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）

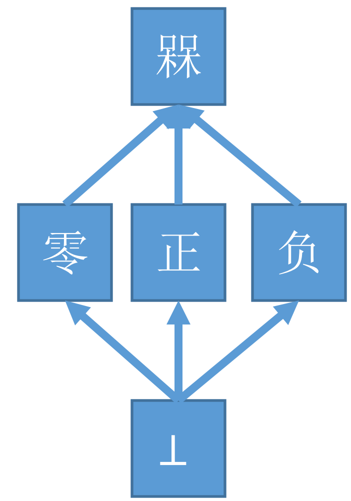
- 半格是一个二元组(S,⊔)，其中S是一个集合，⊔ : $S \times S \rightarrow S$是一个合并运算，并且任意$x, y, z \in S$都满足下列条件：
  - 幂等性idempotence: $x \sqcup x = x$
  - 交换性commutativity: $x \sqcup y = y \sqcup x$
  - 结合性associativity：$(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
- 有界半格是存在最小元⊥的半格，满足
  - $x \sqcup \perp = x$

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：$\forall x, y, z \in S: x \sqsubseteq y \land y \sqsubseteq z \Rightarrow x \sqsubseteq z$
  - 非对称性：$x \sqsubseteq y \land y \sqsubseteq x \Rightarrow x = y$
- 每个半格都定义了一个偏序关系
  - $x \sqsubseteq y$当且仅当$x \sqcup y = y$

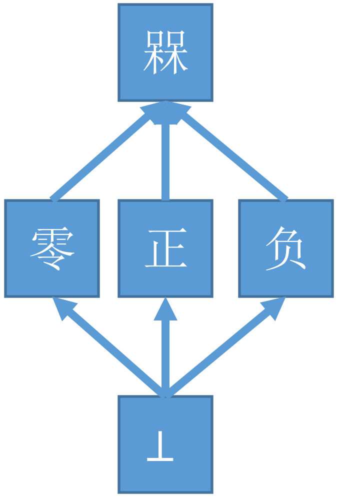
有界半格示例

- 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$f(S) = (S - KILL) \cup$ 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

$\forall v \in (V - entry): OUT_v \leftarrow \perp$
$ToVisit \leftarrow V - entry$
$While(ToVisit.size > 0)$ {
  $v \leftarrow$ ToVisit中任意节点
  $ToVisit$ -= v
  $IN_v \leftarrow \sqcup_{w \in pred(v)} OUT_w$
  $If\left(OUT_v \neq f_v(IN_v)\right) ToVisit \cup= succ(v)$
  $OUT_v \leftarrow f_v(IN_v)$
}

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
  - $\mathrm{F}\left(\mathrm{OUT}_{v_1}, \mathrm{OUT}_{v_2}, \ldots, \mathrm{OUT}_{v_n}\right) =$
$$\begin{aligned}
&(\mathrm{f}_{\mathrm{v}_1}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_1)} \mathrm{OUT}_{\mathrm{w}}\right),\\
&\mathrm{f}_{\mathrm{v}_2}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_2)} \mathrm{OUT}_{\mathrm{w}}\right),\\
&\ldots,\\
&\mathrm{f}_{\mathrm{v}_\mathrm{n}}\left(\sqcup_{\mathrm{w}\in\mathrm{pred}(\mathrm{v}_\mathrm{n})} \mathrm{OUT}_{\mathrm{w}}\right))
\end{aligned}$$
- 容易证明，F是单调函数
- 根据不动点定理，反复在$(\perp, \ldots, \perp)$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$(I, \bot, \ldots, \bot)$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
$$\mathrm{OUT_v} = \mathrm{f_v}(\sqcup_{\mathrm{w \in pred(v)}} \mathrm{OUT_w})$$
- 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：$\sqcup$不会引入可见的不精确性
- 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”，$f_v(\text{正}) \sqcup f_v(\text{零})$

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$\sqcup$操作合并多个抽象状态
  - 设计$\sqcup$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

