软件分析

关系型抽象域

熊英飞
北京大学

非关系抽象

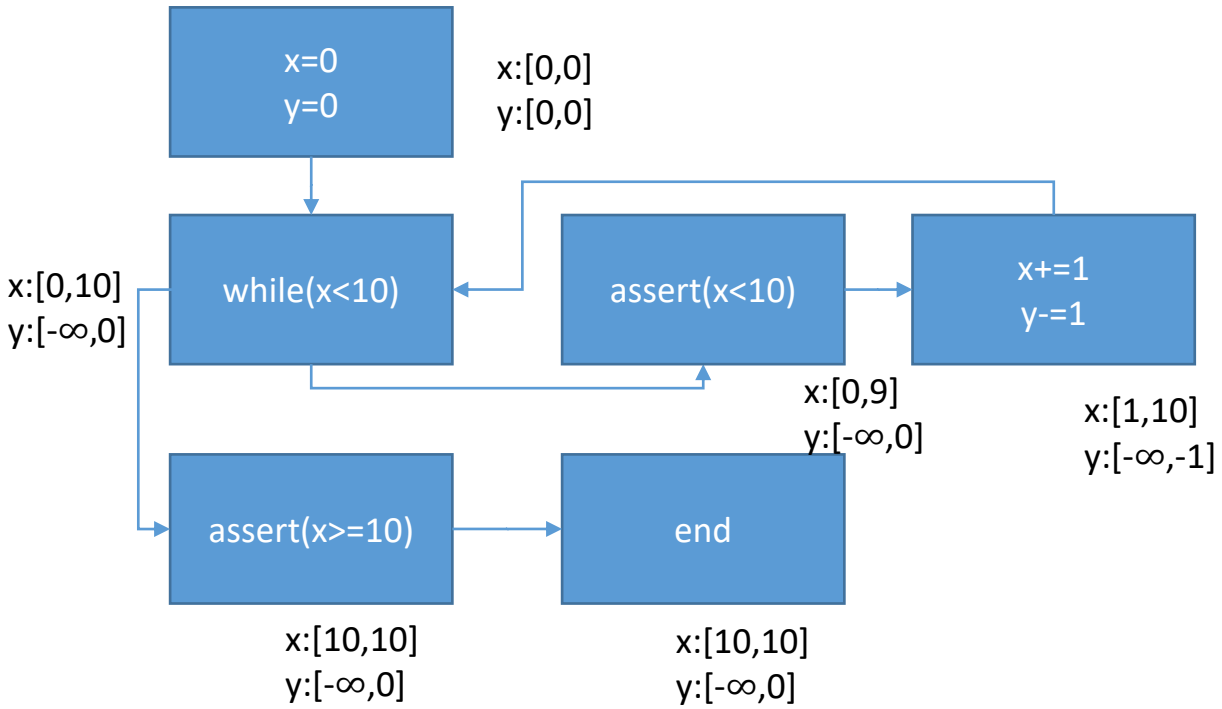
- 大量的分析是关于变量中有什么值的
  - 指向分析
  - 区间分析
  - 符号分析
- 这些分析单独对每个变量进行抽象，不考虑变量之间的关系。
- 这类不考虑变量之间关系的抽象称为非关系抽象。

非关系抽象的问题

- 考虑区间分析
  - x:[0, 1]
  - a=x;
b=x;
c=a-b;
- 区间分析结果：
  - c:[-1, 1]
- 精确结果：
  - c:[0, 0]
- 如果知道a和b相等，我们就能根据a-b推出这一精确结果

非关系抽象的问题

- 考虑区间分析
  - x=0;
    y=0;
    while (x<10){
      x++; y--; }
- 区间分析结果：
  - x:[10, 10]
  - y:[-∞, 0]
- 精确结果：
  - x:[10, 10]
  - y:[-10, -10]
- 如果知道a=-b相等，我们就能推出y的精确范围



关系抽象

- 考虑变量之间关系的抽象称为关系抽象。
- 一种基础关系抽象的表示方式
  - 抽象域：一阶逻辑表达式组成的空间
  - 抽象值c_1和c_2的并：$c_1 \vee c_2$
  - 语句的转换函数，如x=a+b：
    - $OUT = (\exists x.IN) \wedge x = a + b$
  - 条件的压缩函数，如x>0
    - $OUT = IN \wedge x > 0$
- 问题：
  - 抽象域无限，也很难定义出widen算子
    - 分析不收敛
  - 分析结果是一个巨大的逻辑表达式，难以从中导出
    - 逻辑表达式基本和原程序等价，分析难度几乎没变

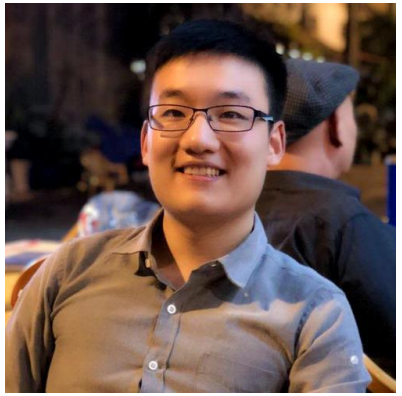
关系抽象

- 实际中的关系抽象通常限定逻辑约束的形式
  - 使得分析可以收敛
  - 使得可以从逻辑约束中导出结果

- 本课介绍两种关系抽象
  - 简单仿射关系抽象
  - 八边形抽象
  - 谓词抽象

简单仿射关系抽象

- 简单仿射关系抽象是对区间抽象的一种改进
  - 仿射关系=线性关系，来源于线性代数的仿射变换
  - 由北大张煜皓同学等在分析神经网络时提出
  - 是完整仿射关系抽象的化简版本
    - 完整版本涉及较多线性代数知识，可参考原始论文
- 通过记录变量和抽象符号之间的线性等价关系计算更精确的区间



北京大学张煜皓
简单仿射关系抽象提出者

	仿射关系	抽象符号区间	推导出的区间
	x=s	s:[0, 1]	x:[0, 1]
a=x;	a=s, x=s	s:[0, 1]	a:[0, 1]
b=x;	b=s, a=s, x=s	s:[0, 1]	b:[0, 1]
c=a-b;	c=s-s=0, …	s:[0, 1]	c:[0, 0]

简单仿射关系抽象—抽象域

- 给定抽象符号集合$\{s_{v,x} \mid v$: 控制流节点, x: 变量名$\}$，抽象域由(申, 酉)组成
  - 申：从变量到抽象符号线性表达式$\Sigma w_{ij} s_{ij}$的部分映射
  - 酉：从抽象符号到区间的部分映射，未定义的符号默认为空区间
- 抽象域的含义
  - 变量之间的关系满足申，且变量的值在基于酉和申计算出的区间内

简单仿射关系抽象转换函数

- 赋值语句x=a+b
  - $OUT^{申}(x) = IN^{申}(a) + IN^{申}(b)$
  - $OUT^{申}(y) = IN^{申}(y) \quad \forall y \neq x$
  - $OUT^{酉} = IN^{酉}$
- 节点v的赋值语句x=a*b
  - $OUT^{申}(x) = s_{vx}$
  - $OUT^{酉}(s_{vx}) =$ 根据a和b的区间计算
  - 其他不变
- 节点v的IN的计算
  - $IN^{申}(x) = \begin{cases} e & \text{如果所有前驱都为}e\text{或未定义} \\ s_{vx} & \text{否则} \end{cases}$
  - $IN^{酉}(s_{ij}) = \begin{cases} \text{前驱对应区间的并} & i \neq v \\ \text{根据前驱对应申和酉计算} & i = v \end{cases}$

