Chương 3

Kỹ thuật kiểm thử hộp trắng

3.1 Tổng quát về kiểm thử hộp trắng

Đối tượng được kiểm thử là 1 thành phần phần mềm (TPPM). TPPM có thể là 1 hàm chức năng, 1 module chức năng, 1 phân hệ chức năng…

Kiểm thử hộp trắng dựa vào thuật giải cụ thể, vào cấu trúc dữ liệu bên trong của đơn vị phần mềm cần kiểm thử để xác định đơn vị phần mềm đó có thực hiện đúng không.

Do đó người kiểm thử hộp trắng phải có kỹ năng, kiến thức nhất định về ngôn ngữ lập trình được dùng, về thuật giải được dùng trong TPPM để có thể thông hiểu chi tiết về đoạn code cần kiểm thử.

Thường tốn rất nhiều thời gian và công sức nếu TPPM quá lớn (thí dụ trong kiểm thử tích hợp hay kiểm thử chức năng).

Do đó kỹ thuật này chủ yếu được dùng để kiểm thử đơn vị. Trong lập trình hướng đối tượng, kiểm thử đơn vị là kiểm thử từng tác vụ của 1 class chức năng nào đó.

Có 2 hoạt động kiểm thử hộp trắng :

- Kiểm thử luồng điều khiển : tập trung kiểm thử thuật giải chức năng.
- Kiểm thử dòng dữ liệu : tập trung kiểm thử đời sống của từng biến dữ liệu được dùng trong thuật giải.

Trong chương 3 này, chúng ta tập trung giới thiệu kiến thức về hoạt động kiểm thử luồng điều khiển của TPPM và trong chương 4, chúng ta tập trung giới thiệu các kiến thức về hoạt động kiểm thử dòng dữ liệu.

3.2 Một số thuật ngữ về kiểm thử luồng điều khiển

Đường thi hành (Execution path) : là 1 kịch bản thi hành đơn vị phần mềm tương ứng, cụ thể nó là danh sách có thứ tự các lệnh được thi hành ứng với 1 lần chạy cụ thể của đơn vị phần mềm, bắt đầu từ điểm nhập của đơn vị phần mềm đến điểm kết thúc của đơn vị phần mềm.

Mỗi TPPM có từ 1 đến n (có thể rất lớn) đường thi hành khác nhau. Mục tiêu của phương pháp kiểm thử luồng điều khiển là đảm bảo mọi đường thi hành của đơn vị phần mềm cần kiểm thử đều chạy đúng. Rất tiếc trong thực tế, công sức và thời gian để đạt mục tiêu trên đây là rất lớn, ngay cả trên những đơn vị phần mềm nhỏ.

Thí dụ đoạn code sau :

```
for (i=1; i<=1000; i++)
  for (j=1; j<=1000; j++)
        for (k=1; k<=1000; k++)
              doSomethingWith(i,j,k);
```

chỉ có 1 đường thi hành, nhưng rất dài : dài 1000*1000*1000 = 1 tỉ lệnh gọi hàm doSomething(i,j,k) khác nhau.

Còn đoạn code gồm 32 lệnh if else độc lập sau :

```
if (c1) s11 else s12;
if (c2) s21 else s22;
if (c3) s31 else s32;
...
if (c32) s321 else s322;
```

có 2^32 = 4 tỉ đường thi hành khác nhau.

Mà cho dù có kiểm thử hết được toàn bộ các đường thi hành thì vẫn không thể phát hiện những đường thi hành cần có nhưng không (chưa) được hiện thực :

```
if (a>0) doIsGreater();
if (a==0) doIsEqual();
// thiếu việc xử lý trường hợp a < 0 - if (a<0) doIsLess();
```

Một đường thi hành đã kiểm tra là đúng nhưng vẫn có thể bị lỗi khi dùng thật (trong 1 vài trường hợp đặc biệt) :

```
int phanso (int a, int b) {
   return a/b;
}
```

khi kiểm tra, ta chọn b <> 0 thì chạy đúng, nhưng khi dùng thật trong trường hợp b = 0 thì hàm phanso bị lỗi.

3.3 Các cấp phủ kiểm thử (Coverage)

Do đó, ta nên kiểm thử 1 số test case tối thiểu mà kết quả độ tin cậy tối đa. Nhưng làm sao xác định được số test case tối thiểu nào có thể đem lại kết quả có độ tin cậy tối đa ?

