can prove that they do not impact other threads.

The idea of combining consecutive transitions labelled with invisible (outside the current thread) actions into a single transition is due to Patrice Godefroid.

Combining Bytecode Instructions

General idea

Combine those bytecode instructions for which we can prove that they do not impact other threads.

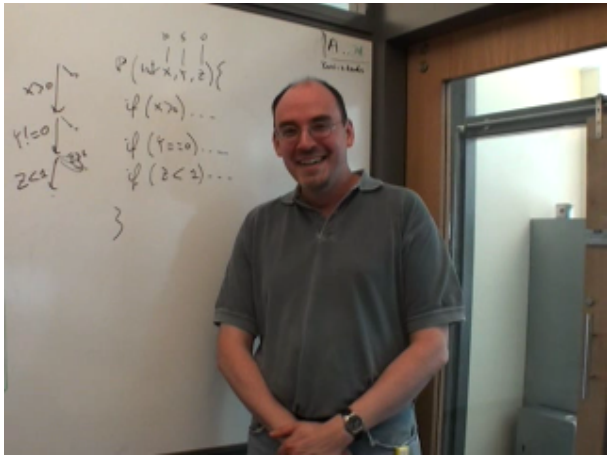
The idea of combining consecutive transitions labelled with invisible (outside the current thread) actions into a single transition is due to Patrice Godefroid.

Examples of invisible actions

- Reading or writing an attribute that can be proved to be not shared.
- Reading or writing a local variable.
- ...

Patrice Godefroid

- Ph.D. degree in Computer Science from the University of Liege, Belgium
- Worked at Bell Laboratories.
- Currently at Microsoft Research.



Source: Patrice Godefroid