软件分析

数据流分析：框架和扩展

熊英飞
北京大学

数据流分析框架

动机

- 上一节我们见到了四种具体的数据流分析
- 可以看出四种分析都有一个类似的形式
  - 能否统一放在一个框架中?
- 如何论证终止和合流?

数据流分析单调框架

- 数据流分析单调框架：对前面所述算法以及所有同类算法的一个通用框架
- 目标：通过配置框架的参数，可以导出各种类型的算法，并保证算法的安全性、终止性、合流性
- 为保证收敛性
  - 需要对抽象域的值加以限定
  - 需要对转换函数加以限定

半格（semilattice）

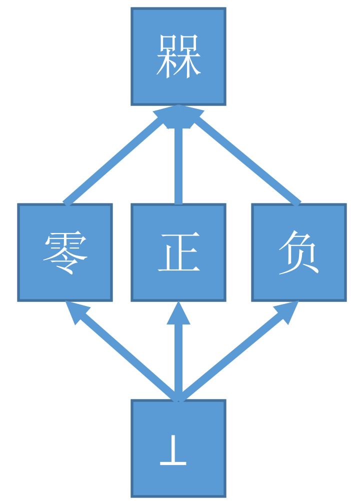
- 半格是一个二元组(S,⊔)，其中S是一个集合，⊔ : $S \times S \rightarrow S$是一个合并运算，并且任意$x, y, z \in S$都满足下列条件：
  - 幂等性idempotence: $x \sqcup x = x$
  - 交换性commutativity: $x \sqcup y = y \sqcup x$
  - 结合性associativity：$(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
- 有界半格是存在最小元⊥的半格，满足
  - $x \sqcup \perp = x$

偏序Partial Order

- 偏序是一个二元组(S, ⊑)，其中S是一个集合，⊑是一个定义在S上的二元关系，并且满足如下性质：
  - 自反性：$\forall a \in S: a \sqsubseteq a$
  - 传递性：$\forall x, y, z \in S: x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$
  - 非对称性：$x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$
- 每个半格都定义了一个偏序关系
  - $x \sqsubseteq y$当且仅当$x \sqcup y = y$

有界半格示例

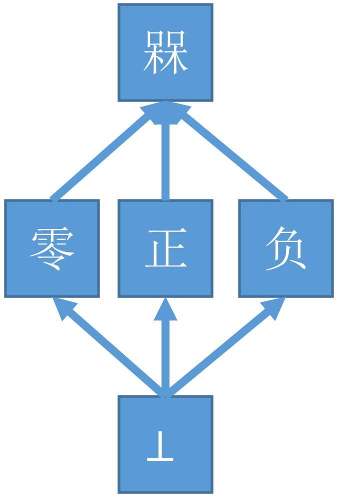
- 抽象符号域的五个元素和合并操作组成了一个有界半格
- 有界半格的笛卡尔乘积$(S \times T, \sqcup_{xy})$还是有界半格
  - $(s_1, t_1) \sqcup_{xy} (s_2, t_2) = (s_1 \sqcup_x s_2, t_1 \sqcup_y t_2)$
- 任意集合和并集操作组成了一个有界半格
  - 偏序关系为子集关系
  - 最小元为空集
- 任意集合和交集操作组成了一个有界半格
  - 偏序关系为超集关系
  - 最小元为全集



半格的高度

- 半格的偏序图中任意两个节点的最大距离+1

- 示例：
  - 抽象符号域的半格高度为3
  - 集合和交集/并集组成的半格高度为集合大小+1
    - 活跃变量分析中半格高度为变量总数+1



练习

- 已知半格 (S,$\sqcup_s$) 和半格 (T,$\sqcup_T$) 的高度分别是x和y，求半格($S \times T$,$\sqcup_{ST}$)的高度
  - $(s_1, t_1) \sqcup_{ST} (s_2, t_2) = (s_1 \sqcup_S s_2, t_1 \sqcup_T t_2)$

- 答案： x+y-1

单调（递增）函数 Monotone (Increasing) Function

- 给定一个偏序关系$(S, \sqsubseteq)$，称一个定义在S上的函数f为单调函数，当且仅当对任意$a, b \in S$满足
  - $a \sqsubseteq b \Rightarrow f(a) \sqsubseteq f(b)$
  - 注意：单调不等于a $\sqsubseteq f(a)$

