Software Model Checking

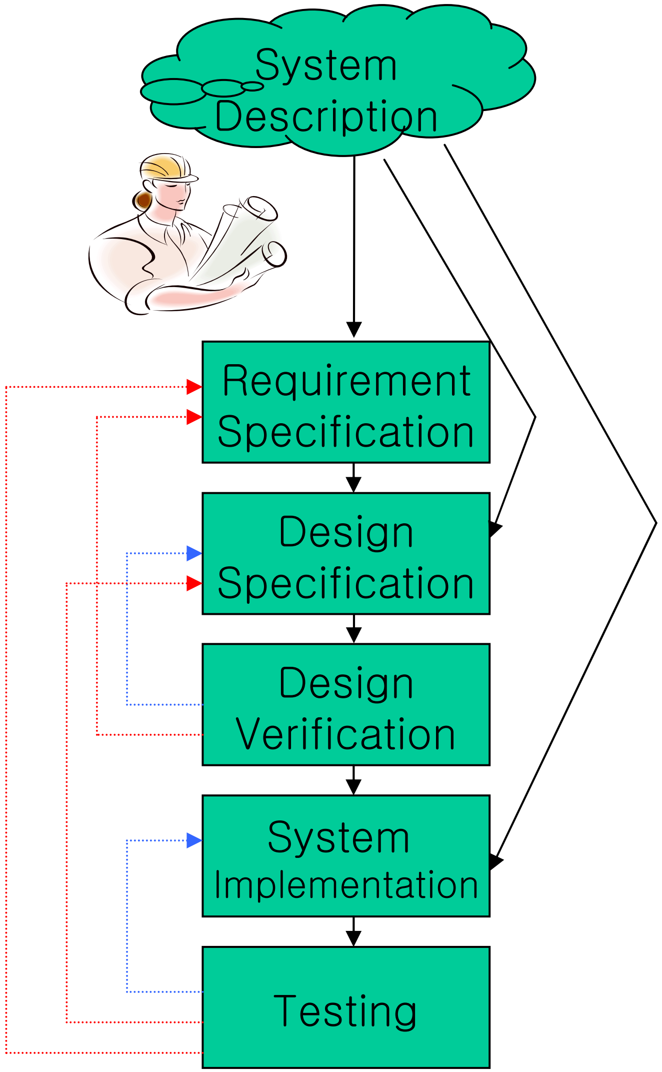
Case Study of Reader/Writer System

Moonzoo Kim

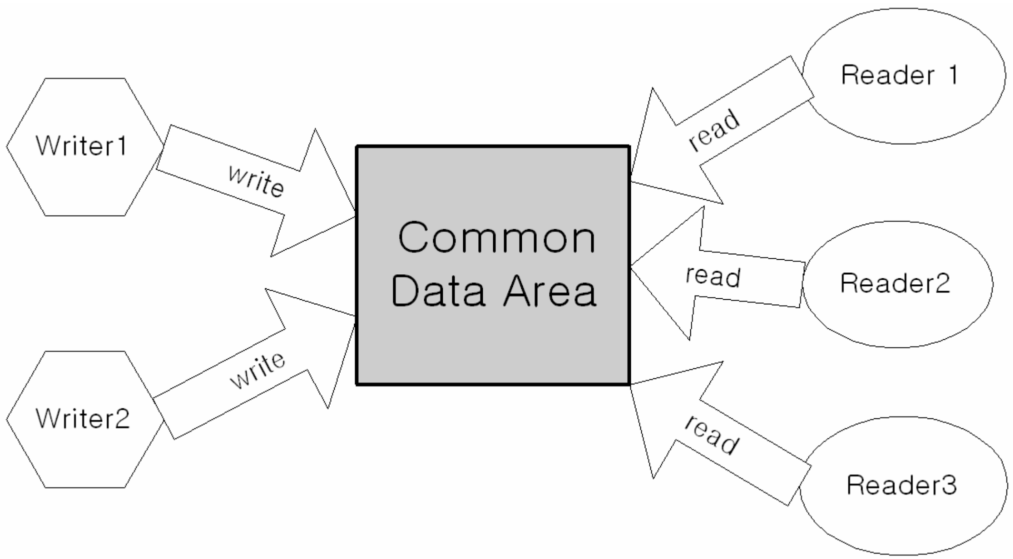
CS Dept. KAIST

Outlines

- System Description
- Formal Requirement Specification
- Formal Design Specification
- Formal Verification
- Testing



