软件分析

数据流分析：框架和扩展

熊英飞
北京大学

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、收敛性
- 需要抽象的内容
  - 不同算法在节点上附加的值的类型不同，需要有一个统一接口
  - 不同算法给出的节点转换函数不同，需要有一个统一接口

半格（semilattice）

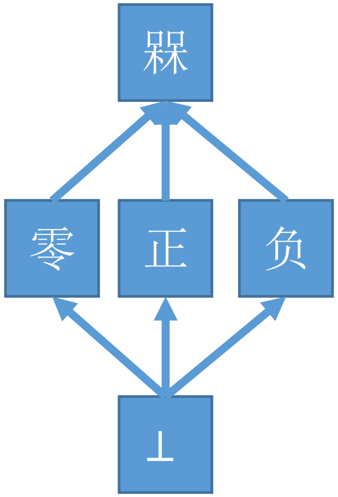
- 半格是一个二元组(S,⊔)，其中S是一个集合，⊔ : $S \times S \to S$是一个合并运算，并且任意$x, y, z \in S$都满足下列条件:
  - 幂等性idempotence: $x \sqcup x = x$
  - 交换性commutativity: $x \sqcup y = y \sqcup x$
  - 结合性associativity：$(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
- 有界半格是存在最小元⊥的半格，满足
  - $x \sqcup \perp = x$

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$x \sqcup y = y$

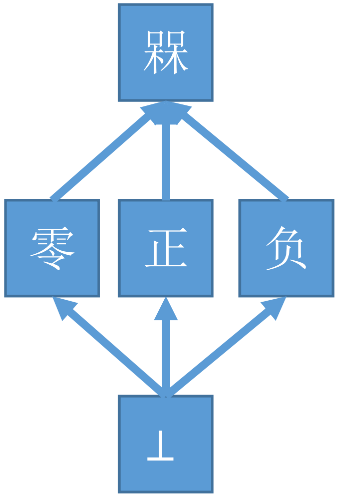
有界半格示例

- 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
  - $(s_1, t_1) \sqcup_{xy} (s_2, t_2) = \left(s_1 \sqcup_x s_2, t_1 \sqcup_y t_2\right)$
- 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 $(S, \sqcap_S)$ 和半格 $(T, \sqcap_T)$ 的高度分别是x和y，求半格$(S \times T, \sqcap_{ST})$的高度
  - $(s_1, t_1) \sqcap_{ST} (s_2, t_2) = (s_1 \sqcap_S s_2, t_1 \sqcap_T t_2)$

- 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：
  - $f(x) = x - 1$
  - 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
  - $f(x, y) = x \sqcap y$，已知$x \in S, y \in S, (S,\sqcap)$是和偏序关系对应的有界半格

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I
- 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

数据流分析实现算法

$\mathrm{OUT}_{\mathrm{entry}} = \mathrm{I}$

$\forall \mathrm{v} \in (\mathrm{V} - \mathrm{entry}): \mathrm{OUT}_{\mathrm{v}} \leftarrow \top$

$\mathrm{ToVisit} \leftarrow \mathrm{V} - \mathrm{entry}$

While(ToVisit.size > 0) {

  v $\leftarrow$ ToVisit中任意节点

  ToVisit -= v

  $\mathrm{IN}_{\mathrm{v}} \leftarrow \sqcup_{\mathrm{w} \in \mathrm{pred}(\mathrm{v})} \mathrm{OUT}_{\mathrm{w}}$

  $\mathrm{If}\left(\mathrm{OUT}_{\mathrm{v}} \neq \mathrm{f}_{\mathrm{v}}(\mathrm{IN}_{\mathrm{v}})\right)$ ToVisit $\cup=$ succ(v)

  $\mathrm{OUT}_{\mathrm{v}} \leftarrow \mathrm{f}_{\mathrm{v}}(\mathrm{IN}_{\mathrm{v}})$