- 单调函数示例
  - 在符号分析的有界半格中，固定任一输入参数，抽象符号的四个操作均为单调函数
  - 在集合和交/并操作构成的有界半格中，给定任意两个集合GEN, KILL，函数$f(S) = (S - KILL) \cup \mathrm{GEN}$为单调函数

练习

- 以下函数是否是单调递增/递减的：
  - $f(x) = x - 1$
  - 定义域为实数，处处可导，且导数各处不为0的函数
  - 求集合x的补集
  - $f(x) = g \circ h(x)$，已知g和h是单调的
  - $f(x, y) = (g(x), h(y))$，已知g和h是单调的
    - 定义域看做由(x,y)组成的对
  - $f(x, y) = x \sqcup y$，已知$x \in S, y \in S, (S,\sqcup)$是和偏序关系对应的有界半格
    - 定义域看做由(x,y)组成的对

数据流分析单调框架

- 一个控制流图(V, E)
- 一个有限高度的有界半格$(S, \sqcup, \bot)$
- 一个entry的初值OUT_{entry}
- 一组单调函数，对任意$v \in V - entry$存在一个单调函数f_v

- 注意：对于逆向分析，变换控制流图方向再应用单调框架即可

数据流分析工单(WorkList)算法

$\forall v \in (V - \text{entry}): \text{OUT}_v \leftarrow \perp$
$\text{ToVisit} \leftarrow V - \text{entry}$

$\text{While}(\text{ToVisit.size} > 0)$ {

$v \leftarrow$ ToVisit中任意节点

$\text{ToVisit} \mathrel{-}= v$

$\text{IN}_v \leftarrow \sqcup_{w \in pred(v)} \text{OUT}_w$

$\text{If}\big(\text{OUT}_v \neq f_v(\text{IN}_v)\big)\ \text{ToVisit} \cup= \text{succ}(v)$

$\text{OUT}_v \leftarrow f_v(\text{IN}_v)$

}

数据流分析收敛性

- 不动点：给定一个函数$f: S \to S$，如果$f(x) = x$，则称x是f的一个不动点
- 不动点定理：给定高度有限的有界半格$(S, \sqcup, \bot)$和一个单调函数f，链$\bot, f(\bot), f\left(f(\bot)\right), \ldots$必定在有限步之内收敛于f的最小不动点，即存在非负整数n，使得$f^n(\bot)$是f的最小不动点。
  - 证明：
    - 收敛于f的不动点
      - $\bot \sqsubseteq f(\bot)$，两边应用f，得$f(\bot) \sqsubseteq f\left(f(\bot)\right)$，
      - 应用f，可得$f\left(f(\bot)\right) \sqsubseteq f\left(f\left(f(\bot)\right)\right)$
      - 因此，原链是一个递减链。因为该格高度有限，所以必然存在某个位置前后元素相等，即，到达不动点。
    - 收敛于最小不动点
      - 假设有另一不动点u，则$\bot \sqsubseteq u$，两边反复应用f可证

数据流分析收敛性

- 定义如下轮询函数
  - $\mathrm{F}\left(\mathrm{OUT}_{v_1}, \mathrm{OUT}_{v_2}, \dots, \mathrm{OUT}_{v_n}\right) =$
    $$\begin{aligned}&(\mathrm{f}_{\mathrm{v}_1}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_1)} \mathrm{OUT}_{\mathrm{w}}\right),\\ &\mathrm{f}_{\mathrm{v}_2}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_2)} \mathrm{OUT}_{\mathrm{w}}\right),\\ &\dots,\\ &\mathrm{f}_{\mathrm{v}_\mathrm{n}}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_\mathrm{n})} \mathrm{OUT}_{\mathrm{w}}\right))\end{aligned}$$
- 容易证明，F是单调函数
- 根据不动点定理，反复在$(\bot, \dots, \bot)$上应用F所形成的链必然在有限步内终止，并且收敛于F的最小不动点