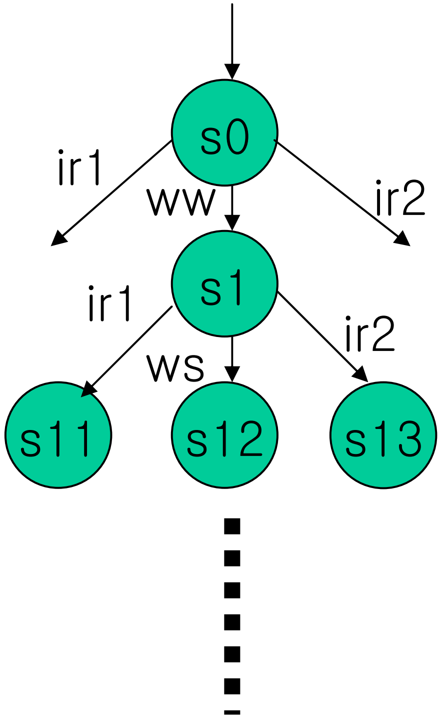
Multiple Reader/Writer System



- System requirement
  - Concurrency (CON)
  - Exclusive writing (EW)
  - High priority of writer (HPW)

Formal Requirement Specification

- 1 writer and 2 readers system
  - Execution tree
  - RW system has 9 events
    - {ir1,rs1,re1,ir2,rs2,re2,ww,ws,we}
  - A state s = $(n_{ir1},n_{rs1},n_{ir2},n_{rs2},n_{ww},n_{ws})$
    - s0 = (0,0,0,0,0,0)
    - s1 = (0,0,0,0,1,0)
    - s11=(1,0,0,0,1,0)
    - s12= (0,0,0,0,0,1)



Formal Requirement Specification (cont.)

Valid execution paths

Correct Event Ordering

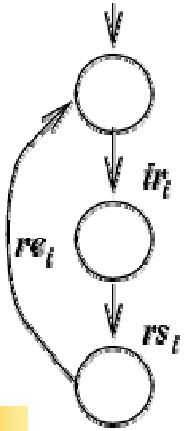
- $n_{ir1}(s_i)\geq 0 \wedge n_{rs1}(s_i)\geq 0 \wedge n_{ir1}(s_i)+n_{rs1}(s_i) \leq 1$
- $n_{ir2}(s_i)\geq 0 \wedge n_{rs2}(s_i)\geq 0 \wedge n_{ir2}(s_i)+n_{rs2}(s_i) \leq 1$
- $n_{ww}(s_i)\geq 0 \wedge n_{ws}(s_i)\geq 0 \wedge n_{ww}(s_i)+n_{ws}(s_i) \leq 1$