}

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$(S, \sqcup)$和一个单调函数f，链$\perp, f(\perp), f\big(f(\perp)\big), \ldots$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$f^n(\perp)$是f的最小不动点。
  - 证明：
    - 收敛于f的不动点
      - $\perp \sqsubseteq f(\perp)$，两边应用f，得$f(\perp) \sqsubseteq f\big(f(\perp)\big)$，
      - 应用f，可得$f\big(f(\perp)\big) \sqsubseteq f\Big(f\big(f(\perp)\big)\Big)$
      - 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$\perp \sqsubseteq u$，两边反复应用f可证

数据流分析收敛性

- 给定任意确定性结点选择策略，原算法可以看做是反复应用一个函数
  - $\left(DATA_{v_1}, DATA_{v_2}, \ldots, DATA_{v_n}\right) := \mathrm{F}\left(DATA_{v_1}, DATA_{v_2}, \ldots, DATA_{v_n}\right)$
    - F是单调的吗?
- 根据不动点定理，原算法在有限步内终止，并且收敛于最小不动点

数据流分析的安全性

- 数据流分析的输出值满足如下等式
$$\mathrm{OUT}_v = f_v(\sqcup_{w \in \mathrm{pred}(v)} \mathrm{OUT}_w)$$
- 结合上节课讲的方法可以论证具体分析上的安全性
- 更通用的安全性论证方式下节课介绍

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：⊔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“乘以零”，$f_v(\text{正}) \sqcap f_v(\text{负})$
- 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$f(DATA) = (DATA - KILL) \cup \mathrm{GEN}$满足分配性
  - $f(x) \cup f(y) = (x - K) \cup G \cup (y - K) \cup G = (x - K) \cup (y - K) \cup G = (x \cup y - K) \cup G = f(x \cup y)$
  - $f(x) \cap f(y) = \big((x - K) \cup G\big) \cap \big((y - K) \cup G\big) = ((x - K) \cap (y - K)) \cup G = (x \cap y - K) \cup G = f(x \cap y)$

数据流分析小结

- 应用单调框架设计一个数据流分析包含如下内容
  - 设计每个节点附加值的定义域
  - 设计交汇函数
  - 设计从语句导出节点变换函数的方法
  - 入口节点的初值
- 需要证明如下内容
  - 在单条路径上，变换函数保证安全性
  - 交汇函数对多条路径的合并方式保证安全性
  - 交汇函数形成一个有界半格
  - 有界半格的高度有限
    - 通常通过节点附加值的定义域为有限集合证明
  - 变换函数均为单调函数
    - 通常定义为$f(D) = (D - KILL) \cup \mathrm{GEN}$的形式

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
- 交汇操作：区间的并
  - $[a,b] \sqcap [c,d] = [\min(a,c), \max(b,d)]$
- 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

Widening & Narrowing

Widening

- 区间分析需要很多步才能达到收敛
  - 格的高度太高
- Widening：通过降低结果的精度来加快收敛速度
  - 基础Widening：降低格的高度
  - 一般Widening：根据变化趋势快速猜测一个结果

基础Widening

- 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:
  - $w([15,75]) = [10,100]$

基础Widening的例子

- 令基础widening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

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

不使用Widening，
收敛慢或不收敛

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

使用基础Widening
收敛快，但不精确

基础Widening的安全性

- 如果$w(x) \sqsubseteq x$，则分析结果保证安全

- 安全性讨论
  - 新转换结果小于等于原结果，意味着$DATA_V$的结果小于等于原始结果

基础Widening的收敛性

- 如果w是单调函数，则基础Widening收敛
  - 因为$w \circ f$仍然是单调函数

一般Widening

- 更一般的widening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：
    - $\mathrm{DATA_v} \leftarrow \mathrm{f_v}(\mathrm{MEET_v})$
  - 引入widen算子∇：
    - $\mathrm{DATA_v} \leftarrow \mathrm{DATA_v}\ \nabla\ \mathrm{f_v}(\mathrm{MEET_v})$

