

Sparse Analysis and Path Conditions Transform

枫聆

2022 年 1 月 9 日

目录

1 The Definition of Sparse Analysis	2
-------------------------------------	---

The Definition of Sparse Analysis

Definition 1.1. 定义 simple language 如下

Program P :=	$F +$	
Function F :=	$f(v_1, v_2, \dots) = \{S; \}$	
	$ f(v_1, v_2, \dots) = \emptyset$	
Statement S :=	$v_1 = \langle v_1 \rangle$:: identity
	$ v_1 = v_2$:: assignment
	$ v_1 = v_1 \oplus v_2$:: binary
	$ v_1 = \text{ite}(v_2, v_3, v_4)$:: if-then-else
	$ v_1 = f(v_2, v_3, \dots)$:: call
	$ \text{return } v_1 = v_2$:: return
	$ \text{if } (v_1 = v_2) \{S_1; \}$:: branching
	$ S_1; S_2$:: sequencing

其中 $\text{ite}(v_2, v_3, v_4)$ 是一个三元表达式. 每个 function 都只有一个 return statement 作为它的唯一 exit node; function 的开头都对应的 identity function 对每一个 function parameter 进行初始化.

Annotation 1.2. 为了方便说明, 规定后面提到的所有 CFG 都是以 statement 为结点, 除了 branch statement $\text{if } (c) \{S\}$, 我们通常将其条件判断作为一个 test node $\text{test}(c)$, 而其 body S 正常展开.

Definition 1.3. 给定 program P 上的两个 statements x, y 和一个 branch condition z . 若 x 中使用到了 y 中定义的变量, 则称 x **数据依赖**(data-dependent) 于 y ; 若当 z 可达且值为 true 时 x 会被执行, 则称 x **控制依赖**(control-dependent) 于 z . 特别地, 关于数据依赖可进一步推广至位于 statement 上的 variables 之间.

Annotation 1.4. 控制依赖更加严格的定义应该是这样: 给定 CFG 上两个不同的结点 x, y , 若满足下述条件:

- 在 CFG 存在一条从 x 到 y 的 nonempty path p 满足对任意的 $v \in p$ 且 $v \neq x$ 都有 $y \text{ !pdom } v$, 其中 pdom 表示 postdominate;
- $y \text{ !pdom } x$.

则称 y 控制依赖于 x . 简而言之**存在某个 x 的后继 $x.\text{succ}_i$ 使得 $y \text{ pdom } x.\text{succ}_i$, 但 $y \text{ !pdom } x$.**

注意往常我们定义 dominate tree 都是以基本块为单位, 这里直接是 CFG 上的结点, 所以你要推广一下: 一个基本块上的结点根据顺序线性关系两个相邻的结点构成 immediate dominance, 再把基本块之间的支配关系放到原来两个基本块的结尾和开始结点.

Definition 1.5. 给定 simple language 上的 program P , 定义它的 **program dependence graph** $G = (V, E)$ 如下

- 对任意结点 $v \in V$, v 表示 P 上的一个 statement 或者 statement 中某个变量.
- E 包括两种 edges 组成
 - **data dependence edges**: 对任意两个 variables x, y , 若 y 的定义数据依赖于 x , 则 $(x, y) \in E$, 所有这样的 edges 记为 E_d .
 - **control dependence edges**: 对任意 statement x 和 branch condition z , 若 x 控制依赖于 z , 则 $(x, z) \in E$, 所有这样的 edges 记为 E_c .

Definition 1.6. 给定 simple language 上的 program P , 构造 E_d 规则如下

$$\begin{array}{c}
 \frac{v_1 = v_2}{(v_2, v_1) \in E_d} \\
 \\
 \frac{v_1 = v_2 \oplus v_3}{(v_2, v_1), (v_3, v_1) \in E_d} \\
 \\
 \frac{v_1 = ite(v_2, v_3, v_4)}{(v_2, v_1), (v_3, v_1), (v_4, v_1) \in E_d} \\
 \\
 \frac{v_1 = f(u_1, \dots) = \{u_1 = \langle u_1 \rangle; \dots; \text{return } w_1 = w_2\}}{(v_2, u_1), (w_2, w_1), (w_1, v_1) \in E_d} \\
 \\
 \frac{v_1 = f(v_2, \dots) \quad f(u_1, \dots) = \emptyset}{(v_2, v_1) \in E_d} \\
 \\
 \frac{\text{if } (v_1 = v_2) \{ \dots \}}{(v_2, v_1) \in E_d}
 \end{array}$$

即对 P 上每一个 statements 都应用上述规则. 根据算法1构造 E_c .

Annotation 1.7. 算法1用到了一个 lemma: 给定 CFG 上两个结点 x, y , 若 y pdom x , 则在 RCFG 上有 y dom x .

Definition 1.8. 给定 program P 的 CFG 上一条 control flow path $p = \{s_0, s_1, \dots, s_n\}$, 设 P 的 program dependence graph 为 $G = (V, E)$. 设关注的数据 facts 为 D , s_i 上 data values along p 为 $\text{in}_{s_i}, \text{out}_{s_i} \subseteq D$, 其中除了 $\text{in}_{s_0} = I \subseteq D$ 其它 data values 都为 \emptyset , s_i 的 transfer function 为 tr_{s_i} . 那么 $\forall s_i \in p, i = 0, 1, \dots, n$,

$$\begin{array}{c}
 \text{out}_{s_i} = \text{tr}_{s_i}(\text{in}_{s_i}) \\
 \text{and } \forall (s_i, s_j) \in E_d, \text{in}_{s_j} = \text{in}_{s_i} \cup \text{out}_{s_i}.
 \end{array}$$

上述分析手法称为 **sparse analysis**. 同时设任意 $s_i \rightarrow s_j$ 之间的 path condition 为 $\phi(s_i, s_j)$, $\text{in}_{s_i}, \text{out}_{s_i} \subseteq D \times \phi(\pi)$, $\text{in}_{s_0} = (I, \phi(s_0, s_0))$. 那么

Algorithm 1: Control Depenence

input : The reverse control flow graph RCFG and the dominance frontier RDF of every every node in RCFG

output The set $CD(X)$ of every node X that are control dependent on X

:

1 **begin**

2 **for** each node $X \in \text{RCFG}$ **do**

3 $CD(X) = \emptyset$

4 **for** each node $X \in \text{RCFG}$ **do**

5 **for** each node $Y \in \text{RCFG}$ **do**

6 Insert Y into $CD(X)$
