软件分析

抽象解释

熊英飞
北京大学

胶片部分参考国防科大陈立前老师的报告

复习：符号分析

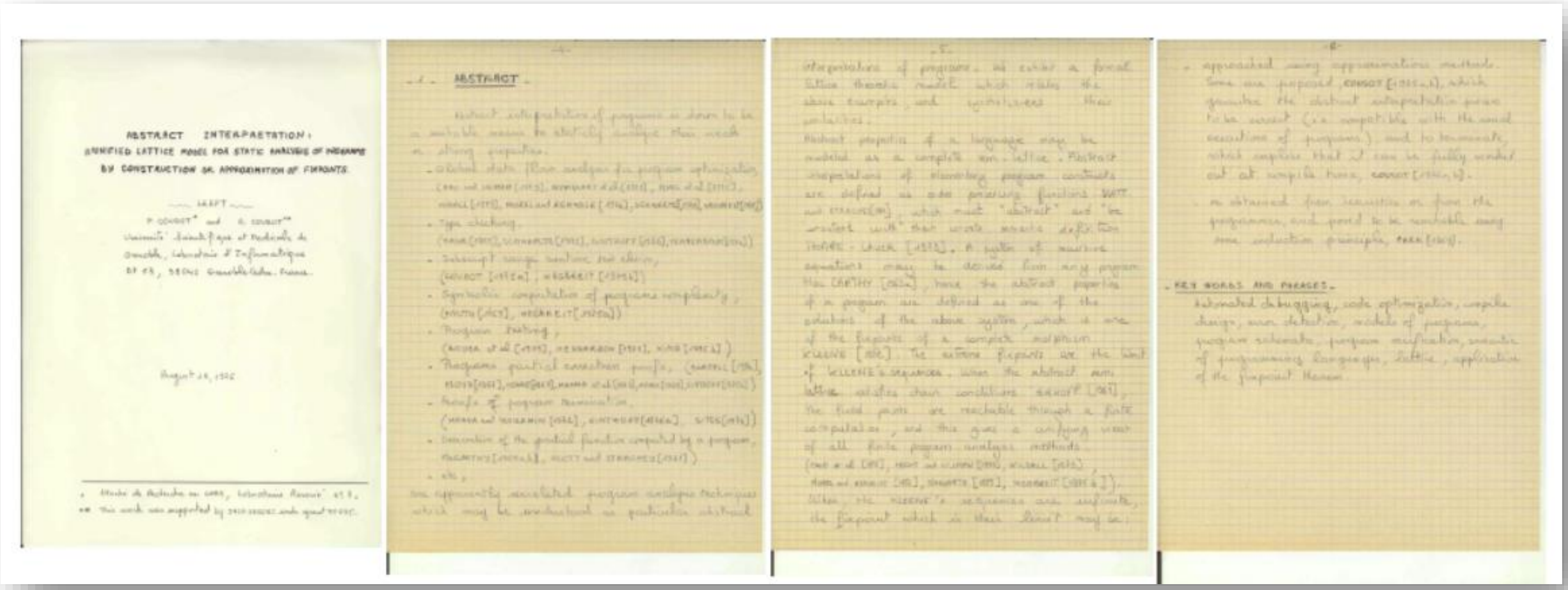
- 正 ={所有的正数}
- 零={0}
- 负= {所有的负数}

如何知道该抽象运算是正确的？

- 乘法运算规则：
  - 正*正=正
  - 正*零=零
  - 正*负=负
  - 负*正=负
  - 负*零=零
  - 负*负=正
  - 零*正=零
  - 零*零=零
  - 零*负=零

抽象解释

- 最早发表于POPL’77（手写的100页论文）


ABSTRACT INTERPRETATION: A UNIFIED LATTICE MODEL FOR STATIC ANALYSIS OF PROGRAMS BY CONSTRUCTION OR APPROXIMATION OF FIXPOINTS

ABSTRACT

KEY WORDS AND PHRASES

所获荣誉

2013年ACM SIGPLAN 程序语言成就奖

Report Problem   Contact Us

SIGPLAN

To explore programming language concepts and tools focusing on design, implementation and efficient use.

Home | Awards | Conferences | Resources | Membership | SIGPLAN Research Highlights | Student Information | Publications | Announcement

Programming Languages Achievement Award

Given by ACM SIGPLAN to recognize an individual or individuals who has made a significant and lasting contribution to the field of programming languages. The contribution can be a single event or a life-time of achievement. The award includes a prize of $5,000. The award is presented at SIGPLAN's PLDI conference the following June.

Nominations

- Details of the nomination and award process (pdf).
- Please use http://awards.sigplan.org/ to submit nominations.

Recipients of the Achievement Award

2013: Patrick and Radhia Cousot

Patrick and Radhia Cousot are the co-inventors of abstract interpretation, a unifying theory of sound abstraction and approximation of structures involved in various domains of computer science, such as formal semantics, specification, proof, and verification. In particular, abstract interpretation has had a major impact on the development of the static analysis of software. In their original work, the Cousots showed how to relate a static analysis to a language's standard semantics by means of a second, abstract semantics that makes precise which features of the full language are being modeled and which are being discarded (or abstracted), providing for the first time both a formal definition of and clear methodology for designing and proving the correctness of static analyses. Subsequently, the Cousots contributed many of the building blocks of abstract interpretation in use today, including chaotic iteration, widening, narrowing, combinations of abstractions, and a number of widely used abstract domains. This work has developed a remarkable set of intellectual tools and has found its way into practice in the form of widely used libraries and frameworks. Finally, the Cousots and their collaborators have contributed to demonstrating the utility of static analysis to society. They led the development of the AstrÈe static analyzer, which is used in the medical, automotive, and aerospace industry for verifying the absence of a large class of common programming errors in low-level embedded systems code. This achievement stands as one of the most substantial successes of program verification to date.

