Lecture 7

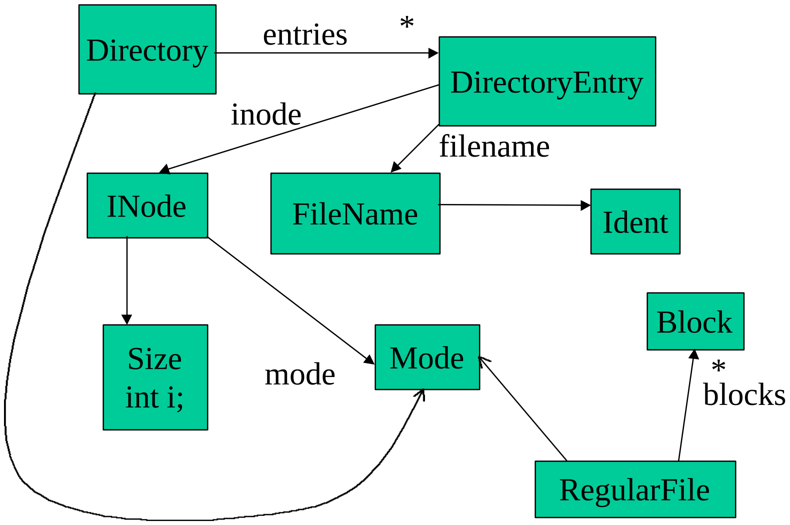
Discuss midterm

Scheduling

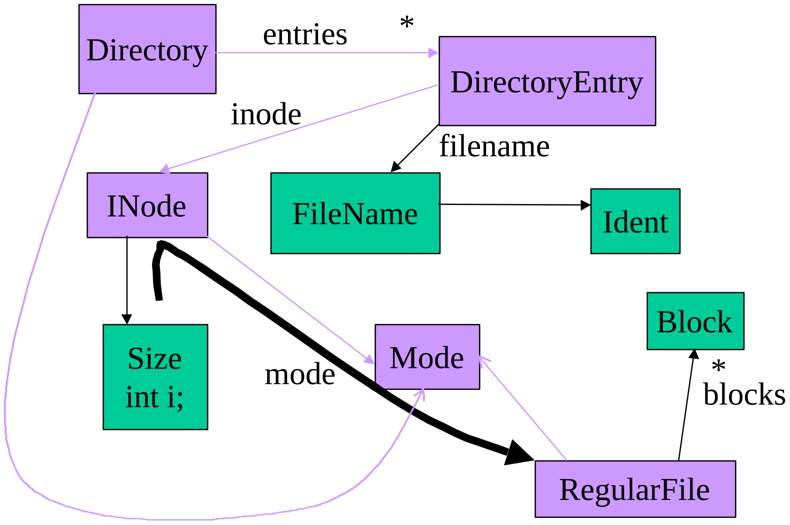
Alternative Directory Structure

- See hw 1 and hw 2.
- This one more aligned with UNIX directory structure.
- Idea for implementing processing is the same: at least conceptually separate traversal from maximum computation.

