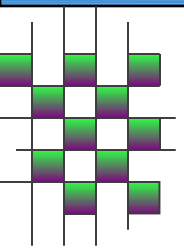


Object Oriented Testing

- Behavior / functional Testing (BBT) is same as conventional Testing.

- Control flow diagram is no longer applicable ... (statement coverage, branch coverage, data flow coverage)

- Structural Testing (WBT) is not same as conventional testing.

Two type structural testing

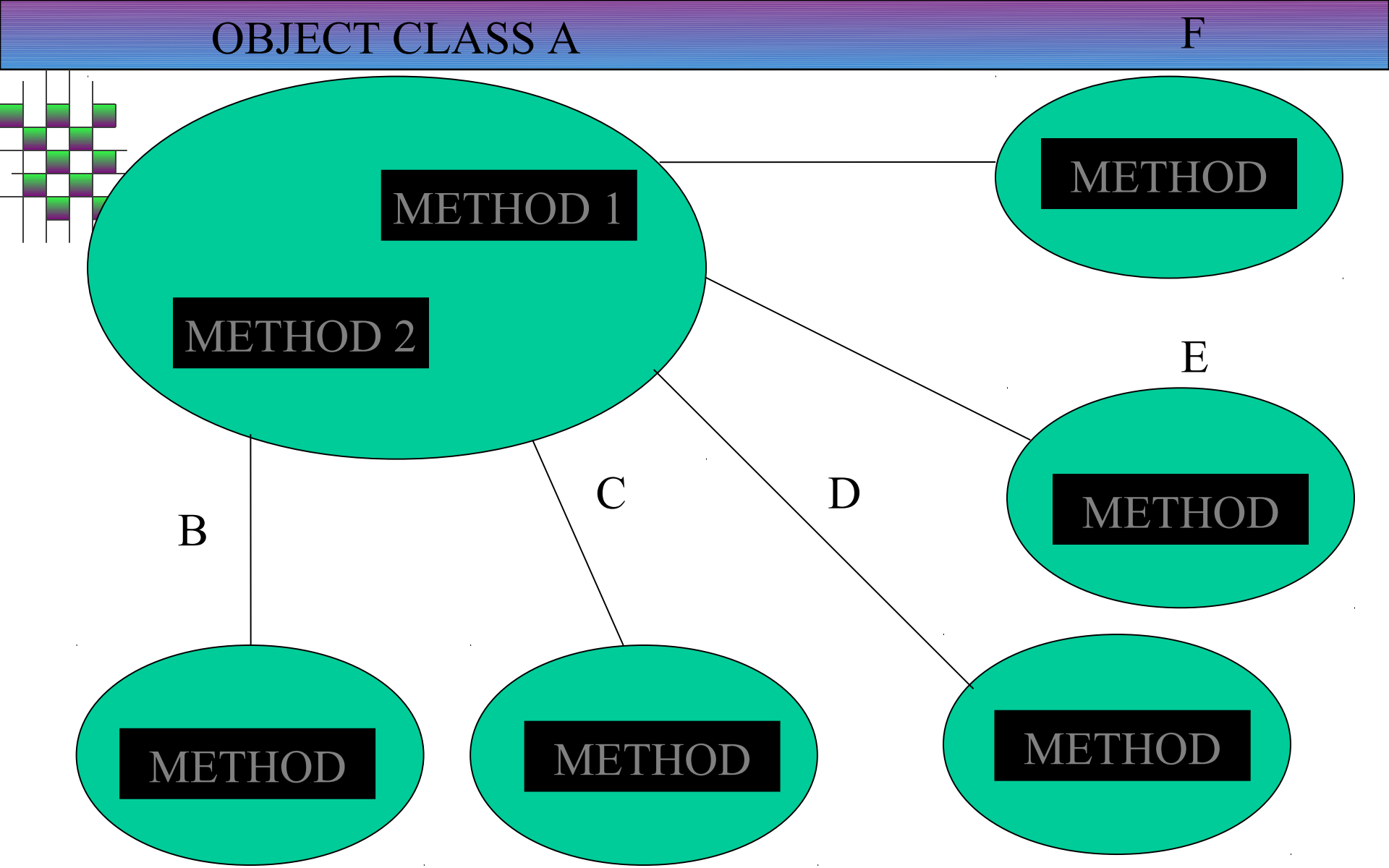
1) MM Testing

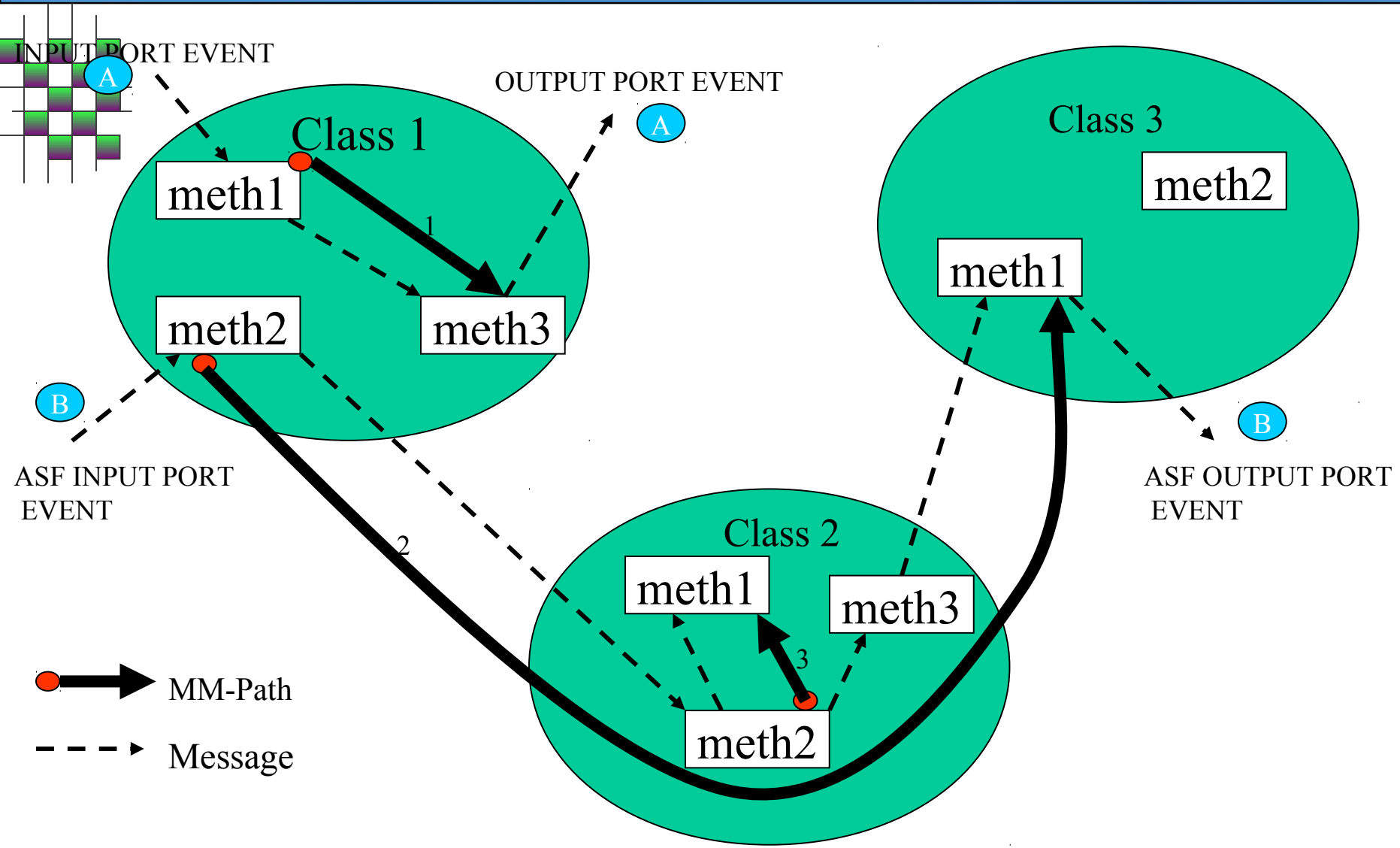
2) Function Pair Testing

MM Testing (Method-message testing)

- Every method call to be tested.
- A method calls another method multiple times, each calls needs to be tested only once.

- MM testing does not subsumes statement coverage


OBJECT CLASS A
METHOD 1
METHOD 2
F
METHOD
E
METHOD
B
C
D
METHOD
METHOD
METHOD


INPUT PORT EVENT
A
OUTPUT PORT EVENT
A
Class 1
meth1
meth2
meth3
1
B
ASF INPUT PORT EVENT
Class 3
meth2
meth1
B
ASF OUTPUT PORT EVENT
2
Class 2
meth1
meth3
3
meth2
MM-Path
Message

