Specifying and Exploiting Advice-Execution Ordering using Dependency State Machines

Eric Bodden

Finite-state properties

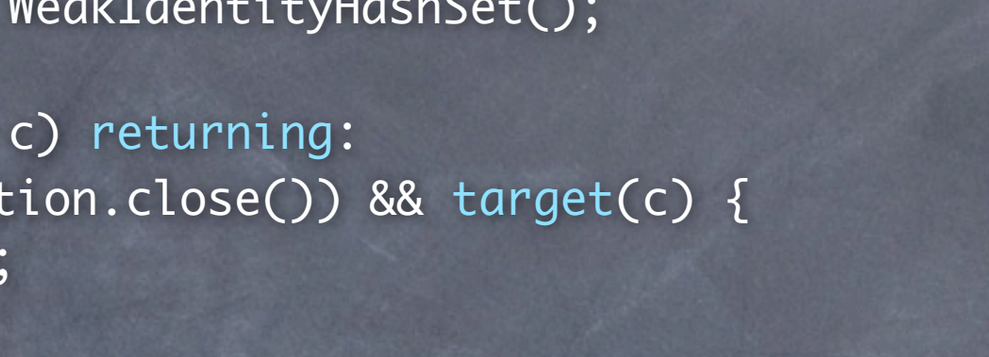
"When disconnecting a connection c, don't write to c until c is reconnected."

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

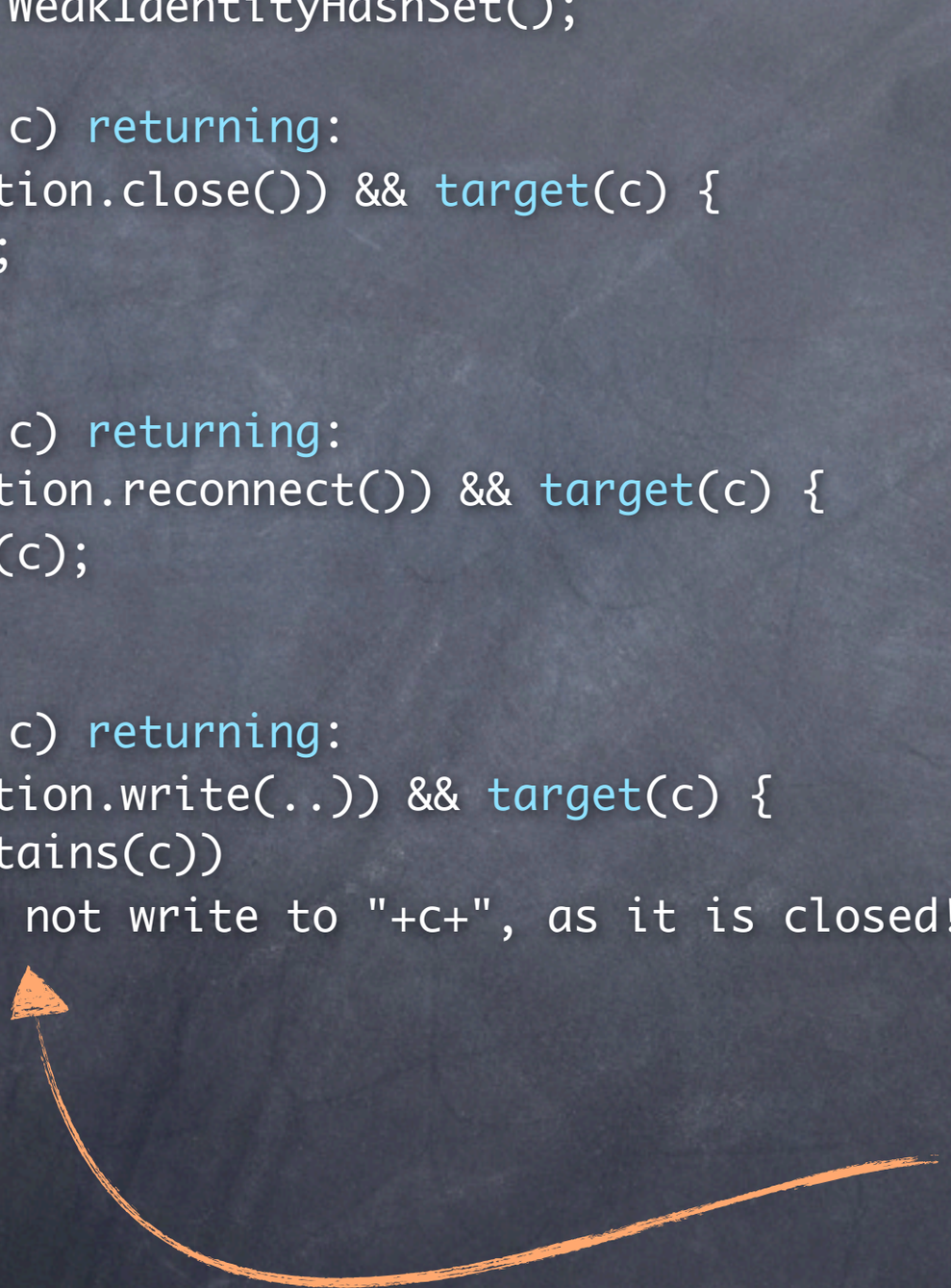


only "externally visible" code

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

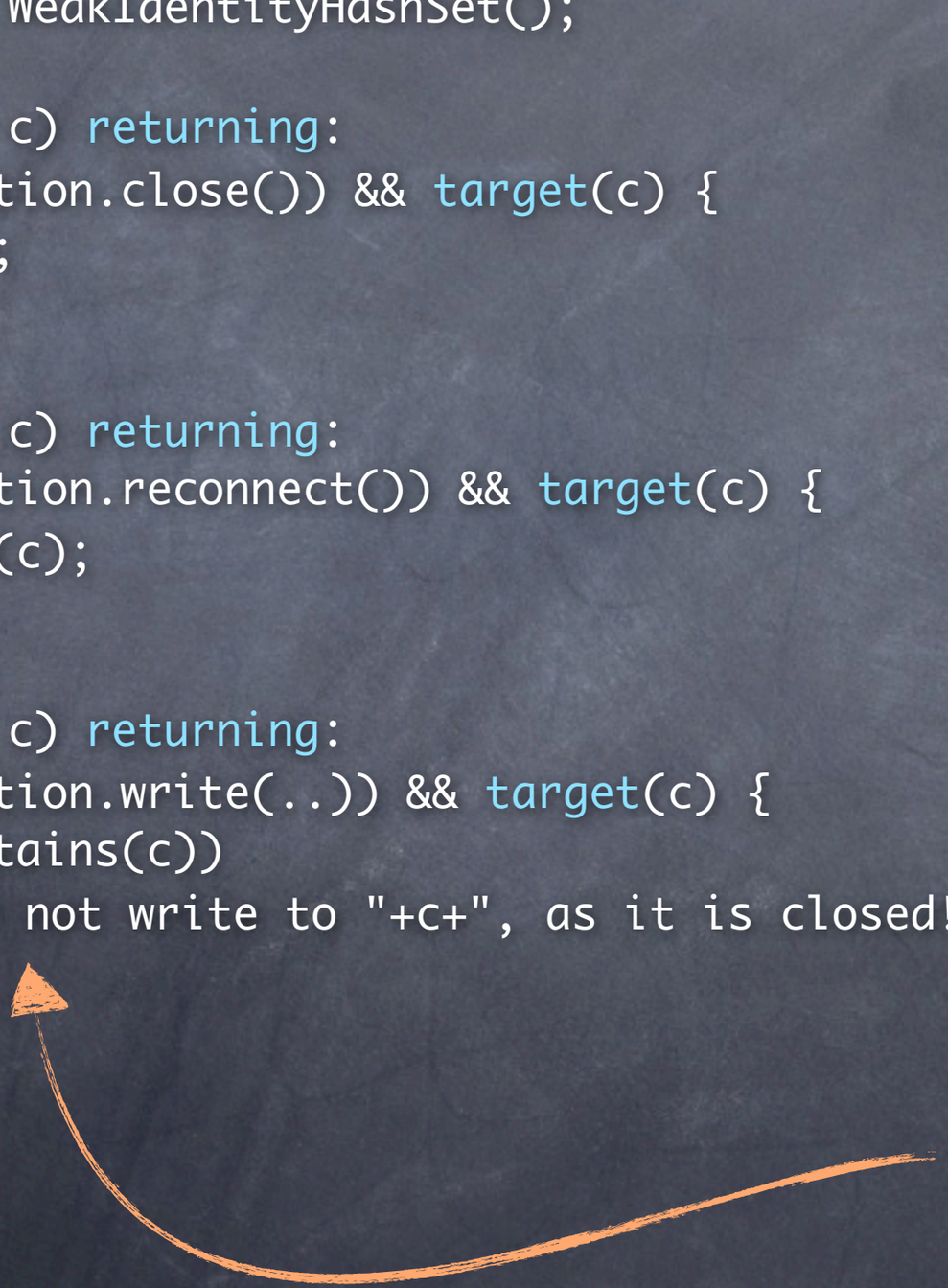
```
c.write(..);
```



Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

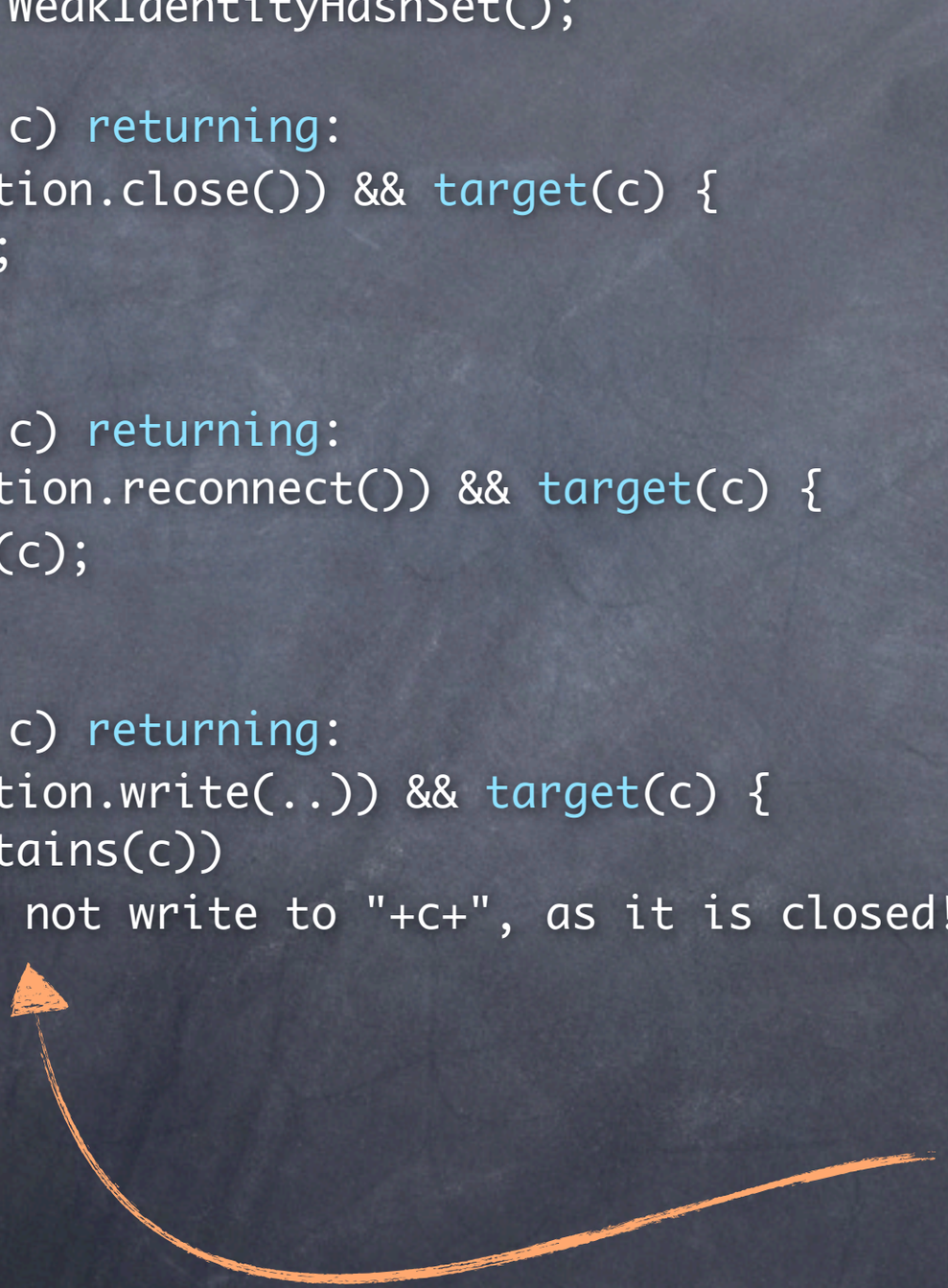
```
c.write(..);

c.close();
```



Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);

c.close();

c.reconnect();
```



Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);
c.close();
c.reconnect();
c.write(..);
```

only "externally visible" code

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);
c.close();
c.reconnect();
c.write(..);
c.write(..);
```

only "externally visible" code

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);
c.close();
c.reconnect();
c.write(..);
c.write(..);
c.close();
```

only "externally visible" code

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);
c.close();
c.reconnect();
c.write(..);
c.write(..);
c.close();
c.write(..);
```

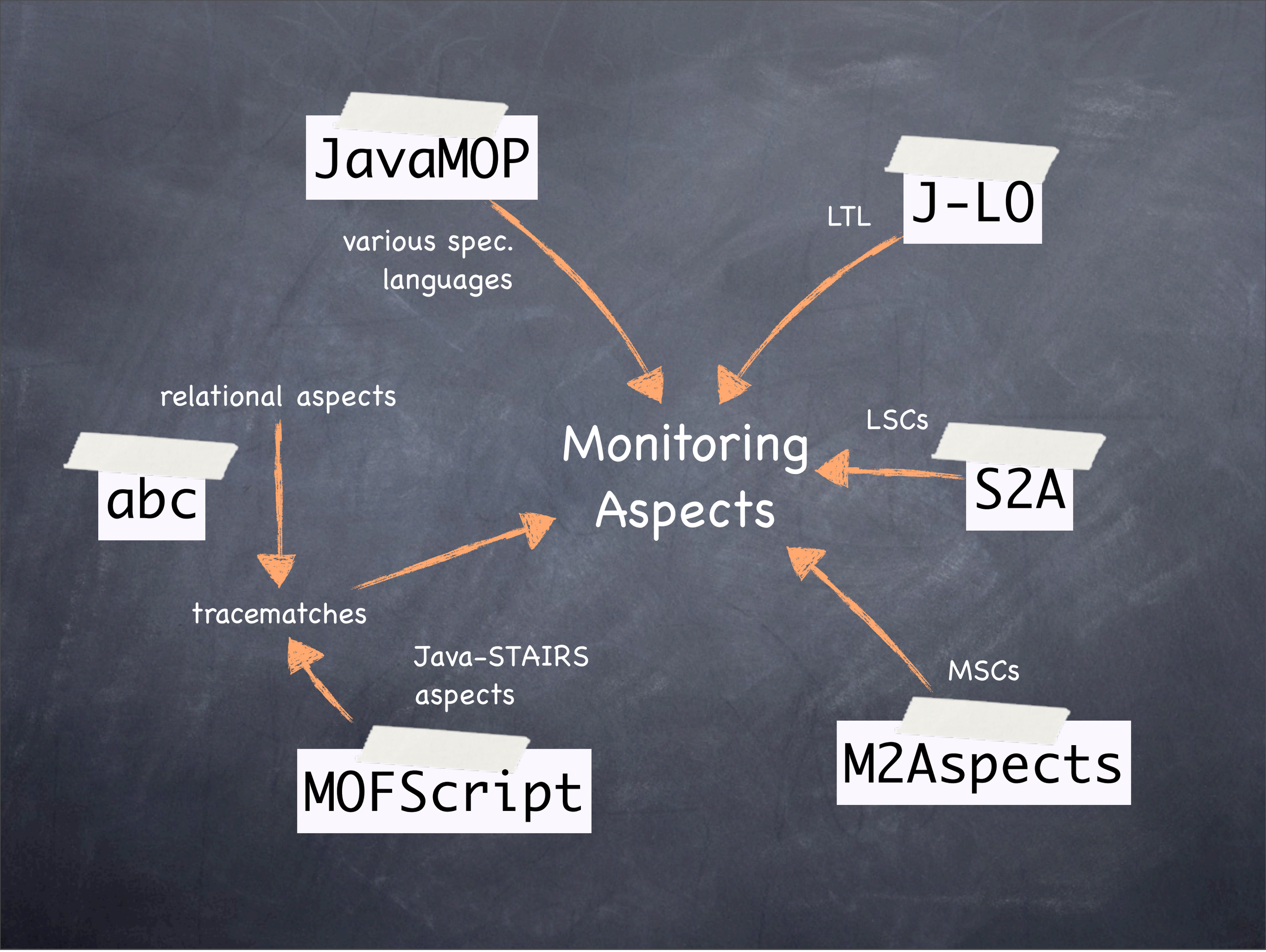
only "externally visible" code

Advice-execution ordering matters

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
c.write(..);
c.close();
c.reconnect();
c.write(..);
c.write(..);
c.close();
c.write(..);
```

only "externally visible" code


JavaMOP
various spec. languages
LTL
J-LO
relational aspects
abc
Monitoring Aspects
LSCs
S2A
tracematches
Java-STAIRS aspects
MSCs
MOFScript
M2Aspects

Dependency State Machines (DSMs)

```
Set closed = new WeakIdentityHashSet();

after(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}
```

```
after(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

after(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

Dependency State Machines (DSMs)

```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent after reconnect(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

dependent after write(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

Dependency State Machines (DSMs)

