分类号 TP301 学号 05069059

U D C 密级 公 开

工学博士学位论文

基于区间线性抽象域的可靠浮点及非凸静态分析

博士生姓名 陈 立 前

学 科 专 业 计算机科学与技术

研 究 方 向 计算机软件与理论

指 导 教 师 王 戟 教授

国防科学技术大学研究生院

二〇一〇年四月

Sound floating-point and non-convex static analysis using interval linear abstract domains

Candidate: Chen Liqian
Supervisor: Prof. Wang Ji

A dissertation
Submitted in partial fulfillment of the requirements
for the degree of Doctor of Engineering
in Computer Science
Graduate School of National University of Defense Technology
Changsha, Hunan, P.R.China
April, 2010

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科学技术大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：基于区间线性抽象域的可靠浮点及非凸静态分析

学位论文作者签名：_____ 日期： 年 月 日

学位论文版权使用授权书

本人完全了解国防科学技术大学有关保留、使用学位论文的规定。本人授权国防科学技术大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：基于区间线性抽象域的可靠浮点及非凸静态分析

学位论文作者签名：_____ 日期： 年 月 日

作者指导教师签名：_____ 日期： 年 月 日

目 录

表 目 录

图 目 录

摘　要

软件的可信性已成为现代软件质量问题的焦点。发现并排除软件缺陷是构建可信软件的重要途径，这对于航天、国防、汽车等安全攸关应用尤其重要。科学与工程应用相关程序，特别是一些安全攸关嵌入式软件，一般与数学和物理有着紧密的联系，从而不可避免地会包含大量数值计算。因此，许多程序缺陷往往和程序中的数值性质密切相关，比如除零错、算术溢出、数组越界等运行时错误。

抽象解释是一种对程序语义进行近似（或抽象）的通用理论，并为静态分析提供了一个通用的框架。而抽象域则是抽象解释框架下的核心要素，通过选择特定的语义表示和操作算法来发现所关注的性质。数值抽象域可用来自动发现程序中的数值性质，即程序变量间的数值关系。经过数十年的发展，在基于抽象解释的数值程序分析领域，出现了许多面向不同数值性质、表达能力多样的数值抽象域。然而，大部分已有数值抽象域（尤其是关系型抽象域）都是采用多精度有理数来实现以保证可靠性，时空开销较大，从而影响到分析的计算效率和可扩展性；而且，大部分已有数值抽象域在表达能力方面存在凸性局限性，对析取的处理能力较弱；此外，已有数值抽象域尚不支持实际程序分析中常出现的区间系数。

本文主要研究数值抽象域的设计和实现。研究目标包括：探索数值抽象域的可靠浮点构造方法，以通过浮点实现来提高计算效率；设计新的数值抽象域使其能够表达非凸性质并支持区间系数，以通过增强表达能力来提高分析精度。本文主要工作如下：

1）提出了浮点多面体抽象域，作为经典凸多面体域的一种可靠浮点构造方法。凸多面体域是目前表达能力最强、应用最广泛的数值抽象域之一，但是其基于多精度有理数的实现在可扩展性和易处理性方面受到限制。本文仅基于凸多面体的约束表示（无需生成子表示）并采用浮点系数，给出了凸多面体抽象域的一种可靠浮点实现方法，以通过浮点实现来提高凸多面体域的计算效率和处理能力。由于基于约束的多面体域的复杂度主要源于其高代价的接合操作（即凸闭包），本文还提出了一系列低代价的弱接合操作。

2）把区间线性代数引入到程序分析中，提出了一系列支持区间系数的区间线性抽象域。区间（算术）为构造可靠的浮点抽象域提供了一种自然且有效的途径，并且在实现浮点多面体域过程中也产生了凸多面体表达能力之外的区间线性约束。本文提出了区间多面体抽象域，用以推导程序中变量间的区间线性不等式关系。该域

可以看作是经典凸多面体域的区间系数扩展版本。作为一种受限情形，本文还提出了行阶梯形区间线性等式抽象域，用以推导区间线性等式关系。其表示方法基于区间线性等式系统的行阶梯形式，使得该域具有多项式的时空复杂度。该域可以看作是经典仿射等式抽象域的区间系数扩展版本。通过采用区间线性系统的弱解语义，这些区间线性抽象域能够天然地表达某类拓扑非凸（甚至非连通）性质。本文采用浮点数并基于向外舍入的区间算术可靠地实现了这些抽象域。

3）提出了一系列可表达非凸性质的线性绝对值抽象域。绝对值是数学中的一个基本概念，常用来描述数学或物理模型中的分段线性特征，并且可以表达非凸性质。本文给出并证明了线性绝对值不等式系统、区间线性不等式系统、广义线性互补问题（系统）三者之间的等价性，以及线性绝对值等式系统、水平线性互补问题（系统）两者之间的等价性。基于凸多面体的双重描述法，给出了广义线性互补问题（系统）的双重描述法，并作为其应用还给出了一种求解绝对值线性规划问题的新方法。在此基础上，本文提出了线性绝对值不等式抽象域（用来推导线性绝对值不等式/区间线性不等式/广义线性互补不等式关系）和线性绝对值等式抽象域（用来推导线性绝对值等式/水平线性互补等式关系）。线性绝对值不等式域是经典多面体域的一般化，虽然表达能力与区间多面体域相同，但是其域操作都是具体域上对应操作的最佳抽象。而线性绝对值等式域是经典仿射等式域的一般化。基于广义线性互补系统的双重描述法并采用多精度有理数实现了这两个抽象域。

本文提出的数值抽象域在分析精度和计算效率间进行了不同的权衡，可以面向不同的应用。这些抽象域都在数值抽象域库 APRON 中得以实现，并且实验结果令人鼓舞。

关键词：静态分析，抽象解释，数值抽象域，多面体，浮点，区间线性系统

ABSTRACT

Trustworthiness of software has become the center of attention when considering modern software quality. Finding bugs before release is fundamental to build trustworthy software, which is extremely important for safety-critical applications such as aerospace, defense and automotive. Scientific and engineering programs especially safety-critical embedded software are usually related to mathematics and physics, and thus often involve a lot of numerical computations. Hence, many program bugs including division by zero, arithmetic overflows and array out-of-bounds, are closely related to numerical properties in the program.

Abstract interpretation is a theory of semantics approximation, allowing the design of sound and efficient static analyses. Abstract domains are a key ingredient in this framework: They enable semantic choices (what properties to infer) and algorithmic choices (how to compute). In this thesis, we focus on design and implementation of numerical abstract domains that are used to automatically discover numerical properties over program variables. Over the past decades, a wide variety of numerical abstract domains have been proposed. However, most existing abstract domains (especially relational ones) need arbitrary precision rational numbers to build implementations, which may degrade the time and memory efficiency. Most of them can only represent convex properties, and hence have limitations in dealing with disjunctions. Besides, currently there is no abstract domain that supports interval variable coefficients which appear naturally in real-life program analysis.

The goal of this thesis is to explore more efficient numerical abstract domains, using sound floating-point constructions, and to design more precise ones, allowing non-convex properties and interval coefficients.

Major contributions of this thesis are listed as follows.

1) We present a so-called floating-point polyhedra abstract domain. The classical convex polyhedra domain is one of the most powerful and commonly used abstract domains in the field, but rational implementations may suffer from scalability problems. To solve this issue, we present an implementation using floating-point arithmetic without sacrificing soundness, based on a constraint-only representation

using floating-point coefficients. Since its complexity mainly comes from the costly join operation (i.e., polyhedral convex hull), a series of cheap weak join operations are then proposed.

2) We introduce interval linear algebra to static analysis and propose interval linear abstract domains that support interval coefficients. Intervals are naturally suited to construct sound floating-point abstract domains. First, a so-called interval polyhedra domain is proposed to infer interval linear inequalities. Then, a restricted abstract domain is proposed based on a system of interval linear equalities in row echelon form, polynomial in time and memory. The two domains can be considered as interval extensions of the existing convex polyhedra domain and the affine equality domain respectively. By interpreting solutions as weak solutions of interval linear systems, both domains can express certain non-convex (even unconnected) properties. They are soundly implemented using floating-point numbers based on interval arithmetic with outward rounding.

3) We present linear absolute value abstract domains that natively allow to express non-convex properties. Absolute value (AV) is fundamental in mathematics and often used to describe piecewise linear behavior. The equivalence among linear AV inequality systems, interval linear inequality systems and extended linear complementarity problem (XLCP) systems is stated. We construct a double description method for XLCP on top of that for polyhedra, based on which a new method for solving AV linear programming problems is shown. On this basis, we propose an abstract domain of linear AV equalities, and an abstract domain of linear AV inequalities that has the same expressiveness as interval polyhedra domain but enjoys optimal transfer functions. They are implemented using rational numbers based on the double description method for XLCP.

The numerical abstract domains presented in this thesis provide different tradeoffs between precision and efficiency, and can be applied to different problems. They are implemented in the APRON library, and experimental results are encouraging.

Key Words: Static analysis, Abstract interpretation, Numerical abstract domain, Polyhedra, Floating-point, Interval linear system

第一章　绪论

1.1　研究背景

随着计算机的应用日益广泛和深入，计算机软件作为计算机的灵魂和应用的载体，承担着越来越重要的功能。因此，人们对软件质量的要求也越来越高。另一方面，随着软件需求的不断增加，软件系统日趋庞大和复杂，软件质量越发难以保证，系统越来越脆弱，发生故障和失效的概率也越来越大。换而言之，软件并非总是完全可以让人信任的，其行为和结果并不完全符合人们的预期，这就是所谓的“软件可信性”问题 [1]。具体而言，软件的可信性质包括正确性、可靠性、安全性、时效性等方面。毋庸置疑，软件的可信性已成为现代软件质量问题的焦点。

软件正确性是传统软件开发的首要目标，也是目前可信软件的极为重要的性质。然而，软件本质上是人类思维创造的产物，软件开发的过程实际上是基于人类思考的一种心智活动。因此，软件中难免会存在各种缺陷或错误。正如软件度量专家 Jones 指出的，在新编写的程序代码中，平均每个功能点（或特征）会存在大约 5 个缺陷，但是在产品发布之前往往只有 85% 的缺陷被修正了 [2]。McConnell 也早在文献 [3] 中指出，按照工业平均经验，交付软件中平均每 1000 行代码会包含约 15 到 50 个错误。美国标准技术研究院 NIST 曾在 2002 年发布的一份报告 [4]中对软件错误问题的影响广度进行了如下描述：软件错误的影响力是巨大的，因为美国现在几乎每个业务都依赖于软件来进行开发、生产、发布以及产品和服务的售后支持。该报告还进一步指出，美国每年因软件失效所造成的经济损失大约有 595 亿美元，而其中至少三分之一（约 222 亿美元）可以通过采用更好的测试（或分析）技术来避免。

特别是在一些重要领域如国防、航空航天、金融、医疗等，软件可信性尤其是正确性显得尤为重要，一旦出现问题可能造成严重影响甚至灾难性后果。例如，因为软件缺陷，1991年 2 月美国爱国者导弹防御系统未能拦截一枚伊拉克飞毛腿导弹的攻击，导致美军军营中 28 名士兵死亡 [5]；同样因为软件缺陷，1996 年 6月欧洲航天局 Ariane 5 型火箭在其首次发射的 37 秒之后爆炸，造成的间接经济损失达数十亿美元 [6]。

正因如此，通过软件测试、分析与验证来提高软件质量，一直是计算机科学中的重要研究内容 [7]。其中，软件测试是发现软件缺陷的有效手段。但是，由于穷尽

所有可能的输入是不现实的，软件测试在理论上存在不完全性问题，即：无法证明系统不存在缺陷，也不能证明其符合一定的属性。形式化分析与验证是保证软件质量的重要途径，可以证明一个系统不存在某个缺陷或符合某个属性。目前常用的形式化分析与验证技术主要包括模型检验、定理证明、静态分析等。

实际上，在如何保证软件可信性方面，形式化分析与验证技术获得了持续关注。早在 60 年代晚期，以 Floyd [8]、Hoare [9]、Naur [10] 为代表的一批计算机科学家率先研究了程序正确性证明技术。经过数十年的发展，形式化分析与验证技术取得了长足的进步，从纯粹学术研究逐步走向实际工业应用。近年来，继 Intel 在其硬件芯片设计中成功运用形式验证技术后 [11]，微软也相继启动了一系列面向软件的形式化分析与验证研究项目（如 SLAM [12]、SDV [13]、Terminator [14] 等），并在 Windows 操作系统尤其是其驱动程序的分析与验证中取得了满意的效果。特别值得注意的是，近年来，国际上出现了许多提供程序分析与验证服务的高科技公司（例如 Coverity、Klocwork、Grammatech、Kestrel Technology、PolySpace、AbsInt 等），他们研制的程序分析和验证工具已成功应用于航空、航天、汽车、医疗、金融等领域的安全攸关软件甚至一些开源项目的源代码保证中。在此形势下，2005 年左右以图灵奖获得者 Hoare 为代表的学者们提出了 Verified Software（可验证软件）计划 [15, 16]，其目标之一就是通过尽可能自动化的方法来验证软件程序的正确性。目前，Verified Software 已被学界认为是计算机科学领域本世纪的重大挑战之一。如今，程序分析与验证不仅得到了学术界和工业界的普遍重视，还引起了社会公众的广泛关注（见《科学美国人》、《经济学人》的近期相关文章 [17, 18, 2]）。

程序分析与验证问题很重要，但是也很难完美解决。根据 1953 年的 Rice 定理[19]，任何关于程序行为的问题都是不可判定的。因此，程序分析与验证问题在最坏情况下是不可判定的。换而言之，不存在任何一种自动化的方法能够证明所有程序的非平凡性质。Rice 定理在实践中的影响就是所有程序分析与验证工具都需要做抽象：把不可判定的或者可判定但复杂度很高的原问题通过抽象转化为可判定的且复杂度不高的“抽象”问题。

抽象解释理论 [20] 对抽象进行了形式化，并提供了通用的理论框架来构造、比较和结合各种抽象。近年来，Astrée [21, 22]、PolySpace [23]、Fluctuat [24, 25]等基于抽象解释的静态分析工具在工业界大规模软件尤其是航空航天控制软件的分析与验证中得到了成功应用 [26]。比如，Astrée 成功地对空客 A340（约 13.2 万行 C 代码）、A380（约 35 万行 C 代码）等系列飞行控制软件进行了验证 [27, 28]，这是验证工具在验证规模上的重大突破 [29]。因此，基于抽象解释的程序分析和验证也必然

是应对 Verified Software 重大挑战的一个重要研究方面 [30]。

1.1.1 程序的数值性质

在科学计算、网络协议、过程控制系统、时间和混成控制系统、制造系统、生化反应系统、信息－物理融合系统 CPS（Cyber-Physical System）等实际应用程序或模型中，往往需要使用数值量来表示有特定含义的物理量，比如计算机学科里的时钟、循环计数、缓冲区长度、网络带宽等，物理学科里的速度、距离等，生化学科里的反应物浓度、温度等。尤其，在航空航天、国防军工、交通等安全攸关领域中，控制软件的一个重要特征就在于其本质上都是基于物理模型的，因此不可避免地会包含大量数值计算。

本文关注的正是程序的数值性质，即程序中与数值型变量相关的性质。例如，变量 x 的取值范围。更为复杂的数值性质则需要考虑多个变量间所具有的数值关系，比如，在程序点 l 处变量 x, y, z 间总满足关系 $x + 2y - z \leq 0$。这样的数值性质可以用来发现程序中的运行时错误，比如除零错（除数为零）、算术溢出（数值超出了计算机的表示范围，如整数溢出、浮点数溢出等）、数组越界（数组成员引用下标超出定义范围）等。值得注意的是，这些错误看似简单，但是据美国 CERT 的统计数据显示，这些错误却在已知安全漏洞中占据了很大比重 [31, 32]。

程序中还有很多其他性质或错误与数值性质紧密相关。比如，目前最常见最重要的一种安全问题—— “缓冲区溢出”问题（即程序向缓冲区内写入的数据超出了缓冲区本身的容量，包括堆溢出、栈溢出、格式化字符串溢出等种类），本质上就是关于新写入起始地址 $addr'$、新写入数据大小 $size'$ 与缓冲区起始地址 $addr$、缓冲区本身容量 $size$ 之间是否满足数值关系 $addr' + size' \leq addr + size$ 的问题。正因如此，关于缓冲区溢出检测方面的许多研究，都是围绕如何发现地址（指针）和长度（范围）之间的数值关系来展开的 [31, 33, 34, 35, 36]。

上述与数值性质相关的程序错误，是针对通用程序而言的。结合实际数学含义或物理含义，程序中还有很多更为丰富的与数值相关的性质。比如，开平方根操作的输入不能是负数；数学三角函数 sin()、cos() 在计算机中实现的输出结果应该总是在 -1 到 1 之间；如果使用整型变量来表示月份，那么无论程序对该变量进行怎样的操作，该变量的取值范围都应该是 1 到 12 之间的整数；求平均数的程序的结果值应该是在两个输入值之间；两个正数的和应该比两个数都大。

尤其，在嵌入式控制程序中，往往会存在更为复杂的性质，诸如“如果一个函数的输入的取值范围是有界的，那么该函数的输出的取值范围也是有界的”。例如，飞

行控制系统中软件速率限制器的输出应该总是在一个给定有界范围内（本文第 2.5.2 节将给出一个软件速率限制器的示例程序），超出该范围就可能危机飞行安全。事实上，本文之前给出的两个因软件缺陷导致灾难事故发生的著名案例本质上都是因程序中数值方面的问题导致的："Ariane 5 型火箭首发爆炸事故"是因为浮点数转化为整数时发生算术溢出，"爱国者导弹拦截飞毛腿导弹失败事件"是因为浮点舍入误差累计过大（更为详细的说明请见第 2.5.2 节）。

同样，根据 Rice 定理，程序的数值性质也是不可判定的，因此也需要应用抽象解释理论来对程序语义进行抽象。本文将通过采用基于抽象解释的数值程序分析技术来自动发现程序中的数值性质。

1.1.2 数值抽象解释及其应用

程序的数值性质是程序验证最早关注的性质之一。数值程序分析的相关研究可以追溯到上世纪 70 年代早期 [37, 38, 39, 20]。经过数十年的发展，基于抽象解释的数值程序分析框架已经趋于成熟，并出现了许多面向不同数值性质的数值抽象域。比如，用来跟踪每个变量上下界信息的区间抽象域 [38]、用来推导任意多个变量间线性关系的多面体抽象域 [40]、用来推导任意两个变量间形如 $\pm x \pm y \leq c$ 关系的八边形抽象域 [41]等。这些表达能力多样的数值抽象域的存在，使得数值程序分析可以根据待分析程序的特点以及具体应用的需求选择某个合适的或者结合某几个抽象域来开展程序分析，以在分析精度和计算代价之间取得合理权衡。目前，这些数值抽象域在学术界和工业界都受到了广泛重视，并在航空航天控制软件的分析与验证等实际工业应用中取得了成功应用 [27, 28, 26, 25]。

数值程序分析不仅可用来检测程序中是否有除零错、算术溢出、数组越界等运行时错误，还可用来分析程序中与数值相关的断言以及契约中的前置条件、后置条件和不变式。而且，许多面向程序或模型的分析与验证问题，虽然不是直接跟程序的数值行为相关，但是也常常依赖于数值程序分析技术 [42, 43, 44]。下面给出一些例子：

- 程序终止性分析：数值程序分析不局限于用在可靠安全性（safety）或可达性的分析上，还可以用在活性（如程序终止性）的分析上。证明程序终止性的经典方法是寻找某个秩函数（即每个程序步严格递减的正函数）。不难看出，秩函数跟程序的数值性质具有天然的联系。因此，许多研究都利用数值程序分析来证明程序终止性。比如，Colón 与 Sipma 基于多面体抽象域来合成非平凡的线性秩函数，将合成秩函数问题转化为多面体锥上的求极锥、求交、投影等基本操

作[45, 46]。Chen 等人基于符号计算工具 DISCOVERER 通过求解半代数系统来合成非线性秩函数 [47]。最近，Berdine 等人也使用了如多面体域、八边形域等数值抽象域来诱导出可变性分析（variance analysis），用来证明程序终止性 [48]。类似地，Chawdhary 等人使用多面体域、八边形域等数值抽象域来诱导出秩抽象（ranking abstractions）以对秩关系（中间状态之间的关系）进行抽象，用来证明程序终止性 [49]。

- 程序计算复杂度的（符号）界分析：近来，Gulwani 等人在 SPEED 项目中，面向程序计算复杂度（包括时间/空间复杂度等）的符号界的自动分析开展了一系列研究 [50, 51, 52, 53, 54]，旨在采用静态的方法来评估程序的性能。对于某些嵌入式系统或性能攸关软件来说，计算出程序的最差时间复杂度往往比单纯证明程序终止性更有实际意义。而且，只要分析出某程序的最差时间复杂度是有穷的，那么该程序的终止性也自然得以证明。从某种程度上可以说，复杂度的界分析往往比终止性证明更难。SPEED 的主要思想是采用静态分析技术来计算某函数（或过程）关于其输入的符号复杂度界（symbolic complexity bounds）[54]。同样，程序的符号复杂度界与程序的数值性质紧密相关。比如，许多复杂数据结构上循环的复杂度常依赖于这些数据结构上的一些量化函数 [53]，诸如列表结构的长度、树结构的高度、位向量的字节数等。另外，计算复杂度往往涉及很多非线性数值表达式上的操作，诸如对数、指数、乘、平方根、max 等。为此，Gulavani 与 Gulwani 通过数值抽象域上的两个提升操作，即表达式抽象和 max 操作，来分别处理复杂度界分析中的非线性不变式和带析取的界 [50]。Gulwani 等人还使用数值抽象域来分析存储划分上的数值性质，以跟踪存储划分大小之间的关系[52]。
- 形态分析（Shape Analysis）：除了数值变量上的性质，程序分析通常还关心一些复杂数据结构上的性质（如，某程序点 l 处指针 p 所指向的列表结构是否总是一个良定义的列表）。形态分析主要用于发现或验证程序中所动态分配的堆中的数据结构上的性质。对堆中数据结构进行形态抽象时，通常也需要使用一些数值量来表示这些数据结构上的相关信息，比如链表的长度、树的高度和结点数、树的平衡性等。例如，Rugina [55] 为了证明在 AVL-树中插入一个节点后重新平衡这个树的算法的正确性，在堆抽象中结合了数值抽象来描述一些数值因素，比如节点的平衡因子（其左右子树的高度差）等。最近，Magill 等人 [56, 57] 把数值程序分析与基于分离逻辑（Separation Logic）的形态分析结合起来，以提高形态分析的精度。其主要思想是：通过引入一些插桩变量，按照分离逻辑语