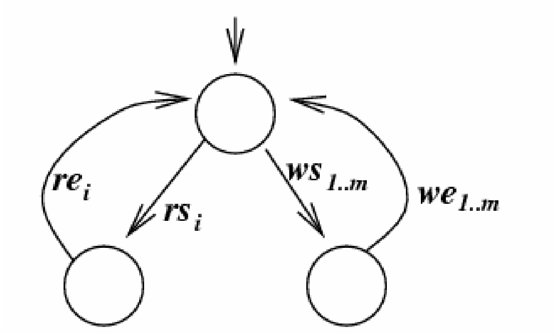
Exclusive Writing

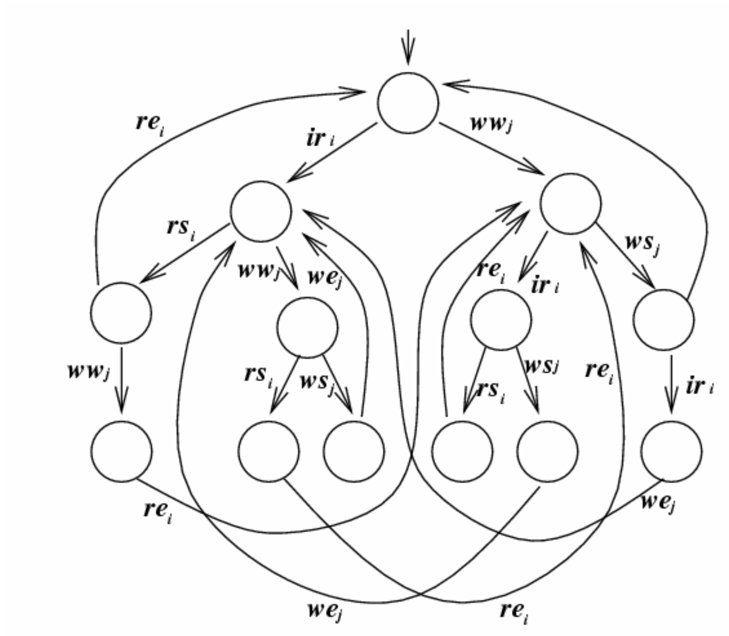
- $n_{ws}(s_i)=1 \rightarrow (n_{rs1}(s_i)=0 \wedge n_{rs2}(s_i)=0)$

High Priority of Writer

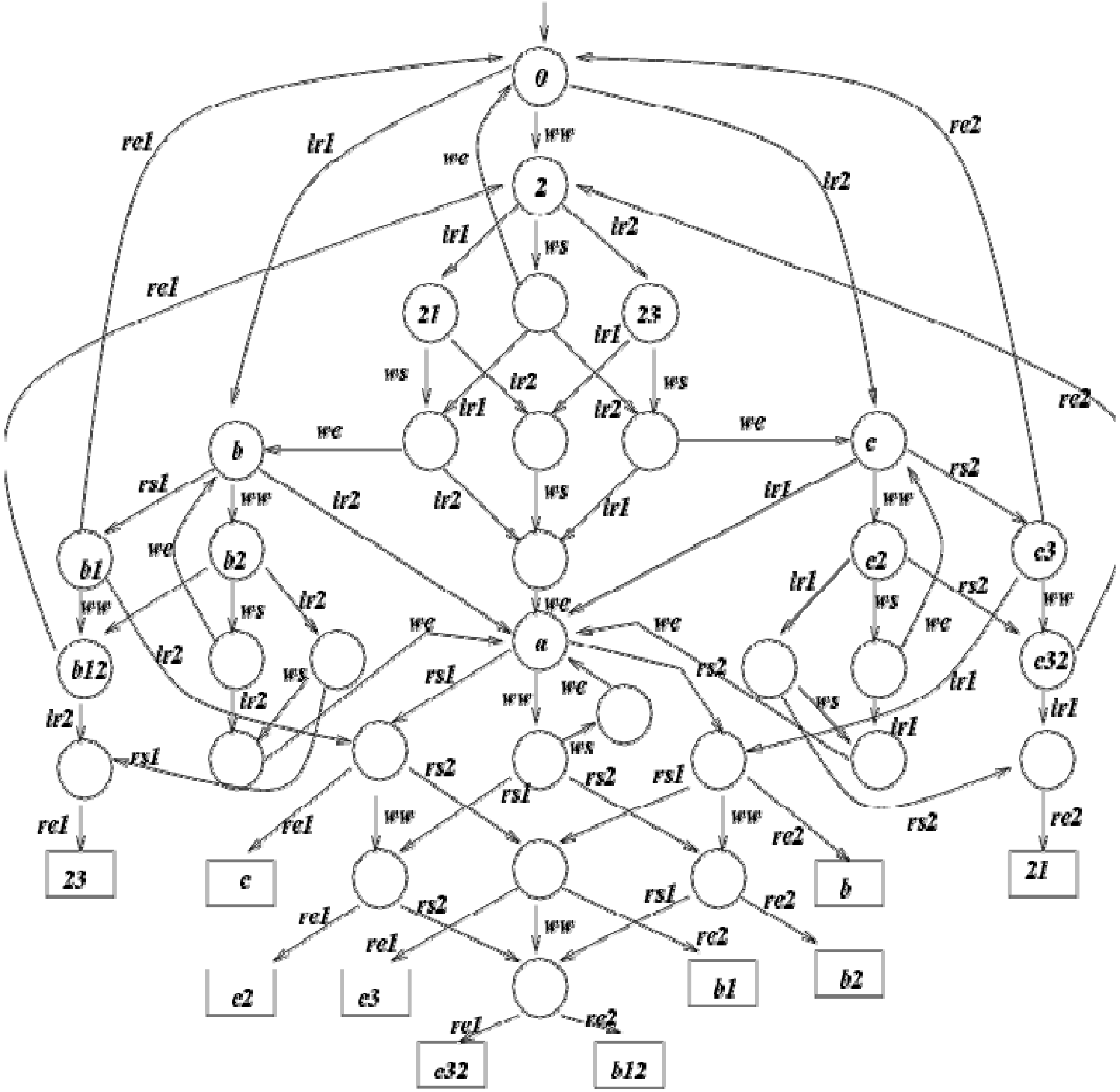
- $(n_{ww}(s_i)=1 \wedge n_{rs1}(s_i)=0 \wedge n_{rs2}(s_i)=0)$
  $\rightarrow (n_{rs1}(s_{i+1})=0 \wedge n_{rs2}(s_{i+1})=0)$