Phủ kiểm thử (Coverage) : là tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể sau khi đã kiểm thử các test case được chọn. Phủ càng lớn thì độ tin cậy càng cao.

Thành phần liên quan có thể là lệnh thực thi, điểm quyết định, điều kiện con hay là sự kết hợp của chúng.

Phủ cấp 0 : kiểm thử những gì có thể kiểm thử được, phần còn lại để người dùng phát hiện và báo lại sau. Đây là mức độ kiểm thử không thực sự có trách nhiệm.

Phủ cấp 1 : kiểm thử sao cho mỗi lệnh được thực thi ít nhất 1 lần.

Phân tích hàm foo sau đây :

```
1   float foo(int a, int b, int c, int d) {
2     float e;
3     if (a==0)
4       return 0;
5     int x = 0;
6     if ((a==b) || ((c==d) && bug(a)))
7       x = 1;
8     e = 1/x;
9     return e;
10  }
```

Với hàm foo trên, ta chỉ cần 2 test case sau đây là đạt 100% phủ cấp 1 :

1. foo(0,0,0,0), trả về 0

2. foo(1,1,1,1), trả về 1

nhưng không phát hiện lỗi chia 0 ở hàng lệnh 8.

Phủ cấp 2 : kiểm thử sao cho mỗi điểm quyết định luận lý đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh (Branch coverage). Phủ các nhánh đảm bảo phủ các lệnh.

Line	Predicate	True	False
3	(a == 0)	Test Case 1 foo(0, 0, 0, 0) return 0	Test Case 2 foo(1, 1, 1, 1) return 1
6	((a==b) OR ((c == d) AND bug(a)))	Test Case 2 foo(1, 1, 1, 1) return 1	Test Case 3 foo(1, 2, 1, 2) division by zero!

Với 2 test case xác định trong slide trước, ta chỉ đạt được 3/4 = 75% phủ các nhánh. Nếu thêm test case 3 :

3. foo(1,2,1,2), thì mới đạt 100% phủ các nhánh.

Phủ cấp 3 : kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các điều kiện con (subcondition coverage). Phủ các điều kiện con chưa chắc đảm bảo phủ các nhánh & ngược lại.

Predicate	True	False
a ==0	TC 1 : foo(0, 0, 0, 0) return 0	TC 2 : foo(1, 1, 1, 1) return 1
(a==b)	TC 2 : foo(1, 1, 1, 1) return value 0	TC 3 : foo(1, 2, 1, 2) division by zero!
(c==d)	TC 4 : foo(1, 2, 1, 1) Return 1	TC 3 : foo(1, 2, 1, 2) division by zero!

bug(a)	TC 4 : foo(1, 2, 1, 1) Return 1	TC 5 : foo(2,1, 1, 1) division by zero!

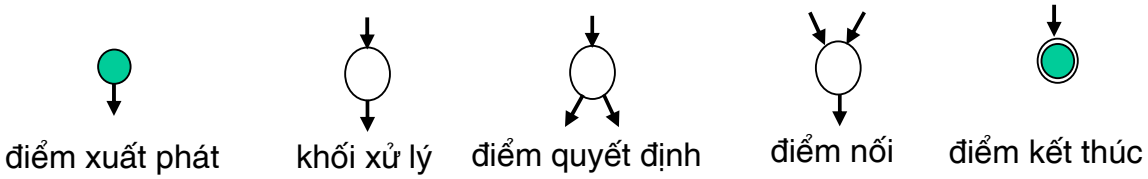
Phủ cấp 4 : kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE & điểm quyết định cũng được kiểm thử cho cả 2 nhánh TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh & các điều kiện con (branch & subcondition coverage). Đây là mức độ phủ kiểm thử tốt nhất trong thực tế. Phần còn lại của chương này sẽ giới thiệu qui trình kỹ thuật để định nghĩa các testcase sao cho nếu kiểm thử hết các testcase được định nghĩa này, ta sẽ đạt phủ kiểm thử cấp 4.

3.4 Đồ thị dòng điều khiển

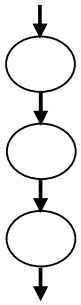
Là một trong nhiều phương pháp miêu tả thuật giải. Đây là phương pháp trực quan cho chúng ta thấy dễ dàng các thành phần của thuật giải và mối quan hệ trong việc thực hiện các thành phần này.