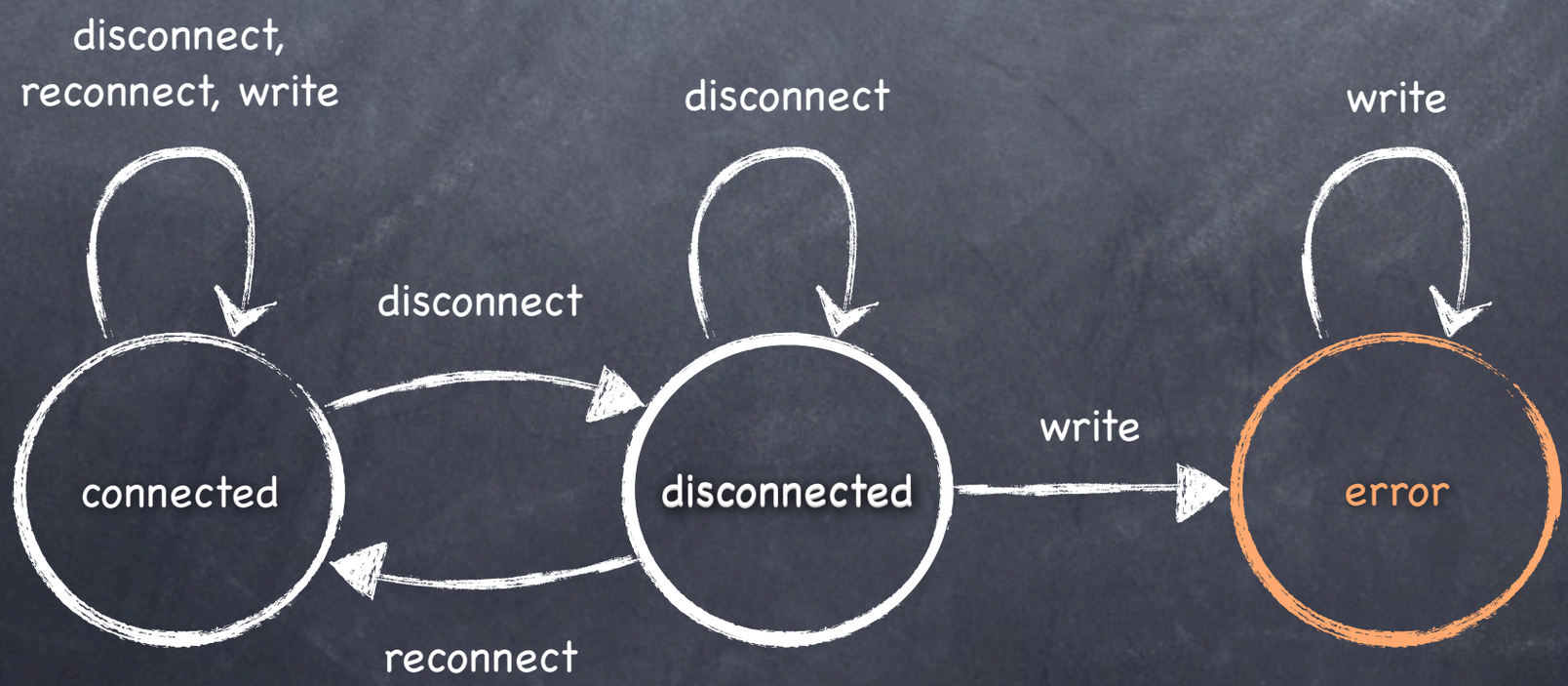
abstract

concrete

```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent after reconnect(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

dependent after write(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent after reconnect(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

dependent after write(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent after reconnect(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

dependent after write(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}
```

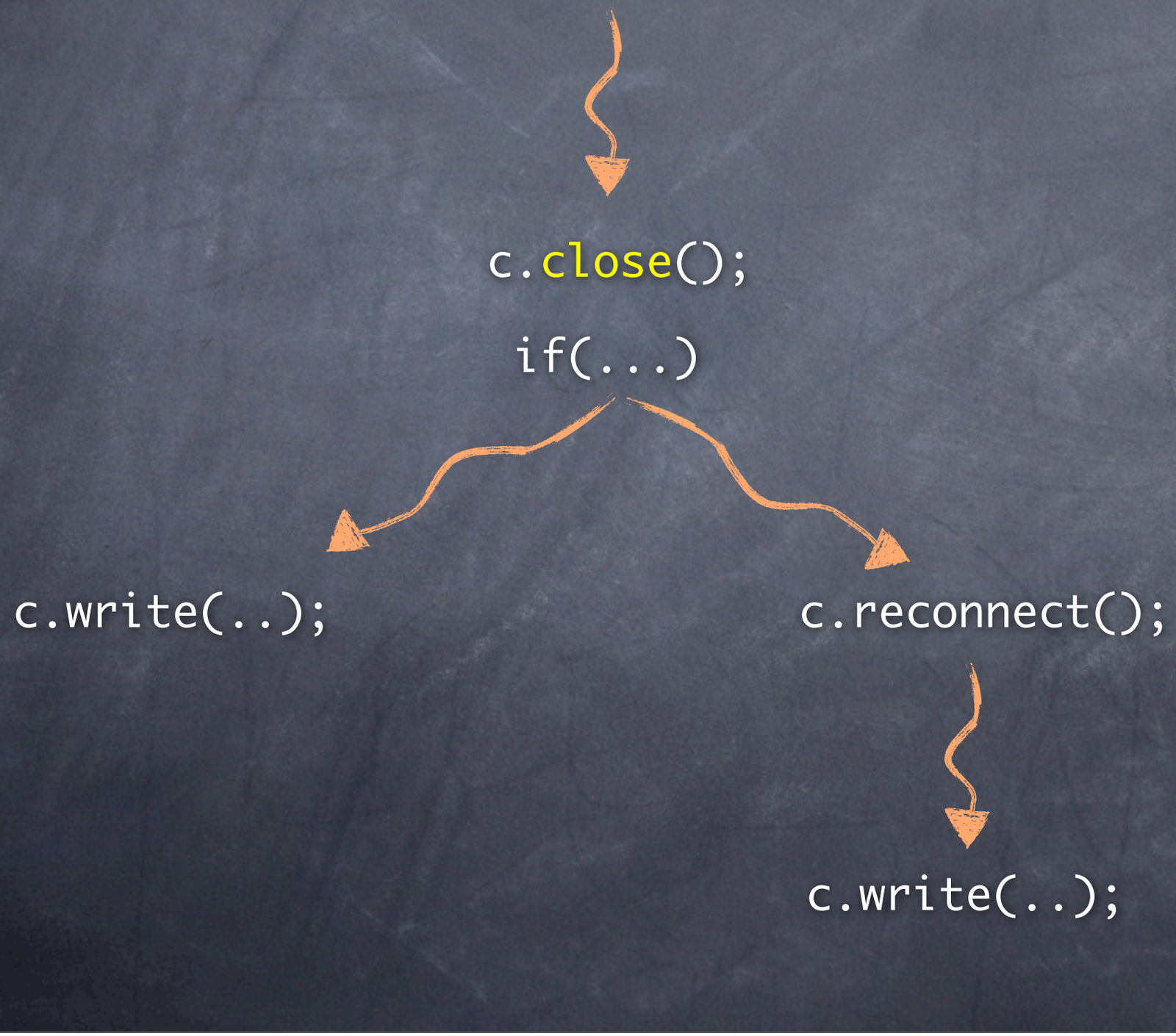


```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent
    call(
    closed
}

dependent
    call(
    if(cl
        er
}
```

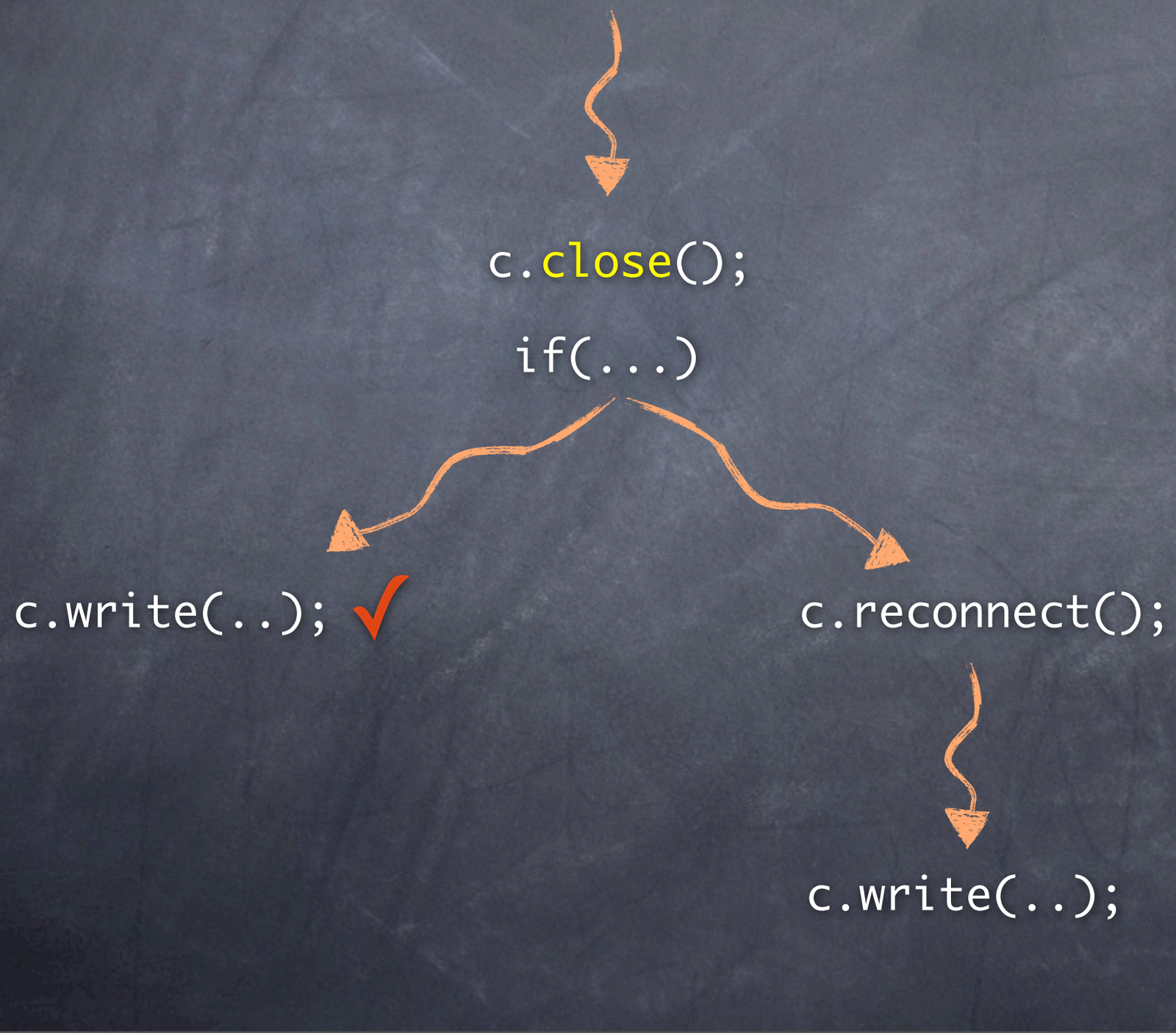
```
Vector monitorList = new Vector();
synchronized public void create(Iterator i, Collection v) {
    HashSet monitorSet = new HashSet();
    monitorList.add(new FailSafeIterMonitor());
    Iterator it = monitorList.iterator();
    while (it.hasNext()){
        FailSafeIterMonitor monitor = (FailSafeIterMonitor)it.next();
        monitor.create(i, v);
        if (monitorSet.contains(monitor) || monitor.failed())
        it.remove();
        else {
            monitorSet.add(monitor);
            if (monitor.suceeded()){
                //System.out.println("the collection is changed during iterating!");
            }
        } // for else
    } // for while
} // end of method
synchronized public void updatesource(Collection v) {
    HashSet monitorSet = new HashSet();
    Iterator it = monitorList.iterator();
    while (it.hasNext()){
        FailSafeIterMonitor monitor = (FailSafeIterMonitor)it.next();
        monitor.updatesource(v);
        if (monitorSet.contains(monitor) || monitor.failed())
        it.remove();
        else {
            monitorSet.add(monitor);
            if (monitor.suceeded()){
                //System.out.println("the collection is changed during iterating!");
            }
        } // for else
    } // for while
} // end of method
synchronized public void next(Iterator i) {
    HashSet monitorSet = new HashSet();
    Iterator it = monitorList.iterator();
    while (it.hasNext()){
        FailSafeIterMonitor monitor = (FailSafeIterMonitor)it.next();
        monitor.next(i);
        if (monitorSet.contains(monitor) || monitor.failed())
        it.remove();
        else {
            monitorSet.add(monitor);
            if (monitor.suceeded()){
                //System.out.println("the collection is changed during iterating!");
            }
        } // for else
    } // for while
} // end of method

ass FailSafeIterMonitor {
  /* %%_%_ERE_%_%% */
  int state = 0;
  public int hashCode() { return state; }
  public boolean equals(Object o) {
      if (! (o instanceof FailSafeIterMonitor)) return false;
```

```
Set closed = new WeakIdentityHashSet();

dependent after disconnect(Connection c) returning:
    call(* Connection.close()) && target(c) {
    closed.add(c);
}

dependent after reconnect(Connection c) returning:
    call(* Connection.reconnect()) && target(c) {
    closed.remove(c);
}

dependent after write(Connection c) returning:
    call(* Connection.write(..)) && target(c) {
    if(closed.contains(c))
        error("May not write to "+c+", as it is closed!");
}

dependency{
    disconnect, write, reconnect;
    initial   connected: disconnect -> connected,
                         write -> connected,
                         reconnect -> connected,
                         disconnect -> disconnected;
              disconnect: disconnect -> disconnected,
                         write -> error;
    final     error: write -> error;
}
```

