软件分析

加宽变窄

熊英飞
北京大学

练习：区间（Interval）分析

- 求结果的上界和下界
  - 要求上近似
  - 假设变量均为数学整数；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⊥
- 合并操作：包含两区间的最小区间（区间凸包）
  - $[a,b] \sqcup [c,d] = [\min(a,c), \max(b,d)]$
- 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：半格高度不是有限的
  - 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] = [a+c,b+d] ⊓ [int_min,int_max]

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

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

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

简易加宽的安全性

- 如果$x \sqsubseteq w(x)$，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

标准区间加宽：逐节点迭代

第 1 / 12 步

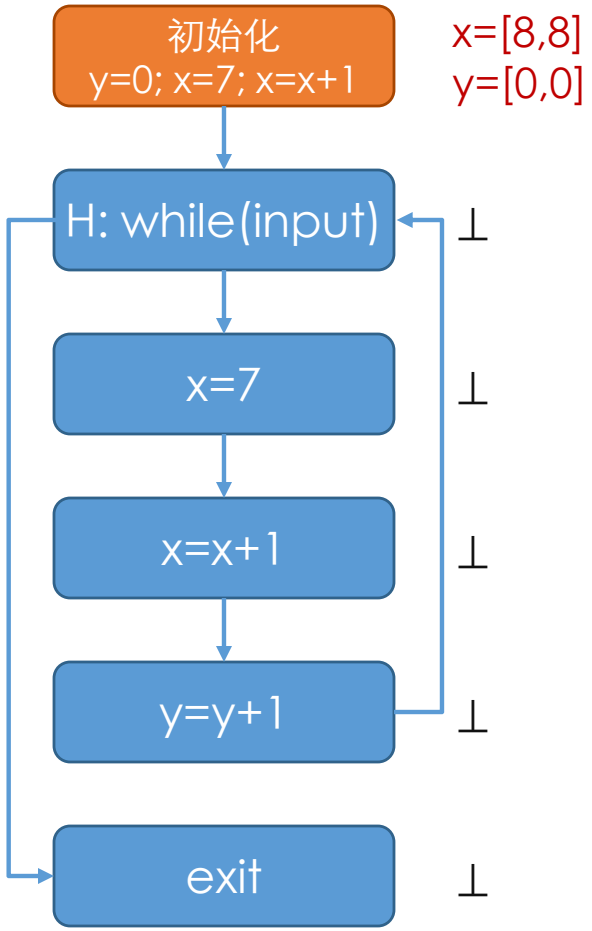
初始化入口基本块

旧值：⊥

候选：x=[8,8]，y=[0,0]

新值：x=[8,8]，y=[0,0]

三条直线语句执行后，x=8，y=0。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 2 / 12 步

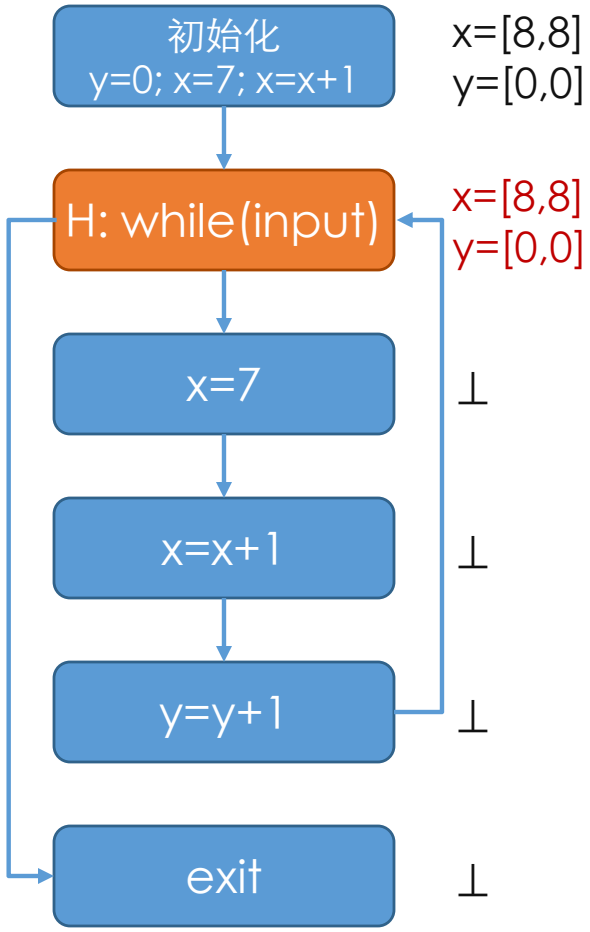
首次到达循环头

旧值：⊥

候选：x=[8,8]，y=[0,0]

新值：x=[8,8]，y=[0,0]