Gồm 2 loại thành phần : các nút và các cung nối kết giữa chúng.

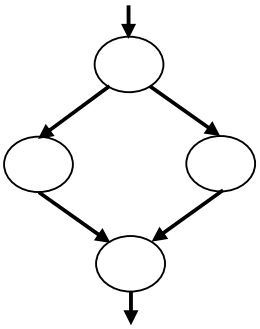
Các loại nút trong đồ thị dòng điều khiển :



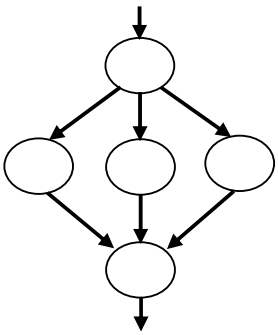
Miêu tả các cấu trúc điều khiển phổ dụng :



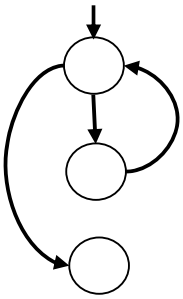
tuần tự



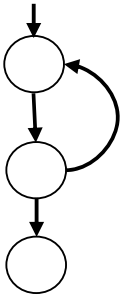
If



switch



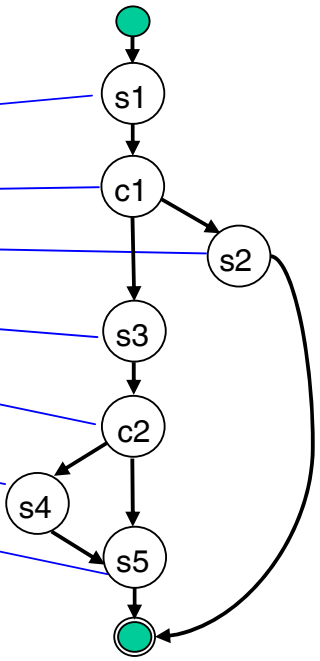
while c do...



do ... while c

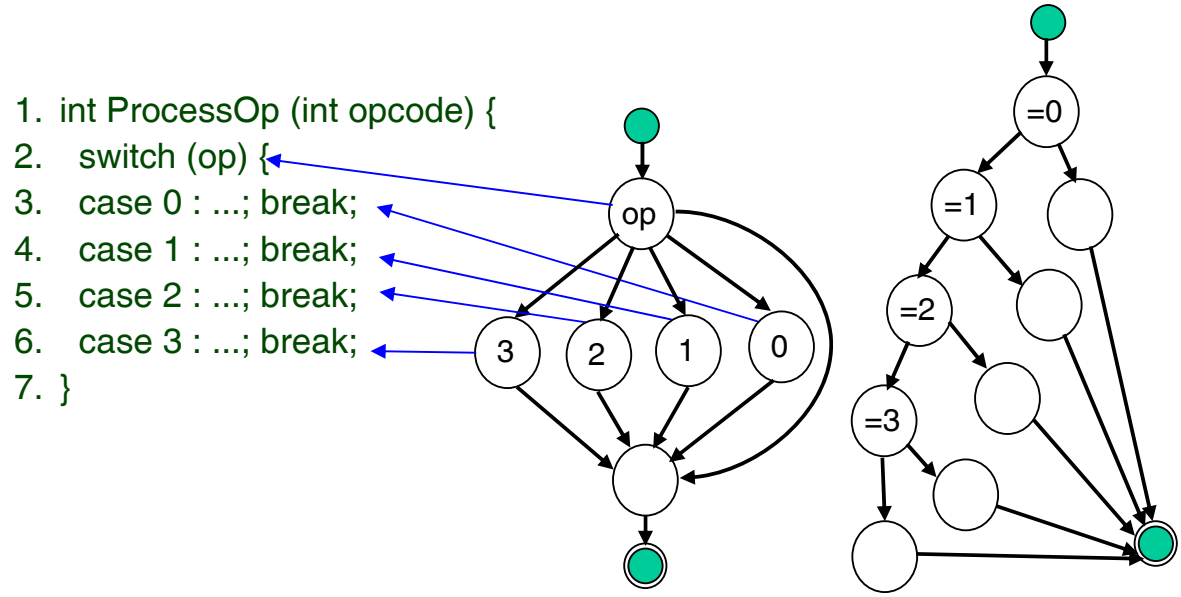
Thí dụ :

```
1.  float foo(int a, int b, int c, int d) {
2.    float e;
3.    if (a==0)
4.       return 0;
5.    int x = 0;
6.    if ((a==b) || ((c==d) && bug(a)))
7.      x = 1;
8.    e = 1/x;
9.    return e;
10.}
```