所获荣誉

2018年 约翰·冯诺依曼奖

Patrick Cousot awarded John von Neumann Medal

Patrick Cousot is the recipient of the IEEE John von Neumann medal, given "for outstanding achievements in computer-related science and technology".

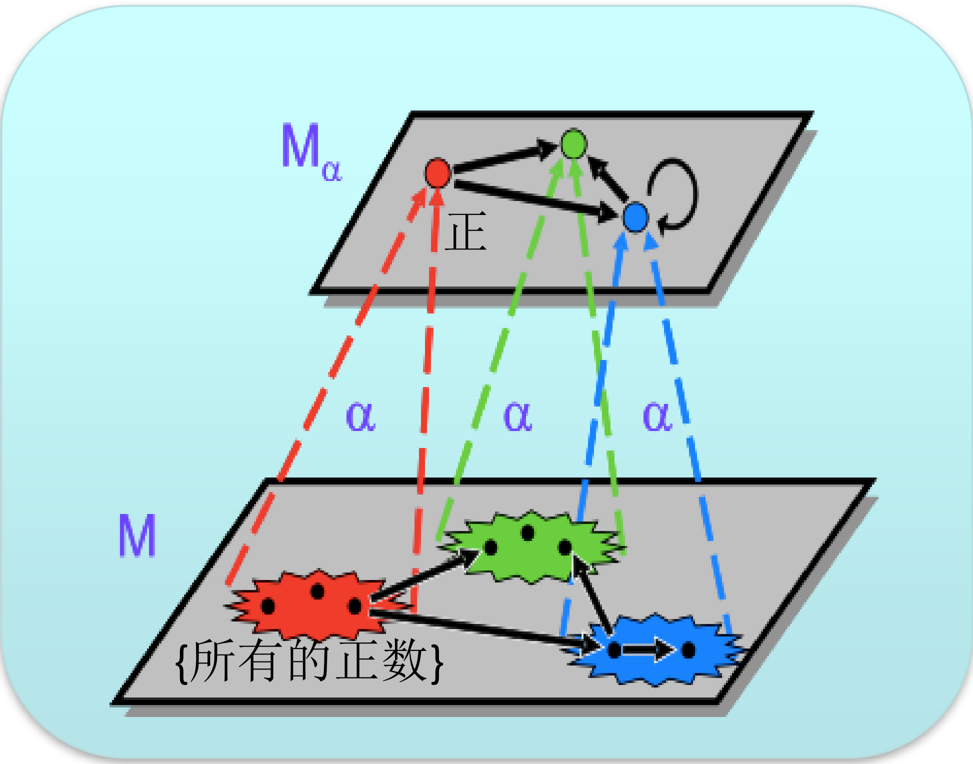
Read More

IEEE JOHN VON NEUMANN MEDAL
RECIPIENTS

2018	PATRICK COUSOT Professor, New York University, New York, New York, USA	"For introducing abstract interpretation, a powerful framework for automatically calculating program properties with broad application to verification and optimization."

抽象解释

- 主要解释抽象空间和具体空间的关系



抽象空间

具体空间

抽象解释

- 具体化函数γ将抽象值映射为具体值的集合
  - γ(正) ={所有的正数}
  - $\gamma(\top) = \emptyset$
- 抽象化函数α将具体值的集合映射为抽象值
  - α({所有的正数})=正
  - $\alpha(\{1,2\}) =$ 正
  - $\alpha(\{-1,0\}) =$ 槑
- 假设抽象域上存在偏序关系$\sqsubseteq$
  - 简便起见，这里假设具体值集合上的偏序关系为子集关系。但抽象解释理论支持其他偏序关系，比如超集。

伽罗瓦连接 Galois Connection

- 我们称γ和α构成抽象域虚和具体域集合的全集$\boldsymbol{D}$之间的一个伽罗瓦连接，记为

$$(\boldsymbol{D}, \subseteq) \leftrightarrows_{\alpha}^{\gamma} (\text{虚}, \sqsubseteq)$$

- 当且仅当

$$\forall X \in \boldsymbol{D}, \text{甲} \in \text{虚}: \alpha(X) \sqsubseteq \text{甲} \Leftrightarrow X \subseteq \gamma(\text{甲})$$

定理

- $(\boldsymbol{D}, \subseteq) \leftrightharpoons_{\alpha}^{\gamma} (\text{虛}, \sqsubseteq)$当且仅当以下所有公式成立

- α是单调的：$\forall X, Y \in \boldsymbol{D}: X \subseteq Y \Rightarrow \alpha(X) \sqsubseteq \alpha(Y)$
- γ是单调的：$\forall \text{甲}, \text{乙} \in \text{虛}: \text{甲} \sqsubseteq \text{乙} \Rightarrow \gamma(\text{甲}) \subseteq \gamma(\text{乙})$
- $\gamma \circ \alpha$不变或增大输入：$\forall X \in \boldsymbol{D}: X \subseteq \gamma\big(\alpha(X)\big)$
- $\alpha \circ \gamma$不变或缩小输入：$\forall \text{甲} \in \text{虛}: \alpha\big(\gamma(\text{甲})\big) \sqsubseteq \text{甲}$

