# AMT 经验研究汇报

# CONTENTS
# 目录

01 实验对象

02 实验变量

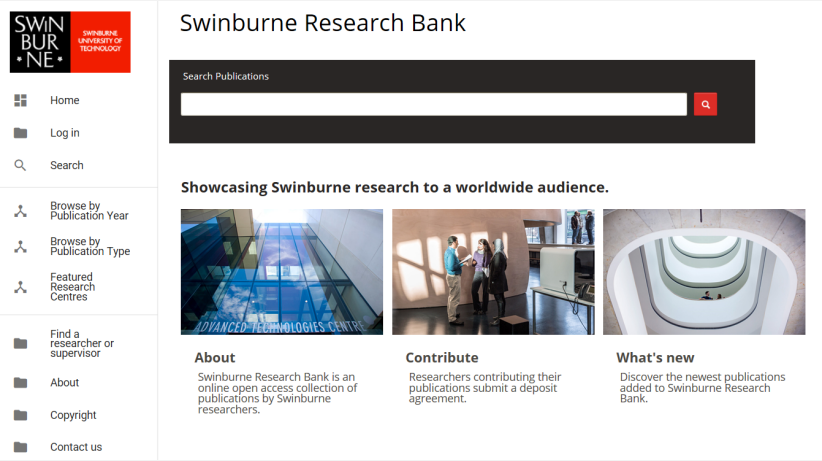
03 实验设置

# 实验对象

| Source | Programs | Language | LOC | All Faults | Used Faults | Number of MRs | Number of Partitions |
|---|---|---|---|---|---|---|---|
| Laboratory | CUBS | Java | 107 | 187 | 4 | 184 | 8 |
| | ACMS | Java | 128 | 210 | 9 | 142 | 4 |
| | ERS | Java | 117 | 180 | 4 | 1130 | 12 |
| | MOS | Java | 135 | 215 | 1 | 3512 | 9 |
| GUN | grep | C | 10,068 | 20 | 20 | 12 | 552 |
| Alibaba | FastJson | Java | 204125 | 5 | 5 | 9 | 9 |

- 潜在的基准系统（METWiKI）
  包含多个程序以及相应的蜕变关系，网址是：
  https://researchbank.swinburne.edu.au/home.do

缺少账号密码



# 实验对象

➢ 实验室程序

① ACMS根据乘客的座舱等级、携带的行李重量、目的地（国内或者国外）以及是否为学生计算需要支付的托运费用

② CUBS根据用户选择的套餐以及当月的实际使用的流量和通话时间计算消费额

③ ERS协助公司销售总监：(1) 确定每个高级销售经理和因使用公司车辆产生的“过度”英里数而应向公司补偿的费用；(2) 处理高级销售经理、销售经理和主管的差旅和电话等各种类型的报销请求

④ MOS被航空餐饮公司用来确定某种型号的飞机需要准备和装载的每类型餐饮和其他特殊要求食物的数量

- 选择ACMS、CUBS、ERS和MOS作为实验对象的理由

① 这些程序在多个与本文相关的研究作为研究对象，具有较高代表性

② 这些程序的输入格式规范，易于生成测试用例；规格说明简单，易于识别蜕变关系

# 实验对象

➢ GUN-grep

查找指定文件中包含指定模式的内容

```
grep [option...] [patterns] [file...]
```

Linux $ grep “\d ”<File>.

- 选择grep作为实验对象的理由

① grep是开源软件，它的源代码、规格所明书与补丁都可以得到

② grep的规模以及复杂度适合作为研究对象

③ grep的输入比较复杂，但是仍然可以自动生成测试用例（正则表达式和与之匹配的目标文件）

④ grep程序已经被多个工作研究，它的蜕变关系以及正则表达式是可以得到的

# 实验对象

➢ Alibaba-FastJson

阿里巴巴旗下的开源JSON解析库，它可以解析JSON格式的字符串，支持将Java Bean/POJO序列化为JSON字符串，也可以从JSON字符串反序列化到JavaBean/POJO