简单仿射关系抽象安全性和收敛性

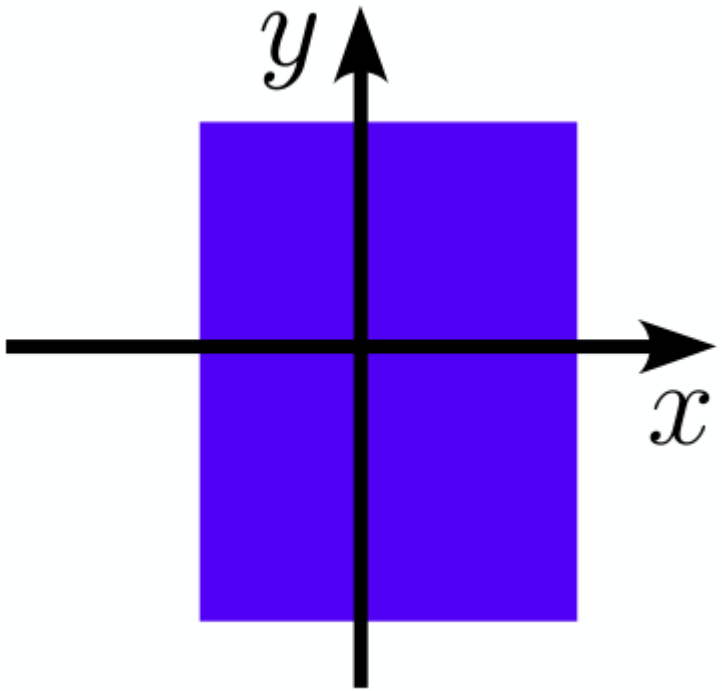
- 安全性：简单仿射关系抽象保证安全性
- 收敛性：
  - 节点v的申对于变量x的值只会按这样的链变化：未定义，特定表达式，S_{vx}
  - 但酉中的区间和区间分析类似，可以一直增加
  - 可以对酉中的区间加上加宽来保证收敛

简单仿射关系抽象的不足

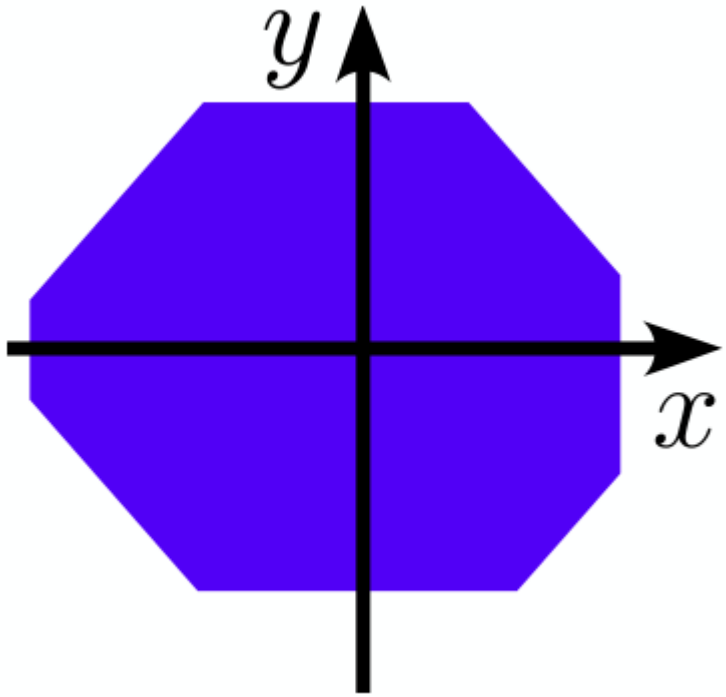
- 仍然不能解决第二个程序的问题
  - x=0;
    y=0;
    while (x<10){
      x++; y--; }
- x和y之间没有互相赋值，无法建立起关系

关系抽象举例：八边形

假设程序中只有x和y两个变量



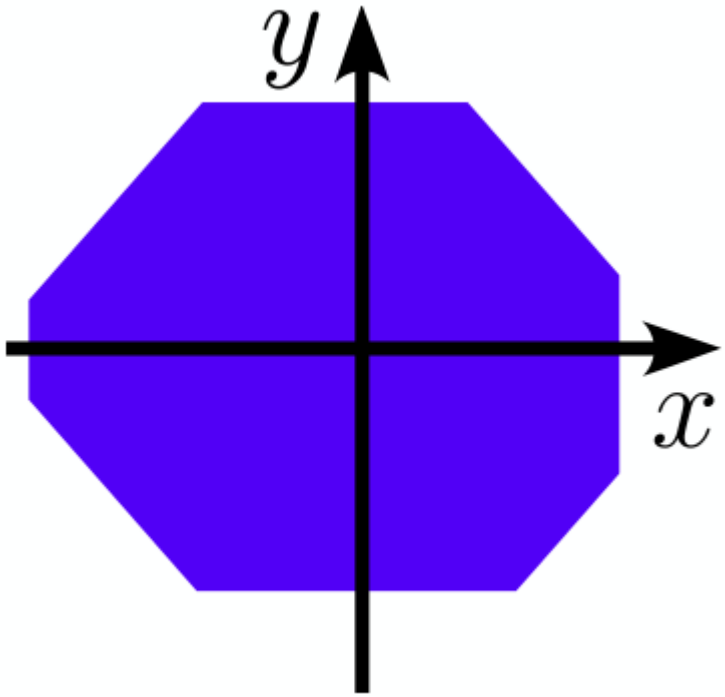
区间抽象形成一个矩形



加上4条45度的线来形成八边形

关系抽象举例：八边形

假设程序中只有x和y两个变量



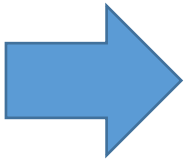
加上4条45度的线来形成八边形

x的上界a_1：$x \leq a_1$
x的下界a_2：$x \geq a_2$
y的上界a_3：$y \leq a_3$
y的下界a_4：$y \geq a_4$
x+y的上界a_5：$x + y \leq a_5$
x+y的下界a_6：$x + y \geq a_6$
x-y的上界a_7：$x - y \leq a_7$
x-y的下界a_8：$x - y \geq a_8$

关系抽象举例：八边形

x的上界a_1：$x \le a_1$
x的下界a_2：$x \ge a_2$
y的上界a_3：$y \le a_3$
y的下界a_4：$y \ge a_4$
x+y的上界a_5：$x + y \le a_5$
x+y的下界a_6：$x + y \ge a_6$
x-y的上界a_7：$x - y \le a_7$
x-y的下界a_8：$x - y \ge a_8$

统一化



x的上界$\frac{1}{2}a_1$：$+x + x \le a_1$
x的下界$-\frac{1}{2}a_2$：$-x - x \le a_2$
y的上界$\frac{1}{2}a_3$：$+y + y \le a_3$
y的下界$-\frac{1}{2}a_4$：$-y - y \le a_4$
x+y的上界a_5：$+x + y \le a_5$
x+y的下界$-a_6$：$-x - y \le a_6$
x-y的上界a_7：$+x - y \le a_7$
x-y的下界$-a_8$：$-x + y \le a_8$