证明

- $\Rightarrow$
  - $\forall X \in \boldsymbol{D}: X \subseteq \gamma\big(\alpha(X)\big)$
    - 由$\alpha(X) \sqsubseteq \alpha(X)$和伽罗瓦连接定义可得$X \subseteq \gamma\big(\alpha(X)\big)$
  - $\forall$甲 $\in$ **虚**: $\alpha\big(\gamma(\text{甲})\big) \sqsubseteq$ 甲
    - 由$\gamma(\text{甲}) \subseteq \gamma(\text{甲})$和伽罗瓦连接定义可得$\alpha\big(\gamma(\text{甲})\big) \sqsubseteq$ 甲
  - $\forall X, Y \in \boldsymbol{D}: X \subseteq Y \Rightarrow \alpha(X) \sqsubseteq \alpha(Y)$
    - $X \subseteq Y \subseteq \gamma\big(\alpha(Y)\big) \Rightarrow \alpha(X) \sqsubseteq \alpha(Y)$
  - $\forall$甲, 乙 $\in$ **虚**: 甲 $\sqsubseteq$ 乙 $\Rightarrow \gamma(\text{甲}) \subseteq \gamma(\text{乙})$
    - $\alpha\big(\gamma(\text{甲})\big) \sqsubseteq$ 甲 $\sqsubseteq$ 乙 $\Rightarrow \gamma(\text{甲}) \subseteq \gamma(\text{乙})$

证明

- $\Leftarrow$
  - $\alpha(X) \sqsubseteq$ 甲
  - $\Rightarrow \gamma(\alpha(X)) \subseteq \gamma(\text{甲})$ 【γ的单调性】
  - $\Rightarrow X \subseteq \gamma(\text{甲})$ 【 $X \subseteq \gamma(\alpha(X))$ 】

  - $X \subseteq \gamma(\text{甲})$
  - $\Rightarrow \alpha(X) \sqsubseteq \alpha(\gamma(\text{甲}))$ 【α的单调性】
  - $\Rightarrow \alpha(X) \sqsubseteq$ 甲 【$\alpha(\gamma(\text{甲})) \sqsubseteq$ 甲】

函数抽象

- 给定伽罗瓦连接$(\boldsymbol{D}, \subseteq) \leftrightarrows_{\alpha}^{\gamma} (\text{虚}, \sqsubseteq)$
- 给定$\boldsymbol{D}$上的函数f和虚上的函数子

- 子是 f 的安全抽象,当且仅当
  - $\alpha \circ f \circ \gamma(\text{甲}) \sqsubseteq \text{子}(\text{甲})$
- 子是 f 的最佳抽象,当且仅当
  - $\alpha \circ f \circ \gamma = \text{子}$
- 子是 f 的精确抽象,当且仅当
  - $f \circ \gamma = \gamma \circ \text{子}$
- 最佳抽象总是存在，但精确抽象不一定存在

数据流分析的安全性-定义

- 考虑指称语义，每条语句是从输入状态到输出状态的函数
- 考虑关于程序行为的上近似分析，即程序被看做从输入状态到输出状态的函数（参考莱斯定理）

- 令程序状态集合的全集为D，状态的抽象域为$\mathbb{D}$，二者形成一个伽罗瓦连接。
- 对于语句S_i，令f_i为该语句的具体语义，$\mathbb{F}_i$是其安全抽象
  - 我们假设f_i自然扩展到状态集合，即$f_i(X) = \{ f_i(x) \mid x \in X \wedge f_i(x)$有定义$\}$
  - 容易看出在状态集合上f_i是单调的

数据流分析的安全性

- 之前我们证明了
  - 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$\mathrm{DATA}_{\mathrm{v_i}} \sqsubseteq \bigsqcup_{v_1 v_2 v_3 \ldots v_i \in P} f_{v_i} \circ f_{v_{i-1}} \circ \cdots \circ f_{v_1}\left(I_{entry}\right)$
  - 即数据流分析相对“所有路径结果的合并”是安全的
- 下面我们论证“所有路径结果的合并”在顺序、分支、循环三种基本结构上是安全的，从而整体是安全的

因为本节只讨论上近似，替换交汇操作为合并操作以方便理解

数据流分析的安全性-顺序

- 语法：$s_1; s_2$
- 指称语义： $f_2 \circ f_1$
- 控制流路径抽象：$\mathfrak{F}_2 \circ \mathfrak{F}_1$

- 定理：$\mathfrak{F}_2 \circ \mathfrak{F}_1$是$f_2 \circ f_1$的安全抽象。
- 证明：
  - $\alpha \circ f_1 \circ \gamma(\text{甲}) \sqsubseteq \mathfrak{F}_1\,(\text{甲})$ 【 f_1安全性】
  - $\Rightarrow f_1 \circ \gamma(\text{甲}) \sqsubseteq \gamma \circ \mathfrak{F}_1\,(\text{甲})$ 【 伽罗瓦连接定义】
  - $\Rightarrow \alpha \circ f_2 \circ f_1 \circ \gamma(\text{甲}) \sqsubseteq \alpha \circ f_2 \circ \gamma \circ \mathfrak{F}_1\,(\text{甲})$ 【 单调性】
  - $\Rightarrow \alpha \circ (f_2 \circ f_1) \circ \gamma(\text{甲}) \sqsubseteq \mathfrak{F}_2 \circ \mathfrak{F}_1\,(\text{甲})$ 【f_2安全性】

数据流分析的安全性-选择1

- 语法：if (c) then s_1 else s_2
  - 令T为满足c的状态集合
- 指称语义： $f(x)=\begin{cases} f_1(x),\ x\in T \\ f_2(x),\ x\notin T \end{cases}$
- 控制流路径抽象：$\text{予}(\text{甲})=\text{予}_2(\text{甲})\sqcup\text{予}_1(\text{甲})$