Nếu đồ thị dòng điều khiển chỉ chứa các nút quyết định nhị phân thì ta gọi nó là đồ thị dòng điều khiển nhị phân.

Ta luôn có thể chi tiết hóa 1 đồ thị dòng điều khiển bất kỳ thành đồ thị dòng điều khiển nhị phân.



Độ phức tạp Cyclomatic C

Độ phức tạp Cyclomatic C = V(G) của đồ thị dòng điều khiển được tính bởi 1 trong các công thức sau :

- V(G) = E - N + 2, trong đó E là số cung, N là số nút của đồ thị.
- V(G) = P + 1, nếu là đồ thị dòng điều khiển nhị phân (chỉ chứa các nút quyết định luận lý - chỉ có 2 cung xuất True/False) và P số nút quyết định.

Độ phức tạp Cyclomatic C chính là số đường thi hành tuyến tính độc lập của TPPM cần kiểm thử.

Nếu chúng ta chọn lựa được đúng C đường thi hành tuyến tính độc lập của TPPM cần kiểm thử và kiểm thử tất cả các đường thi hành này thì sẽ đạt được phủ kiểm thử cấp 3 như đã trình bày trong các slide trước.

3.5 Đồ thị dòng điều khiển cơ bản

Xét đồ thị dòng điều khiển nhị phân : nếu từng nút quyết định (nhị phân) đều miêu tả 1 điều kiện con luận lý thì ta nói đồ thị này là đồ thi dòng điểu khiển cơ bản.

Ta luôn có thể chi tiết hóa 1 đồ thị dòng điều khiển bất kỳ thành đồ thị dòng điều khiển nhị phân. Tương tự, ta luôn có thể chi tiết hóa 1 đồ thị dòng điều khiển nhị phân bất kỳ thành đồ thị dòng điều khiển cơ bản.

Tóm lại, ta luôn có thể chi tiết hóa 1 đồ thị dòng điều khiển bất kỳ thành đồ thị dòng điều khiển cơ bản.

Độ phức tạp Cyclomatic C của đồ thị dòng điều khiển cơ bản chính là số đường thi hành tuyến tính độc lập cơ bản của TPPM cần kiểm thử.

Nếu chúng ta chọn lựa được đúng C đường thi hành tuyến tính độc lập cơ bản của TPPM cần kiểm thử và kiểm thử tất cả các đường thi hành này thì sẽ đạt được phủ kiểm thử cấp 4 như đã trình bày trong các slide trước.

3.6 Qui trình kiểm thử hộp trắng

Tom McCabe đề nghị qui trình kiểm thử TPPM gồm các bước công việc sau :

1. Từ TPPM cần kiểm thử, xây dựng đồ thị dòng điều khiển tương ứng, rồi chuyển thành đồ thị dòng điều khiển nhị phân, rồi chuyển thành đồ thị dòng điều khiển cơ bản.
2. Tính độ phức tạp Cyclomatic của đồ thị (C = P +1).
3. Xác định C đường thi hành tuyến tính độc lập cơ bản cần kiểm thử (theo thuật giải chi tiết ở slide kế).
4. Tạo từng test case cho từng đường thi hành tuyến tính độc lập cơ bản.
5. Thực hiện kiểm thử trên từng test case.