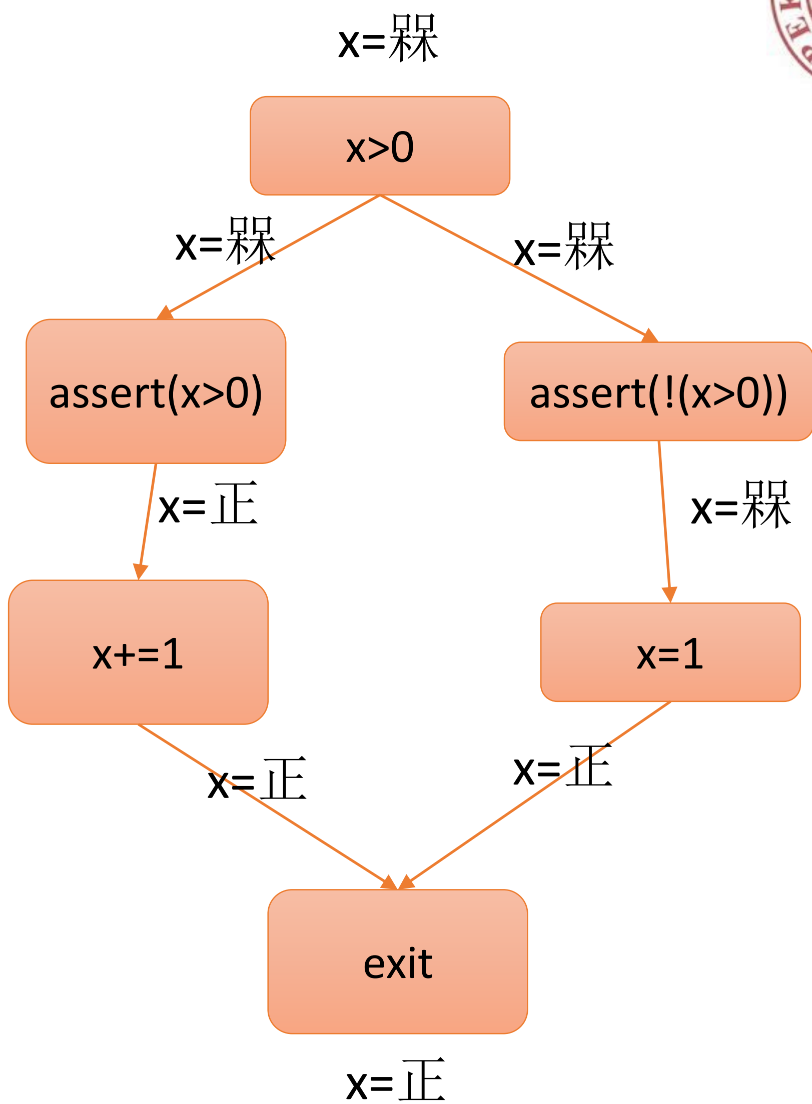
如何设计节点转换函数?

- 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{⊥, 真, 假, 值}，其中γ(值) = {true, false}
  - 反向[∧](⊥) = (⊥, ⊥)
  - 反向[∧](真) = (真, 真)
  - 反向[∧](假) = (值, 值)
  - 反向[∧](值) = (值, 值)
  - 反向[> 0](⊥) = (值)
  - 反向[> 0](真) = (正)
  - 反向[> 0](其他) = (槑)
  - 反向[*](⊥) = (⊥, ⊥)
  - 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, $\bot$) = ($\bot$, $\bot$)
  - 反向[*](正, 甲, 正) = (正, 甲 $\sqcap$ 正)
  - 反向[*](负, 甲, 正) = (负, 甲 $\sqcap$ 负)
  - 反向[*](零, 甲, 正) = ($\bot$, $\bot$)
  - 反向[*](槑, 正, 正) = (正, 正)
  - ……
  - 反向[>](甲, 乙, $\bot$) = ($\bot$, $\bot$)
  - 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
  - $x > 0 \wedge y > x \wedge z > y$
  - 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。

练习：区间（Internval）分析

- 求结果的上界和下界
  - 要求上近似
  - 假设程序中的运算只含有加减运算
  - 例：
    1. a=0;
    2. for(int i=0; i<b; i++)
    3.   a=a+1;
    4. return a;
    - 结果为a:[0,+∞]

区间（Internval）分析

- 正向分析
- 有界半格元素：程序中每个变量的区间，最小元为空集
- 合并操作：区间的并
  - $[a, b] \sqcup [c, d] = [\min(a, c), \max(b, d)]$
- 变换函数:
  - 在区间上执行对应的加减操作
  - $[a, b] + [c, d] = [a + c, b + d]$
  - $[a, b] - [c, d] = [a - d, b - c]$