Formal Requirement Specification (cont.)



Formal Requirement Specification (cont.)

```
*****************************
* Requirement Specification  *
*****************************
proc S0 = ir1.B + ww.S2 + ir2.C
proc S2 = ir1.S21 + ws.S22 + ir2.S23
proc S21 = ws.S212 + ir2.S213
proc S22 = ir1.S212 + we.S0+ ir2.S232
proc S23 = ir1.S213 + ws.S232
proc S212 = we.B + ir2.S2123
proc S213 = ws.S2123
proc S232 = ir1.S2123 + we.C
proc S2123 = we.A

proc A = rs1.A1 + ww.A2 + rs2.A3
proc A1 = re1.C + ww.A12 + rs2.A13
proc A2 = rs1.A12 + ws.we.A  + rs2.A32
proc A3 = rs1.A13 + ww.A32 + re2.B
proc A12 = re1.C2 + rs2.A123
proc A13 = re1.C3 + ww.A123 + re2.B1
proc A32 = rs1.A123 + re2.B2
proc A123 = re1.C32 + re2.B12
```

```
proc B = rs1.B1 + ww.B2 + ir2.A
proc B1 = re1.S0 + ww.B12 + ir2.A1
proc B2 = rs1.B12 + ws.B22 + ir2.B23
proc B12 = re1.S2 + ir2.B123
proc B22 = we.B + ir2.B223
proc B23 = ws.B223 + rs1.B123
proc B123 = re1.S23
proc B223 = we.A

proc C = ir1.A + ww.C2 + rs2.C3
proc C2 = ir1.C21 + ws.C22 +rs2.C32
proc C3 = ir1.A3 + ww.C32 + re2.S0
proc C21 = ws.C221 + rs2.C321
proc C22 = ir1.C221 + we.C
proc C32 = ir1.C321 + re2.S2
proc C221 = we.A
proc C321 = re2.S21
```