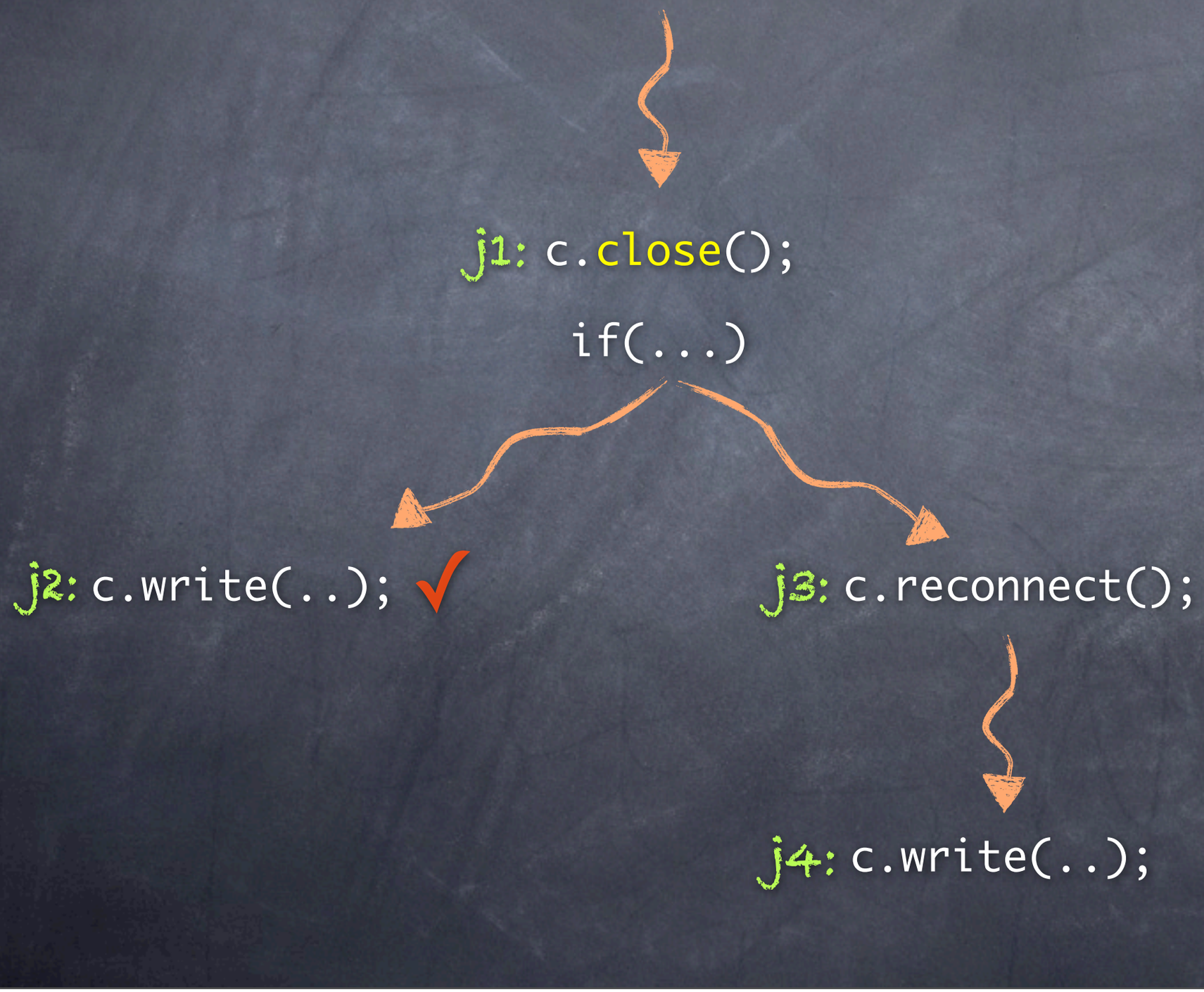
Semantics of Dependency State Machines



Semantics of Dependency State Machines

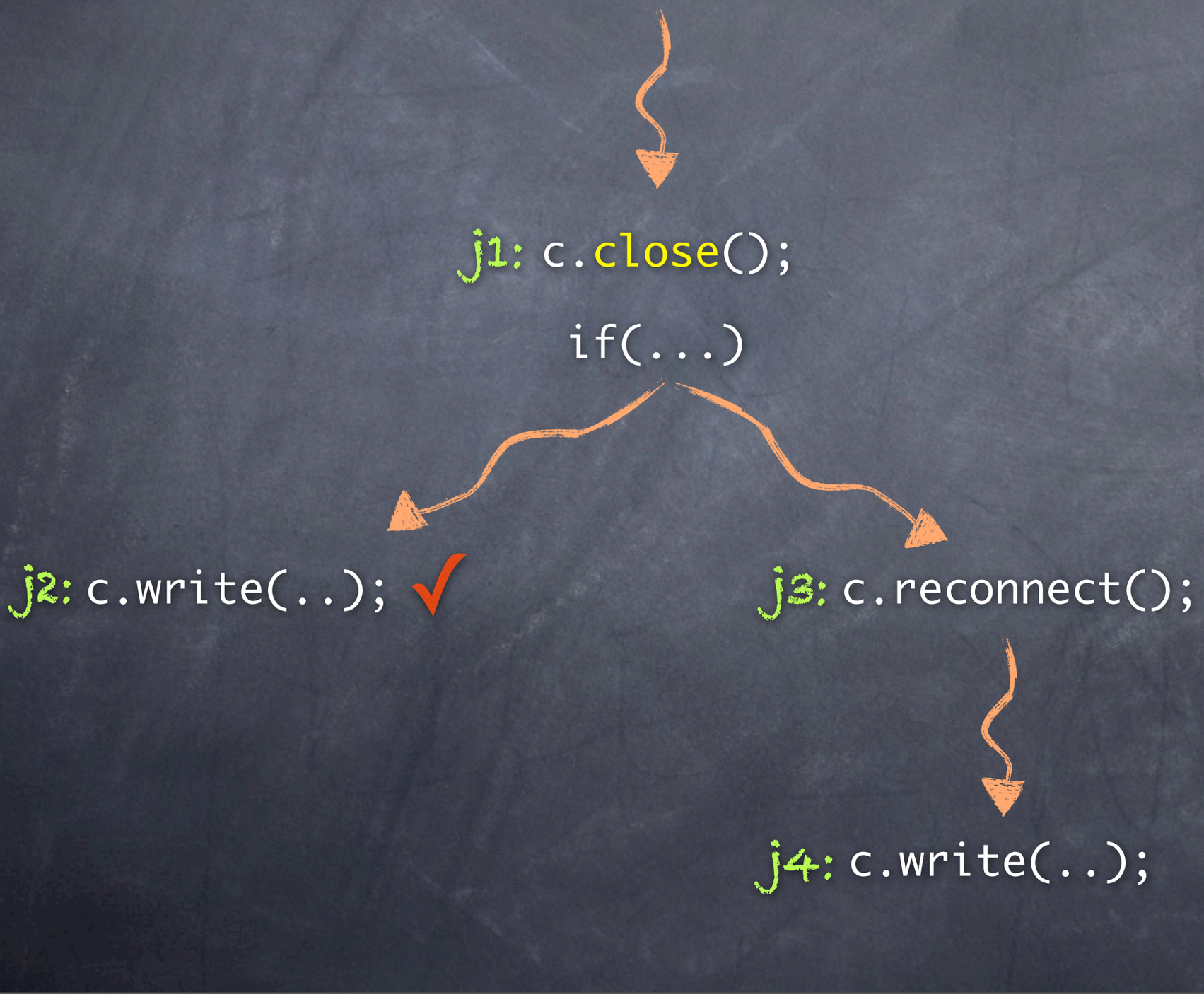


Semantics of Dependency State Machines



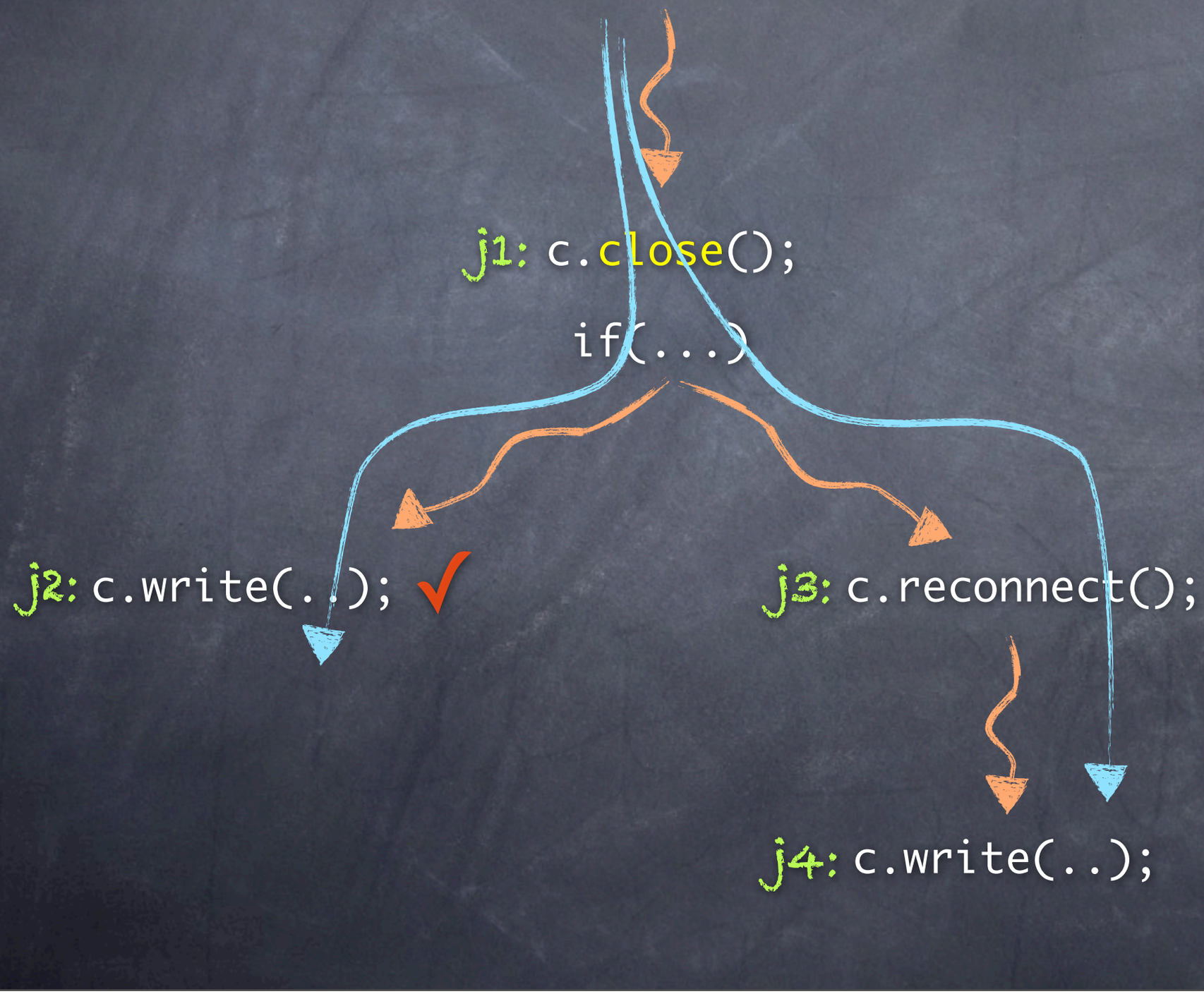
Semantics of Dependency State Machines

advice "close" must execute at j1 if



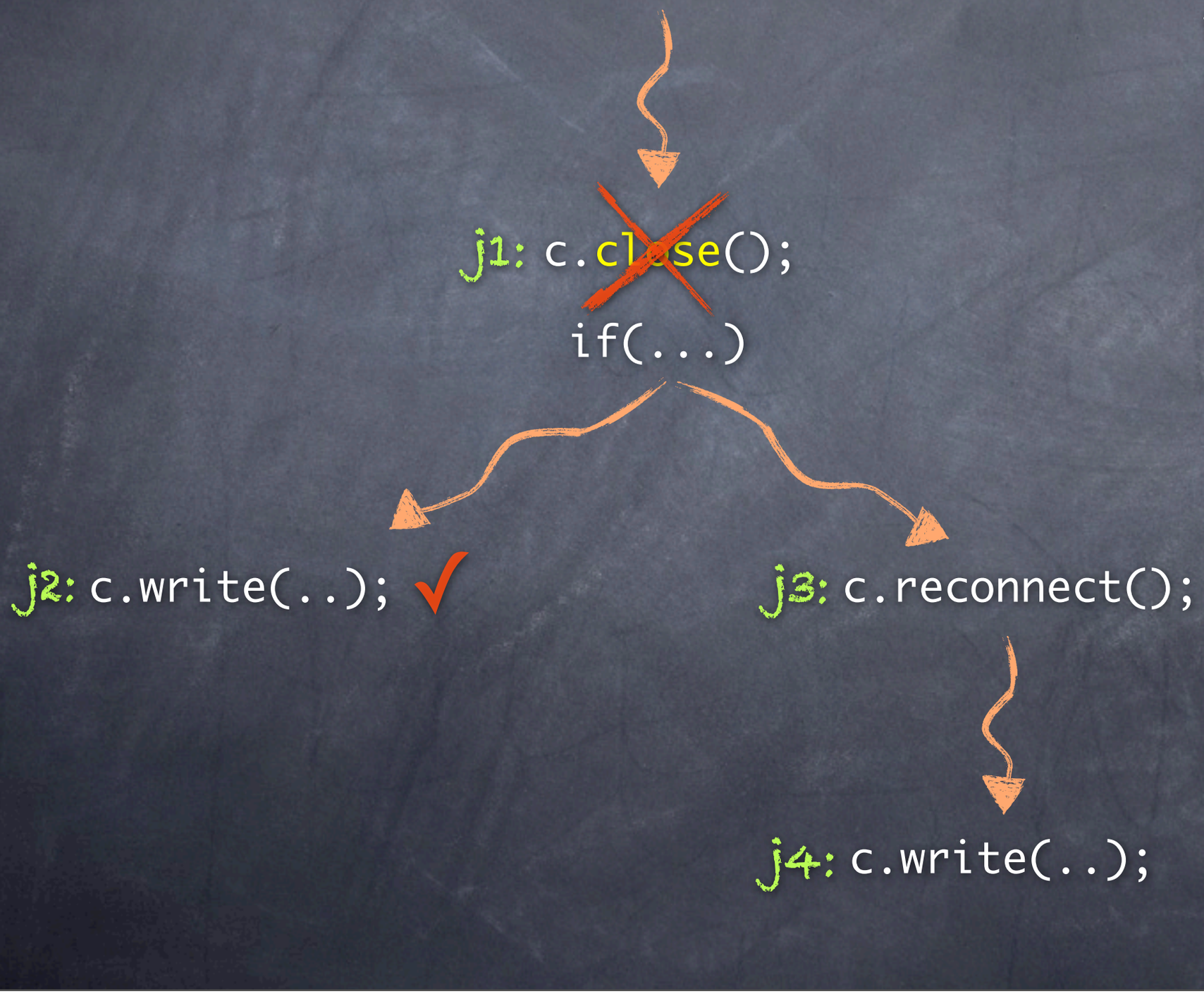
Semantics of Dependency State Machines

advice "close" must execute at j1 if there exists a path

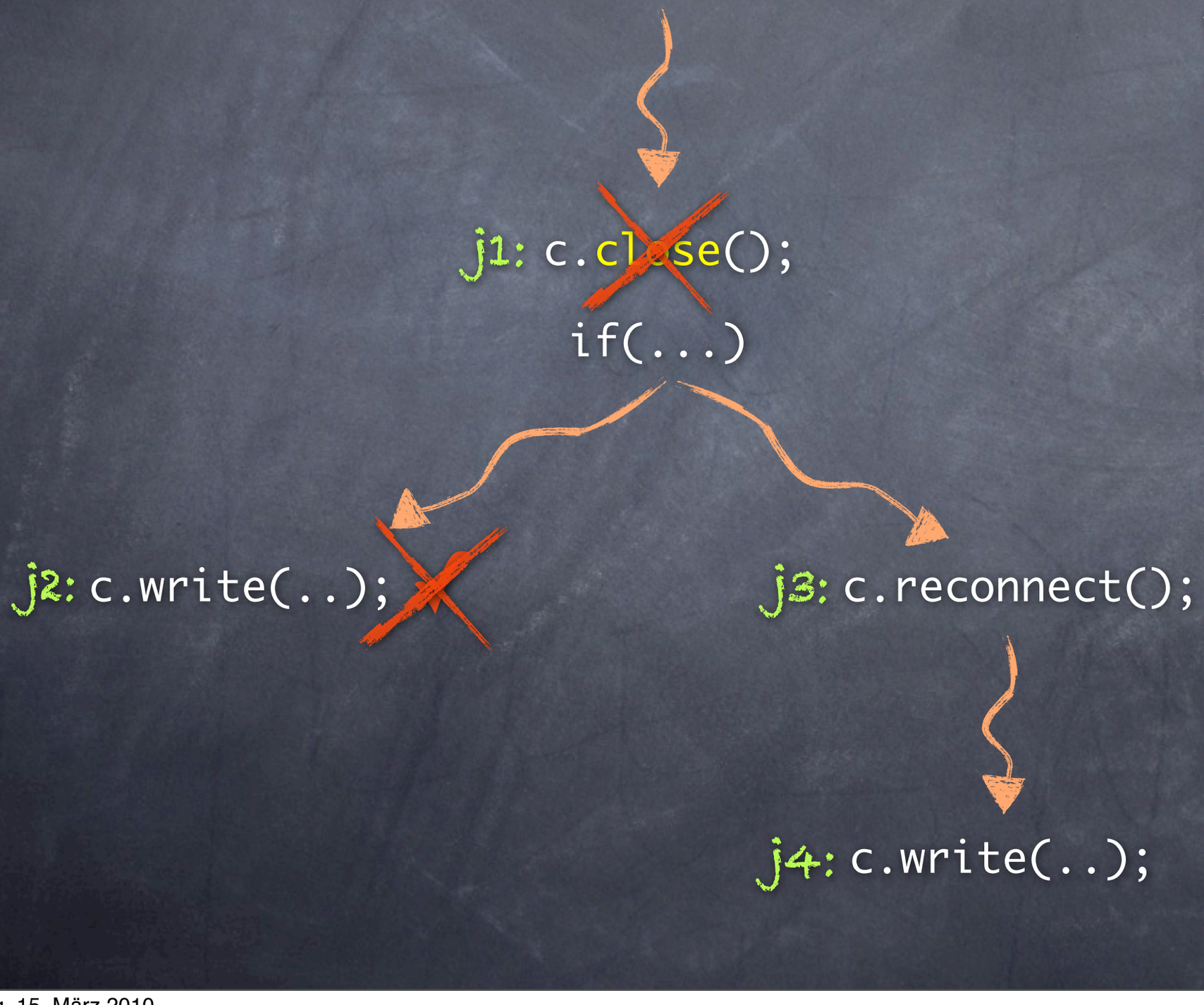


Semantics of Dependency State Machines

advice "close" must execute at j1 if there exists a path such that omitting "close" at j1

