单变量区间线性不等式抽象域

陈立前 王 戟 侯苏宁

(国防科学技术大学计算机学院并行与分布处理国防科技重点实验室 长沙 410073)

摘 要 程序变量的值范围信息对于编译器优化、程序分析与验证等应用至关重要.抽象解释理论提供了一种通用框架为程序变量计算近似的但是可靠的值范围.然而该框架下已有的数值抽象域在表达非凸性质方面存在一定的局限性,影响了值范围分析的精度.文中基于抽象解释理论,提出一个新的数值抽象域——单变量区间线性不等式抽象域.其主要思想是使用单变量区间线性不等式约束作为域元素的约束表示方法.该抽象域的表达能力强于经典的区间抽象域,并允许表达某类非凸、非连通性质.同时,其域操作存在高效的实现算法.该抽象域具有很强的可扩展性,能够应用在实际大规模的程序分析中.

关键词 值范围分析;静态分析;抽象解释;抽象域;区间分析

中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2010.00427

An Abstract Domain of One-Variable Interval Linear Inequalities

CHEN Li-Qian WANG Ji HOU Su-Ning

(*National Laboratory for Parallel and Distributed Processing, School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract The value range information of program variables is crucial for many applications, such as compiler optimization, program analysis and verification, etc. The theory of abstract interpretation provides a general framework to compute statically approximate but sound value ranges for program variables. However, most existing numerical abstract domains under the framework have limitations in expressing non-convex properties, which may lead to imprecision during the value range analysis. This paper proposes a new numerical abstract domain under the framework of abstract interpretation, namely an abstract domain of one-variable interval linear inequalities. The main idea is to use one-variable interval linear inequality constraints as the representation of domain elements. The new domain is more expressive than the classic interval abstract domain and allows expressing certain non-convex, unconnected properties. Moreover, its domain operations can be implemented via efficient algorithms. Thus, it has high scalability and can be applied in large-scale program analysis in practice.

Keywords value range analysis; static analysis; abstract interpretation; abstract domain; interval analysis

1 引 言

程序中变量的值范围信息对于编译器构造/优化/并行化、程序分析与验证、可配置体系结构中数据最小位宽计算、实时系统程序最差情况执行时间分析等应用至关重要[1-3].然而,静态的方法往往难以获取变量精确的值范围信息.一种通常的方法是,

收稿日期:2009-04-20;最终修改稿收到日期:2009-10-13.本课题得到国家自然科学基金(60725206,60621003,60673118)、湖南省自然科学基金(07JJ1011)资助.**陈立前**,男,1982 年生,博士研究生,主要研究方向为程序分析与验证、抽象解释.E-mail: lqchen@nudt.edu.cn. **王 戟**,男,1969 年生,博士,教授,博士生导师,主要研究领域为高可信软件技术、软件方法学、软件工程等. **侯苏宁**,男,1985 年生,硕士研究生,主要研究方向为面向源代码的程序分析.

基于抽象解释理论来为程序变量计算近似的但是可靠的值范围[3].

抽象解释[4]是一种关于语义近似的通用理论.该理论提供了一个通用框架来设计和构建静态分析以自动化地推导程序的动态性质.其中,抽象域是抽象解释框架下的一个核心概念.抽象域包括一个特定类别的、计算机可表示的对象(称为域元素)集合以及一系列用来操纵这些对象的操作(称为域操作)集合.程序分析中,程序的状态集合通过抽象域中的域元素来近似,程序语义动作(赋值、条件测试、控制流接合、循环等)通过抽象域中的域操作来可靠建模.抽象域的设计与实现往往是在计算效率和分析精度间取得合理折衷.

区间分析可以发现变量的上下界信息,是最早用于变量值范围分析的方法[1].由于实数区间所构成的完全格上存在无穷递增链,区间分析可能不终止.为了保证区间分析的计算效率和终止性,Cousot和Cousot于1976年把经典的区间算术适配到抽象解释框架下,并提出了区间抽象域[5].该抽象域的简单易用和很高的计算效率(线性的时空复杂度)使得它在静态分析中得到了极为广泛的应用,包括常量传播、无用代码消除、运行时错误检查、循环时间分析等[5-6].相对于后来出现的、能够表示变量间关系的数值抽象域(如多面体抽象域[7]、八边形抽象域[8]等),区间抽象域只能表示单个变量的性质,因此表达能力比较弱、分析精度不高.但是,通过某些策略(比如带阈值的加宽策略[6]),区间抽象域的分析精度可以得到进一步的改进.至今,区间抽象域仍然是应用最广泛的数值抽象域之一.尤其,在实际大规模的程序分析中,区间抽象域由于其高计算效率以及强可扩展性,备受学术界与工业界程序分析工具开发者的青睐.许多静态分析工具都使用区间抽象域作为整个分析的基础,包括ASTRÉE[6]、AbsInt PAG①、Sparrow②等.

然而,即使在表达单个变量的值范围性质上,区间抽象域仍存在一定局限性.类似于大部分数值抽象域(包括多面体抽象域、八边形抽象域等),区间抽象域只能表示凸(convex)的性质.但是,这种凸性的限制可能影响分析的精确性,导致出现误报.尤其,除法是程序开发中很常用的算术运算,"除零错"成为实际程序中一种重要的、常见的运行时错误.但是,在当前数值程序分析中,由于除法表达式不是线性的,一般把除法表达式转化为线性表达式来处理:使用一个区间来上近似分母表达式的值,然后通过除以该区间将整个表达式抽象为线性表达式.但是,一旦该区间包含了零,则不仅可能产生"除零错"的误报,而且整个除法表达式的抽象值将变成$[-\infty,+\infty]$,并将影响到后续程序分析的精确性.比如,对于除法表达式$2*z/(x*y)$,如果分母表达式$x*y$的值范围是$[2,4]$,则除法表达式$2*z/(x*y)$将抽象成$2*z/[2,4]$,即$[0.5,1]*z$,从而成为线性的.但是如果$x*y$的值范围区间包含了0,则传统方法将只能把$2*z/(x*y)$抽象为$[-\infty,+\infty]$.为了能够对除法表达式进行更为精确的抽象并尽可能地消除"除零错"误报,需要设计一个数值抽象域尽可能地把0排除在值范围之外.

本文使用图1中所示程序来展示已有数值抽象域,尤其是区间抽象域、所存在的凸性局限性并说明本文的研究动机.对于图1中所示程序Prog1,如果使用区间抽象域进行分析,在循环中x的取值范围将是$[-1,1]$,因此会产生误报,即程序中第5行的赋值语句有"除零错".注意,即使是使用表达能力更强的凸的数值抽象域(如多面体抽象域、八边形抽象域等)来分析该程序,仍然会产生"除零错"的误报.然而,根据程序的具体语义,在循环中变量x的取值始终是前一次循环执行后x的值的相反数,因此x的取值要么是1要么是-1,第5行的赋值语句不会发生"除零错".但是,非凸性质$x=-1 \vee x=1$是通常的凸的数值抽象域表达不了的,需要一种非凸的抽象域才能表达.

```
Prog 1
1.  real x, y;
2.  x:=1;
3.  while (true) do
4.      x:=-x;
5.      y:=1/x;①
6.  done;
```

位置	区间抽象域	程序具体语义
①	$x\in[-1,1]$ $y\in[-\infty,+\infty]$	$(x=1\wedge y=1)$ $\vee(x=-1\wedge$ $y=-1)$

---------> 除零错 安全的!

图1 数值抽象域凸性局限性的示例
程序Prog1(左)及其不变式(右)

另一方面,在实际软硬件系统的分析与验证中,所给应用数据可能是不精确的或者受到非确定性因素的影响,使得这些数据仅能定位在某个特定的区间范围内.因此,在建模或抽象后,抽象模型或抽象程序中可能存在区间形式的输入数据.特别地,为了分析包含非线性操作(如两表达式的乘/除)或浮点算术的程序,常常需要使用一种称为线性化的技术来把非线性或浮点表达式抽象成带区间系数的线性表达式(形如$\Sigma_k[a_k,b_k]x_k+[c,d]$)[9].另外,当使用数值抽象域的浮点实现(如文献[10])来分析程序

① http://www.absint.com/pag/
② http://www.spa-arrow.com/

时,为了保证可靠性,待分析程序中的实数需要抽象成浮点区间(比如,项 $0.1*x$ 中的实数系数 0.1 不是一个 IEEE754 标准能够表示的浮点数,从而程序分析时需要使用一个浮点区间$[a,b]$来上近似,其中 a 是比 0.1 小的最大浮点数,b 是比 0.1 大的最小浮点数).而且,许多浮点算法只能输出可靠的界,即使输入参数都是精确的浮点值.比如,由于不同执行环境的浮点舍入模式(IEEE754 标准支持向上、向下、向零、向最近 4 种舍入模式)可能不一样,同样两个浮点数 a 和 b 的加法可能得到不同的浮点结果,需要使用一个浮点区间$[a\oplus_{f,-\infty}b, a\oplus_{f,+\infty}b]$来进行上近似,其中$\oplus_{f,-\infty}$与$\oplus_{f,+\infty}$分别表示浮点格式 f 中向下和向上舍入的浮点加法.换言之,在实际程序分析中,区间作为变量系数的情形很自然地出现了.然而,传统区间抽象域(形如 $a\leq x\leq b$,即$\{x\leq b, -x\leq -a\}$或 $x\in[a,b]$)尚不支持使用区间作为变量的系数.