义把操作堆的程序转化为整数程序；然后，对整数程序开展数值程序分析，分析所得数值不变式最终又映射到原堆操作程序上的空间（spatial）不变式。这样，数值程序上的安全性和终止性也保证了原来堆操作程序的安全性和终止性。另外，Chang 与 Rival [58] 也把形态分析与数值程序分析结合起来，开展关系型形态分析，以追踪不相交存储区域之间的关系。

- 软件模型检验：软件模型检验，即应用模型检验技术来对程序进行自动验证，已成为当前模型检验理论与技术的研究热点 [59, 60]。模型检验与程序分析有着紧密的联系，Schmidt 等人早已指出程序分析其实可以看成是基于抽象解释的模型检验 [61, 62]。软件模型检验一般采用谓词抽象技术来对程序状态进行抽象，然后使用反例制导抽象精化（CEGAR）技术来精化模型。谓词抽象把程序验证问题归结为命题推理，比较适合于验证控制驱动型程序及其相关性质，但是并不擅长与数据相关的数值推理。最近，许多研究工作考虑把谓词抽象和数值抽象结合起来进行软件模型检验。Chaki 等人对二元决策图 BDD 进行了扩展，使其能表达某些数值约束，提出了数值决策图 NDD (Numeric Decision Diagrams)、线性决策图 LDD (Linear Decision Diagrams) 等新的数据结构 [63, 64]。Jain 等人在 NEC F-Soft 工具中使用数值抽象域来计算出数值不变式，然后用这些数值不变式来增强程序中的迁移关系，以使得后续基于谓词抽象的验证过程更加有效 [65]。值得关注的是，最近以图灵奖得主 Clarke 为首的一批科学家获得了美国 NSF “计算探险”（Expeditions in Computing）计划的支持，启动了“Model Checking and Abstract Interpretation”(MCAI 2.0) 研究项目 [66, 67]，主要研究面向系统生物学与嵌入式系统的下一代模型检验与抽象解释技术。其主要目标是将模型检验和抽象解释结合起来，为胰腺癌、心房颤动等医疗相关复杂生物系统提供有效的建模和分析途径，并为汽车、航空航天等领域的复杂嵌入式控制系统提供可靠性保障。
- 实时/混成系统验证：在实时/混成系统中，都需要对时间进行建模和处理。实时系统一般采用时间自动机来建模，并根据时钟的取值将状态划分为不同的区域。目前，大部分实时系统模型检验工具都采用基于数据结构 DBM（Difference Bound Matrix）表示的 Zone 区域（对应抽象解释框架下的 Zone 抽象域 [68]）来表示状态集 [69, 70]，比如 Uppaal [71]、Kronos [72]等。混成系统（既包含离散的状态变化，又包含连续的状态变化）则一般通过混成自动机来建模。同样地，数值抽象域（尤其是多面体抽象域）在混成系统验证（尤其是基于线性混成自动机的验证）中取得了广泛应用 [73, 74, 75]。为了能刻画变量随时间的连续变化，一

般是在数值抽象域上引入一个 time-elapse 操作来对此进行抽象 [74]。Henzinger 等人最早基于多面体域开发了 HyTech 工具来验证线性混成系统 [73]。考虑到多面体域的高复杂度，许多相关研究还尝试了多面体域的各种受限形式，比如 orthogonal polyhedra [76, 77] 和 zonotopes [78] 等。最近，Frehse 基于多面体域并采用类似于 HyTech 的技术开发了新的工具 PHAVer，并针对 HyTech 所存在的问题提出了许多有效改进方案，比如通过限制多面体中约束的个数和系数的位数等启发式策略对多面体进行上近似以提高可扩展性等 [75]。Sankaranarayanan 等人则尝试了使用模版多面体域对线性混成系统开展符号化模型检验 [79]。

除此之外，数值程序分析还延伸到许多其他的应用领域，包括编译优化/并行化、指针分析、移动代码分析、低级代码分析、同步程序的分析、最差执行时间分析 WCET、模拟电路的验证等 [42, 43, 44]。

1.2 研究目标及主要结果

抽象解释是一种对程序语义进行可靠近似（或抽象）的通用理论 [20, 80]。该理论为静态程序分析的设计和构建提供了一个通用的框架，并保证了所构建的静态分析的可靠性（即考虑了所有的程序行为）。抽象解释本质上是通过不同程度的近似（或抽象）在分析精度和计算效率之间取得权衡。而抽象域则是抽象解释框架下的核心要素，是语义近似（或抽象）在抽象解释框架实例化所得静态分析中的具体体现，也是静态分析在分析精度和计算效率之间进行权衡的落脚点。

本文聚焦在数值抽象域的设计和实现上，针对当前数值程序分析相关研究所存在的某些问题，从改进分析精度和提高计算效率两方面来开展研究，并取得了一些成果。

1.2.1 研究动机

本节将讨论当前已有数值抽象域在设计和实现上所存在的一些问题和局限性。

(1) 实现数值抽象域的数据类型选择问题

数值抽象域通常是在数学意义上的实数域 $\mathbb{R}$ 或有理数域 $\mathbb{Q}$上设计的。这些数域对于加法、减法、乘法和除法（除数不为0）运算具有封闭性，因此在这些数域上编写程序来实现抽象域时容易保证抽象域的可靠性。但是，我们知道，计算机硬件所能表达的数都是有穷范围和有穷精度的，所以，实际程序中所操作的数并非数学

意义上“完美”的实数、有理数或整数，而是“不完美”的机器整数或浮点数（本文统称为“机器数”）。

可靠性（Soundness）是抽象域设计的中心目标。为此，在编写程序来实现数值抽象域时，一般采用多精度有理数（比如，使用 GMP 库 [81]）。多精度有理数的内部实现一般是为每一个有理数分别存储其分子和分母，而且分子和分母都通过多精度整数来表示。多精度有理数（理论上）没有表示范围和精度的限制，并且运算属于精确计算，不存在“机器数”可能面临的算术溢出或浮点运算的舍入误差问题。但是，使用多精度有理数来实现抽象域存在如下不足：

- 运行速度慢（时间开销大）：不像机器数上的运算可以被当前计算机硬件直接支持，多精度有理数上的运算需要通过软件的方法来实现，一个运算往往需要多条机器指令才能完成。而且，分子和分母之间需要频繁且耗时的最大公约数（GCD）求解及化简计算；
- 耗费大量存储空间（空间开销大）：需要为每一个有理数分别存储其分子和分母，而且分子和分母都需要使用多精度整数来存储。因此，一个有理数往往需要使用很多个机器字长来表示；
- 容易导致“大系数”问题（数值约束中系数的分子或分母的数值很大）：由于多精度有理数运算属于精确计算，域表示中数值约束的系数经过多次乘/除操作后，容易出现大分子或大分母。大系数不仅占用更多空间，还会消耗更长时间的 GCD 求解和化简计算。而且，这种“大系数”问题会在基于“迭代”的程序分析过程中传播，导致程序分析的复杂度和计算效率进一步恶化，甚至导致程序分析任务不能在给定时间内完成。

简而言之，基于多精度有理数的实现在时间和空间上的开销很大，可扩展性受到极大限制，难以在实际大规模程序分析中应用。

为此，我们需要考虑使用“机器数”来实现数值抽象域。其中，选择之一是使用机器整数。基于机器整数的实现方法主要基于如下事实：对于任意一条有理数系数的约束，将该约束乘以有理数系数的最小公分母（LCD）将得到一条整数系数的约束，且该整数系数的约束与原有理数系数的约束是等价的，即两者的解集是相等的。例如，对于多面体抽象域 [40]中的一条有理数系数的线性不等式约束 $\varphi : \Sigma_i \frac{a_i}{a'_i} x_i \leq \frac{b}{b'}$，其中 a_i, a'_i, b, b' 为数学意义上的整数，设该约束中有理数系数的最小公分母为 d。那么，将整条约束 φ 乘以 d，将得到一条整数约束 $\varphi' : \Sigma_i a''_i x_i \leq b''$，其中 $a''_i \stackrel{\text{def}}{=} a_i \times \frac{d}{a'_i}, b'' \stackrel{\text{def}}{=} b \times \frac{d}{b'_i}$。不难看出，$\varphi$ 与 φ' 是等价的，且 a''_i 和 b'' 必然是分别不小于 a_i 和 b 的整数。接下来，我们考虑使用机器整数来表示数学意义上的整数约束 φ'。

如果数学意义上的整数约束 φ' 中有系数超出了机器整数的表示范围，为了保证可靠性，需要把该约束从抽象域表示中舍弃。但是，这将导致抽象域在瞬间损失大量精度。不难看出，实际上，基于机器整数的实现比基于多精度有理数的实现更容易出现“大系数”问题（因为对分母进行通分了），只是基于机器整数的实现将通过舍弃约束（即损失精度）的方式来处理该问题。

值得注意的是，对于基于机器整数或多精度有理数实现的数值抽象域，尤其是关系型数值抽象域（如多面体抽象域），“大系数”问题在实际程序分析过程中经常出现 [82, 83, 75]。实际上，即使待分析源程序中只涉及小整数值，也可能导致“大系数”问题（文献 [83] 第 5 章给出了很多具体的例子）。尤其，对于浮点程序，一般需要采用区间线性化的抽象技术把浮点程序（即变量为浮点型，运算为浮点运算）抽象成实数程序（即变量为实数型，运算为精确运算），在抽象后的实数程序中不可避免地会出现高阶的数值（如 2^{-149}、2^{-1074} 等，更为详细的描述见本文第 2.5.2 节）。因此，就面向浮点程序的分析而言，“大系数”问题对于采用多精度有理数或机器整数的实现来说是不可回避的。

本文的主要思想就是采用另外一种机器数——浮点数来实现抽象域。相比多精度有理数，使用浮点数具有如下优势：

- 运行速度快（节省时间开销）：浮点运算被当前主流硬件直接支持，运算性能高，运行速度快，能有效节省程序分析的时间；
- 具有紧凑的存储格式（节省空间开销）：一个浮点数只需要一个机器字长就可以表示，并且浮点数存在多种精度格式（如 32 位单精度浮点格式、64 位双精度浮点格式等），可根据待分析程序的数值特点灵活选择；
- 可表示的数值范围足够大（能有效缓解“大系数”问题）：比如 64 位双精度浮点格式可表示的数值范围大约是 $[-1.7\mathrm{e}^{+308}, 1.7\mathrm{e}^{+308}]$，这个数值范围对于当前大部分程序分析相关应用来说是足够的；
- 支持一种渐进式的精度损失过程（能有效缓解“大系数”问题）：浮点数是有理数的有穷子集，其分布是离散的、非均匀的，相邻两个浮点数间的距离随着浮点数值的增大而增大。比如，为了表示高阶有理数 $\frac{2^{1024}-1}{2^{1024}}$，浮点表示中可以简单地使用 1 来近似，从而有效地避免了“大系数”问题及其衍生问题。另外，浮点舍入误差所带来的精度损失，在某些时候可以加速程序分析过程不动点迭代的收敛速度，使得计算结果提前到达包含了所有程序行为的稳定的区域。

相比机器整数，使用浮点数具有明显的优势：基于相同的二进制位数，浮点数可表示的数值范围远远大于机器整数可表示的数值范围，从而可以极大地缓解因算

术上溢而舍弃约束导致大量精度损失的问题。比如 64 位双精度浮点格式可表示的数值范围（大约是 $[-1.7\mathrm{e}^{+308}, 1.7\mathrm{e}^{+308}]$）远远大于 64 位机器整数可表示的数值范围（大约是 $[-9.2\mathrm{e}^{+18}, 9.2\mathrm{e}^{+18}]$）。

使用浮点数据类型来编程实现数值抽象域，为提高数值程序分析的计算效率提供了有效途径，给实际大规模数值程序分析带来了机遇。但是，同时这也是一个极大的挑战。挑战主要包括如下几个方面：

- 如何保证基于浮点数实现的抽象域的可靠性？首先，我们重申可靠性是抽象域设计和实现的中心目标。但是，由于浮点舍入误差的存在，浮点计算的结果不是精确的。而且，实数集 $\mathbb{R}$ 和有理数集 $\mathbb{Q}$ 上一些良好的代数性质如结合律、分配律等，对于浮点算术不再适用。另外，浮点计算的结果极大地依赖于其计算顺序。这些都给基于浮点数实现的抽象域的可靠性保证带来了困难。
- 如何保证基于浮点数实现的抽象域的分析精度？程序分析追求的目标之一就是分析的精度。但是，由于浮点舍入误差的“无所不在”，每一条浮点指令都可能引入舍入误差。而且，舍入误差可能在复杂的、长序列的浮点计算中累计。这些都给基于浮点数实现的抽象域的分析精度的保证带来了困难。
- 如何保证使用了基于浮点数实现的抽象域所开展的程序分析的稳定性？基于抽象解释的程序分析一般通过“迭代”的方式来计算程序不动点。但是，浮点舍入误差可能给计算带来扰动，使得迭代计算难以产生有意义的、稳定的约束。

本文将应对这些挑战，开展相关研究。

(2) 数值抽象域的凸性问题

数值抽象域设计的关键点是选择适合用户需求的某个特定类别的数值约束系统来作为抽象域的域表示。该类约束系统的解集在几何意义上对应满足一些几何性质的某类图形区域。比如，多面体抽象域使用线性不等式系统作为其域表示，在几何上就对应多面体图形。

当前，大部分已有数值抽象域（包括区间域[38]、八边形域[41]、多面体域 [40]等）对应的几何图形区域都是凸的 [1]，因此只能表达凸的性质。但是，实际上，程序的行为在具体语义或聚集语义下一般都是非凸的。比如，程序中经常会用到 if-then-else 语句来进行分情况讨论。在数值程序中，人们会经常用到求最大值 max()、求最小值 min()、求绝对值 abs() 等非凸的函数来编写程序。另一方面，就程序分析而言，仍

[1]如果一个图形内任意两个点之间连线线段上的所有点恒在该图形内，则称该图形是凸的，否则称该图形是非凸的。

有许多用户所关心的程序的数值性质是非凸的。比如，要证明程序没有“除零错”需要证明形如 $x \neq 0$ 的性质。

当前数值抽象域的这种凸性限制往往会影响到程序分析的精度，导致出现误报。比如，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：使用区间抽象域中的区间来对分母表达式的值进行上近似，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 $\top$（Top，表示没有任何信息），并将影响到后续程序分析的精度。这里，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，需要设计非凸的数值抽象域以尽可能地把 0 排除在取值范围之外。当然，还有许多的程序性质需要使用非凸的数值抽象域来验证。

设计非凸的数值抽象域，为提高当前数值程序分析的分析精度提供了一种思路。但是，设计非凸的数值抽象域并非易事。主要困难在于：数值抽象域一般是使用约束系统，即有穷条约束的合取（conjunction），来作为抽象域的域表示；而非凸集合或性质一般需要通过析取（disjunction）来表达。简单地使用约束的析取作为抽象域的域表示，难以保证抽象域的封闭性（即域操作的结果仍在该抽象域中）和可计算性。而且，为了尽可能地保证程序分析的另一个目标——计算效率，所设计的非凸数值抽象域的相关算法的复杂度不能太高。

(3) 数值抽象域对区间系数的支持问题

数学上，区间是用来刻画现实世界中不精确性、不确定性等因素的有力工具。在实际软硬件系统的分析与验证中，在建模或抽象后，给定的应用数据（尤其是一些物理量）可能受不精确性或者不确定性因素的影响，使得仅知道这些数据在特定区间范围内。比如，有些数据可能是通过不精确的测量方法或者专家估计得到的。

特别地，就数值程序分析而言，对程序进行抽象的过程很可能会引入区间系数。首先，由于当前大部分数值抽象域都是面向线性约束的，为了能够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，往往需要使用一种区间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 $\sum_k [a_k, b_k]x_k + [c, d]$）[84, 85]。例如，设变量 x 的取值范围是 $[-1, 1]$，那么待分析程序中非线性表达式 $x * y$ 可能被区间线性化为带区间系数的线性表达式 $[-1, 1] \times y$。其次，当使用基于浮点数实现的数值抽象域（比如，本文第 3 章提出的浮点多面体抽象域）来分析

程序时，为了保证可靠性，待分析程序中的实数或有理数需要抽象成两个相邻浮点数所构成的区间。比如，$\frac{1}{10} * x$ 将被抽象成 $[0.99\ldots5,\ 0.10\ldots5] \times x$，因为 $\frac{1}{10}$（即小数 0.1）不是一个可被浮点格式精确表示的数。简而言之，在实际程序分析中，区间系数很自然地出现在抽象之后的待分析程序中。

但是，目前尚没有一种数值抽象域可以直接支持区间系数作为域表示约束中的变量系数。注意，区间抽象域只支持区间作为常量项在约束中出现（即形如 $x = [a, b]$）。为了使得已有数值抽象域能处理这种区间变量系数，当前的方法是通过“去区间化”操作把带区间系数的表达式（形如 $\sum_k [a_k, b_k]x_k + [c, d]$）抽象成线性表达式（形如 $\sum_k a'_k x_k + [c', d']$，即区间只出现在常量项）[85]。但是，这一抽象过程将导致大量精度损失（更为详细的讨论见本文第 3 章）。

因此，设计新的数值抽象域使其能够直接支持区间线性形式具有实用价值。这对于提高当前数值程序分析的分析精度也非常有意义。

1.2.2 研究目标

本文的研究目标在于：立足于数值抽象域的设计和实现，在保证可靠性的前提下，从分析精度和计算效率两方面来改进当前基于抽象解释的数值程序分析所存在的问题和局限性。主要关注点包括前一节所阐述的当前数值抽象域设计与实现所存在的三个方面的问题：抽象域实现的数据类型选择问题、抽象域表达能力的凸性限制问题、抽象域对区间系数的支持问题。具体而言，本文研究所面向的目标如下：

① 基于浮点数来实现数值抽象域：首先，考虑为已有的有代表性的数值抽象域（如多面体抽象域）设计可靠的浮点实现算法，从而能够为其他抽象域的可靠浮点实现提供经验性的借鉴；其次，设计一些新的、容易通过浮点数来可靠实现的数值抽象域，从而可以总结一些设计理念为新数值抽象域的设计和实现提供启发式的指导。本文还将对浮点实现的可靠性进行证明，并通过一些程序分析实验来展示浮点实现在计算效率方面的优势。

② 非凸数值抽象域的设计和实现：设计新的数值抽象域使其能够描述某些非凸性质。并研究这些非凸性质的数学根源、几何特征、以及在程序分析中的可能应用。此外，还将研究这些非凸数值抽象域与已有凸数值抽象域之间的关系（比如，在表达能力方面），从而为新抽象域与其他相关抽象域的结合和转换（以结合起来开展程序分析）提供依据。

③ 设计支持区间系数的数值抽象域：设计新的数值抽象域使其能够在域表示和域操作上直接支持区间系数。研究带区间系数约束的数学含义和几何性质，以及

在程序分析应用中的物理意义，从而为支持区间系数的数值抽象域的设计、实现和应用提供理论指导。研究带区间系数约束与传统标量系数约束的关系，从而进一步得到区间系数的数值抽象域与传统数值抽象域的关系。

本文研究的意义在于文中工作将进一步推动数值抽象域在实际程序分析与验证中的应用。

1.2.3 主要结果

本文主要工作及结果如下：

(1) 给出了经典多面体抽象域的一种可靠浮点实现方法（第 3 章）：
多面体抽象域是目前表达能力最强、应用最广泛的数值抽象域之一。但是，由于其高复杂度，多面体抽象域在许多实际应用中受到可扩展性和易处理性方面的限制。可以说，多面体抽象域的可扩展性仍是一个开放的问题。

- 本文采用多面体的约束表示（无需生成子表示），并基于严格线性规划技术和 Fourier-Motzkin 变量消除算法的一个浮点变种，给出了多面体抽象域的一种可靠浮点实现方法，以通过浮点实现来提高多面体抽象域的计算效率。
- 基于这样一个观察，即多面体抽象域的处理能力主要受限于其高代价的接合操作，本文提出了一系列低代价的弱接合操作作为接合操作的可靠替代候选，以进一步提高多面体抽象域的计算效率和可扩展性。

(2) 把区间线性代数引入到程序分析中，提出了区间多面体抽象域、行阶梯形区间线性等式抽象域等一系列支持区间系数的非凸数值抽象域（第 4-5 章）：
区间系数在实际程序分析中经常出现，并且区间（算术）为构造可靠的浮点抽象域提供了一种自然且有效的途径。