- 不满足单调框架条件：半格不是有限的
  - 分析可能会不终止

区间分析改进

- 程序中的数字都是有上下界的，假设超过上下界会导致程序崩溃
  - $[a,b]+[c,d]=$
    $\begin{cases} \emptyset & a+c>int\_max \\ (a+c,\min(b+d,int\_max)) & a+c\leq int\_max \end{cases}$

- 原分析终止，但需要int_max步才能收敛

扩展：加宽和变窄

加宽Widening

- 区间分析需要很多步才能达到收敛
  - 格的高度太高
- 加宽：通过降低结果的精度来加快收敛速度
  - 简易加宽：降低格的高度
  - 通用加宽：根据变化趋势快速猜测一个结果

简易加宽

- 定义单调函数w把结果进一步抽象
  - 原始转换函数f
  - 新转换函数$w \circ f$

- 定义有限集合B={-∞, 10, 20, 50, 100, +∞}
- 定义映射函数

$$w([l,h]) = [max\{i \in B \mid i \leq l\}, min\{i \in B \mid h \leq i\}]$$

- 如：
  - $w([15,75]) = [10,100]$

简易加宽的例子

- 令简易加宽的有限集合为$\{-\infty, 0,1,7, +\infty\}$

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

- while(input)处的结果变化

$[x \mapsto\perp, y \mapsto\perp]$	$[x \mapsto\perp, y \mapsto\perp]$
$[x \mapsto [8,8], y \mapsto [0,0]]$	$[x \mapsto [7, \infty], y \mapsto [0,0]]$
$[x \mapsto [8,8], y \mapsto [0,1]]$	$[x \mapsto [7, \infty], y \mapsto [0,1]]$
$[x \mapsto [8,8], y \mapsto [0,2]]$	$[x \mapsto [7, \infty], y \mapsto [0,7]]$
$[x \mapsto [8,8], y \mapsto [0,3]]$	$[x \mapsto [7, \infty], y \mapsto [0, \infty]]$
…	
不使用加宽， 收敛慢或不收敛	使用简易加宽 收敛快，但不精确

简易加宽的安全性

- 如果$\mathrm{x} \sqsubseteq w(x)$，则分析结果保证安全

- 安全性讨论
  - 新转换结果大于等于原结果，意味着$\mathrm{OUT_v}$的结果大于等于原始结果
  - 利用之前类似的方式可以证明最终分析结果一定大于等于原始结果

简易加宽的收敛性

- 如果w是单调函数，则简易加宽收敛
  - 因为$w \circ f$仍然是单调函数

通用加宽

- 更通用的加宽同时参考更新前和更新后的值来猜测最终会收敛的值
  - 原数据流分析算法更新语句：
    - $\mathrm{OUT_v} \leftarrow \mathrm{f_v}(\mathrm{IN_v})$
  - 引入加宽算子∇：
    - $\mathrm{OUT_v} \leftarrow \mathrm{OUT_v}\ \nabla\ \mathrm{f_v}(\mathrm{IN_v})$

- 通用加宽可以实现更快速的收敛，如
  - $[a,b]\nabla \perp = [a,b]$
  - $\perp \nabla [c,d] = [c,d]$
  - $[a,b]\nabla[c,d] = [m,n]$ where
    - $m = \begin{cases} a & c \geq a \\ -\infty & c < a \end{cases}$
    - $n = \begin{cases} b & d \leq b \\ +\infty & d > b \end{cases}$

解读：$\mathrm{x} \in [a,b]$意味着两个约束
- $x \geq a$
- $x \leq b$

该算子本质是去掉循环不保持的约束
这也是一种设计加宽算子的思路

通用加宽的例子

- 令简易加宽的有限集合为$\{-\infty, 0,1,7, +\infty\}$

```
y = 0; x = 7; x = x+1;
while (input) {
   x = 7;
   x = x+1;
   y = y+1;
}
```

- while(input)处的结果变化

$[x \mapsto \bot, y \mapsto \bot]$
$[x \mapsto [8,8], y \mapsto [0,0]]$
$[x \mapsto [8,8], y \mapsto [0,1]]$
$[x \mapsto [8,8], y \mapsto [0,2]]$
$[x \mapsto [8,8], y \mapsto [0,3]]$
…

不使用加宽，
收敛慢或不收敛

$[x \mapsto \bot, y \mapsto \bot]$
$[x \mapsto [7, \infty], y \mapsto [0,0]]$
$[x \mapsto [7, \infty], y \mapsto [0,1]]$
$[x \mapsto [7, \infty], y \mapsto [0,7]]$
$[x \mapsto [7, \infty], y \mapsto [0, \infty]]$