为此,本文提出一个新的抽象域——单变量区间线性不等式抽象域,来推导关于程序变量 x 形如$[a,b]x\leq c$ 的关系,其中常数 $a,b,c\in\mathbb{R}$ 将由基于抽象解释的静态分析器自动推导出来.直观地讲,单变量区间线性不等式抽象域可以看成是经典区间抽象域的区间系数扩展版本.然而,与经典的数值抽象域(包括区间抽象域、八边形抽象域、多面体抽象域等)不同的是,几何上,单变量区间线性不等式抽象域中的域元素不一定是凸或连通的.因此,该抽象域可以自然地表示某些析取信息,而不需要显式地使用析取逻辑符 $\vee$ 来表达.而且,单变量区间线性不等式抽象域只需要通过简单的区间算术就可以实现,有着较高的计算效率.本文实验结果表明:单变量区间线性不等式抽象域可以比区间抽象域发现更精确的值范围信息,同时所耗费的计算代价并不高.

本文第 2 节回顾经典区间抽象域的设计并给出本文单变量区间线性不等式的语义;第 3 节提出一个新的数值抽象域——单变量区间线性不等式抽象域,并给出该抽象域上的常用域操作;第 4 节讨论单变量区间线性不等式抽象域的原型实现及其实验结果;第 5 节给出相关工作比较;第 6 节总结全文并展望未来工作.

2　预备知识

2.1　经典的区间抽象域

在基于区间抽象域的静态分析中,程序中某程序点处变量的值由该变量可能取到的最小值和最大值所构成的区间来近似.基于抽象解释理论,实数域(具体域)与区间抽象域(抽象域)之间的关系可以通过下面一个 Galois 连接来表示,即

$$(\wp(\mathbb{R}),\leq)\underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}}(Itvs,\sqsubseteq_i),$$

其中,$Itvs$ 是实数 $\mathbb{R}$ 上的区间集合$\{[a,b]\mid a\in\mathbb{R}\cup\{-\infty\}, b\in\mathbb{R}\cup\{+\infty\}, a\leq b\}\cup\{\bot_i\}$.$\mathbb{R}$ 上的区间集合 $Itvs$ 构成了一个完全格$(Itvs,\sqsubseteq_i,\sqcap_i,\sqcup_i,\bot_i,\top_i)$,其中,$\top_i\triangleq[-\infty,+\infty]$表示整个实数集合 $\mathbb{R}$,$\bot_i$表示空区间.同时,定义$[b,a]\triangleq\bot_i$,若 $b>a$.具体函数 $\gamma_i\in[Itvs\to\wp(\mathbb{R})]$定义为 $\gamma_i(\bot_i)\triangleq\varnothing$,$\gamma_i([a,b])\triangleq\{x\in\mathbb{R}\mid a\leq x\leq b\}$.抽象函数$\alpha_i\in[\wp(\mathbb{R})\to Itvs]$定义为 $\alpha_i(\varnothing)\triangleq\bot_i$,$\alpha_i(S)\triangleq[\min S,\max S]$,其中 $S\subseteq\mathbb{R}$ 是一个实数集合.$Itvs$ 上的偏序关系 $\sqsubseteq_i$ 定义为$[a,b]\sqsubseteq_i[a',b']$当且仅当$[a,b]=\bot_i$或 $a\geq a'\wedge b\leq b'$.

在基于抽象解释的程序分析中,通常为每一个程序点构建一个抽象环境,记作 $X^\sharp$,用以把每个程序变量映射到抽象域中的一个域元素.区间抽象环境$[Vars\to Itvs]$把每个程序变量映射到一个表示其值范围的区间上,其中 $Vars$ 表示待分析程序的变量集合.在 n-维空间中,如果一个连通区域所对应的约束系统中的每个约束都形如 $x_j\leq c$ 或 $-x_j\leq c$,其中 c 是一个实数常量或 $+\infty$,则称该连通区域是一个盒(box).盒可以是无界的,但一定是凸的.

下面给出区间抽象域上的一些基本域操作.

(1) 交(meet)与接合(join)

$$I\sqcap_i I'\triangleq\begin{cases}[\max(a,a'),\min(b,b')], & I=[a,b]\wedge I'=[a',b']\wedge\\ & \max(a,a')\leq\min(b,b'),\\ \bot_i, & \text{否则}\end{cases}$$

$$I\sqcup_i I'\triangleq\begin{cases}[\min(a,a'),\max(b,b')], & I=[a,b]\wedge I'=[a',b']\\ I, & I'=\bot_i\\ I', & I=\bot_i\end{cases}.$$

注意:$\sqcap_i$是精确的,即 $\gamma(I\sqcap_i I')=\gamma(I)\bigcap\gamma(I')$.但是,$\sqcup_i$不是精确的,$\sqcup_i$只能保证是两个区间的(集合)并的上近似,即 $\gamma(I)\bigcup\gamma(I')\subseteq\gamma(I\sqcup_i I')$.例如,设 $I=[0,1]$且 $I'=[3,4]$,则 $I\sqcup_i I'=[0,4]$,显然 $2\in I\sqcup_i I'$,但是 $2\notin\gamma(I)\bigcup\gamma(I')$.

(2) 区间算术

当参与区间算术计算的任意一个参数为$\bot_i$时,其计算结果也为$\bot_i$.

$-_i[a',b'] \triangleq [-b',-a']$;

$[a,b]+_i[a',b'] \triangleq [a+a',b+b']$;

$[a,b]-_i[a',b'] \triangleq [a-b',b-a']$;

$[a,b]\times_i[a',b'] \triangleq [\min\{a\times a',a\times b',b\times a',b\times b'\},$
$\max\{a\times a',a\times b',b\times a',b\times b'\}]$;

$1/_i[a',b'] \triangleq [\min\{1/a',1/b'\},$
$\max\{1/a',1/b'\}]$,
如果 $a'\geq 0$ 或 $b'\leq 0$;

$1/_i[a',b'] \triangleq [-\infty,+\infty]$, 如果 $a'<0<b'$;

$[a,b]/_i[a',b'] \triangleq [a,b]\times_i(1/_i[a',b'])$.

这里,我们把+,-,×,/按照标准方式扩展到 $\mathbb{R}\cup\{+\infty,-\infty\}$ 上. 特别地,$(+\infty)\times 0 \triangleq 0$,$(-\infty)\times 0 \triangleq 0$,$\forall x>0.x/0 \triangleq +\infty$,$\forall x<0.x/0 \triangleq -\infty$,$\forall x.x/(+\infty) \triangleq 0$,$\forall x.x/(-\infty) \triangleq 0$.

(3) 迁移函数

①赋值迁移函数

对于赋值语句 $x:=expr$,在抽象环境 $X^\sharp$ 下,其赋值迁移函数定义为

$$[\![x:=expr]\!]^\sharp X^\sharp \triangleq X^\sharp[x\mapsto [\![expr]\!]^\sharp X^\sharp],$$

其中,$[\![expr]\!]^\sharp X^\sharp$ 表示在抽象环境 $X^\sharp$ 下采用上述区间算术操作来计算表达式 $expr$ 所得到的区间抽象值.

②测试迁移函数

设 $X^\sharp(x)=[a,b]$,则有

$$[\![x\leq c]\!]^\sharp X^\sharp \triangleq \begin{cases} \perp_i, & a>c \\ X^\sharp[x\mapsto[a,\min\{b,c\}]], & \text{否则} \end{cases},$$

$$[\![x\geq c]\!]^\sharp X^\sharp \triangleq \begin{cases} \perp_i, & b<c \\ X^\sharp[x\mapsto[\min\{a,c\},b]], & \text{否则} \end{cases}.$$

注意,任意形式的(测试条件)约束均可以抽象成一个或多个形如 $x\leq c$ 或 $x\geq c$ 的约束.

(4) 加宽与变窄算子

本文沿用文献[5]关于区间抽象域上的加宽 ∇_i 与变窄 Δ_i 算子的定义:

$\perp_i \nabla_i I = I \nabla_i \perp_i \triangleq I$;

$[a,b]\nabla_i[a',b'] \triangleq [a\leq a'?a:-\infty, b\geq b'?b:+\infty]$;

$[a,b]\Delta_i[a',b'] \triangleq [a=-\infty?a':a, b=+\infty?b':b]$.

2.2 单变量区间线性不等式的语义

本文使用单变量区间线性不等式

$$[a,b]x\leq c$$

表示所有单变量线性不等式 $dx\leq c$ 所构成的不等式族,其中 $d\in[a,b]$.

