Model Checking

EECS 4315

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

Model Checking

Develop a model (states connected by transitions) of the code and check properties of the model.

Model Checking

Model checking was developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980s.

Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In, Dexter Kozen, editor, *Proceedings of Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Yorktown Heights, NY, USA, May 1981. Springer-Verlag.

Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In, Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings of the 5th International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Torino, Italy, April 1982. Springer-Verlag.

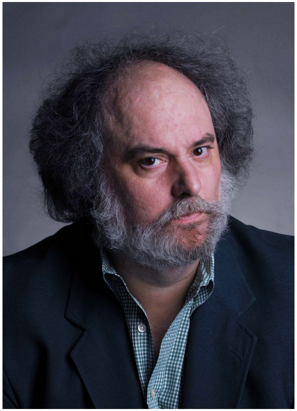
Edmund Clarke

- Recipient of the Turing Award (2007)
- Recipient of the ACM Paris Kanellakis Award (1999)
- Member of the National Academy of Engineering (2005)
- Member of the American Academy of Arts and Sciences (2011)

Source: Dennis Hamilton

Allen Emerson

- Recipient of the Turing Award (2007)
- Recipient of the ACM Paris Kanellakis Award (1999)
- Recipient of the CMU Newell Medal (1999)


Source: Marsha Miller

Joseph Sifakis

- Recipient of the Turing Award (2007)
- Grand officer of France's national order of merit (2008)
- Commander in France's legion of honour (2011)

Source: David Monniaux

Jean-Pierre Queille

Source: unknown

Model of a System

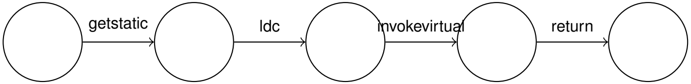
A model of a system is an abstraction of the system.

THE PROJECT MANAGEMENT FRAMEWORK EMBODIES A PROJECT LIFE CYCLE AND FIVE MAJOR PROJECT MANAGEMENT PROCESS GROUPS.
OH NO! THE EXTREME LEVEL OF ABSTRACTION HAS MADE US WEIGHTLESS!
THAT DOESN'T EVEN MAKE SENSE.
Dilbert.com DilbertCartoonist@gmail.com
1-7-12 ©2012 Scott Adams, Inc./Dist. by Universal Uclick

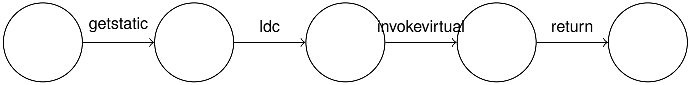
There are many levels of abstraction and, hence, a system can be modelled in many different ways.

A Model of a System

```
public class HelloWorld
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
  }
}
```



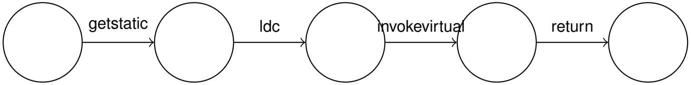
A Model of a System



Question

What are the three entities that make up the above model?

A Model of a System



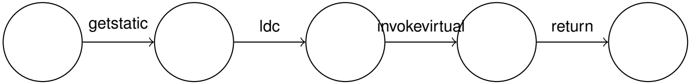
Question

What are the three entities that make up the above model?

Answer

States, transitions and actions (such as getstatic, ldc, . . .).

A Model of a System

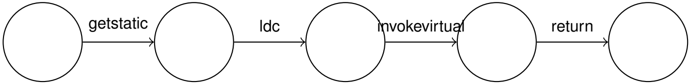


Question

Given a set of states S and a set of actions A, how can we mathematically model a transition from state s_1 to state s_2 labelled with action a?

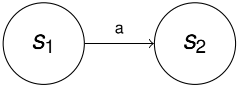
$$\begin{tikzcd} s_1 \arrow[r, "a"] & s_2 \end{tikzcd}$$

A Model of a System



Question

Given a set of states S and a set of actions A, how can we mathematically model a transition from state s_1 to state s_2 labelled with action a?



Answer

(s_1, a, s_2)

A Model of a System

$$\begin{tikzcd}
s_1 \arrow[r, "\text{getstatic}"] & s_2 \arrow[r, "\text{ldc}"] & s_3 \arrow[r, "\text{invokevirtual}"] & s_4 \arrow[r, "\text{return}"] & s_5
\end{tikzcd}$$

Question

How can we model all the labelled transitions?

A Model of a System

$$\begin{tikzcd} s_1 \arrow[r, "\text{getstatic}"] & s_2 \arrow[r, "\text{ldc}"] & s_3 \arrow[r, "\text{invokevirtual}"] & s_4 \arrow[r, "\text{return}"] & s_5 \end{tikzcd}$$

Question

How can we model all the labelled transitions?

Answer

$$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$$

A Model of a System

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$ is a subset of $S \times A \times S$

Question

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$ is a . . . over the sets S, A and S.

A Model of a System

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$
is a subset of $S \times A \times S$

Question

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$
is a . . . over the sets S, A and S.

Answer

relation

A Model of a System

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$ is a subset of $S \times A \times S$

Question

$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}$ is a … over the sets S, A and S.