使用简易加宽
收敛快，但不精确

$[x \mapsto \bot, y \mapsto \bot]$
$[x \mapsto [8,8], y \mapsto [0,0]]$
$[x \mapsto [8,8], y \mapsto [0, \infty]]$

使用通用加宽
收敛更快，
结果(恰好)精确

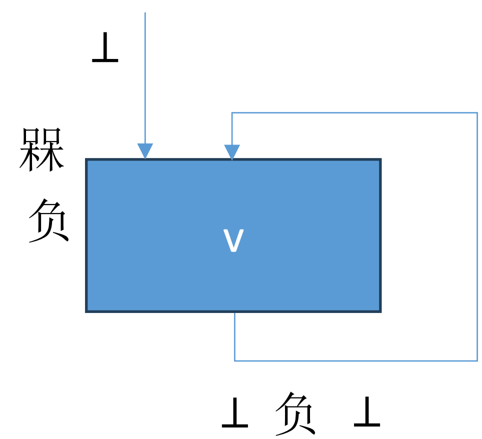
通用加宽的安全性

- 如果$y \sqsubseteq x \nabla y$，则通用加宽的分析结果保证安全性
  - 其他教材上还要求$x \sqsubseteq x \nabla y$，但我感觉不需要

- 定理：加上加宽算子之后，如果分析收敛，则分析结果大于等于轮询函数F的分析结果
  - 定义函数G(X)=X∇F(X)
  - 则原始分析是在⊥ 上不断应用F，而新分析是不断应用G
  - 现在证明对任意i，$F^i(\bot) \sqsubseteq G^i(\bot)$
    - $i = 0$时，显然成立
    - 假设$i = k$时成立，因为F的单调性，那么有$F^{k+1}(\bot) \sqsubseteq F(G^k(\bot))$
    - 又因为∇的性质，我们有$F(G^k(\bot)) \sqsubseteq \mathrm{G}^{\mathrm{k}}(\bot) \nabla\, F(G^k(\bot)) = \mathrm{G}^{\mathrm{k+1}}(\bot)$
    - 即$i = k + 1$时也成立
  - 给定足够大的i，使F和G都到不动点，仍有$F^i(\bot) \sqsubseteq G^i(\bot)$

讨论：$x \sqsubseteq x\nabla y$的作用？

- 如果没有$x \sqsubseteq x\nabla y$的性质，则意味着下一轮的值可以比上一轮更小，可能导致震荡不终止
- 考虑符号抽象域和如下加宽算子
  - $x\nabla \mathrm{y} = \begin{cases} 槑 & y = \perp \\ y & otherwise \end{cases}$
  - $f_v(甲) = \begin{cases} 负 & 甲 = 槑 \\ \perp & otherwise \end{cases}$