即$\pm v_1 \pm v_2 \le a$，其中$v_1, v_2 \in \{x, y\}$

对多个变量进行抽象

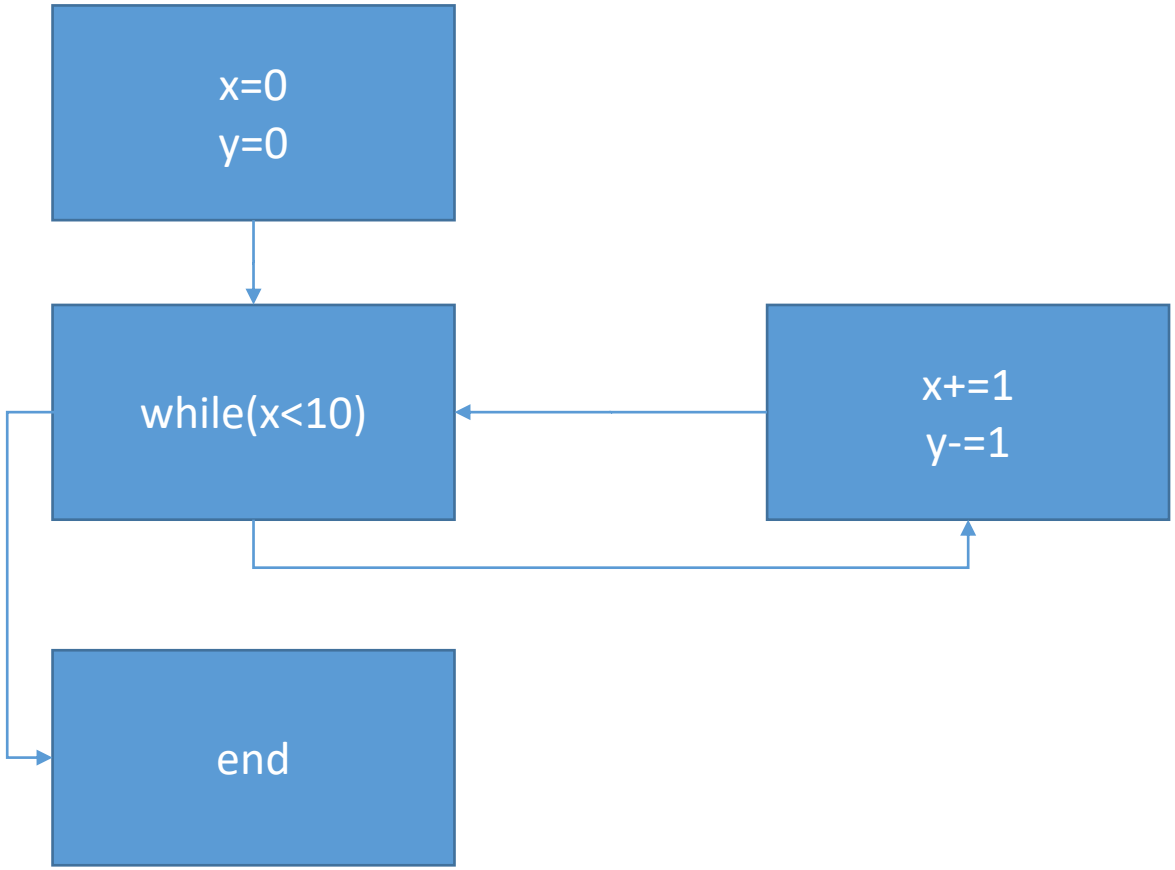
- 对任意两个变量记录八边形
- $\pm v_1 \pm v_2 \leq a$，其中v_1, v_2为程序上的任意变量
- 即对任意两个变量记录八个值

	+x	-x	+y	-y
+x	10	-	-	-
-x	-	0	-	-
+y	10	5	20	-
-y	-2	5	-	-10

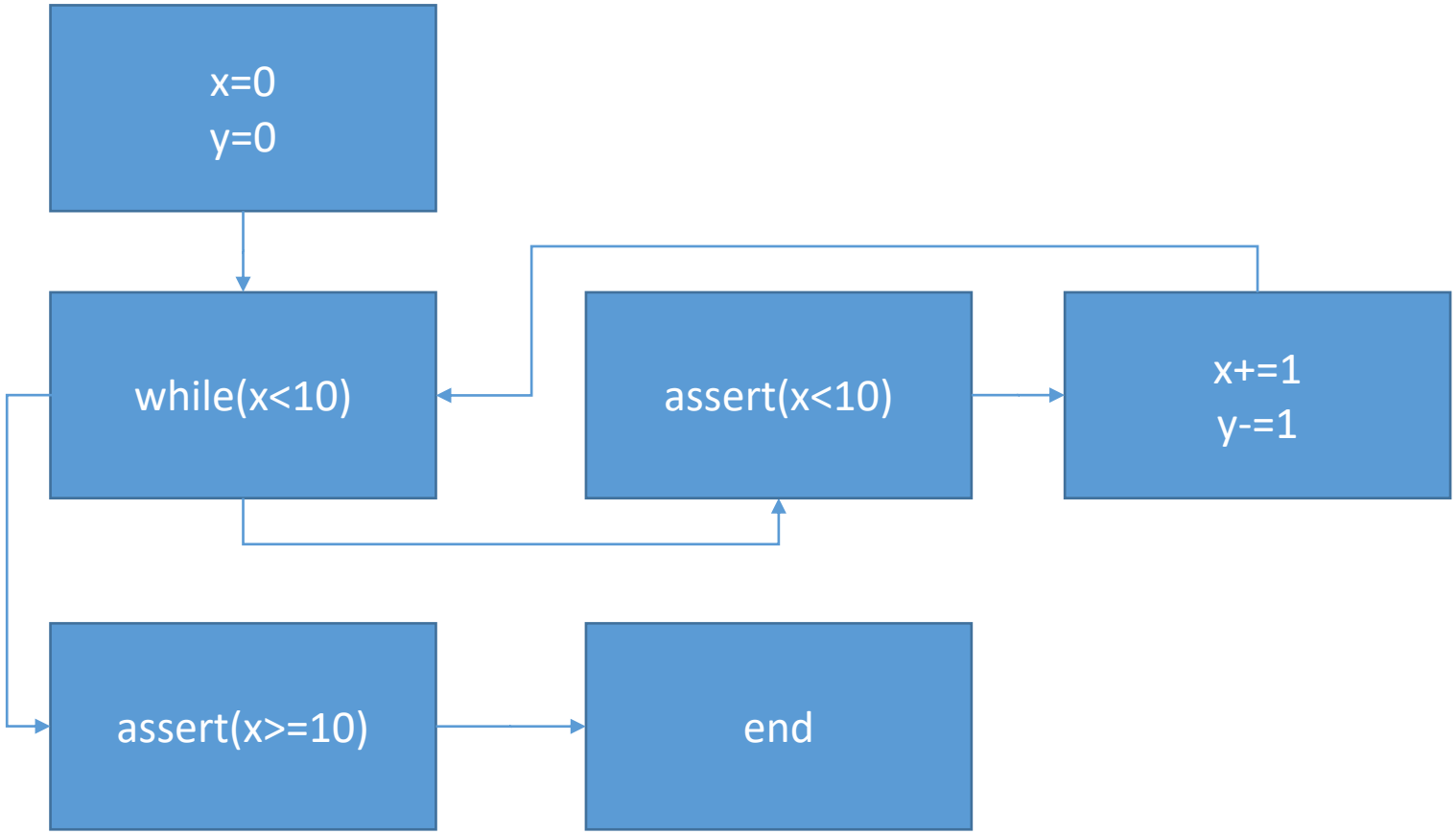
八边形上的计算

- x=x+1
  - 将x有关的八边形沿x轴移动1个单位
- $z = x \cup y$
  - 对于任意变量v，令<z,v>的八边形为包住<x,v>和<y,v>的最小八边形
- $z = x \cap y$
  - 对于任意变量v，令<z,v>的八边形为包住<x,v>和<y,v>公共部分的最小八边形
- 更多计算方法参考原始论文：
  - Miné A. The octagon abstract domain[J]. Higher-Order and Symbolic Computation, 2006, 19(1):31-100.