- 用更一般的widening可以实现更快速的收敛，如
  - $[a, b]\nabla\top = [a, b]$
  - $\top\nabla[c, d] = [c, d]$
  - $[a, b]\nabla[c, d] = [m, n]$ 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：$x \in [a, b]$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widen算子的思路

一般Widening的例子

- 令基础widening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

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

不使用Widening，
收敛慢或不收敛

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

使用基础Widening
收敛快，但不精确

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

使用一般Widening
收敛更快，
结果(恰好)精确

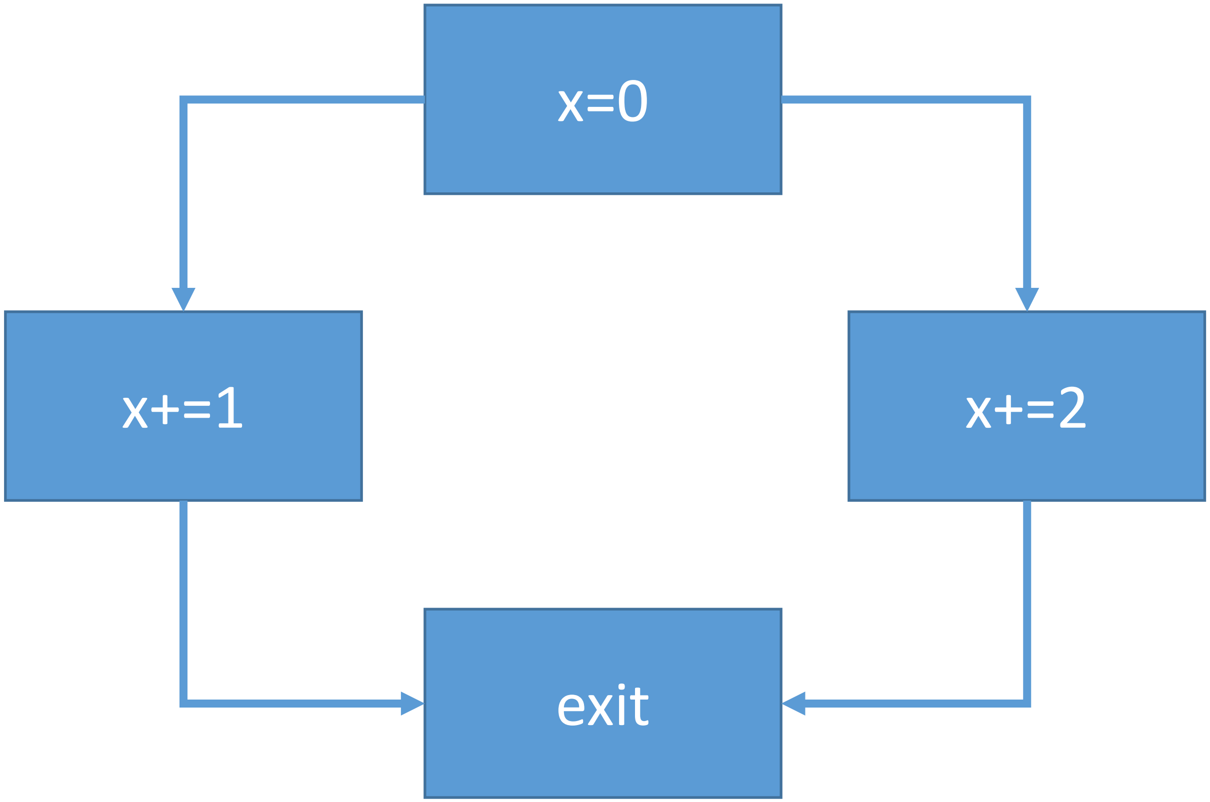
一般Widening的性质

- 如果$x\nabla y \sqsubseteq y \land x\nabla y \sqsubseteq x$，则一般Widening的分析结果保证安全性