H 的两个分支都可能执行。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 3 / 12 步

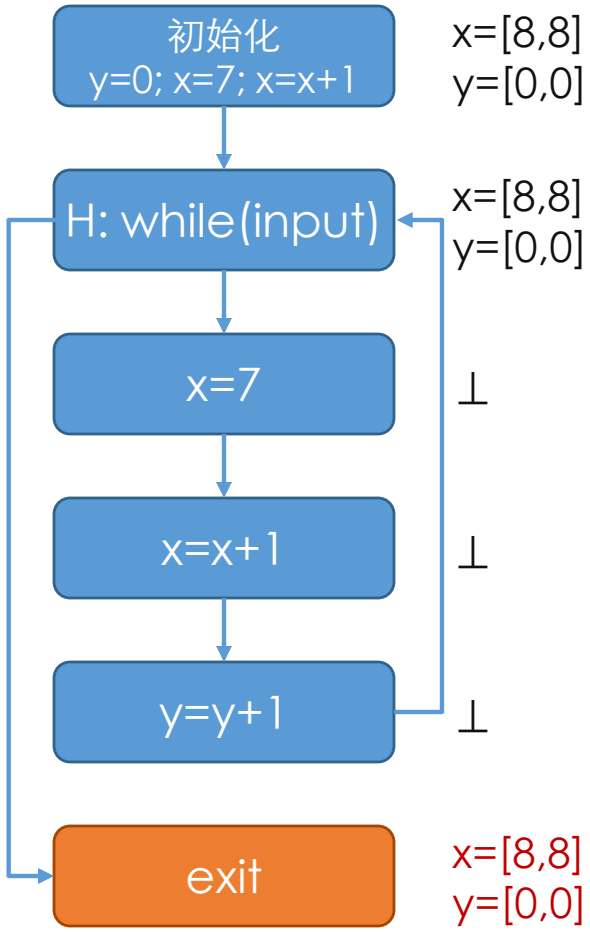
先传播假分支

旧值：⊥

候选：x=[8,8]，y=[0,0]

新值：x=[8,8]，y=[0,0]

input 可为假，出口先取得 y=0。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 4 / 12 步

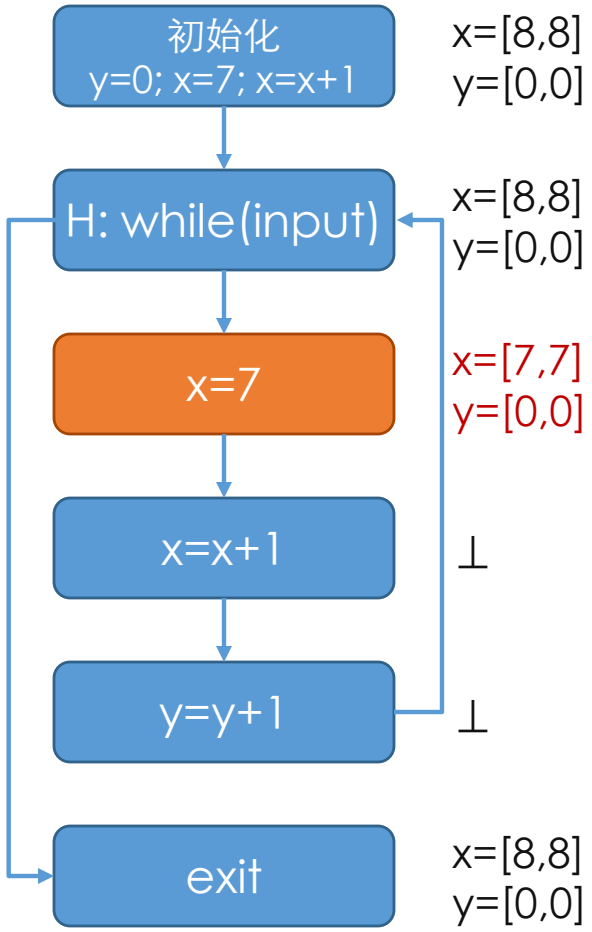
进入循环体：x=7

旧值：⊥

候选：x=[7,7]，y=[0,0]

新值：x=[7,7]，y=[0,0]