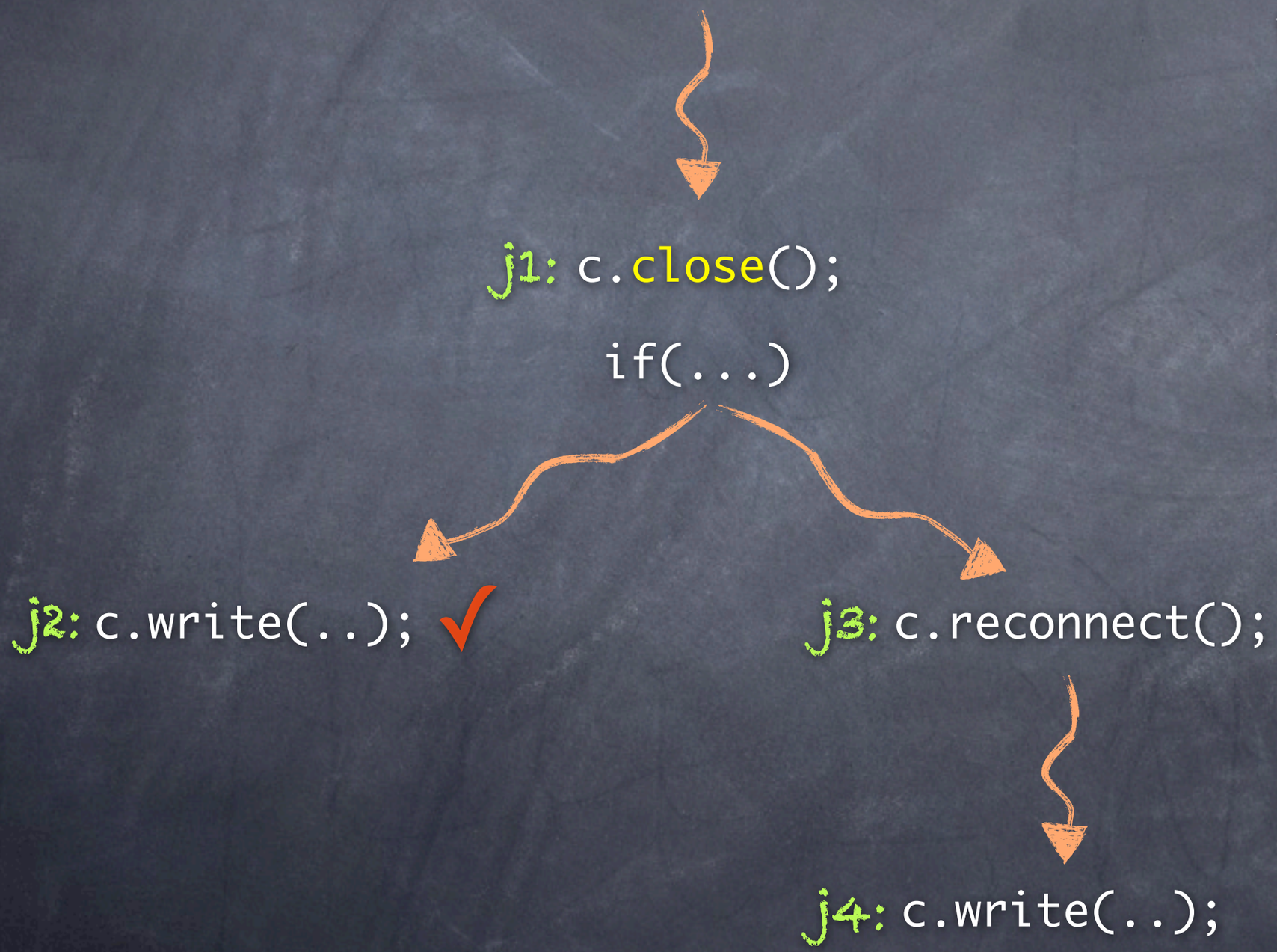


Semantics of Dependency State Machines



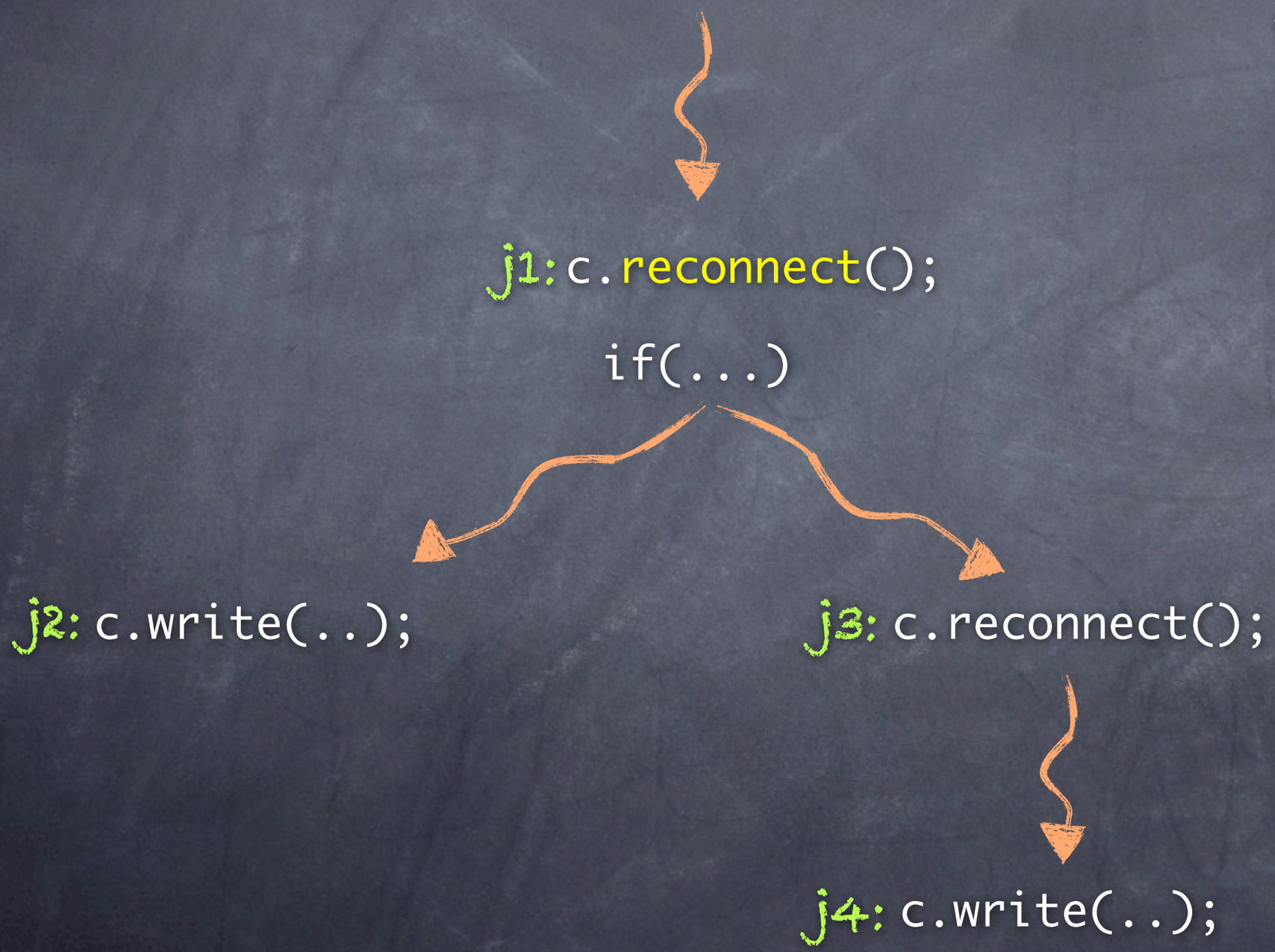
advice "close" must execute at j1 if there exists a path such that omitting "close" at j1 would change the joinpoints at which a DSM reaches an accepting state

Semantics of Dependency State Machines



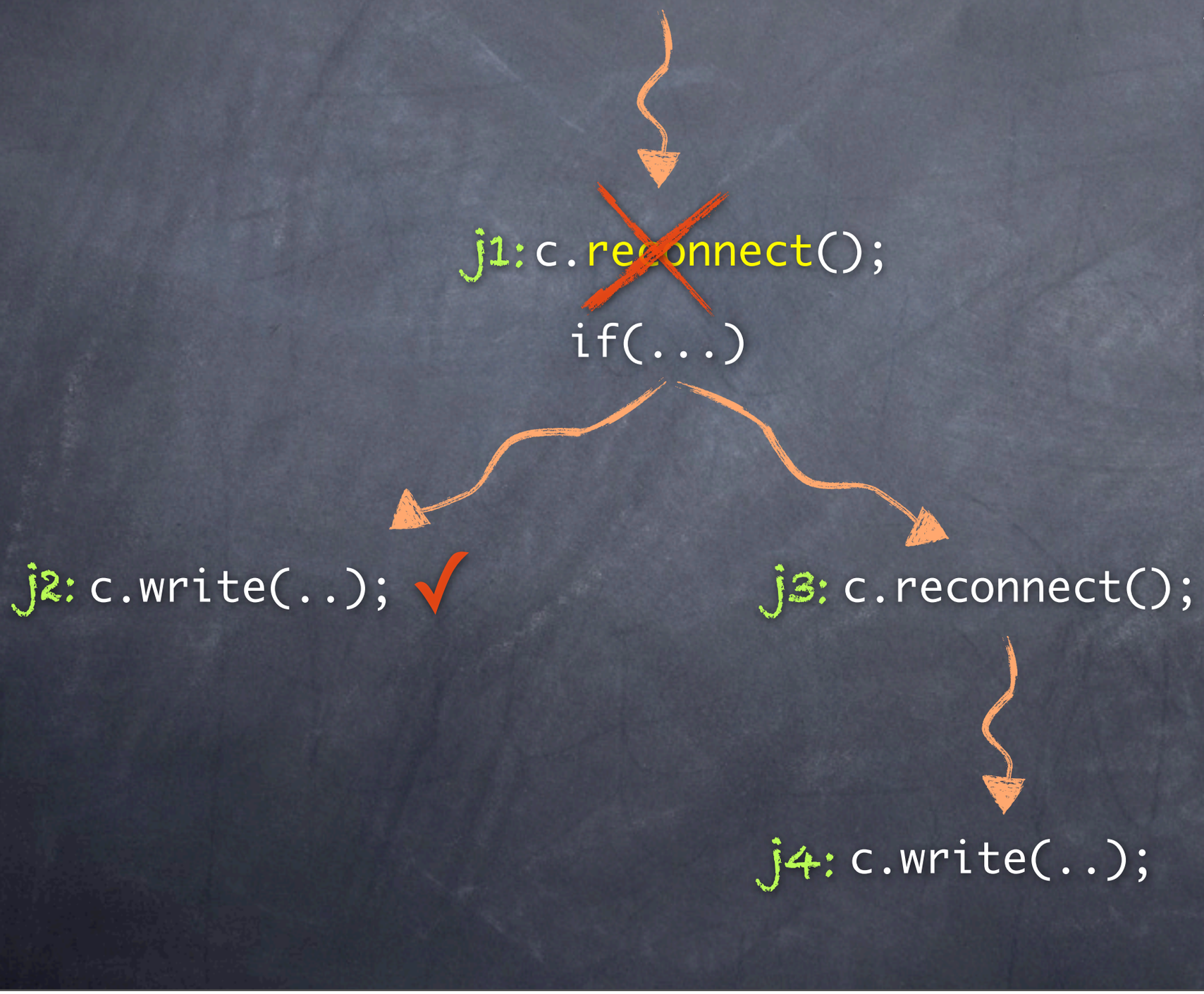
advice "close" must execute at j1 if there exists a path such that omitting "close" at j1 would change the joinpoints at which a DSM reaches an accepting state

Inverse case: match-preventing shadows



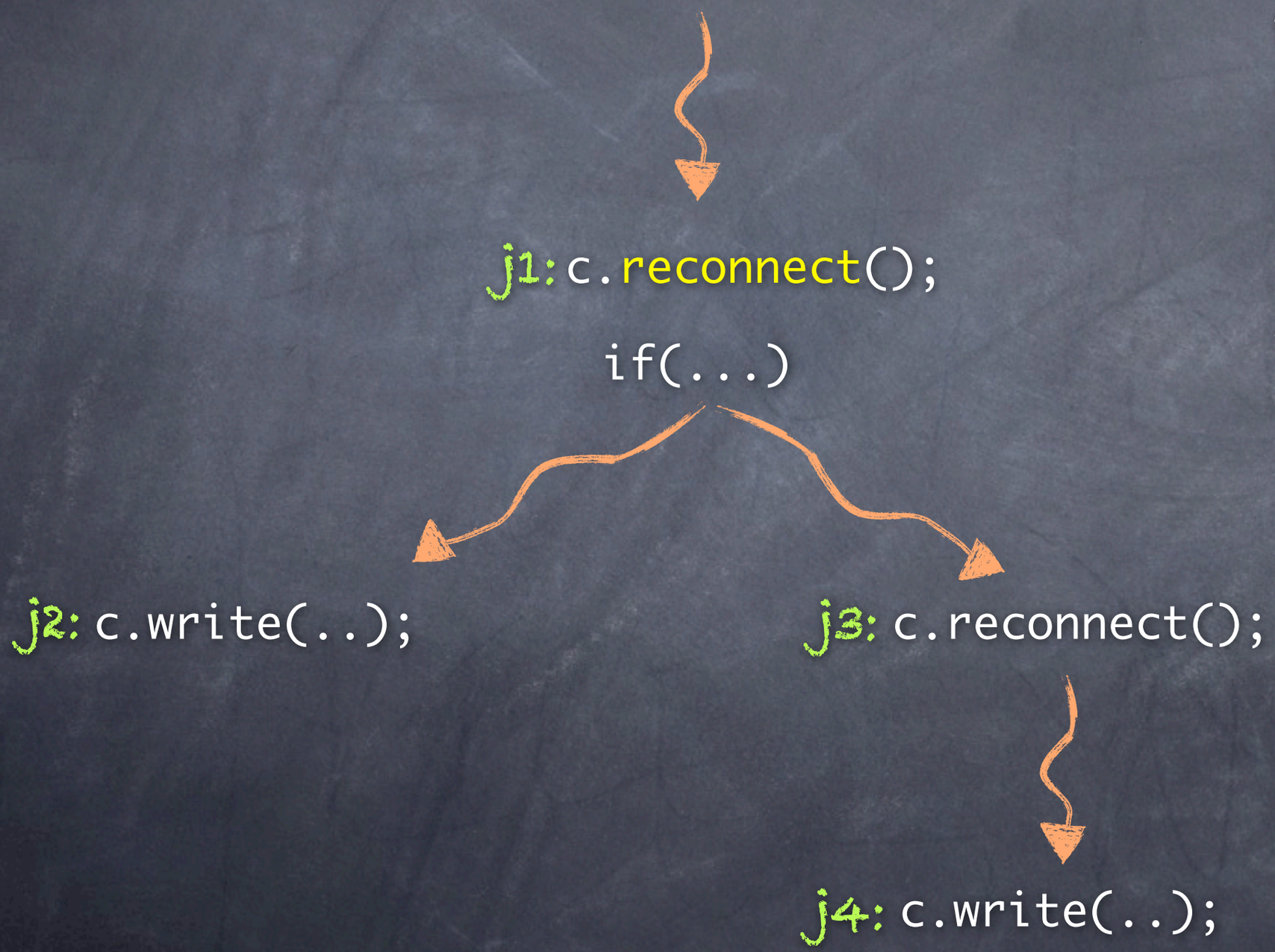
advice "reconnect" must execute at j1 if there exists a path such that omitting "reconnect" at j1 would change the joinpoints at which a DSM reaches an accepting state

Inverse case: match-preventing shadows



advice "reconnect" must execute at j1 if there exists a path such that omitting "reconnect" at j1 would change the joinpoints at which a DSM reaches an accepting state

Inverse case: match-preventing shadows



advice "reconnect" must execute at j1 if there exists a path such that omitting "reconnect" at j1 would change the joinpoints at which a DSM reaches an accepting state

Variable bindings matter

```
c.close();
```

```
j1: c.reconnect();

if(...)
```

j2:

```
j3: c.reconnect();
```

```
j4: c.write(..);
```

Variable bindings matter

```
c.close();
```

```
j1: c.reconnect();

if(...)
```

```
j2: c2.write(..);
```

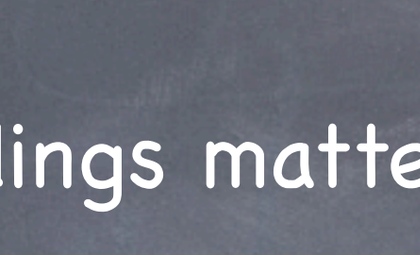
```
j3: c.reconnect();
```

```
j4: c.write(..);
```

Variable bindings matter

```
c.close();
```

```
j1: c.reconnect();
if(...)
```



```
j3: c.reconnect();
```

```
j4: c.write(..);
```

Variable bindings matter

```
c.close();
```

```
j1: c.reconnect();
```

```
if(...)
```

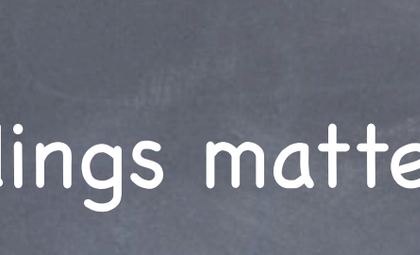
```
j3: c.reconnect();
```

```
j4: c.write(..);
```

Variable bindings matter

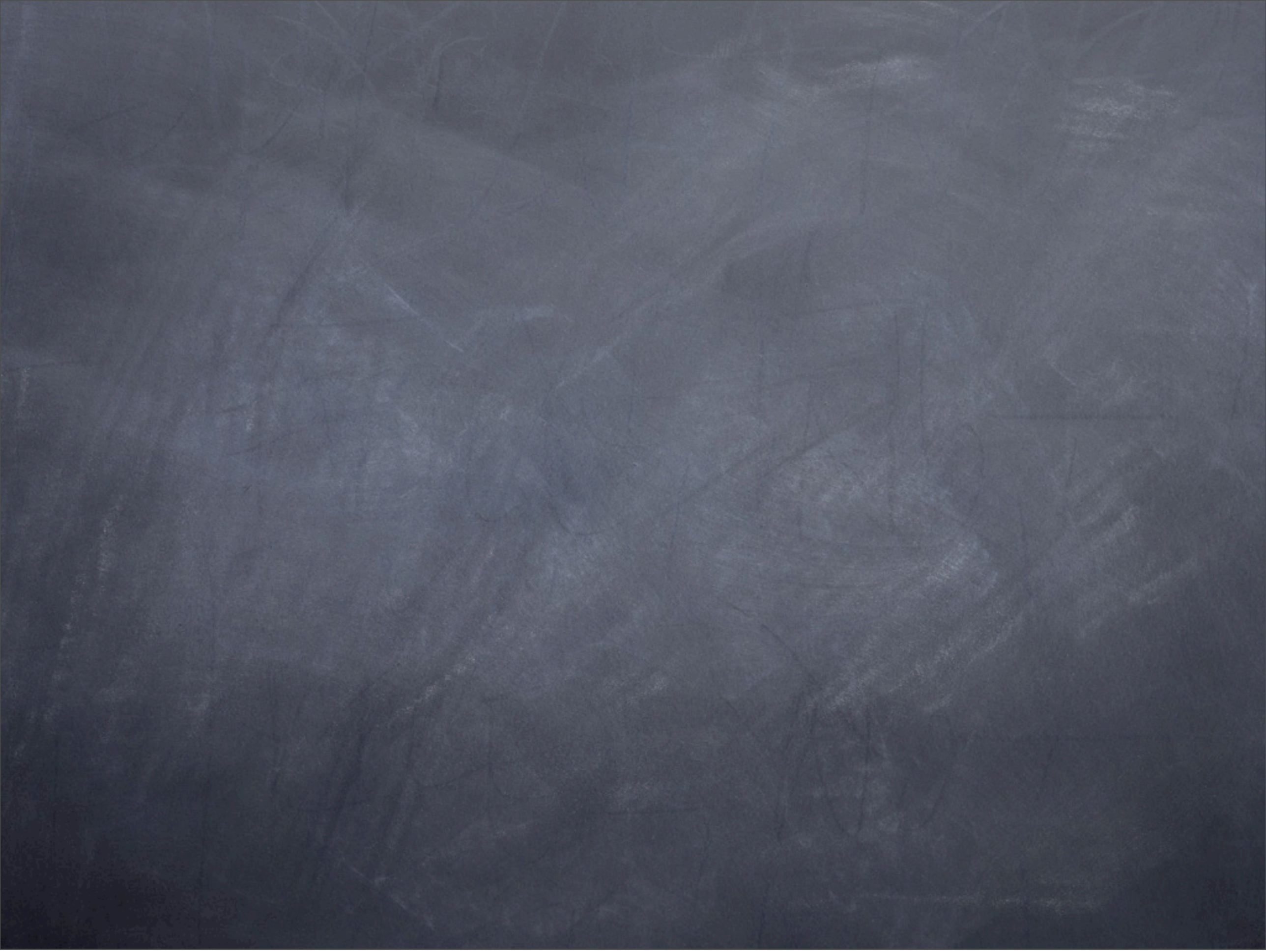
```
c.close();
```

```
j1: c.reconnect();
if(...)
```



```
j3: c.reconnect();
```

```
j4: c.write(..);
```



Specifying and Exploiting Advice-Execution Ordering using Dependency State Machines

Specifying and Exploiting Advice-Execution Ordering using Dependency State Machines

The Clara Framework

The Clara Framework

Compile-time
approximation
of runtime analyses

The Clara Framework

dependent after(): call(...)