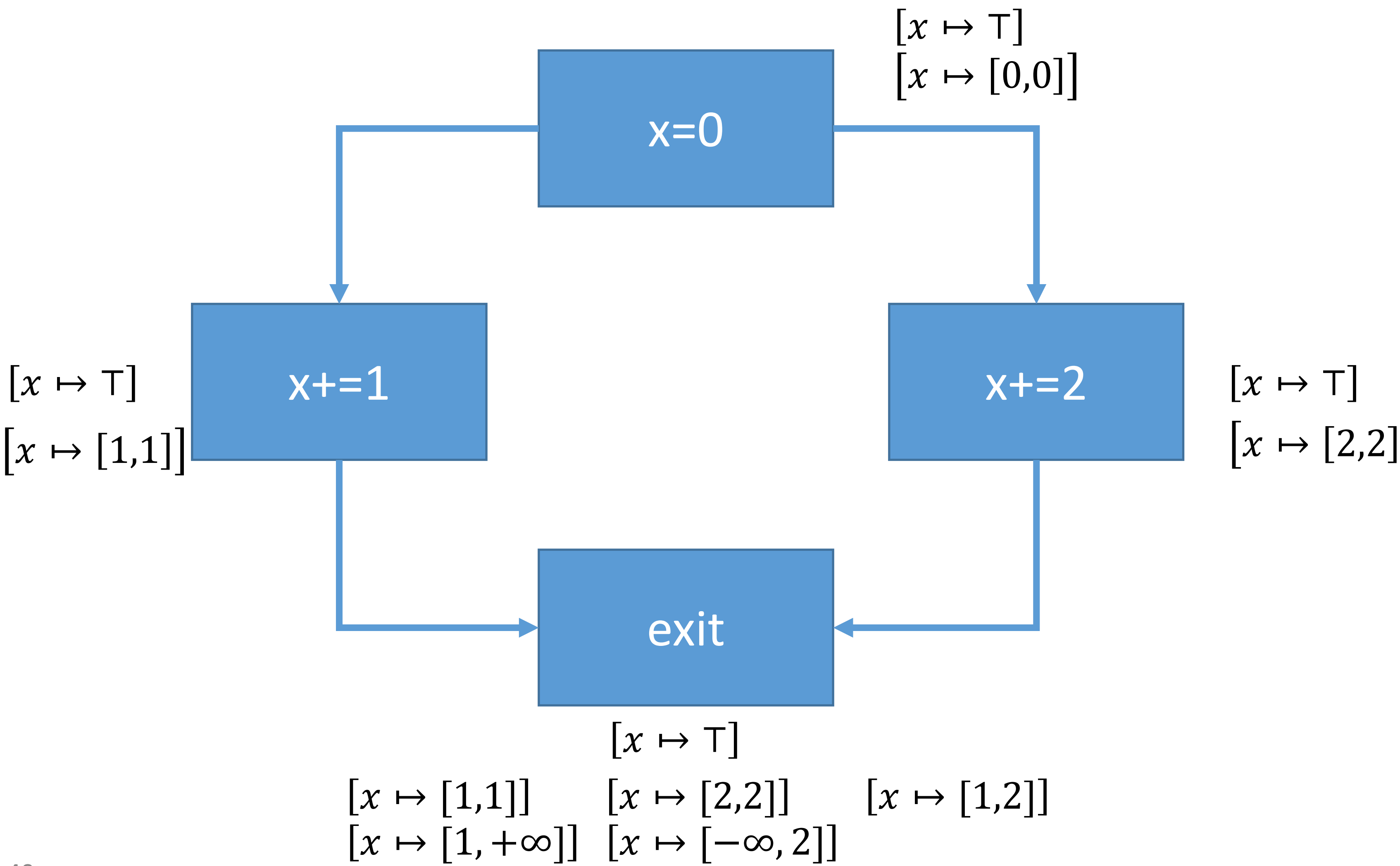
- 目前没有找到容易判断的属性来证明一般Widening的收敛性
  - Widening算子本身通常不保证变换函数单调递增
  - $[1,1]\nabla[1,2] = [1,\infty]$
  - $[1,2]\nabla[1,2] = [1,2]$

  - 能否给出一个区间分析上不收敛的例子?

