软件分析

数据流分析：性质和扩展

熊英飞
北京大学
2019

复习：数据流分析

- 数据流分析中采用了哪两种近似方案？

- 近似方案1：忽略掉程序的条件判断，认为所有分支都有可能到达
- 近似方案2：不在路径末尾做合并，在控制流汇合的所有位置提前做合并

复习：半格

- 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

复习：单调函数

- 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)
- 一个有限高度的半格$(S, \sqcap)$
- 一个entry的初值I
- 一组结点转换函数，对任意$v \in V - entry$存在一个结点转换函数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

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

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

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

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

$\quad \mathrm{v} \leftarrow \mathrm{ToVisit}$中任意结点

$\quad \mathrm{ToVisit}\ \text{-=}\ \mathrm{v}$

$\quad \mathrm{MEET}_{\mathrm{v}} \leftarrow \sqcap_{\mathrm{w} \in \mathrm{pred}(\mathrm{v})} \mathrm{DATA}_{\mathrm{w}}$

$\quad \mathrm{If}\big(\mathrm{DATA}_{\mathrm{V}} \neq \mathrm{f}_{\mathrm{v}}(\mathrm{MEET}_{\mathrm{v}})\big)\ \mathrm{ToVisit} \cup= \mathrm{succ}(\mathrm{v})$

$\quad \mathrm{DATA}_{\mathrm{v}} \leftarrow \mathrm{f}_{\mathrm{v}}(\mathrm{MEET}_{\mathrm{v}})$

$\}$

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
  - 设计每个结点附加值的定义域
  - 设计交汇函数
  - 设计从语句导出结点变换函数的方法
  - 入口结点的初值
- 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
  - 交汇函数形成一个半格
  - 半格的高度有限
    - 通常通过结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

集合的最大下界

- 下界：给定集合S，如果满足$\forall s \in S: u \sqsubseteq s$，则称u是S的一个下界
- 最大下界：设u是集合S的下界，给定任意下界u′，如果满足$u' \sqsubseteq u$，则称u是S的最大下界，记为$\sqcap_S$
- 引理：$\sqcap_{s\in S} s$是S的最大下界
  - 证明：
    - 根据幂等性、交换性和结合性，我们有$\forall v \in S$: $(\sqcap_{s\in S} s) \sqcap v = \sqcap_{s\in S} s$，所以$\sqcap_{s\in S} s$是S的下界
    - 给定另一个下界u，我们有$\forall s \in \mathrm{S} : s \sqcap u = u$，$(\sqcap_{s\in S} s) \sqcap u = (\sqcap_{s\in S}(s \sqcap u)) = \mathrm{u}$，所以$\sqcap_{s\in S} s$是最大下界
- 推论：半格的任意子集都有最大下界

数据流分析的安全性-定义

- 安全性：对控制流图上任意结点v_i和从entry到v_i的所有可行路径集合P，满足$\mathrm{DATA}_{\mathrm{v_i}} \sqsubseteq \sqcap_{v_1 v_2 v_3 \ldots v_i \in P} f_{v_i} \circ f_{v_{i-1}} \circ \cdots \circ f_{v_1}\left(I_{entry}\right)$
  - 示例：符号分析的偏序关系中槑比较小，T比较大，结果是上近似
  - 示例：活跃变量分析的偏序关系为超集关系，所以数据流分析产生相等或者较大集合，是上近似

数据流分析的安全性-证明

- 给定任意路径的$v_1 v_2 v_3 \ldots v_i$，$\mathrm{DATA}_{\mathrm{v_i}}$的计算相当于在每两个相邻转换函数$f_{v_i} \circ f_{v_{i-1}}$之间加入了MEET交汇计算，根据幂等性，任意交汇计算的结果一定在偏序上小于等于原始结果。再根据转换函数的单调性，$\mathrm{DATA}_{\mathrm{v_i}}$的值一定小于等于$f_{v_i} \circ f_{v_{i-1}} \circ \cdots \circ f_{v_1}\left(I_{entry}\right)$。由于原路径的任意性，$\mathrm{DATA}_{\mathrm{v_i}}$是一个下界。
- 再根据前面的引理，$\sqcap_{v_1 v_2 v_3 \ldots v_i \in P} f_{v_i} \circ f_{v_{i-1}} \circ \cdots \circ f_{v_1}\left(I_{entry}\right)$是最大下界，所以原命题成立。