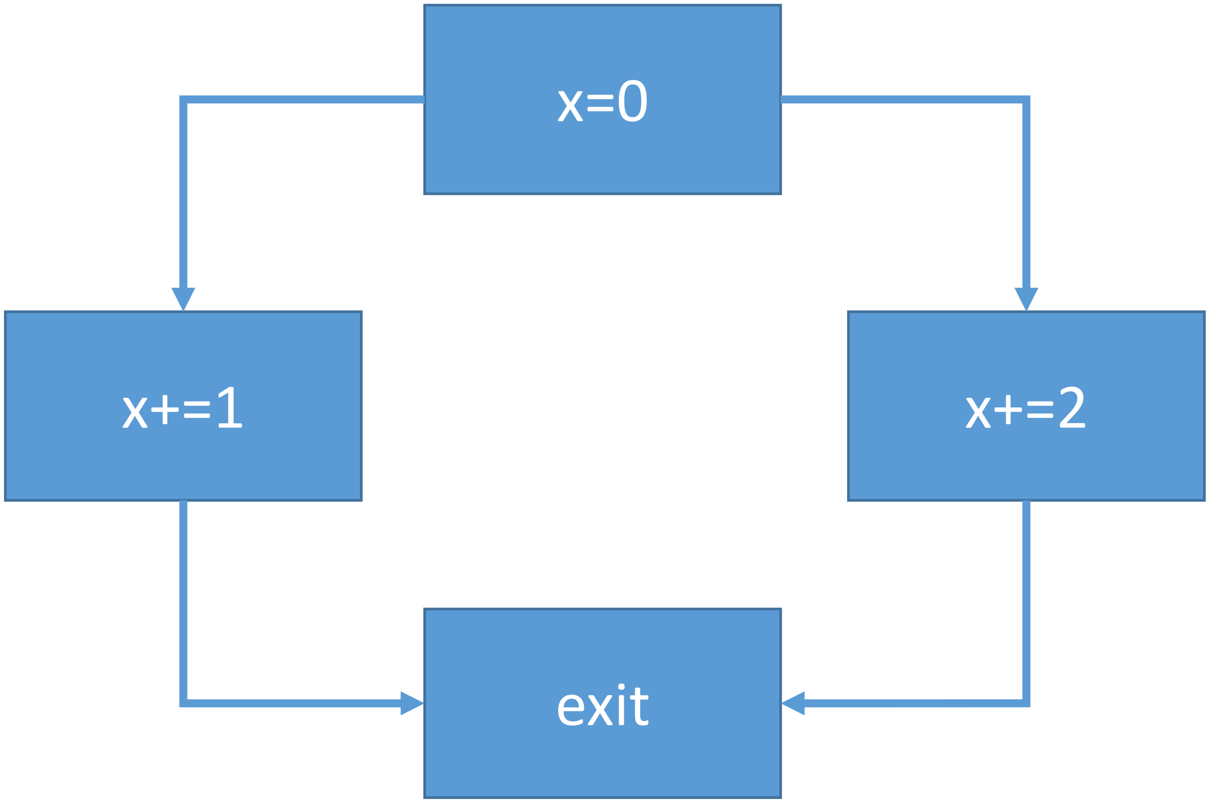


通用加宽的收敛性

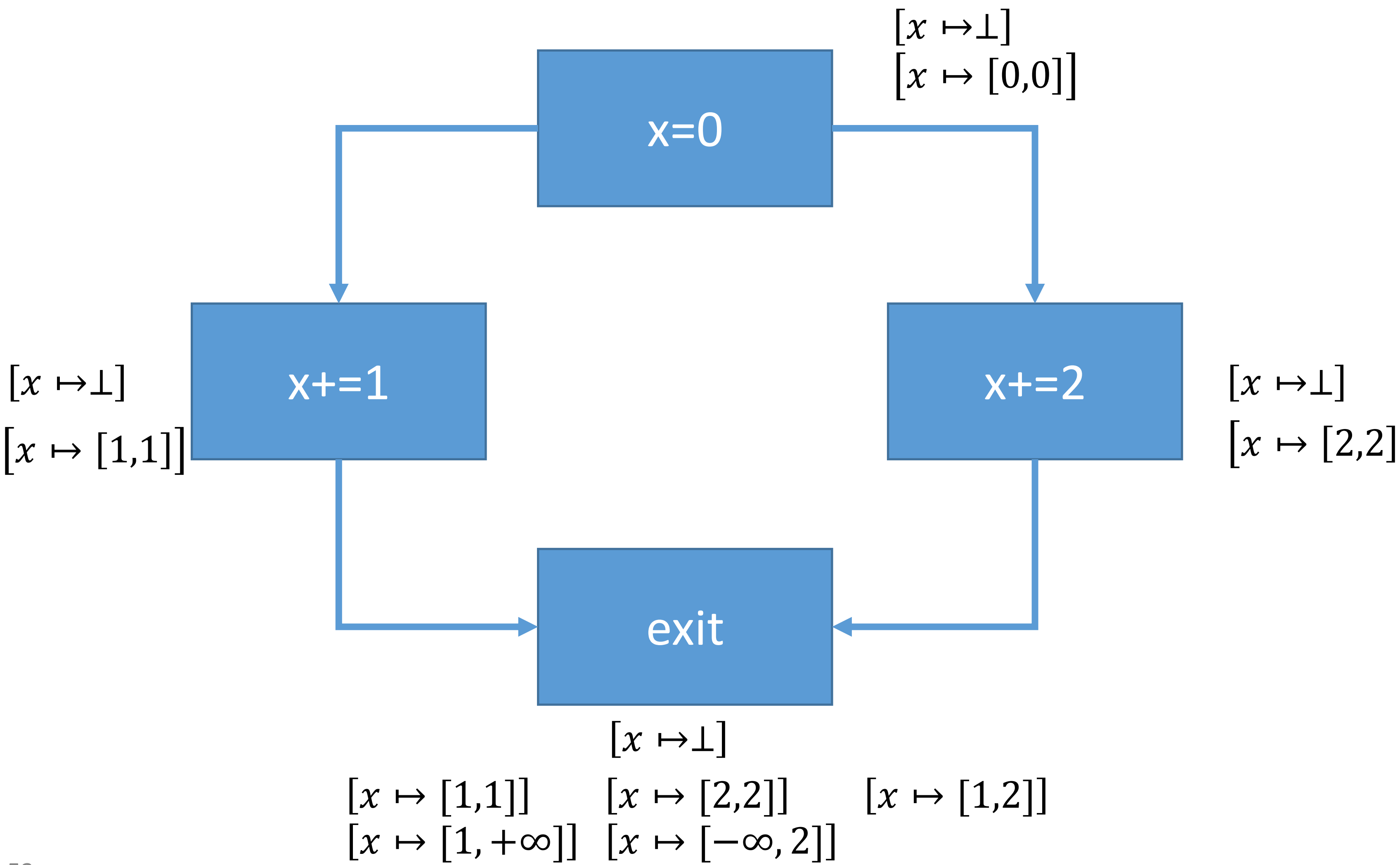
- 目前没有找到容易判断的属性来证明通用加宽的收敛性
  - 加宽算子本身通常不保证变换函数单调递增
  - $[1,1]\nabla[1,2]=[1,\infty]$
  - $[1,2]\nabla[1,2]=[1,2]$