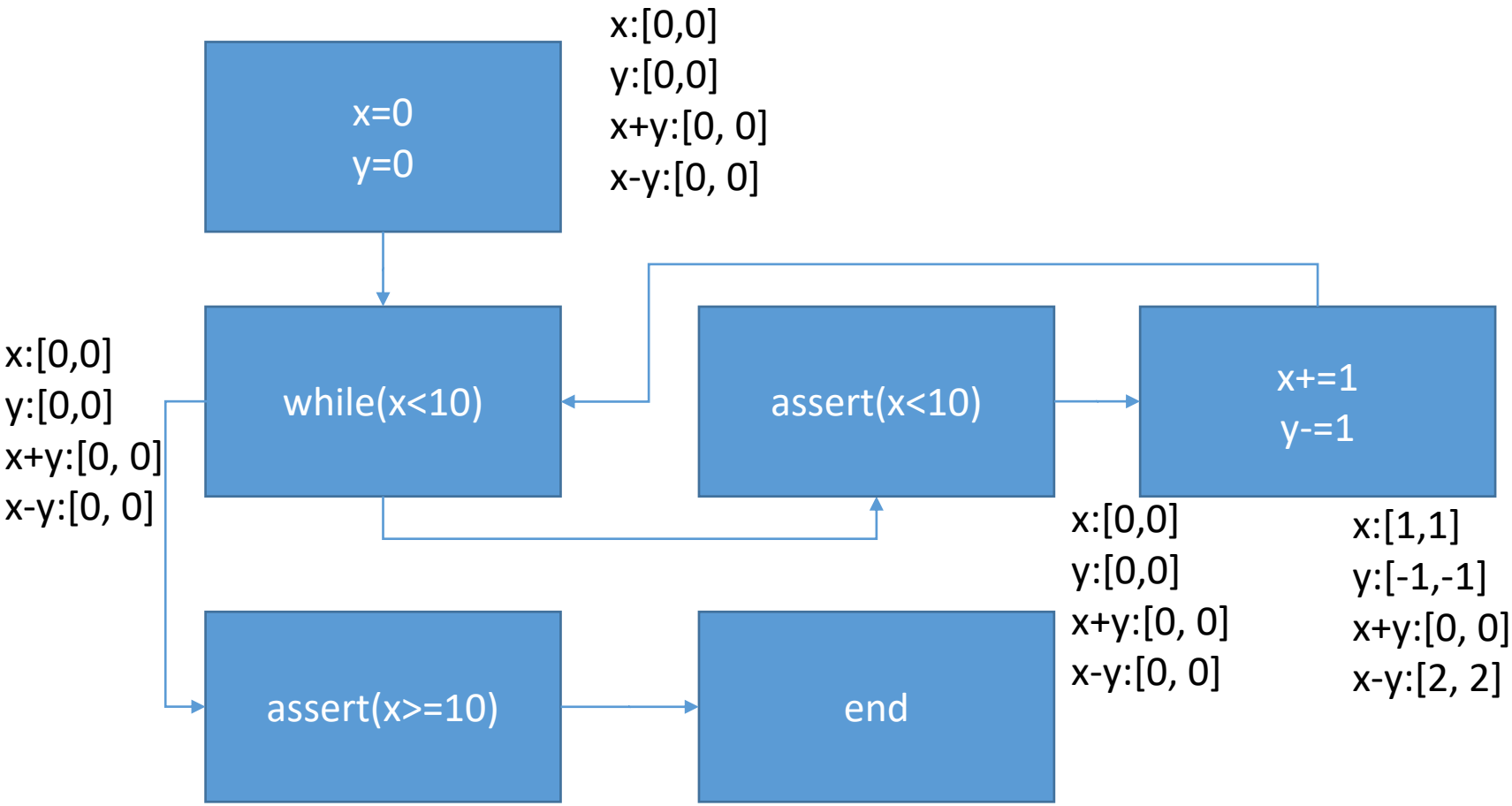
八边形计算举例



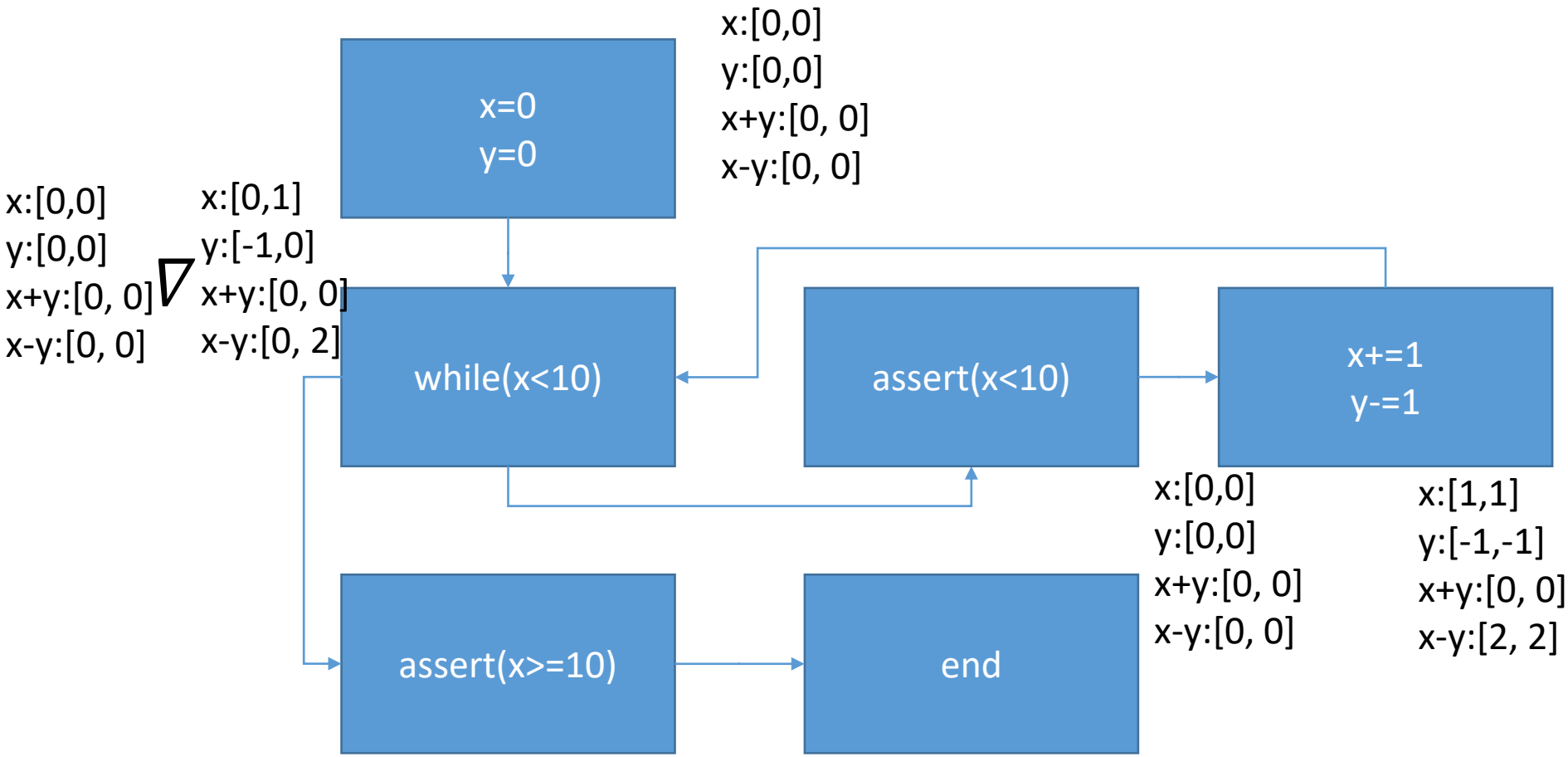
八边形计算举例



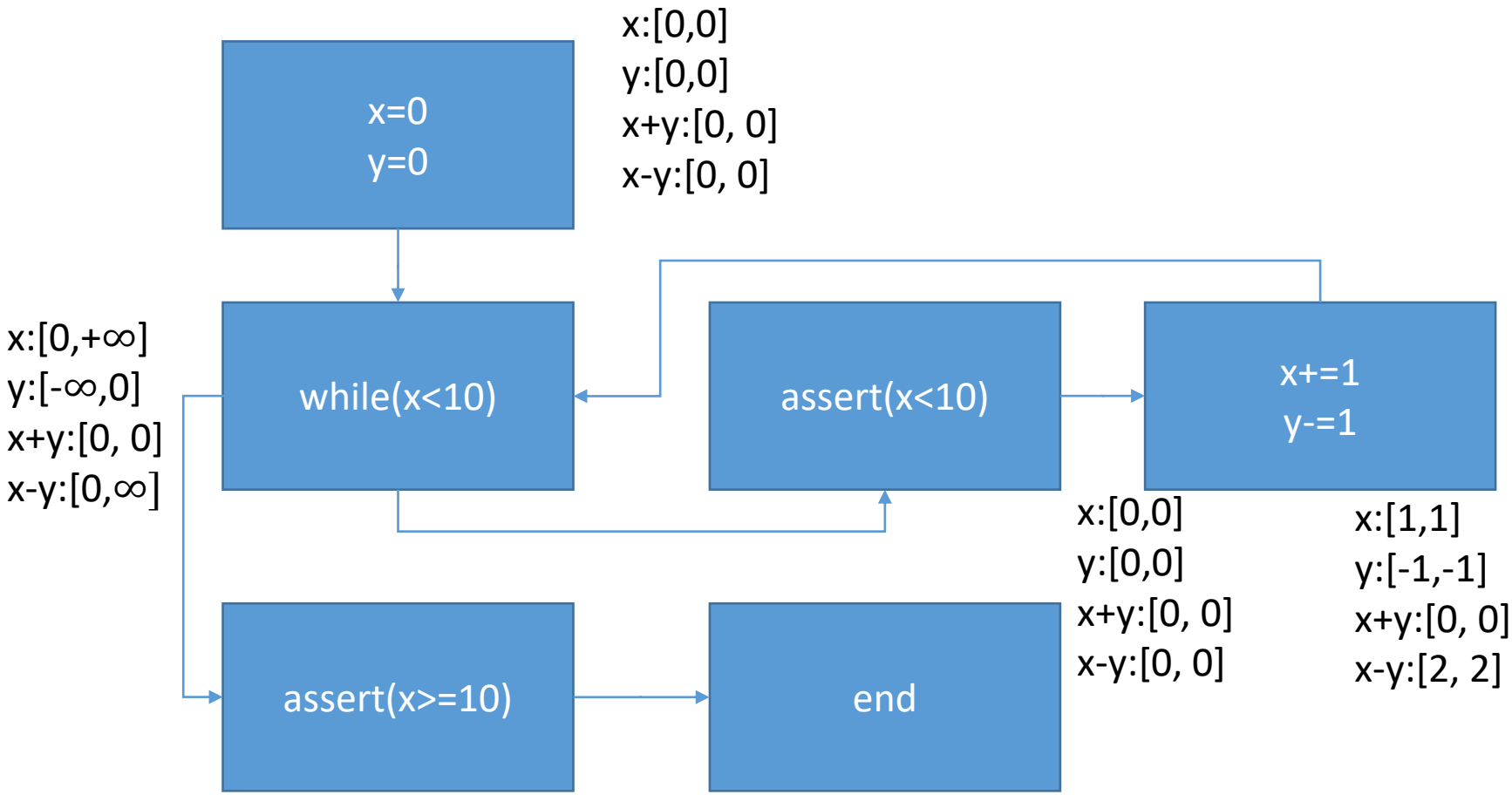
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
while(x<10)
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
assert(x<10)
x+=1
y-=1
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[1,1]
y:[-1,-1]
x+y:[0, 0]
x-y:[2, 2]
assert(x>=10)
end