- 本文提出了区间多面体抽象域，用以推导区间线性不等式关系。该域相当于经典多面体抽象域的区间扩展版本，能够天然地表达某类拓扑非凸（甚至非连通）性质。并基于区间线性规划技术和 Fourier-Motzkin 变量消除算法的一个区间变种，采用浮点数可靠地实现了区间多面体域。
- 本文提出了行阶梯形区间线性等式抽象域，用以推导区间线性等式关系。该域相当于经典仿射等式抽象域的区间扩展版本，其表示方法基于区间线性等式的行阶梯形系统，也可以表示某类拓扑非凸（甚至非连通、非封闭）性质。并且，其域操作存在多项式时间的可靠浮点实现算法。
- 本文提出了单变量区间线性不等式抽象域，用以推导单个变量的取值范围。该域相当于经典区间抽象域的区间系数扩展版本。同样地，该域可以表示

某类拓扑非凸（甚至非连通）性质。其域操作可以通过简单高效的浮点区间算术来可靠构造，并具有线性的时空复杂度。

(3) 考虑了线性绝对值关系/广义线性互补关系等分段线性关系，提出了线性绝对值不等式抽象域与线性绝对值等式抽象域等非凸抽象域（第 6 章）：
绝对值作为数学中的一个基本概念，在数值程序和模型中经常出现，并且为分段线性和非凸性质的表达提供了有力途径。

- 本文给出并证明了线性绝对值不等式系统、广义线性互补问题（所对应的约束系统）、区间线性不等式系统三者之间的等价性，以及线性绝对值等式系统、水平线性互补问题（所对应的约束系统）两者之间的等价性。并指出了这些系统的分段线性描述能力。
- 本文对多面体的双重描述法进行了扩展，给出了广义线性互补问题（系统）的双重描述法。该双重描述法同样适用于与广义线性互补问题等价的约束系统。并作为其应用，给出了一种求解绝对值线性规划问题的新方法。
- 本文基于广义线性互补问题（系统）的双重描述法，设计了线性绝对值不等式抽象域（用来推导区间线性不等式/线性绝对值不等式/ 广义线性互补不等式关系）和线性绝对值等式抽象域（用来推导线性绝对值等式/ 水平线性互补等式关系）。并采用多精度有理数实现了这两个抽象域。线性绝对值不等式抽象域是经典多面体抽象域的一般化，而线性绝对值等式抽象域是经典仿射等式抽象域的一般化。并且这两个域都是非凸的且域操作都是具体域上对应操作的最佳抽象。

图 1.1 给出了本文主要工作在基于抽象解释的静态分析框架中的位置。图中可以看到，本文工作聚焦在数值抽象域的设计和实现上。为了对本文工作进行实验性评估，我们把文中所提出的数值抽象域在开源的数值抽象域库 Apron [86] 中进行了实现。并主要采用支持 Apron 库的开源静态分析工具 Interproc [87] 来开展程序分析实验。实际上，图 1.1 给出了本文实验平台（同时也是典型的基于抽象解释的静态分析器）的组成结构。

图 1.2 给出了本文提出的数值抽象域之间，以及与已有数值抽象域之间的关系。在图中，按抽象域的表达能力对这些数值抽象域进行了排序。其中，单实线表示位于单实线上方的抽象域的表达能力要强于位于下方的抽象域；双实线表示所连接的两个抽象域的表达能力（几乎）相同。实际上，线性绝对值不等式域的表达能力与区间多面体域相同，但是在域操作的实现算法上，前者的域操作都是具体域上对应操作的最佳抽象而后者则不是，因此前者要比后者更精确（即使两者都采用多精度

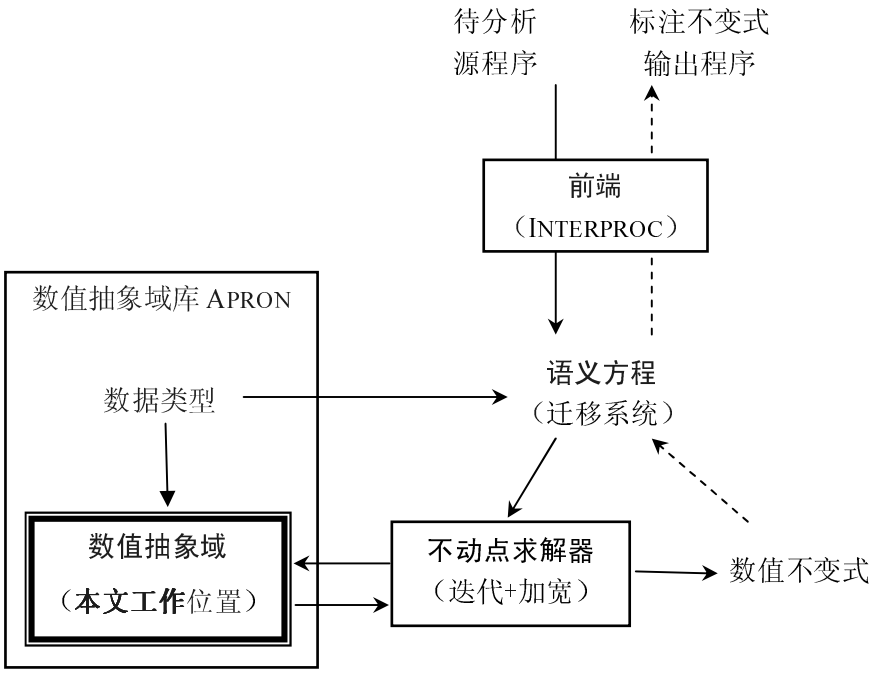


图 1.1 本文工作在基于抽象解释的静态分析框架中的位置

有理数实现）。多面体域的系数可以是任意多精度有理数，而浮点多面体域中的系数只能是有穷的浮点数，并且在域操作的实现算法上，前者（采用多精度有理数实现）也要比后者（采用浮点数实现）更精确。

1.3 相关研究工作

1.3.1 数值抽象域

三十多年来，抽象解释领域出现了许多表达能力、计算效率各有特色的数值抽象域（已有约近 30 种 [42, 44]），包括区间域 [38]、凸多面体域[40]、八边形域 [41] 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效率之间取得某种权衡。已有数值抽象域的分类、所能表示的数值性质及其复杂度，将在本文第 2.4.3节进行更为详细地描述。

在这些数值抽象域中，最具代表性的一个就是 Cousot 与 Halbwachs 在 1978 年提出的多面体抽象域 [40]。该域可用来推导程序中变量之间的线性关系，这也是大部分应用所关心的数值性质。因此，多面体抽象域在软硬件系统的分析与验证中有着很广泛的应用 [88]。但是，由于多面体抽象域本身固有的高复杂度，该域在许多应

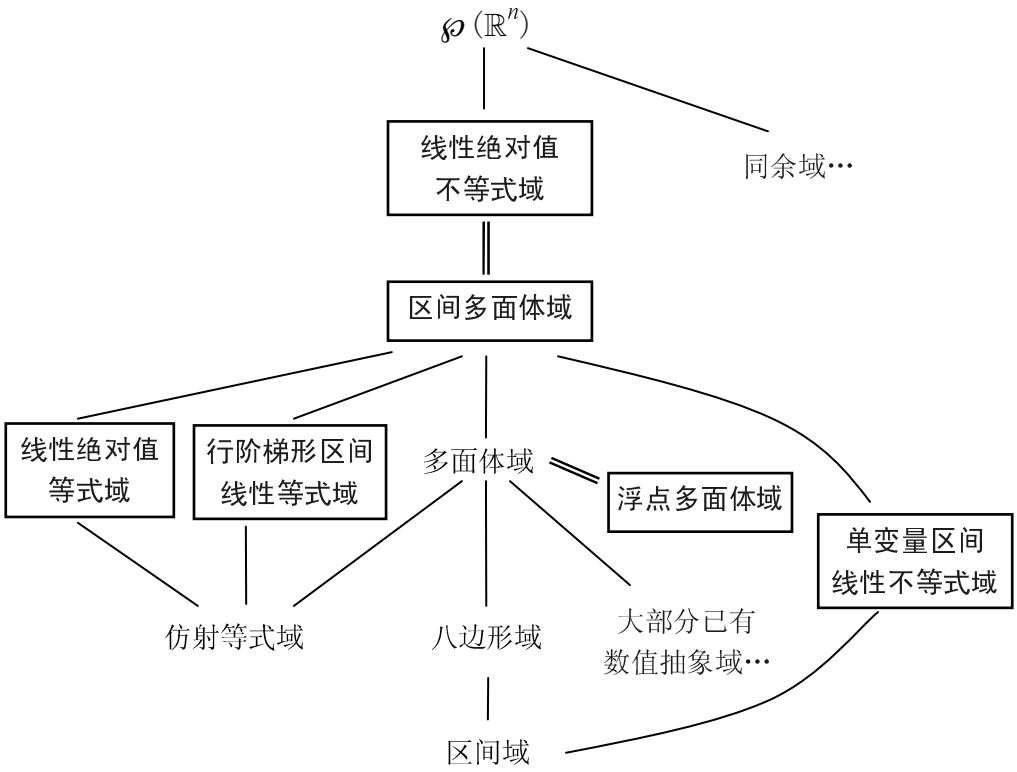


图 1.2 本文提出的数值抽象域与已有数值抽象域之间的关系

用中仍受到可扩展性和易处理性方面的限制 [83]。为了降低多面体域的复杂度，近 10 年来数值抽象域相关研究大部分都聚焦在研究表达能力比多面体域稍弱的弱关系型抽象域上，即某种受限形式的多面体，在这些域往往存在高效的算法来实现域操作。例如，八边形域 [41]、带域（Zones）[68]、八面体域 [89]、每不等式两变量域[90]、模版多面体域（又称模版约束矩阵域）[91]、符号范围约束域 [92]、五边形域 [93]、子多面体域 [94] 等（在图 1.2 中对应“大部分已有数值抽象域 ...”位置）。

本文的研究也与多面体域紧密相关。首先，在横向上，本文将从区别于上述研究的另外一个角度来开展工作以提高多面体域的计算效率和可扩展性，即采用浮点数来实现多面体域。其次，在纵向上，本文将研究多面体域的扩展和一般化，即表达能力比多面体域更强的抽象域，以满足某些特定的应用需求（比如，非凸表达能力、支持区间系数等），并研究扩展后抽象域的弱抽象域（通过降低表达能力来提高计算效率）。

关于数值抽象域的浮点实现问题，Miné 最早在文献 [84] 中系统地考虑了该问题，并给出了区间抽象域和八边形抽象域的可靠浮点实现方法。Feret 设计了面向特定应用的数字滤波器抽象域 [95]、等差等比数列抽象域 [96]，这些抽象域是专门面

向特定的浮点程序行为设计的，因此比较容易采用浮点数来可靠实现。这些已知存在浮点可靠实现的数值抽象域在静态分析工具 Astrée [21, 22] 中得以实现，并在工业界大规模的程序分析中得到成功应用 [27, 28]。

1.3.2 非凸静态分析

当前，大部分数值抽象域只能表示凸的性质，这种凸性限制对于某些应用可能导致分析精度不够而出现误报。为了处理析取（描述非凸性质的常用逻辑操作）以获得某种程度的路径敏感的分析，Cousot 与 Cousot 在抽象解释框架下提供了析取完全化（Disjunctive Completion）技术 [80, 97, 98]（析取完全化只是理论上的一个构造性定义，实际中，可以理解为抽象状态的无冗余集合并）。新的抽象域可以通过使用标准的方法（如幂集扩展）在基抽象域上系统地构造出来。但是，该技术的计算代价很高：即使是应用在势为 n 的有穷基域上，导出的幂集抽象域中将会有 2^n 个域元素且对应格的高度为 $n+1$。令问题更为棘手的是，对于无穷基域而言，如果没有一个合适的加宽算子，基于幂集的抽象域可能导致“无界问题”，而通过基域上的加宽算子来构造幂集域上的加宽算子往往很难 [99]：好的加宽算子应合理地决定对哪些域元素需要合并，对哪些域元素需要加宽。此外，不难看出，幂集抽象域需要耗费大量的存储空间（尤其对于关系型抽象域而言），而且域操作的计算复杂度也会很高（往往是基域操作复杂度的指数倍）。

因此，许多研究通过对程序的控制结构进行精化，以延迟接合操作（抽象域中使用该操作来对逻辑析取进行抽象），来改进析取静态分析。Rival 与 Mauborgne[100] 提出了迹划分抽象域：根据控制流历史来对程序的迹语义进行抽象，把迹集合划分成一些束（作为某种受限形式的析取），然后对每个束进行单独处理。迹划分抽象域在静态分析工具 Astrée 中得以实现，并在工业界大规模的程序分析中取得成功应用。Sankaranarayanan 等人 [101] 采用一种 on-the-fly 的方式来构造程序控制流图的精化版本，使得在精化后的控制流图上采用基域分析就可以得到在原控制流图上采用幂集域分析得到的不动点。另外，Jeannet [102] 把布尔分析与多面体分析结合起来，对抽象状态进行动态划分，以对控制流进行精化。Simon [103] 通过引入一个布尔变量，把两个 n 维多面体的集合并通过一个 $n+1$ 维的多面体来精确表达。但是，上述技术往往需要一些启发式或者用户标注的方式来指导静态分析何时对抽象状态进行划分、何时对抽象状态进行接合 [101, 100]。

本文关注如何在抽象域层面设计非凸抽象域来支持某类析取信息的表达，这样就可以适合经典的抽象解释框架而不需要额外对程序控制结构进行精化。但是，目

前只存在很少的抽象域能够天然地表示非凸集合，诸如同余域 [104]、max-plus 多面体域 [105] 等。最近，Gulavani 与 Gulwani [50] 通过引入基于 max 表达式的域提升操作来支持某类非凸信息，并应用在时间界分析上。但是，该提升操作也是基于启发式的，因此分析结果可能会不够精确。本文将设计新的数值抽象域以支持某些程序分析中常见的非凸信息，比如绝对值、max/min 函数等。

1.3.3 浮点计算的分析与验证

由于舍入误差的存在，浮点计算不是精确的。但是，在许多应用中，人们仍然希望浮点计算的结果与精确计算的结果满足一定的关系，比如相差在一定的范围内等。甚至，一些应用（比如定理证明）要求浮点计算结果等于精确计算结果，在此方向上，Zhang（张景中）与 Feng（冯勇）[106] 考虑了使用近似的浮点计算来获得精确计算结果的理论、方法等问题。近年来，在数学规划领域，严格的（rigorous）也称验证的（verified）数学规划技术得到了关注并取得了一些突破性进展，出现了一些新的计算代价不高的技术来分析浮点数学规划所得目标值与精确数学规划所得结果之间的严格错误界 [107, 108, 109, 110]。比如，Neumaier 与 Shcherbina [107] 给出了通过在对偶问题上应用浮点线性规划来求解原问题上目标函数的严格界的方法，从而使得仅使用浮点计算就可以求得线性规划问题目标值的可靠近似（不小于精确的极大目标值或不大于精确的极小目标值）。根据模型检验与程序分析的当前需求，Monniaux[111] 最近建议在可满足性模理论（Satisfiability Modulo Theories，SMT）算法中使用浮点线性规划来辅助精确的线性算术决策过程，以提高计算效率，同时保证决策结果依然是可靠且完全的。

另一方面，在形式化方法领域，由于浮点算术的复杂性及其具有的一些使得形式化验证变得很困难的缺陷 [112, 113]，关于浮点计算是否满足规约的问题被搁置了很长一个时期。1994年，Intel 的奔腾芯片在浮点除法运算（FDIV）指令的设计上存在缺陷 [114]，导致 Intel 面临“浮点运算危机”，最终造成近 4.75 亿美元的经济损失。由此，浮点计算的验证问题最早在硬件浮点功能部件的验证方面得到了许多硬件公司（包括 Intel、AMD、IBM 等）的广泛关注。例如，Harrison 使用高阶定理证明器 HOL Light，对 Intel 硬件设计中的浮点算术和一些浮点算法（如除法、求平方根、超越函数 sin/cos 等）进行了形式化验证 [115, 116]。近年来，受来自航空航天等安全攸关软件应用需求的推动，许多研究开始关注针对浮点程序的分析与验证。静态分析工具 Astrée [21, 22] 能够分析浮点程序是否会出现运行时错误，比如浮点上溢、除零错等。静态分析工具 Fluctuat [117, 24, 25] 用来分析浮点程序中舍入误差的传播

情况，从而能够发现舍入误差的阶及其源头。最近，程序验证工具 Caduceus[118, 119] 通过把支持浮点的自动证明器 Gappa 与证明辅助器 Coq 集成起来，用来证明一些浮点程序满足某些规约，以认证浮点程序的正确性。

本文关注使用浮点计算来可靠地实现数值抽象域，以提高计算效率。上述严格的（或验证的）浮点计算技术，尤其是严格线性规划技术，为本文的研究尤其是浮点实现的可靠性保证提供了技术支撑。而针对浮点程序的分析与验证则为本文的研究提供了动机和应用背景，因为使用基于多精度有理数的实现来分析浮点程序时往往会出现“大系数”问题，而使用基于浮点数的实现来分析浮点程序则是再自然不过的选择。

1.3.4 区间线性代数

区间分析 [120]又称区间数学、区间计算，是一门用区间变量代替点变量进行运算的数学分支。它最初是在上世纪 50、60年代从计算数学的误差理论研究发展起来的 [121]，并在全局优化、计算机辅助证明、控制理论等领域取得应用 [122, 123, 124]。将区间分析与线性代数相结合就形成了区间线性代数，研究对象包括区间向量、区间矩阵、区间线性系统等 [125, 126]。

求解区间线性系统一直是区间线性代数领域的一个挑战性问题。这一问题最先是在上世纪 60 年代中期由 Oettli 和 Prager 提出并开始研究的 [127]。从那时起，该问题便得到了持续关注 [128, 129]。在此过程中，出现了多种不同的对区间线性系统的解的语义解释，包括弱解、强解、容限解、控制解、代数解等 [129]。不同的语义可以表达不同的性质，面向不同的应用。Rohn 对区间线性系统关于不同解语义下系统的解存在条件、求解方法及其复杂度进行了系统的研究 [128, 129]。本文只关心区间线性系统的弱解集合。然而, 检查区间线性系统是否存在弱解是 NP-hard 问题 [129]。如果存在弱解，计算区间线性系统弱解集合的包围盒（bounding box，也称为区间包 Interval hull）也是 NP-hard 问题 [129]。Jansson 最先对对区间线性系统之弱解集合的几何拓扑性质进行了较系统的研究 [130]。

此外，区间线性规划问题，即带区间系数的线性规划问题，得到了来自区间分析领域和数学规划领域研究人员的广泛关注。文献 [131] 根据语法表示把区间线性规划问题分成几个类型，然后为不同的类型设计不同的求解算法。Jansson [108] 提出了一种迭代方法，通过求解一系列中点线性规划问题，来计算区间线性规划问题目标函数的可靠上下界，并且在许多情况下，该方法所需计算代价并不高。

与上述研究不同的是，本文关注抽象域的设计。本文将利用区间线性代数领域

的一些结果来设计支持区间系数的数值抽象域。因此，本文主要聚焦在如何根据程序的语义来设计域操作用以对区间线性约束进行操作。

1.4 论文结构

本文论文题目为“基于区间线性抽象域的可靠浮点及非凸静态分析”。其中，“静态分析”是本文研究的领域，“抽象域”是本文研究的关注点，“可靠浮点”和“非凸”是本文研究的目标和特色，“区间线性”是本文研究的主要技术途径。

本文共分为 7 章，论文的结构如图 1.3 所示。

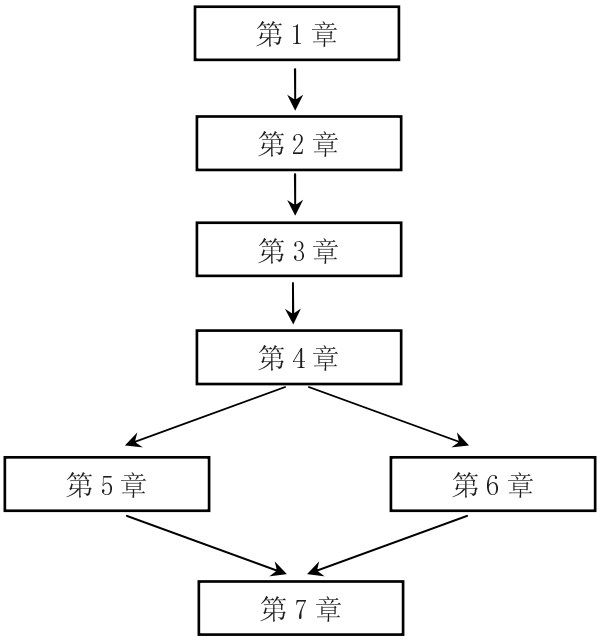


图 1.3 论文结构图

各个章节内容具体安排如下。

- 第 1 章是绪论，首先介绍了论文的研究背景，本文的研究动机、目标和主要结果，然后介绍并总结了相关的研究工作，最后介绍了论文的结构。
- 第 2 章介绍本文研究的理论基础和框架，即抽象解释理论及其在数值程序分析中的应用框架。综述已有数值抽象域的相关研究，并讨论与浮点计算相关的程序分析技术。
- 第 3 章给出多面体抽象域的一种可靠浮点实现方法，即浮点多面体抽象域；提出一系列低代价的弱接合操作以提高多面体抽象域的可扩展性，并给出这些弱

接合操作的启发式结合方法。

- 第 4 章提出一种新的数值抽象域，即区间多面体抽象域，用来推导程序变量间的区间线性不等式关系；还提出单变量区间线性不等式抽象域，用来推导单个变量上的区间线性不等式关系；并给出这些抽象域的可靠浮点实现方法。
- 第 5 章提出一种新的数值抽象域，即行阶梯形区间线性等式抽象域，用来推导程序变量间的区间线性等式关系；并给出该抽象域的一种多项式时间的可靠浮点实现方法。
- 第 6 章提出两种新的数值抽象域，即线性绝对值不等式抽象域和线性绝对值等式抽象域，用来推导程序变量间的线性绝对值关系；并给出两个抽象域的基于多精度有理数的实现方法。
- 第 7 章总结本文的主要贡献，并对下一步工作给予展望。