- 序列化：将对象的状态信息转换为可以存储或传输的形式的过程
- 反序列化：将存储区的内容恢复为对象的过程称为反序列化

- 选择FastJson作为实验对象的理由

① 可以得到源代码、API文档与补丁

② 作为目前世界上最快的JSON解析库，该程序被多个知名IT公司以及众多开发者使用，并且程序规模较大，适合作为开源软件的代表程序

③ 虽然输入格式复杂，但是仍能自动生成测试用例

```java
package com.alibaba.fastjson;

public abstract class JSON {
    // 将JSON字符串反序列化为JavaBean
    public static <T> T parseObject(String jsonStr,
                                    Class<T> clazz,
                                    Feature... features);

    // 将JSON字符串反序列化为JavaBean
    public static <T> T parseObject(byte[] jsonBytes,  // UT
                                    Class<T> clazz,
                                    Feature... features);

    // 将JSON字符串反序列化为泛型类型的JavaBean
    public static <T> T parseObject(String text,
                                    TypeReference<T> type,
                                    Feature... features);

    // 将JSON字符串反序列为JSONObject
    public static JSONObject parseObject(String text);
}
```

# 实验变量

## ➢ 自变量

用来检测故障的测试技术称为自变量

| Techniques | | Test case selection strategy | MRs selection strategy |
|---|---|---|---|
| Proposed Techniques | M-AMT | MAPT | RT |
| | | MAPT | PBMR |
| | R-AMT | RAPT | RT |
| | | RAPT | PBMR |
| Baselines | MT | RT | RT |
| | MT | RT | Static Strategy |
| | MT-DRT | DRT | RT |

Static Strategy：执行测试之前选择蜕变关系，该策略的依据以及具体步骤在蜕变关系方法总结文档中。

# 实验变量

➢ 因变量

用来衡量测试技术有效性和性能的度量标准

- 有效性度量标准

① F-measure: 检测第一个故障需要的测试用例数目

② F2-measure：检测第一个故障后再检测一个故障需要的测试用例数目

③ T-measure: 检测所有的故障需要的测试用例数目

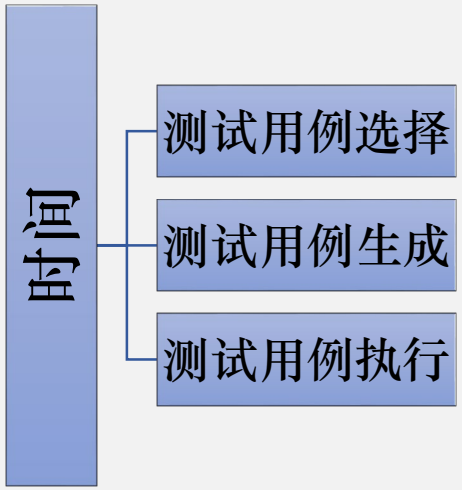
需要统计生成衍生测试用例的时间吗？

- 时间度量标准

① F-time: 检测第一个故障需要的时间

② F2-time：检测第一个故障后再检测一个故障需要的时间

③ T-time: 检测所有的故障需要的时间



# 实验变量

## ➢ 因变量

- 假设检验

$H_0 : A_{AMT} = A_{MT} = A_{MT\text{-}ART} = A_{MT\text{-}DRT}$

$H_1 : A_{AMT} \neq A_{MT} \neq A_{MT\text{-}ART} \neq A_{MT\text{-}DRT}$

计算自由度，计算卡方，拒绝零假设，然后利用Holm-Bonferroni方法量化不同测试技术的故障检测能力

- 效应值

用来比较每对测试技术在故障检测以及性能方面的差异

# 实验设置

➢ 故障

- 实验室程序的故障产生方式

  利用MuJava中可使用的变异算子为每一个程序生成变异体

- 变异体筛选流程

  一些变异体能被简单的测试策略轻易地杀死，我们采取以下步骤筛选出“难”杀得变异体

  ① 采用随机测试策略，利用不同的随机数种子产生大小为10000的30个测试用例集，然后在每一个变异体上执行所有的测试用例集，统计每一个变异体被杀死需要的平均测试用例数目

  然而不能获得满足一个理想阈值（至少10个测试用例才能杀死）的变异体集合

  ② 利用蜕变测试策略（逐个执行每个测试用例），利用CPM方法来生成测试用例集，然后在每一个变异体上执行所有的测试用例，统计每一个变异体被杀死需要的测试用例数目

  ③ 从能被同一个测试用例杀死的变异体集合中随机选择一个作为研究对象

| 程序 | 变异体数目 |
|---|---|
| ACMS | 3 |
| CUBS | 1 |
| ERS | 1 |
| MOS | 1 |

# 实验设置

## ➢ 故障

- GUN程序的故障获取方式

grep实验对象包含20个故障，这些故障可以分为两种：

① 真实故障，即grep在演变的过程中官方公布的故障，该故障与正则表达式有关

② 变异分析，利用两种变异算子：语句变异算子（将continue替换为break）和操作变异算子（替换算术或逻辑运算符）

不使用SIR中grep的故障原因是：SIR中grep故障与正则表达式无关

# 实验设置

## ➢ 故障

- Alibaba程序故障的获取方式

  从FastJson（V1.2.49）版本发布公告中获得6个与反序列化相关的故障

  （ https:/ ases?after=1.2.50 ）

- 故障1：反 下丢失精度（V1.2.48，#1944）

```
public static void main(String[] args) {
    String json="{\"num\":90.82195113}";
    Num num= JSONObject.parseObject(json,Num.class);
    System.out.println(num.getNum());
}


public static class Num {
    private float num;

    public float getNum() {
        return num;
    }

    public void setNum(float num) {
        this.num = num;
    }
}
```