6. So sánh kết quả có được với kết quả được kỳ vọng.
7. Lập báo cáo kết quả để phản hồi cho những người có liên qu

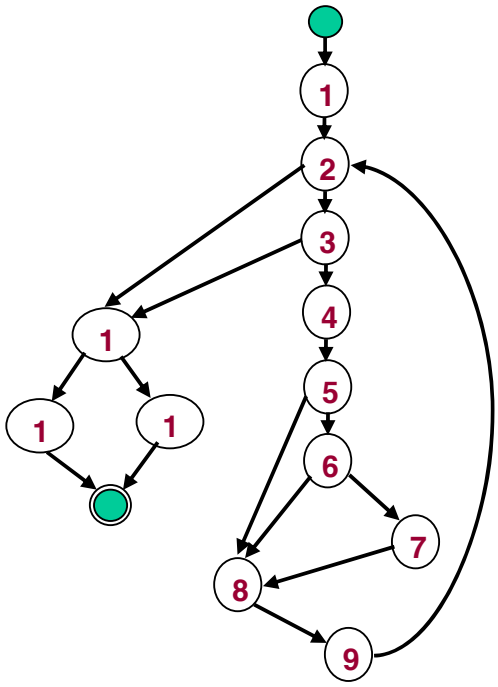
Qui trình xác định các đường tuyến tính độc lập

Tom McCabe đề nghị qui trình xác định C đường tuyến tính độc lập gồm các bước :

1. Xác định đường tuyến tính đầu tiên bằng cách đi dọc theo nhánh bên trái nhất của các nút quyết định. Chọn đường này là pilot.
2. Dựa trên đường pilot, thay đổi cung xuất của nút quyết định đầu tiên và cố gắng giữ lại maximum phần còn lại.
3. Dựa trên đường pilot, thay đổi cung xuất của nút quyết định thứ 2 và cố gắng giữ lại maximum phần còn lại.
4. Tiếp tục thay đổi cung xuất cho từng nút quyết định trên đường pilot để xác định đường thứ 4, 5,... cho đến khi không còn nút quyết định nào trong đường pilot nữa.
5. Lặp chọn tuần tự từng đường tìm được làm pilot để xác định các đường mới xung quanh nó y như các bước 2, 3, 4 cho đến khi không tìm được đường tuyến tính độc lập nào nữa (khi đủ số C).

3.7 Thí dụ

```
double average(double value[], double min,
        double max, int& tcnt, int& vcnt) {
  double sum = 0;
  int i = 1;
  tcnt = vcnt = 0;
  while (value[i] <> -999 && tcnt <100) {
    tcnt++;
    if (min<=value[i]  && value[i] <= max) {
      sum += value[i];
      vcnt ++;
    }
    i++;
  }
  if (vcnt > 0) return sum/vcnt;
  return -999;
}
```



Đồ thị bên có 5 nút quyết định nhị phân nên có độ phức tạp C = 5+1 = 6.

6 đường thi hành tuyến tính độc lập cơ bản là :

1. 1→2→10→11
2. 1→2→3→10→11
3. 1→2→3→4→5→8→9
4. 1→2→3→4→5→6→8→9
5. 1→2→3→4→5→6→7→8→9
6. 1→2→10→12

Thiết kế các test case

Phân tích mã nguồn của hàm average, ta định nghĩa 6 testcase kết hợp với 6 đường thi hành tuyến tính độc lập cơ bản như sau :

Test case cho đường 1 :

value(k) <>-999, với 1< k < i

value(i) = -999 với 2 ≤ i ≤ 100

Kết quả kỳ vọng : (1) average=Giá trị trung bình của i-1 giá trị hợp lệ. (2) tcnt = i-1. (3) vcnt = i-1

Chú ý : không thể kiểm thử đường 1 này riêng biệt mà phải kiểm thử chung với đường 4 hay 5 hay 6.

Test case cho đường 2 :

value(k) <>-999, với $\forall k < i$, i >100

Kết quả kỳ vọng : (1) average=Giá trị trung bình của 100 giá trị hợp lệ. (2) tcnt = 100. (3) vcnt = 100

Test case cho đường 3 :

value(1) = -999

Kết quả kỳ vọng : (1) average = -999. (2) tcnt = 0 (3) vcnt = 0

Test case cho đường 4 :

value(i) <> -999 $\forall i <= 100$

và value(k) < min với k < i

Kết quả kỳ vọng : (1) average=Giá trị trung bình của n giá trị hợp lệ. (2) tcnt = 100. (3) vcnt = n (số lượng giá trị hợp lệ)

Test case cho đường 5 :

value(i) <>-999 với $\forall i <= 100$

và value(k) > max với k <= i

Kết quả kỳ vọng : (1) average=Giá trị trung bình của n giá trị hợp lệ. (2) tcnt = 100. (3) vcnt = n (số lượng giá trị hợp lệ)