总体上，本文工作可分为三个部分：浮点多面体抽象域部分（第 3 章）、区间线性抽象域部分（第 4–5 章）、线性绝对值抽象域部分（第 6 章）。

具体而言，各个章节之间的关系如下：第 2 章中介绍的抽象解释理论及其提供的数值程序分析框架是第 3–6 章的理论基础。第 3 章所介绍的浮点多面体抽象域及其相关技术为第 4–6 章提供了技术准备，且第 3 章所观察到的一些问题为第 4–6 章的研究提供了动机和依据。第 4 章所介绍的区间多面体抽象域是对第 3 章浮点多面体抽象域的扩展和一般化。第 5 章所介绍的行阶梯形区间线性等式抽象域，通过限制表达能力来改进第 4 章区间多面体抽象域的计算效率。第 6 章所介绍的线性绝对值不等式抽象域是第 4 章区间多面体抽象域的强实现，两者表达能力一样，但是线性绝对值不等式抽象域的域操作都是具体域上操作的最佳抽象，而区间多面体域中的许多域操作（尤其接合操作）都是弱抽象（即不是最佳抽象）。除了第 6 章的抽象域是基于多精度有理数来实现的外，第 3–5 章的抽象域都是基于浮点数来实现的以提高计算效率（当然，这些抽象域也可以基于多精度有理数来实现）。第 4–6 章的抽象域都可以直接支持区间系数，且可以描述某种非凸甚至非连通性质。其中，第 5 章的抽象域还可以支持无穷区间作为变量系数，并可以描述某种非封闭性质。

第二章　基于抽象解释的数值程序分析

2.1　引言

抽象解释由 P. Cousot 和 R. Cousot 于 1977 年提出，是一种对程序语义进行可靠近似（或抽象）的通用理论 [20]。该理论在静态分析中的一个重要应用就是数值程序分析。本章主要介绍文中使用的抽象解释及数值程序分析相关的理论和技术（主要参考了文献 [132, 42]）。本章内容组织如下：

- 2.2 节介绍本章用到的与程序语义相关的基础概念，诸如偏序、格、不动点等。
- 2.3 节介绍抽象解释理论及其为静态程序分析所提供的通用框架。
- 2.4 节介绍基于抽象解释的数值程序分析框架，着重介绍该框架下的核心要素——数值抽象域，并概述已有数值抽象域。
- 2.5 节介绍浮点数的表示方法及其计算模型；讨论浮点计算给数值程序分析所带来的困难，并介绍一种面向浮点程序的分析方法；讨论基于浮点数来实现数值抽象域的意义。

2.2　预备知识

本节主要介绍偏序、格、不动点等基本数学概念及其符号表示。

2.2.1　偏序、格

定义 2.2.1 (偏序) *若集合 D 上的二元关系 $\sqsubseteq$ 是自反的、传递的和反对称的，则称 $\sqsubseteq$ 是 D 上的*偏序关系*，称 $(D, \sqsubseteq)$ 为*偏序集*。*

本文中，我们有时把偏序关系理解为精度序（即元素所含信息的精确程度）：对于域元素而言，$x \sqsubseteq y$ 说明域元素 x 比域元素 y 更精确；对于性质而言，$x \sqsubseteq y$ 说明性质 x 比性质 y 更强。

定义 2.2.2 (格) *偏序集 $(D, \sqsubseteq)$ 称为*格*，如果 D 中任何两个元素 a, b 都有最小上界（即上确界，记作 $a \sqcup b$）和最大下界（即下确界，记作 $a \sqcap b$）。该格记作 $(D, \sqsubseteq, \sqcup, \sqcap)$。*

定义 2.2.3 (完全格) *偏序集 $(D, \sqsubseteq)$ 称为*完全格*，如果 D 的所有子集 X 都有最小上界 $\bigsqcup X$ 和最大下界 $\bigsqcap X$。特别地，D 总存在最小元 $\bot = \bigsqcup \emptyset$ 和最大元 $\top = \bigsqcup D$。该完全格记作 $(D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$。*

设 $(D,\sqsubseteq)$ 为偏序集，称 D 的非空子集 S 为**链**，如果 S 中任意两个元素关于 $\sqsubseteq$ 皆可比。

定义 2.2.4 (完全偏序) 设 $(D,\sqsubseteq)$ 为偏序集，若

1. D 有最小元，记为 $\bot_D$;
2. 对 D 中每个链 S，其最小上界 $\sqcup S$ 存在。

则称 $(D,\sqsubseteq)$ 为**完全偏序**（CPO）。该 CPO 记作 $(D,\sqsubseteq,\sqcup,\bot)$。

2.2.2 不动点

设 f 是偏序集 $(D,\sqsubseteq)$ 上的函数，则 D 中元素 x 称为函数 f 的

- **不动点**，如果 $f(x)=x$；
- **前不动点**，如果 $x\sqsubseteq f(x)$；
- **后不动点**，如果 $f(x)\sqsubseteq x$。

记函数 f 的所有不动点构成的集合为 $\mathsf{fp}f=\{x\in D: f(x)=x\}$，所有前不动点构成的集合为 $\mathsf{prefp}f=\{x\in D: x\sqsubseteq f(x)\}$，所有后不动点构成的集合为 $\mathsf{postfp}f=\{x\in D: f(x)\sqsubseteq x\}$。不动点 $x\in\mathsf{fp}f$ 称为函数 f 的**最小不动点**，记为 $\mathsf{lfp}f$，如果对于任意 $y\in\mathsf{fp}f$ 皆有 $x\sqsubseteq y$；不动点 $x\in\mathsf{fp}f$ 称为函数 f 的**最大不动点**，记为 $\mathsf{gfp}f$，如果对于任意 $y\in\mathsf{fp}f$ 皆有 $y\sqsubseteq x$。记 $\mathsf{lfp}_d f$ 为关于偏序 $\sqsubseteq$ 比 d 大的最小不动点（若存在），$\mathsf{gfp}_d f$ 为关于偏序 $\sqsubseteq$ 比 d 小的最大不动点（若存在）。

偏序集 $(D,\sqsubseteq)$ 和 $(D',\sqsubseteq')$ 上的函数 $f:D\to D'$ 是**单调的**，如果只要 $x\sqsubseteq y$ 则 $f(x)\sqsubseteq' f(y)$。

设 $(D,\sqsubseteq)$和 $(D',\sqsubseteq')$ 是 CPO，函数 $f: D\to D'$。若对于 D 中的每个链 S，$\sqcup f(S)$ 存在且有 $f(\sqcup S)=\sqcup f(S)$，则称 f 是 $\sqcup$-**连续**的。对偶地，若对于 D 中的每个链 S，$\sqcap f(S)$ 存在且有 $f(\sqcap S)=\sqcap f(S)$，则称 f 是 $\sqcap$-**连续**的。

f 是 $\sqcup$-连续的，当且仅当 f 是单调的且对于 D 中每个链 S，有 $f(\sqcup S)\sqsubseteq\sqcup f(S)$。特别地，若 D 只含有穷链，则 f 是 $\sqcup$-连续的当且仅当 f 是单调的。对偶地，$\sqcap$-连续函数与单调函数之间也有类似关系。

下面给出序结构上的函数的不动点的两个基本定理。

定理 2.1 (Tarski 不动点 [133]) 完全格 $(D,\sqsubseteq,\sqcup,\sqcap,\bot,\top)$ 上单调函数 $f:D\to D$ 的不动点集合构成一个非空完全格，且

$$\mathsf{lfp}f=\sqcap\,\mathsf{postfp}f=\sqcap\{x\in D: f(x)\sqsubseteq x\}$$
$$\mathsf{gfp}f=\sqcup\,\mathsf{prefp}f\ =\sqcup\{x\in D: f(x)\sqsupseteq x\}$$

上述 Tarski 定理保证了完全格上任意单调函数的最小及最大不动点的存在性。同时，也说明了最小及最大不动点是唯一的。但是，上述定理没有显式地说明如何计算这些不动点。对于连续函数，下面的定理构造性地给出了最小不动点的计算方法[134]。

定理 2.2 (Kleene 不动点[135]) 设 $(D, \sqsubseteq)$ 为 *CPO*。

- 若函数 $f : D \to D$ 是 $\sqcup$-连续的，则有

$$\mathsf{lfp} f = \sqcup\{f^i(\bot) \mid i \in \mathbb{N}\}$$

- 若函数 $f : D \to D$ 是 $\sqcap$-连续的，则有

$$\mathsf{gfp} f = \sqcap\{f^i(\top) \mid i \in \mathbb{N}\}$$

上述 Kleene 定理通过把函数 f 迭代地应用到最小元 $\bot$ 来构造最小不动点，并有 $f^0(\bot) \sqsubseteq f^1(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \cdots$。因为完全格是 CPO，所以 Kleene 定理可以应用在完全格上。由于有穷格上的任意单调函数都是 $\sqcup$-连续的且 $\sqcap$-连续的，因此有穷完全格上我们能够高效地计算出任意单调函数的最小及最大不动点：使得 $f^i(\bot) = f^{i+1}(\bot)$ 成立的最小 i 值 i_0 所对应的 $f^{i_0}(\bot)$ 即为最小不动点，即 $\mathsf{lfp} f = f^{i_0}(\bot)$。

下面给出 Kleene 不动点定理的两个变种，来说明 Kleene 迭代其实可以从任何一个前不动点（后不动点）开始，以计算最小不动点（最大不动点），而不一定需要从最小值 $\bot$（最大值 $\top$）开始，如图 2.1 所示。据此，如果已知最小不动点（最大不动点）一个可靠的近似 d，我们可以从 d 开始迭代以加速不动点计算。

定理 2.3 (Kleene 不动点的变种1) 设 $(D, \sqsubseteq)$ 为*CPO*，且 $d \in D$。

- 若函数 $f : D \to D$ 是 $\sqcup$-连续的且 d 是 f 的前不动点，则有

$$\mathsf{lfp}_d f = \sqcup\{f^i(d) \mid i \in \mathbb{N}\}$$

- 若函数 $f : D \to D$ 是 $\sqcap$-连续的且 d 是 f 的后不动点，则有

$$\mathsf{gfp}_d f = \sqcap\{f^i(d) \mid i \in \mathbb{N}\}$$

定理 2.4 (Kleene 不动点的变种2) 设 $(D, \sqsubseteq)$ 为*CPO*，且 $a \in D$。

- 若函数 $f : D \to D$ 是 $\sqcup$-连续的且 $a \sqsubseteq \mathsf{lfp} f$，则有

$$\mathsf{lfp} f = \mathsf{lfp}_\bot f = \sqcup\{f^i(a) \mid i \in \mathbb{N}\}$$

- 若函数 $f : D \to D$ 是 $\sqcap$-连续的且 $a \sqsupseteq \mathsf{gfp} f$，则有

$$\mathsf{gfp} f = \mathsf{gfp}_\top f = \sqcap\{f^i(a) \mid i \in \mathbb{N}\}$$

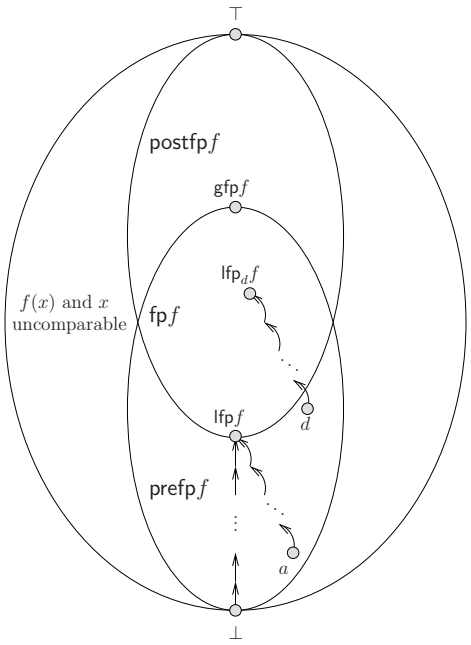


图 2.1 不动点及 Kleene 不动点迭代

2.3 抽象解释理论

抽象解释理论的一个重要思想是通过偏序结构上单调函数的不动点来表达程序的语义，并把各种语义都在抽象解释统一框架下进行形式化，从而可以很方便地对这些语义进行比较。并且，抽象解释还允许通过对已有语义进行抽象来构造新的语义，并保证了这些通过抽象所构造出来的语义都是可靠的。此外，抽象解释还允许把多个抽象组合起来进行应用。

2.3.1 基于 Galois 连接的抽象解释

“Galois 连接”最初是法国著名数学家伽罗瓦（Évariste Galois）用在群论里来描述两个群之间的关系。抽象解释以此来刻画具体域与抽象域之间的关系。

(1) 域元素抽象

定义 2.3.1 (Galois 连接) 给定偏序集 $(D, \sqsubseteq)$ 和 $(D^\sharp, \sqsubseteq^\sharp)$，函数 $\alpha : D \to D^\sharp$ 及 $\gamma : D^\sharp \to D$ 构成的函数对 (α, γ) 称为 D 与 $D^\sharp$ 之间的***Galois* 连接**，当且仅当

$$\forall x \in D, x^\sharp \in D^\sharp, \alpha(x) \sqsubseteq^\sharp x^\sharp \Longleftrightarrow x \sqsubseteq \gamma(x^\sharp)$$

并记作

$$(D, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (D^\sharp, \sqsubseteq^\sharp) \tag{2.1}$$

其中，$(D, \sqsubseteq)$ 称为**具体域**，$(D^\sharp, \sqsubseteq^\sharp)$ 称为**抽象域**，α 称为**抽象化函数**，γ 称为**具体化函数**。性质 $\alpha(x) \sqsubseteq^\sharp x^\sharp$ （亦即 $x \sqsubseteq \gamma(x^\sharp)$）形式化地描述了如下事实：$x^\sharp$ 是 x 的可靠抽象（或可靠近似）。而且，$\alpha(x)$ 是 x 在 $D^\sharp$ 中的最佳抽象（即最精确的可靠近似），而 $\gamma(x^\sharp)$ 则是在 D 中能被 $x^\sharp$ 可靠近似的最不精确的元素。

定义 2.3.2 (Galois 连接的等价定义) 给定偏序集 $(D, \sqsubseteq)$ 和 $(D^\sharp, \sqsubseteq^\sharp)$，函数 $\alpha : D \to D^\sharp$ 及 $\gamma : D^\sharp \to D$ 构成的函数对 (α, γ) 称为 D 与 $D^\sharp$ 之间的 ***Galois* 连接**，当且仅当

1. α, γ 是单调的，
2. $(\alpha \circ \gamma)(x^\sharp) \sqsubseteq^\sharp x^\sharp$ 且 $x \sqsubseteq (\gamma \circ \alpha)(x)$。

直观上，α 单调说明具体域 D 中更精确的元素的最佳抽象在抽象域 $D^\sharp$ 上也是更精确的。$x \sqsubseteq (\gamma \circ \alpha)(x)$ 说明对具体域中元素 x 先抽象化然后具体化所得的结果不会比 x 更精确（反而可能导致精度损失）。正因如此，在 Galois 连接的图示表示（2.1）中，γ 对应箭头总位于 α 对应箭头的上方。

如果具体域和抽象域都是完全格，则抽象化函数 α 和具体化函数 γ 彼此都可以通过另一个函数来构造。

定理 2.5 (α, γ 的规范型) 设 $(D, \sqsubseteq, \sqcup, \sqcap, \bot, \top), (D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp)$ 为完全格，(α, γ) 为 D 与 $D^\sharp$ 之间的 *Galois* 连接，则有

- $\alpha(x) = \sqcap^\sharp\{x^\sharp \in D^\sharp \mid x \sqsubseteq \gamma(x^\sharp)\}$，
- $\gamma(x^\sharp) = \sqcup\{x \in D \mid \alpha(x) \sqsubseteq^\sharp x^\sharp)\}$。

(2) 函数抽象

给定 Galois 连接 $(D, \sqsubseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (D^\sharp, \sqsubseteq^\sharp)$，设 f 是具体域 D 上的函数，$f^\sharp$ 是抽象域 $D^\sharp$ 上的函数。则 $f^\sharp$ 是 f 的**可靠抽象**，当且仅当 $\forall x^\sharp, (\alpha \circ f \circ \gamma)(x^\sharp) \sqsubseteq^\sharp f^\sharp(x^\sharp)$，即 $\forall x^\sharp, (f \circ \gamma)(x^\sharp) \sqsubseteq (\gamma \circ f^\sharp)(x^\sharp)$。$f^\sharp$ 是 f 的**最佳抽象**，当且仅当 $f^\sharp = \alpha \circ f \circ \gamma$。$f^\sharp$ 是 f 的**精确抽象**，当且仅当 $\gamma \circ f^\sharp = f \circ \gamma$。因为存在 Galois 连接，$f$ 的最佳抽象总存在，但是其精确抽象不一定存在。精确抽象的存在意味着 f 在具体域上的操作结果可以通过抽象域上的操作精确地刻画出来。如果函数 f 的精确抽象存在且 γ 是单射，那么该精确抽象唯一并且是最佳抽象。

Galois 连接的存在，实际上描述了一个抽象域可以获得的最大精度（通过为域元素和函数都选择其最佳抽象）。当然，选择域元素或函数的非最佳抽象则会导致更多的精度损失。

2.3.2 基于具体化函数的抽象解释

(1) 域元素抽象

对于某些实际应用，要求存在 Galois 连接可能太苛刻了。比如，当用有理数区间来对实数集合 $\{x \mid x \leq \sqrt{2}\}$ 进行抽象，或用多面体来对圆盘 $\{(x,y) \mid x^2+y^2 \leq 1\}$ 进行抽象时，都将不存在最佳抽象，即不存在良定义的 α，因此也就不存在 Galois 连接。为此，抽象解释理论提供了只基于 α 或 γ 的抽象解释框架。在实际应用中，基于 γ 的抽象解释应用更为广泛，而且本文工作也是在这一框架下开展的，所以本文只关注基于 γ 的抽象解释框架。下面，我们只使用具体化函数 γ 来描述具体域与抽象域之间的关系。

给定偏序集 $(D,\sqsubseteq)$ 和 $(D^\sharp,\sqsubseteq^\sharp)$，设具体化函数 $\gamma : D^\sharp \to D$ 是单调的。则称 $x^\sharp \in D^\sharp$ 是 $x \in D$ 的**抽象**，如果 $x \sqsubseteq \gamma(x^\sharp)$。此时，因为不存在抽象函数 α，我们无法为 D 中元素 x 定义其最佳抽象。

(2) 函数抽象

给定偏序集 $(D,\sqsubseteq)$、$(D^\sharp,\sqsubseteq^\sharp)$ 及单调的具体化函数 $\gamma : D^\sharp \to D$，设 f 是具体域 D 上的函数， $f^\sharp$ 是抽象域 $D^\sharp$ 上的函数。则 $f^\sharp$ 是 f 的**可靠抽象**，当且仅当 $\forall x^\sharp, (f \circ \gamma)(x^\sharp) \sqsubseteq (\gamma \circ f^\sharp)(x^\sharp)$。$f^\sharp$ 是 f 的**精确抽象**，当且仅当 $\gamma \circ f^\sharp = f \circ \gamma$。同样地，因为不存在抽象化函数 α，我们无法为函数 f 定义其最佳抽象。

特别地，当具体域 D 的任意子集都有最大下界时，我们称 $f^\sharp$ 是 f 的**最优抽象**当且仅当 $\forall x^\sharp, (\gamma \circ f^\sharp)(x^\sharp) = \bigsqcap\{\gamma(y^\sharp) \mid (f \circ \gamma)(x^\sharp) \sqsubseteq \gamma(y^\sharp)\}$。但是，一般情形下，函数 f 的最优抽象不一定存在，即使存在也不一定唯一。与之不同的是，在 Galois 连接存在的情形下（见第 2.3.1 节），函数 f 的最优抽象总存在且唯一，即最佳抽象 $\alpha \circ f \circ \gamma$。

2.3.3 抽象不动点

设 f 为具体域上的单调函数，$f^\sharp$ 为函数 f 在抽象域上的可靠抽象。下面，我们给出 f 在具体域上的最小不动点与 $f^\sharp$ 在抽象域上的最小不动点之间的关系。

定理 2.6 *给定完全格 $(D,\sqsubseteq,\sqcup,\sqcap,\bot,\top)$ 和 $(D^\sharp,\sqsubseteq^\sharp,\sqcup^\sharp,\sqcap^\sharp,\bot^\sharp,\top^\sharp)$，设 (α,γ) 是两者之间的 Galois 连接，f 与 $f^\sharp$ 分别是 D 与 $D^\sharp$ 上的单调函数。那么，若 $f^\sharp$ 是 f 的可靠抽象，则有 $\mathsf{lfp}_\bot f \sqsubseteq \gamma(\mathsf{lfp}_{\bot^\sharp} f^\sharp)$，亦即 $\alpha(\mathsf{lfp}_\bot f) \sqsubseteq^\sharp \mathsf{lfp}_{\bot^\sharp} f^\sharp$。*

上述定理是抽象解释理论中的一个重要结果，说明了在抽象域上所计算得到的最小不动点是可靠的（即，是具体域上最小不动点的上近似），进而保证了在抽象

域上开展程序分析所得结果的可靠性。上述定理类似于 Tarski 不动点定理（定理 2.1），要求具体域和抽象域是完全格，并且两个域之间存在 Galois 连接。下面，我们给出一个类似于 Kleene 不动点定理（定理 2.2）的结果，只要求具体域和抽象域是 CPO，且适用于只基于具体化函数的抽象解释框架。