输出结果如下：

```
4.922605
```

# 实验设置

➢ 故障

- 故障2：当一个类中有Map和Class[]时，整个类会反序列化失败

  该类型的故障会在控制台报错

- 故障3：LocalDateTime包含纳秒,反序列化失败

  该类型的故障会在控制台报错

- 故障4：反序列化英文格式时间异常

  该类型的故障会在控制台报错

- 故障5：不能转化时间戳

  该类型的故障会在控制台报错

- 故障6：反序列化Map<String,String>中属性Date类型值为1970年前失败

  该类型的故障会在控制台报错

报错的故障不适合作为研究对象

# 实验设置

➢ 故障（从其它版本中获取故障）

- 故障筛选步骤：

  ① FastJson的1.2.X版本（1.1.X没有故障信息）

  ② 与反序列化有关

  ③ 不报异常信息

  ④ 不报错误信息

- 故障7：超出float的精度范围，结果出现负数 (V1.2.46, #1723)

- 故障8：反序列化枚举，错误的枚举值不抛异常 (V1.2.45, #1393)

- 故障9：转换对象时失败，对象中有两个map集合 (V1.2.32, #1189)

```
// Bean 对
 public cl
    privat
    public
        re
    }
    public
        th
    }
    @Overr
    public
        re


    }
}
```

```
// 反序列化
User user
 System.ou
```

结果：

期望结果是：

实际结果是：

```
public enum Code{
        SUCCESS, FAILURE
}

public static class Result{
        private int id;
        private Code code;
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public Code getCode() {
                return code;
        }
        public void setCode(Code code) {
                this.code = code;
        }
        @Override
        public String toString() {
                return JSON.toJSONString(this);
        }
}

public static void main(String[] args) {
        String jsonText = "";
        jsonText = "{\"code\":\"SUCCESS\",\"id\":1}";
        System.out.println(JSON.parseObject(jsonText, Result.class)
        jsonText = "{\"code\":\"FAILURE\",\"id\":2}";
        System.out.println(JSON.parseObject(jsonText, Result.class)
        jsonText = "{\"code\":\"ERROR\",\"id\":3}";
        System.out.println(JSON.parseObject(jsonText, Result.class)
}
```

测试结果：

{"code":"SUCCESS","id":1}

{"code":"FAILURE","id":2}

{"id":3}

# 实验设置

## ➢ 测试用例集

- 实验室程序的范畴和选项

Definition of Categories and Choices for ACMS

| # | Category | Choice |
|---|---|---|
| 1 | class | first calss |
| | | Business class |
| | | Economy class |
| 2 | region | infant |
| | | Domestic flights |
| | | International flights |
| 3 | isStudent | True |
| | | False |
| 4 | luggage | luggage < Free checked-in |
| | | luggage ≤ Free checked-in |
| 5 | fee | fee = 0 |
| | | fee ≤ 0 |

Definition of Categories and Choices for CUBS

| # | Category | Choice |
|---|---|---|
| 1 | plan | A |
| | | B |
| 2 | options | 46CNY |
| | | 96CNY |
| | | 286CNY |
| | | 886CNY |
| | | 126CNY |
| | | 186CNY |
| 3 | calls | calls < free calls |
| | | calls ≥ free calls |
| 4 | data | data < free data |
| | | data ≥ free data |

Definition of Categories and Choices for ERS

| # | Category | Choice |
|---|---|---|
| 1 | title | senior sales manager |
| | | sales manager |
| | | sales executive |
| 2 | mileage | $0 \leq mileage \leq 3000$ |
| | | $3000 \leq mileage \leq 4000$ |
| | | $mileage \geq 4000$ |
| 3 | sales | $0 \leq sales \leq 50,000$ |
| | | $50,000 \leq sales \leq 80,000$ |
| | | $80,000 \leq sales \leq 100,000$ |
| | | $sales \geq 100,000$ |
| 4 | airline ticket | $airlineticket = 0$ |
| | | $airlineticket \leq 0$ |
| 5 | other expenses | $otherexpenses = 0$ |
| | | $otherexpenses \leq 0$ |

