程序的规格说明和测试

乔海燕
中山大学信息学院

2014 年 10 月 8 日

程序的规格说明(Specifications)陈述程序输入与输出应该满足的关系。程序测试的目的是找出程序中的错误，即试图找出不满足规格说明的输入和输出，因此程序测试可以依据规格说明设计，并尝试找出一对不满足规格说明的输入和输出。

下面以排序和二分查找为例，说明如何依据其规格说明设计测试代码。

§1 排序的规格说明和测试

§1.1 排序函数

假定排序的函数类型如下：

```
vector<int> sort(vector<int> &v);
//将v中元素从小到大排序，并返回排序后的向量。
```

我们的目的是测试sort是否“正确”，或者说sort是否满足注释说明的功能。

§1.2 排序的规格说明

按照sort的功能要求说明，正确的实现应该满足这样的命题P：

“对于整数向量v, 令u = sort(v)，则u 是有序的，并且u是v的重新排列。”

注意：仅仅说明“u是有序的”并不能表达排序的正确性。

假定用V表示向量集合，I_v表示向量v的下标集合$\{0, 1, .., v.size() - 1\}$, 用形式化命题可以表示成

$P = \forall v \in V(ordered(u) \wedge reordering(u, v))$ 其中$u = sort(v)$, $ordered(u)$是表达“u是从小到大排列”的一元谓词，$reordering(u, v)$是表示“u是v的重新排列”的二元谓词。

测试就是尝试找出不满足以上规格说明命题的v和u, 或者满足规格说明的否命题“¬P”的v和u.

命题P的否命题：

$\neg P = \exists v \in V(\neg ordered(u) \vee \neg reordering(u, v))$ 其中$u = sort(v)$。

§1.3 排序的测试

测试的目标是找出使得“¬P”成立的v。测试的方法是通过枚举V中的v,检查命题公式“¬P”是否成立。

因为测试只能测试有限个输入，所以我们应该选择集合V的一个有限子集V′作为测试集，而且尽可能使得该子集覆盖不同的输入情况。

假定我们选定V的一个有穷子集U作为测试集，而且U覆盖了尽可能多的不同情况的向量，如长度覆盖了0,1,2,...等，包含有序（递增）的向量，逆序向量，向量元素均为某个常数，随机生成的向量等各种情况。

按照以上说明，可以设计如下测试算法：

```
对U中每个向量v
    令 u = sort(v)
    如果ordered(u)不成立，则输出反例(v,u)，并结束测试
    如果reodering(u,v)不成立，则输出反例(v,u)，并结束测试
```

读者不妨按照以下步骤，将以上算法转换成代码，并对某种排序算法sort进行测试。

习题 1 实现一种排序算法：

```
vector<int> sort(vector<int> &v);
//将v中元素从小到大排序，并返回排序后的向量。
```

习题 2 实现一元谓词：

```
bool ordered(const vector<int> &v);
//如果v中元素是从小到大有序的，则返回true，否则返回false.
```

习题 3 实现一种排序算法：

```
bool reordering(const vector<int> &u, const vector<int>&v);
//如果u是v的重新排列，则返回true，否则返回false.
```

习题 4 `set<vector<int> > gen(int n)`

```
//生成长度为n的向量集合，并尽可能包含各种情况的向量，
//例如有序、逆序、常数向量、随机生成的向量等。
```

习题 5 综合以上函数的实现，对排序函数sort进行尽可能多输入的自动测试。

§2 二分查找的规格说明和测试

§2.1 二分查找函数

假定二分查找的函数类型如下：

```
bool binary_search(const vector<int> &v, int x);
//v是整数非递减序列，如果x在v中出现，则返回true，否则返回false.
```

在这里，二分查找函数binary_search的输入是v和x, 输出是函数的返回值true或者false。注释说明了其功能要求。

我们的目的是测试binary_search是否“正确”，或者说binary_search是否满足其功能要求。

§2.2 二分查找的规格说明

按照binary_search的功能要求说明，正确的实现应该满足这样的命题P：

“对于任意递增整数向量v, 对于任意整数x, 如果x在v中出现，则binary_search(v,x)返回true, 否则binary_search(v,x)返回false.”