定理 2.7 给定 $CPO\ (D, \sqsubseteq, \sqcup, \bot)$ 和 $(D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \bot^\sharp)$，设具体化函数 $\gamma : D^\sharp \to D$ 是单调的，f 与 $f^\sharp$ 分别是 D 与 $D^\sharp$ 上的单调函数。那么，若 $f^\sharp$ 是 f 的可靠抽象且 $\gamma(\bot^\sharp)$ 是 f 的前不动点，则有 $\mathsf{lfp}_\bot f \sqsubseteq \mathsf{lfp}_{\gamma(\bot^\sharp)} f \sqsubseteq \gamma(\mathsf{lfp}_{\bot^\sharp} f^\sharp)$。

定理 2.8 给定 $CPO\ (D, \sqsubseteq, \sqcup, \bot)$ 和 $(D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \bot^\sharp)$，设具体化函数 $\gamma : D^\sharp \to D$ 是单调的，f 与 $f^\sharp$ 分别是 D 与 $D^\sharp$ 上的单调函数。那么，若 $f^\sharp$ 是 f 的可靠抽象且 $\gamma(x^\sharp)$ 与 $x^\sharp$ 分别是 f 与 $f^\sharp$ 的前不动点，则有 $\mathsf{lfp}_{\gamma(x^\sharp)} f \sqsubseteq \gamma(\mathsf{lfp}_{x^\sharp} f^\sharp)$。

(1) 抽象最小不动点计算

假设我们已经设计了一个抽象域，其域元素是计算机可表示的，其域操作是可计算的。接下来，我们需要考虑如何在抽象域上有效地计算不动点。

如果抽象域中没有无穷递增链，那么 Kleene 不动点定理（定理 2.2）给我们提供了一种构造性的方法来计算抽象最小不动点。

定理 2.9 (没有无穷递增链的 Kleene 迭代) 若 $CPO\ (D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \bot^\sharp)$ 上没有无穷递增链，$f^\sharp$ 是 $D^\sharp$ 上的单调函数，且 $x_0^\sharp$ 是 $f^\sharp$ 的前不动点，那么 *Kleene* 迭代序列 $x_{i+1}^\sharp = f^\sharp(x_i^\sharp)$ 将会在有穷时间内收敛于 $\mathsf{lfp}_{x_0^\sharp} f^\sharp$。

对于有穷抽象域（即域元素个数是有穷的）如符号域 [38]，以及满足递增链条件的无穷抽象域（即抽象域对应格的高度是有穷的）如常量域 [37] 及仿射等式抽象域[39]，上述定理提供了迭代方法来精确地计算抽象最小不动点，并且保证了迭代计算的终止性。

对于一些包含无穷递增链的域而言，通常的 Kleene 迭代方法难以保证迭代的终止性，此时难以精确地计算出抽象最小不动点。对于某些特殊情形，仍存在一些巧妙的迭代方法来保证不动点计算的终止性。比如，区间抽象域上存在无穷递增链，但是对于某类区间约束以及受限的迁移函数，文 [136, 137] 提供了非标准的迭代模式能在多项式时间内精确地计算出最小不动点。

最近，文 [138, 139, 140] 使用源于博弈理论的“策略”迭代（policy iteration）方法来计算不动点，其主要思想是把复杂抽象操作的最小不动点计算分解为一系列简单操作的最小不动点计算。“策略”迭代方法不仅能保证不动点求解的终止性，而且在许多情况下能够精确地计算出最小不动点。在区间抽象域、模版多面体抽象域上

的实验表明，“策略”迭代的方法往往比（下面将介绍的）结合“加宽/变窄”的 Kleene 迭代方法要高效并精确。但是，目前“策略”迭代方法还不能够在（强）关系型抽象域（如，多面体抽象域）上应用。因为本文大部分工作是在关系型抽象域尤其是多面体抽象域的基础上开展的，因此本文对此不作详细介绍。

(2) 基于加宽/变窄的抽象近似不动点计算

为了保证含无穷递增链的抽象域上 Kleene 不动点迭代的终止性，以及加速 Kleene 不动点迭代的收敛速度（同样适用只含有穷递增链的抽象域），抽象解释框架提供了“加宽”（widening）算子 $\triangledown$。

定义 2.3.3 (加宽) 偏序集 $(D^\sharp, \sqsubseteq^\sharp)$ 上的函数 $\triangledown : D^\sharp \times D^\sharp \to D^\sharp$ 称为**加宽**算子，当且仅当

- $\forall x^\sharp, y^\sharp \in D^\sharp, x^\sharp \sqsubseteq^\sharp x^\sharp \triangledown y^\sharp \wedge y^\sharp \sqsubseteq^\sharp x^\sharp \triangledown y^\sharp$，且
- 对于所有链 $(x_i^\sharp)_{i\in\mathbb{N}}$，如下递增链 $(y_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} y_0^\sharp & \stackrel{\text{def}}{=} & x_0^\sharp \\ \forall i \in \mathbb{N}, \quad y_{i+1}^\sharp & \stackrel{\text{def}}{=} & y_i^\sharp \triangledown x_i^\sharp \end{cases}$$

将在有穷时间内收敛，即 $\exists k \in \mathbb{N}, y_{k+1}^\sharp = y_k^\sharp$。

下面的定理将保证加宽算子会在有穷次迭代里计算出 Kleene 不动点的上近似。

定理 2.10 (基于加宽算子的不动点近似) 给定完全格 $(D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp)$，设函数 $f^\sharp : D^\sharp \to D^\sharp$ 是单调的。那么，如下递增链 $(y_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} y_0^\sharp & \stackrel{\text{def}}{=} & \bot^\sharp \\ \forall i \in \mathbb{N}, \quad y_{i+1}^\sharp & \stackrel{\text{def}}{=} & \begin{cases} y_i^\sharp & \text{if } f^\sharp(y_i^\sharp) \sqsubseteq^\sharp y_i^\sharp \\ y_i^\sharp \triangledown f^\sharp(y_i^\sharp) & \text{otherwise} \end{cases} \end{cases}$$

将在有穷时间内收敛于某个 $y_k^\sharp$（$k \in \mathbb{N}$）且 $y_k^\sharp$ 是 $\mathsf{lfp} f^\sharp$ 的上近似，即 $\mathsf{lfp} f^\sharp \sqsubseteq^\sharp y_k^\sharp$。

从上述定理，不难看出：基于加宽的迭代序列是递增的，且将在有穷迭代步内收敛于某个 $y_k^\sharp$；并且 $y_k^\sharp$ 是 $f^\sharp$ 的后不动点，从而是最小不动点 $\mathsf{lfp} f^\sharp$ 的上近似，即：$\mathsf{lfp} f^\sharp \sqsubseteq y_k^\sharp$。

定理 2.11 (基于加宽算子的不动点近似的可靠性) 给定完全格 $(D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ 和 $(D^\sharp, \sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp, \bot^\sharp, \top^\sharp)$，设具体化函数 $\gamma : D^\sharp \to D$ 是单调的，函数 $f : D \to D$ 是单调的，函数 $f^\sharp : D^\sharp \to D^\sharp$ 是 f 的可靠抽象，$x^\sharp \in D^\sharp$ 且 $\gamma(x^\sharp)$ 是 f 的前不动点。那么，如下链 $(y_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} y_0^\sharp & \stackrel{\text{def}}{=} & x^\sharp \\ \forall i \in \mathbb{N},\ y_{i+1}^\sharp & \stackrel{\text{def}}{=} & \begin{cases} y_i^\sharp & \text{if } f^\sharp(y_i^\sharp) \sqsubseteq^\sharp y_i^\sharp \\ y_i^\sharp \triangledown f^\sharp(y_i^\sharp) & \text{otherwise} \end{cases} \end{cases}$$

将在有穷时间内收敛于某个 $y_k^\sharp$（$k \in \mathbb{N}$）且 $y_k^\sharp$ 是 $\mathsf{lfp}_{\gamma(x^\sharp)} f$ 的上近似，即 $\mathsf{lfp}_{\gamma(x^\sharp)} f \sqsubseteq \gamma(y_k^\sharp)$。

从上述定理，不难看出：如果在抽象域中从某个使得 $\gamma(x^\sharp) = \bot$ 成立的 $x^\sharp$ 开始应用基于加宽的迭代，最终将得到 $\mathsf{lfp} f$ 的上近似。

注意，上述定理并不要求 D 与 $D^\sharp$ 之间存在 Galois 连接，因此同样适用于基于具体化函数的抽象解释。另外，上述定理只要求 f 是单调的，并不要求 $f^\sharp$ 是单调的。事实上，$\mathsf{lfp}_x f$ 关于 x 是单调的，但是基于加宽的迭代所得结果 $\mathsf{lfp}_{\gamma(x^\sharp)}$ 关于 $x^\sharp$ 未必是单调的。而且，加宽算子本身不具有单调性，因此即使有 $x_1 \sqsubseteq x_2$ 也未必有 $x_1 \triangledown y \sqsubseteq x_2 \triangledown y$ 或 $y \triangledown x_1 \sqsubseteq y \triangledown x_2$。事实上，文 [141] 证明了加宽算子不能关于第一参数单调，否则该算子在程序分析中将没有实际意义（例如，设计某加宽算子使其结果恒等于 $\top^\sharp$，虽然该算子满足单调性，但是基于该算子的分析得不到任何有意义的结果）。正因如此，静态分析的局部改进反而可能导致全局分析精度的降低。

值得注意的是，加宽算子的使用可能导致大量精度损失。因此，设计并选择一个好的加宽算子，对于程序分析来说至关重要。另外，也可以考虑一些改进的迭代策略，如“延迟加宽”策略即在迭代前几次使用更精确的接合操作来代替加宽算子。当然，在基于加宽的迭代稳定后，简单地使用 $f^\sharp$ 来继续迭代可以提高精度（因为 $f^\sharp(y_k^\sharp) \sqsubseteq^\sharp y_k^\sharp$）并且是可靠的，但是终止性得不到保证。

为此，抽象解释框架提供了一个变窄算子 $\vartriangle$，在加宽迭代收敛后使用变窄算子来进行递减迭代并保证在有穷时间内收敛。

定义 2.3.4 (变窄) 偏序集 $(D^\sharp, \sqsubseteq^\sharp)$ 上的函数 $\vartriangle : D^\sharp \times D^\sharp \to D^\sharp$ 称为**变窄**算子，当且仅当

- $\forall x^\sharp, y^\sharp \in D^\sharp, y^\sharp \sqsubseteq^\sharp x^\sharp \implies y^\sharp \sqsubseteq^\sharp (x^\sharp \vartriangle y^\sharp) \sqsubseteq^\sharp x^\sharp$，且
- 对于任意递减链 $(x_i^\sharp)_{i\in\mathbb{N}}$，如下递减链 $(y_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} y_0^\sharp & \stackrel{\text{def}}{=} & x_0^\sharp \\ \forall i \in \mathbb{N},\ y_{i+1}^\sharp & \stackrel{\text{def}}{=} & y_i^\sharp \vartriangle x_{i+1}^\sharp \end{cases}$$

将在有穷时间内收敛，即 $\exists k \in \mathbb{N}, y_{k+1}^\sharp = y_k^\sharp$。

下面的定理将给出基于变窄算子的不动点精化过程。

定理 2.12 (基于变窄算子的不动点精化) 设 $(D^\sharp, \sqsubseteq^\sharp)$ 为偏序集，函数 $f^\sharp : D^\sharp \to D^\sharp$ 是单调的，$x^\sharp, y^\sharp \in D^\sharp$ 且 $x^\sharp \sqsubseteq^\sharp y^\sharp$，$x^\sharp$ 是 $f^\sharp$ 的不动点，$y^\sharp$ 是 $f^\sharp$ 的后不动点。那

么，如下递减链 $(z_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} z_0^\sharp \overset{\text{def}}{=} y^\sharp \\ \forall i\in\mathbb{N},\ z_{i+1}^\sharp \overset{\text{def}}{=} \begin{cases} z_i^\sharp & \text{if } z_i^\sharp \sqsubseteq^\sharp f^\sharp(z_i^\sharp) \\ z_i^\sharp \vartriangle f^\sharp(z_i^\sharp) & \text{otherwise} \end{cases} \end{cases}$$

将在有穷时间内收敛于某个 $z_k^\sharp$（$k\in\mathbb{N}$）且 $x^\sharp \sqsubseteq z_k^\sharp \sqsubseteq y^\sharp$。

从上述定理，不难看出：从函数 $f^\sharp$ 的后不动点 $y^\sharp$ 开始的基于变窄算子的迭代序列是递减的且将在有穷迭代步内收敛于某个$z_k^\sharp$；特别地，如果 $\mathsf{lfp}f^\sharp$ 存在且 $\mathsf{lfp}f^\sharp \sqsubseteq^\sharp y^\sharp$，则有 $\mathsf{lfp}f^\sharp \sqsubseteq^\sharp z_k^\sharp \sqsubseteq^\sharp y^\sharp$。在上述定理中，如果 $z_k^\sharp \sqsubseteq^\sharp f^\sharp(z_k^\sharp)$，则有 $z_{k+1}^\sharp = z_k^\sharp$，从而 $z_k^\sharp$ 不能再被变窄算子精化了。事实上，基于变窄的迭代可以在任何迭代步停下来，因为所有基于变窄的迭代的中间结果都比不动点 $x^\sharp$ 大，因此中间结果也都是可靠的。等到迭代稳定下来，只是为了得到最好的精度。

定理 2.13 (基于变窄算子的不动点精化的可靠性) 给定完全格 $(D,\sqsubseteq,\sqcup,\sqcap,\bot,\top)$ 和 $(D^\sharp,\sqsubseteq^\sharp,\sqcup^\sharp,\sqcap^\sharp,\bot^\sharp,\top^\sharp)$，设具体化函数 $\gamma: D^\sharp \to D$ 是单调的，函数 $f: D\to D$ 是单调的，函数 $f^\sharp: D^\sharp \to D^\sharp$ 是 f 的可靠抽象，$x^\sharp, y^\sharp \in D^\sharp$ 且 $\mathsf{lfp}_{\gamma(x^\sharp)} f \sqsubseteq \gamma(y^\sharp)$。那么，如下链 $(z_i^\sharp)_{i\in\mathbb{N}}$

$$\begin{cases} z_0^\sharp \overset{\text{def}}{=} y^\sharp \\ \forall i\in\mathbb{N},\ z_{i+1}^\sharp \overset{\text{def}}{=} \begin{cases} z_i^\sharp & \text{if } z_i^\sharp \sqsubseteq^\sharp f^\sharp(z_i^\sharp) \\ z_i^\sharp \vartriangle f^\sharp(z_i^\sharp) & \text{otherwise} \end{cases} \end{cases}$$

将在有穷时间内收敛于某个 $z_k^\sharp$（$k\in\mathbb{N}$）且 $z_k^\sharp$ 是 $\mathsf{lfp}_{\gamma(x^\sharp)}$ 的上近似，即 $\mathsf{lfp}_{\gamma(x^\sharp)} f \sqsubseteq \gamma(z_k^\sharp) \sqsubseteq \gamma(y^\sharp)$。

(3) 基于加宽/变窄的抽象解释

基于加宽/变窄的抽象解释过程可概述如下：在抽象域中，首先应用加宽算子计算出抽象最小不动点的上近似即一个后不动点，然后应用变窄算子对该后不动点进行精化，从而可以得到关于抽象最小不动点的一个更为精确的上近似，如图 2.2。

简单地，加宽/变窄算子的重要性可以概括为如下两方面：

1. 对含有无穷递增链的无穷抽象域，可以保证分析的终止性；
2. 可以加快不动点收敛速度，提高计算效率，这对于提高大规模程序分析的可扩展性尤其重要。

直观上讲，加宽/变窄算子是在语义计算过程中通过启发式来对历史计算进行“归纳”并对后续计算的发展趋势进行猜测，以形成候选不变式（使得历史计算和后续计算都将满足该不变式），因而包含着很多的“智慧”成分。严格而言，加宽/变

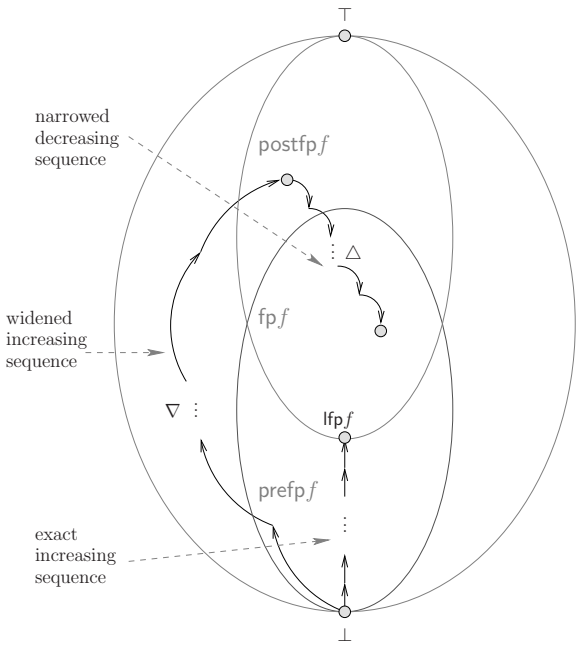


图 2.2 基于加宽/变窄的不动点迭代

窄算子的定义只提供了较弱的代数性质，这给两者的设计带来了较大困难。

正因如此，曾有研究讨论加宽/变窄算子在理论上是否有其存在的价值，即，所有基于加宽/变窄的分析能否等价地在一个有穷抽象域上开展？答案是否定，文 [141] 证明了“无穷抽象域+加宽/变窄”的方法比“有穷抽象域”（或满足递增链条件的抽象域）的方法更通用，具有更强的分析能力（可得到更精确的分析结果且同样高效）。

2.3.4 迭代策略

至此，我们把某个程序的语义通过一个语义函数 f。实际程序分析中，一个程序通常含有多个程序点，设程序中程序点的个数为 m。那么，语义函数 f 可以分解成 m 个子函数，其中，每个程序点 pc_i 都对应 f 的一个子函数 f_i。那么语义函数 f 的最小不动点，可以看成是如下（具体）语义方程系统的最小解：

$$\begin{cases} x_1 = f_1(x_1, \ldots, x_m) \\ \qquad \vdots \\ x_m = f_m(x_1, \ldots, x_m) \end{cases}$$

设具体域上的函数 f_i 在抽象域上对应的函数为 $f_i^\sharp$，下面我们考虑在抽象域上来计算具体域上语义方程系统的最小解的可靠抽象。实际上，如果不使用加宽/变窄算子，通过迭代方式来计算语义方程系统的最小解所得结果并不依赖于我们迭代方程的顺序，因此可以按任意顺序迭代（但是每个方程在迭代序列中都必需是公平

的，即在无穷序列中都出现无穷多次），因此称为“混沌迭代”（chaotic iterations）。然而，选择一个好的迭代顺序可以极大地节省计算时间。

下面我们考虑在“混沌迭代”中加入加宽/变窄算子。首先，需要选择加宽点集合 $W \subseteq PC$，然后按如下方式计算抽象向量 $x^\sharp = (x_1^\sharp, \ldots, x_m^\sharp)$ 的值：

$$x_k^{\sharp\, i+1} \stackrel{\text{def}}{=} \begin{cases} x_k^{\sharp\, i} & \text{if } k \neq L_{i+1} \text{ or } f_k^\sharp(x_1^{\sharp\, i}, \ldots, x_m^{\sharp\, i}) \sqsubseteq^\sharp x_k^{\sharp\, i} \\ x_k^{\sharp\, i} \triangledown f_k^\sharp(x_1^{\sharp\, i}, \ldots, x_m^{\sharp\, i}) & \text{otherwise if } k \in W \\ x_k^{\sharp\, i} \sqcup^\sharp f_k^\sharp(x_1^{\sharp\, i}, \ldots, x_m^{\sharp\, i}) & \text{otherwise} \end{cases}$$

其中，$x_k^{\sharp\, i}$ 表示在第 i 迭代后程序点 k 处的抽象值，L_{i+1} 表示在第 $i+1$ 次迭代中所考虑的程序点（方便起见，每次迭代只考虑更新一个程序点处的抽象值）。

但是，基于加宽的混沌迭代依赖于加宽点 W 和迭代顺序 $(L_i)_{i\in\mathbb{N}}$ 的选择。因此，需要选择一个好的加宽点集合和迭代顺序，以节省计算时间并保证计算精度。一般是根据 x_i 之间的依赖关系来设计迭代顺序，比如如果知道 x 依赖于 y 那么就需要考虑先计算 y 然后再计算 x。

一般的方法 [142] 是，首先构建各个程序点之间的依赖图，即一个有向图：图中每个结点对应一个程序点 k 及该点处的当前抽象值 $x_k^{\sharp\, i}$（如一个多面体）；如果 $f_j^\sharp$ 依赖于结点 k 处的抽象值 $x_k^\sharp$，那么结点 k 与结点 j 之间存在一条有向边。依赖图中的圈（如程序循环）通过一个后向边来刻画，比如结点 k 到结点 j 的边，其中 $k > j$。然后，把依赖图的分解成一系列可能嵌套的强连通子图（每个强连通子图对应一个循环）。迭代计算时，按拓扑序从前往后由内向外进行计算和传播。先等内层的强连通子图稳定后，再向外传播计算结果。因此，需要在每个强连通子图（循环）内设置至少一个加宽点，通常的策略是：为依赖图中的每个后向边，即每个强连通子图中最后一个结点指向第一个结点的边，设置一个加宽点。

类似地，也可以在基于加宽的混沌迭代收敛后，使用基于变窄的混沌迭代来精化其计算结果。前一节基于加宽/变窄的不动点迭代的可靠性，也保证了本节基于加宽/变窄的混沌迭代的可靠性。

2.3.5 抽象域

至此，不难看出，抽象解释框架下的所有计算都是在抽象域 $D^\sharp$ 上开展。因此，抽象域成为抽象解释框架下的一个核心要素。从设计角度看，抽象域的构成包括两个方面：