表 5-49 MOS 范畴选项划分

| Categories | Choices |
|---|---|
| 1. Aircraft model | 1a. 747200 |
| | 1b. 747300 |
| | 1c. 747400 |
| | 1d. 000200 |
| | 1e. 000300 |
| 2. Change in the number of crew members | 2a yes |
| | 2b. no |
| Categories | Choices |
| 3. Number of crew(crewNum) | 3a. crewNum > default value |
| | 3b. crewNum = default value |
| | 3c. crewNum < default value |
| 4. Change in the number of pilots | 4a. yes |
| | 4b. no |
| 5. Number of pilots(pilotNum) | 5a. pilotNum > default value |
| | 5b. pilotNum = default value |
| | 5c. pilotNum < default value |
| 6. Number of Child Passengers(childNum) | 6a. childNum > 0 |
| | 6b. childNum = 0 |
| 7. Number of bundles of flowers(flowerNum) | 7a. flowerNum > 0 |
| | 7b. flowerNum = 0 |

- 将不同范畴中的选项进行组合依次得到40、32、70和720个完整有效的测试帧

- 每个完整有效的测试帧实例化一个测试用例

# 实验设置

## ➢ 测试用例集

- GUN程序的范畴和选项

| independent category (A) | dependent category (B) |
|---|---|
| NormalChar | Bracket |
| wordSymbol | Iteration |
| DigitSymbol | Parentheses |
| SpaceSymbol | Line |
| NamedSymbol | Word |
| AnyChar | Edge |
| Range | Combine |

- 将不同范畴中的选项进行组合，得到101210个完整有效的测试帧（在原有171634个有效的基础上移除重复的测试帧）

- 将每个测试帧实例化一个测试用例

# 实验设置

➢ 测试用例集

- FastJson测试数据的组成部分
  ① POJO (简单的Java对象)
  ② JSON文件
- FastJson输入空间异常庞大

  JavaBean是一个从现实世界通过映射得到的对象，其成员

  和复杂化的特点
- FastJson的基准系统（benchmark）
  ① Eishay：一个用来比较序列化库性能的基准系统，包含
  ② FastJson的内部测试用例数目超过3321个，但是找不到

```java
class MediaContent {
        public Media media;
        public Image[] images;
}
class Media implements java.io.Serializable {
```

```
// The standard test value.
{
        "media": {
                "uri": "http://javaone.com/keynote.mpg",
                "title": "Javaone Keynote",
                "width": 640,
                "height": 480,
                "format": "video/mpg4",
                "duration": 18000000,     // half hour in milliseconds
                "size": 58982400,         // bitrate * duration in seconds / 8 bits per byte
                "bitrate": 262144,        // 256k
                "persons": ["Bill Gates", "Steve Jobs△"],
                "player": "JAVA",
                "copyright": null
        },
        "images": [
                {
                        "uri": "http://javaone.com/keynote_large.jpg",
                        "title": "Javaone Keynote",
                        "width": 1024,
                        "height": 768,
                        "size": "LARGE"
                },
                {
                        "uri": "http://javaone.com/keynote_small.jpg",
                        "title": "Javaone Keynote",
                        "width": 320,
                        "height": 240,
                        "size": "SMALL"
                }
        ]
}
```

```java
        public int    height;
        public Size   size;
}
```