定义 1. $x^*\in\mathbb{R}$ 称为单变量区间线性不等式 $[a,b]x\leq c$ 的一个弱解(weak solution),若存在某个 $d\in[a,b]$ 使得 x^* 满足 $dx^*\leq c$,并且,集合 $\Sigma_\exists([a,b]x\leq c)=\{x^*\in\mathbb{R}\mid \exists d\in[a,b].dx^*\leq c\}$ 称为单变量区间线性不等式 $[a,b]x\leq c$ 的弱解集合.

如下定理从代数角度刻画了单变量区间线性不等式的弱解集合.

定理 1. $x^*\in\mathbb{R}$ 是 $[a,b]x\leq c$ 的弱解当且仅当 x^* 满足 $(a+b)x^*-(b-a)|x^*|\leq c$.

证明. 分以下两种情况考虑:

(1) 当 $x^*\geq 0$ 时,$(a+b)x^*-(b-a)|x^*|\leq 2c$ 等价于 $ax^*\leq c$. 首先,若 x^* 满足 $ax^*\leq c$,由于 $a\in[a,b]$,则 x^* 是 $[a,b]x\leq c$ 的弱解;其次,对于 $\forall d\in[a,b]$ 有 $ax^*\leq dx^*$,从而 x^* 如果满足 $dx^*\leq c$ 则必然也满足 $ax^*\leq c$. 因此,当 $x^*\geq 0$ 时,x^* 是 $[a,b]x\leq c$ 的弱解当且仅当 x^* 满足 $ax^*\leq c$.

(2) 当 $x^*<0$ 时,$(a+b)x^*-(b-a)|x^*|\leq 2c$ 等价于 $bx^*\leq c$. 首先,若 x^* 满足 $bx^*\leq c$,由于 $b\in[a,b]$,则 x^* 是 $[a,b]x\leq c$ 的弱解;其次,对于 $\forall d\in[a,b]$ 有 $bx^*\leq dx^*$,从而 x^* 如果满足 $dx^*\leq c$ 则必然也满足 $bx^*\leq c$. 因此,当 $x^*<0$ 时,x^* 是 $[a,b]x\leq c$ 的弱解当且仅当 x^* 满足 $bx^*\leq c$. 证毕.

总体来说,一个单变量区间线性不等式的弱解集合可以是无界的、非凸的、非连通的. 其非凸性质源于定理 1 中的非线性因子 $|x^*|$. 在一个给定的关于 x 的半轴上,变量 x 的符号是固定的,因此一个单变量区间线性不等式的弱解集合与每个半轴的交可通过一个(凸)区间来表示.

例 1. 对于单变量区间线性不等式 $[-1,1]x\leq -1$,其弱解集合与 x 正半轴的交得到 $-x\leq -1$,即 $x\in[1,+\infty]$,其弱解集合与 x 负半轴的交得到 $x\leq -1$,即 $x\in[-\infty,-1]$. 因此,$[-1,1]x\leq -1$ 的弱解集合为 $x\in[-\infty,-1]\cup[1,+\infty]$,如图 2 所示.

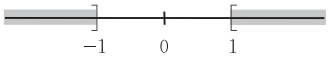


图 2 单变量区间线性不等式 $[-1,1]x\leq -1$ 的弱解集合

关于弱解及弱解集合的定义可以提升到由多个、多维单变量区间线性不等式所构成的单变量区间线性系统(其中每个约束只涉及一个变量)上. 同样地,单变量区间线性系统的弱解集合也可能是无界的、非凸的、非连通的. 但是,在每个(闭)象限上所有变量的符号是固定的,因此一个单变量区间线性系统的弱解集合与每个象限的交可通过一个(凸)盒来表示.

3　单变量区间线性不等式抽象域

本节提出一个新的数值抽象域——单变量区间线性不等式抽象域.考虑到该抽象域与经典区间抽象域的相似性,本文亦把该抽象域简称为扩展区间抽象域,记作 $extItv$.其主要思想是对经典区间抽象域进行扩展,采用区间作为变量系数.类似于已有大部分数值抽象域,$extItv$ 也是通过某类特定约束系统的解集来设计和构建抽象域的.在表示方法上,$extItv$ 使用单变量区间线性不等式约束(形如$[a,b]x\leq c$).类似于经典区间抽象域,扩展区间抽象域也是一个非关系型抽象域,只能表示单个变量的性质,即每个变量各自的取值范围.

3.1　抽象域表示

首先,根据 2.2 节关于弱解的定义,一个单变量区间线性不等式 $\varphi:[a,b]x\leq c$ 可以按如下方式进行约简:

$$\zeta(\varphi)\triangleq\begin{cases}0\leq 1, & 0\in[a,b]\wedge c\geq 0\\ ax\leq c, & x\geq 0 \text{ 或 } a\geq 0\wedge c\geq 0 \text{ 或 } b\leq 0\wedge c\leq 0\\ bx\leq c, & x\leq 0 \text{ 或 } a\geq 0\wedge c\leq 0 \text{ 或 } b\leq 0\wedge c\geq 0\\ [a,b]x\leq c, & x \text{ 符号无限制且 } a<0<b\wedge c<0\end{cases},$$

其中,$0\leq 1$ 表示不等式 φ 是恒成立的,可以从不等式系统中删除;$ax\leq c$ 和 $bx\leq c$ 通过分别除以 a 和 b 的绝对值可以进一步约简成 $x\leq c'$ 或 $-x\leq -c'$ 形式;$[a,b]x\leq c$(其中 $a<0<b\wedge c<0$)通过除以 c 的绝对值可以进一步约简成$[a',b']x\leq -1$ 的形式,其中 $a'<0<b'$.

定理 2.　给定一个单变量区间线性不等式 $\varphi:[a,b]x\leq c$,有 $\Sigma_{\exists}(\varphi)=\Sigma_{\exists}(\zeta(\varphi))$,即约简操作 ζ 是精确的.

证明.

(1)若 $0\in[a,b]\wedge c\geq 0$ 时,线性不等式 $0x\leq c$ 的解集为整个实数集,因此 $\varphi:[a,b]x\leq c$ 恒成立,是冗余的;

(2)当 $x\geq 0$ 时,根据定理 1,有 x 是$[a,b]x\leq c$ 的弱解当且仅当 x 是 $ax\leq c$ 的弱解;当 $a\geq 0\wedge c\geq 0$ 时,$[a,b]x\leq c\Leftrightarrow x\leq sup(c/[a,b])\Leftrightarrow x\leq c/a\Leftrightarrow ax\leq c$,其中 sup 表示取区间的上界;当 $b\leq 0\wedge c\leq 0$ 时,$[a,b]x\leq c\Leftrightarrow x\geq inf(c/[a,b])\Leftrightarrow x\geq c/a\Leftrightarrow ax\leq c$,其中 inf 表示取区间的下界;

(3)当 $x\leq 0$ 或 $a\geq 0\wedge c\leq 0$ 或 $b\leq 0\wedge c\geq 0$ 时,按(2)同理可得 $\Sigma_{\exists}([a,b]x\leq c)=\Sigma_{\exists}(bx\leq c)$;

(4)当 x 无符号限制且 $a<0<b\wedge c<0$ 时,$\zeta(\varphi)=\varphi$.　　证毕.

对于任意变量 x,我们使用关于 x 的单变量区间线性系统 $\vec{a}x\leq\vec{c}$ 来表示变量 x 的可能取值范围,其中 $\vec{a}$ 是一个区间向量,$\vec{c}$ 是一个标准的(非区间)实数向量.其弱解集合 $\Sigma_{\exists}(\vec{a}x\leq\vec{c})=\{x^*\in\mathbb{R}\mid\exists\vec{a}\in\vec{a}.\vec{a}x^*\leq\vec{c}\}$称为一个扩展区间,其中 $\vec{a}$ 是一个标准的实数向量,而每个点 x^* 表示变量 x 一个可能的取值.通过上述约简操作 ζ,一个扩展区间的约束表示(任意多个约束)可以转化成如下正规型(至多 3 个约束):

$$\left\{\begin{array}{c}-x\leq -c_1,\\ x\leq c_2,\\ [a,b]x\leq -1\end{array}\right\},$$

其中,$c_1\in\mathbb{R}\cup\{-\infty\}$,$c_2\in\mathbb{R}\cup\{+\infty\}$,$a<0<b$.若 $c_1=-\infty$,则表示 $-x\leq -c_1$ 恒成立,可以从约束系统中删除;同理,若 $c_2=+\infty$,则表示 $x\leq c_2$ 恒成立,可以从约束系统中删除;若$[a,b]=[-\infty,+\infty]$,即 $[a,b]=\top_i$,本文表示$[a,b]x\leq -1$ 恒成立[①],可以从约束系统中删除.给定一个扩展区间 $\boldsymbol{I}$,$\boldsymbol{I}$ 的正规型扩展区间记为 $\Theta(\boldsymbol{I})$,根据定理 2 有 $\gamma(\Theta(\boldsymbol{I}))=\gamma(\boldsymbol{I})$.方便起见,本文采用$[[c_1,c_2],[a,b]]$来表示上述正规型扩展区间.