数据流分析的分配性

- 一个数据流分析满足分配性，如果
  - $\forall v \in V, x, y \in S: f_v(x) \sqcap f_v(y) = f_v(x \sqcap y)$
- 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(正) $\sqcap$ f_v(负)
- 例：在集合和交/并操作构成的半格中，给定任意两个集合GEN, KILL，函数$f(DATA) = (DATA - KILL) \cup$ GEN满足分配性
  - $f(x) \cup f(y) = (x - D) \cup G \cup (y - D) \cup G = (x - D) \cup (y - D) \cup G = (x \cup y - D) \cup G = f(x \cup y)$
  - $f(x) \cap f(y) = \big((x - D) \cup G\big) \cap \big((y - D) \cup G\big) = ((x - D) \cap (y - D)) \cup G = (x \cap y - D) \cup G = f(x \cap y)$

数据流分析的分配性

- 一个数据流分析满足分配性，如果
  - $\forall v \in V, x, y \in S: f_v(x) \sqcap f_v(y) = f_v(x \sqcap y)$
- 令P’为所有控制流图上的路径，含不可行路径。当数据流分析满足分配性的时候，$\mathrm{DATA}_{\mathrm{v_i}} = \sqcap_{v_1 v_2 v_3 \ldots v_i \in P'} f_{v_i} \circ f_{v_{i-1}} \circ \cdots \circ f_{v_1}(I_{entry})$
  - 也就是说，此时近似方案2不是近似，而是等价变换
  - 但是，数据流分析本身还可能是近似
    - 近似方案1是近似
    - 结点转换函数有可能是近似

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：给定高度有限的半格$(S, \sqcap)$和一个单调函数f，链$\top_S, f(\top_S), f\big(f(\top_S)\big), \ldots$必定在有限步之内收敛于f的最大不动点，即存在非负整数n，使得$f^n(\top_S)$是f的最大不动点。
    - 证明：
        - 收敛于f的不动点
            - $f(\top_S) \sqsubseteq \top_S$，两边应用f，得$f(f(\top_S)) \sqsubseteq f(\top_S)$，
            - 应用f，可得$f\Big(f\big(f(\top_S)\big)\Big) \sqsubseteq f\big(f(\top_S)\big)$
            - 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。
        - 收敛于最大不动点
            - 假设有另一不动点u，则$u \sqsubseteq \top_S$，两边反复应用f可证

数据流分析收敛性

- 给定固定的结点选择策略，原算法可以看做是反复应用一个函数
  - $\left(DATA_{v_1}, DATA_{v_2}, \ldots, DATA_{v_n}\right) := \mathrm{F}\left(DATA_{v_1}, DATA_{v_2}, \ldots, DATA_{v_n}\right)$
    - 为什么没有$DATA_{entry}$
    - F是单调的吗？
- 根据不动点定理，原算法在有限步内终止，并且收敛于最大不动点

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
- 半格元素：程序中每个变量的区间
- 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widening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
  - $[a,b]\nabla\top = [a,b]$
  - $\top\nabla[c,d] = [c,d]$
  - $[a,b]\nabla[c,d] = [x,y]$ where
    - $x = \begin{cases} a & c \geq a \\ -\infty & c < a \end{cases}$
    - $y = \begin{cases} b & d \leq b \\ +\infty & d > b \end{cases}$