- 域表示（域元素的表示方法）：选择一种合适的表示方法（比如一族约束）来刻画一个特定类别的、计算机可表示的对象（称为域元素）集合。即考虑抽象域元

素 $x^\sharp \in D^\sharp$ 在计算机内的表示方法（存储）。

- 域操作（域元素的操作算法）：设计可靠的、高效的算法来实现一系列用来操纵域元素的操作（称为域操作）集合。比如，格相关操作 $\sqsubseteq^\sharp, \sqcup^\sharp, \sqcap^\sharp$ 的实现算法。

在程序分析中，程序状态集合通过抽象域中的域元素来近似，而程序语义动作（赋值、条件测试、控制流接合、循环等，可以理解为上一节中的语义子函数 f_i）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分析精度和计算效率间取得合理权衡。

控制流信息和数据流信息是程序分析需要考虑的两个重要方面。控制抽象主要关注如何对程序中的控制结构进行抽象。从第 2.3.4 节，我们知道基于抽象解释的程序分析需要求解语义方程系统，这些方程实际上对应了程序的控制结构。通过对程序结构进行精化（比如，循环展开、延迟控制流接合），可能得到更为精确的分析结果。此类抽象域包括迹划分抽象域 [100] 等。而数据抽象主要关注如何对程序的数据（比如，静态存储区里的全局变量、栈里的局部变量、堆里的动态分配内存）进行抽象，研究程序中数据本身的性质（非关系型抽象域），以及与其他数据之间的关系（关系型抽象域）。本文主要关注数值抽象域，即只考虑程序中数值变量上的（单个变量或多个变量间）性质。

2.4 数值程序分析与数值抽象域

本节主要介绍基于抽象解释的数值程序分析框架及该框架下的核心要素——数值抽象域，并概述已有数值抽象域。

2.4.1 数值程序的建模与分析

本文假设待分析程序中所有程序变量均为数值型变量（并将此类程序简称“数值程序”），并使用迁移系统来对待数值程序进行建模。

设程序中有 n 个数值型程序变量 $Vars = \{v_1, \ldots, v_n\}$，$\mathbb{I}$ 为数值的集合（如自然数 $\mathbb{N}$、整数 $\mathbb{Z}$、有理数 $\mathbb{Q}$、实数 $\mathbb{R}$、浮点数 $\mathbb{F}$）。一个程序环境 $\rho : Vars \to \mathbb{I}$ 把每个程序变量 v_i 映射成其实际值 $\rho(v_i)$。设程序中程序点的集合为 $\mathcal{L}$，程序状态通过程序点和程序环境所构成的序偶来描述，即 $\mathcal{S} \stackrel{\text{def}}{=} \mathcal{L} \times (Vars \to \mathbb{I})$。程序状态的幂集构成一个完全格 $(\mathcal{P}(\mathcal{S}), \subseteq, \cup, \cap, \emptyset, \mathcal{S})$。

待分析程序通过迁移系统 $(\mathcal{S}, \Lambda, \tau, \mathcal{I})$ 来建模，其中

- $\mathcal{S}$ 是程序状态的（非空）集合；

- Λ 是程序语义动作的（非空）集合；
- $\tau \subseteq \mathcal{S} \times \Lambda \times \mathcal{S}$ 是一个迁移关系，即一个状态与其可能后继关于一个给定动作的迁移关系，且 $(s, a, s') \in \tau$ 记作 $s \xrightarrow{a} s'$；
- $\mathcal{I} \subseteq \mathcal{S}$ 是初始程序状态的（非空）集合。

程序分析时，迁移关系 τ 将通过域操作中的迁移函数来建模（详见第 2.4.2 节）。比如，对于条件测试语句（语义动作），将使用域操作中的测试迁移函数来建模；对于赋值语句（语义动作），将使用域操作中的赋值迁移函数来建模。对于初始状态 $\mathcal{I}$，我们指定程序入口点 $L_0 \in \mathcal{L}$ 处的环境为 $\top \stackrel{\text{def}}{=} (Vars \to \mathbb{I})$（所有可能取值），其他程序点处的环境为 $\bot \stackrel{\text{def}}{=} \emptyset$（不可达），即 $\mathcal{I} = \{(L_0, \top)\} \cup \{(L_i, \bot) \mid L_i \in \mathcal{L}, i \in \mathbb{N}, i \geq 1\}$。

数值程序分析的目标是为程序中每个程序点产生数值不变式，即计算所有数值变量在该程序点处的所有可能取值。这是一个可达性问题，程序在具体语义上的不动点可以描述为

$$\mathsf{lfp} f \text{ 其中 } f(\mathcal{S}) \stackrel{\text{def}}{=} \mathcal{I} \cup \{s' \mid \exists s \in \mathcal{S}, \exists a \in \Lambda, s \xrightarrow{a} s'\}$$

根据第 2.3.4 节，语义函数 f 可以分解成一系列子函数 f_i，每个程序点 L_i 对应一个子函数 f_i。我们将为每个具体语义下的语义子函数找一个抽象域里的操作来建模。

抽象解释理论通过集合来表示程序性质，把一个性质理解成一个满足该性质的所有语义值所构成的集合。而程序性质之间的逻辑蕴涵关系则简单地通过子集包含关系来刻画。为了定义用户所关心的最强的静态性质（这里，我们关心的是不变式性质），抽象解释使用聚集语义来把标准语义扩展到语义值的幂集上。

这里，我们关心的只是每个程序点处数值变量的取值情况（用于回答诸如“程序点 L_i 处变量 v_i 的值是否总为正数”之类的问题），因此可以在每个程序点把该点处已经计算出来的可能环境“聚集”起来，把 $\mathcal{P}(\mathcal{L} \times (Vars \to \mathbb{I}))$ 提升到上下文集合 $\mathcal{L} \to \mathcal{P}(Vars \to \mathbb{I})$，从而得到了程序的聚集语义。其中，每个上下文 $c \in \mathcal{L} \to \mathcal{P}(Vars \to \mathbb{I})$ 把一个给定程序点映射到该点处所有可能环境所组成的集合。那么，$c(L_k)$ 定义了程序点 L_k 处的聚集语义。

这样，数值抽象域的设计只需要考虑如何对单个程序点处的所有可能环境进行抽象。抽象域中域操作的对象也变成了（某个程序点 k 处的）单个抽象环境 $x_k^\sharp$，而不再是一个（全局）状态 $x^\sharp$（即每个程序点都映射到一个环境）。进一步，我们就可以在数值抽象域上通过混沌迭代策略来计算抽象不动点。

上述几种程序语义可以通过下面的图来示意：

可达语义	聚集语义	抽象	抽象语义
$\mathcal{P}(\mathcal{L} \times (Vars \to \mathbb{I}))$	$\mathcal{L} \to \mathcal{P}(Vars \to \mathbb{I})$	(α, γ)	$\mathcal{L} \to \mathcal{D}^\sharp$

这里，我们只描述如何在面向可达语义的程序分析中使用抽象域，实际上，抽象域也可以用在面向其他语义（比如，迹语义 [100]）的程序分析中。

2.4.2 数值抽象域的设计

(1) 域表示

数值抽象域中域元素描述的是数值变量的可能取值。实际上，每个程序环境 $\rho : Vars \to \mathbb{I}$ 都可以看作是 n 维空间 $\mathbb{I}^n$ 上的一个点 $p = (\rho(v_1), \ldots, \rho(v_n))$。直观上讲，每个程序变量 v_i 对应 n 维空间 $\mathbb{I}^n$ 中的一维；点 p 在该维对应坐标轴上的值即为变量 v_i 在程序环境 ρ 中的值。程序环境的集合 $Env \stackrel{\text{def}}{=} \mathcal{P}(Vars \to \mathbb{I})$ 对应于空间 $\mathbb{I}^n$ 的子集，即该空间上一些点的集合。不同的数值抽象域就是从不同表达能力上来表示、从不同计算效率上来操作这些点集。

在聚集语义层面上，数值具体域可以看作是完全格 $(D, \subseteq, \cup, \cap, \bot, \top)$，其中$D \stackrel{\text{def}}{=} \mathcal{P}(Vars \to \mathbb{I}), \bot \stackrel{\text{def}}{=} \emptyset, \top \stackrel{\text{def}}{=} (Vars \to \mathbb{I})$。下面，我们为其设计抽象域 $(D^\sharp, \sqsubseteq^\sharp, \bot^\sharp, \top^\sharp)$。注意，聚集语义上的具体域 D 必然是一个完全格，但是我们并不要求抽象域 $D^\sharp$ 是一个格，只要求 $D^\sharp$ 是带最小、最大元的偏序集。如果具体域与抽象域之间存在 Galois 连接 (α, γ)，其中抽象化函数 $\alpha : \mathcal{P}(Vars \to \mathbb{I}) \to D^\sharp$ 及具体化函数 $\gamma : D^\sharp \to \mathcal{P}(Vars \to \mathbb{I})$，那么具体域上程序环境的任何集合在抽象域 $D^\sharp$ 上都存在一个由 α 定义的最佳抽象。比如，实数域（即 $\mathbb{I} = \mathbb{R}$）上的区间抽象域与具体域之间就存在 Galois 连接。但是，某些数值抽象域与具体域之间只存在 γ 而不存在 α，比如实数域（即 $\mathbb{I} = \mathbb{R}$）上的多面体抽象域。为此，针对一般情形，我们在数值抽象域与具体域之间只定义具体化函数 γ，且 γ 满足如下条件：① γ 是单调的；② $\gamma(\bot^\sharp) = \emptyset$；③ $\gamma(\top^\sharp) = (Vars \to \mathbb{I})$。

数值抽象域的主要设计思想是：把每个程序环境，即 n 维空间 $\mathbb{I}^n$ 上的一个点，都看作是某类 n 维约束系统的一个解。其几何意义是：把程序环境的某个集合（具体域元素）抽象成一个几何图形（抽象域元素），使得该几何图形包含了该程序环境集合所对应的所有的点。

例如，区间抽象域 [38] 选择使用单变量线性不等式系统作为其约束表示，形如 $a \le v_i \le b$ 其中 $i = 1, \ldots, n$。假设程序中只有两个变量（即 $n = 2$），且 $\mathbb{I} = \mathbb{R}$。那么，具体环境集合 $\{(v_1 = 1, v_2 = 1), (v_1 = 2, v_2 = 3)\}$ 在区间抽象域中对应的抽象元素为 $\{(v_1, v_2) \mid 1 \le v_1 \le 2, 1 \le v_2 \le 3, v_1, v_2 \in \mathbb{R}\}$。几何上，该抽象元素可以用 v_1-v_2 平面上的一个矩形填充区域来表示（类似图 2.3(a) 所示）。

(2) 常用域操作

下面介绍数值抽象域上用于静态分析的常用域操作，一般可分为以下几类：

1) 格相关操作 [1]

- 包含测试 $\sqsubseteq^\sharp$：对应抽象域上的偏序关系，应满足可靠性 $x^\sharp \sqsubseteq^\sharp y^\sharp \Rightarrow \gamma(x^\sharp) \subseteq \gamma(y^\sharp)$。在程序分析中，需要使用包含测试来检查是否达到不动点，若是则可以终止迭代计算。数值抽象域要求 γ 总存在且单调，因此设计抽象域时 $\sqsubseteq^\sharp$ 可以通过具体域上的偏序关系 $\subseteq$（集合包含关系）来定义，即 $x^\sharp \sqsubseteq^\sharp y^\sharp$ 成立当且仅当 $\gamma(x^\sharp) \subseteq \gamma(y^\sharp)$ 成立。此时，$\sqsubseteq^\sharp$ 称为包含测试的最佳抽象。但是，实际中，$\gamma(x^\sharp) \subseteq \gamma(y^\sharp)$ 的计算代价可能很高，因此可能设计低计算代价的、可靠的近似序关系 $\sqsubseteq_s^\sharp$，使得若 $x^\sharp \sqsubseteq_s^\sharp y^\sharp$ 成立则必有 $\gamma(x^\sharp) \subseteq \gamma(y^\sharp)$ 成立，反之未必成立。在程序分析过程中使用 $x^\sharp \sqsubseteq_s^\sharp y^\sharp$ 来取代 $\sqsubseteq^\sharp$ 并不影响到程序分析的可靠性，只是可能导致不动点迭代次数增多，降低总体程序分析的计算效率。因此，包含测试的设计需要与加宽算子的设计综合起来考虑，如果使用近似序关系 $\sqsubseteq_s^\sharp$，则可以配套使用收敛速度较快的加宽算子。
- 交 $\sqcap^\sharp$：即求两个抽象域元素的最大下界的上近似，应满足可靠性 $\gamma(x^\sharp) \cap \gamma(y^\sharp) \subseteq \gamma(x^\sharp \sqcap^\sharp y^\sharp)$。类似地，$\sqcap^\sharp$ 可以通过具体域上的最大下界 $\cap$（即，集合交）来定义，即 $x^\sharp \sqcap^\sharp y^\sharp$ 成立当且仅当 $\gamma(x^\sharp) \cap \gamma(y^\sharp)$ 成立。此时，$\sqcap^\sharp$ 称为交的最佳抽象。对于数值抽象域而言，其域元素的表示方法往往可以看成一个约束系统，因此，$\gamma(x^\sharp) \cap \gamma(y^\sharp)$ 可以通过把两个约束系统合取起来组成一个新的约束系统（然后消除冗余约束）来实现。但是，实际中，在某些数值抽象域中（如第 5 章将介绍的行阶梯形区间线性等式抽象域），两个抽象域元素的最大下界不存在。这种情况下，我们仅要求求交结果是两元素最大下界的上近似，以保证可靠性。最差情形下，我们可以把两个抽象域元素的求交结果定义为 $\top^\sharp$。
- 接合（join）$\sqcup^\sharp$：即求两个抽象域元素的最小上界的上近似，应满足可靠性 $\gamma(x^\sharp) \cup \gamma(y^\sharp) \subseteq \gamma(x^\sharp \sqcup^\sharp y^\sharp)$。接合操作用于对程序中控制流接合（如，if-then-else-endif 语句的 endif 处）进行抽象。类似地，$\sqcup^\sharp$ 可以通过具体域上的最下上界 $\cup$（集合并）来定义，即 $x^\sharp \sqcup^\sharp y^\sharp$ 成立当且仅当 $\gamma(x^\sharp) \cup \gamma(y^\sharp)$ 成立。此时，$\sqcup^\sharp$ 称为接合的最佳抽象，又称为强接合。简单地，$\gamma(x^\sharp) \cup \gamma(y^\sharp)$ 可以通过把两个约束系统析取起来来理解。但是，数值抽象域的表示方法一般是一个约束系统，即一系列约束的合取。因此，需要使用一个只使用合取的约束系统来上近似两个约束系统

[1] 注意：实际上，数值抽象域可以不是格。

的析取（两个对应解集的集合并）。这在数学上通常不是一个容易的问题，因此接合操作往往是抽象域设计的难点之一。实际中，在某些数值抽象域中（如第 5 章将介绍的行阶梯形区间线性等式抽象域），两个抽象域元素的最小上界不存在。这种情况下，我们仅要求接合操作的结果是两元素最小上界的上近似，以保证可靠性。最差情形下，我们可以把两个抽象域元素的接合结果定义为 $\top^\sharp$。

2) 迁移函数

记数值变量集合 $Vars$ 上的数值表达式集合为 $Exprs$。表达式 $expr \in Exprs$ 的语义定义为 $\epsilon[\![expr]\!] : \mathcal{P}(Vars \to \mathbb{I}) \to \mathcal{P}(\mathbb{I})$，其抽象语义定义为 $\epsilon[\![expr]\!]^\sharp : D^\sharp \to \mathbb{M}^\sharp$，其中，$\mathbb{M}^\sharp$ 表示计算机可表示的抽象值集合。记 $Cmds$ 为程序命令（程序语句）集合，那么命令 $c \in Cmds$ 的语义可以定义为 $\tau[\![c]\!] : \mathcal{P}(Vars \to \mathbb{I}) \to \mathcal{P}(Vars \to \mathbb{I})$，其抽象语义可定义为 $\tau[\![c]\!]^\sharp : D^\sharp \to D^\sharp$。

命令 c 的迁移函数描述了命令 c 执行之前和之后两处抽象环境之间的关系。具体而言，对于条件测试语句 $\text{if}(expr_1 \bowtie expr_2)$（其中 $\bowtie \in \{<, \leq\}$），条件测试迁移函数的语义为

$$\tau[\![expr_1 \bowtie expr_2]\!](Env) \stackrel{\text{def}}{=} \{\rho \mid \rho \in Env, \exists val_1 \in \epsilon[\![expr_1]\!](\rho), \exists val_2 \in \epsilon[\![expr_2]\!](\rho), val_1 \bowtie val_2\}$$

测试迁移函数的语义在于把那些不满足约束 $expr_1 \bowtie expr_2$ 的抽象环境过滤掉。对于赋值语句 $v_i := expr$，赋值迁移函数的语义为

$$\tau[\![v_i := expr]\!](Env) \stackrel{\text{def}}{=} \{\rho[v_i \mapsto val] \mid \rho \in Env, \exists val \in \epsilon[\![expr]\!](\rho)\}$$

- 条件测试迁移函数 $\tau[\![expr_1 \bowtie expr_2]\!]^\sharp(x^\sharp)$，其中 $x^\sharp$ 为语句执行之前的抽象环境。设 $y^\sharp = \tau[\![expr_1 \bowtie expr_2]\!]^\sharp(x^\sharp)$，条件测试迁移函数的设计需满足可靠性 $\tau[\![expr_1 \bowtie expr_2]\!](\top) \cap \gamma(x^\sharp) \subseteq \gamma(y^\sharp)$。如果 $\tau[\![expr_1 \bowtie expr_2]\!](\top) \cap \gamma(x^\sharp) = \gamma(y^\sharp)$，那么条件测试抽象迁移函数是最佳抽象。一般地，条件测试抽象迁移函数可以通过把约束 $expr_1 \bowtie expr_2$ 抽象成一组抽象域可表示的约束（比如把非线性约束抽象成线性约束），然后把这些约束加入到 $x^\sharp$ 的约束系统中（然后消除冗余约束）来实现。但是，实际中，在某些数值抽象域中（如第 5 章将介绍的行阶梯形区间线性等式抽象域），条件测试迁移函数的最佳抽象可能不存在。
- 投影操作 $\pi(x^\sharp, v_i) \stackrel{\text{def}}{=} \{x^\sharp[v_i \mapsto val^\sharp] \mid val^\sharp \in \mathbb{M}\}$，满足可靠性 $\{x[v_i \mapsto val] \mid x \in \gamma(x^\sharp), val \in \mathbb{I}\} \subseteq \gamma(\pi(x^\sharp, v_i))$。投影操作用来从抽象环境 $x^\sharp$ 中删除关于变量 v_i 的信息，同时又不影响到其他变量间的关系。投影操作对于非确定性赋值语句和过程间分析的处理非常重要。比如，关于非确定性赋值语句的迁移函数 $[\![v_i := random()]\!]^\sharp(x^\sharp)$ 就可以通过 $\pi(x^\sharp, v_i)$ 来定义。因此，本文把投影操作也

看作是一种迁移函数。

- 赋值迁移函数 $\tau[\![v_i := expr]\!]^\sharp(x^\sharp)$，一般可以通过条件测试迁移函数、投影操作、变量重命名来建模：

$$\tau[\![v_i := expr]\!]^\sharp(x^\sharp) \stackrel{\text{def}}{=} \big(\pi(\tau[\![v_i' - expr = 0]\!]^\sharp(x^\sharp), v_i)\big)\,[v_i'/v_i]$$

其中，新鲜变量 v_i' 引入进来用于表示表达式 $expr$ 的值。赋值语句可以分为两类：可逆赋值语句和不可逆赋值语句。对于不可逆赋值语句（比如 $v_i := v_j + 1$），引入 v_i' 是有必要的。对于可逆赋值语句（指 v_i 在 $expr$ 中出现，比如 $v_i := 2 * v_i + 1$），赋值迁移函数还可以通过替换方式来实现，即在抽象环境 $x^\sharp$ 对应的约束系统中把所有变量 v_i 的出现使用一个逆表达式 $expr'$ 来替换，因为此时赋值语句右端 v_i 可以通过关于左端 v_i 的表达式 $expr'$ 来表达（如 $\frac{1}{2}(v_i - 1)$）。

3) 外推操作（extrapolation）

外推操作统用来统称加速不动点计算收敛的操作。注意：一般意义上的外推操作不要求保证终止性，但加宽/变窄操作要求终止性。

- 加宽操作 $\triangledown$：由于程序中存在循环或者递归调用，我们需要设计加宽操作用来加速并保证程序不动点迭代的终止性。数值抽象域中，加宽操作的设计一般可以从两个角度来考虑：一是从约束系统角度，保留那些稳定的约束，同时丢弃不稳定的约束；二是从几何图形角度，观察图形中哪些部分是稳定。但是，加宽操作的设计也是数值抽象域设计的一个难点。
- 变窄操作 $\triangle$：变窄操作用来精化加宽操作所得到的（后）不动点，该精化过程的终止性也需要得到保证。但是，数值抽象域中，变窄操作没有加宽操作那么重要，而且其设计较加宽操作更难。数值程序分析中，一个简单的策略是在加宽序列稳定后，继续使用求交操作来代替变窄操作进行迭代，但是只迭代有穷次。这一精化过程也是终止的且得到的结果也是可靠的。