几何上,一个扩展区间在数轴上对应的图形区域可能是非凸的、非连通的、无界的,但是它与数轴 $\mathbb{R}$ 上每个半轴的交是一个可能为空的(凸)区间.具体而言,一个扩展区间的形状只可能是图 3 中所示 7 种情况之一.并且,数轴上对应的图形区域可以被一个扩展区间所精确描述,当且仅当该图形区域在几何上同时满足如下两个性质:

(1)至多存在两个非连通的子区域;

(2)如果存在两个非连通的子区域,那么这两个子区域必须分布在数轴原点的两侧(且原点不在这两个子区域内).

基于抽象解释理论,实数域(具体域)与扩展区间域(抽象域)之间的关系可以通过如下 Galois 连接来表示

$$(\wp(\mathbb{R}),\leq)\underset{\alpha_{ei}}{\overset{\gamma_{ei}}{\leftrightarrows}}(ExtItvs,\sqsubseteq_{ei}),$$

其中,$ExtItvs$ 是 $\mathbb{R}$ 上的所有正规型扩展区间所组成的集合$\{[[c_1,c_2],[a,b]]\mid c_1\in\mathbb{R}\cup\{-\infty\},c_2\in\mathbb{R}\cup$

① 实际上,$[-\infty,+\infty]x\leq -1$ 可以表示 $x\neq 0$.方便起见,本文定义$[-\infty,+\infty]x\leq -1$ 的解集为整个实数 $\mathbb{R}$.

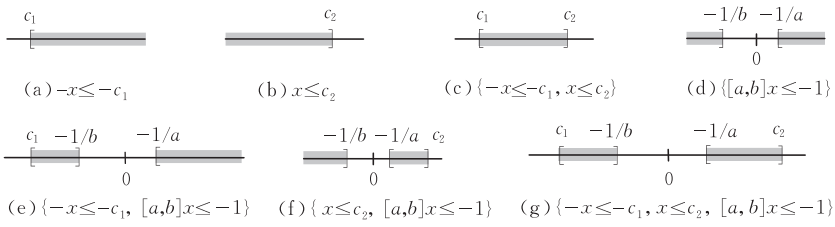


图 3 一个扩展区间的 7 种可能几何形状

$\{+\infty\}, a\in\mathbb{R}, b\in\mathbb{R}, c_1\leq c_2, a<0<b\}\cup\{[[c_1,c_2],[-\infty,+\infty]]\mid c_1\in\mathbb{R}\cup\{-\infty\}, c_2\in\mathbb{R}\cup\{+\infty\}, c_1\leq c_2\}\cup\{\perp_{ei}\}$. $\mathbb{R}$ 上的正规型扩展区间集合 $ExtItvs$ 构成了一个完全格($ExtItvs, \sqsubseteq_{ei}, \sqcap_{ei}, \sqcup_{ei}, \perp_{ei}, \top_{ei}$). 其中,$\top_{ei}\triangleq[[-\infty,+\infty],[-\infty,+\infty]]$表示整个实数$\mathbb{R}$;$\perp_{ei}$表示空扩展区间. 同时,定义$[[c_1,c_2],[a,b]]\triangleq\perp_i$,其中$[c_1,c_2]=\perp_i$或$c_1>-1/b\wedge -1/a>c_2$. 具体函数 $\gamma_{ei}\in[ExtItvs\to\wp(\mathbb{R})]$定义为 $\gamma_{ei}(\perp_{ei})\triangleq\varnothing$, $\gamma_{ei}([[c_1,c_2],[a,b]])\triangleq\{x\in\mathbb{R}\mid -x\leq -c_1, x\leq c_2, [a,b]x\leq -1\}$. 抽象函数 $\alpha_{ei}\in[\wp(\mathbb{R})\to ExtItvs]$定义为 $\alpha_{ei}(\varnothing)\triangleq\perp_{ei}$, $\alpha_{ei}(S)\triangleq[[c_1,c_2],[a,b]]$,其中 $S\subseteq\mathbb{R}$ 是一个实数集合,$[c_1,c_2]=[\min S, \max S]$,

$$[a,b]=\begin{cases}[-\infty,+\infty], & 0\in S \text{ 或} \{x>0\mid x\in S\}=\varnothing \text{ 或} \\ & \{x<0\mid x\in S\}=\varnothing \\ \left[\dfrac{-1}{\min\{x>0\mid x\in S\}}, \dfrac{-1}{\max\{x<0\mid x\in S\}}\right], & \text{否则}\end{cases}.$$

$ExtItvs$ 上的偏序关系$\sqsubseteq_{ei}$定义为$[[c_1,c_2],[a,b]]\sqsubseteq_{ei}[[c_1',c_2'],[a',b']]$当且仅当$[[c_1,c_2],[a,b]]=\perp_{ei}$或$[c_1,c_2]\sqsubseteq_i[c_1',c_2']\wedge[a,b]\sqsubseteq_i[a',b']$. 操作$\sqcap_{ei}$,$\sqcup_{ei}$将在 3.2 节给出.

类似于区间抽象环境,扩展区间抽象环境$[Vars\to ExtItvs]$把每个程序变量映射到一个表示其值范围的扩展区间上. 对应地,在 n-维空间中,如果一个(可能非连通的)区域对应的约束系统中的每个约束形如$[a,b]x_j\leq c$,其中 c 是一个实数常量或$+\infty$,$[a,b]$是一个实数区间,$j\in\{1,2,\cdots,n\}$,则该区域称为是一个 n-维扩展盒(extended box). 同样地,几何上,一个扩展盒可以是非凸的、非连通的、无界的. 图 4 中给出了二维平面上一些扩展盒的例子. 其中,图 4(a)所示扩展盒对应单变量区间线性约束系统$\{-x\leq 1, x\leq 1, [-1,1]x\leq -1, -y\leq 1, y\leq 1, [-1,1]y\leq -1\}$,图 4(b)所示扩展盒对应单变量区间线性约束系统$\{-x\leq 2, x\leq 3, [-0.5,1]x\leq -1, -y\leq 3, y\leq 2, [-1,0.5]y\leq -1\}$.

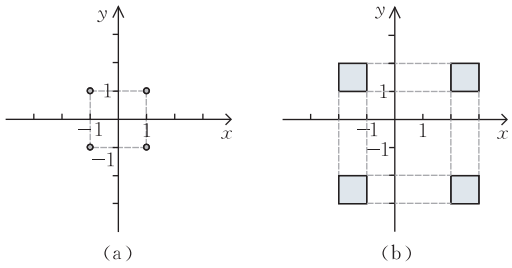


图 4 扩展盒示例

n 维空间上的一个(可能非连通的)图形区域可以被一个扩展盒所精确描述,当且仅当该图形区域同时满足如下两个性质:

(1)每个闭象限内至多存在一个盒;

(2)对于每个变量 x,$x\leq 0$ 和 $x\geq 0$ 将整个空间划分为两个子空间,如果两个子空间同时分别存在某些子区域,则其中一个子空间内所有子区域的集合并(union)在 x 轴上的投影扩展区间必须与另一个子空间内所有子区域的集合并(union)在 x 轴上的投影扩展区间相等.

图 5 中给出了二维平面上一些不能够被扩展盒所精确描述的图形区域. 其中,图 5(a)不满足上述性质(2),图 5(b)不满足上述性质(1).

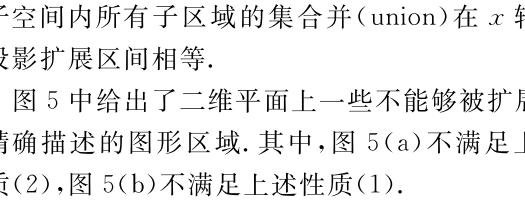


图 5 非扩展盒示例

给定 n-维扩展盒 $\boldsymbol{B}$,记该扩展盒在变量 x_j 对应数轴上的投影扩展区间为 $\boldsymbol{B}x_j$. 扩展盒上的序关系 $\boldsymbol{B}\sqsubseteq_{eb}\boldsymbol{B}'$定义为$\gamma(\boldsymbol{B})\subseteq\gamma(\boldsymbol{B}')$. 给定两个 n-维扩展盒 $\boldsymbol{B}$ 和$\boldsymbol{B}'$,$\boldsymbol{B}\sqsubseteq_{eb}\boldsymbol{B}'$成立当且仅当对于任意$x_j\in\{x_1,\cdots,x_n\}$,皆有 $\boldsymbol{B}x_j\sqsubseteq_{ei}\boldsymbol{B}'x_j$. 关于变量 x_j 的单变量区间线性不等式 φ:$[a,b]x_j\leq c$ 被一个扩展盒 $\boldsymbol{B}$ 蕴含,记为 $\boldsymbol{B}\models\varphi$,当且仅当 $\boldsymbol{B}x_j\sqsubseteq_{ei}\Theta(\varphi)$.