- 定理：予是f的安全抽象。
- 证明：
  - 根据半格和偏序的关系，有
  - $\text{予}_1\sqsubseteq\text{予}_2\sqcup\text{予}_1$和$\text{予}_2\sqsubseteq\text{予}_2\sqcup\text{予}_1$
  - 同时因为予_1和予_2分别是f_1,f_2的安全抽象，所以定理成立。

数据流分析的安全性-选择2

- 语法：if (c) then s_1 else s_2
  - 令T为满足c的状态集合，F为不满足c的状态集合
  - $\text{丑}(\text{甲}) = \alpha(\gamma(\text{甲}) \cap T)$
  - $\text{寅}(\text{甲}) = \alpha(\gamma(\text{甲}) \cap F)$
- 指称语义： $f(x) = \begin{cases} f_1(x), & x \in T \\ f_2(x), & x \notin T \end{cases}$
- 控制流路径抽象：$\text{子}(\text{甲}) = \text{子}_2 \circ \text{寅}(\text{甲}) \sqcup \text{子}_1 \circ \text{丑}(\text{甲})$

- 定理：子是f的安全抽象。
- 证明：
  - 由$f\big(\gamma(\text{甲})\big) = f_1(\gamma(\text{甲}) \cap T) \cup f_2(\gamma(\text{甲}) \cap F) \subseteq \gamma\big(\text{子}_1 \circ \text{丑}(\text{甲}) \sqcup \text{子}_2 \circ \text{寅}(\text{甲})\big)$可知定理成立

证明

- 首先证明辅助定理
  - $\gamma(甲) \cup \gamma(乙) \subseteq \gamma(甲 \sqcup 乙)$
  - 由
    - $甲 \sqsubseteq 甲 \sqcup 乙$可知$\gamma(甲) \subseteq \gamma(甲 \sqcup 乙)$
    - $乙 \sqsubseteq 甲 \sqcup 乙$可知$\gamma(乙) \subseteq \gamma(甲 \sqcup 乙)$
  - 因此$\gamma(甲 \sqcup 乙)$是集合$\{\gamma(甲), \gamma(乙)\}$的一个上界
  - 同时因为$\gamma(甲) \cup \gamma(乙)$是最小上界，原定理成立
- 再证明原定理
  - $f\big(\gamma(甲)\big)$
  - $= f_1(\gamma(甲) \cap T) \cup f_2(\gamma(甲) \cap F)$
  - $\subseteq f_1\big(\gamma \circ \alpha(\gamma(甲) \cap T)\big) \cup f_2\big(\gamma \circ \alpha(\gamma(甲) \cap F)\big)$
  - $= f_1\big(\gamma \circ 丑(甲)\big) \cup f_2\big(\gamma \circ 寅(甲)\big)$
  - $\subseteq \gamma \circ 子_1 \circ 丑(甲) \cup \gamma \circ 子_2 \circ 寅(甲)$
  - $\subseteq \gamma\big(子_1 \circ 丑(甲) \sqcup 子_2 \circ 寅(甲)\big)$
  - 再根据伽罗瓦连接的性质，原定理成立。

数据流分析的安全性-循环

- 语法：while (c) s_1
  - 令T为满足c的状态集合
- 指称语义：$f(x) = \begin{cases} f \circ f_1(x), & x \in T \\ x, & x \notin T \end{cases}$
  - 如果输入为x时出现无限循环，则认为f(x)未定义
  - 非严格定义，可以用不动点算子转换成更严格的定义
- 控制流路径抽象：$\mathfrak{F}(\text{甲}) = \lim\limits_{j \to +\infty} \sqcup_{0 \le i \le j} \mathfrak{F}_1^i(\text{甲})$

- 定理：$\mathfrak{F}$是f的安全抽象。
- 证明：
  - 假设具体域循环最多执行i次，易证抽象域计算结果安全
  - 因为i选择的任意性，所以原定理成立

常见抽象域

- 设计抽象解释的关键是为具体语句设计函数抽象
  - 比如为四则运算设计函数抽象
- 接下来介绍一些常见抽象域，主要针对数值计算

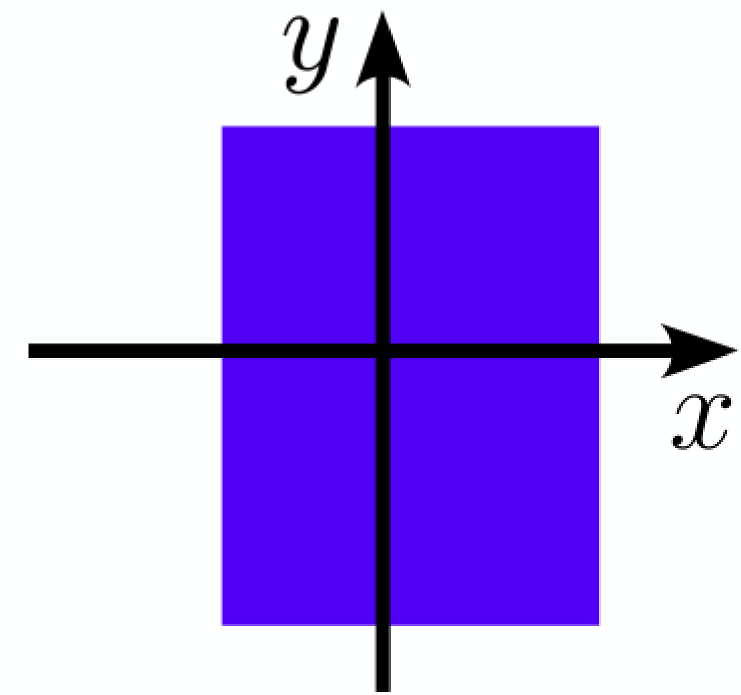
关系抽象

- 在数值计算上我们已经看见过区间分析、符号分析的抽象域
- 区间分析/符号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。