The Clara Framework

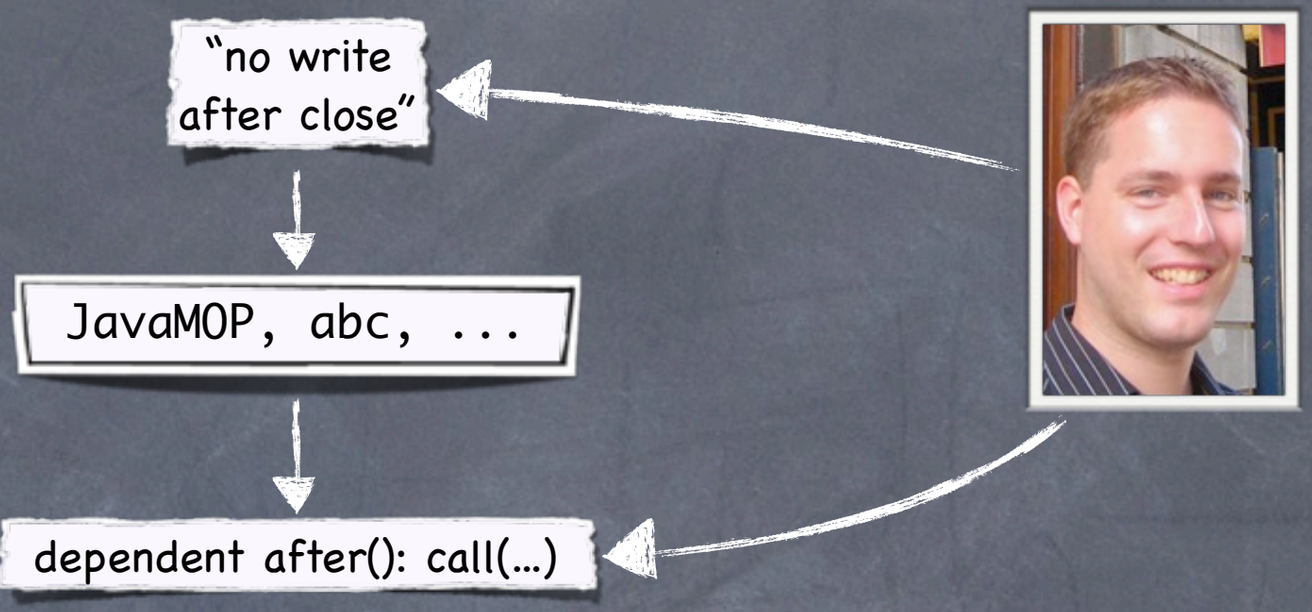
dependent after(): call(...)

The Clara Framework

"no write after close"

dependent after(): call(...)

The Clara Framework



The Clara Framework

"no write after close"

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

JavaMOP, abc, ...

dependent after(): call(...)

abc

compile & weave

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

The Clara Framework

"no write after close"

JavaMOP, abc, ...

dependent after(): call(...)

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

abc

compile & weave

Quick Check

Clara

Orphan-Shadows Analysis

Nop-Shadows Analysis

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

The Clara Framework

"no write after close"

JavaMOP, abc, ...

dependent after(): call(...)

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

abc

compile & weave

Clara

Quick Check

Orphan-Shadows Analysis

Nop-Shadows Analysis

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

The Clara Framework

"no write after close"

JavaMOP, abc, ...

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```

dependent after(): call(...)

abc

compile & weave

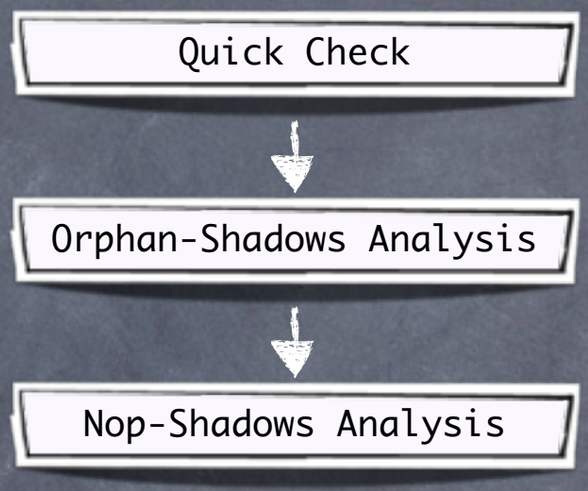
Clara

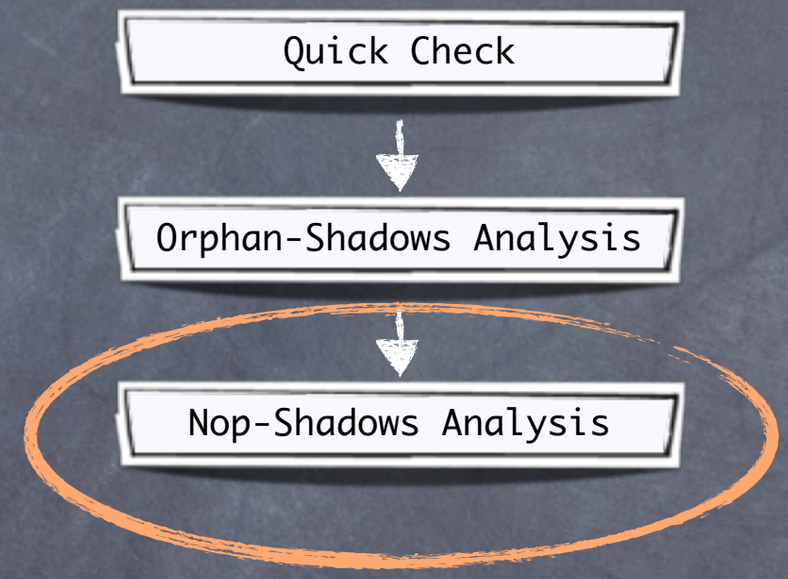
Quick Check

Orphan-Shadows Analysis

Nop-Shadows Analysis

```
public class ClaraTest {

    public static void main(String args[]) {
        Connection c1 = new Connection(args[0]);
        c1.close();
        c1.reconnect();
        c1.close();
        c1.close();
        c1.write(args[1]);
        c1.close();
        c1.reconnect();
        c1.write(args[1]);
    }

}
```


Quick Check
Orphan-Shadows Analysis
Nop-Shadows Analysis

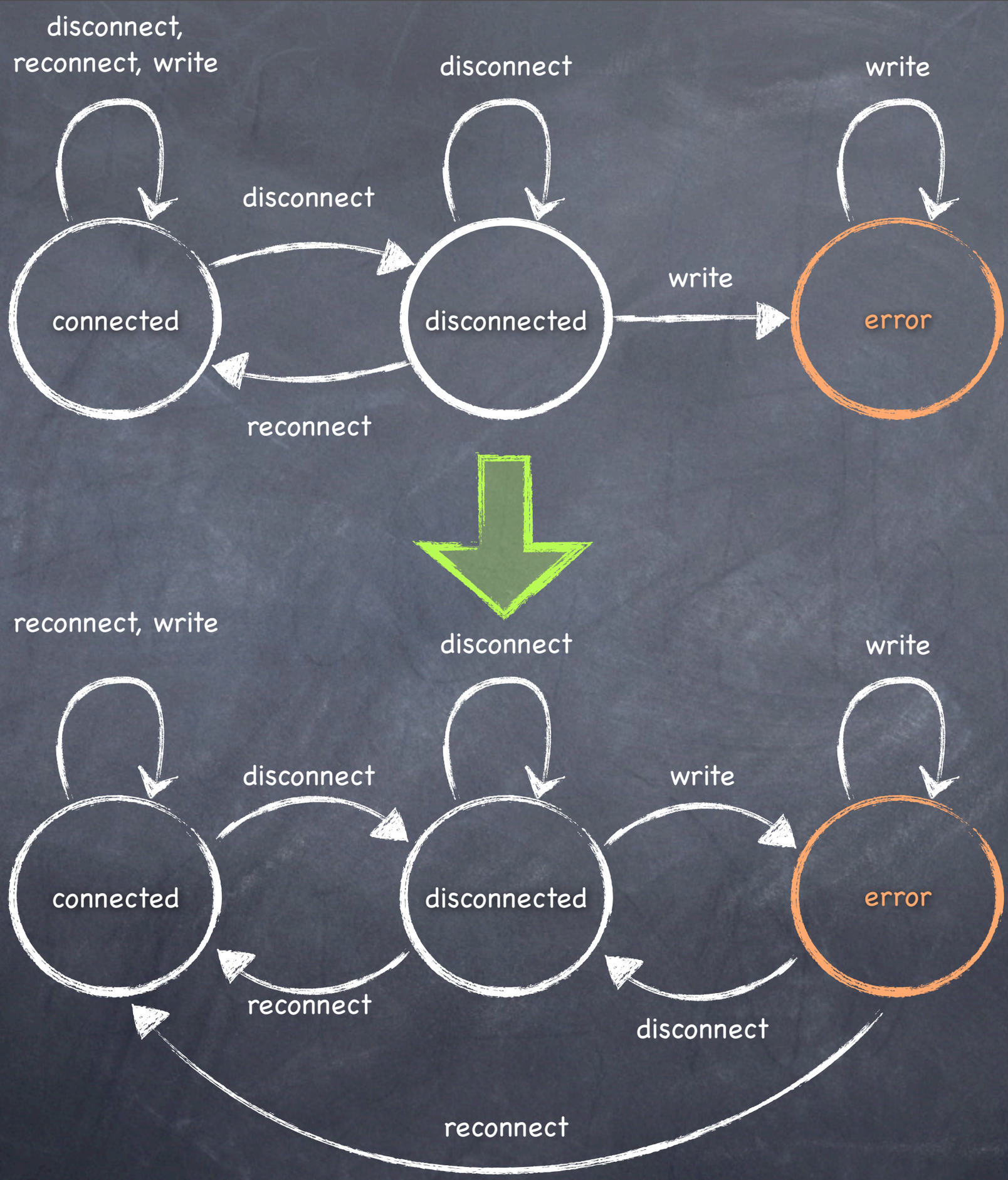

Quick Check
Orphan-Shadows Analysis
Nop-Shadows Analysis

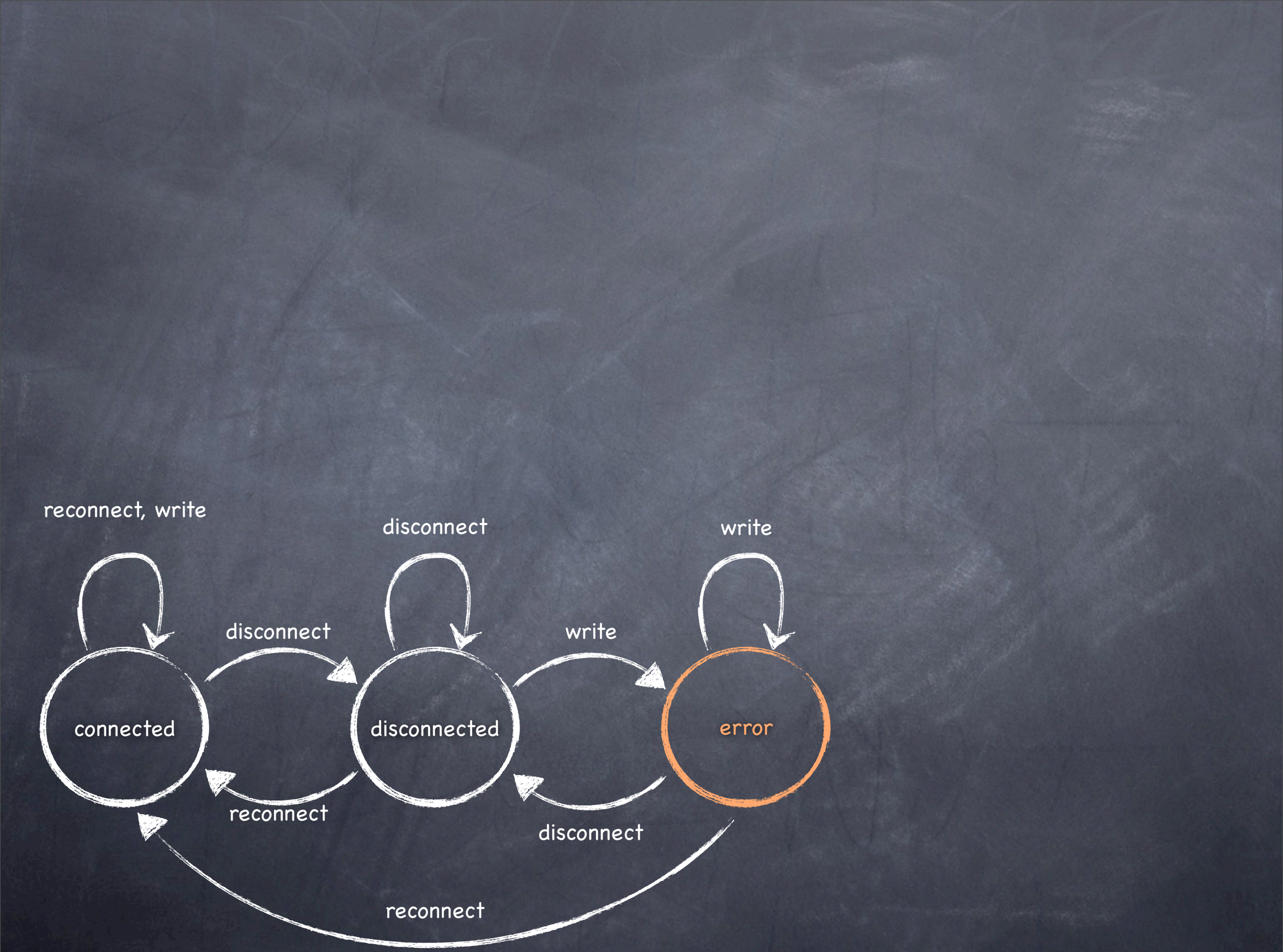
Nop-Shadows Analysis

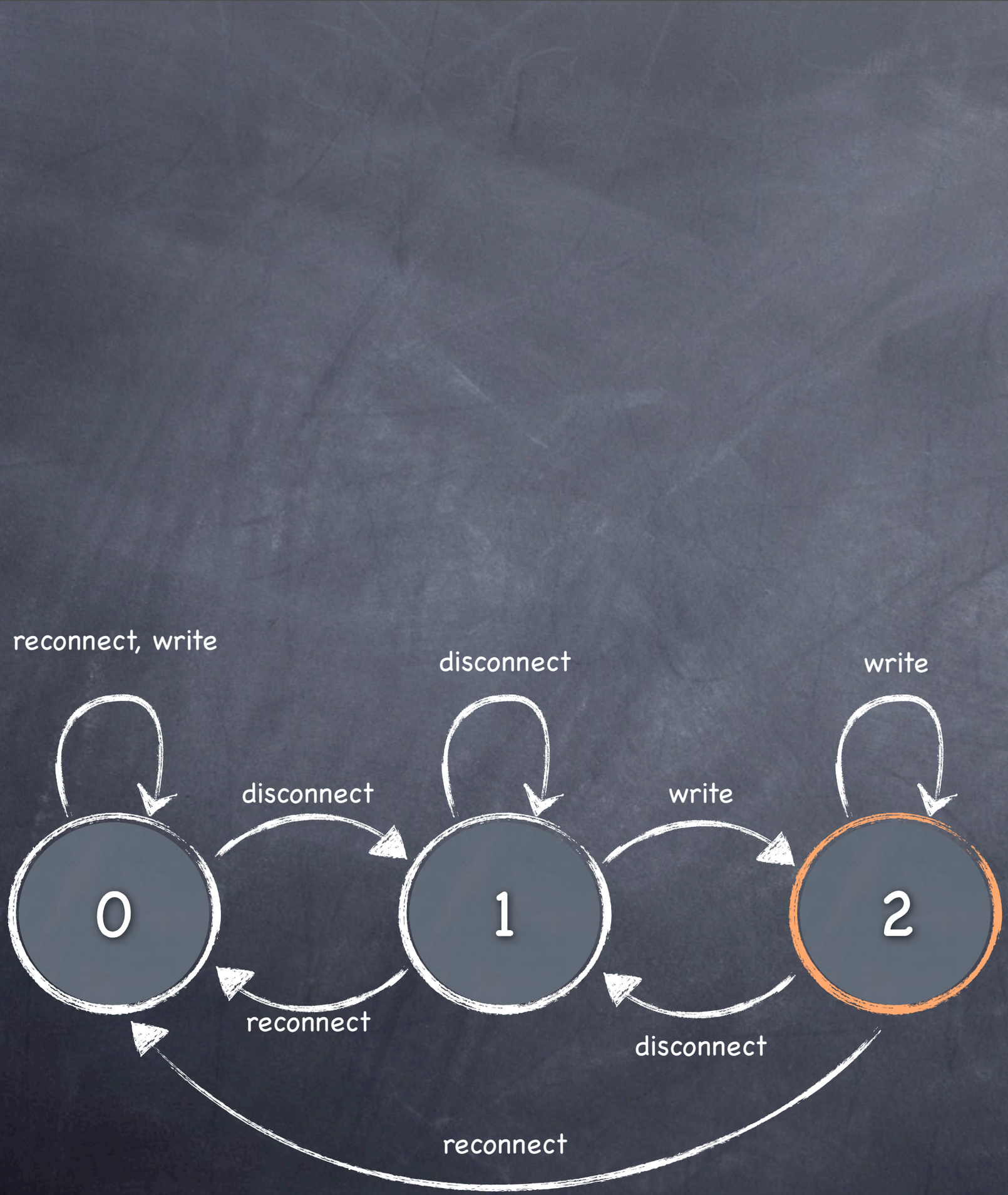
Idea:

For every joinpoint shadow s:

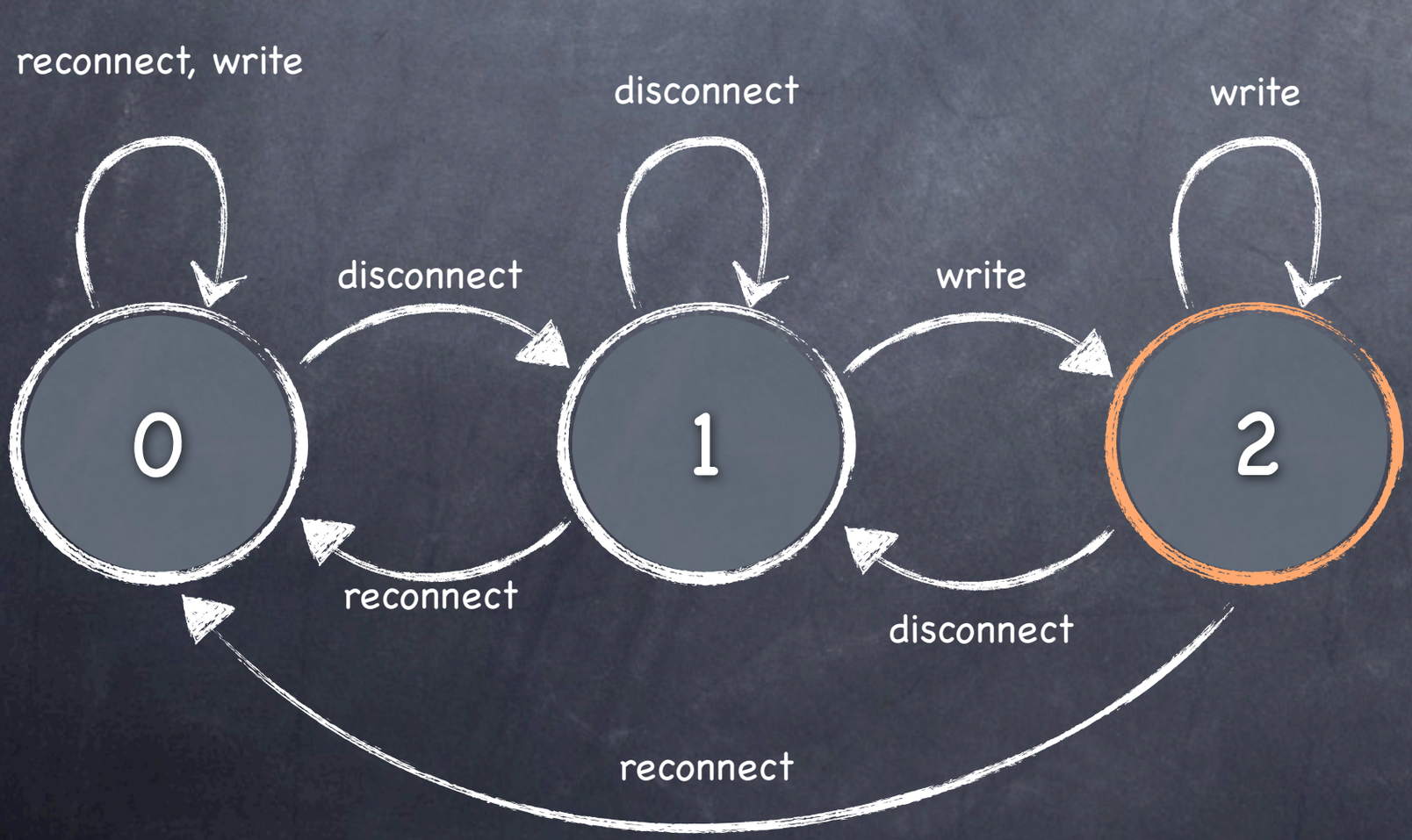
- Identify states that are "equivalent" at s.
- If s may transition only between equivalent states then disable s.


disconnect,
reconnect, write
disconnect
write
disconnect
connected
disconnected
write
error
reconnect
reconnect, write
disconnect
write
disconnect
connected
disconnected
error
reconnect
disconnect
reconnect


reconnect, write
disconnect
write
disconnect
write
connected
disconnected
error
reconnect
disconnect
reconnect

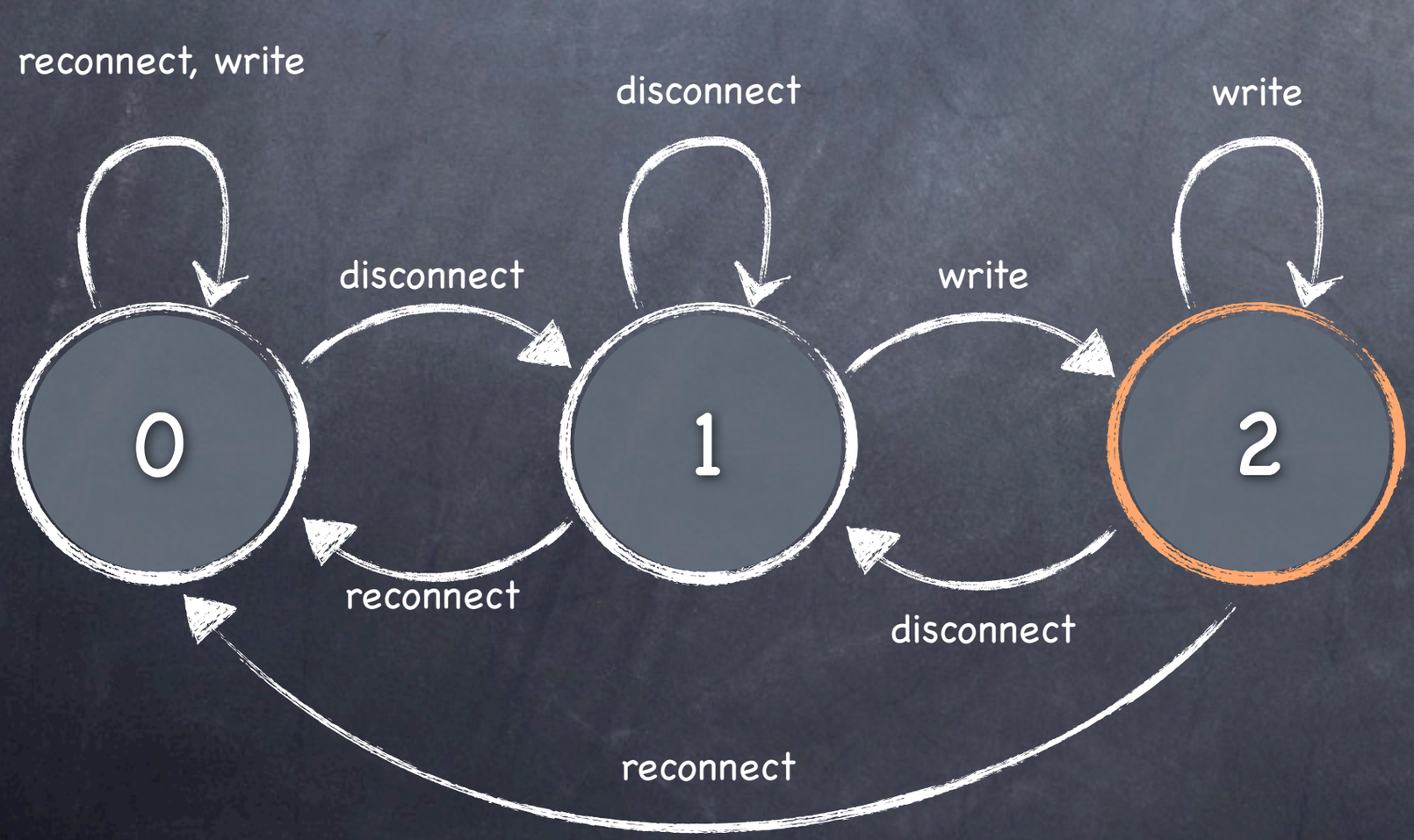

reconnect, write
disconnect
write
disconnect
write
0
1
2
reconnect
disconnect
reconnect

```
c1.close();                0
c1.reconnect();            1
c1.close();                0
c1.close();                1
c1.write(..);              1
c1.close();                2
c1.reconnect();            1
c1.write(..);              0
                           0
```



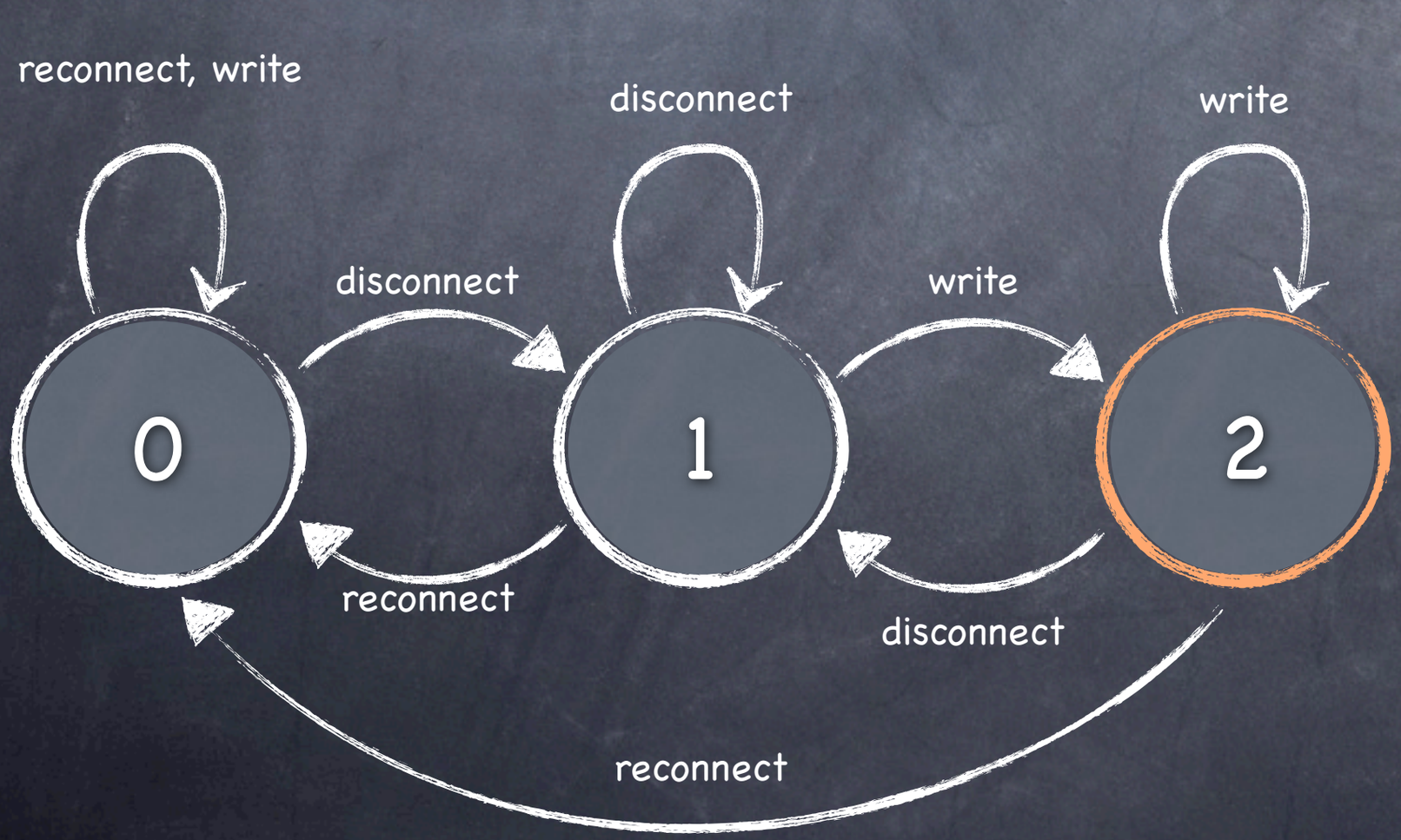
```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0 1 0 1 1 2 1 0 0



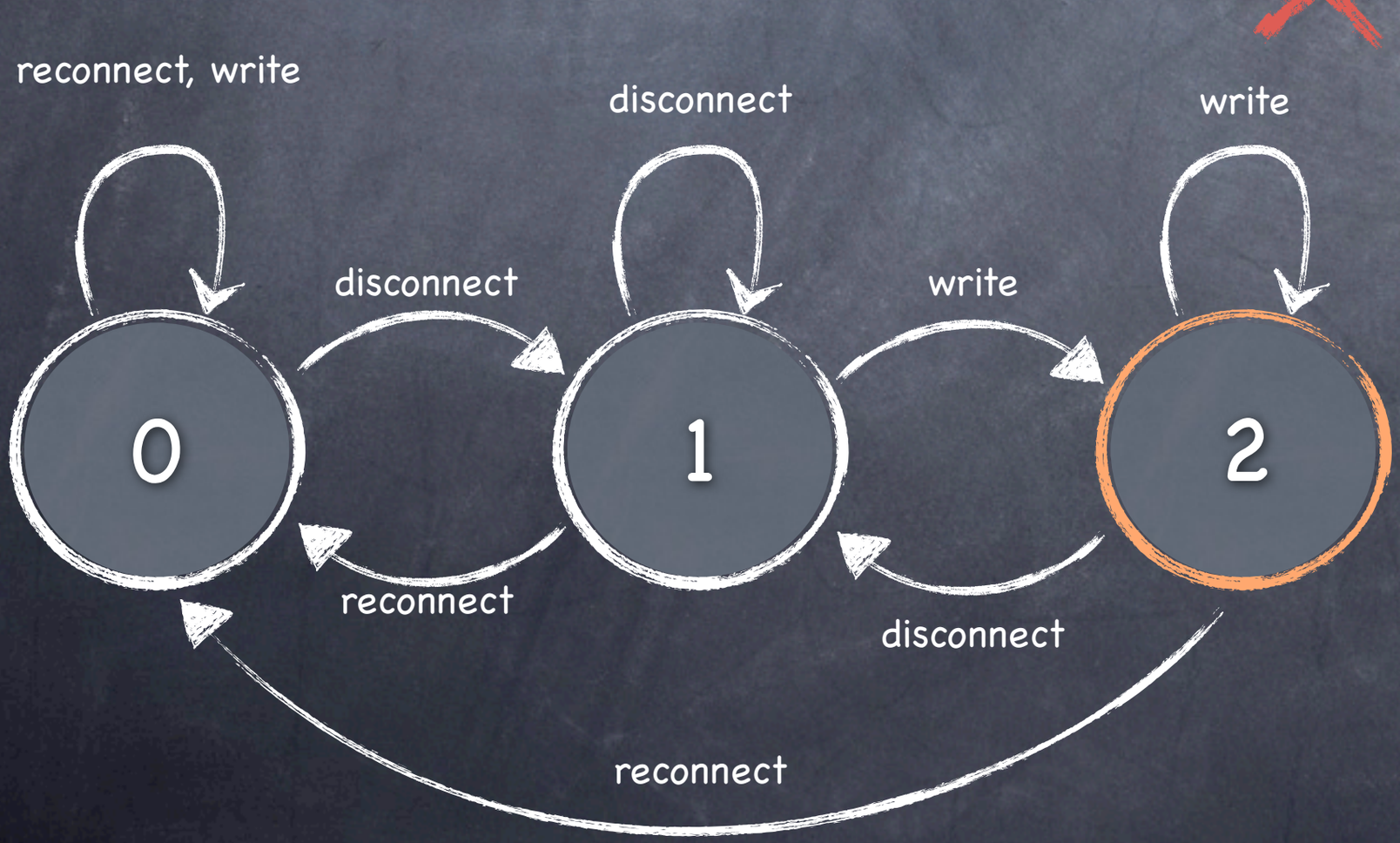
```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0 1 0 1 1 2 1 0 0



```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0 1 0 1 1 2 1 0 0