Answer

relation

The relation is usually denoted by $\rightarrow$ and called the transition relation.

Labelled Transition System

Systems can be modelled by means of labelled transition systems.

Definition

A labelled transition system is a tuple $\langle S, A, \rightarrow \rangle$ consisting of

- a set S of states,
- a set A of actions, and
- a transition relation $\rightarrow \subseteq S \times A \times S$.

Labelled Transition System

Systems can be modelled by means of labelled transition systems.

Definition

A labelled transition system is a tuple $\langle S, A, \rightarrow \rangle$ consisting of

- a set S of states,
- a set A of actions, and
- a transition relation $\rightarrow \subseteq S \times A \times S$.

Instead of $(s_1, a, s_2) \in \rightarrow$, we usually write $s_1 \xrightarrow{a} s_2$.

Labelled Transition System

$$\begin{tikzcd} s_1 \arrow[r, "\text{getstatic}"] & s_2 \arrow[r, "\text{ldc}"] & s_3 \arrow[r, "\text{invokevirtual}"] & s_4 \arrow[r, "\text{return}"] & s_5 \end{tikzcd}$$

Question

Give the corresponding labelled transition system.

Labelled Transition System

$$\begin{tikzcd} s_1 \arrow[r, "\text{getstatic}"] & s_2 \arrow[r, "\text{ldc}"] & s_3 \arrow[r, "\text{invokevirtual}"] & s_4 \arrow[r, "\text{return}"] & s_5 \end{tikzcd}$$

Question

Give the corresponding labelled transition system.

Answer

$\langle\{s_1, s_2, s_3, s_4, s_5\},$
$\{\text{getstatic}, \text{ldc}, \text{invokevirtual}, \text{return}\},$
$\{(s_1, \text{getstatic}, s_2), (s_2, \text{ldc}, s_3), (s_3, \text{invokevirtual}, s_4), (s_4, \text{return}, s_5)\}\rangle$

Intermezzo

Concurrency.

Concurrency

Threads can exchange information by accessing and updating shared attributes.

Question

One thread executes

```
v = 1;
v = v + 1;
```

and another thread executes

```
v = 0;
```

What is the final value of v?

Concurrency

Threads can exchange information by accessing and updating shared attributes.

Question

One thread executes

```
v = 1;
v = v + 1;
```

and another thread executes

```
v = 0;
```

What is the final value of v?

Answer

0, 1 or 2.

Concurrency

Question

One thread executes

```
v = v + 1;
```

and another thread executes

```
v = v + 1;
```

If the initial value of v is 0, then what is the final value of v?

Concurrency

Question

One thread executes

```
v = v + 1;
```

and another thread executes

```
v = v + 1;
```

If the initial value of v is 0, then what is the final value of v?

Answer

1 or 2.

Concurrency

Question

How can the final value of v be 1?

Concurrency

Question

How can the final value of v be 1?

Answer

The assignment `v = v + 1` is not atomic.

Concurrency

Question

How can the final value of v be 1?

Answer

The assignment `v = v + 1` is not atomic.

```
0: getstatic
3: iconst_1
4: iadd
5: putstatic
```

Concurrency

Question

One thread executes

```
v = 0;
```

and another thread executes

```
v = Long.MAX_VALUE;
```

How many different final values can v have?

Concurrency

Question

One thread executes

```
v = 0;
```

and another thread executes

```
v = Long.MAX_VALUE;
```

How many different final values can v have?

Answer

4.

Concurrency

Question

How can v have 4 different final values?

Concurrency

Question

How can v have 4 different final values?

Answer

The assignments `v = 0` and `v = Long.MAX_VALUE` may not be atomic.

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.

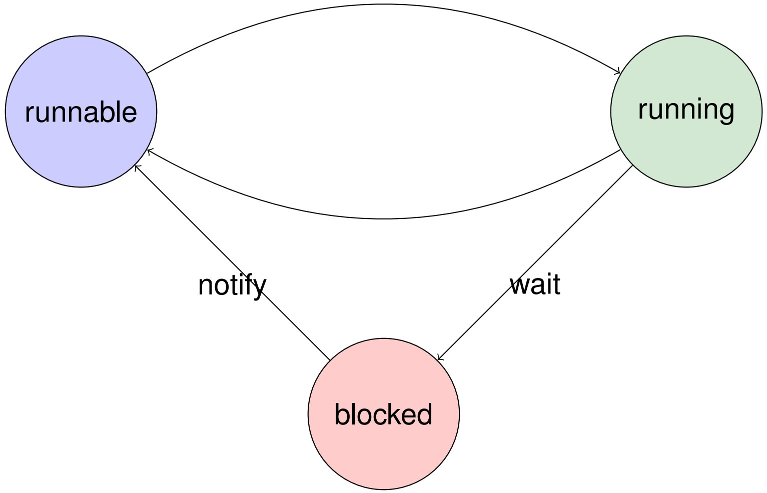
Wait and Notify

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



Software Engineering Accreditation

On Monday February 1, 2016, as part of the accreditation visit for our Software Engineering program, there will be a meeting of the visitors with Software Engineering students from 16:30 until 17:30. It would be great of you could be part of this meeting.