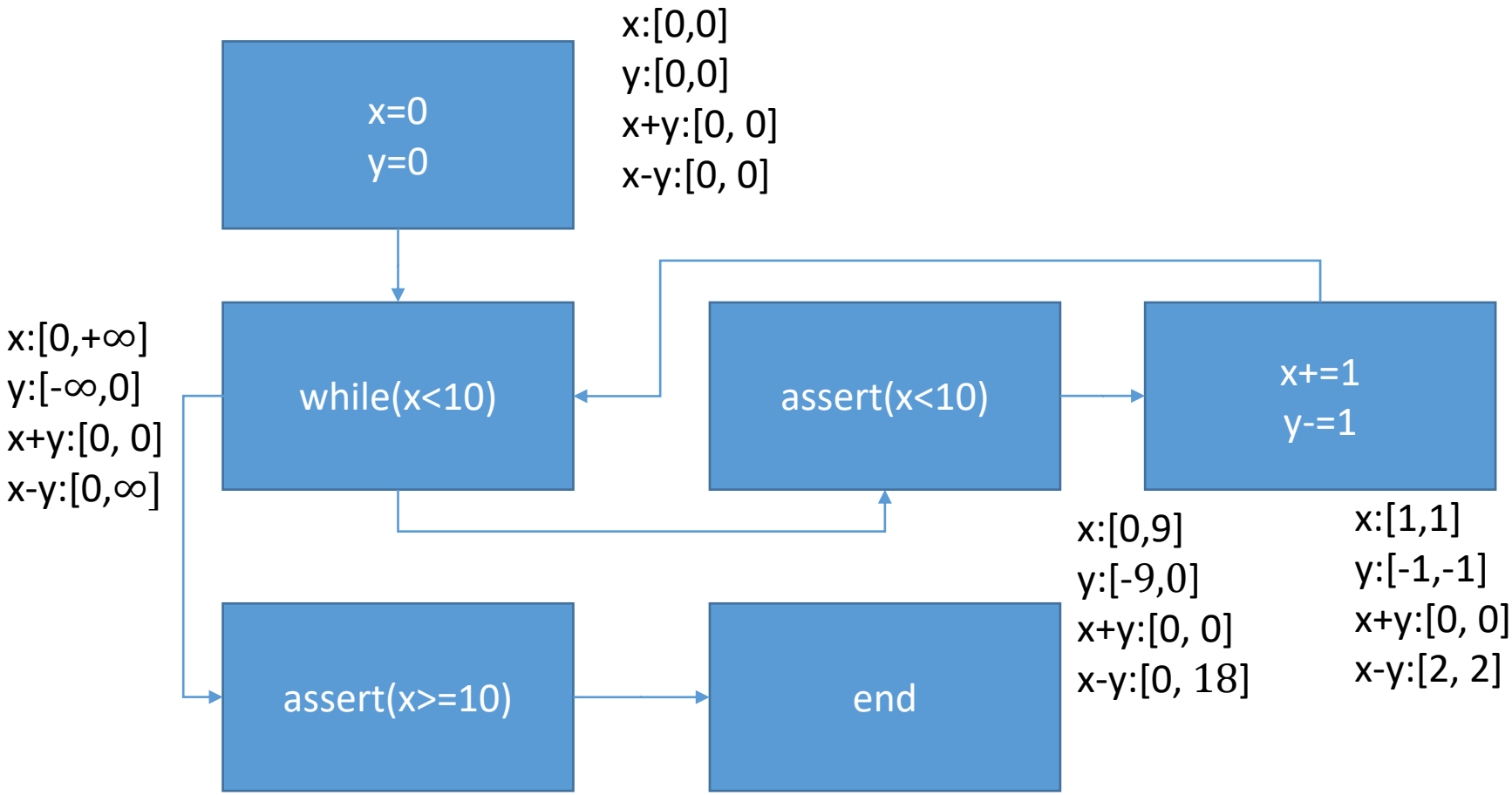
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[0,1]
y:[-1,0]
x+y:[0, 0]
x-y:[0, 2]
while(x<10)
assert(x<10)
x+=1
y-=1
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[1,1]
y:[-1,-1]
x+y:[0, 0]
x-y:[2, 2]
assert(x>=10)
end

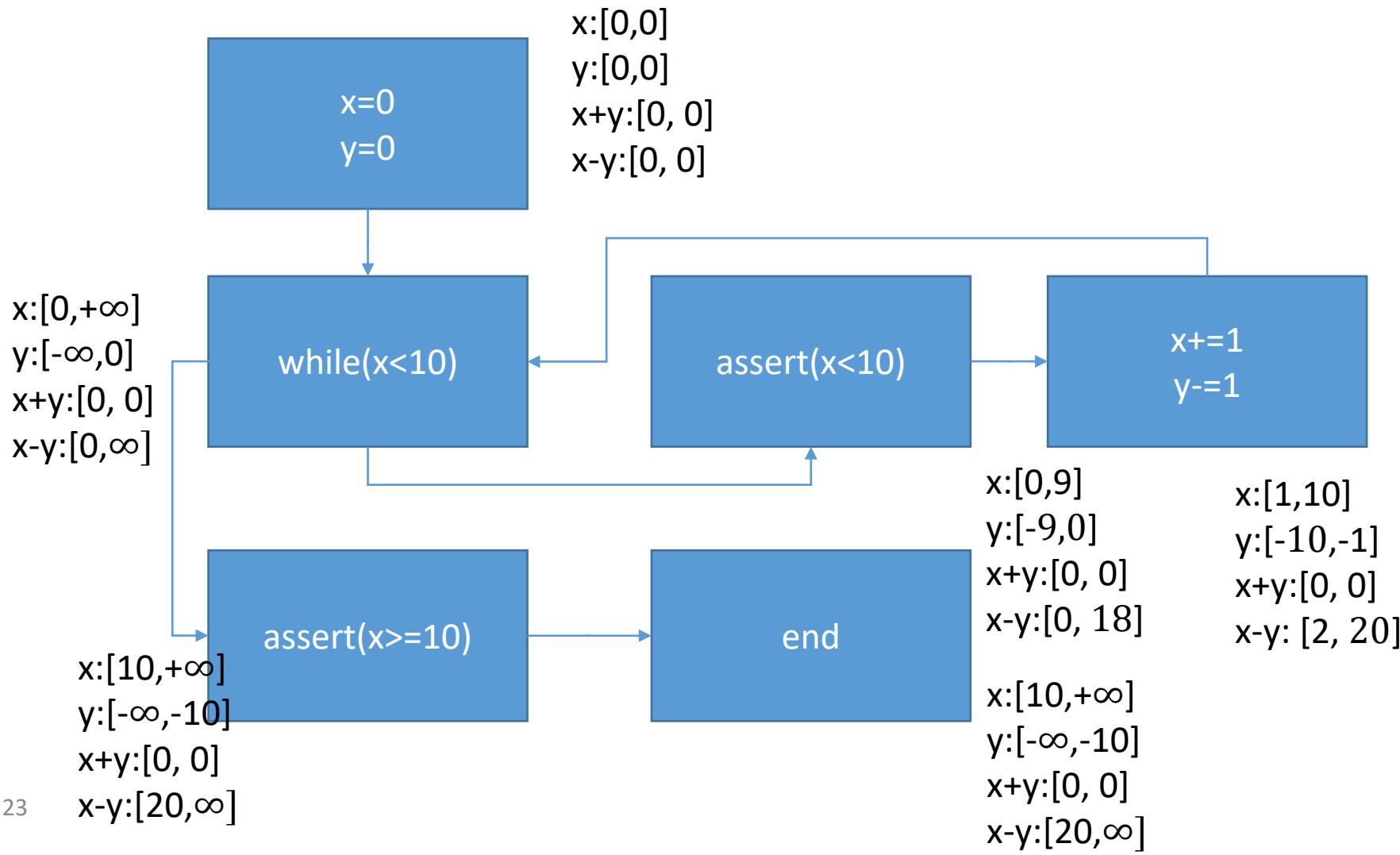
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[0,+∞]
y:[-∞,0]
x+y:[0, 0]
x-y:[0,∞]
while(x<10)
assert(x<10)
x+=1
y-=1
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[1,1]
y:[-1,-1]
x+y:[0, 0]
x-y:[2, 2]
assert(x>=10)
end