数据流分析收敛性

- 现在证明F和工单算法的结果等价
  - 二者主要的区别是工单每次随机选择ToVisit中的节点更新
- 终止性：
  - 令OUT_v^i为迭代第i轮之后的OUT_v值，v为任意节点
  - 现在证明对任意节点v，$\mathrm{OUT}_v^0, \mathrm{OUT}_v^1, \ldots$是一个递增序列，即每次增大或不变
    - 因为$\mathrm{OUT}_v^0 = \bot$，所以有$\mathrm{OUT}_v^0 \sqsubseteq \mathrm{OUT}_v^1$
    - 假设到第k个元素都递增，现在证明$\mathrm{OUT}_v^k \sqsubseteq \mathrm{OUT}_v^{k+1}$
      - 如果K+1轮没有更新v，则显然成立
      - 如果$\mathrm{OUT}_v^k = \bot$，则显然成立
      - 如果$\mathrm{OUT}_v^k \neq \bot$，则必然在某轮j被更新过。那么第j轮更新转换函数和合并操作的输入值都必然小于等于第k轮，同时因为转换函数和合并操作都是单调的，所以有$\mathrm{OUT}_v^k \sqsubseteq \mathrm{OUT}_v^{k+1}$
  - 由于格的高度有限，所以对任意v，OUT_v增大次数有限
  - ToVisit集合只在结果变化的时候才增加，否则减少，所以给定足够长的轮数，必然变为空集

数据流分析收敛性

- 合流性：
  - 令X_i为工单算法第i轮的计算结果$\left(\mathrm{OUT}_{v_1}^{i}, \mathrm{OUT}_{v_2}^{i}, \ldots, \mathrm{OUT}_{v_n}\right)$
  - 令Y_i为在$(I, \perp, \ldots, \perp)$上反复应用F的序列
  - 现在证明对任意i，有$X_i \sqsubseteq Y_i$
    - $X_0 \sqsubseteq Y_0$
    - 假设$X_k \sqsubseteq Y_k$，因为工单只是更新一部分节点值，F对所有节点值进行更新，根据上一页的分析，所以$X_{k+1} \sqsubseteq F(X_k) \sqsubseteq F(Y_k)$
  - 因此，当工单算法最终收敛的时候，收敛的结果⊑F的最小不动点
  - 但由于工单算法收敛的结果也是F的不动点，所以 工单算法收敛结果 = F的最小不动点

数据流分析的安全性

- 数据流分析的输出值满足如下等式
$$\mathrm{OUT_v} = \mathrm{f_v}(\sqcup_{\mathrm{w} \in \mathrm{pred(v)}} \mathrm{OUT_w})$$
- 如果$\mathrm{f_v}$保证单步转换的安全性， $\sqcup$保证合并的安全性，则分析整体安全
- 以上两者的安全性论证方式将之后结合抽象解释理论介绍

数据流分析的分配性

- 一个数据流分析满足分配性，如果
  - $\forall v \in V, x, y \in S: f_v(x) \sqcup f_v(y) = f_v(x \sqcup y)$
- 即：近似方案4（提前合并）不会引入不精确性
- 例：符号分析中的结点转换函数不满足分配性
  - 为什么?
  - 令f_v等于“加1”，$f_v(正) \sqcup f_v(零)$

数据流分析的分配性

- 例：在集合和交/并操作构成的有界半格中，给定任意两个集合GEN, KILL，函数$f(\mathrm{OUT}) = (\mathrm{OUT} - KILL) \cup \mathrm{GEN}$满足分配性
  - $f(x) \cup f(y) = (x - K) \cup G \cup (y - K) \cup G = (x - K) \cup (y - K) \cup G = (x \cup y - K) \cup G = f(x \cup y)$
  - $f(x) \cap f(y) = \big((x - K) \cup G\big) \cap \big((y - K) \cup G\big) = ((x - K) \cap (y - K)) \cup G = (x \cap y - K) \cup G = f(x \cap y)$