# 实验设置

## ➢ 测试用例集

- 基于场景的FastJson测试用例集生成方法

  将人映射为POJO，通过人的各种成员变量模拟FastJson反序列化的各种场景

- 范畴与选项

| 范畴 | 选项 |
|---|---|
| 1:变量命名规则 | 1-1：蛇形；1-2：驼峰；1-3：大驼峰（Pascal） |
| 2:float类型 | 2-1:不存在；2-2:存在，其数值在精度范围内；2-3:存在，其数值不在精度范围内 |
| 3:double类型 | 3-1:不存在；3-2:存在，其数值在精度范围内；3-3:存在，其数值不在精度范围内 |
| 4:short类型 | 4-1:不存在；4-2:存在，其数值在精度范围内；4-3:存在，其数值不在精度范围内 |
| 5:byte类型 | 5-1:不存在；5-2:存在，其数值在精度范围内；5-3:存在，其数值不在精度范围内 |
| 6:int类型 | 6-1:不存在；6-2:存在，其数值在精度范围内；6-3:存在，其数值不在精度范围内 |
| 7:long类型 | 7-1:不存在；7-2:存在，其数值在精度范围内；7-3:存在，其数值不在精度范围内 |
| 8:boolean类型 | 8-1：不存在；8-2：存在 |
| 9:char类型 | 9-1:不存在；9-2:存在 |
| 10:date类型 | 10-1:不存在；10-2:yyyy-MM-dd HH:mm:ss；10-3:yyyy/MM/dd HH:mm:ss；10-4:yyyy年M月d日 HH:mm:ss；10-5:yyyy年M月d日 H时m分s秒；10-6:yyyy년M월d일 HH:mm:ss；10-7:MM/dd/yyyy HH:mm:ss；10-8:dd/MM/yyyy HH:mm:ss；10-9:dd.MM.yyyy HH:mm:ss；10-10:dd-MM-yyyy HH:mm:ss；10-11:yyyyMMdd；10-12:yyyy/MM/dd；10-13:yyyy年M月d日；10-14:yyyy년M월d일；10-15:MM/dd/yyyy；10-16:dd/MM/yyyy；10-17:dd.MM.yyyy；10-18:dd-MM-yyyy |
| 11:string类型 | 11-1：不存在；11-2：存在，小于80个字符；11-3：存在，大于80个字符 |
| 12:enum类型 | 12-1：不存在；12-2：存在 |
| 13：Map | 13-1：不存在；13-2：HashMap；13-3：LinkedHashMap；13-4：dentityHashMap |
| 14：List | 14-1：不存在；14-2：ArrayList；14-3：LinkedList |
| 15：Set | 15-1：不存在；15-2：HashSet；15-3：LinkedHashSet |

最简单的选项组合有382,954,448种

# 实验设置

➢ 测试用例集

通过为不同范畴中的选项组合添加约束来约

- 测试规格说明书描述语言——TSL

通过为选项添加属性、声明和选择表达式，测试用例数目的目的

① 属性：[property A, B, … ]

② 声明：[error]、[single]

③ 选择表达式：[if A]

- 基于TSL的测试规格说明书示例

```
Parameters:
    Pattern size:
        empty                                   [property Empty]
        single character                        [property NonEmpty]
        many character                          [property NonEmpty]
        longer than any line in the file        [error]

    Quoting:
        pattern is quoted                       [property Quoted]
        pattern is not quoted                   [if NonEmpty]
        pattern is improperly quoted            [error]

    Embedded blanks:
        no embedded blank                       [if NonEmpty]
        one embedded blank                      [if NonEmpty and Quoted]
        several embedded blanks                 [if NonEmpty and Quoted]

    Embedded quotes:
        no embedded quotes                      [if NonEmpty]
        one embedded quote                      [if NonEmpty]
        several embedded quotes                 [if NonEmpty] [single]

    File name:
        good file name
        no file with this name                  [error]
        omitted                                 [error]


Environments:
    Number of occurrences of pattern in file:
        none                                    [if NonEmpty] [single]
        exactly one                             [if NonEmpty] [property Match]
        more than one                           [if NonEmpty] [property Match]

    Pattern occurrences on target line:
    #  assumes line contains the pattern
        one                                     [if Match]
        more than one                           [if Match] [single]
```

# 实验设置

## ➢ 测试用例集

- FastJson测试用例约减方案

| 组别 | 范畴 |
| --- | --- |
| 第一组 | 1 |
| 第二组 | 2,3,4,5,6,7,8,9 |
| 第三组 | 10 |
| 第四组 | 11 |
| 第五组 | 12 |
| 第六组 | 13 |
| 第七组 | 14 |
| 第八组 | 15 |

有34992个选项组合

# 实验设置

## ➢ 蜕变关系

- 实验室程序的蜕变关系获取

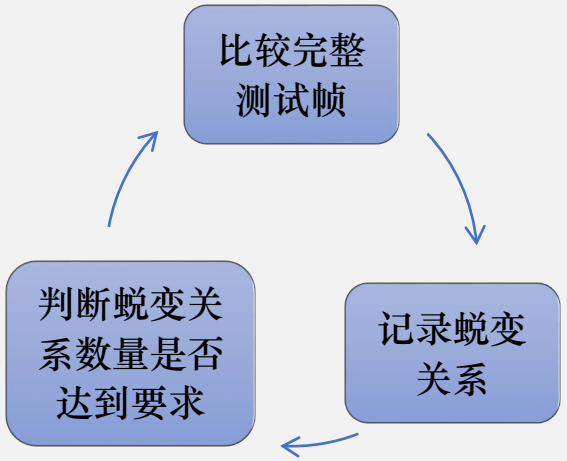
  利用METRIC技术分别得到184、142、1130和3512个蜕变关系

- METRIC技术的关键步骤

  ① 挑选两个有意义有区别的完整测试帧作为识别蜕变关系的候选对

  ② 用户识别蜕变关系，并记录

  ③ 重复以上步骤直到遍历所有的候选对或者到达预定数目的蜕变关系




TF1: Typy of vehicle$_{motorcycle}$, day of week$_{weekday}$, discount coupon$_{no}$, actual hours$_{(0.0,\ 2.0)}$