在抽象域域操作实现算法的设计上，接合操作和加宽操作是重点也是难点。

2.4.3 已有数值抽象域概览

目前，抽象解释领域存在许多表达能力、计算效率各有特色的数值抽象域（已有约近 30 种），包括区间域 [38]、八边形域 [41]、仿射等式域 [39]、多面体域[40] 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某种权衡。表 2.1 列出了部分目前已存在的和本文新提出的数值抽象域，以及这些抽象域所能表达的数值性质。当然，还有许多数值抽象域并没有在表 2.1 中列出，比如区间同余域 [143]、带同余域 [42]、网格域 [144]、椭圆域 [21]、子多面体域 [94] 等。

表 2.1 已有数值抽象域列表

类别	抽象域名称	约束表示形式
非关系型	符号 [38]	$\pm v_i \geq 0$
	区间 [38]	$a_i \leq v_i \leq b_i$ ($a_i, b_i \in \mathbb{Q}$ 或 $a_i, b_i \in \mathbb{F}$)
	简单同余 [104]	$v_i \equiv a_i \mod b_i$ ($a_i, b_i \in \mathbb{Z}$ 或 $a_i, b_i \in \mathbb{Q}$)
	单变量区间线性不等式 [145]（本文）	$[a_i, b_i]v_i \leq c_i$ ($a_i, b_i, c_i \in \mathbb{Q}$ 或 $a_i, b_i, c_i \in \mathbb{F}$)
弱关系型	带（Zones,DBM） [68]	$v_i - v_j \leq c_{ij}$ ($c_{ij} \in \mathbb{Q}$ 或 $c_{ij} \in \mathbb{F}$)
	八边形 [41]	$\pm v_i \pm v_j \leq c_{ij}$ ($c_{ij} \in \mathbb{Q}$ 或 $c_{ij} \in \mathbb{F}$)
	五边形 [93]	$v_i \in [a_i, b_i] \wedge v_i < v_j$ ($a_i, b_i \in \mathbb{Q}$)
	每不等式两变量 [90]	$a_{ij}v_i + b_{ij}v_j \leq c_{ij}$ ($a_{ij}, b_{ij}, c_{ij} \in \mathbb{Q}$)
	Logahedra [146]	$a_{ij}v_i + b_{ij}v_j \leq c_{ij}$ $(a_{ij}, b_{ij} \in \{-2^k, 0, 2^k \mid k \in \mathbb{Z}\}, c_{ij} \in \mathbb{Q})$
	八面体 [89]	$\Sigma_j a_{ij}v_j \leq c_i$ ($a_{ij} \in \{-1, 0, 1\}, c_i \in \mathbb{Q}$)
	模版多面体 [91]	$\Sigma_j a_{ij}v_j \leq c_i$ ($a_{ij}, c_i \in \mathbb{Q}$ 且a_{ij}的值预先固定)
	符号范围约束 [92]	$\Sigma_{j=i+1}^{n} a_{ij}v_j + c_i \leq v_i \leq \Sigma_{j=i+1}^{n} b_{ij}v_j + d_i$ $(a_{ij}, b_{ij}, c_i, d_i \in \mathbb{Q})$
	Max-Plus 多面体 [105]	$\max(a_0, x_1 + a_1, \ldots, x_n + a_n)$ $\leq \max(b_0, x_1 + b_1, \ldots, x_n + b_n)$ $(a_0, a_1, \ldots, a_n, b_0, b_1, \ldots, b_n \in \mathbb{Q})$
	行阶梯形区间线性等式 [147]（本文）	$\Sigma_j [a_{ij}, b_{ij}]v_j = [c_i, d_i]$ ($a_{ij}, b_{ij}, c_i, d_i \in \mathbb{Q}$ 或 $a_{ij}, b_{ij}, c_i, d_i \in \mathbb{F}$) 但整个约束系统是行阶梯形
	线性绝对值等式（本文）	$\Sigma_j a_{ij}v_j + \Sigma_j b_{ij}\lvert v_j\rvert = c_i$ ($a_{ij}, b_{ij}, c_i \in \mathbb{Q}$)
关系型	仿射等式 [39]	$\Sigma_j a_{ij}v_j = c_i$ ($a_{ij}, c_i \in \mathbb{Q}$)
	线性同余 [148]	$\Sigma_j a_{ij}v_j \equiv c_i \mod b_i$ ($a_{ij}, b_i, c_i \in \mathbb{Q}$)
	梯形同余 [143]	$\Sigma_j a_{ij}v_j \equiv [c_i, d_i] \mod b_i$ ($a_{ij}, b_i, c_i, d_i \in \mathbb{Q}$)
	多面体 [40]	$\Sigma_j a_{ij}v_j \leq c_i$ ($a_{ij}, c_i \in \mathbb{Q}$)
	浮点多面体 [149]（本文）	$\Sigma_j a_{ij}v_j \leq c_i$ ($a_{ij}, c_i \in \mathbb{F}$)
	区间多面体 [150]（本文）	$\Sigma_j [a_{ij}, b_{ij}]v_j \leq c_i$ ($a_{ij}, b_{ij}, c_i \in \mathbb{Q}$ 或 $a_{ij}, b_{ij}, c_i \in \mathbb{F}$)
	线性绝对值不等式（本文）	$\Sigma_j a_{ij}v_j + \Sigma_j b_{ij}\lvert v_j\rvert \leq c_i$ ($a_{ij}, b_{ij}, c_i \in \mathbb{Q}$)

按照表达性质的类型，数值抽象域一般可分为如下三类：

- 非关系型抽象域：这些抽象域只能表示单个变量的性质。比如，区间抽象域 [38] 只能表示一个变量的取值范围。这些抽象域虽然表达能力弱，不能跟踪变量间的关系，但是一般存在高效的实现算法，可用于大规模的程序分析；
- 关系型抽象域：这些抽象域能够表示变量间的（某类）任意关系。比如，多面体抽象域 [40] 能够表示任意线性关系。然而，这些域往往有着较高的计算代价，在实际分析中可扩展性较差；
- 弱关系型抽象域：这些域在关系型与非关系型抽象域之间进行折中，通过某种方式限制其能够表达的数值关系。比如，每不等式两变量域 [90] 限制每个约束中所涉及的变量数（不超过 2）；另一些，比如八面体域 [89]，则限制约束中变量系数的值（为 ± 1）; 还有一些，如八边形域 [41] 则对两个方面都限制，限制每不等式两变量约束的系数必须是 ± 1，Logahedra 域 [146] 则限制每不等式两变量约束的系数（的绝对值）必须是 2 的幂，从而使得该抽象域的表达能力介于八边形域和每不等式两变量域之间。

在域操作的实现算法上，数值抽象域也各有差异。一些域如模版多面体域 [91]、每不等式两变量域 [90] 等，依赖于线性规划技术；另一些域如带域（zones） [68]、八边形域 [41] 等，都是基于图算法来实现；还有，算术自动机（arithmetic automata）域 [151, 152, 153] 则使用有穷状态自动机来识别整数值的二进制表示。表 2.2 给出了常用或本文相关数值抽象域的时空复杂度及其核心算法。

从约束系统的角度，抽象域可以根据其特征进行如下分类：(1) 允许的数值表达式：单个变量、线性表达式、模表达式等，还可以对系数的取值进行约束（比如只能是 ± 1）；(2) 在一个约束中可同时出现的变量的个数：只能 1 个（非关系型）、最多 2 个（弱非关系型）、任意个（关系型）；(3) 使用等式还是不等式：比如，仿射等式域使用仿射等式，与之对应的多面体域则使用线性不等式；此外，能表示非严格不等式（$\leq$ 型）的抽象域也可以有一个版本来支持严格不等式（$<$ 型），比如未必封闭多面体域 [154]。

从代数角度上，数值抽象域一般是基于某类代数理论，比如多面体抽象域 [40] 基于线性代数、Max-Plus 多面体抽象域 [105] 基于 Max-Plus 代数、同余抽象域 [104]基于模代数、区间抽象域 [38] 基于区间代数、区间多面体域 [150] 基于区间线性代数等。

另一方面，在几何上，数值抽象域往往对应某类特定几何图形区域，其中位于该图形区域内的每个点代表了某个可能的程序环境（对所有程序变量的赋值）。图 2.3 给出了几个具有代表性的已有数值抽象域及其对应的几何图形。对应地，图 2.4

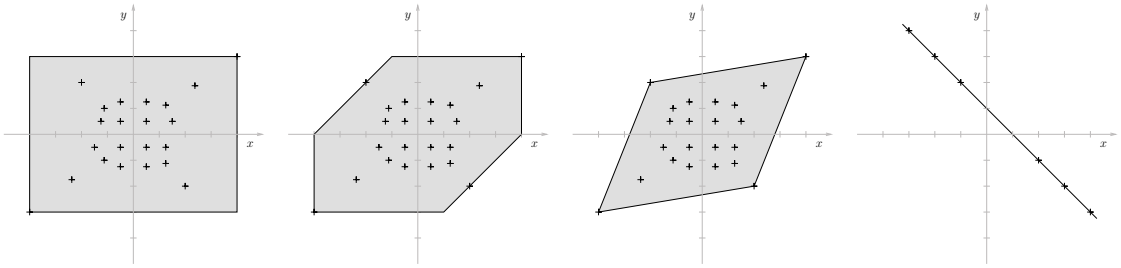
表 2.2 常用或本文相关数值抽象域及其时空复杂度

抽象域	复杂度（最差情况）		核心算法
	时间	空间	
区间 [38]	$O(n)$	$O(n)$	区间算术
带（Zones,DBM） [68]	$O(n^3)$	$O(n^2)$	基于图的最短路径算法
八边形 [41]	$O(n^3)$	$O(n^2)$	基于图的最短路径算法
每不等式两变量 [90]	指数级	指数级	二维线性规划
模版多面体 [91]	多项式级	$O(mn)$, m为模版数	线性规划
符号范围约束 [92]	多项式级	$O(n^2)$	线性规划
仿射等式 [39]	$O(n^3)$	$O(n^2)$	高斯消去法
多面体 [40]	指数级	指数级	双重描述法的对偶转化
单变量区间线性不等式 [145] (本文)	$O(n)$	$O(n)$	区间算术
浮点多面体 [149](本文)	指数级	指数级	浮点线性规划
区间多面体 [150] (本文)	指数级	指数级	区间线性规划
行阶梯形区间线性等式 [147](本文)	$O(n^4)$	$O(n^2)$	高斯消去法的区间变种
线性绝对值不等式(本文)	指数级	指数级	双重描述法的对偶转化
线性绝对值等式 (本文)	指数级	$O(n^2)$	双重描述法的对偶转化

给出了几个本文新提出的数值抽象域及其对应的几何图形。

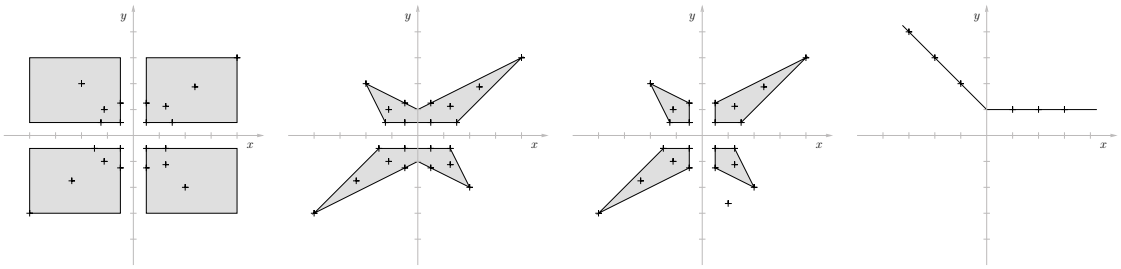
2.5 面向浮点程序的静态分析与基于浮点的抽象域实现

上一节中，我们假设待分析程序中数值 $\mathbb{I} \in \{\mathbb{R}, \mathbb{Q}, \mathbb{Z}\}$，其中实数 $\mathbb{R}$、有理数 $\mathbb{Q}$、整数 $\mathbb{Z}$ 都是数学概念上的数。在计算机领域，由于计算机只能存储有限的位数来表示数，因此计算机所能表示的数的个数和范围都是有限的。比如，若采用二进制补码，64 位机器字所能表示的有符号整数的范围为 $[-2^{63}, 2^{63}-1]$。因此，计算机里的机器整数与数学概念上的整数在计算方面有一定的区别，比如机器整数可能出现上溢（如，机器整数正最大值加 1 将超出机器整数的表示范围，结果变成机器整数的



(a) 区间域　(b) 八边形域　(c) 多面体域　(d) 仿射等式域

图 2.3 典型数值抽象域



(a) 单变量区间线性不等式域　(b) 行阶梯形区间线性等式域　(c) 区间多面体域/线性绝对值不等式域　(d) 线性绝对值等式域

图 2.4 本文提出的数值抽象域

负最小值）。同样地，在计算机中，数学上的实数也是不能被完全表示的，一般采用浮点数来表示实数的一个有穷子集。实际程序语言（如 C、Java 等）也是通过机器整数和浮点数来表示数据类型。

2.5.1 IEEE 754 浮点模型

IEEE 754 [155] 是目前最广泛使用的浮点数运算标准，也是硬件浮点运算部件事实上的工业标准，被许多处理器（如 Intel、PowerPC）所采用，并被大多数程序设计语言（如 C、Java）所支持。

(1) IEEE 754 浮点数表示

在 IEEE 754 标准中，一个浮点数的二进制表示（其存储格式见图 2.5）采用科学计数法来刻画，即 $(-1)^S \times M \times 2^E$，包括三个域：

- 符号位 S，决定该数是正数（$S=0$）还是负数（$S=1$）；
- 阶码 $E = e - \mathbf{bias}$，其中 e 为 $\mathbf{e}$ 位偏置指数，偏置量 $\mathbf{bias} = 2^{\mathbf{e}-1} - 1$；
- 尾数 $M = m_0.m_1m_2\ldots m_{\mathbf{p}}$，其中 $f = .m_1m_2\ldots m_{\mathbf{p}}$ 表示 $\mathbf{p}$ 位有效小数，而 m_0 是隐含位，无需存储。

其中 $\mathbf{e}, \mathbf{bias}, \mathbf{p}$ 是格式相关的。IEEE 标准定义了多种格式，其中最基本的两种格式是：

- 32 位单精度格式，其中 $\mathbf{e} = 8, \mathbf{p} = 23, \mathbf{bias} = 127$；
- 64 位双精度格式，其中 $\mathbf{e} = 11, \mathbf{p} = 52, \mathbf{bias} = 1023$。

精度	符号位	偏置指数	有效小数
	S	e	f
单精度(32位)	1位	8位	23位
双精度(64位)	1位	11位	52位

图 2.5 IEEE 754 标准浮点表示的存储格式

根据偏置指数 e 的取值，浮点数可分成三种类型（如图 2.6 所示）：

- 规格化数：$(-1)^S \times 1.f \times 2^{e-\mathbf{bias}}$，当 $1 \leq e \leq 2^{\mathbf{e}} - 2$ 时；
- 非规格化数：$(-1)^S \times 0.f \times 2^{1-\mathbf{bias}}$，当 $e = 0$ 且 $f \neq 0$ 时；
- 特殊数
  - $+0$ 或 -0：当 $e = 0$ 且 $f = 0$ 时；
  - $+\infty$ 或 $-\infty$：当 $e = 2^{\mathbf{e}} - 1$ 且 $f = 0$ 时；
  - NaN（Not a Number）：当 $e = 2^{\mathbf{e}} - 1$ 且 $f \neq 0$ 时。

设浮点格式的集合为 $\mathbf{F}$，对于每种浮点格式 $\mathbf{f} \in \mathbf{F}$，我们定义

- $mf_{\mathbf{f}} \stackrel{\text{def}}{=} 2^{1-\mathbf{bias}-\mathbf{p}}$：最小的非零正浮点数；
- $Mf_{\mathbf{f}} \stackrel{\text{def}}{=} (2 - 2^{-\mathbf{p}})2^{2^{e}-\mathbf{bias}-2}$：最大的有穷正浮点数。

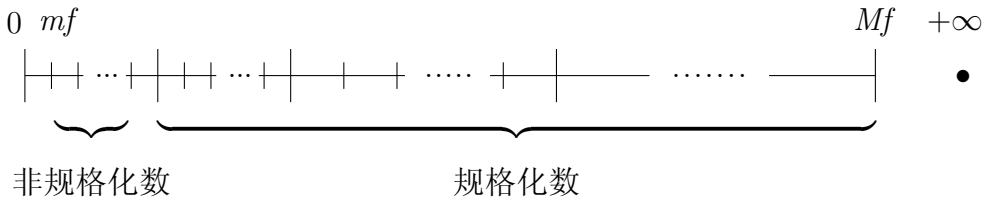


图 2.6 IEEE 浮点数及其分布

(2) IEEE 754 浮点计算模型

由于 IEEE 浮点数是通过有限的存储位元来表示，因此它能够精确表示的数是有限的，也是离散的。在相邻两个可精确表示的浮点数之间，存在无穷多个 IEEE

浮点数所不能精确表示的实数。例如，实数 0.1 就不能被 IEEE 754 标准精确表示。为了表示这些实数，IEEE 754 的方法是用距离该实数最近的浮点数来近似表示。

IEEE 754 标准支持 4 种不同的舍入模式：

- 向最近舍入：舍入结果 $R_{\mathbf{f},n}(x)$ 为最接近原数 x 的浮点值，当有两个最接近的浮点值时首选“偶数”值；
- 向 $+\infty$（向上）舍入：舍入结果 $R_{\mathbf{f},+\infty}(x)$ 为不小于原数 x 的最小浮点值；
- 向 $-\infty$（向下）舍入：舍入结果 $R_{\mathbf{f},-\infty}(x)$ 为不大于原数 x 的最大浮点值；
- 向 0（截断）舍入：舍入结果 $R_{\mathbf{f},0}(x)$ 与原数 x 有相同的符号，且其绝对值 $|R_{\mathbf{f},0}(x)|$ 为不大于 $|x|$ 的最大浮点值。

本中使用 $\oplus_{\mathbf{f},r}, \ominus_{\mathbf{f},r}, \otimes_{\mathbf{f},r}, \oslash_{\mathbf{f},r}$ 来表示浮点加法、减法、乘法、除法四则算术，其中 $\mathbf{f} \in \{32, 64\}$ 表示浮点格式，$r \in \{+\infty, -\infty, 0, n\}$ 表示舍入模式，本文有时也采用 ? 来表示任意舍入模式。实际上，浮点运算一般采用一种称为精确舍入的方式来实现 [112]：有效浮点数上的中间运算结果存放在较长的寄存器中，只有当中间结果被转换到某个具体精度的浮点格式时，有效位以外的位元才被舍去（按照某种舍入模式）。

由于有效位以外的数字将被舍去，因此可能产生一些误差，称为舍入误差。在程序开发中，我们一般把计算机的浮点算术看作是数学概念上实数算术的一种近似。IEEE 754 标准确保了浮点运算将产生具有预期性质的结果。并且，会在浮点状态寄存器中标识出五种类型的浮点异常：

- 无效运算：运算的操作数或者产生的结果是 NaN，即该运算在数学上是无效的，例如 $(+\infty)+(-\infty)$、$0\times\infty$、$0/0$、∞/∞、$\sqrt{-1}$ 等；
- 被零除：$x/0$ 其中 $x \neq 0$，例如 $1/0$ 等；
- 上溢：运算产生的结果的绝对值大于 Mf，例如 $Mf + Mf$ 等；
- 下溢：运算产生的结果的绝对值小于 mf，例如 $mf \times mf$ 等；
- 不精确：运算产生的结果无法用浮点数来精确表示，例如 1/3、1/10 等。

在实际安全攸关软件中，下溢和不精确异常一般认为是容许的，但是上溢、被零除、无效运算异常会被认为是运行时错误，将导致程序终止。本文关心如何可靠地分析浮点程序中可能存在的上溢、被零除、无效运算等运行时错误。

2.5.2 面向浮点程序的分析

(1) 浮点程序分析与验证的目标性质

尽管单个浮点操作所带来的舍入误差通常都很小，但如果舍入误差在长序列的

浮点计算中不断累加时，也可能产生严重后果，甚至导致灾难性后果。实际上，本文在第 1.1 节所给出的两个因软件缺陷导致灾难事故发生的著名案例本质上都是由于浮点方面的问题导致的：

- Ariane 5 型火箭首发爆炸事故 [6]：原因是火箭中惯性参考系统 SRI 把一个与火箭在水平方向上的速度相关的 64 位浮点数转换为 16 位的有符号整数的时候，产生了溢出。
- 爱国者（Patriot）导弹拦截飞毛腿（Scud）导弹失败事件 [5]：原因是爱国者导弹内部的系统时钟使用整数来计时，一个嘀嗒相当于 0.1 秒，由于 0.1 不是可精确表示的浮点数 [2]，通过“整数嘀嗒数×0.1” 计算所得的浮点时间存在着自系统启动以来所累积的时间误差，最终导致不能及时拦截来袭导弹。

上面的案例也反映了浮点程序分析与验证所需要关心的目标性质。具体而言，浮点程序相关的目标性质一般包括如下几个方面：

① 浮点程序是否会出现运行时错误，比如上溢、除零错、无效运算等。例如，静态分析工具 Astrée [21, 22] 就是面向此类目标性质。

② 分析程序中舍入误差的源头及累计上界。例如，静态分析工具 Fluctuat [117, 24, 25] 就是面向此类目标性质，可用来分析浮点程序中舍入误差传播情况。

③ 证明浮点程序在规定的误差范围内实现了某种目标计算。例如，程序验证工具 Caduceus [118, 119] 就是面向此类目标性质，可用来证明浮点程序满足某些规约。

这里，本文主要关注浮点程序是否会出现运行时错误。

(2) 浮点数给分析与验证带来的困难

由于舍入误差的存在，实数 $\mathbb{R}$ 上的一些代数性质如结合律和分配律，对于浮点数不再适用。例如，64 位双精度浮点计算中，$(10^{30} \ominus_{64,?} 10^{30}) \oplus_{64,?} 1 = 1 \neq 0 = (10^{30} \oplus_{64,?} 1) \ominus_{64,?} 10^{30}$。而且，0 也不再是唯一的（有 $+0$ 和 -0 之分）。注意，交换律对于浮点加法与浮点乘法仍然适用，浮点数的取反操作和比较操作（$+0$ 和 -0 在比较操作中是相等的）仍然是精确的。