3.2　抽象域操作

本节给出扩展区间抽象域上用于静态分析的常用域操作.为了方便描述,我们以下面两个正规型扩展区间作为可能的操作参数:

$$\boldsymbol{I}'[[c_1',c_2'],[a',b']]\triangleq\begin{Bmatrix}-x\leq -c_1',\\ x\leq c_2',\\ [a',b']x\leq -1\end{Bmatrix},$$

$$\boldsymbol{I}''=[[c_1'',c_2''],[a'',b'']]\triangleq\begin{Bmatrix}-x\leq -c_1'',\\ x\leq c_2'',\\ [a'',b'']x\leq -1\end{Bmatrix},$$

并以如下正规型扩展区间来表示域操作的结果扩展区间:

$$\boldsymbol{I}=[[c_1,c_2],[a,b]]\triangleq\begin{Bmatrix}-x\leq -c_1,\\ x\leq c_2,\\ [a,b]x\leq -1\end{Bmatrix}.$$

(1)交(meet)与接合(join)

两个扩展区间的交仍然是一个扩展区间.$\boldsymbol{I}'$与$\boldsymbol{I}''$的交定义为

$$\boldsymbol{I}=\boldsymbol{I}'\sqcap_{ei}\boldsymbol{I}''\triangleq\begin{cases}\bot_{ei}, & I'=\bot_{ei}\text{或}I''=\bot_{ei}\\ [[c_1,c_2],[a,b]], & \text{否则}\end{cases},$$

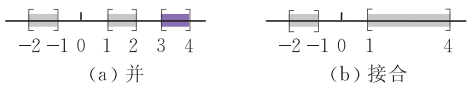
其中,

$$\begin{cases}[c_1,c_2]=[c_1^*,c_2^*] & [a,b]=\top_i, & [a^*,b^*]=\top_i\text{或}[c_1^*,c_2^*]=\bot_i\\ [c_1,c_2]=[c_1^*,c_2^*] & [a,b]=[a^*,b^*], & c_1^*\leq -1/b^*\wedge -1/a^*\leq c_2^*\\ [c_1,c_2]=[\max\{-1/a^*,c_1^*\},c_2^*] & [a,b]=\top_i, & c_1^*>-1/b^*\wedge -1/a^*\leq c_2^*\\ [c_1,c_2]=[c_1^*,\min\{-1/b^*,c_2^*\}] & [a,b]=\top_i, & c_1^*\leq -1/b^*\wedge -1/a^*>c_2^*\end{cases},$$

其中,$[c_1^*,c_2^*]=[c_1',c_2']\sqcap_i[c_1'',c_2'']$,$[a^*,b^*]=[a',b']\sqcap_i[a'',b'']$.

为了抽象程序流图中的控制流接合(control-flow join),我们需要计算抽象环境的并(union).然而,两个扩展区间的并未必是一个扩展区间.例如,给定扩展区间$\boldsymbol{I}'=\{x\leq 2,-x\leq 2,[-1,1]x\leq -1\}$与$\boldsymbol{I}''=\{x\leq 4,-x\leq -3\}$,$\boldsymbol{I}'$与$\boldsymbol{I}''$的并对应的图形区域如图 6(a)所示,该区域不满足 3.1 节关于数轴上图形区域能被一个扩展区间所精确表示的判断之性质(1),即"至多存在两个非连通的子区域".

为此,我们使用下面定义的接合操作$\sqcup_{ei}$,来计算包含该并的最小的扩展区间,以保证精度损失最少.



(a)并　　(b)接合

图 6　两扩展区间的并与接合

$$\boldsymbol{I}=\boldsymbol{I}'\sqcup_{ei}\boldsymbol{I}''\triangleq\begin{cases}\boldsymbol{I}', & I''=\bot_{ei}\\ \boldsymbol{I}'', & I'=\bot_{ei}\\ [[c_1,c_2],[a,b]], & \text{否则}\end{cases},$$

其中,

$$[c_1,c_2]=[c_1',c_2']\sqcup_i[c_1'',c_2''],$$

$$[a,b]=\begin{cases}[a',b']\sqcup_i[a'',b''], & [a',b']\neq\top_i\wedge[a'',b'']\neq\top_i\\ [\min\{a'',-1/c_1'\},\max\{b'',-1/c_2'\}, & [a',b']=\top_i\wedge[a'',b'']\neq\top_i\wedge(c_1'>0\vee c_2'<0)\\ [\min\{a',-1/c_1''\},\max\{b',-1/c_2''\}, & [a'',b'']=\top_i\wedge[a',b']\neq\top_i\wedge(c_1''>0\vee c_2''<0)\\ [\min\{-1/c_1',-1/c_1''\},\max\{-1/c_2',-1/c_2''\}, & [a',b']=\top_i\wedge[a'',b'']=\top_i\wedge((c_1'>0\wedge c_2''<0)\vee(c_1''>0\wedge c_2'<0))\\ \top_i, & \text{否则}\end{cases}.$$

注意:$\sqcap_{ei}$是精确的,没有精度损失,即$\gamma(\boldsymbol{I}'\sqcap_{ei}\boldsymbol{I}'')=\gamma(\boldsymbol{I}')\cap\gamma(\boldsymbol{I}'')$.然而$\sqcup_{ei}$不是精确的,$\boldsymbol{I}'\sqcup_{ei}\boldsymbol{I}''$只能保证是两个扩展区间$\boldsymbol{I}'$与$\boldsymbol{I}''$的(集合)并的上近似,即$\gamma(\boldsymbol{I}')\cup\gamma(\boldsymbol{I}'')\subseteq\gamma(\boldsymbol{I}'\sqcup_{ei}\boldsymbol{I}'')$.但是,不一定有$\gamma(\boldsymbol{I}'\sqcup_{ei}\boldsymbol{I}'')\subseteq\gamma(\boldsymbol{I}')\cup\gamma(\boldsymbol{I}'')$,例如,对于图 6 中的例子,$2.5\in\boldsymbol{I}'\sqcup_{ei}\boldsymbol{I}''$,但是$2.5\notin\gamma(\boldsymbol{I}')\cup\gamma(\boldsymbol{I}'')$.

(2)算术操作(arithmetic)

当参与扩展区间算术计算的任意一个参数为$\bot_{ei}$时,其计算结果也为$\bot_{ei}$.下面,我们假设参与扩展区间算术计算的参数均不为$\bot_{ei}$.

①求反$-_{ei}\boldsymbol{I}'$

$\boldsymbol{I}'$的求反运算定义如下:

$$\boldsymbol{I}=-_{ei}\boldsymbol{I}'=[[c_1,c_2],[a,b]],$$

其中,$[c_1,c_2]=[-c_2',-c_1']$,$[a,b]=[-b',-a']$.

②加法$\boldsymbol{I}'+_{ei}\boldsymbol{I}''$

$\boldsymbol{I}'$与$\boldsymbol{I}''$的和定义为

$$\boldsymbol{I}=\boldsymbol{I}'+_{ei}\boldsymbol{I}''=[[c_1,c_2],[a,b]],$$

其中,

$$[c_1,c_2]=[c_1',c_2']+_i[c_1'',c_2''],$$

$$[a,b]=\begin{cases}[a^*,b^*], & a^*<0 \text{ 且 } b^*>0\\ \top_i, & \text{否则}\end{cases},$$

其中，a^*,b^* 定义如下

$$\begin{cases}a^*=\dfrac{-1}{\min\{c_1'-1/a'',c_1''-1/a'\}},\ b^*=\dfrac{-1}{\max\{c_2'-1/b'',c_2''-1/b'\}}, & [a',b']\neq\top_i\wedge[a'',b'']\neq\top_i\\ a^*=\dfrac{-1}{c_1'-1/a''},\ b^*=\dfrac{-1}{c_2'-1/b''}, & [a',b']=\top_i\wedge[a'',b'']\neq\top_i\\ a^*=\dfrac{-1}{c_1''-1/a'},\ b^*=\dfrac{-1}{c_2''-1/b'}, & [a',b']\neq\top_i\wedge[a'',b'']=\top_i\\ a^*=-\infty,\ b^*=+\infty, & [a',b']=\top_i\wedge[a'',b'']=\top_i\end{cases}.$$

③减法 $\boldsymbol{I}'-_{ei}\boldsymbol{I}''$

$\boldsymbol{I}'$与$\boldsymbol{I}''$的差定义为

$$\boldsymbol{I}=\boldsymbol{I}'-_{ei}\boldsymbol{I}''=\boldsymbol{I}'+_{ei}(-_{ei}\boldsymbol{I}'').$$