UNIX style directory structure

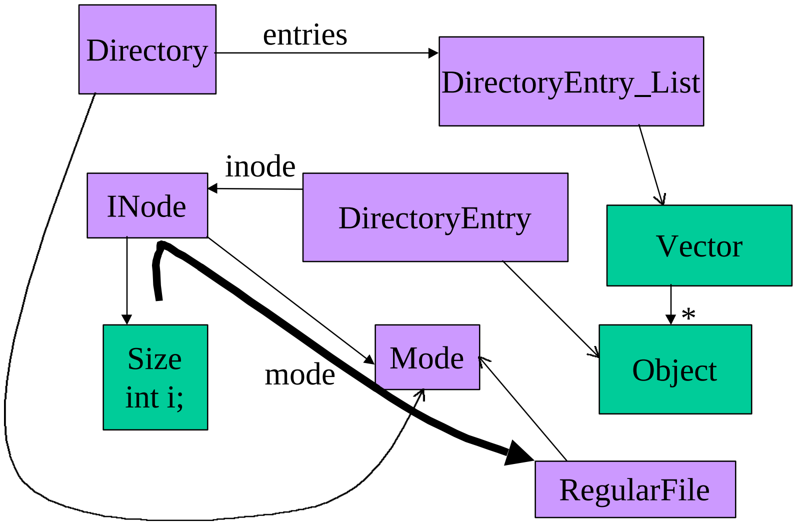


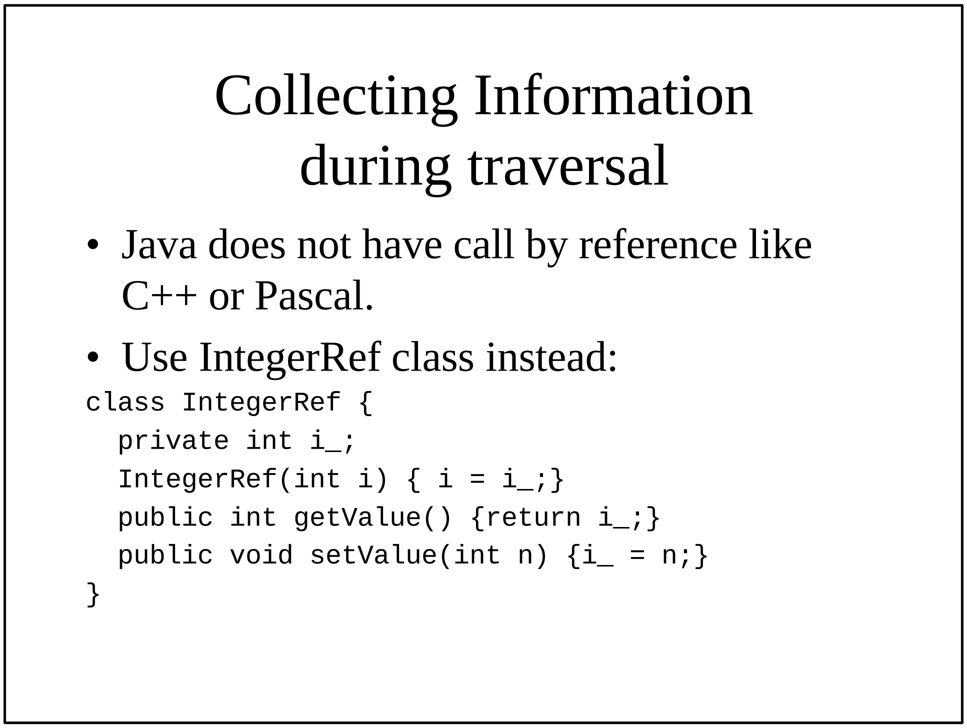
Maximum RegularFile size



Traversal: from Directory to RegularFile

Maximum RegularFile size: implementation structure





Collecting Information during traversal

- Java does not have call by reference like C++ or Pascal.
- Use IntegerRef class instead:

```
class IntegerRef {
  private int i_;
  IntegerRef(int i) { i = i_;}
  public int getValue() {return i_;}
  public void setValue(int n) {i_ = n;}
}
```

From IntegerRef to Visitor pattern

- Using class IntegerRef is not optimal: The code for computing the maximum is mixed with the traversal code.
- A better but more elaborate solution would be to apply the visitor design pattern: instead of an IntegerRef-object, we give a MaxVisitor-object to the traversal.

MaxVisitor

```
class MaxVisitor {
  private int max; int size;
  MaxVisitor() { max = 0;}
  public int get_result() {return max;}
  public void before(INode host)
    {size = host.get_size();}
  public void before(RegularFile host)
    {if (size > max) max=size;}
}
// this visitor takes care of transporting size
// the traversal is: from Directory to RegularFile
```

DJ

- If you use a library like DJ you can easily program in this style.
- DJ: www.ccs.neu.edu/research/demeter/DJ
- To learn about software development technology in general and Demeter in particular, take the COM 3360 class in the fall of 2000. See my home page.

Vector in Java 2

- Vector now implements ListInterface. Use following idiom to iterate through list:

```
for (ListIterator I = this.listIterator();
  I.hasNext();) {
  … I.next(); …
}
```

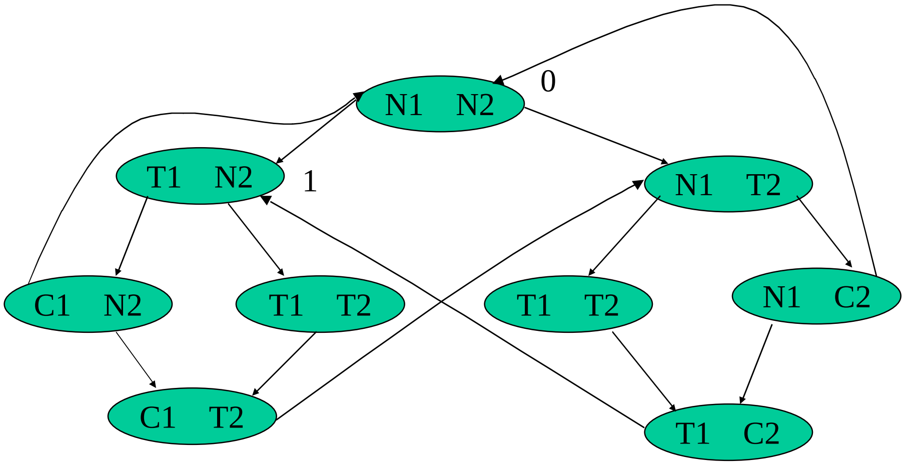
- Start using Collection framework as much as possible.