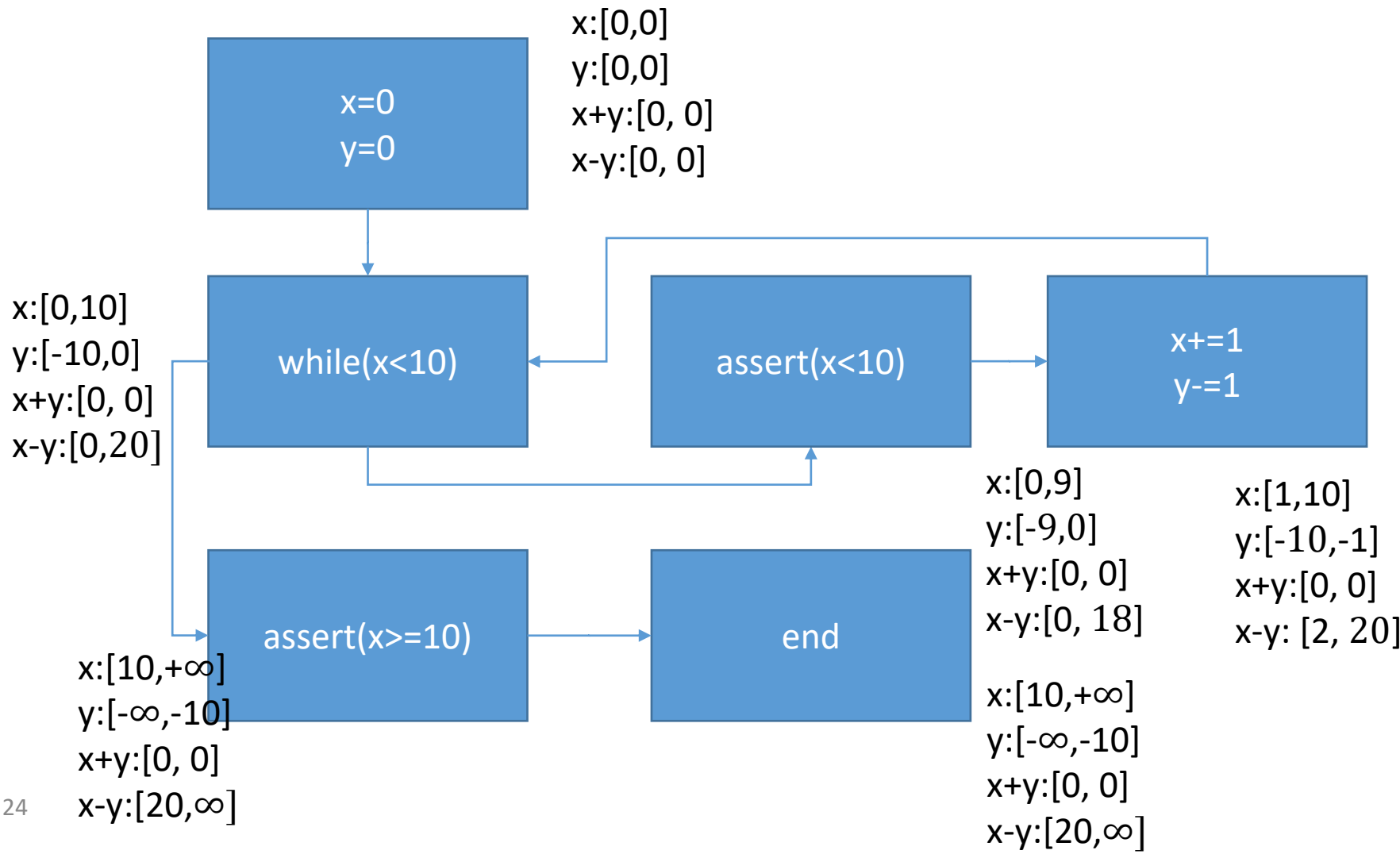
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
x:[0,+∞]
y:[-∞,0]
x+y:[0, 0]
x-y:[0,∞]
while(x<10)
assert(x<10)
x+=1
y-=1
x:[0,9]
y:[-9,0]
x+y:[0, 0]
x-y:[0, 18]
x:[1,1]
y:[-1,-1]
x+y:[0, 0]
x-y:[2, 2]
assert(x>=10)
end

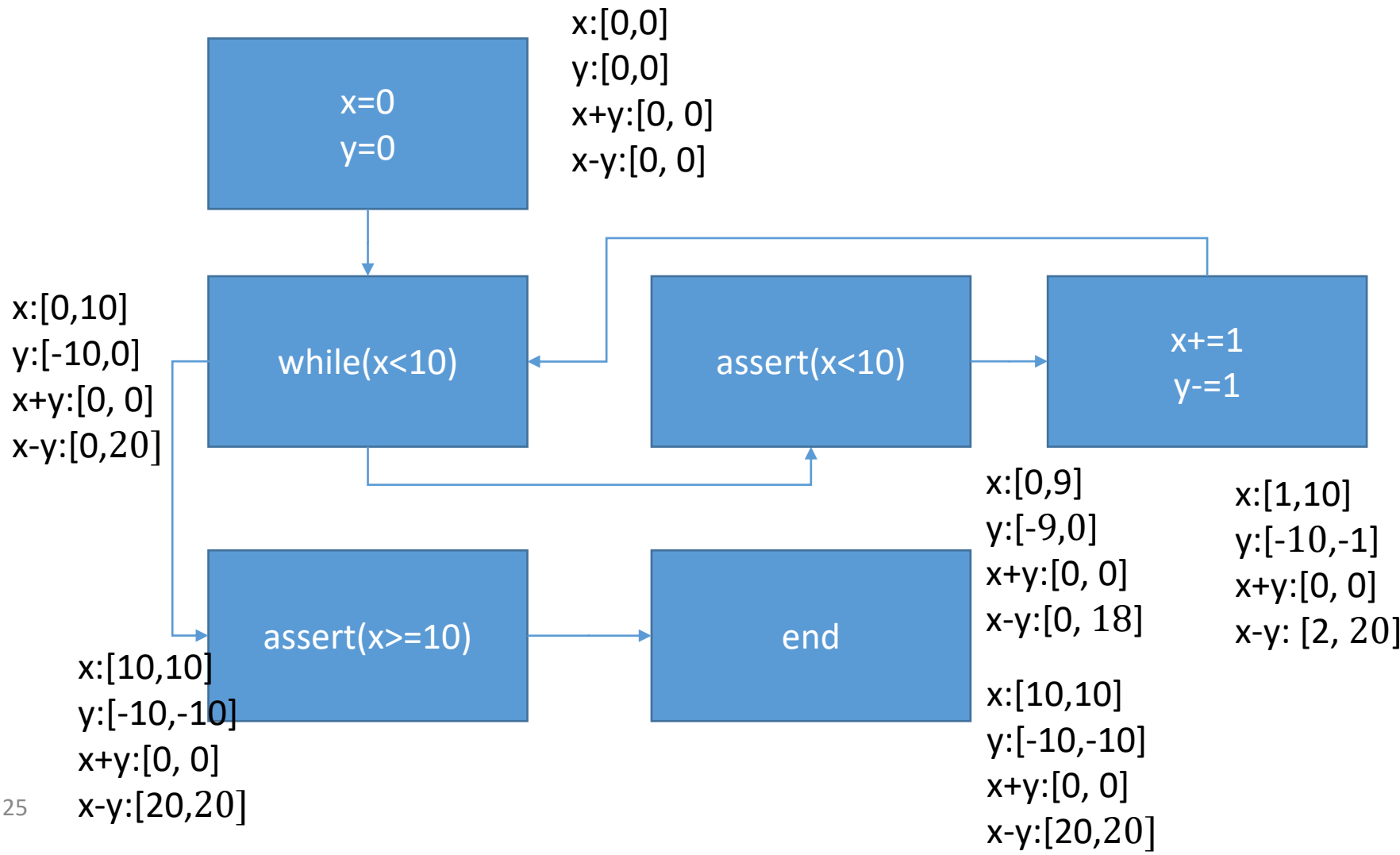
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
while(x<10)
x:[0,+∞]
y:[-∞,0]
x+y:[0, 0]
x-y:[0,∞]
assert(x<10)
x+=1
y-=1
x:[0,9]
y:[-9,0]
x+y:[0, 0]
x-y:[0, 18]
x:[1,10]
y:[-10,-1]
x+y:[0, 0]
x-y: [2, 20]
assert(x>=10)
x:[10,+∞]
y:[-∞,-10]
x+y:[0, 0]
x-y:[20,∞]
end
x:[10,+∞]
y:[-∞,-10]
x+y:[0, 0]
x-y:[20,∞]