Formal Design Specification

- RW system designed in “Concurrent Programming in Java[Lea99]”
- proc S =

  (R1|R2|W|AR0|WW0|AW0|

  LOCK|SLEEP0)\

  { dec_WW, inc_WW, dec_AW,inc_AW,…}

  proc R1 = …

  - Processes (R1, R2, W, Lock, etc) communicate each other through signals

  (dec_WW, inc_WW, etc)

  - variables in RW code are represented as processes (AR0, AW0, etc)

```
class RW {
   int activeReaders_ = 0;
   int activeWriters_= 0;
   int waitingReaders_= 0;
   int waitingWriters_ = 0;

   void read()    {
           beforeRead();
           read_();
           afterRead();
   }

   void beforeRead()         {…}
   void read_()              {…}
   void afterRead()          {…}
           …
}
```

Testing using Formal Specification

- Insert probe into the RW source code
  - Probe generates event signal
- Testing RW code utilizing formal requirement spec as a test oracle
  - Use CWB-NC based simulation feature
  - Inappropriate event signal means violation

```
public abstract class RW{
    protected int activeReaders_ = 0;
    protected int activeWriters_= 0;
    protected int waitingReaders_= 0;
    protected int waitingWriters_ = 0;

    public void read(String id)    {
             beforeRead();
             read_(id);
             afterRead();
    }
    protected synchronized void beforeRead(){
             Event("ir");…
    }
    public void read_() {
              Event(“rs");…
    }
    protected synchronized void afterRead(){
              Event(“re");…
    }
…
}
```

RW Java Code

```
public abstract class RW2  {
    protected int activeReaders_ = 0;  //threads executing read_
    protected int activeWriters_= 0;      //always 0 or 1
    protected int waitingReaders_= 0; //threads not yet in read_
    protected int waitingWriters_ = 0;  //same for write_

    protected abstract void read_(String id);
    protected abstract void write_(String id);

    void Event(String s){ }//System.out.println(s);}

    public void read(String id)    {
        /*Event("ir" + id);*/  beforeRead();
        /*Event("rs" + id);*/ read_(id);
        /*Event("re" + id);*/ afterRead();
    }

    public void write(String id)    {
        /*Event("ww");*/   beforeWrite();
        /*Event("ws");*/   write_(id);
        /*Event("we");*/   afterWrite();
    }

    protected boolean allowReader()    {
        if (waitingWriters_ == 0 && activeWriters_ == 0)      {
            return true;
        }
        else
            return false;
    }
```

```
protected boolean allowWriter()    {
    if(activeReaders_ == 0 && activeWriters_ == 0) {
        return true;
    } else return false;
}

protected synchronized void beforeRead()    {
    ++waitingReaders_;
    while(!allowReader())
         try{ wait();}
         catch(InterruptedException ex){}
    --waitingReaders_;
    ++activeReaders_;
}

protected synchronized void afterRead()    {
    --activeReaders_;
    notifyAll();
}

protected synchronized void beforeWrite()    {
    ++waitingWriters_;
    while(!allowWriter())
        try{wait();}
        catch(InterruptedException ex){}
    --waitingWriters_;
    ++activeWriters_;
}

protected synchronized void afterWrite()    {
    --activeWriters_;
    notifyAll();
}
```