④乘法 $\boldsymbol{I}'\times_{ei}\boldsymbol{I}''$

$\boldsymbol{I}'$与$\boldsymbol{I}''$的积定义为

$$\boldsymbol{I}=\boldsymbol{I}'\times_{ei}\boldsymbol{I}''=[[c_1,c_2],[a,b]],$$

其中，$[c_1,c_2]=[c_1',c_2']\times_i[c_1'',c_2'']$，$[a,b]$的取值分如下 4 种情况讨论：

(i)若$[a',b']\neq\top_i$且$[a'',b'']\neq\top_i$，则

$$[a,b]=[\min\{-a'a'',-b'b''\},\max\{-a'b'',-a''b'\}].$$

(ii)若$[a',b']=\top_i$且$[a'',b'']\neq\top_i$，则

$$[a,b]=\begin{cases}\top_i, & c_1'\leq 0\leq c_2'\\ [a''/c_1',b''/c_1'], & c_1'>0\\ [b''/c_2',a''/c_2'], & c_2'<0\end{cases}.$$

(iii)若$[a',b']\neq\top_i$且$[a'',b'']=\top_i$，则

$$[a,b]=\begin{cases}\top_i, & c_1''\leq 0\leq c_2''\\ [a'/c_1'',b'/c_1''], & c_1''>0\\ [b'/c_2'',a'/c_2''], & c_2''<0\end{cases}.$$

(iv)若$[a',b']=\top_i$且$[a'',b'']=\top_i$，则

$$[a,b]=\top_i.$$

⑤除法 $\boldsymbol{I}'/_{ei}\boldsymbol{I}''$

$\boldsymbol{I}'$的倒数定义为

$$\boldsymbol{I}=1/_{ei}\boldsymbol{I}'=[[c_1,c_2],[a,b]],$$

其中，

$$\begin{cases}[c_1,c_2]=1/_i[c_1',c_2'],[a,b]=\top_i, & [a',b']=\top_i\\ [c_1,c_2]=[-b',-a'],[a,b]=[-c_2',-c_1'], & \text{否则}\end{cases}.$$

$\boldsymbol{I}'$与$\boldsymbol{I}''$的商定义为

$$\boldsymbol{I}=\boldsymbol{I}'/_{ei}\boldsymbol{I}''=\boldsymbol{I}'\times_{ei}(1/_{ei}\boldsymbol{I}').$$

(3)加宽(widening)/变窄(narrowing)

类似于区间完全格，扩展区间所构成的完全格也是无穷格.因此，在程序分析过程中，对于有循环或递归调用的程序，为了加速不动点计算并保证其终止性，需要设计加宽算子.而变窄算子则能以递减迭代的方式，进一步精化应用加宽算子所得到的近似不动点并保证递减迭代的终止性.

扩展区间抽象域上的加宽∇_{ei}定义为

$$\perp_{ei}\nabla_{ei}\boldsymbol{I}'=\boldsymbol{I}'\nabla_{ei}\perp_{ei}\triangleq\boldsymbol{I}',$$

$$\boldsymbol{I}=\boldsymbol{I}'\nabla_{ei}\boldsymbol{I}''\triangleq[[c_1,c_2],[a,b]],$$

其中，

$$[c_1,c_2]=[c_1',c_2']\nabla_i[c_1'',c_2''],$$

$$[a,b]=\begin{cases}[a',b'], & [a',b']\supseteq[a'',b'']\\ \top_i, & \text{否则}\end{cases}.$$

扩展区间抽象域上的变窄Δ_{ei}定义为

$$\boldsymbol{I}=\boldsymbol{I}'\Delta_{ei}\boldsymbol{I}''\triangleq[[c_1,c_2],[a,b]],$$

其中，

$$[c_1,c_2]=[c_1',c_2']\Delta_i[c_1'',c_2''],$$

$$[a,b]=\begin{cases}[a'',b''], & [a',b']=\top_i\\ [a',b'], & \text{否则}\end{cases}.$$

(4)迁移函数(transfer functions)

赋值语句和条件测试语句是程序语言中的基本指令.在基于抽象解释的程序分析中，这两种指令的语义一般通过迁移函数(即从抽象环境集合到抽象环境集合的映射)来抽象.赋值迁移函数$[\![v:=expr]\!]^\sharp$将给出赋值语句$v:=expr$执行之后的新环境，该环境与旧环境的唯一区别就是变量v的值发生了改变.测试迁移函数$[\![expr_1\bowtie expr_2\,?]\!]^\sharp$则将从旧环境中过滤掉不满足布尔表达式$expr_1\bowtie expr_2$的那些环境.

①赋值迁移函数(assignment transfer function)

在抽象环境$X^\sharp$下，采用本文之前定义的扩展区间算术操作，基于结构归纳，可以得到表达式$expr$的抽象语义值$[\![expr]\!]^\sharp X^\sharp$，如图 7 所示.

$$\begin{array}{ll}[\![expr]\!]^\sharp:(Vars\to ExtBoxes)\to ExtBoxes;&\\ [\![v]\!]^\sharp X^\sharp & \triangleq X^\sharp(v);\\ [\![[a,b]]\!]^\sharp X^\sharp & \triangleq [a,b];\\ [\![-expr]\!]^\sharp X^\sharp & \triangleq -_{ei}([\![expr]\!]^\sharp X^\sharp);\\ [\![expr_1\odot expr_1]\!]^\sharp X^\sharp & \triangleq ([\![expr_1]\!]^\sharp X^\sharp)\odot_{ei}([\![expr_2]\!]^\sharp X^\sharp)\\ & \text{其中}\odot\in\{+,-,\times,/\}.\end{array}$$

图 7 数值表达式基于扩展区间抽象域的抽象语义

由此,扩展区间抽象域上的赋值迁移函数可以定义为

$$[\![v:=expr]\!]^\sharp X^\sharp \triangleq X^\sharp[v\mapsto [\![expr]\!]^\sharp X^\sharp].$$

② 测试迁移函数(test transfer function)

给定某个可能的环境 $\rho\in X^\sharp$,布尔表达式 $expr_1\bowtie expr_2$ 的语义定义如下:

$$\begin{aligned}[\![expr_1\bowtie expr_2]\!]\rho\triangleq\{&\text{true},\text{若 }\exists val_1\in[\![expr_1]\!]\rho,\\ &val_2\in[\![expr_2]\!]\rho,val_1\bowtie val_2\}\cup\\ &\{\text{false},\text{若 }\exists val_1\in[\![expr_1]\!]\rho,\\ &val_2\in[\![expr_2]\!]\rho,val_1\not\bowtie val_2\},\\ &\text{其中}\bowtie\in\{=,\neq,<,\leq\}.\end{aligned}$$

测试迁移函数旨在过滤掉不满足布尔表达式 $expr_1\bowtie expr_2$ 的那些环境,定义如下:

$$\begin{aligned}&[\![expr_1\bowtie expr_2?]\!]^\sharp X^\sharp\triangleq\\ &\quad\{\rho\mid\rho\in X^\sharp,\text{true}\in[\![expr_1\bowtie expr_2]\!]\rho\}.\end{aligned}$$

对测试语句 $expr_1\bowtie expr_2$? 进行抽象时,除了需要分别对 $expr_1$ 和 $expr_2$ 进行计值以判定 $expr_1\bowtie expr_2$ 为真($\{\text{true}\}$)还是为假($\{\text{false}\}$)还是不确定($\{\text{true},\text{false}\}$)外,在 $expr_1\bowtie expr_2$ 的计值为不确定($\{\text{true},\text{false}\}$)时,还需要利用测试条件 $expr_1\bowtie expr_2$ 来对其中所涉及的变量的取值范围进行精化.记表达式 $expr$ 中的变量集合为 $Vars(expr)$.对于 $Vars(expr_1\bowtie expr_2)$ 中的任何一个变量 v,$expr_1\bowtie expr_2$ 可用来精化变量 v 的取值范围.

在抽象环境 $X^\sharp$ 下,关于变量 $v\in Vars(expr_1\bowtie expr_2)$,$expr_1\bowtie expr_2$ 可以通过一个形如 $[a,b]v\leq c$ 的单变量区间线性不等式来抽象:变量 v 的符号出现保持不变,其它变量 v' 均使用其抽象值 $X^\sharp(v')$ 来替换,变量 v 的系数以及不含变量 v 的项使用扩展区间算术来计算,并转化成形如 $[[c_1',c_2'],[a',b']]v\leq[[c_1'',c_2''],[a'',b'']]$ 的不等式,该不等式可进一步抽象成 $[c_1',c_2']v\leq c_2''$;严格不等式"$<$"使用"$\leq$"来抽象.设 $expr_1\bowtie expr_2$ 导出的关于 v 的单变量区间线性不等式所对应的正规型扩展区间为 $\boldsymbol{I}''$,并设测试迁移函数执行之前,变量 v 的取值范围为扩展区间 $\boldsymbol{I}'$,即 $X^\sharp(v)=\boldsymbol{I}'$,则测试迁移函数执行之后,变量 v 新的取值范围为扩展区间 $\boldsymbol{I}\triangleq\boldsymbol{I}'\sqcap_{ei}\boldsymbol{I}''$.