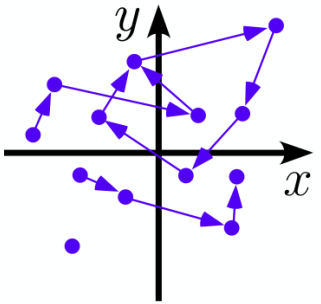
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
while(x<10)
x:[0,10]
y:[-10,0]
x+y:[0, 0]
x-y:[0,20]
assert(x<10)
x+=1
y-=1
x:[0,9]
y:[-9,0]
x+y:[0, 0]
x-y:[0, 18]
x:[1,10]
y:[-10,-1]
x+y:[0, 0]
x-y: [2, 20]
assert(x>=10)
x:[10,+∞]
y:[-∞,-10]
x+y:[0, 0]
x-y:[20,∞]
end
x:[10,+∞]
y:[-∞,-10]
x+y:[0, 0]
x-y:[20,∞]

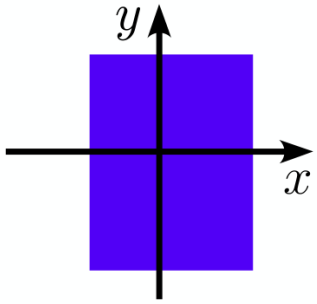
八边形计算举例


x=0
y=0
x:[0,0]
y:[0,0]
x+y:[0, 0]
x-y:[0, 0]
while(x<10)
x:[0,10]
y:[-10,0]
x+y:[0, 0]
x-y:[0,20]
assert(x<10)
x+=1
y-=1
x:[0,9]
y:[-9,0]
x+y:[0, 0]
x-y:[0, 18]
x:[1,10]
y:[-10,-1]
x+y:[0, 0]
x-y: [2, 20]
assert(x>=10)
x:[10,10]
y:[-10,-10]
x+y:[0, 0]
x-y:[20,20]
end
x:[10,10]
y:[-10,-10]
x+y:[0, 0]
x-y:[20,20]

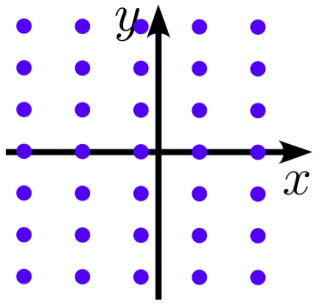
其他数值常用抽象



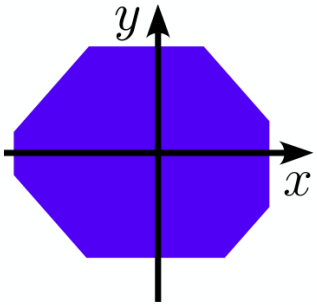
Collecting semantics:
partial traces



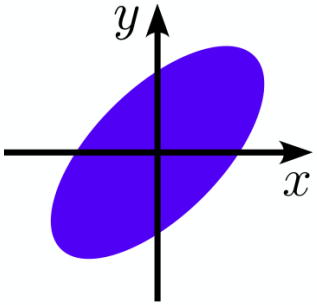
Intervals.
$\mathtt{x} \in [a, b]$



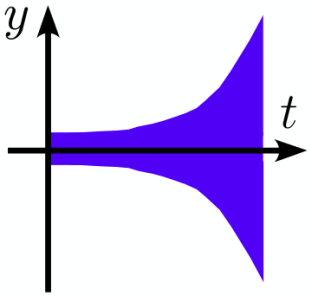
Simple congruences:
$\mathtt{x} \equiv a[b]$



Octagons:
$\pm\mathtt{x} \pm \mathtt{y} \leqslant a$



Ellipses.
$\mathtt{x}^2 + b\mathtt{y}^2 - a\mathtt{xy} \leqslant d$



Exponentials:
$-a^{bt} \leqslant \mathtt{y}(t) \leqslant a^{bt}$

谓词抽象

- 用一系列布尔表达式的值作为抽象域
- 其他很多抽象形式可以看做谓词抽象的一种
- 需要针对谓词设计转换函数
- 如，符号分析可以用谓词抽象表达
  - 对任意变量x，有如下谓词
    - $x > 0, x < 0, x = 0$

在线抽象解释工具

- Interproc
  - http://pop-art.inrialpes.fr/interproc/interprocweb.cgi
  - 开源工具
    - 用于展示开源抽象域库APRON的静态分析工具
    - 支持整型、浮点型等运算的分析
    - 支持过程间分析（包括递归函数）
    - 不支持数组、结构体等复杂数据结构、也不支持动态内存分配等

Interproc Analyzer

http://pop-art.inrialpes.fr/interproc/interprocweb.cgi

Google

The Interproc Analyzer

This is a web interface to the Interproc analyzer connected to the APRON Abstract Domain Library and the Fixpoint Solver Library, whose goal is to demonstrate the features of the APRON library and, to a less extent, of the Analyzer fixpoint engine, in the static analysis field.

There are two compiled versions: interprocweb, in which all the abstract domains use underlying multiprecision integer/rational numbers, and interprocwebf, in which box and octagon domains use underlying floating-point numbers in safe way.