赋值覆盖 x 的旧值，y 保持 0。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 5 / 12 步

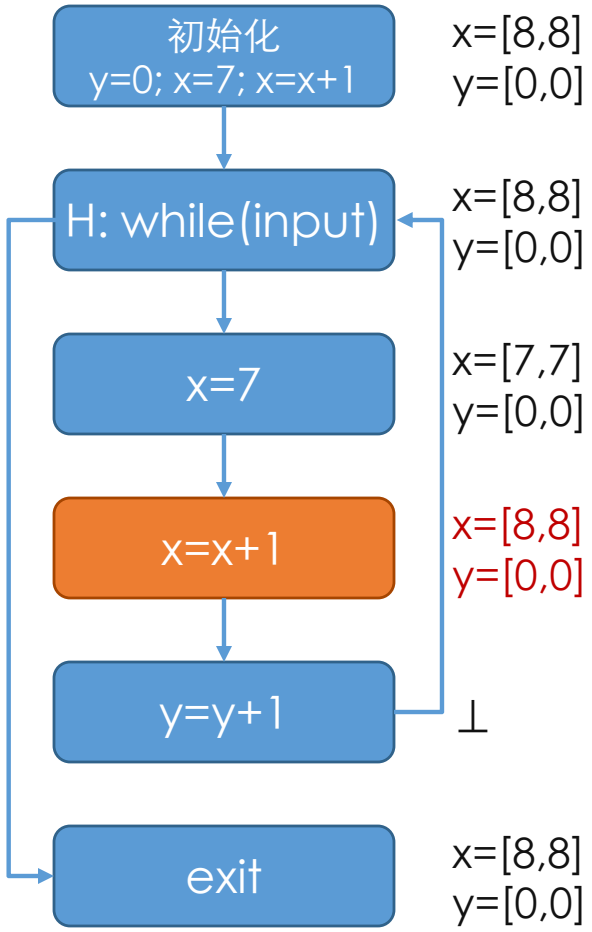
循环体：x=x+1

旧值：⊥

候选：x=[8,8]，y=[0,0]

新值：x=[8,8]，y=[0,0]

x 从 7 变为 8。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 6 / 12 步

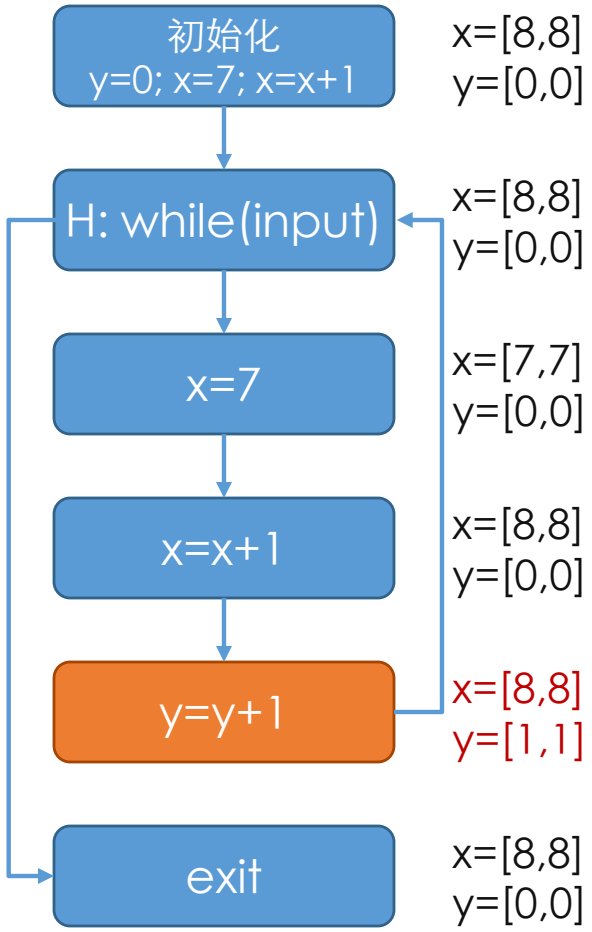
循环体：y=y+1

旧值：⊥

候选：x=[8,8]，y=[1,1]

新值：x=[8,8]，y=[1,1]

