HW7 报告

胥昊天

2025-05-08 19:16:30

算法原理

将四元式转换为 SSA 形式主要包括以下步骤：

1. **消除不可达代码**：删除控制流图中不可达的基本块，简化后续分析
2. **计算支配关系**：构建支配树和计算支配边界
3. **放置 Phi 函数**：在变量的支配边界处插入 Phi 函数
4. **变量重命名**：为每个变量的不同定义分配唯一版本
5. **清理无用 Phi 函数**：删除不必要的 Phi 函数

支配边界与 Phi 函数放置

对于每个变量 v，如果它在某个基本块 B 中被定义，那么在 B 的支配边界 DF(B) 的每个节点中，可能需要放置一个 Phi 函数。具体来说，当以下条件满足时在支配边界 F 中放置 Phi 函数：

1. 变量 v 在该支配边界 F 上还没有 Phi 函数
2. 变量 v 在 F 入口处是活跃的（控制流离开 F 后还会使用该变量）

```
for (int F : domInfo->dominanceFrontiers[block->entry_label->num]) {
    auto F_block = domInfo->labelToBlock[F];
    const auto& F_liveout = dataFlowInfo.liveout->at(*F_block->quadlist->begin());
    if (already_phi[F].find(var) == already_phi[F].end() &&
        F_liveout.find(var) != F_liveout.end()) {
        // 插入 Phi 函数
        auto phi = new QuadPhi(...);
        domInfo->labelToBlock[F]->quadlist->insert(..., phi);
        already_phi[F].insert(var);
    }
}
```

变量重命名

变量重命名是通过递归地遍历支配树来实现的。对于每个基本块：

1. 处理当前块中的所有语句
    - 对于使用的变量，替换为当前版本（栈顶）
    - 对于定义的变量，创建新版本并更新栈
2. 更新后继块中的 Phi 函数参数
3. 递归处理支配树中的子节点
4. 恢复栈状态（弹出在当前块中定义的变量版本）

```
// 为每个变量维护一个版本栈
map<int, stack<Temp*>> stacks;
map<int, int> counter;

// 递归函数：重命名变量
std::function<void(int)> rename = [&](int blockLabel) {
    // 处理当前块中的语句...

    // 更新后继块中的 Phi 函数...

    // 递归处理子块...

    // 恢复栈状态...
};
```

代码细节

详见代码仓库。

git graph

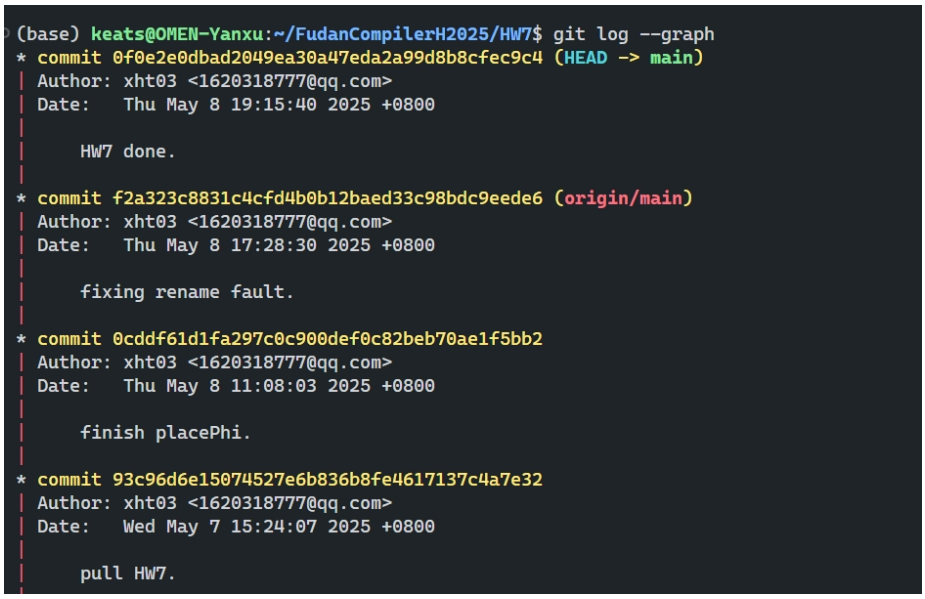

Figure 1: alt text