```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0 1 0 1 1 2 1 0 0

reconnect, write

disconnect

write

disconnect

write

0

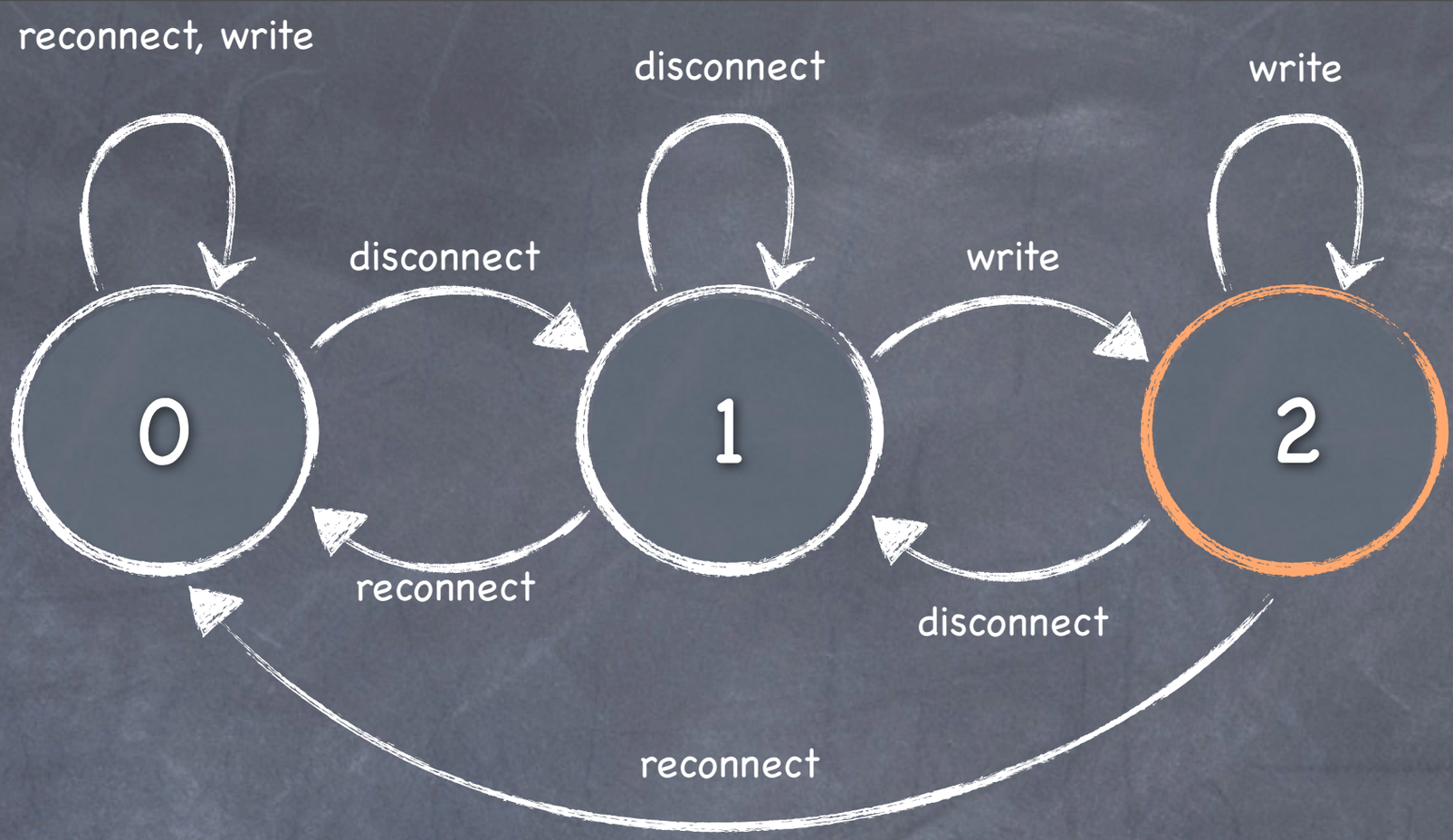
1

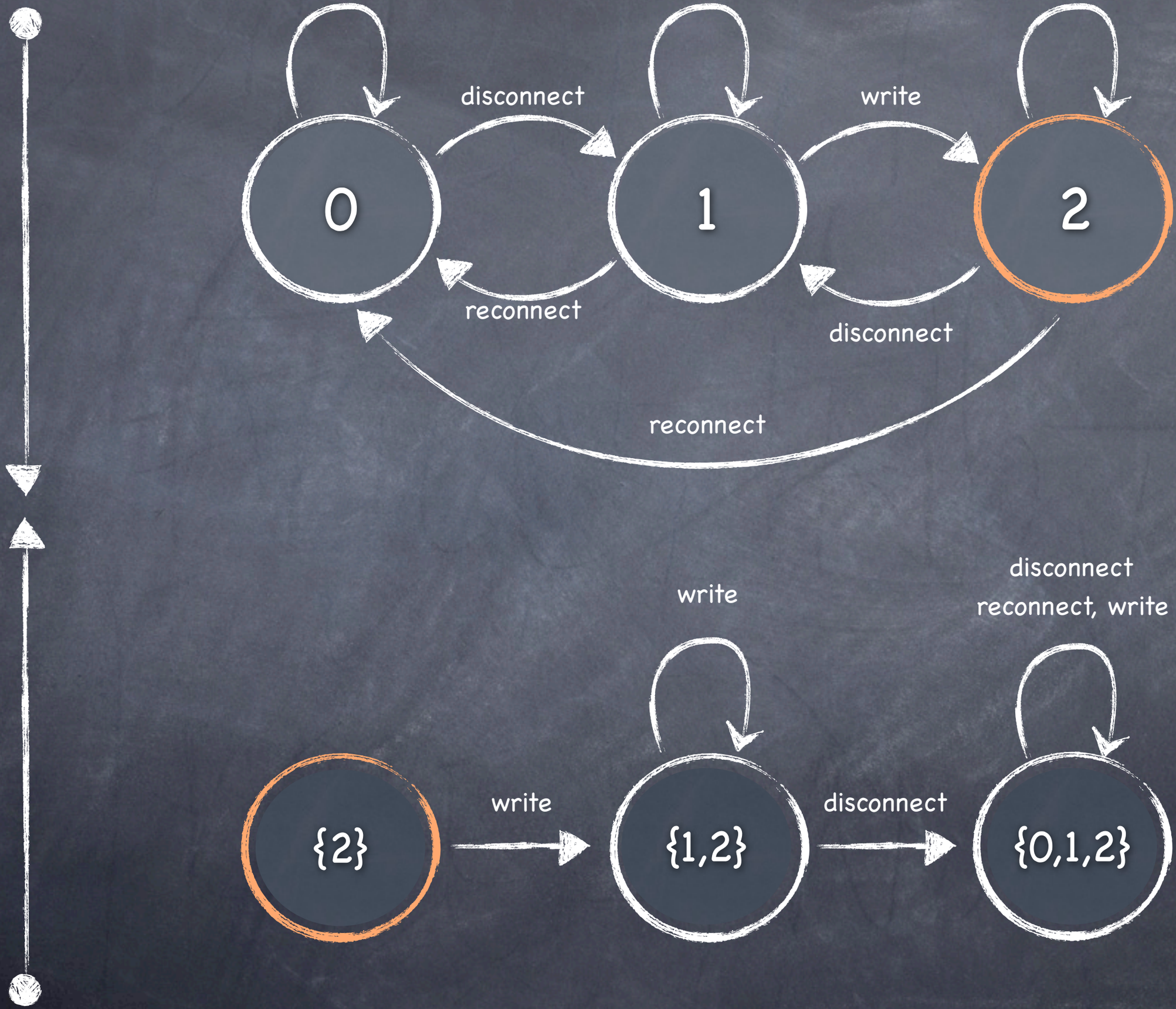
2

reconnect

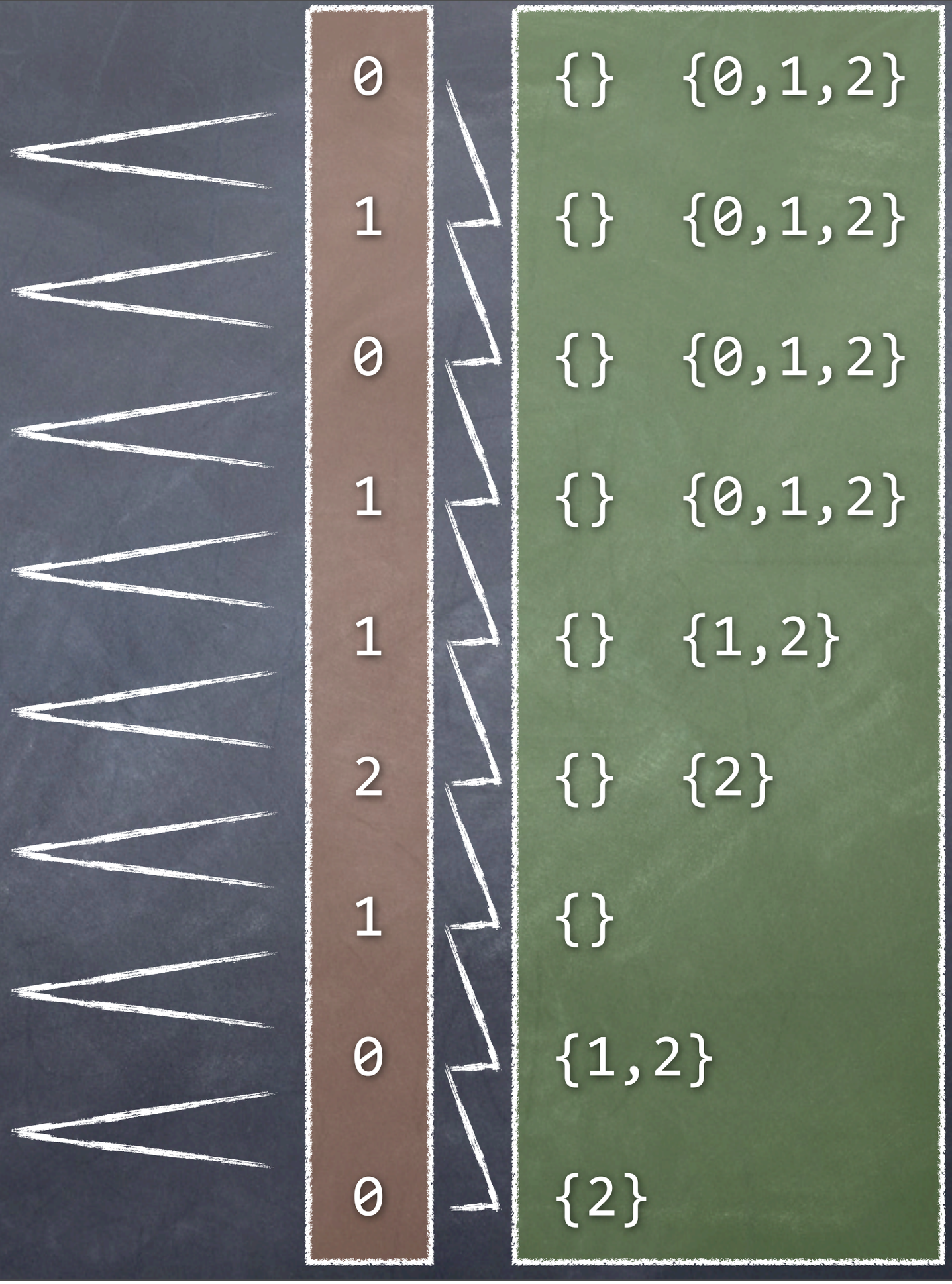
disconnect

reconnect


reconnect, write
disconnect
write
disconnect
write
0
1
2
reconnect
disconnect
reconnect


reconnect, write
disconnect
write
disconnect
write
0
1
2
reconnect
disconnect
reconnect
write
disconnect
reconnect, write
{2}
write
{1,2}
disconnect
{0,1,2}

```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```



```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{}	{0,1,2}
1	{}	{0,1,2}
0	{}	{0,1,2}
1	{}	{0,1,2}
1	{}	{1,2}
2	{}	{2}
1	{}	
0	{1,2}	
0	{2}	

```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{}	{0,1,2}
1	{}	{0,1,2}
0	{}	{0,1,2}
1	{}	{0,1,2}
1	{}	{1,2}
2	{}	{2}
1	{}	
0	{1,2}	
0	{2}	

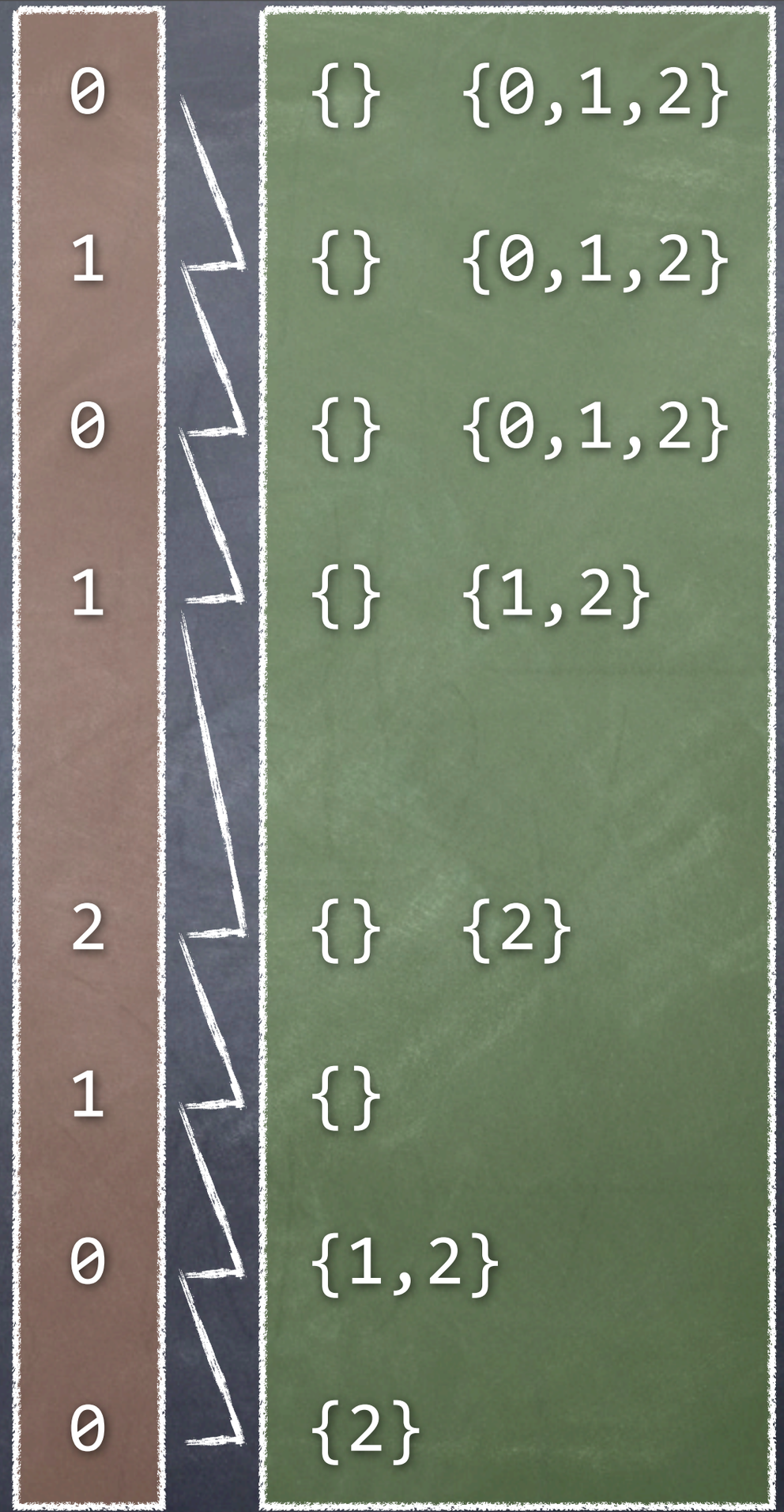
```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{}	{0,1,2}
1	{}	{0,1,2}
0	{}	{0,1,2}
1	{}	{0,1,2}
1	{}	{1,2}
2	{}	{2}
1	{}	
0	{1,2}	
0	{2}	

```
c1.close();
c1.reconnect();
c1.close();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{}	{0,1,2}
1	{}	{0,1,2}
0	{}	{0,1,2}
1	{}	{0,1,2}
1	{}	{1,2}
2	{}	{2}
1	{}	
0	{1,2}	
0	{2}	

```
c1.close();
c1.reconnect();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```



```
c1.close();
c1.reconnect();
c1.close();

c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

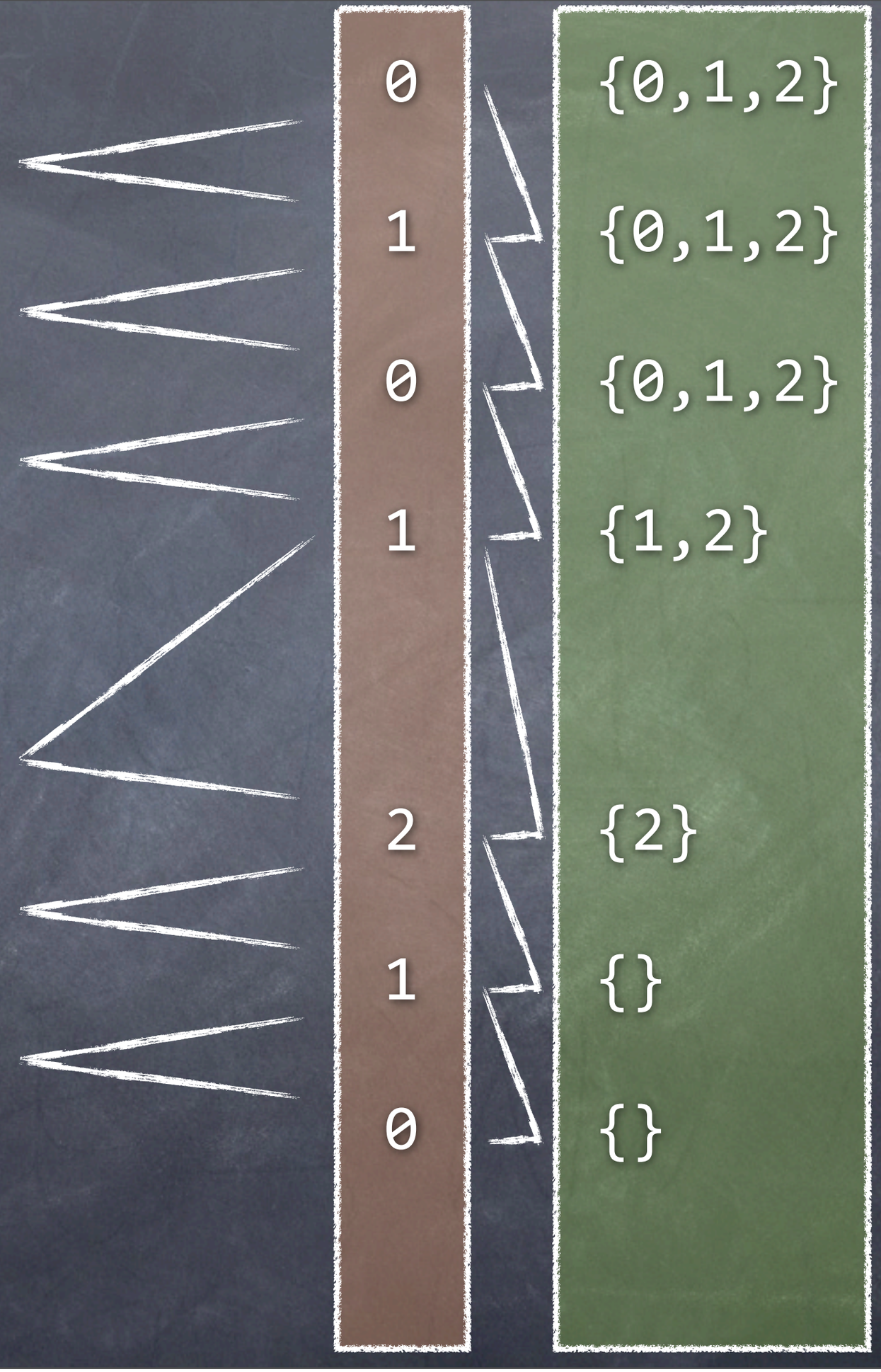
```
c1.close();
c1.reconnect();
c1.close();

c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{} {0,1,2}
1	{} {0,1,2}
0	{} {0,1,2}
1	{} {1,2}
2	{} {2}
1	{}
0	{1,2}
0	{2}

```
c1.close();
c1.reconnect();
c1.close();
c1.write(..);
c1.close();
c1.reconnect();
c1.write(..);
```

0	{}	{0,1,2}
1	{}	{0,1,2}
0	{}	{0,1,2}
1	{}	{1,2}
2	{}	{2}
1	{}	
0	{1,2}	
0	{2}	

```
c1.close();
c1.reconnect();
c1.close();

c1.write(..);
c1.close();
c1.reconnect();
```



```
c1.close();
c1.reconnect();
c1.close();

c1.write(..);
c1.close();
c1.reconnect();
```