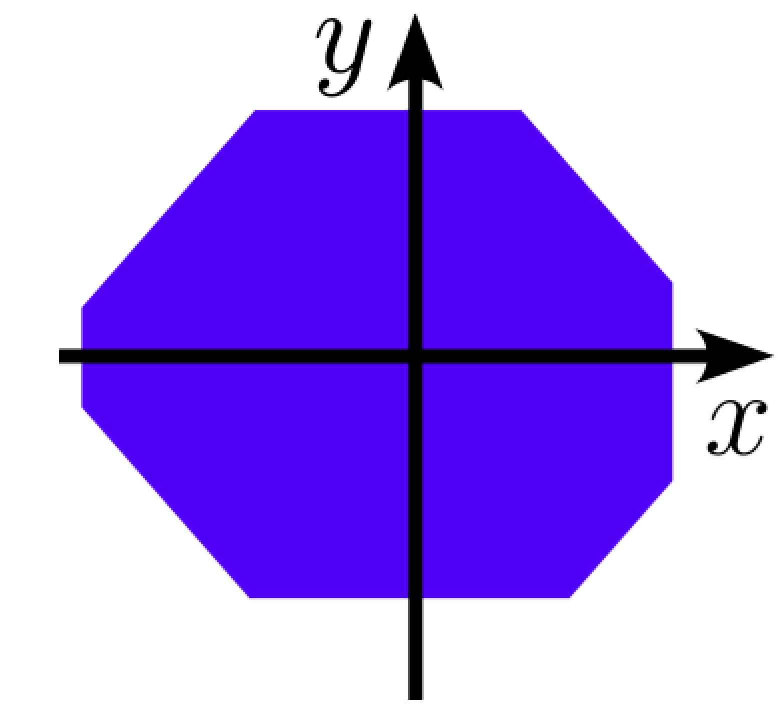
- 考虑变量之间关系的抽象称为关系抽象。

关系抽象举例：八边形

假设程序中只有x和y两个变量



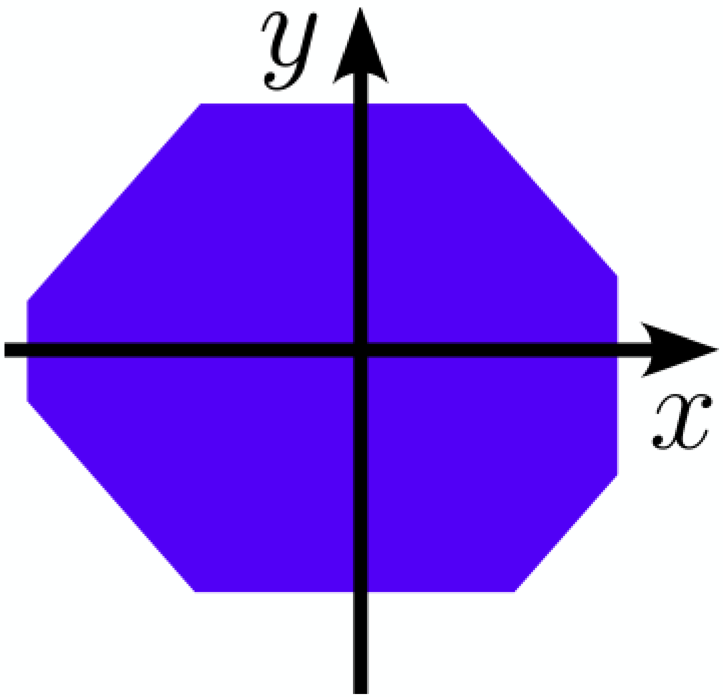
区间抽象形成一个矩形



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



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：$x - y \geq a_8$

加上4条45度的线来形成八边形

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
- 即$\pm v_1 \pm v_2 \leq a$，其中v_1, v_2为程序上的任意变量
- 可用矩阵表示