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

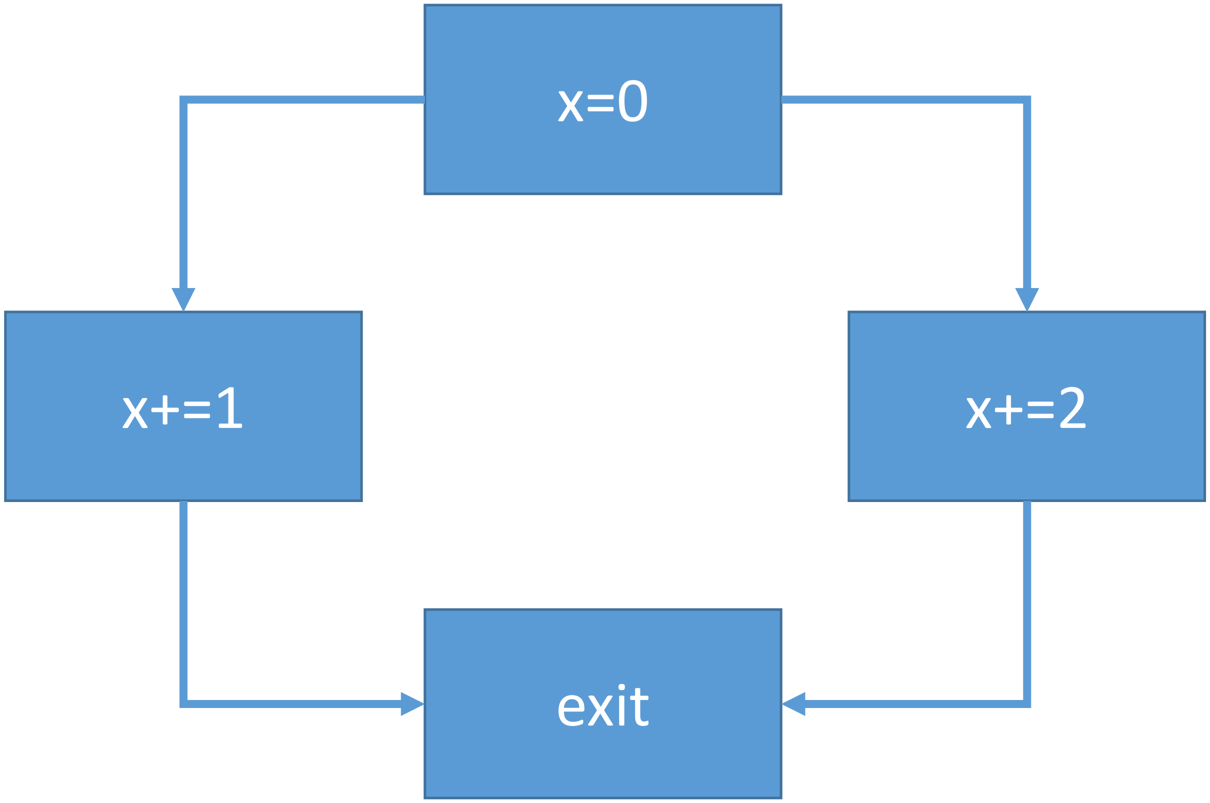
一般Widening的性质

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

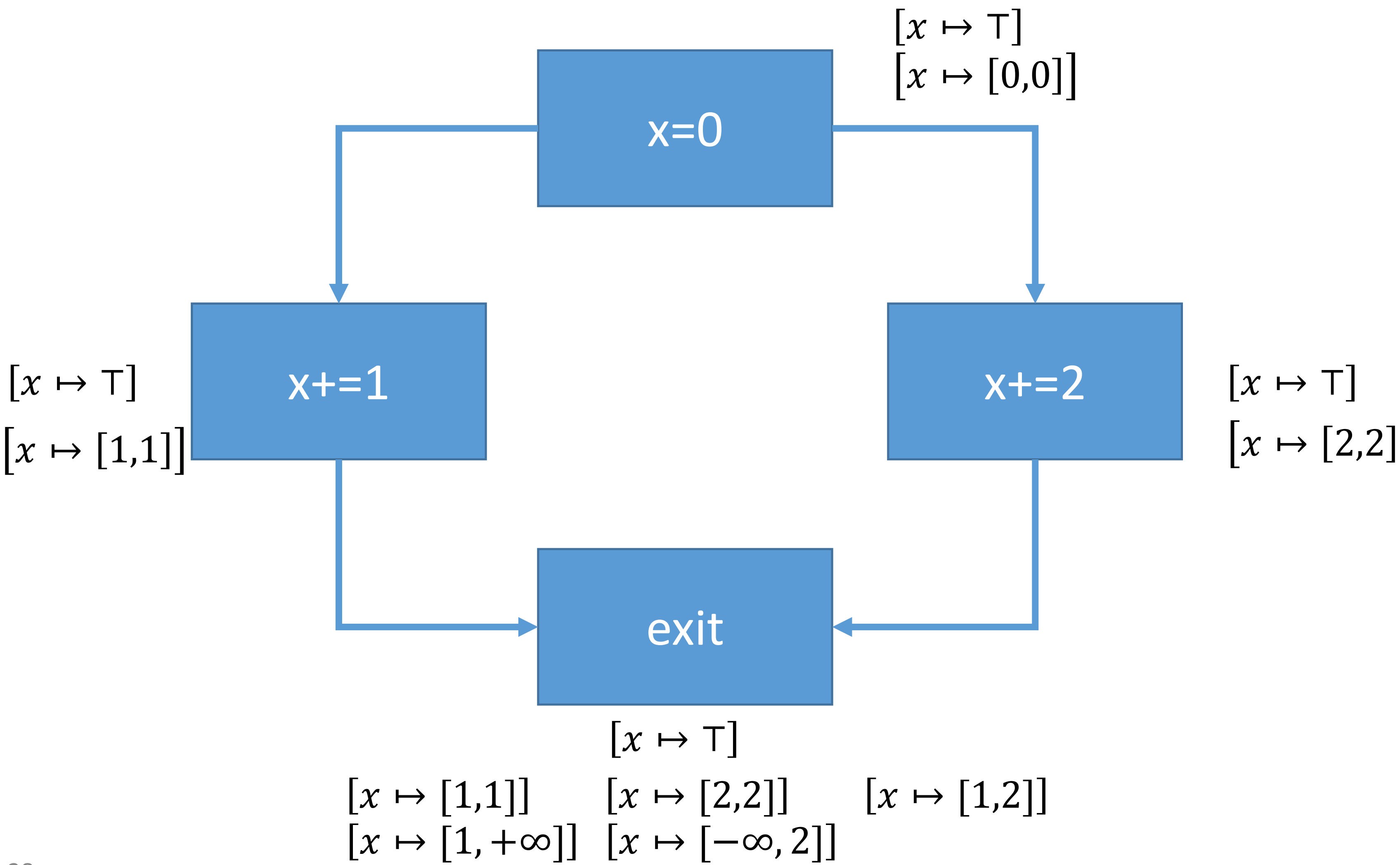
- 目前没有找到容易判断的属性来证明一般Widening的收敛性
  - Widening算子本身通常不保证变换函数单调递增
  - $[1,1]\nabla[1,2] = [1,\infty]$
  - $[1,2]\nabla[1,2] = [1,2]$