RW System Design

```
*****************************************************
* RW system description of 2 Readers and 1 Writer *
*****************************************************
proc S = (R1|R2|W|AR0|WW0|AW0|LOCK|SLEEP0)₩
                  {dec_WW, inc_WW, dec_AW,inc_AW, dec_AR, inc_AR,
                   zero_WW, zero_AW, zero_AR, non_zero_WW,non_zero_AW,
                   non_zero_AR, lock, unlock,
                   zero_sleep, one_sleep, two_sleep, dec_sleep, inc_sleep,
                   wake_up}

proc WW0 = zero_WW.WW0 + inc_WW.WW1
proc WW1 = dec_WW.WW0 + non_zero_WW.WW1

proc AW0 = zero_AW.AW0 + inc_AW.AW1
proc AW1 = dec_AW.AW0 + non_zero_AW.AW1

proc AR0 = zero_AR.AR0 + inc_AR.AR1
proc AR1 = dec_AR.AR0 + inc_AR.AR2
        + non_zero_AR.AR1
proc AR2 = dec_AR.AR1 + non_zero_AR.AR2

proc SLEEP0 = zero_sleep.SLEEP0 + inc_sleep.SLEEP1
proc SLEEP1 = one_sleep.SLEEP1 + inc_sleep.SLEEP2 + dec_sleep.SLEEP0
proc SLEEP2 = two_sleep.SLEEP2 + dec_sleep.SLEEP1

proc R1 =     'lock.ir1.
     (   'zero_WW.
                   ('zero_AW.'inc_AR.'unlock.READ1
                      + 'non_zero_AW.'inc_sleep.'unlock.R1')
                   + 'non_zero_WW.'inc_sleep.'unlock.R1')
proc R1' =     wake_up.'lock.
     (   'zero_WW.
                   ('zero_AW.'inc_AR.'unlock.READ1
                      + 'non_zero_AW.'inc_sleep.'unlock.R1')
                   + 'non_zero_WW.'inc_sleep.'unlock.R1')
```

```
proc R2 =    'lock.ir2.
    (   'zero_WW.
         ('zero_AW.'inc_AR.'unlock.READ2
            + 'non_zero_AW.'inc_sleep.'unlock.R2')
         + 'non_zero_WW.'inc_sleep.'unlock.R2')
proc R2' =     wake_up.'lock.
     (   'zero_WW.
          ('zero_AW.'inc_AR.'unlock.READ2
             + 'non_zero_AW.'inc_sleep.'unlock.R2')
          + 'non_zero_WW.'inc_sleep.'unlock.R2')

proc W =    'lock.ww.'inc_WW.
   ( 'zero_AR.
        ('zero_AW.'dec_WW.'inc_AW.'unlock.WRITE
           +'non_zero_AW.'inc_sleep.'unlock.W')
        + 'non_zero_AR.'inc_sleep.'unlock.W')
proc W' =     wake_up.'lock.
     ( 'zero_AR.
          ('zero_AW.'dec_WW.'inc_AW.'unlock.WRITE
             +'non_zero_AW.'inc_sleep.'unlock.W')
          + 'non_zero_AR.'inc_sleep.'unlock.W')

proc READ1 = rs1.re1.'lock.'dec_AR.
         ('zero_sleep.'unlock.R1 +
          'one_sleep.'wake_up.'dec_sleep.'unlock.R1 +
          'two_sleep.'wake_up.'dec_sleep.'wake_up.'dec_sleep.'unlock.
        R1)
proc READ2 = rs2.re2.'lock.'dec_AR.
         ('zero_sleep.'unlock.R2+
          'one_sleep.'wake_up.'dec_sleep.'unlock.R2+
          'two_sleep.'wake_up.'dec_sleep.'wake_up.'dec_sleep.'unlock.
        R2)
proc WRITE = ws.we.'lock.'dec_AW.
        ('zero_sleep.'unlock.W +
         'one_sleep.'wake_up.'dec_sleep.'unlock.W +
         'two_sleep.'wake_up.'dec_sleep.'wake_up.'dec_sleep.'unlock.
        W)

proc LOCK = lock.unlock.LOCK
```

Formal Verification Result

```
cwb-nc> le -S may S S0
Building automaton...
States: 620
Transitions: 1016
Done building automaton.
Building automaton...
States: 34
Transitions: 75
Done building automaton.
Transforming automaton...
Done transforming automaton
TRUE
cwb-nc>
```

```
cwb-nc> le -S may S0 S
Building automaton...
States: 34
Transitions: 75
Done building automaton.
Building automaton...
States: 620
Transitions: 1016
Done building automaton.
Transforming automaton...
Done transforming automaton.
FALSE...
S0 has trace:
     ir1 ww ws
S does not.
cwb-nc>
```