第一次回边带来 y=1。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 7 / 12 步

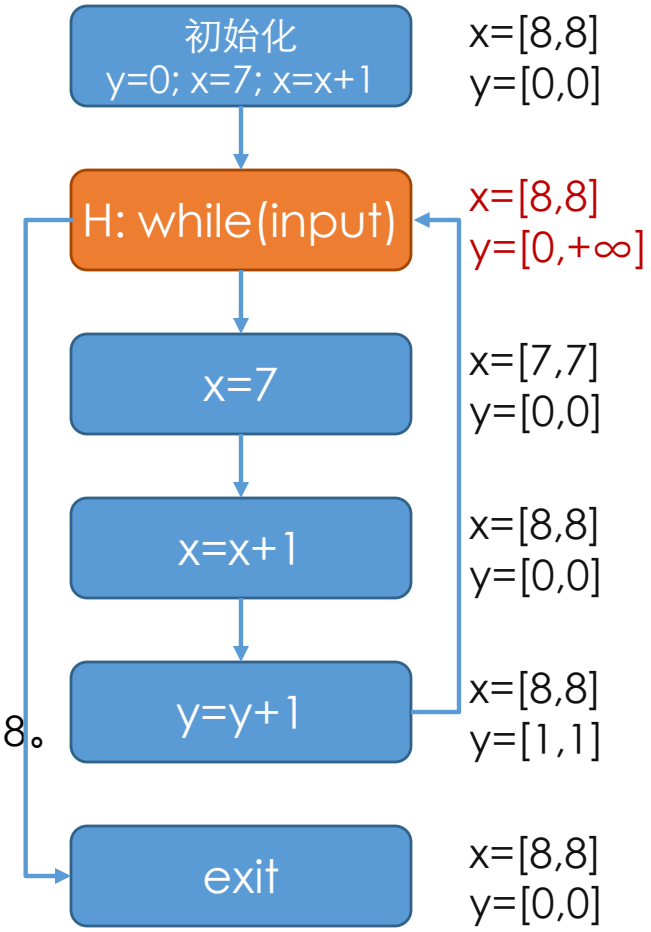
回边触发循环头加宽

旧值：x=[8,8]，y=[0,0]

候选：x=[8,8]，y=[0,1]

新值：x=[8,8]，y=[0,+∞]

y 的上界 0 被 1 突破，升为 +∞；x 的两端保持 8。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 8 / 12 步

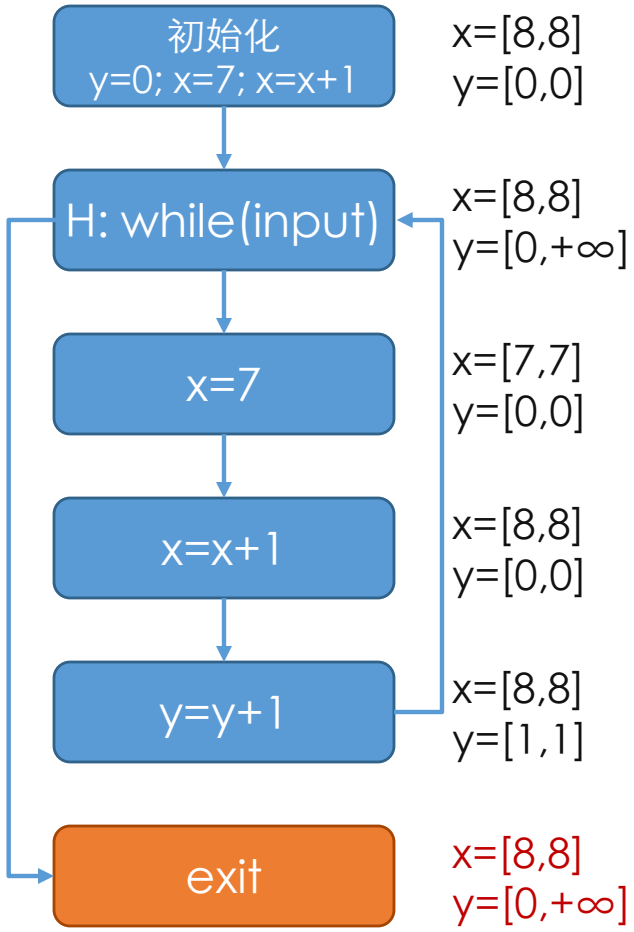
更新出口

旧值：x=[8,8]，y=[0,0]

候选：x=[8,8]，y=[0,+∞]

新值：x=[8,8]，y=[0,+∞]

出口收到加宽后的循环头值。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 9 / 12 步

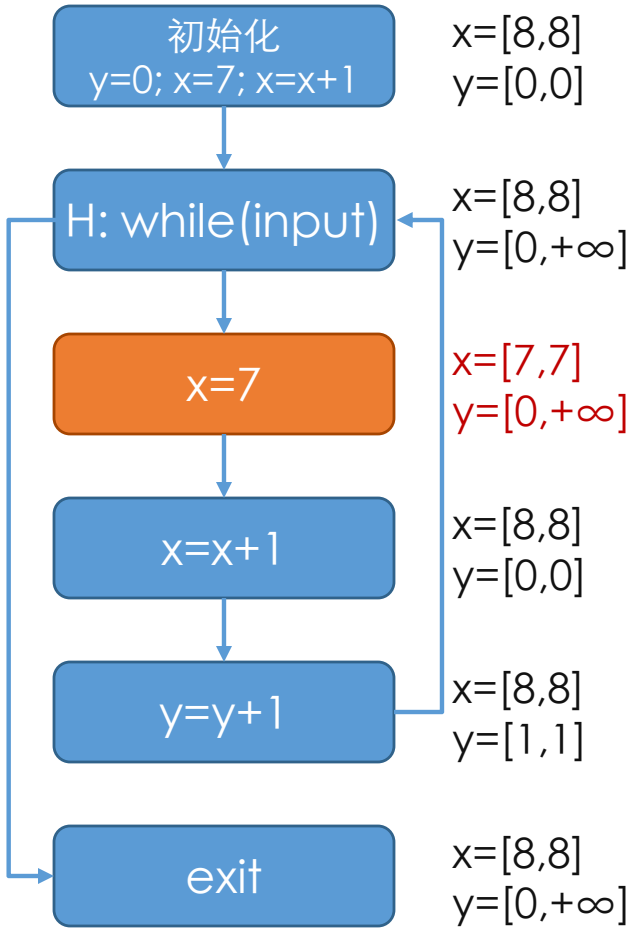
再传播 x=7

旧值：x=[7,7]，y=[0,0]