或者

“对于任意递增整数向量v, 对于任意整数x = v[i] (i=0, ..., v.size()-1), binary_search(v,x)返回true, 并且对于任意的x, 如果x不等于任何v[i] (i=0, ..., v.size()-1), 则binary_search(v,x)返回false.”

假定用V表示递增整数向量集合，I_v表示向量v的下标集合$\{0, 1, .., v.size()-1\}$, 用形式化命题可以表示成

$\mathrm{P} = \forall v \in \mathrm{V}(\forall i \in \mathrm{I}_v.(binary\_search(v, v[i]) == true) \wedge \forall x \notin v.binary\_search(v, x) == false)$

测试就是尝试找出不满足以上规格说明命题的v和x, 或者满足规格说明的否命题“¬P”的v和x.

命题P的否命题：

“存在一个递增整数向量v, 存在某个整数x = v[i] (i=0, ..., v.size()-1)使得binary_search(v,x)返回false, 或者存在某个x，x不等于任何v[i] (i=0, ..., v.size()-1), 使得binary_search(v,x)返回true.”或者

$\neg\mathrm{P} = \exists v \in \mathrm{V}(\exists i \in \mathrm{I}_v.(binary\_search(v, v[i]) == false) \vee \exists x \notin v.binary\_search(v, x) == true)$

§2.3 二分查找的测试

测试的目标是找出使得“¬P”成立的v, i或者x。测试的方法是通过枚举命题公式“¬P”中的v、i和x, 检查命题“¬P”是否为真。

因为测试只能测试有限个输入，所以我们应该选择集合V的一个有限子集V′作为测试集，而且尽可能使得该子集覆盖不同的情况。为了简单起见，我们选择奇数递增序列的子集：$\mathrm{V}' = \{(), (1), (1,3), (1,3,5), ..., (1,3,5,..., N)\}$, 其中N是某个正整数。

同样，对于$x \notin v$这样的输入x, 也可以取$\bar{v}$的一个子集，如取偶数子集。

按照以上说明，可以设计如下测试代码：

```
int N = 10001; //最大向量长度
vector<int> v; //递增向量
int x; //查找的对象
for (int k=1; k<=N; k=k+2) {
```

```
    //对递增向量v进行系统查找测试
    for (int i=0; i<v.size(); i++) {
        x = v[i];
        //如果查找x返回false，则报告错误。
        if (!binary_search(v,x)){
            cout << "Obs! Error found! Counter example:";
            cout <<"v= ";
            print(v); //伪代码，输出有问题的向量v
            cout<<"x = " << x <<endl;
            return;
        };
    }
    for (x = 0; x<N; x=x+2) {
       //如果查找偶数x返回true，则报告错误。
       if (binary_search(v,x)){
            cout << "Obs! Error found! Counter example:";
            cout <<"v= ";
            print(v); //伪代码，输出有问题的向量v
            cout<<"x = " << x <<endl;
            return;
        };
    }
    v.push_back(k); //生成更长的奇数递增向量，进入下一轮测试
}
```

注意，以上测试中V′的选择很有限，比如并没有包含非严格递增的向量，如$v = (1, 2, 2, 3, 4, 4)$, 也没有包含所有元素相同的情况，如$v = (3, 3, 3, 3)$，也没有包含负数的情况，如$v = (-3, 0, 1)$等.

我们不妨再添加一段代码，尝试覆盖这些情况。也可以用随机函数生成一个向量，然后排序，用这样的向量作为测试输入。

习题 6 完成二分查找的实现和二分查找的自动测试。

总之，我们可以首先将程序的规格说明及其否定表达为命题，并设计代码设法在输入集合中找出满足规格说明否定命题的输入和输出。

另外，测试没有发现错误，并不能说明程序中没有错误。正如Dijkstra所讲：“Testing shows the presence, not the absence of bugs.”