复习：设计数据流分析

- 近似方案1：抽象状态代表程序的多个具体执行
  - 设计抽象域，对应的α、γ函数和初始值
- 近似方案2：针对控制流节点编写转换函数
  - 设计从基本语句导出转换函数的方法
- 近似方案3：在控制流路径分叉时，复制抽象状态到所有分支
  - 设计从条件导出压缩函数的方法（之后介绍）
- 近似方案4：在控制流路径合并时，用⊔操作合并多个抽象状态
  - 设计⊔操作

设计数据流分析（细化版）

- 近似方案1：抽象状态代表程序的多个具体状态
  - 设计抽象状态（=节点附加值）集合和入口初值
  - 需要和⊔操作构成高度有限的半格
  - 需要存在保证分析正确性的α、γ函数
- 近似方案2：针对控制流节点编写转换函数
  - 设计从基本语句导出转换函数的方法
  - 需要保证转换函数为单调函数
  - 需要保证分析正确性
- 近似方案3：在控制流路径分叉时，复制抽象状态到所有分支
  - 设计从条件导出压缩函数的方法（之后介绍）
- 近似方案4：在控制流路径合并时，用⊔操作合并多个抽象状态
  - 设计⊔操作
  - 需要和抽象状态集合构成高度有限的半格

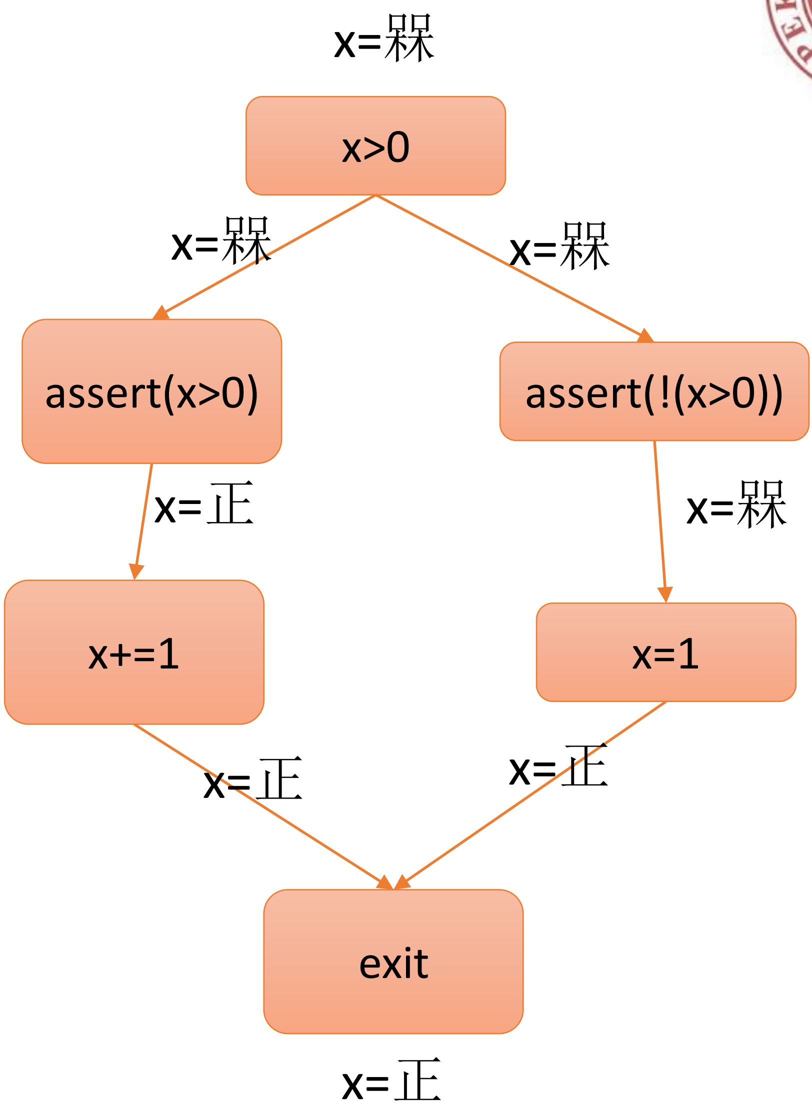
如何设计节点转换函数?