TF2: Typy of vehicle$_{motorcycle}$, day of week$_{weekday}$, discount coupon$_{no}$, actual hours$_{(2.0,\ 4.0)}$

MR：摩托车车主在工作日停车，没有提供折扣券，如果停车时间从 (0.0, 2.0)增加到 (2.0, 4.0)，那么停车费用也增加

# 实验设置

➢ 蜕变关系

- grep蜕变关系获取

RES：原始测试用例的正则表达式；IFS：原始测试用例的输入文件；REF：衍生测试用例的正则表达式；IFF：衍生测试用例的输入文件；O1：原始测试的输出；O2：衍生测试用例的输出

MR1：REF将RES的范围字符集转化为它的等价形式，并且转化后的范围字符集中的元素顺序是随机排列的；IFS与IFF完全相同；如果O1与O2不相同，那么该蜕变关系被违反　RES: [1-3]; REF: [312]

MR2：REF随机地枚举RES范围字符集中的元素，并且在元素之间插入“|”；IFS与IFF完全相同；如果O1与O2不相同，那么该蜕变关系被违反　RES: [1-3]; REF: 2|3|1

MR3：RES是用“[]”封装的简单字符集合，REF将RES中的每个元素用“[]”封装，并且元素之间插入“|”；IFS与IFF完全相同；如果O1与O2不相同，那么该蜕变关系被违反　RES: [123]; REF: [1]|[2]|[3]

RES: [1-4]; REF: [1-2]|[3-4]

MR4：REF将RES的范围字符集分割成两个较小的范围字符集，分割后的范围字符集之间插入“|”

# 实验设置

## ➢ 蜕变关系

- grep蜕变关系获取

RES: “123”; REF: [312]

MR5：RES是一个简单字符集合，并且不包含元字符，REF将RES集合中的元素重新排列，并将排列后的字符集合用”[]”封装； IFS与IFF完全相同；如果O2不包含O1，那么该MR被违反

MR6：RES是一个简单字符集合，并且不包含元字符，REF将RES中的元素重新排列，并在元素之间插入”|”；IFS与IFF完全相同；如果O2不包含O1，那么该MR被违反

RES: “123”; REF: “3|1|2“

MR7：RES是一个范围字符集，REF是RES的一个真子集； IFS与IFF完全相同；如果O1不包含O2，那么该MR被违反

RES: “[1-9]”; REF: “[1-4]“

MR8：RES存在一个范围字符集，REF的范围字符集包含RES的范围字符集；IFS与IFF完全相同；如果O2不包含O1，那么该MR被违反

RES: “[1-4]”; REF: “[1-9]“

# 实验设置

## ➢ 蜕变关系

- grep蜕变关系获取

MR9：REF在RES的末尾插入“|”和如下元字符中的任意一个：[[:digit:]]，[^[:digit:]]，\w，and \W；IFF在IFS的基础上增加一行内容，该内容匹配增加的元字符；如果O1和新增的内容不包含在O2中，那么该MR被违反

RES: “123”; REF: “123|\w”

MR10：REF在RES的基础上增加一个表示重复的元字符“+”or{1}， IFF与IFS完全相同；如果O2与O1不相同，那么该MR被违反

RES: “123”; REF: “123{1}”

MR11：RES必须包含成对元字符（例如：\w和\W；[[:digit:]] 和[^[:digit:]]）中的一个，REF将RES中的元字符替换为另一个与之配对的元字符；IFF与IFS完全相同，且包含至少两行内容，这两行内容分别匹配成对的元字符；如果O1与O2相等，或者O1与O2的内容合并之后与IFS不同，那么该MR被违反

RES: “^\w$”; REF: “^\W$”

# 实验设置

- 蜕变关系
  - grep蜕变关系获取

MR12：REF将RES的一个或者多个字符替换成“.”，IFS与IFF完全相同；如果O2不包含O1，那么该MR被违反

RES: “1234”; REF: “1..4”