Identify the MM testing coverage for the linked list of rectangles problem

```
class point {
   float x;
   float y;
public:
   point(float newx, float newy) {x=newx; y=newy;}
   getx(){return x;}
   gety(){return y;}
};
class rectangle {
   point pt1, pt2, pt3, pt4;
public:
   rectangle(float pt1x, pt1y, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y)
      { pt1 = new point(pt1x, pt1y); pt2 = new point(pt2x, pt2y);
      pt3 = new point(pt3x, pt3y); pt4 = new point(pt4x, pt4y);}
   float length(point r, point s){return sqrt((r.getx()-s.getx())^2+
      (r.gety()-s.gety())^2); }
   float area(){return length(pt1,pt2) * length(pt1,pt3);}
};
class linklistnode {
   rectangle* node;
   linklistnode* next;
public:
   linklistnode(rectangle* newRectangle){node=newRectangle; next=0;}
   linklistnode* getNext(){return next;}
   rectangle* getRectangle(){return node;}
   void setnext(linklistnode* newnext){next=newnext;}
};
class rectanglelist {
   linklistnode* top;
public:
   rectanglelist(){top = 0;}
   void addRectangle(float x1, y1, x2, y2, x3, y3, x4, y4) {
      linklistnode* tempLinkListNode; rectangle* tempRectangle;
      tempRectangle = new rectangle(x1,y1,x2,y2,x3,y3,x4,y4);
      tempLinkListNode = new linkListNode(tempRectangle);
      tempLinkListNode->setnext(top);
      top=tempLinkListNode; }
   float totalArea(){float sum; sum=0; linklistnode* temp; temp=top;
      while (temp !=0){sum=sum + temp->getRectangle()->area();
         temp=temp->getNext();}
      return sum;}
};
```

```
class point
  point()
  getx()
  gety()

class rectangle
  rectangle()
    point::point()
    point::point()
    point::point()
    point::point()
  length()
    point::getx()
    point::getx()
    point::gety()
    point::gety()
  area()
    length()
    length()

class linklistnode
  linklistnode()
  getNext()
  getRectangle()
  setnext()
class rectanglelist
  rectanglelist()
  addRectangle()
    rectangle::rectangle()
    linklistnode::linklistnode()
    linklistnode::setnext()
  totalArea()
    linklistnode::getRectangle()
    rectangle::area()
    linklistnode::getNext()
```

MM Testing Strategy:

Any test case that builds at least one Rectangle and then gets the Total area will execute all of these calls

Function Pair Coverage Testing

- All possible sequence of method execution, Those of length 2 must be tested
- Usually done with state machine diagram or on regilar expression showing the possible method execution.

Identify the function pair testing coverage for the linked list of rectangles program of Example 13.1. Consider the calling structure of the functions.

```
class point
  point()
  getx()
  gety()

class rectangle
  rectangle()
    point()point()point()point()
  length()
    getx()getx()gety()gety()
  area()
    length()length()

class linklistnode
  linklistnode()
  getNext()
  getRectangle()
  setnext()'

class rectanglelist
  rectanglelist()
  addRectangle()
    rectangle()linklistnode()setnext()
  totalArea()
    (getRectangle()area()getNext())*
```

... done gives the following regular expression:

```
rectanglelist ((addRectangle rectangle point point point point
linklistnode setnext) | (totalArea (getRectangle area length getx getx
gety gety length getx getx gety gety getNext)*)
```

National Institute of Technology
Durgapur, India

Function Pair Testing Strategy:

Put all of function together into one regular expression..

One pair (addrectangle and total area two) function is generated and apply the following strategy

1. Create one rectangle and calculate area
2. Create 2/more rectangle and calculate area
3. Not create any rectangle and calculate area
4. Create rectangle after calculating area.

EXAMPLE 13.3

Identify the function pair testing coverage for the finite stack example shown in the state machine in Fig. 13-1. The error transitions are shown as arcs without destination states.

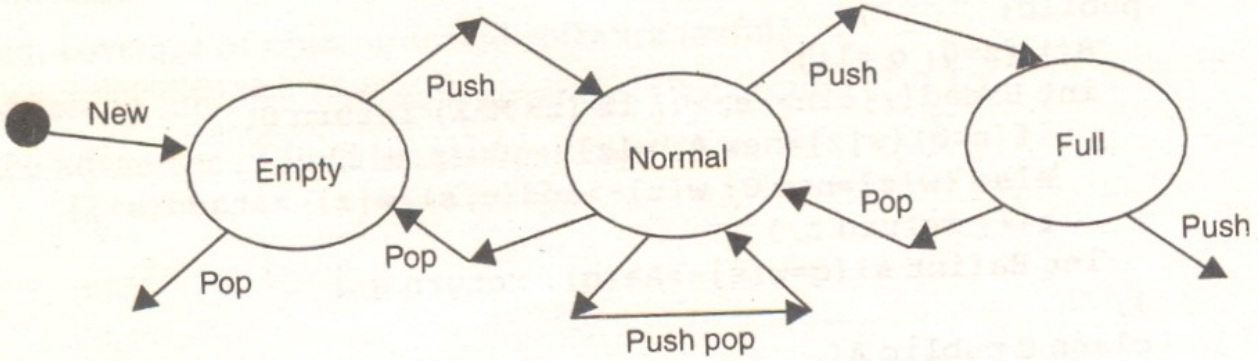


Fig. 13-1

1. new pop(on empty – error)
2. new push
3. push (from empty) push
4. push (from empty) pop
5. push (from normal to normal) push (still in normal)
6. push (from normal to normal) push (into full)
7. push (from normal to normal) pop
8. push (from normal to full) push (error)
9. push (from normal to full) pop
10. pop (from normal to normal) push (still in normal)
11. pop (from normal to normal) pop (still in normal)
12. pop (from normal to normal) pop (into empty)
13. pop (into empty) push
14. pop (into empty) pop (error)

Funtion pair coverage subsumes MM testing