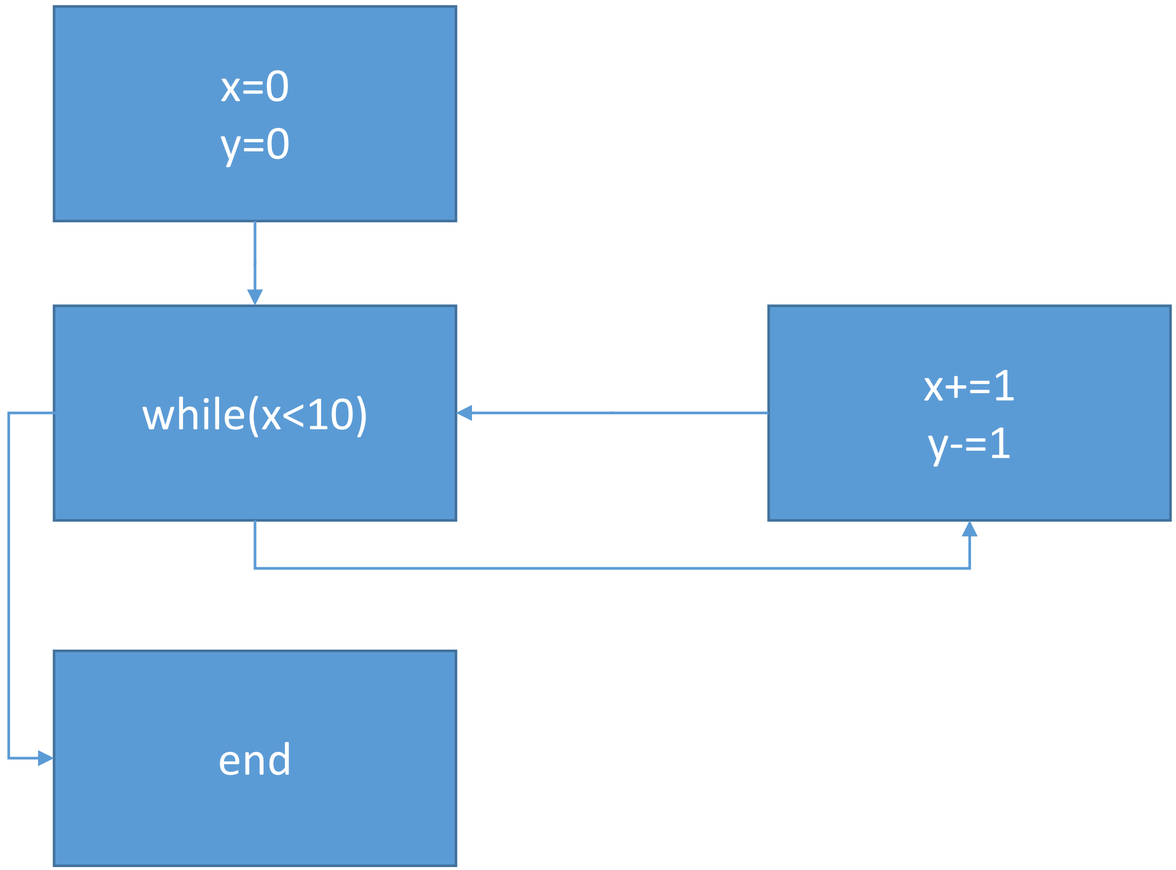
	+x	-x	+y	-y
+x	10	-	-	-
-x	-	0	-	-
+y	10	5	20	-
-y	-2	5	-	-10

2个变量的矩阵。更多变量需要更大的矩阵。

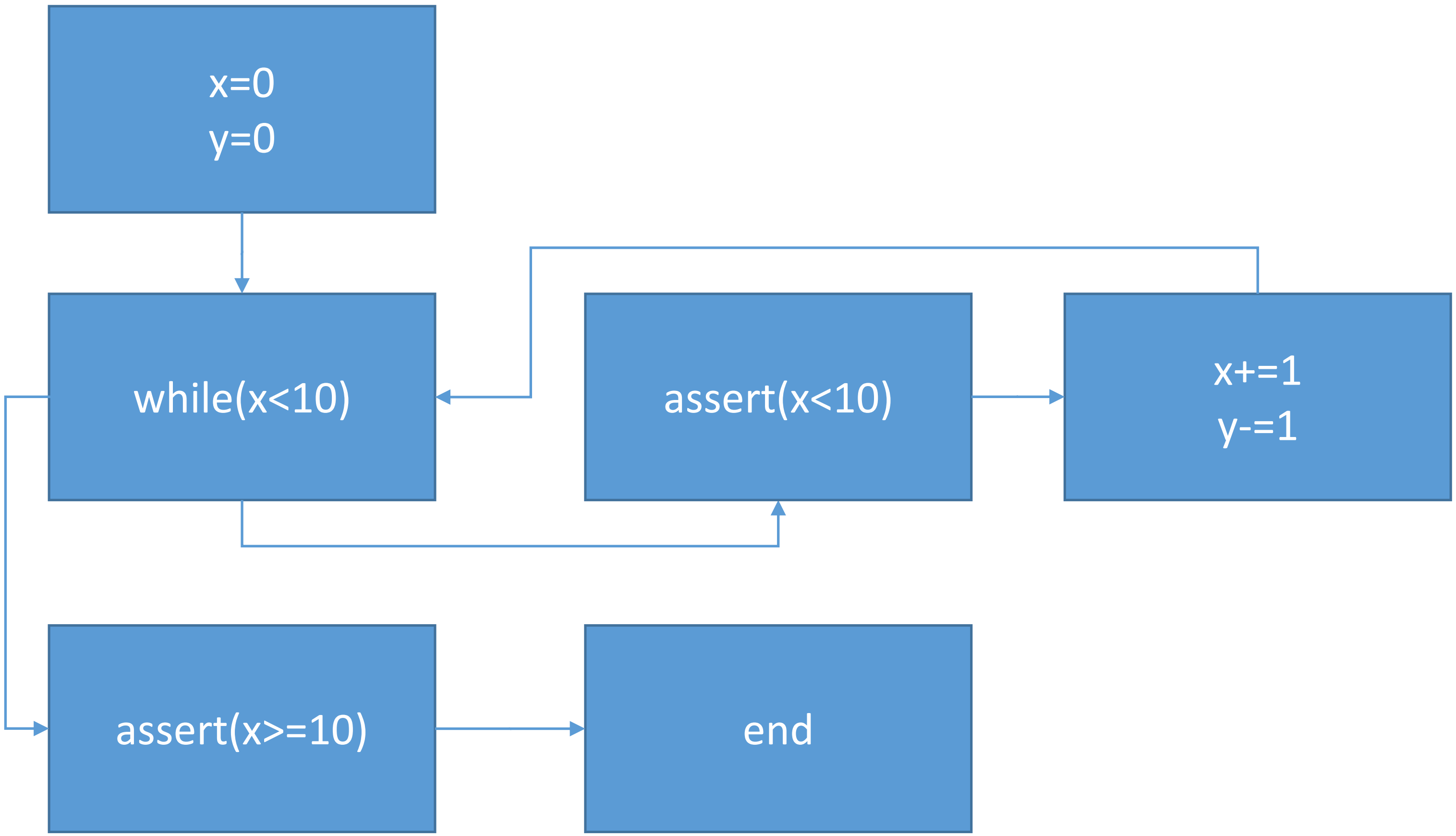
八边形上的计算

- 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.