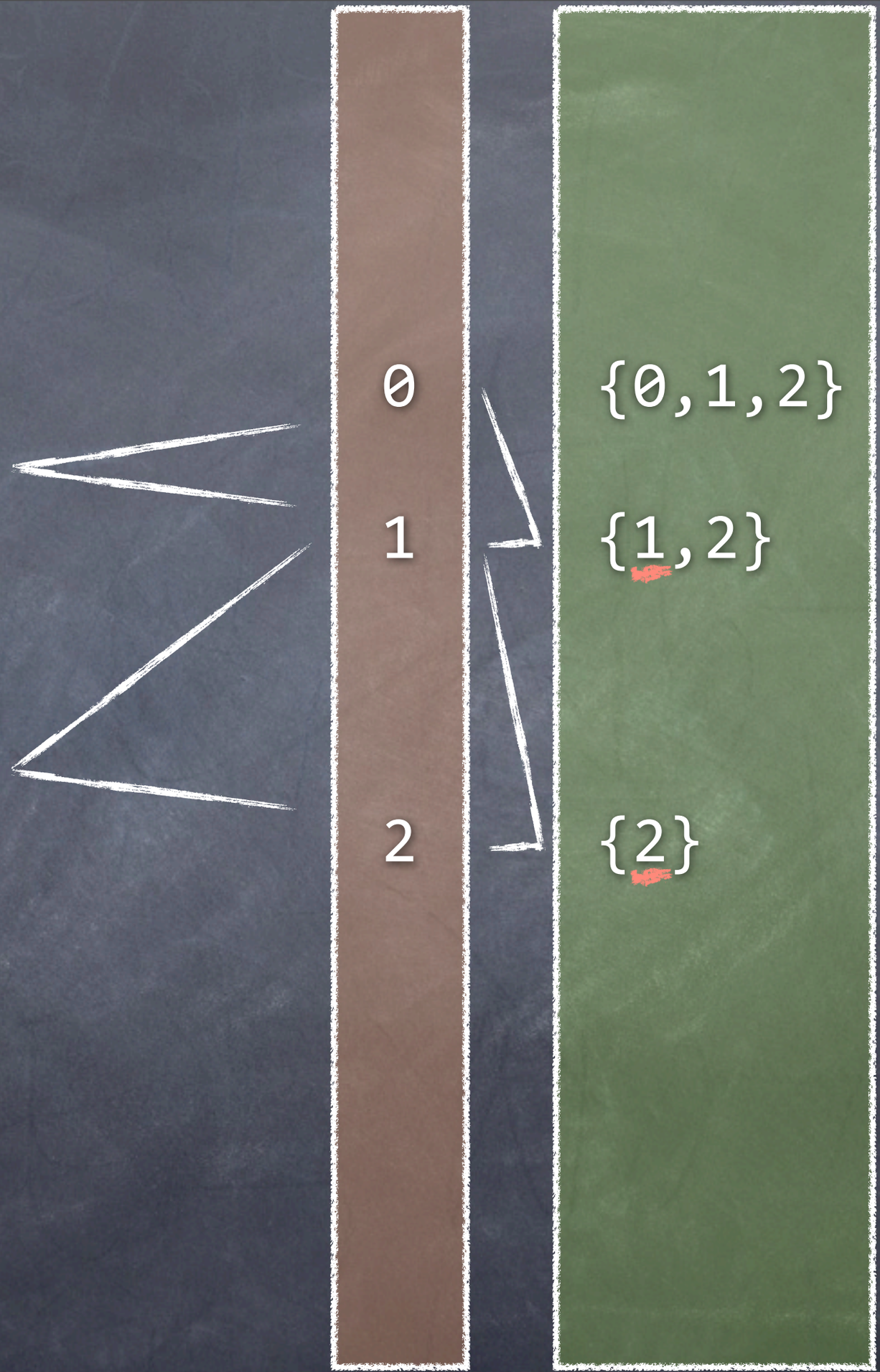
0	{0,1,2}
1	{0,1,2}
0	{0,1,2}
1	{1,2}
2	{2}
1	{}
0	{}

```
c1.close();
c1.reconnect();
c1.close();

c1.write(..);
c1.close();
c1.reconnect();
```

0	{0,1,2}
1	{0,1,2}
0	{0,1,2}
1	{1,2}
2	{2}
1	{}
0	{}

```
c1.close();
```

```
c1.write(..);
```



Tested Properties

ASyncContainsAll	FailSafeIterMap
ASyncIterC	HasNextElem
ASyncIterM	HasNext
FailSafeEnum	LeakingSync
FailSafeEnumHT	Reader
FailSafeIter	Writer

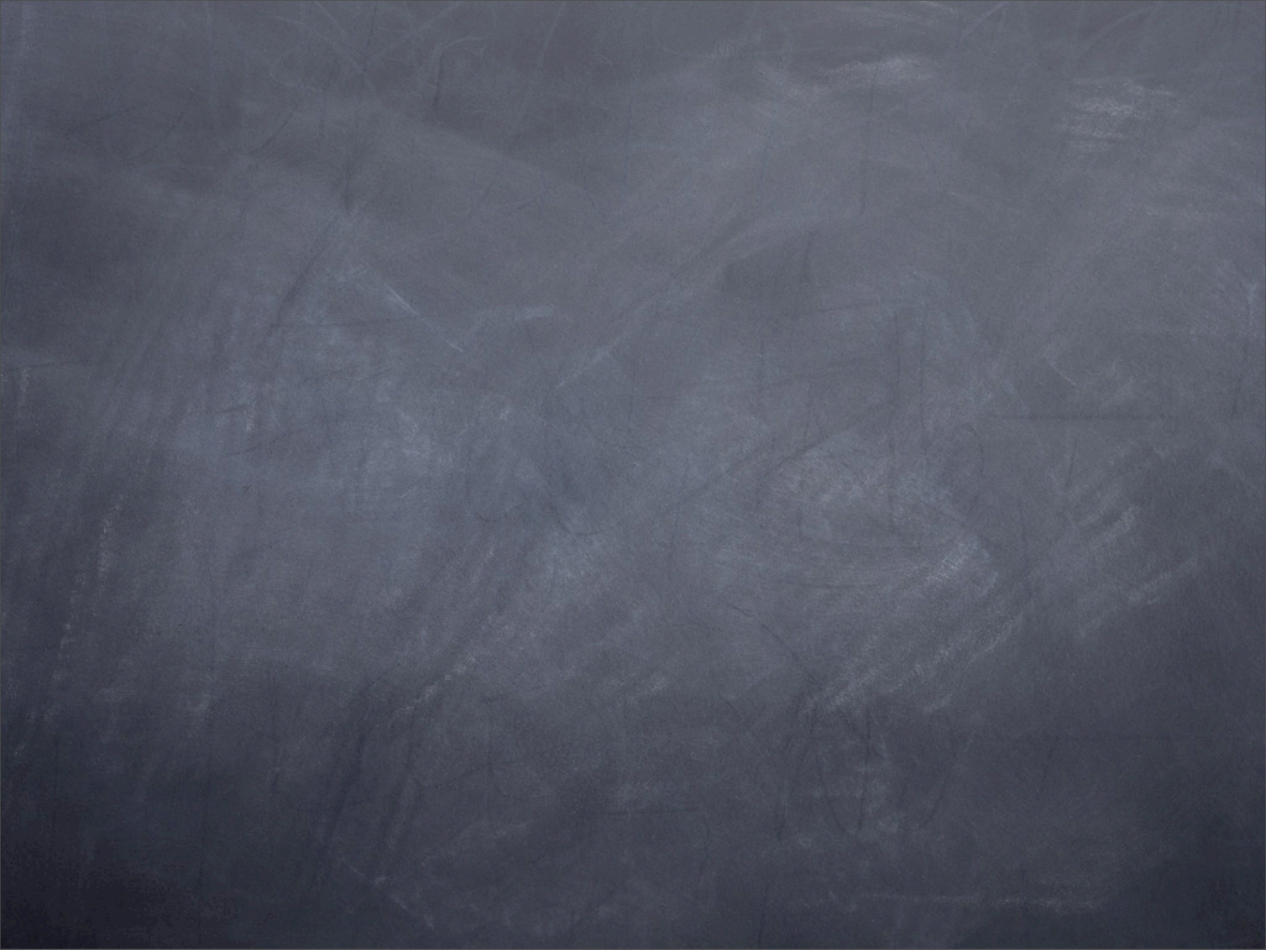
Benchmark programs

antlr	jython
bloat	luindex
chart	lusearch
fop	pmd
hsqldb	xalan

(whole DaCapo benchmark suite, except eclipse)

Overall success

	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751



Related Work

Static analysis of Aspects:

- Dependent Advice (Bodden, Chen & Rosu, AOSD '09)

Static analysis of Runtime Monitors:

- Tracematches (Naeem & Lhotak, OOPSLA '08)
- Tracematches (Bodden et al., ECOOP '07 and FSE '08)
- General aspects with DSMs (Bodden, ICSE '10)

Related Work

Aspect-internal optimizations:

- Indexing and Leak Elimination for Tracematches (Avgustinov, Tibble & de Moor, OOPSLA '07)
- Formalism-independent Indexing (Chen & Rosu, OOPSLA '07)
- PQL (Goldsmith, O'Callahan & Eiken, OOPSLA '05)

Related Work

Model Checking for aspects:

- Goldman & Katz, TACAS '07

Related Work

Generating history-based aspects:

- Association Aspects (Sakurai et al., AOSD '04)
- Tracematches (Avgustinov et al., AOSD '05)
- MOP (Chen & Rosu, OOPSLA '07)
- Relational Aspects (Bodden et al., AOSD '08)
- M2Aspects (Krüger et al., SCESM '06)
- S2A (Maoz & Harel, FSE '06)
- J-LO (Bodden et al., RV '07)
- Java-STAIRS Aspects (Oldevik, Haugen, AOSD '09)

... what will we see this year?

Conclusion

	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751

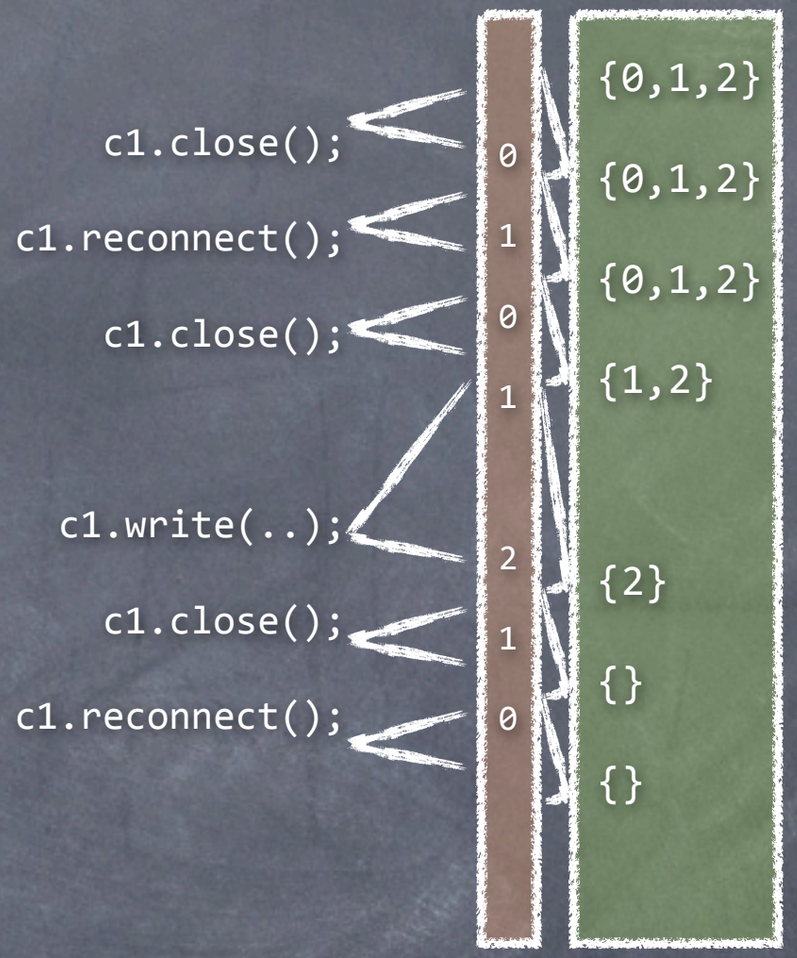
Conclusion

```
dependency{
    disconnect, write, reconnect;
    initial  connected: disconnect -> connected,
                        write -> connected,
                        reconnect -> connected,
                        disconnect -> disconnected;
                    disconnect: disconnect -> disconnected,
                        write -> error;
    final       error: write -> error;
}
```

	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751

Conclusion

```
dependency{
    disconnect, write, reconnect;
    initial  connected: disconnect -> connected,
                        write -> connected,
                        reconnect -> connected,
                        disconnect -> disconnected;
                    disconnect: disconnect -> disconnected,
                        write -> error;
    final       error: write -> error;
}
```

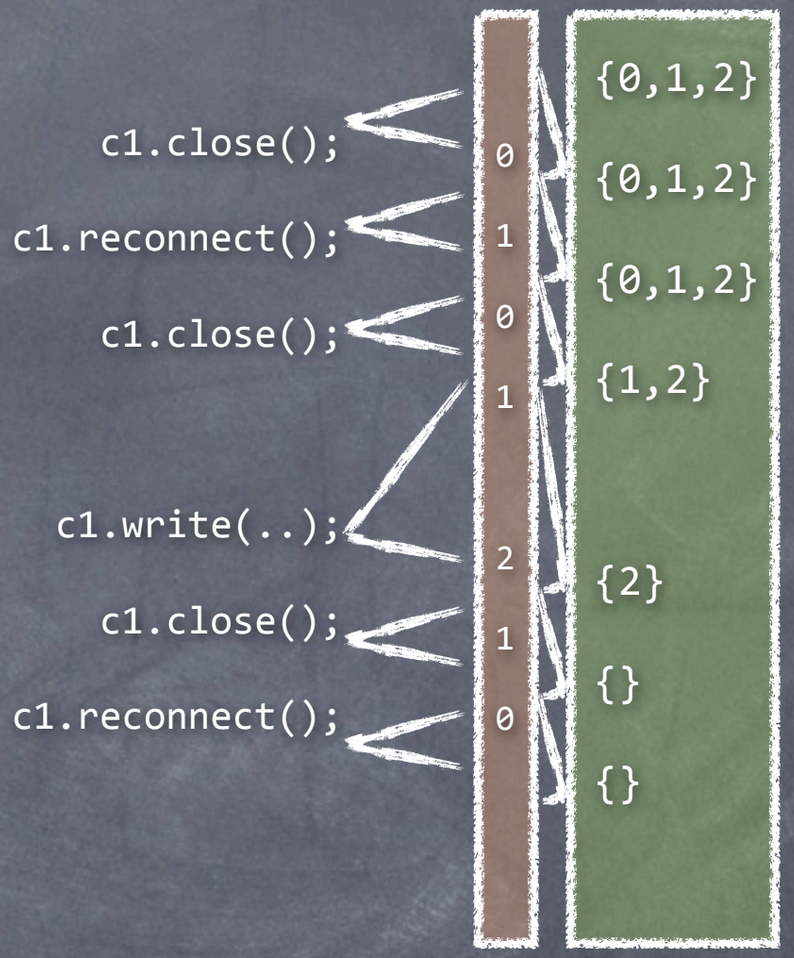


	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751

Conclusion

```
dependency{
    disconnect, write, reconnect;
    initial  connected: disconnect -> connected,
                        write -> connected,
                        reconnect -> connected,
                        disconnect -> disconnected;
                    disconnect: disconnect -> disconnected,
                        write -> error;
    final       error: write -> error;
}
```

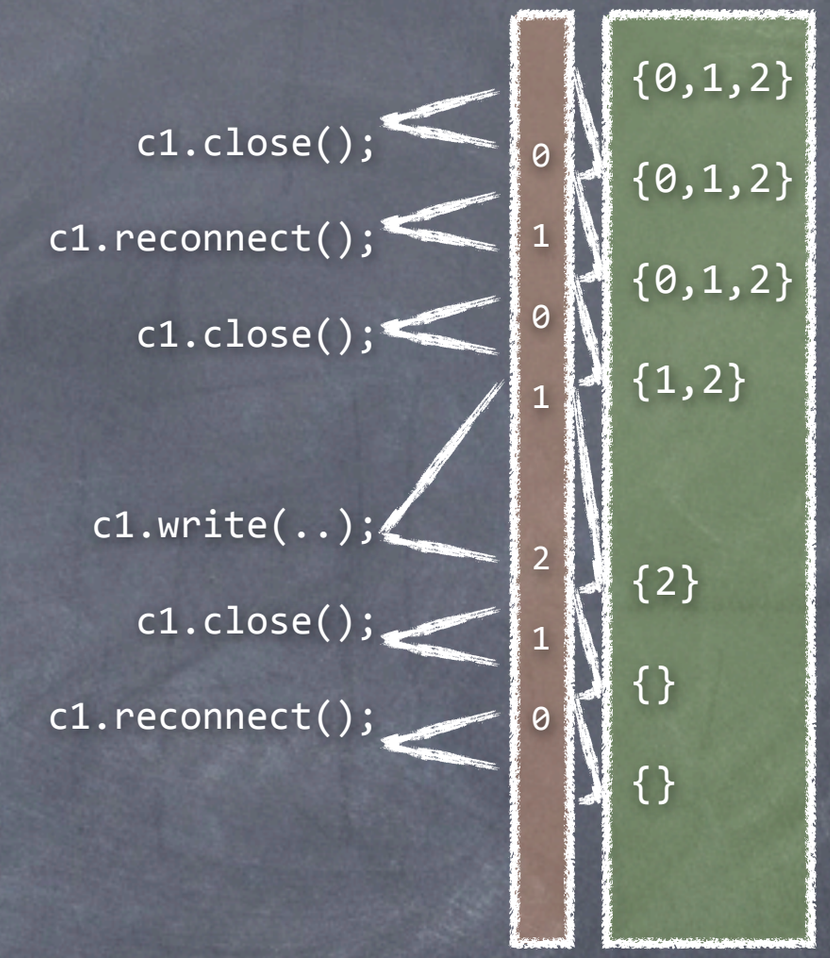
www.bodden.de/clara/



	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751

Conclusion

```
dependency{
    disconnect, write, reconnect;
    initial  connected: disconnect -> connected,
                        write -> connected,
                        reconnect -> connected,
                        disconnect -> disconnected;
                    disconnect: disconnect -> disconnected,
                        write -> error;
    final       error: write -> error;
}
```



www.bodden.de/clara/

	antlr	bloat	chart	fop	hsqldb	jython	luindex	lusearch	pmd	xalan
ASyncContainsAll		0/71	0/6			0/31	0/18	0/18	0/10	
ASyncIterC		0/1621	0/498	0/146	0/33	0/128	0/149	0/149	0/671	
ASyncIterM		0/1684	0/507	0/176	0/39	0/138	0/152	0/152	0/718	
FailSafeEnum	0/76	0/3	0/1	6/18	0/120	44/110	0/61	0/61	0/21	0/222
FailSafeEnumHT	26/133	0/102	0/44	0/205	3/114	61/153	0/37	0/37	0/100	0/319
FailSafeIter	0/23	830/1394	149/510	0/288	0/112	112/253	0/217	16/217	287/546	0/158
FailSafeIterMap	0/130	444/1180	49/374	OOME	0/252	133/250	0/136	0/136	204/583	0/540
HasNextElem	0/117	0/4		0/12	0/53	34/64	0/22	0/22	0/11	1/63
HasNext		452/849	48/248	0/72	0/16	24/63	0/74	0/74	184/346	
LeakingSync	0/170	0/1994	0/920	0/2347	0/528	0/1082	0/629	0/629	0/986	0/1005
Reader	0/50	0/7	0/65	0/102	3/1216	4/139	0/226	0/226	0/102	0/106
Writer	35/171	15/563	0/70	0/429	10/1378	0/462	0/146	0/146	0/62	0/751

What else can we use DSMs for?