# 实验设置

## ➢ 蜕变关系

- FastJson蜕变关系获取

分析FastJson的用户手册以及eishay基准系统的输入输出得到如下蜕变关系：

### 蜕变关系

该文档主要记录识别的蜕变关系。为了下文方便描述做以下定义：

SJB：原始测试用例的JavaBean；
FJB：衍生测试用例的JavaBean；
SF：原始测试用例的JSON文件；
FF：衍生测试用例的JSON文件；
SO：原始测试用例反序列化后的对象；
FO：衍生测试用例反序列化后的对象；
O1：原始测试用例的输出；
O2：衍生测试用例的输出。

### 蜕变关系1

如果SJB含有一个空的构造函数，FJB在SJB的基础上去掉构造函数；原始测试用例与衍生测试用例的JSON文件完全相同；如果存在一个SO与FO中名字一样的成员变量具有不同的值，则该蜕变关系被违反。

```
public class SJB{
  public SJB(){}
  ...
}

public calss FJS{
  ...
}
```

### 蜕变关系2

FF通过在SF中每一个键对应值的末尾添加一行注释得到；SJB与FJB完全一致；如果存在一个SO与FO中名字一样的成员变量具有不同的值，该蜕变关系被违反。

```
SF文件内容：
{
    "1": {
        "airClass": 2,
        "area": 0,
        "economicfee": 5153.698093040778,
        "luggage": 22.28105377560367 5,
        "student": false
    },
    "2": {
        "airClass": 3,
        "area": 0,
        "economicfee": 1936.2348923480338,
        "luggage": 48.14329753468175,
        "student": false
    }
}

FF文件内容：
{
    "1": {
        "airClass": 2,/*这是注释*/
        "area": 0,/*这是注释*/
        "economicfee": 5153.698093040778,/*这是注释*/
        "luggage": 22.28105377560367 5,/*这是注释*/
        "student": false/*这是注释*/
    },/*这是注释*/
    "2": {
        "airClass": 3,/*这是注释*/
        "area": 0,/*这是注释*/
        "economicfee": 1936.2348923480338,/*这是注释*/
        "luggage": 48.14329753468175,/*这是注释*/
        "student": false/*这是注释*/
    }/*这是注释*/
}
```

# 实验设置

## ➢ 分区

- 实验室程序-ACMS

随机选取两个范畴，然后将它们的选项进行组合，并删除无效的组合，最后得到分区模式

分区依据：

1. 实际测试过程中不知道故障的具体位置
2. 课题组前期研究表明，APT的故障检测效率与分区数目没有明显的关联

分区范畴：座舱等级和航班类型。

| partition | choices |
|---|---|
| partition0 | I-1a;I-2a;* |
| partition1 | I-1a;I-2b;* |
| partition2 | I-1b;I-2a;* |
| partition3 | I-1b;I-2b;* |
| partition4 | I-1c;I-2a;* |
| partition5 | I-1c;I-2b;* |
| partition6 | I-1d;I-2a;* |
| partition7 | I-1d;I-2b;* |

# 实验设置

## ➢ 分区

- 实验室程序-CUBS

随机选取两个范畴，然后将它们的选项进行组合，并删除无效的组合，最后得到分区模式

分区依据：

1. 实际测试过程中不知道故障的具体位置
2. 课题组前期研究表明，APT的故障检测效率与分区数目没有明显的关联

分区范畴：套餐和通话时间。

| partition | choices |
|---|---|
| partition0 | I-1a;I-3a;* |
| partition1 | I-1a;I-3b;* |
| partition2 | I-1b;I-3a;* |
| partition3 | I-1b;I-3b;* |

# 实验设置

➢ 分区

- 实验室程序-ERS

  随机选取两个范畴，然后将它们的选项进行组合，并删除无效的组合，最后得到分区模式

  分区依据：

  1. 实际测试过程中不知道故障的具体位置
  2. 课题组前期研究表明，APT的故障检测效率与分区数目没有明显的关联

分区范畴：职称和月销售额。

| partition | choices |
|---|---|
| partition0 | I-1a;I-3a;* |
| partition1 | I-1a;I-3b;* |
| partition2 | I-1a;I-3c;* |
| partition3 | I-1a;I-3d;* |
| partition4 | I-1b;I-3a;* |
| partition5 | I-1b;I-3b;* |
| partition6 | I-1b;I-3c;* |
| partition7 | I-1b;I-3d;* |
| partition8 | I-1c;I-3a;* |
| partition9 | I-1c;I-3b;* |
| partition10 | I-1c;I-3c;* |
| partition11 | I-1c;I-3d;* |