候选：x=[7,7]，y=[0,+∞]

新值：x=[7,7]，y=[0,+∞]

x 仍被重置为 7；y 上界保持 +∞。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 10 / 12 步

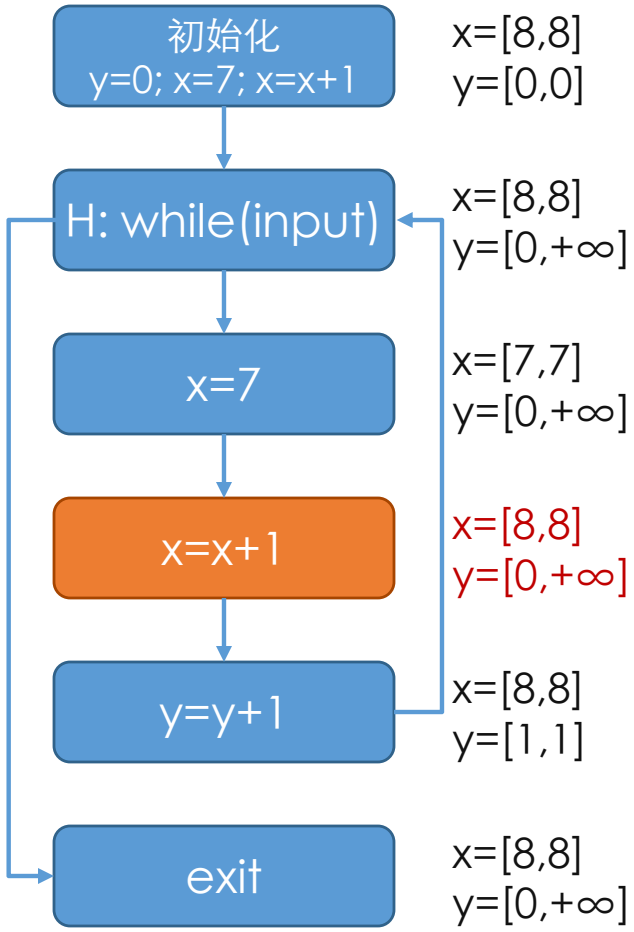
再传播 x=x+1

旧值：x=[8,8]，y=[0,0]

候选：x=[8,8]，y=[0,+∞]

新值：x=[8,8]，y=[0,+∞]

x 回到 8；y 的范围不变。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 11 / 12 步

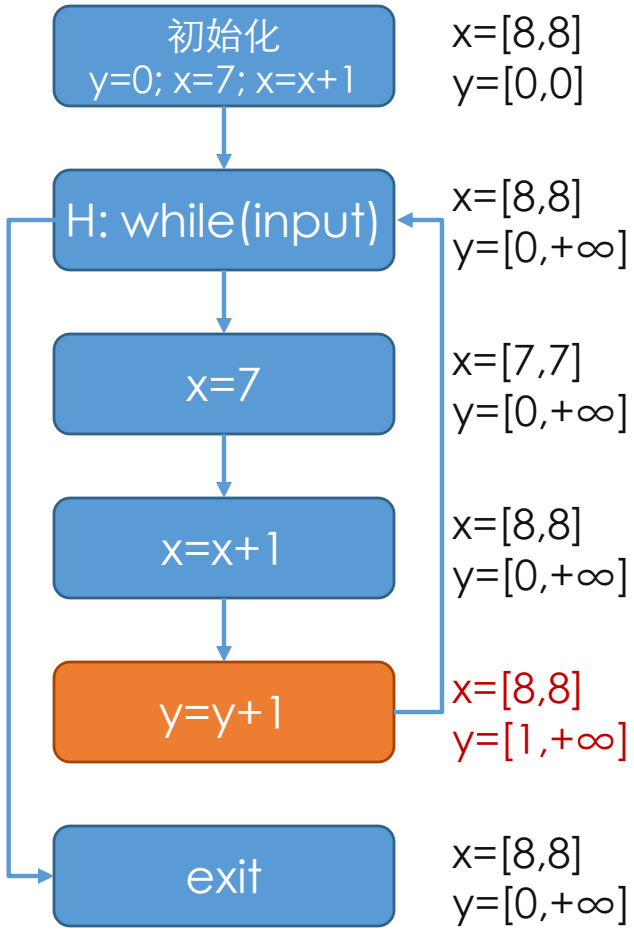
再传播 y=y+1

旧值：x=[8,8]，y=[1,1]