例如,给定条件测试 $y*x+2z\leq[1,2]x+1$,在抽象环境 $X^\sharp(x)=[[-\infty,+\infty],[-\infty,+\infty]]$,$X^\sharp(y)=[[-1,2],[-1,1]]$,$X^\sharp(z)=[[2,4],[-\infty,+\infty]]$ 下,对于变量 x,该条件测试将导出 $[[-1,2],[-1,1]]x+2*[[2,4],[-\infty,+\infty]]\leq[[1,2],[-\infty,+\infty]]x+1$,即 $[[-3,1],[-\infty,+\infty]]x\leq[[-7,-3],[-\infty,+\infty]]$,进一步抽象得 $[-3,1]x\leq-3$.因此,测试迁移函数执行之后,变量 x 新的取值范围为扩展区间 $[[-\infty,+\infty],[-1,1/3]]$.

当条件测试语句 $expr_1\bowtie expr_2$ 中的变量数大于 1 时,各个变量间的取值范围将因为这个约束而存在互相依赖关系.因此,可以使用约束传播技术来缩紧(tighten)每个变量的取值范围:将一个变量的约束信息(取值范围)传播到其它变量上.变量 v 的取值范围可以利用其它变量当前的取值范围,按照测试迁移函数来缩紧.而变量 v 缩紧后的取值范围又可以用来缩紧其它变量的取值范围.

例如,对于条件测试约束 $z=x+y$,通过分解,我们可以得到如下 3 个投影函数:$\boldsymbol{I}_z\leftarrow\boldsymbol{I}'_{x+y}\cap\boldsymbol{I}'_z$,$\boldsymbol{I}_x\leftarrow\boldsymbol{I}'_{z-y}\cap\boldsymbol{I}'_x$,$\boldsymbol{I}_y\leftarrow\boldsymbol{I}'_{z-y}\cap\boldsymbol{I}'_y$.设在测试迁移语句执行之前,$x,y,z$ 的取值范围为 $\boldsymbol{I}'_x=[[-5,5],[-1/5,1/5]]$,$\boldsymbol{I}'_y=[[1,3],[-\infty,+\infty]]$,$\boldsymbol{I}'_z=[[3,7],[-\infty,+\infty]]$.投影 $\boldsymbol{I}_z\leftarrow\boldsymbol{I}'_{x+y}\cap\boldsymbol{I}'_z$ 将导出测试迁移函数 $[\![z=\boldsymbol{I}'_{x+y}?]\!]^\sharp X^\sharp$ 即 $[\![z=[[-4,8],[-1/6,1/2]]?]\!]^\sharp X^\sharp$,该迁移函数执行后 $\boldsymbol{I}_z=[[6,7],[-\infty,+\infty]]$.同理可得,测试迁移函数执行之后,$\boldsymbol{I}_x=[[5,5],[-\infty,+\infty]]$,$\boldsymbol{I}_y=[[1,2],[-\infty,+\infty]]$.

4　实现与实验结果

至此,本文提出的单变量区间线性不等式抽象域 $extItv$ 都是在实数域 $\mathbb{R}$ 上考虑的.在编码实现时,考虑到计算效率,我们使用双精度浮点数开发了该抽象域的原型系统.整个原型系统的浮点实现基于向外舍入的浮点区间算术(即,上界向上舍入;下界向下舍入),从而保证了该实现的可靠性.我们还把该原型实现适配到为数值抽象域提供了公共接口的开源数值抽象域库 APRON① 上.本文使用支持 APRON 库的静态分析工具 Interproc② 来开展程序分析实验.我们对 Interproc 进行了扩展,使其支持区间形式的输入数据(如带区间系数的表达式和约束).为了评估 $extItv$ 域的分析精度和效率,本文把基于 $extItv$ 域的分析与基于经典区间抽象域的分析在结果不变式和性能等方面进行了比较.

为了展示扩展区间抽象域 $extItv$ 的表达能力,

① http://apron.cri.ensmp.fr/library/
② http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/interproc/

下面给出两个简单、有代表性的循环程序以及静态分析器所产生的不变式，如图 8、9 所示. 对于图 8 中的程序 Prog1(即本文图 1 中的示例程序)，基于经典区间抽象域的分析结果将得出结论：程序第 5 行的赋值语句存在“除零错”. 而从基于扩展区间抽象域的分析可知：循环体中变量 x 的取值范围是$\{-x\leq -1, x\leq 1, [-1,1]x\leq -1\}$，即 $x=-1 \vee x=1$，由此可以断定程序第 5 行的赋值语句是安全的. 图 9 中的程序 Prog2 包括两个阶段，首先在内部循环中增加变量 y 的值，然后再在外部循环中增加变量 x 的值. 基于扩展区间抽象域的分析可以证明在程序 Prog2 中①处，x 的取值范围是$\{-x\leq 20, x\leq 19, [-0.1,0.05]x\leq 1\}$，即 $x=-20 \vee 10\leq x\leq 19$，$y$ 的取值范围是$\{-y\leq 20 \wedge [-0.1,0.05]y\leq -1\}$，即 $y=-20 \vee y\geq 10$，这比基于区间抽象域的分析结果要精确.

```
Prog 1
1.  real x, y;
2.  x:=1;
3.  while (true) do
4.      x:=-x;
5.      y:=1/x;①
6.  done;
```

位置	经典区间抽象域	扩展区间抽象域
①	$x\in[-1,1]$ $y\in[-\infty,+\infty]$	$-x\leq -1, x\leq 1, [-1,1]x\leq -1,$ $-y\leq -1, y\leq 1, [-1,1]y\leq -1$

图 8 程序 Prog1(左)及产生的不变式(右)

```
Prog 2
1:  int x, y;
2:  x:=-20;
3:  y:=-20;
4:  while (x≤19) do
5:  ① while (y≤9) do
6:      y:= y + 1;
7:    done;
8:    if (x≤0) x:=y;
9:    x:= x+1;
10: done;
```

位置	经典区间抽象域	扩展区间抽象域
①	$x\in[-20,19]$ $y\in[-20,+\infty]$	$-x\leq 20 \wedge x\leq 19 \wedge [-0.1,0.05]x\leq -1$ $\wedge -y\leq 20 \wedge [-0.1,0.05]y\leq -1$

图 9 程序 Prog2(左)及产生的不变式(右)

本文采用的实验平台为 Fedora 9 Linux 操作系统，768MB 物理内存，Intel P4 1.6GHz 单核 CPU 处理器. 针对一系列测试程序，表 1 给出并比较了基于两种抽象域的分析的实验结果. Interproc 采用传统的基于抽象解释的不动点迭代方法，并支持“延迟”加宽策略. 在本文的实验中，Interproc 的加宽算子延迟参数(加宽算子应用之前的迭代次数)设置为 3. 测试程序中的 Prog1、2，分别对应图 8、9 所示的例子.

表 1 测试程序实验结果

程序名	变量数	以毫秒为单位的计算时间(迭代次数)		精度比较
		扩展区间抽象域	区间抽象域	
Prog1	2	6.999(4)	6.000(4)	>
Prog2	2	13.998(6)	12.999(6)	>
multiwhile	3	14.997(10)	9.999(10)	>
sas98	3	21.997(5)	11.999(5)	>
policy	2	9.998(4)	7.999(4)	>
bubblesort	5	25.996(7)	20.997(7)	=
heapsort	8	45.993(4)	27.996(4)	=
maccathy91	4	7.998(4)	6.999(4)	=

表 1 中的“精度比较”栏比较了两种抽象域的分析结果不变式. 其中，“>”表示扩展区间抽象域能得到比区间抽象域更强的值范围信息. “=”表示扩展区间抽象域得到的值范围与区间抽象域一样. 如果程序中存在符号未受限制的变量，扩展区间抽象域常能够找到一些有意义的非连通值范围信息，比如表 1 中的前 5 个测试程序. 另外，从表 1 可以看出，扩展区间抽象域的计算代价比区间抽象域略高一些，但是总计算时间一般不会超过区间抽象域计算时间的两倍. 相比多面体抽象域等关系型数值抽象域而言，扩展区间抽象域仍是一种轻量级的非关系型抽象域.

5　相关工作

(1)值范围分析

Harrison[1]最先基于区间分析并采用范围传播(range propagation)和范围分析(range analysis)两种方法对程序变量进行了值范围分析研究,其中,范围传播因为不是归纳的不能有效处理程序中的循环结构,而范围分析则忽略了循环中的条件结构从而影响了分析的精度. Cousot 和 Cousot[5]把区间分析纳入到抽象解释框架下并提出了区间抽象域,其中加宽与变窄算子能够有效地加速循环和递归程序结构的不动点计算,并能保证分析的终止性和分析结果的可靠性. 基于加宽与变窄算子的分析一般不是得到精确的值范围,而只是精确值范围的上近似. 为了弥补区间抽象域所缺失的变量间关系表达能力,Sankaranarayanan 等人[11]提出了一个符号范围抽象域(一种受限形式的多面体)来表示和利用程序变量间的关系用以推导程序变量的符号界信息.