  - 能否给出一个区间分析上不收敛的例子?

加宽不收敛的例子



加宽不收敛的例子



x=0
$[x \mapsto \perp]$
$[x \mapsto [0,0]]$
x+=1
$[x \mapsto \perp]$
$[x \mapsto [1,1]]$
x+=2
$[x \mapsto \perp]$
$[x \mapsto [2,2]]$
exit
$[x \mapsto \perp]$
$[x \mapsto [1,1]]$
$[x \mapsto [2,2]]$
$[x \mapsto [1,2]]$
$[x \mapsto [1,+\infty]]$
$[x \mapsto [-\infty,2]]$

通用加宽的终止性

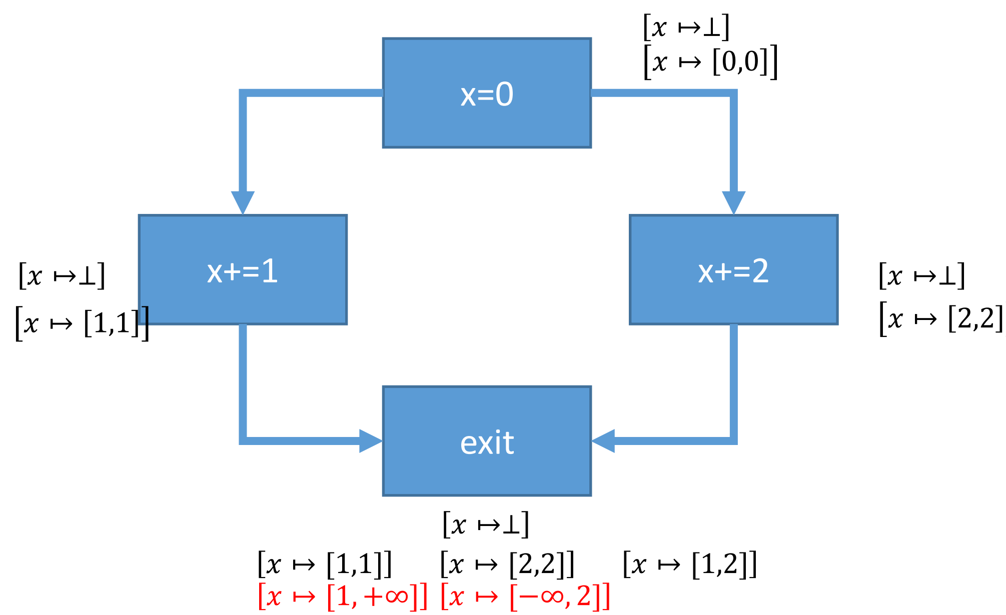
- 相对合流，终止性更加重要
- 但同样，目前也没有找到容易判断的属性来证明通用加宽的终止性

- 多数教材要求下面的属性来保证终止
  - 对任意抽象域上的无穷序列$x_0, x_1, \dots$
  - 如下序列将在有限长度内稳定，即存在k，令$y_k = y_{k+1}$
    - $y_0 = x_0$
    - $y_{k+1} = y_k \nabla x_k$
- 显然该性质保证终止性，但形式和终止性定义一致，要求却强了很多，基本只会增加证明难度

如何拯救加宽带来的不精确?