候选：x=[8,8]，y=[1,+∞]

新值：x=[8,8]，y=[1,+∞]

y 的下界为 1，上界仍为 +∞。



所有节点使用标准区间加宽；橙色表示当前更新节点。

标准区间加宽：逐节点迭代

第 12 / 12 步

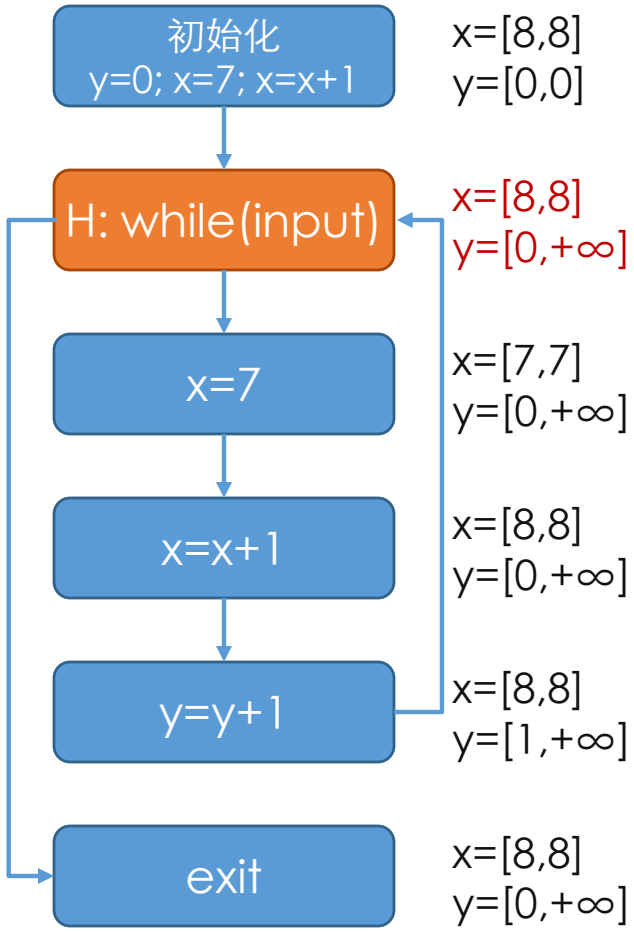
再次检查循环头：稳定

旧值：x=[8,8]，y=[0,+∞]

候选：x=[8,8]，y=[0,+∞]

新值：x=[8,8]，y=[0,+∞]

候选与旧值相同，不再加入后继节点。



所有节点使用标准区间加宽；橙色表示当前更新节点。

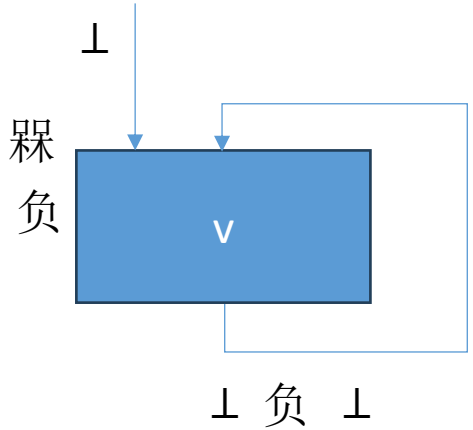
通用加宽的安全性

- 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，如果分析终止，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$\perp$ 上不断应用F，而新分析是不断应用G
  - 现在证明对任意i，$F^i(\perp) \sqsubseteq G^i(\perp)$
    - $i = 0$时，显然成立
    - 假设$i = k$时成立，因为F的单调性，那么有$F^{k+1}(\perp) \sqsubseteq F(G^k(\perp))$
    - 又因为∇的性质，我们有$F(G^k(\perp)) \sqsubseteq \mathrm{G}^{\mathrm{k}}(\perp) \nabla F(G^k(\perp)) = \mathrm{G}^{\mathrm{k+1}}(\perp)$
    - 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$F^i(\perp) \sqsubseteq G^i(\perp)$

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
  - $x \nabla \mathrm{y} = \begin{cases} \text{踩} & y = \bot \\ y & otherwise \end{cases}$
  - $f_v(\text{甲}) = \begin{cases} \text{负} & \text{甲} = \text{踩} \\ \bot & otherwise \end{cases}$