另一方面,为了避免使用加宽与变窄算子所带来的不精确性,Su 与 Wagner[12]对数据流等式进行了"严格"分析,针对一类整数范围约束,给出了一个基于图表示的、多项式时间内可以找到最优范围解的算法. Rugina 等[13]基于线性规划技术为程序变量(包括指针、数组下标、被访问的存储区域等)提供了一个新的符号界分析框架. 这种基于线性规划的方法虽然避免了启发式的加宽/变窄过程,但是提供的只是一个弱的证明系统,因为该方法中线性不等式系统所蕴含的新的结果不等式是通过比较系数来构造的,因此可能产生弱的不变式而不能得到精确的界信息.

与上述工作相比,本文的工作属于第一类方法,即采用基于加宽与变窄算子的抽象解释方法来分析变量的值范围信息. 本文提出的单变量区间线性不等式抽象域虽然不能表示变量之间的关系,但是可以表示非凸的、非连通的值范围信息. 而上述相关工作都主要集中在找到尽量精确的凸的界信息. 并且基于线性规划或者符号界的值范围分析算法的时空复杂度一般都比较高,相对而言,本文的扩展区间抽象域与经典的区间抽象域一样,都是一种轻量级、计算效率很高的方法.

(2)数值抽象域

近 30 年来,基于抽象解释的程序分析与验证技术取得了长足的发展. 该领域已出现了许多表达能力、运行效率各有特色的数值抽象域,包括区间域[5]、凸多面体域[6]、八边形域[8]、区间多面体域[14]等. 这些抽象域面向各种不同的数值性质,并在分析精度和计算代价间取得合理折衷. 按照表达性质的能力,数值抽象域一般可分为如下 3 类:非关系型抽象域,只能表示单个变量的性质;关系型抽象域,能够表示任意多个变量间的(某类)任意关系;弱关系型抽象域,只能表达变量间的受限数值关系(比如,限制关系中变量数、变量系数等). 本文提出的单变量区间线性不等式抽象域 $extItv$ 属于非关系型抽象域. 非关系型抽象域还包括常量传播($x_i=c_i$)、符号($\pm x_i\geq 0$)[5]、区间[5]、简单同余($x_i\equiv a_i \bmod b_i$)[15]、区间同余($x_i\in[a_i,b_i] \bmod c_i$)等.

目前,绝大部分数值抽象域只能表示凸的(必然是连通的)性质,包括区间域[5]、八边形域[8]、凸多面体域[6]等,很少有抽象域能够天然地表示非凸的性质,例如,max-plus 多面体域[16]等. 其中,能够天然地表示非连通的性质的抽象域就更少了,例如,同余域[15]. 而本文提出的 $extItv$ 域则能够自然地表示某类非连通性质. 因此,在程序分析中能够起到不可替代的作用.

最近,Chen 等人[14]把区间线性代数引入到静态分析中,提出了区间多面体抽象域,作为凸多面体抽象域的区间扩展(本文部分作者参与了文献[14]中所述工作). 区间多面体抽象域属于关系型抽象域,可以表示任意多个变量间的区间线性不等式关系. 而本文提出的 $extItv$ 域属于非关系型抽象域,域中每个区间线性不等式约束只涉及 1 个变量,只能表示单个变量的性质. 相对于区间多面体抽象域,$extItv$ 域虽然表达能力较弱但是有着较高的计算效率.

6　总结和进一步工作

本文提出了一个新的数值抽象域,称为单变量区间线性不等式抽象域(亦称为扩展区间抽象域,$extItv$). 该抽象域能够表示和操作单变量区间线性不等式约束(形如$[a,b]x\leq c$). 其表达能力强于经典的区间抽象域,还能够天然地表示某类非凸、非连通的性质. 同时,其域操作可以基于简单、高效的区间算术来构造. 本文实验结果表明:程序分析时,$extItv$域能够发现一些别的数值抽象域(如区间抽象域)所不能发现的、有意义的非凸或非连通的值范围信息,而且有着较高的计算效率. 作为一个轻量级

的非关系型抽象域，$extItv$ 域计算效率高、可扩展性强并且存在紧致的空间表示，可以在实际大规模的程序分析中得以应用.

将来的工作包括使用机器整数来实现 $extItv$ 抽象域，并把 $extItv$ 抽象域扩展到区间同余代数上.

参 考 文 献

[1] Harrison W H. Compiler analysis of the value ranges for variables. IEEE Transactions on Software Engineering, 1977, 3(3): 243-250

[2] Simon A. Value-Range Analysis of C Programs: Towards Proving the Absence of Buffer Overflow Vulnerabilities. Berlin: Springer, 2008

[3] Ji Meng-Luo, Wang Huai-Min, Li Meng-Jun, Dong Wei, Qi Zhi-Chang. An value range analysis based on abstract interpretation and generalized monotone data flow framework. Journal of Computer Research and Development, 2006, 43(11): 2020-2026(in Chinese)
(姬孟洛，王怀民，李梦君，董威，齐治昌. 一种基于抽象解释和通用单调数据流框架的值范围分析方法. 计算机研究与发展，2006，43(11)：2020-2026)

[4] Cousot P, Cousot R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints//Proceedings of the 4th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages(POPL'77). Los Angeles, California, USA, 1977: 238-252

[5] Cousot P, Cousot R. Static determination of dynamic properties of programs//Proceedings of the 2nd International Symposium on Programming. Paris, France, 1976: 106-130

[6] Blanchet B, Cousot P, Cousot R, Feret J, Mauborgne L, Mine A, Monniaux D, Rival X. A static analyzer for large safety-critical software//Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03). San Diego, California, USA, 2003: 196-207

[7] Cousot P, Halbwachs N. Automatic discovery of linear restraints among variables of a program//Proceedings of the 5th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages(POPL'78). Tucson, Arizona, USA, 1978: 84-97

[8] Mine A. The octagon abstract domain. Higher-Order and Symbolic Computation, 2006, 19(1): 31-100

[9] Mine A. Symbolic methods to enhance the precision of numerical abstract domains//Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'06). Charleston, South Carolina, USA, 2006: 348-363

[10] Chen L, Mine A, Cousot P. A sound floating-point polyhedra abstract domain//Proceedings of the 6th Asian Symposium on Programming Languages and Systems (APLAS'08). Bangalore, India, 2008: 3-18

[11] Sankaranarayanan S, Ivancic F, Gupta A. Program analysis using symbolic ranges//Proceedings of the 14th International Static Analysis Symposium (SAS'07). Kongens Lyngby, Denmark, 2007: 366-383

[12] Su Z, Wagner D. A class of polynomially solvable range constraints for interval analysis without widenings. Theoretical Computer Science(TCS), 2005, 345(1): 122-138

[13] Rugina R, Rinard M C. Symbolic bounds analysis of pointers, array indices, and accessed memory regions//Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation(PLDI'00). Vancouver, Britith Columbia, Canada, 2000: 182-195

[14] Chen L, Mine A, Wang J, Cousot P. Interval polyhedra: An abstract domain to infer interval linear relationships//Proceedings of the 16th International Static Analysis Symposium (SAS'09). Los Angeles, CA, USA, 2009: 309-325

[15] Granger P. Static analysis of arithmetical congruences. International Journal of Computer Mathematics, 1989, 30(3): 165-190

[16] Allamigeon X, Gaubert S, Goubault E. Inferring Min and Max invariants using Max-plus polyhedra//Proceedings of the 15th International Static Analysis Symposium(SAS'08). Velencia, Spain, 2008: 189-204

CHEN Li-Qian, born in 1982, Ph. D. candidate. His research interests include program analysis and verification, abstract interpretation.

WANG Ji, born in 1969, Ph. D., professor, Ph. D. supervisor. His research interests include high confidence software, software methodology, software engineering, etc.

HOU Su-Ning, born in 1985, M. S. candidate. His research interests focus on source code analysis.

Background

The correctness of programs has always been considered a great challenge. The theory of abstract interpretation provides a general framework to automatically infer dynamic properties of computer systems at compile-time. Especially, the value range information of program variables is crucial for many applications, such as compiler optimization, program analysis and verification, etc. However, most existing numerical abstract domains (such as, intervals, polyhedra, octagons) under the abstract interpretation framework have limitations in expressing non-convex properties, which may lead to imprecision during the analysis.

The main contribution of this paper is a new non-convex numerical abstract domain, so-called an abstract domain of one-variable interval linear inequalities (i. e., of the form $[a,b]x\leq c$). It is more expressive than the classic interval abstract domain and allows expressing certain non-convex, unconnected properties. Moreover, its domain operations can be implemented via efficient algorithms based on interval arithmetic. As a non-relational abstract domain, it has high scalability and thus can be applied in large-scale program analysis in practice.

This work is supported by the National Natural Science Foundation of China under grant Nos. 60725206, 60621003, 60673118 and the Natural Science Foundation of Hunan Province under grant No. 07JJ1011. These projects aim to build a systematic framework for analysis and verification of software system, and to provide a methodology to check certain important properties for some specific software systems by integrating various formal methods, such as model checking, static analysis, theorem proving, etc.