Semaphores in Java: Compare with WaitingStack

```
public final class CountingSemaphore {
  private int count_ = 0;
  public CountingSemaphore(int inC) {
    count_ = inC;}
  public void P() { // down -- pop
    while (count_ = 0)
      try {wait();}
        catch (InterruptedException ex) {}
      --count_;}
  public void V() { // up -- push
    ++count_; notifyAll(); }
}
```

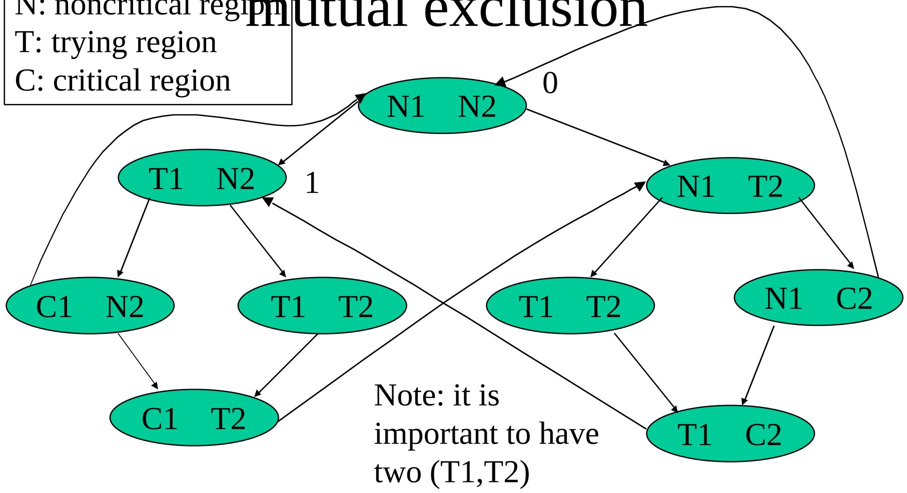
From: Concurrent Programming in Java by Doug Lea

Puzzle: what is this?



State modeling: two-process mutual exclusion

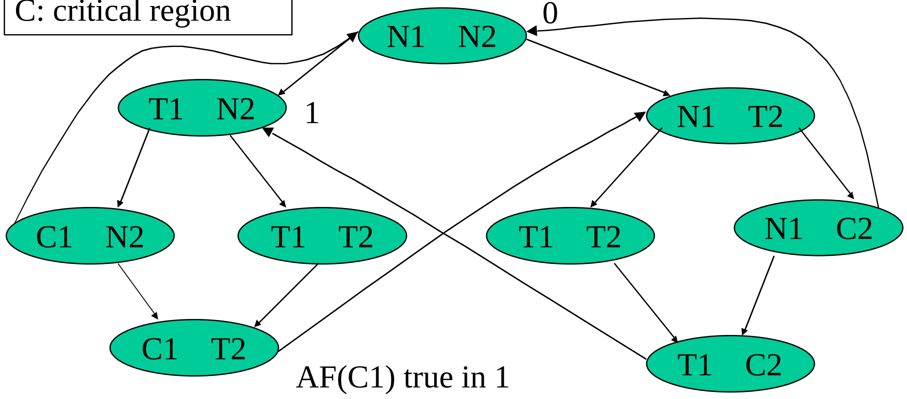
N: noncritical region
T: trying region
C: critical region



Note: it is important to have two (T1,T2)

State modeling: two-process mutual exclusion

N: noncritical region
T: trying region
C: critical region



AF(C1) true in 1
EF(C1 and C2) false in 0

Reasoning about concurrency

- Abstract from code
- Computation tree logic reasons about systems at this level
- Uses model-checking techniques

