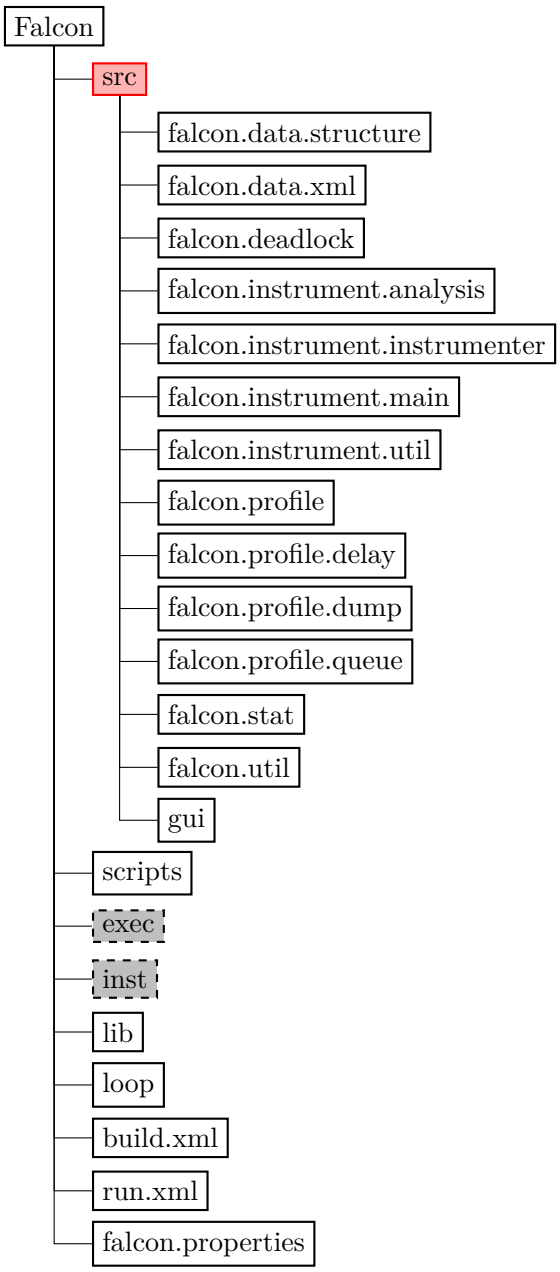
Falcon
src
falcon.data.structure
falcon.data.xml
falcon.deadlock
falcon.instrument.analysis
falcon.instrument.instrumenter
falcon.instrument.main
falcon.instrument.util
falcon.profile
falcon.profile.delay
falcon.profile.dump
falcon.profile.queue
falcon.stat
falcon.util
gui
scripts
exec
inst
lib
loop
build.xml
run.xml
falcon.properties

| | 执行语句序列 | 对 x 的访问序列 | 对 y 的访问序列 |
|---|---|---|---|
| 1 | 1-6-4-2-7-5-3 | $R_{1,1} - W_{1,1} - R_{3,6} - W_{3,6} - R_{2,4} - R_{1,3}$ | $R_{1,2} - W_{1,2} - R_{3,7} - W_{3,7} - R_{2,5} - R$ |
| 2 | 1-6-4-2-5-7-3 | $R_{1,1} - W_{1,1} - R_{3,6} - W_{3,6} - R_{2,4} - R_{1,3}$ | $R_{1,2} - W_{1,2} - R_{2,5} - W_{2,5} - R_{3,7} - R$ |
| 3 | 1-4-6-2-7-5-3 | $R_{1,1} - W_{1,1} - R_{2,4} - W_{2,4} - R_{3,6} - R_{1,3}$ | $R_{1,2} - W_{1,2} - R_{3,7} - W_{3,7} - R_{2,5} - R$ |
| 4 | 1-4-6-2-5-7-3 | $R_{1,1} - W_{1,1} - R_{2,4} - W_{2,4} - R_{3,6} - R_{1,3}$ | $R_{1,2} - W_{1,2} - R_{2,5} - W_{2,5} - R_{3,7} - R$ |

```
Initialize x
Thread1
1: lock(L)
2: write(x)
3: read(x)
4: unlock(L)
Thread2
5: lock(L)
6: read(x)
7: unlock(L)
```

```
x=0; y=0;
Thread1
1: if(x==0) x=1;
2: if(y==0) y=1;
3: if(x==2 and y==2) assert(false);
Thread2
4: if(x==1) x=2;
5: if(y==1) y=2;
Thread3
6: if(x==1) x=3;
7: if(y==1) y=3;
```

$$suspiciousness(s) = \frac{failed(s)}{totalfailed + passed(s)} \tag{1}$$

| | 运行 1 | 运行 2 | 运行 3 | 运行 4 | 可疑度 |
|---|---|---|---|---|---|
| x 的访问模式:$W_1 - W_3 - R_1$ | * | * | | | 0 |
| x 的访问模式:$W_1 - W_2 - R_1$ | | | * | * | 0.5 |
| y 的访问模式:$W_1 - W_3 - R_1$ | * | | * | | 0 |
| y 的访问模式:$W_1 - W_2 - R_1$ | | * | | * | 0.5 |
| 运行结果 | Pass | Pass | Pass | Failed | |

表 1: 冲突的线程交错模式

| | 线程读写 | 描述 |
|---|---|---|
| 1 | $R_1 - W_2$ | 写入意外的数值 |
| 2 | $W_1 - R_2$ | 读出意外的数值 |
| 3 | $W_1 - W_2$ | 线程 1 写入的数值丢失 |

表 2: 非顺序化的线程交错模式

| | 线程读写 | 描述 |
|---|---|---|
| 1 | $R_1 - W_2 - R_1$ | 不可重复读 |
| 2 | $W_1 - W_2 - R_1$ | 线程 1 数据被线程 2 意外修改 |
| 3 | $W_1 - R_2 - W_1$ | 线程 2 读入 “脏” 数据 |
| 4 | $R_1 - W_2 - W_1$ | 丢失修改 |
| 5 | $W_1 - W_2 - W_1$ | 丢失修改 |

**算法 1:** GatherPatterns

**输入**: $m$:shared memory location
$b$:memory access type
$t$:thread ID
$s$:meomory access location
$P_t$:current set of patterns(initially null)

**输出**: $P_t$:updated set of patterns

```
if m does not yet have any window then
    w←createWindow()
    w.insert(b, t, s)
    registerWindow(w, m)
else
    w←getWindow(m)
    (b_2, t_2, s_2)← w.getLastAccess()
    if t = t_2 then
        w.update(b, s)
    else
        if w is full then
            P_t ←getPatterns(w)
            w ←slideWindow(w)
        end
        w.insert(b, t, s)
    end
end
return P_t
```