- 节点代码可能包含复杂表达式x:=x+1-y
- 如何从节点代码得到转换函数?
- 方法1：考虑表达式求值的语义，对应定义抽象语义并证明安全性
  - $Eval[e_1+e_2](m)=Eval[e_1](m)+Eval[e_2](m)$
  - 符号分析$[e_1+e_2]$(甲) = 符号分析$[e_1]$(甲) $\oplus$ 符号分析$[e_2]$ (甲)
  - 可用表达式$[e_1+e_2]=\{e_1+e_2\}\cup$ 可用表达式$[\mathrm{e}_1]\cup$ 可用表达式$[\mathrm{e}_2]$
- 方法2：转成三地址码，只处理每一条指令

扩展：条件压缩函数

条件压缩函数

- 近似方案3：在控制流路径分叉时，复制抽象状态到所有分支
  - 每个具体状态只能到达一个分支，形成不精确
- 在每个分支添加条件压缩函数节点，根据条件压缩抽象值



如何设计条件压缩函数?

- 设计反向执行语义：给定输出的抽象值，计算输入的抽象值
  - 整数采用符号抽象，布尔值采用$\{\bot, 真, 假, 值\}$，其中$\gamma(值) = \{\mathrm{true}, \mathrm{false}\}$
  - 反向$[\wedge](\bot) = (\bot, \bot)$
  - 反向$[\wedge](真) = (真, 真)$
  - 反向$[\wedge](假) = (值, 值)$
  - 反向$[\wedge](值) = (值, 值)$
  - 反向$[> 0](\bot) = (\bot)$
  - 反向$[> 0](真) = (正)$
  - 反向$[> 0](其他) = (槑)$
  - 反向$[*](\bot) = (\bot, \bot)$
  - 反向$[*](其他) = (槑, 槑)$
- 根据反向执行语义计算出变量的抽象值，然后和原来的值求交
  - 需要在抽象域上定义求交操作

更精确的反向执行语义

- 参考输入的反向执行语义：给定输入输出的抽象值，压缩输入的抽象值
  - 反向[*](甲, 乙, $\perp$) = ($\perp$, $\perp$)
  - 反向[*](正, 甲, 正) = (正, 甲 $\sqcap$ 正)
  - 反向[*](负, 甲, 正) = (负, 甲 $\sqcap$ 负)
  - 反向[*](零, 甲, 正) = ($\perp$, $\perp$)
  - 反向[*](槑, 正, 正) = (正, 正)
  - ……
  - 反向[>](甲, 乙, $\perp$) = ($\perp$, $\perp$)
  - 反向[>](负, 甲, 真) = (负, 甲 $\sqcap$ 负)
  - 反向[>](零, 甲, 真) = (零, 甲 $\sqcap$ 负)
  - ……
- 首先采用正向语义计算出表达式的值，然后再用反向语义压缩变量的值

正向反向迭代

- 反向压缩变量的值之后，再进行一次正向反向流程可能得到更精确的值
  - $x > 0 \land y > x \land z > y$
  - 假设一开始x, y, z的值都是槑
  - 第一轮得到$\{x \rightarrow 正, y \rightarrow 槑, z \rightarrow 槑\}$
  - 第二轮得到$\{x \rightarrow 正, y \rightarrow 正, z \rightarrow 槑\}$
  - 第三轮得到$\{x \rightarrow 正, y \rightarrow 正, z \rightarrow 正\}$

- 如果反向语义函数保持单调，并且确保压缩或保持输入值，同时半格高度有限，那么该迭代过程一定收敛。

作业：

- 整数采用区间抽象，布尔值采用{⊥, 真, 假, 值}
  - 整数是数学定义，无上下界
  - 区间抽象：[a, b]表示大于等于a小于等于b的整数集合，其中a和b为任意整数
- 请针对下列操作设计参考输入的反向抽象语义
  - 逻辑与
  - 逻辑非
  - 大于
  - 加法
- 要求尽可能精确
- 同时分析基于你设计的反向语义，对任意表达式是否能保证收敛
  - 表达式中可包含变量、常量和以上四种操作符

参考资料

- 《编译原理》第9章
- Lecture Notes on Static Analysis
  - https://cs.au.dk/~amoeller/spa/