Widening不收敛的例子



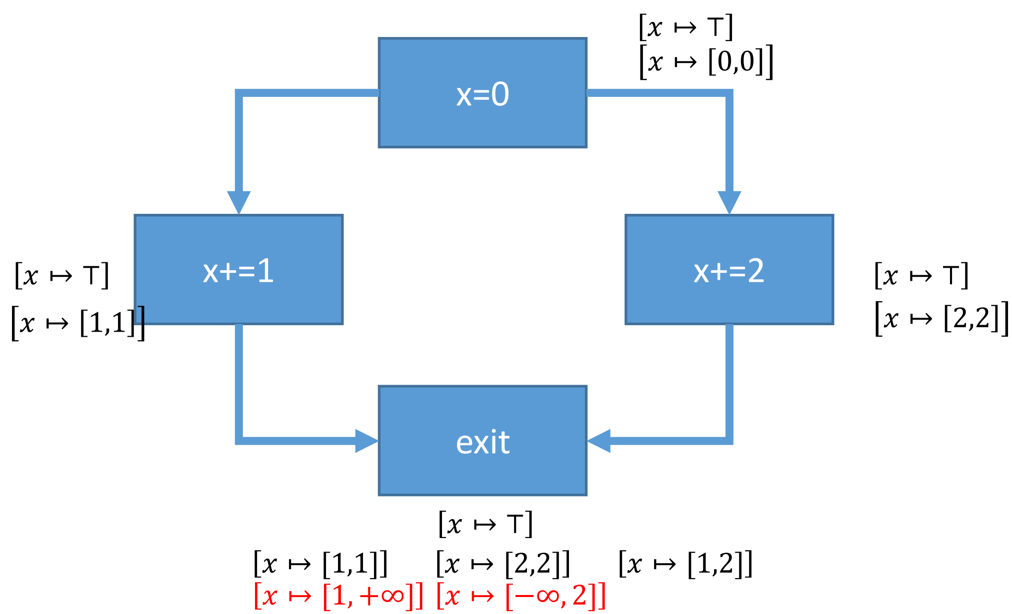
Widening不收敛的例子



如何拯救Widening带来的不精确?

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

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



Narrowing

- 通过再次应用原始转换函数对Widening的结果进行修正

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

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

Narrowing的安全性

- 分析数据流分析收敛性时，整体数据流分析可以看做一个函数F
- 令
  - 原数据流分析的函数为F，收敛于I_F
  - 经过Widening的函数为G，收敛于I_G
- 那么有
  - 因为 $I_F \sqsupseteq I_G$
  - 所以 $I_F = F(I_F) \sqsupseteq F(I_G) \sqsupseteq G(I_G) = I_G$
  - 即 $I_F \sqsupseteq F(I_G) \sqsupseteq I_G$
- 类似可得
  - $I_F \sqsupseteq F^k(I_G) \sqsupseteq I_G$
- 即Narrowing保证安全性

Narrowing的收敛性

- Narrowing不保证收敛
- 收敛的情况下也不保证快速收敛
  - 例子需要用到关系型抽象域

参考资料

- 《编译原理》第9章
- Lecture Notes on Static Analysis
  - https://cs.au.dk/~amoeller/spa/
- A Gentle Introduction to Abstract Interpretation
  - Patrick Cousot
  - TASE 2015 Keynote speech
- 抽象解释及其在静态分析中的应用
  - 陈立前
  - SWU-RISE Computer Science Tutorial