  - 能否给出一个区间分析上不收敛的例子？

Widening不收敛的例子



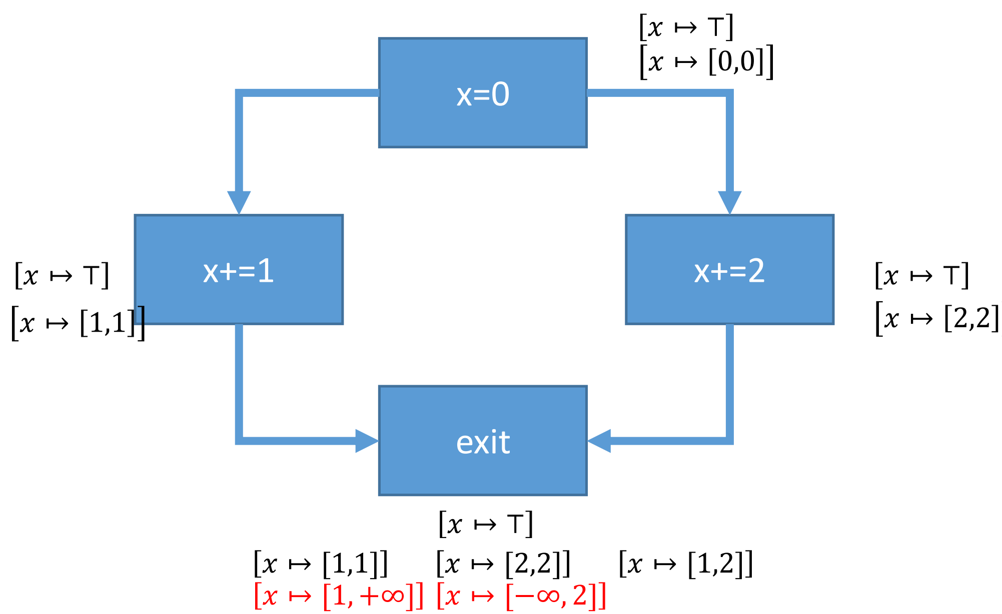
Widening不收敛的例子



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的时候，我们说过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 $I_F \sqsupseteq F(I_G) \sqsupseteq I_G$
- 类似可以得到
  - $I_F \sqsupseteq F^k(I_G) \sqsupseteq I_G$
- 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
  - 具体例子在路径敏感分析阶段学习
- 解决方案：应用widening技术到narrowing过程中

Narrowing算子

- 引入narrowing算子Δ：

$$\mathrm{DATA_v} \leftarrow \mathrm{DATA_v}\ \Delta\ \mathrm{f_v}(\mathrm{MEET_v})$$

- 如
  - $[a,b]\Delta[c,d] = [x,y]$, where
    - $x = \begin{cases} a & a \neq -\infty \\ c & a = -\infty \end{cases}$
    - $y = \begin{cases} b & b \neq +\infty \\ d & b = +\infty \end{cases}$
  - 即：已经收敛到的整数不改动，只重新计算被widening扩展到的无穷大

Narrowing算子的性质

- 同widening的情形类似：

- 如果$x \Delta y \sqsubseteq y$，则narrowing保证安全
- 目前没有找到容易判断的属性来证明Narrowing的收敛性

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