# 实验设置

➢ 分区

- 实验室程序-MOS

随机选取两个范畴，然后将它们的选项进行组合，并删除无效的组合，最后得到分区模式

分区依据：

1. 实际测试过程中不知道故障的具体位置
2. 课题组前期研究表明，APT的故障检测效率与分区数目没有明显的关联

分区范畴：飞机模型和是否改变机组成员

| partition | choices |
|---|---|
| partition0 | I-1a;I-2a;* |
| partition1 | I-1a;I-2b;* |
| partition2 | I-1b;I-2a;* |
| partition3 | I-1b;I-2b;* |
| partition4 | I-1c;I-2a;* |
| partition5 | I-1c;I-2b;* |
| partition6 | I-1d;I-2a;* |
| partition7 | I-1d;I-2b;* |
| partition8 | I-1e;I-2a;* |
| partition9 | I-1e;I-2b;* |

# 实验设置

➢ 分区

- grep程序

在实验的过程中，如果将每个完整测试帧作为一个分区，将得到最“细”粒度的划分。AMT是基于分区上的技术，并且需要在一个分区中多次选择测试用例，因此这种分区方式不可行，需要“粗”粒度的分区模式

- grep的范畴分组

| independent category (A) | dependent category (B) |
|---|---|
| NormalChar | Bracket |
| wordSymbol | Iteration |
| DigitSymbol | Parentheses |
| SpaceSymbol | Line |
| NamedSymbol | Word |
| AnyChar | Edge |
| Range | Combine |

1) Category1: NA; NP
2) Category2: YW;NW
3) Category3: YD; ND
4) Category4: YS; NS
5) Category5: N1; N2; …, N12
6) Category6: DOT
7) Category7: UR; LR; NR
8) Category8: NB; CB
9) Category9: QM; ST; PL, RM
10) Category10: UR; LR; NR
11) Category11: BL; EL; LL
12) Category12: BW; EW; WW
13) Category13: YB; YE; YY; EN; NE; NN
14) Category14: CO; AL

# 实验设置

## ➢ 分区

- grep粗粒度的分区模式

  忽略一部分范畴，考虑剩下的范畴中选项的组合：

  - 分区模式1：考虑A组范畴，忽略B组范畴
  - 分区模式2：考虑B组范畴，忽略A组范畴

测试帧1：#;NA;YW

测试帧2：#;YW;NA;#

- 分区模式1

  经统计，包含A组选项的测试帧数目为99762。为了让每个分区中的平均测试用例数目超过两个，基于包含相同选项的测试帧具有相似的执行路径这一考虑，忽略测试帧中选项的次序，得到552个“粗粒度的测试帧”

- 分区模式2

  与得到分区模式1的方法相似，对所有的完整测试帧进行处理，最后得到3380个“粗粒度的测试帧”

# 实验设置

## ➢ 测试剖面

将均等概率分布作为初始测试剖面，其依据为：

① 课题组前期的研究表明，APT的故障检测效率与初始测试剖面没有明显的关系

② 在实际的测试活动中测试人员可能对待测软件没有较深的理解，即不能利用测试经验设置初始测试剖面，在这种情况下，我们想探究AMT的故障检测效率

③ AMT具有适应性，可以在测试的过程中根据测试历史调整测试剖面，使得效率大的分区被检测的概率大

第1PPT
HTTP://WWW.1PPT.COM

## 可以在下列情况使用

- 不限次数的用于您个人/公司、企业的商业演示。
- 拷贝模板中的内容用于其它幻灯片母版中使用

## 不可以在以下情况使用

- 用于任何形式的在线付费下载。
- 收集整理我们发布的免费资源后，刻录光碟销售

PPT模板下载：www.1ppt.com/moban/
节日PPT模板：www.1ppt.com/jieri/
PPT背景图片：www.1ppt.com/beijing/
优秀PPT下载：www.1ppt.com/xiazai/
Word教程：www.1ppt.com/word/
资料下载：www.1ppt.com/ziliao/
范文下载：www.1ppt.com/fanwen/
教案下载：www.1ppt.com/jiaoan/

行业PPT模板：www.1ppt.com/hangye/
PPT素材下载：www.1ppt.com/sucai/
PPT图表下载：www.1ppt.com/tubiao/
PPT教程：www.1ppt.com/powerpoint/
Excel教程：www.1ppt.com/excel/
PPT课件下载：www.1ppt.com/kejian/
试卷下载：www.1ppt.com/shiti/
PPT论坛：www.1ppt.cn