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \mapsto \perp, y \mapsto \perp]$
$[x \mapsto [7,\infty], y \mapsto [0,0]]$
$[x \mapsto [7,\infty], y \mapsto [0,1]]$
$[x \mapsto [7,\infty], y \mapsto [0,7]]$
$[x \mapsto \boxed{[7,\infty]}, y \mapsto [0,\infty]]$



如何拯救加宽带来的不精确?

- 对于后者，可以只在循环入口点添加加宽算子
- 但识别这样的位置比较麻烦
- 同时，循环入口点仍然可能产生不精确

```
y = 0; x = 7; x = x+1;
while (input) {
   x = 8;
   x = x+1;
   y = y+1;
}
```

采用通用加宽，x的值可能因为更新顺序不同加宽到+∞或−∞

变窄Narrowing

- 通过再次应用原始转换函数对加宽的结果进行修正

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \mapsto [7,\infty], y \mapsto [0,0]]$
$[x \mapsto [7,\infty], y \mapsto [0,\infty]]$
$[x \mapsto [7,7], y \mapsto [0,\infty]]$
$[x \mapsto [7,\infty], y \mapsto [0,\infty]]$
$[x \mapsto [7,\infty], y \mapsto [1,\infty]]$

加宽收敛之后的结果

变窄Narrowing

- 通过再次应用原始转换函数对加宽的结果进行修正

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \mapsto [8,8], y \mapsto [0,0]]$
$[x \mapsto [7,\infty], y \mapsto [0,\infty]]$
$[x \mapsto [7,7], y \mapsto [0,\infty]]$
$[x \mapsto [8,8], y \mapsto [0,\infty]]$
$[x \mapsto [7,\infty], y \mapsto [1,\infty]]$

应用一遍F

变窄Narrowing

- 通过再次应用原始转换函数对加宽的结果进行修正

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \mapsto [8,8], y \mapsto [0,0]]$

$[x \mapsto [7,\infty], y \mapsto [0,\infty]]$

$[x \mapsto [7,7], y \mapsto [0,\infty]]$

$[x \mapsto [8,8], y \mapsto [0,\infty]]$

$[x \mapsto [8,8], y \mapsto [1,\infty]]$

应用两遍F

变窄Narrowing

- 通过再次应用原始转换函数对加宽的结果进行修正

```
y = 0; x = 7; x = x+1;
while (input) {
  x = 7;
  x = x+1;
  y = y+1;
}
```

$[x \mapsto [8,8], y \mapsto [0,0]]$
$[x \mapsto [8,8], y \mapsto [0,\infty]]$
$[x \mapsto [7,7], y \mapsto [0,\infty]]$
$[x \mapsto [8,8], y \mapsto [0,\infty]]$
$[x \mapsto [8,8], y \mapsto [1,\infty]]$

应用三遍F

变窄的安全性

- 针对轮询函数F讨论安全性
- 令
  - 原数据流分析的函数为F，收敛于I_F
  - 经过加宽的函数为G，收敛于I_G
- 那么有
  - 因为 $I_F \sqsubseteq I_G$
  - 所以 $I_F = F(I_F) \sqsubseteq F(I_G) \sqsubseteq G(I_G) = I_G$
  - 即 $I_F \sqsubseteq F(I_G) \sqsubseteq I_G$
- 类似可得
  - $I_F \sqsubseteq F^k(I_G) \sqsubseteq I_G$
- 即变窄保证安全性

变窄的收敛性

- 变窄不保证收敛，也不保证终止
- 通常需要针对应用证明收敛/终止性，或者只应用固定次数F

作业1:

- 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
- 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

作业2：

- 对于下面程序，如果我们在条件分支的地方加上节点根据条件压缩抽象值，采用今天课上讲的加宽算子进行区间分析，每条语句对应的OUT值是什么？如果加上变窄，对应的OUT值是什么？

```
1.  x=1;
2.  while (x < 100) {
3.    x++;}
4.  skip;
```

参考资料

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