Symbolic Model Checking

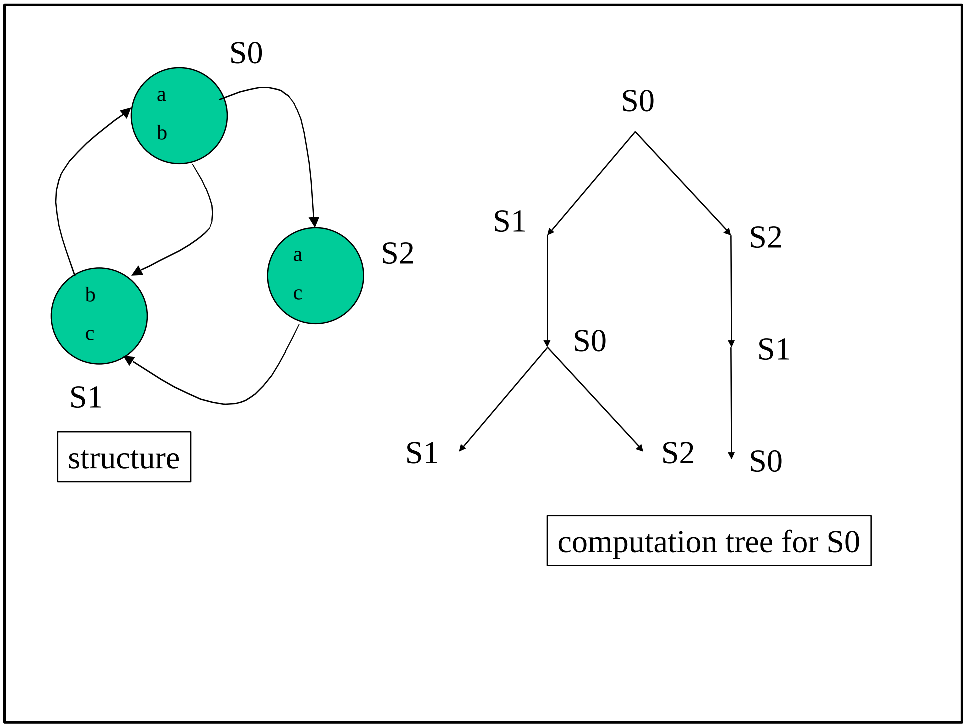
- Determine correctness of finite state systems.
- Developed at Harvard and later at CMU by Clarke/Emerson/Sistla
- Specifications are written as formulas in a propositional temporal logic.
- Temporal logic: expressing ordering of events without introducing time explicitly

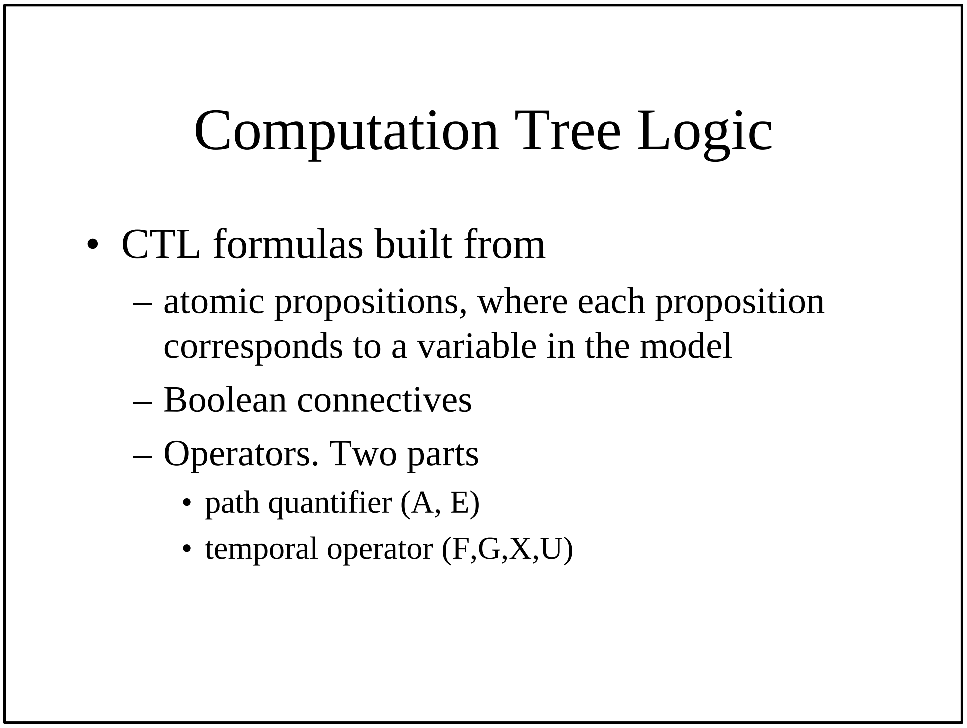
Temporal Logic

- A kind of modal logic. Origins in Aristotle and medieval logicians. Studied many modes of truth.
- Modal logic includes propositional logic. Embellished with operators to achieve greater expressiveness.
- A particular temporal logic: CTL (Computation Tree Logic)

Computation Tree Logic

- Used to express properties that will be verified
- Computation trees are derived from the state transition graphs
- State transition graphs unwound into an infinite tree rooted at initial state





Computation Tree Logic

- CTL formulas built from
  - atomic propositions, where each proposition corresponds to a variable in the model
  - Boolean connectives
  - Operators. Two parts
    - path quantifier (A, E)
    - temporal operator (F,G,X,U)

Computation Tree Logic

- Paths in tree represent all possible computations in model.
- CTL formulas refer to the computation tree

$$AG(req\ implies\ AF\ ack)$$

If the signal *req* is high then eventually *ack* will also be high