软件分析技术lab1 实验报告

组员及分工：
张伟 2100013122 算法搜集 代码编写 性能优化
寿晨宸 2100012945 测试样例编写与提交
于佳琨 2100013119 报告编写

项目github链接： https://github.com/En-2863/PointerAnalysis_PKU

1. 算法介绍

算法参考南京大学指针分析算法，基于非上下文敏感的指针分析。

1.1 Solve(m^{entry})

$WL = [], PFG = \{\}, S = \{\}, RM = \{\}, CG = \{\}$
AddReachable(m^{entry})
while WL is not empty do
  remove $< n, pts >$ from WL
  $\Delta = pts – pt(n)$
  Propagate(n, Δ)
  if n represents a variable x then
    foreach $o_i \in \Delta$ do
      foreach $x.f = y \in S$ do
        AddEdge($y, o_i.f$)
      foreach $y = x.f \in S$ do
        AddEdge($o_i.f, y$)
      ProcessCall(x, o_i)

其中：WL代表工作列表， PFG代表指针流图， S代表可达语句集合，S_m代表方法 m 中的语句集合，RM代表可达方法集合，CG代表调用图的边的集合。

1.2 AddEdge(s, t)

if $s \rightarrow t \notin PFG$ then
  add $s \rightarrow t$ to PFG
  if $pt(s)$ is not empty then
    add $< t, pt(s) >$ to WL

AddEdge函数：用于在指针流图中添加边，如果$s \rightarrow t$不在PFG中，添加$s \rightarrow t$， 然后通过检验确保s指向的所有object被t指向。

1.3 Propagate(n, pts)

if pts is not empty then
  $pt(n) \bigcup = pts$
  foreach $n \rightarrow s \in PFG$ do
    add<s, pts> to WL

Propagate函数：将pts加入到n的points-to 关系集合中<,并将pts传播给n的后继结点.

1.4 AddReachable(m)

if $m \notin RM$ then
  add m to RM
  $S \cup = S_m$
  foreach $i : x = new \ \ T() \in S_m$ do
    add $< x, \{o_i\} >$ to WL
  foreach $x = y \in S_m$ do
    AddEdge(y, x)

AddReachable函数: 根据method m添加新的reachable method and statements,首先判断m是否在RM中，如果需要补充，则将m添加到RM中，并将method m中的语句添加到可达语句集合S中,接下来对于S_m中的new语句进行处理，将对应的$< x, \{o_i\} >$加入工作集,最后处理赋值语句，为赋值语句添加相应的边。

1.5 ProcessCall(x, o_i)

foreach $l : r = x.k(a1, \ldots, an) \in S$ do
  m = Dispatch(o_i, k)
  add $< m_{this}, o_i >$ to WL
  if $l \rightarrow m \notin CG$ then
    add $l \rightarrow m$ to CG,
    AddReachable(m)
    foreach parameter p_i of m do
      AddEdge(a_i, p_i)
    AddEdge(m_{ret}, r)

ProcessCall函数：在实例内部对于每个调用语句进行相应操作，首先，使用 Dispatch 函数解析虚拟分派，确定调用 k在 o_i 上的目标方法m，然后将目标方法m和对应的o_i添加到工作列表WL中，接下来判断 $l \rightarrow m$ 是否存在，如果不存在，将调用边 $l \rightarrow m$ 添加到调用图CG中，之后使用AddReachable函数根据method m添加新的reachable method and statements，最后对于m的每个参数p_i以及m_{ret}进行加边操作。

1.6 算法流程如下：

对于m^{entry},先利用AddReachable现将其设置为可达状态
在工作列表不为空的情况下，从工作列表中移除指针n及其的一个实例pts,<n, pts>
对指针n进行传播处理，传播$\Delta = pts - pt(n)$
如果n代表的是变量x
则对于 Δ 中的每个实例 o_i 进行以下操作：
对于每个 $x.f = y$ 的语句，添加从 y 到 $o_i.f$ 的边;
对于每个 $y = x.f$ 的语句，添加从 $o_i.f$ 到 y 的边;
调用 ProcessCall(x, o_i) 处理函数调用

1.7 针对静态字段、数组索引和静态方法调用，引入新的指针分析规则来处理

1.7.1 静态字段：

处理方法：在静态字段和变量之间传值

$Static\ \ Store$:
语句为：$T.f = y$，规则为：$\frac{o_i \in pt(y)}{o_i \in pt(T.f)}$， PFG边的表示为：$T.f \leftarrow y$

$Static\ \ Load$:
语句为：$y = T.f$，规则为：$\frac{o_i \in pt(T.f)}{o_i \in pt(y)}$， PFG边的表示为：$y \leftarrow T.f$

1.7.2 静态索引：

处理方法：

$Array\ \ Store$:
语句为：$x[i] = y$，规则为：$\frac{o_u \in pt(x), o_v \in pt(y)}{o_v \in pt(o_u[*])}$， PFG边的表示为：$o_u[*] \leftarrow y$

$Array\ \ Load$:
语句为：$y = x[i]$，规则为：$\frac{o_u \in pt(x), o_v \in pt(o_u[*])}{o_v \in pt(y)}$， PFG边的表示为：$y \leftarrow o_u[*]$

1.7.3 静态方法：

处理方法：

$Static\ \ Call$:
语句为：$r = T.m(a1, ..., an)$，规则为：$\frac{o_u \in pt(a_j), 1 \le j \le n, o_v \in pt(m_{ret})}{o_u \in pt(m_{pj}), 1 \le j \le n, o_v \in pt(r)}$,
PFG边的表示为：$a_1 \rightarrow m_{p1}\ \ \ldots\ a_n \rightarrow m_{pn}\ \ ,\ r \rightarrow m_{ret}$

2. 参考文献

https://cs.nju.edu.cn/tiantan/software-analysis/PTA-FD.pdf
https://tai-e.pascal-lab.net/pa5.html
https://blog.csdn.net/m0_53632564/article/details/127255320