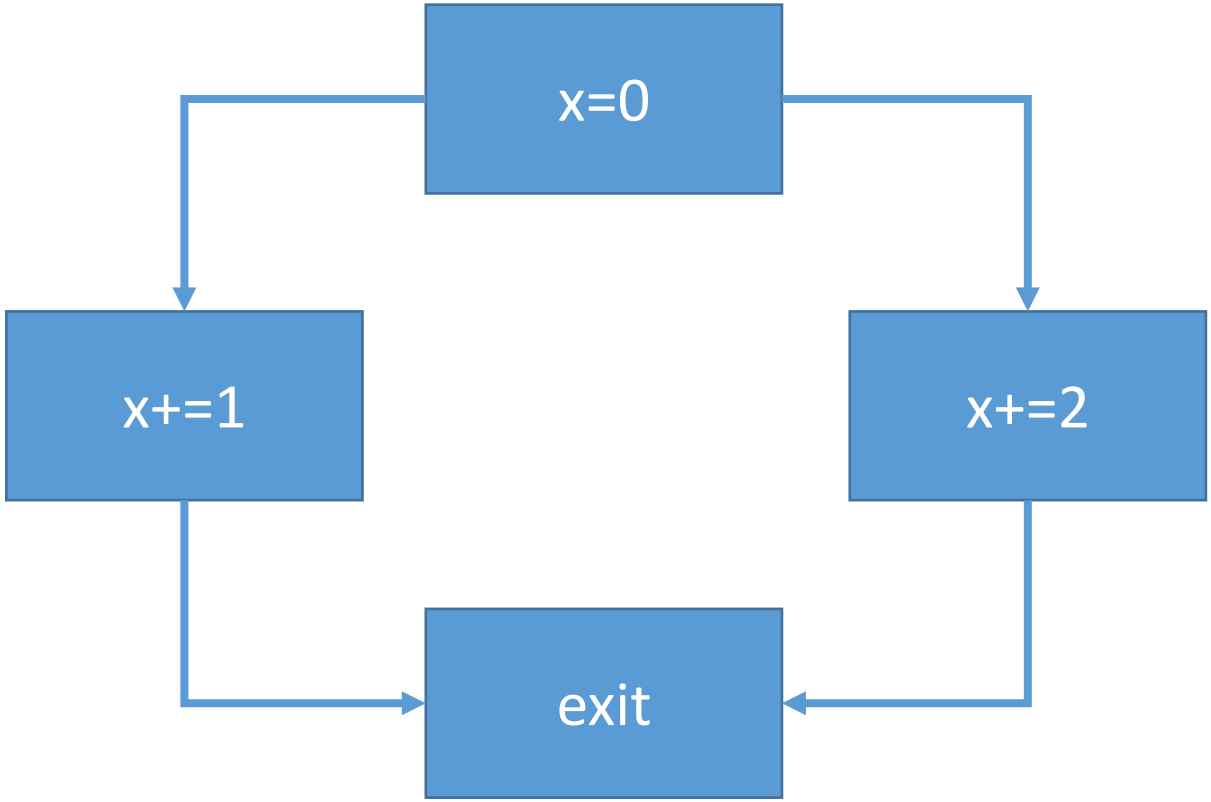


通用加宽的收敛性

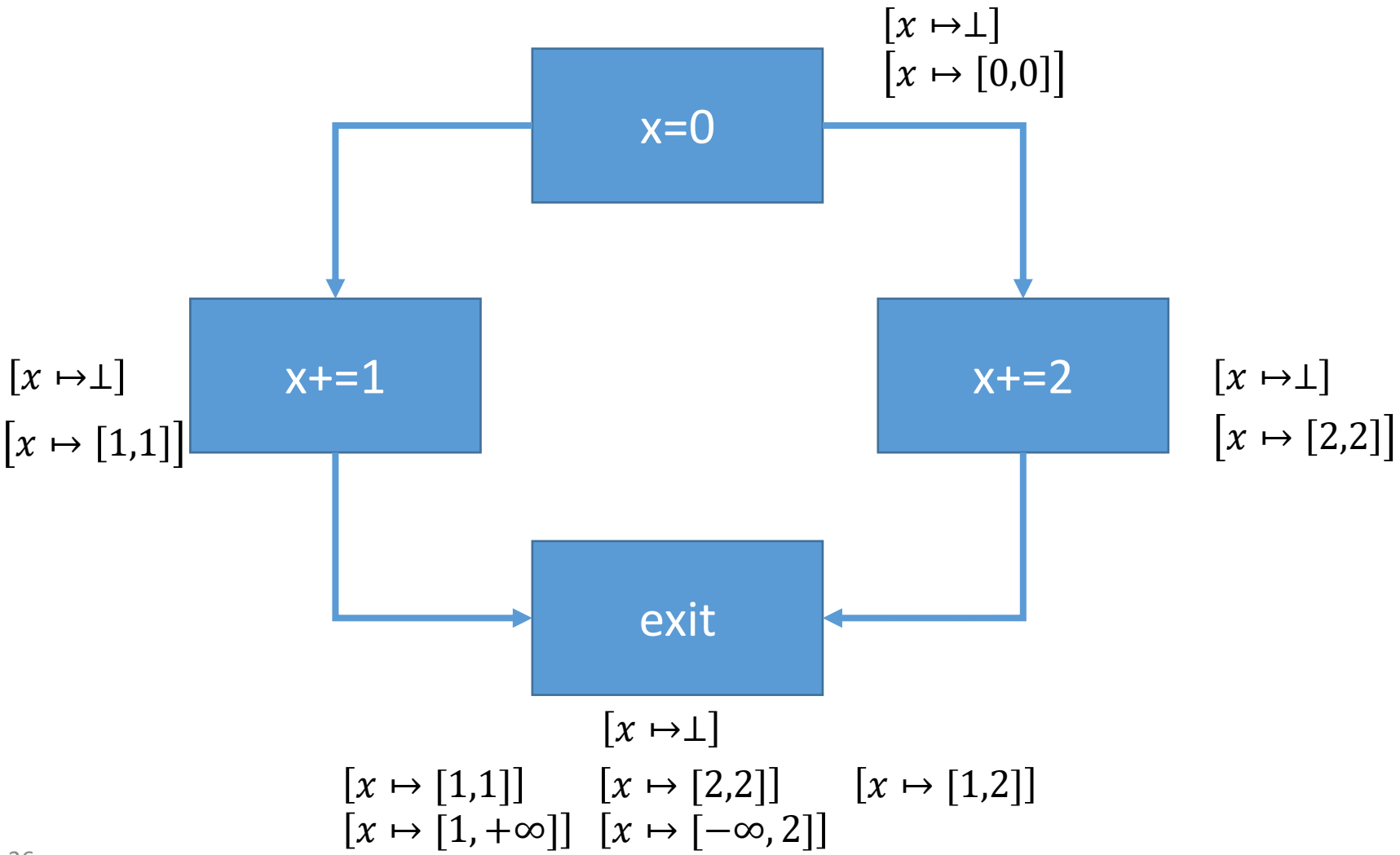
- 目前没有找到容易判断的属性来证明通用加宽的收敛性
  - 加宽算子本身通常不保证变换函数单调
  - $[1,1]\nabla[1,2] = [1,\infty]$
  - $[1,2]\nabla[1,2] = [1,2]$

  - 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


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

通用加宽的终止性

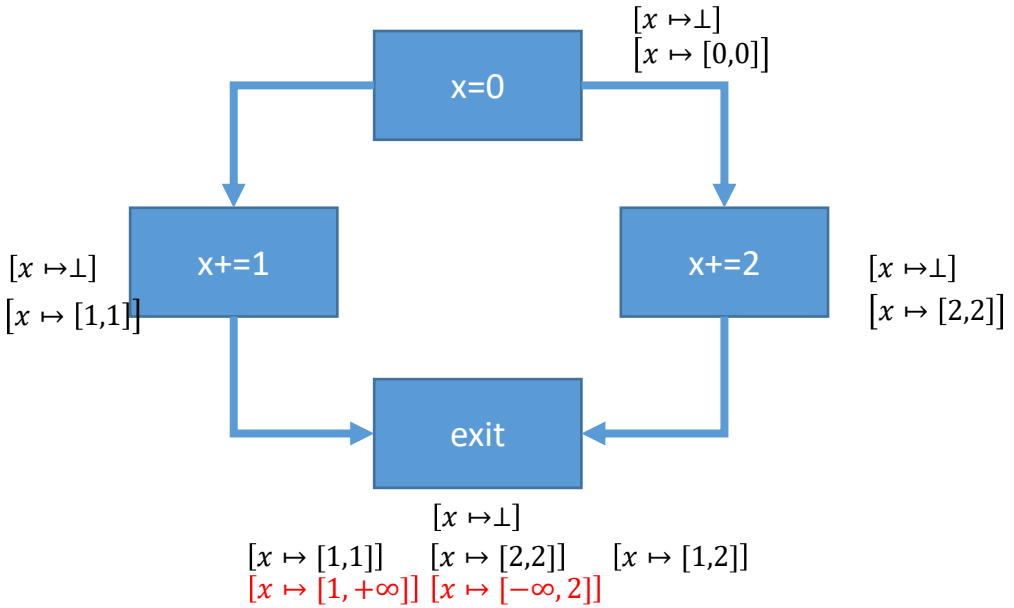
- 相对合流，终止性更加重要
- 但同样，目前也没有找到容易判断的属性来证明通用加宽的终止性

- 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$x_0, x_1, \ldots$
  - 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;
}
```

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



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：

- 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/
- 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
- 《Introduction to Static Analysis》Rival and Yi