Test case cho đường 6 :

value(i) <>-999 và min <= value(i) <= max với $\forall i <= 100$

Kết quả kỳ vọng : (1) average=Giá trị trung bình của 100 giá trị hợp lệ. (2) tcnt = 100. (3) vcnt = 100

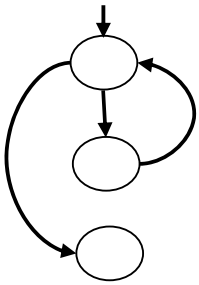
3.8 Kiểm thử vòng lặp

Thường thân của 1 lệnh lặp sẽ được thực hiện nhiều lần (có thể rất lớn). Chi phí kiểm thử đầy đủ rất tốn kém, nên chúng ta sẽ chỉ kiểm thử ở những lần lặp mà theo thống kê dễ gây lỗi nhất. Ta xét từng loại lệnh lặp, có 4 loại :

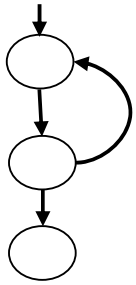
1. lệnh lặp đơn giản : thân của nó chỉ chứa các lệnh khác chứ không chứa lệnh lặp khác.
2. lệnh lặp lồng nhau : thân của nó có chứa ít nhất lệnh lặp khác...

3. lệnh lặp liền kề : 2 hay nhiều lệnh lặp kế tiếp nhau
4. lệnh lặp giao nhau : 2 hay nhiều lệnh lặp giao nhau.

1. Kiểm thử loại vòng lặp n lần đơn giản :



while c do...

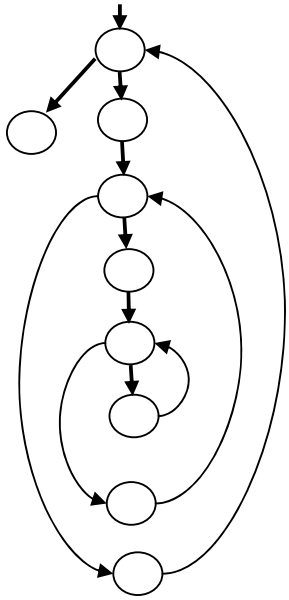


do ... while c

Nên chọn các test case để kiểm thử thân lệnh lặp ở các vị trí sau :

- chạy 0 bước.
- chạy 1 bước.
- chạy 2 bước.
- chạy k bước, k là giá trị nào đó thỏa $2 < k < n-1$.
- chạy n-1 bước
- chạy n bước
- chạy n+1 bước.

2. Kiểm thử vòng lặp lồng nhau :



Kiểm thử tuần tự từng vòng lặp từ trong ra ngoài theo đề nghị sau đây :

- kiểm thử vòng lặp trong cùng : cho các vòng ngoài chạy với giá trị min, kiểm thử vòng lặp trong cùng bằng 7 test case đã giới thiệu ở silde trước.
- kiểm thử từng vòng lặp còn lại : cho các vòng ngoài nó chạy với giá trị min, còn các vòng bên trong nó chạy với giá trị điển hình, kiểm thử nó bằng 7 test case đã giới thiệu ở slide trước.

3. Kiểm thử các vòng lặp liền kề : Kiểm thử tuần tự từng vòng lặp từ trên xuống, mỗi vòng thực hiện kiểm thử bằng 7 test case đã giới thiệu.

4. Riêng các vòng lặp giao nhau thì thường do việc viết code chưa tốt tạo ra $\Rightarrow$ nên cấu trúc lại đoạn code sao cho không chứa dạng giao nhau này.

3.9 Kết chương

Chương này đã giới thiệu 1 kỹ thuật thiết yếu để kiểm thử hộp trắng TPPM, đó là kỹ thuật kiểm thử dòng điều khiển.

Chúng ta đã giới thiệu các cấp độ phủ kiểm thử khác nhau, giới thiệu đồ thị dòng điều khiển và đồ thị dòng điều khiển cơ bản

của TPPM, độ phức tạp Cyclomatic C, qui trình tổng quát để kiểm thử dòng điều khiển.

Chương này cũng đã giới thiệu 1 thí dụ cụ thể về qui trình kiểm thử dòng điều khiển trên 1 TPPM.