This is the **Interproc** version

Arguments

Please type a program, upload a file from your hard-drive, or choose one the provided examples:

Choose File no file selected

Mac Carthy 91

```
/* type your program here ! */
```

Numerical Abstract Domain: convex polyhedra (polka)

Kind of Analysis: f (sequence of forward and/or backward analysis)

Iterations/Widening options:

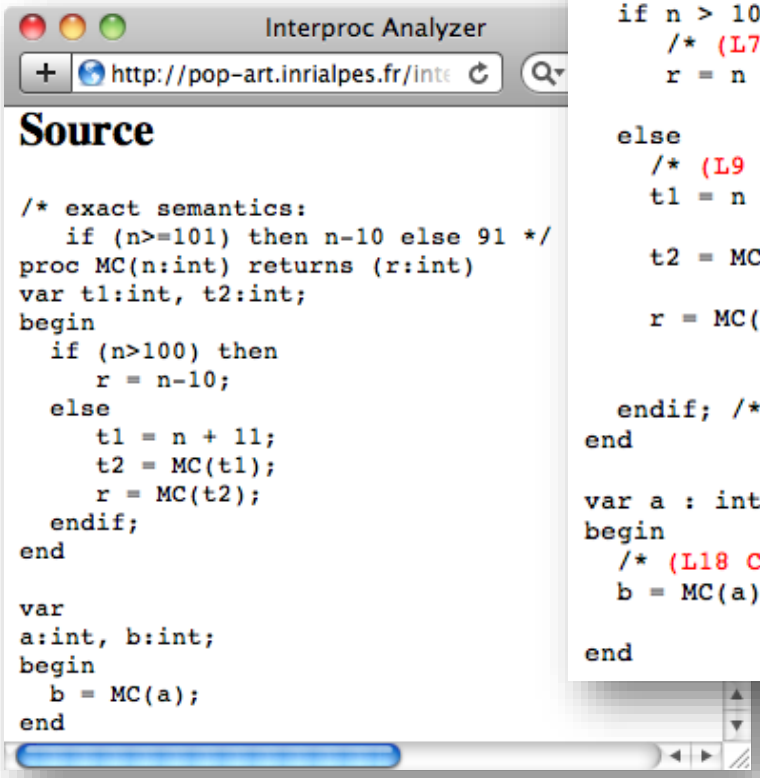
guided iterations 1 widening delay 2 descending steps

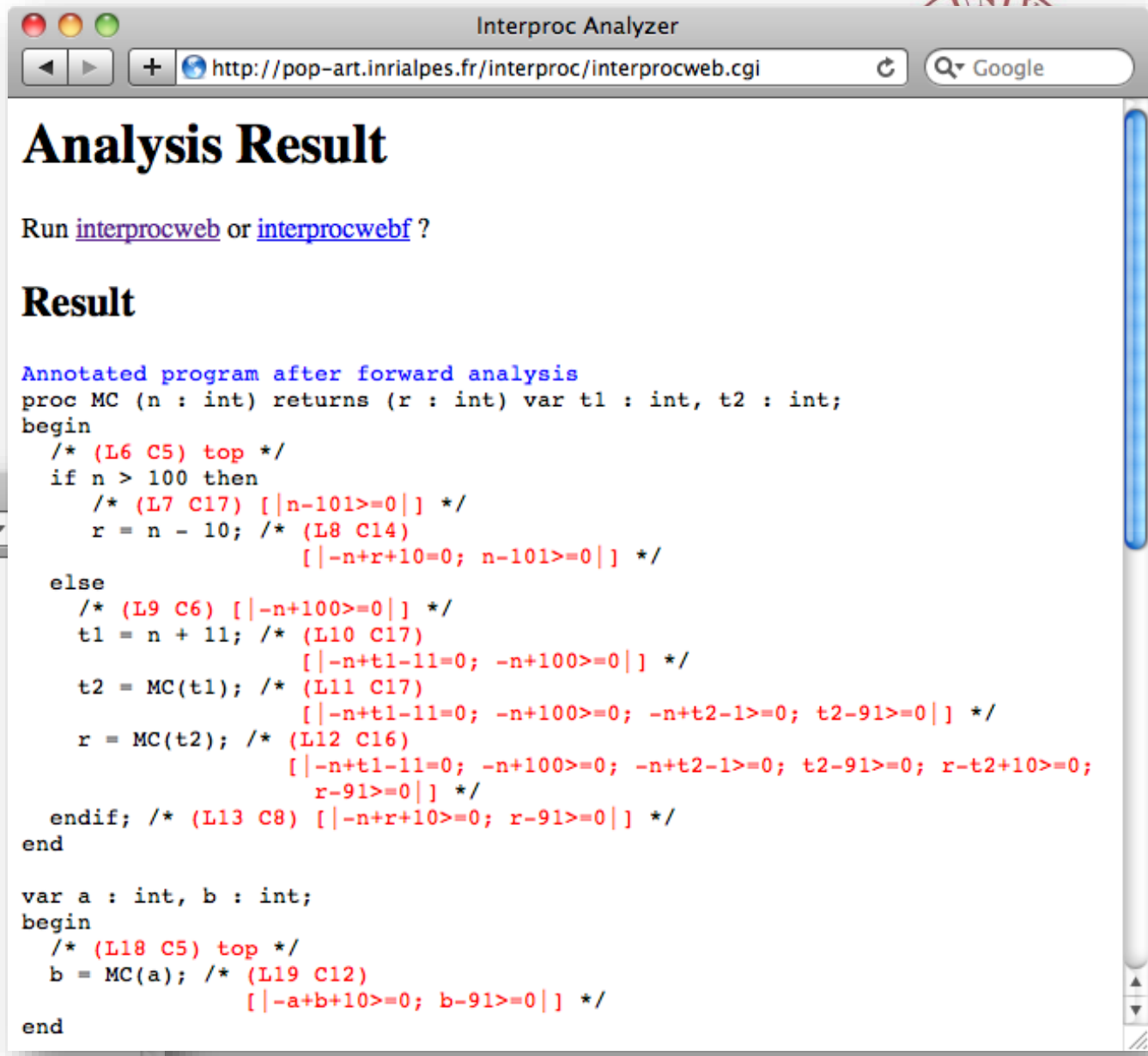
0 debugging level (0 to 6)

Hit the OK button to proceed: OK ! Reset

可选择APRON中的抽象域

Choose an Abstract Domain:
box
box with policy iteration
octagon
✓ convex polyhedra (polka)
convex polyhedra (PPL)
strict convex polyhedra (polka)
strict convex polyhedra (PPL)
linear equalities (polka)
linear congruences (PPL)
convex polyhedra + linear congruences


Interproc Analyzer
http://pop-art.inrialpes.fr/int
Source
/* exact semantics:
if (n>=101) then n-10 else 91 */
proc MC(n:int) returns (r:int)
var t1:int, t2:int;
begin
if (n>100) then
r = n-10;
else
t1 = n + 11;
t2 = MC(t1);
r = MC(t2);
endif;
end
var
a:int, b:int;
begin
b = MC(a);
end


Interproc Analyzer
http://pop-art.inrialpes.fr/interproc/interprocweb.cgi
Google
Analysis Result
Run interprocweb or interprocwebf ?
Result
Annotated program after forward analysis
proc MC (n : int) returns (r : int) var t1 : int, t2 : int;
begin
/* (L6 C5) top */
if n > 100 then
/* (L7 C17) [|n-101>=0|] */
r = n - 10; /* (L8 C14)
[|-n+r+10=0; n-101>=0|] */
else
/* (L9 C6) [|-n+100>=0|] */
t1 = n + 11; /* (L10 C17)
[|-n+t1-11=0; -n+100>=0|] */
t2 = MC(t1); /* (L11 C17)
[|-n+t1-11=0; -n+100>=0; -n+t2-1>=0; t2-91>=0|] */
r = MC(t2); /* (L12 C16)
[|-n+t1-11=0; -n+100>=0; -n+t2-1>=0; t2-91>=0; r-t2+10>=0;
r-91>=0|] */
endif; /* (L13 C8) [|-n+r+10>=0; r-91>=0|] */
end
var a : int, b : int;
begin
/* (L18 C5) top */
b = MC(a); /* (L19 C12)
[|-a+b+10>=0; b-91>=0|] */
end

作业

- 在Interproc中构造一个程序，使得：
  - 八边形抽象的结果精度>区间抽象的结果精度
  - 最理想的结果精度>八边形抽象的结果精度

- 提交：
  - Interproc的运行截图
  - 解释你的结果，包括
    - 最理想的结果精度是什么
    - 为什么八边形的结果不如最理想结果精确
    - 为什么区间抽象的结果不如八边形的结果精确

参考资料

- 简单仿射关系抽象： Yuhao Zhang, Luyao Ren, Liqian Chen, Yingfei Xiong, Shing-Chi Cheung, Tao Xie. Detecting Numerical Bugs in Neural Network Architectures. ESEC/FSE'20: ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, November 2020.
- 仿射关系抽象： Michael Karr. Affine Relationships Among Variables of a Program. Acta Informatica 6, 133--151 (1976)
- 八边形抽象： Miné A. The octagon abstract domain[J]. Higher-Order and Symbolic Computation, 2006, 19(1):31-100.