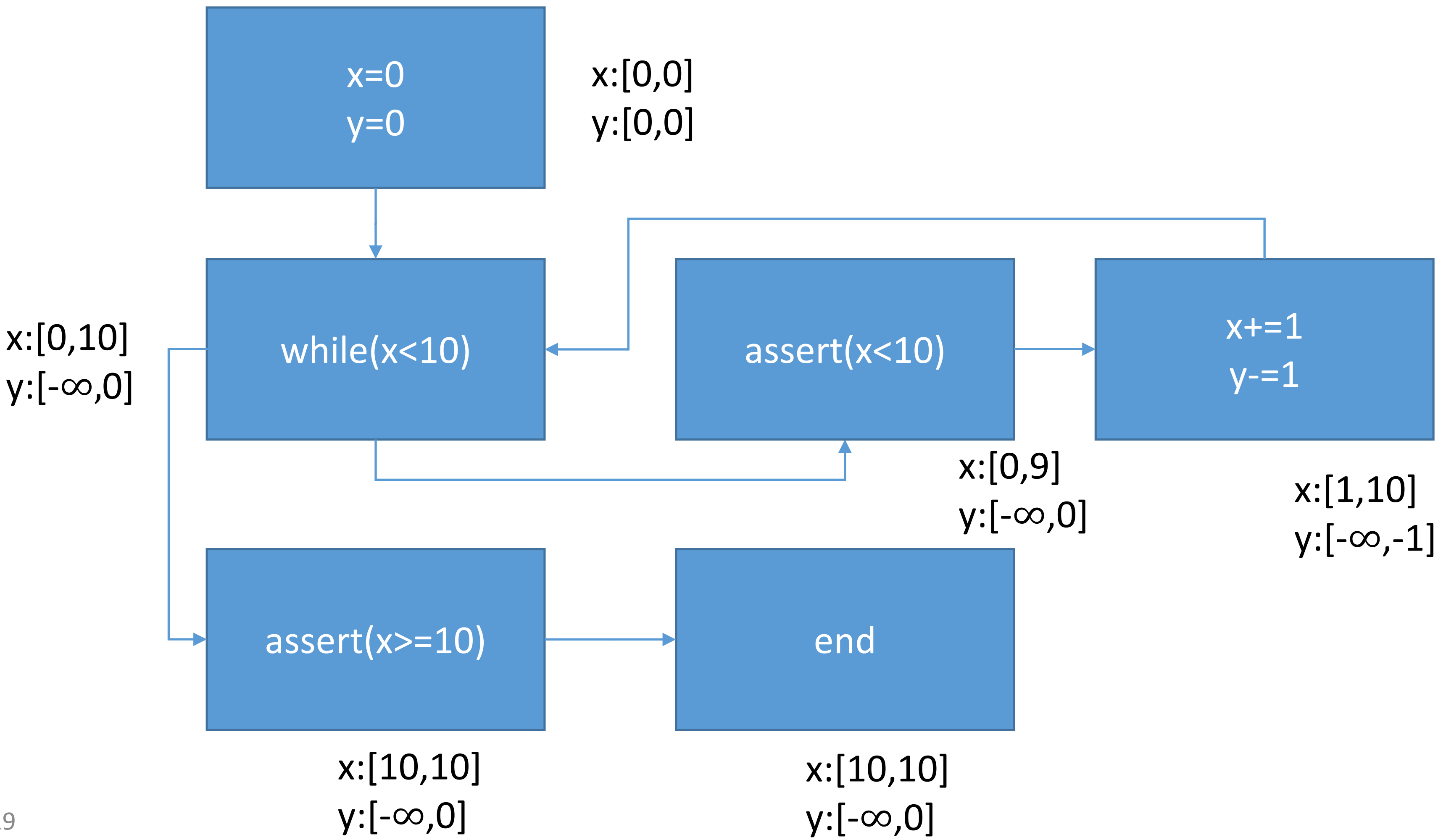
八边形计算举例



八边形计算举例

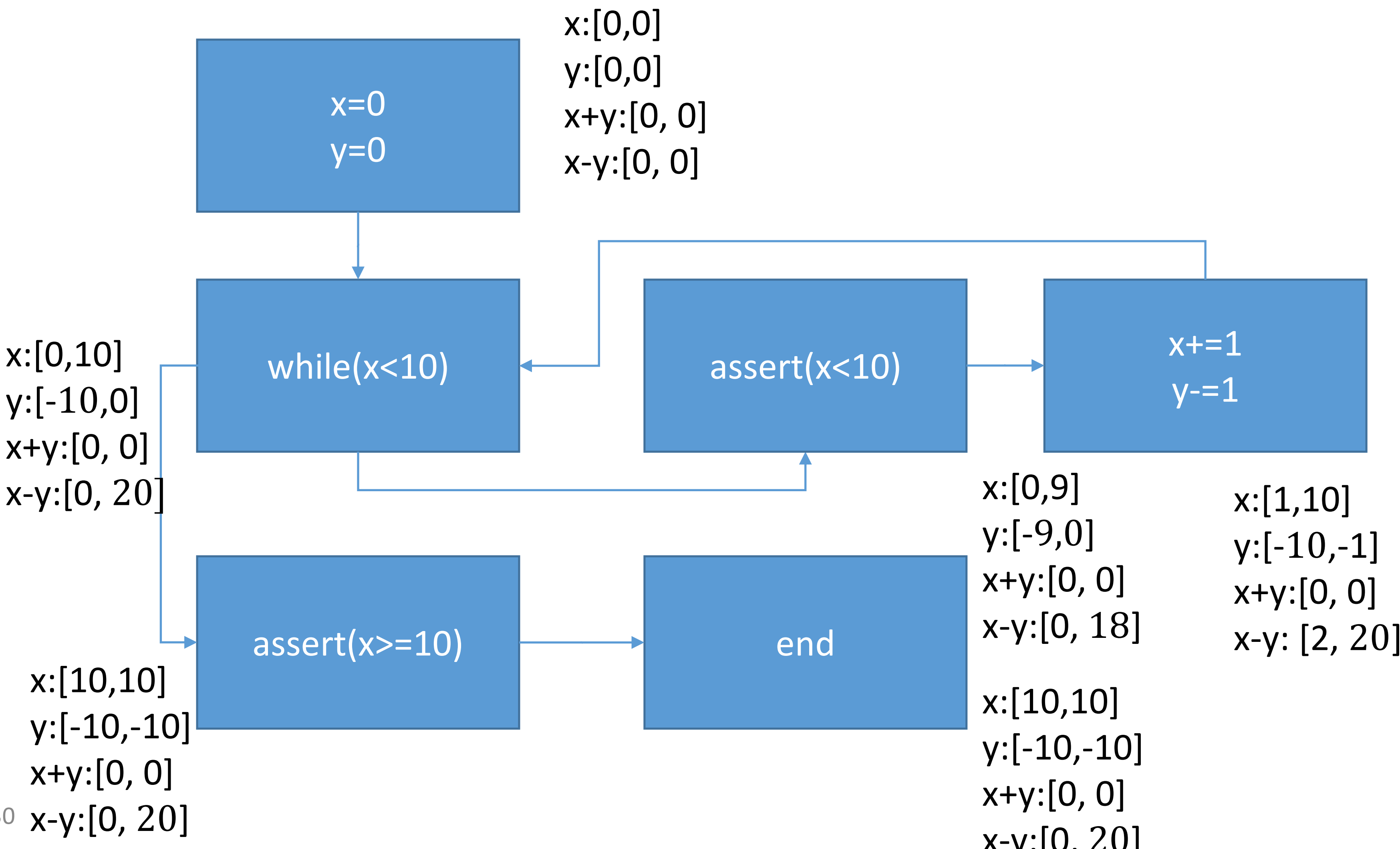


区间计算结果

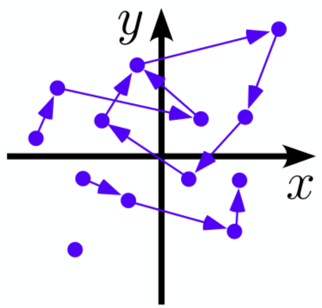


x=0
y=0
x:[0,0]
y:[0,0]
x:[0,10]
y:[-∞,0]
while(x<10)
assert(x<10)
x+=1
y-=1
x:[0,9]
y:[-∞,0]
x:[1,10]
y:[-∞,-1]
assert(x>=10)
end
x:[10,10]
y:[-∞,0]
x:[10,10]
y:[-∞,0]

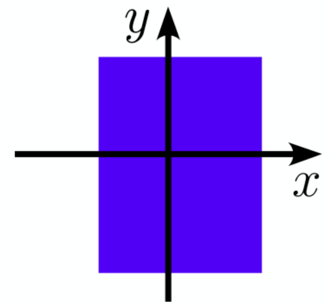
八边形计算结果


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

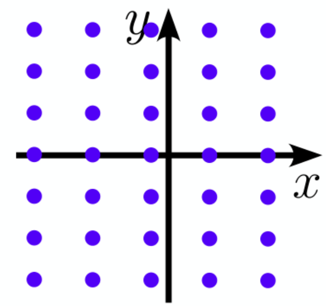
其他数值常用抽象



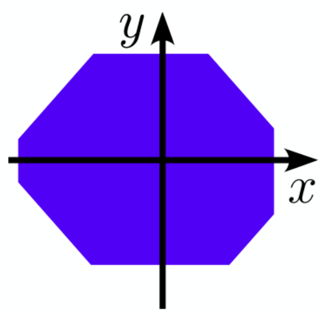
Collecting semantics:
partial traces



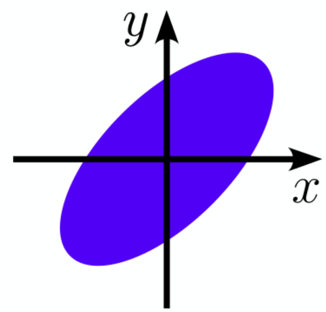
Intervals.
$\mathrm{x} \in [a, b]$



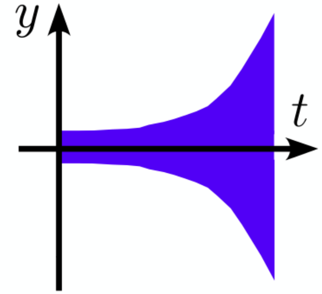
Simple congruences:
$\mathrm{x} \equiv a[b]$



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



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



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

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

- 示例：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

可选择APRON中的抽象域

Interproc Analyzer

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

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

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/inte

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

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
```

参考资料

- MIT Course 16.399: Abstract Interpretation
  - Patrick Cousot, 2016
- Abstract Interpretation Tutorial
  - Patrick Cousot, TASE 2015
- 抽象解释及其在静态分析中的应用
  - 陈立前，西南大学Computer Science Week