此外，相同的浮点程序在不同的体系结构（如 Intel IA32, Intel x86_64, PowerPC 等）上的运行可能也不一致。IEEE 754 标准规定的关于浮点运算的具体语义依赖于硬件平台。比如，对于精确舍入 $a \oplus b = Round_{\mathbf{f},\mathbf{r}}(a+b)$ 而言，其中的操作 $a+b$ 需要在比 $\mathbf{f}$ 位数更多的寄存器里计算，而使用何种寄存器就与硬件相关了，不同平台可能选择使用不同位数的寄存器。另外，一些编译器往往依赖于一些良好的代数性

[2]实际上，Patriot 导弹使用 24 位定点寄存器来存储 0.1，但与使用浮点寄存器存在类似问题。

质来对程序进行优化，因此，对浮点程序的优化可能改变浮点计算的执行顺序，导致结果不可靠。关于浮点计算性质和缺陷的更为详细的描述请见文献 [112, 113]。

本文只关注浮点程序在源代码级是否存在运行时错误，而忽略体系结构、编译器层面可能给浮点程序执行带来的不一致性。

源代码级浮点程序分析的困难主要在于：

① 难以对源程序中的浮点表达式进行抽象：由于浮点算术不满足结合律和分配律等代数性质，因此在精确代数表达式上可以进行的变换和化简不能应用在浮点表达式上。比如，我们不能把赋值语句 $x := x \oplus (1 \ominus x)$ 化简成 $x := 1$。

② 基于浮点表达式的程序不变式（形如 $\bigoplus_i a_i \otimes x_i \leq c$）很难在（关系型）抽象域中可靠地维护：抽象域的实现需要对这些系数进行操作（如合并同一变量的系数），但是在浮点意义下这些操作的可靠性很难得到保证。比如，$(a_i \otimes x_i) \oplus (a'_i \otimes x_i)$ 不一定等于 $(a_1 \oplus a'_1) \otimes x_i$。又如，在带域或者八边形域中，由于$x \ominus y \leq c \wedge y \ominus z \leq d$ 不能导出 $x \ominus z \leq c + d$（注意即使是使用 $c \oplus_{\mathbf{f},+\infty} d$ 来计算 $c + d$ 也不能保证可靠性），因此这两个域中很关键的求闭包操作很难可靠地实现。

③ 由浮点变量 $x_i (i = 1, \ldots, n)$ 构成的抽象环境集合对应 $n-$维空间上的离散点集合且这些点不是均匀分布的（两相邻浮点数间的距离不是固定的常数）：为这些非均匀分布的离散点设计最优抽象域往往很困难。

④ 舍入操作的语义不是线性的：首先存在多种舍入模式；且即使对于某种舍入模式，舍入操作所带来的舍入误差的大小与原值不能简单通过线性关系来表达，至少取决于原值是规格化数还是非规格化数。当然，由于浮点格式规定的二进制位数是有限的且舍入模式也是固定的几种，舍入操作理论上可以通过枚举所有的可能性来表达，但是实际上这种枚举的空间太大了。因此，直接应用模型检验技术来验证浮点程序的可行性也不大。

(3) 浮点程序的一种分析方法

对于浮点程序分析，最简单的方法是使用区间抽象域 [21]。区间抽象域是非关系型抽象域，只需要为每个变量 x 维护一个区间 $[\underline{x}, \overline{x}]$ 来表示其取值范围。而区间抽象域上的操作也可以很容易地通过浮点算术来可靠实现，即采用向外舍入的区间算术：计算 $\underline{x}$ 时向下舍入，计算 $\overline{x}$ 时向上舍入。使用浮点区间抽象域来分析浮点程序时，抽象域上的数据类型可以是跟待分析程序中的数据类型一样，也可以是位数更少的数据类型。比如，分析 64 位双精度浮点程序时，我们可以使用 64 位双精度浮点实现的区间抽象域，也可以使用 32 位单精度浮点实现的区间抽象域（此时，如

果程序中有数值超出 32 位单精度浮点表示的范围就使用无穷大来表示)。注意，如果使用 128 位长双精度浮点实现的区间抽象域来分析 64 位双精度浮点程序时，可靠性则不能保证。

然而，区间抽象域不能表达变量间的关系，而许多应用需要维护两个或两个以上变量间的关系。另一方面，关系型抽象域往往是在一些具有良好代数性质（如结合律和分配律）的理想数学结构（如有序环等）上设计的。但是，浮点算术不具有结合律和分配律。

为此，Miné [84, 42] 建议仍然使用实数表达式来表达浮点程序中的数值不变式，即把不变式中所有变量都看成是实数变量且不变式中的运算也是实数意义下的精确运算。通过把浮点型变量构成的程序环境的集合 $Env_{\mathbb{F}}$ 抽象成实数型变量构成的程序环境的集合 $Env_{\mathbb{R}}$，使得 $Env_{\mathbb{F}} \subseteq Env_{\mathbb{R}}$，从而保证可靠性。这是可行的，因为 $\mathbb{F} \subseteq \mathbb{R}$，即所有的浮点数都是实数。但是，因为实数比浮点数更稠密，这个过程可能带来一些精度损失：浮点数之间（本不应存在的）的实数可能组合构造出新的但是实际上不可能出现的浮点数。这一问题与使用实数型变量构成的程序环境集合来对整数型变量构成的程序环境集合进行抽象所存在的问题非常类似。例如，对于区间抽象域上的程序环境集合 $\{(x,y) \mid 0 \leq x \leq 0.5, 0 \leq y \leq 0.5\}$，如果把 x, y 看成实数则 $x+y$ 可能取到整数 0 或 1，但是，如果把 x, y 看成整数则 $x+y$ 只可能等于整数 0。这样，在实数集 $\mathbb{R}$ 上所设计的在实数意义下是最佳或精确的抽象域操作，在浮点数意义下不一定是最佳或精确的。

基于这样的思想，Miné 建议通过如下步骤来分析浮点程序 [84, 42]：

① 把浮点数上的确定语义抽象成实数上的非确定语义（把浮点舍入误差考虑在该非确定语义中）；

② 在实数上的非确定语义（“具体”语义）下基于理想数据类型（如实数 $\mathbb{R}$ 或有理数 $\mathbb{Q}$）来设计抽象域；

③ 为理想的抽象域设计其可靠浮点实现方法，以提高计算效率（但可能带来某些精度损失）。

按照抽象解释理论 [156]，上述抽象过程可以通过图 2.7 所示的语义层次来示意。下面，本文将对各个步骤进行详细描述。

① 浮点程序到实数程序的抽象过程

为了把浮点数上的运算可靠地抽象成实数上的运算，Miné 建议使用一个区间线性化的技术来把源程序中的浮点表达式（形如 $\bigoplus_i a_i \otimes x_i$）抽象成带区间系数的

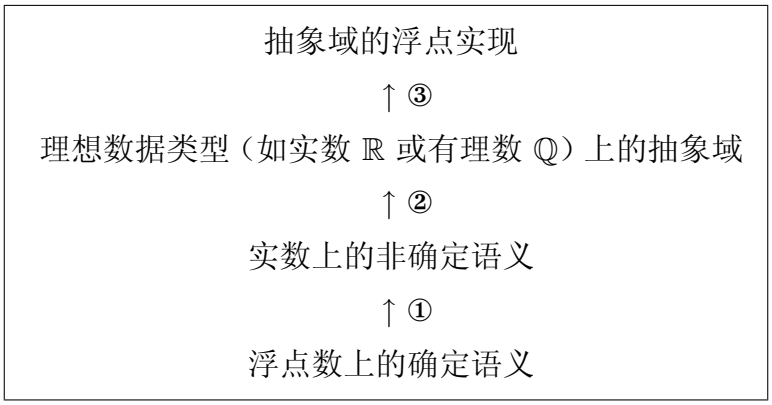


图 2.7 浮点相关的抽象语义层次
(这里，$S_1 \to S_2$ 表示语义 S_2 是语义 S_1 的抽象)

实数表达式（形如 $\Sigma_i[b_i, c_i] \times x_i$）[84, 42]。抽象后，我们无需为带区间系数的实数表达式指定其计值顺序，因为实数满足结合律、分配律等代数性质。当然，如果我们改变原浮点表达式的计值顺序可能会影响到区间线性化技术的输出结果。

首先，我们需要知道单个浮点运算所带来的舍入误差的上界。浮点舍入误差 $R_{\mathbf{f},r}(x)$ 依赖于原数 x 大小，具体而言：

- 如果 x 是规格化数，那么 $|R_{\mathbf{f},r}(x) - x|$ 将小于 $\varepsilon_{\text{rel}} \cdot |x|$，其中 $\varepsilon_{\text{rel}} = 2^{-\mathbf{p}}$（$\mathbf{p}$ 是浮点格式 $\mathbf{f}$ 中有效小数的位数）。此时，我们考虑的是大小为 ε_{rel} 的相对舍入误差。
- 如果 x 是非规格化数，那么 $|R_{\mathbf{f},r}(x) - x|$ 将小于 ε_{abs}，其中 $\varepsilon_{abs} = mf_{\mathbf{f}}$（$mf_{\mathbf{f}}$ 是浮点格式 $\mathbf{f}$ 中最小的非零正非规格化数，也是两个相邻非规格化数之间的距离）。此时，我们考虑的是大小为 ε_{abs} 的绝对误差。

因此，我们可以用如下公式把两种情形统一起来

$$|R_{\mathbf{f},r}(x) - x| \leq \max(\varepsilon_{\text{rel}} \cdot |x|, \varepsilon_{\text{abs}})$$

但是，max 操作是非线性的。为此，我们进一步把舍入误差上近似为

$$|R_{\mathbf{f},r}(x) - x| \leq \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}}$$

由于绝对值操作也是非线性的，我们希望去掉绝对值操作。根据如下观察：$b \geq 0$ 时，$|y| \leq b$ 等价于 $y = [-1, 1] \times b$（即 $y = [-b, b]$），我们进一步把上述不等式写成

$$R_{\mathbf{f},r}(x) - x = [-1, 1](\varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}})$$

即

$$R_{\mathbf{f},r}(x) = [1 - \varepsilon_{\text{rel}}, 1 + \varepsilon_{\text{rel}}] \times x + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$$

这样，我们就可以把所有浮点表达式转化成实数上的区间线性表达式。比如，

$$x \oplus_{\mathbf{f},r} y$$

即

$$R_{\mathbf{f},r}(x+y)$$

可以抽象成

$$[1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times (x+y) + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$$

即

$$[1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$$

这种浮点抽象方法的好处在于，其表示形式已与舍入模式无关，对于所有的舍入模式都是可靠的。因为 $R_{\mathbf{f},r}(x)$ 总满足 $R_{\mathbf{f},-\infty}(x) \leq R_{\mathbf{f},r}(x) \leq R_{\mathbf{f},+\infty}(x)$，而 $|R_{\mathbf{f},r}(x) - x| \leq \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}}$ 已经把 $r = -\infty$ 和 $r = +\infty$ 两种极端情况都考虑在内了。这在实际程序分析中是很有意义的，因为不一定会知道每个浮点操作（执行时机器所处）的舍入模式。

当然，如果知道当前浮点操作所处的舍入模式，可以定义更为精确的浮点抽象：

- 向最近舍入模式:

$$|R_{\mathbf{f},n}(x) - x| \leq (\varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}})/2$$

- 向 $+\infty$ 舍入模式:

$$0 \leq R_{\mathbf{f},n}(x) - x \leq \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}}$$

- 向 $-\infty$ 舍入模式:

$$-\varepsilon_{\text{rel}} \cdot |x| - \varepsilon_{\text{abs}} \leq R_{\mathbf{f},n}(x) - x \leq 0$$

- 向 0 舍入模式:

$$|R_{\mathbf{f},n}(x) - x| \leq \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}} \ \wedge \ |R_{\mathbf{f},n}(x)| \leq |x|$$

另外，如果知道 x 的取值范围，也可以定义更为精确的浮点抽象：

- 若 $x \geq 0$:

$$R_{\mathbf{f},r}(x) \geq 0 \ \wedge \ (1-\varepsilon_{\text{rel}}) \cdot x - \varepsilon_{\text{abs}} \leq R_{\mathbf{f},r}(x) \leq (1+\varepsilon_{\text{rel}}) \cdot x + \varepsilon_{\text{abs}}$$

- 若 $x \leq 0$:

$$R_{\mathbf{f},r}(x) \leq 0 \ \wedge \ -(1-\varepsilon_{\text{rel}}) \cdot x - \varepsilon_{\text{abs}} \leq R_{\mathbf{f},r}(x) \leq -(1+\varepsilon_{\text{rel}}) \cdot x + \varepsilon_{\text{abs}}$$

- 若 x 是规格数（即 $|x| \geq mMf$，其中mMf 为最小的正规格数）:

$$|R_{\mathbf{f},r}(x) - x| \leq \varepsilon_{\text{rel}} \cdot |x|$$

- 若 x 是非规格数（即 $|x| \leq mMf$，其中mMf 为最小的正规格数）:

$$|R_{\mathbf{f},r}(x) - x| \leq \varepsilon_{\text{abs}}$$

- 若 x 是个常数，且可被浮点格式 $\mathbf{f}$ 精确表示:

$$R_{\mathbf{f},r}(x) = x$$

简单起见，本文使用最具有通用性的浮点抽象方法：

$$R_{\mathbf{f},r}(x) = [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$$

例 2.5.1 飞机作动器速率饱和会降低系统稳定性，并增加驾驶员诱发振荡（*PIO*）的趋势，可能危及到飞行安全。为此，飞行控制系统中一般引入软件速率限制器（*rate limiter*）来防止作动器速率饱和。图 *2.8* 给出了软件速率限制器的一个浮点实现程序，及其抽象后对应的实数程序。

② 为实数上的非确定语义设计抽象域

经过步骤 ① 后，输入程序中将出现区间作为变量系数的表达式（如图 2.8）。这给抽象域的设计带来了挑战，因为目前已有的数值抽象域均不支持区间变量系数。已有数值抽象域的设计仍停留在线性约束的层面，即抽象域的约束表示中变量的系数必须是一个标量（常数）。

为了能够使已有的数值抽象域（如多面体域等）也可以分析这种区间系数表达的非确定语义。Miné [42] 建议把区间系数抽象成标量系数，相当于把相对误差转化成绝对误差。主要思想如下：给定区间系数表达式 $\Sigma_i[a_i, b_i] \times x_i + [c, d]$，设变量 x_i 的取值范围为 $[\underline{x_i}, \overline{x_i}]$，那么下面的标量系数表达式必然是 $\Sigma_i[a_i, b_i] \times x_i + [c, d]$ 的上近似

$$\Sigma_i e_i \times x_i + [c', d']$$

```
float x, y, d, s, r;
y ← 0;
while random() {
  x ← [−128, 128];
  d ← [1, 16];
  s ← y;
  r ← x ⊖32,? s;
  y ← x;
  if r ≤ ⊖d { y ← s ⊖32,? d; } else
  if d ≤ r  { y ← s ⊕32,? d; }
}
```

(a) 原浮点程序

```
real x, y, d, s, r;
y ← 0;
while random() {
  x ← [−128, 128];
  d ← [1, 16];
  s ← y;
  r ← [1 − 2^−23, 1 + 2^−23] × x − [1 − 2^−23, 1 + 2^−23] × s + [−2^−149, 2^−149];
  y ← x;
  if r ≤ −d {
    y ← [1 − 2^−23, 1 + 2^−23] × s − [1 − 2^−23, 1 + 2^−23] × d + [−2^−149, 2^−149];
  } else
  if d ≤ r {
    y ← [1 − 2^−23, 1 + 2^−23] × s + [1 − 2^−23, 1 + 2^−23] × d + [−2^−149, 2^−149];
  }
}
```

(b) 抽象后对应的实数程序

图 2.8 速率限制器的浮点程序及其抽象后对应的实数程序

其中，e_i 是区间 $[a_i, b_i]$ 内的任意常数，$[c', d'] \triangleq [c, d] + \Sigma_i [a_i - e_i, b_i - e_i] \times [\underline{x}_i, \overline{x}_i]$）。这样，待分析程序中的表达式都将进一步抽象成标量系数表达式，区间只会出现在常数项。已有数值抽象域将可以直接分析这种只含标量系数表达式的程序。

但是，上述把区间系数抽象成标量系数的过程可能导致大量精度损失。为此，本文的一个重要目标就是设计抽象域使其直接支持区间变量系数，比如区间多面体抽象域（第 4 章）、行阶梯形区间线性等式抽象域（第 5 章）等。

2.5.3 基于浮点的抽象域实现

本节主要考虑图 2.7 中所示步骤 ③。目前，已有数值抽象域都是在理想数据类型（如实数 $\mathbb{R}$ 或有理数 $\mathbb{Q}$）上设计的。实现时，为了保证可靠性，一般采用多精度有理数（比如，使用 GMP 库 [81]）来编程实现域操作的相关算法。多精度有理数运算属于精确计算，不存在浮点数运算中的舍入误差问题，因而不用担心实现会影响到抽象域的可靠性或者导致额外精度损失。但是，多精度有理数实现在时间和空间上的开销很大，而且容易导致“大系数”问题，可扩展性难以提高。尤其，对浮点程序而言，在抽象成实数程序后，不可避免地会出现高阶的数值，比如，对于 32 位单精度浮点格式而言 $\varepsilon_{\text{rel}} = 2^{-23}, \varepsilon_{\text{abs}} = 2^{-149}$，对于 64 位双精度浮点格式而言 $\varepsilon_{\text{rel}} = 2^{-52}, \varepsilon_{\text{abs}} = 2^{-1074}$。对于高阶数值，其多精度有理数的表示方法将耗费大量存储空间，且涉及高阶数值的最大公约数求解和化简计算的时间开销也很大。

相比多精度有理数，基于浮点数来实现抽象域可以极大地提高计算效率并节省存储开销，从而为提高数值抽象域在实际应用中的扩展性提供了有效途径。注意，抽象域的数据类型选择与待分析程序中使用的数据类型没有依赖关系：基于浮点实现的抽象域也可以用来分析整数程序，基于多精度有理数实现的抽象域也可以用来分析浮点程序。即使待分析程序使用的是精确算术（如整数程序），使用基于浮点实现的抽象域也会极大地提高分析的计算效率。尤其，对于浮点程序而言，使用基于浮点实现的抽象域比使用基于多精度有理数实现的抽象域要更自然、更适合。

但是，许多抽象域在设计时往往假设所使用的算术具有良好的代数性质，这样可以减少算法设计的难度。因此，如果直接简单地使用浮点数来实现已有抽象域，得到的结果可能不可靠。因为浮点算术不是精确的且不具有结合律、分配律等代数性质，为了保证可靠性，在浮点算法的设计时，需要考虑浮点计算的顺序，还要考虑舍入误差的存在。此外，除了保证浮点实现的可靠性，还应该尽量保证浮点实现的精度，不能导致太多精度损失。

简而言之，浮点舍入误差的存在给面向浮点程序的分析与验证带来了挑战。另

一方面，浮点计算的时空高效性也给抽象域的高效实现带来了机遇。

2.6 小结

本章主要介绍了文中将使用到的相关理论和技术，为后面章节的工作介绍起到了一个铺垫作用。

抽象解释为本文研究提供了坚实的理论基础，并提供了一个完整的上层理论框架。数值程序分析作为抽象解释在静态程序分析中的典型应用，是本文的研究领域。本章介绍了抽象解释的基本理论和框架，并详细阐述了如何把抽象解释理论应用到数值程序分析中。数值抽象域作为基于抽象解释的数值程序分析的核心要素，是本文研究的聚焦点。本章介绍了数值抽象域的主要设计思想，并对已有数值抽象域进行了总结和分析。另外，本章介绍了浮点数的表示模型和计算模型，还介绍了面向浮点程序的分析方法，并从中启发出本文研究的动机和目标。最后，本文说明了基于浮点数来实现数值抽象域的动机和意义。

第三章　浮点多面体抽象域

3.1　引言

本章给出经典多面体抽象域的一种可靠浮点实现方法，即浮点多面体抽象域。

多面体抽象域由 Cousot 和 Halbwachs 于 1978 年提出 [40]，主要用来推导程序中变量之间的线性关系。自提出以来，该抽象域在软硬件系统分析与验证领域取得很广泛的应用 [88]。至今，多面体抽象域仍然是目前表达能力最强、应用最广泛的数值抽象域之一。但是，由于其高复杂度，多面体抽象域在许多实际应用中受到可扩展性和易处理性方面的限制 [83]。

首先，目前公开可用的多面体域的实现库，如 Polylib [157]、NewPolka [158]、Parma Polyhedra Library （PPL）库 [159] 等，都是基于多精度有理数来实现的。因此，在时间和空间上的开销很大，而且容易导致“大系数”问题，从而极大地限制了其可扩展性 [83]。

另一方面，多面体域的实现 [158, 159] 一般是基于一种双重描述法（Double Description Method） [40]，即基于多面体的一种对偶表示方法：约束表示（Constraint Representation）和生成子表示（Generator Representation 或称 Frame Representation）。在约束表示中，一个多面体可以描述成一组有穷线性不等式的合取，即一个有穷线性不等式系统。对偶地，在生成子表示中，一个多面体可以表示成一组生成子（顶点和射线）的有穷集合。在多面体域中，一些域操作（如，交和测试）比较容易在约束表示上高效地实现出来；而另外一些域操作（如，投影和接合）则比较容易在生成子表示上高效地实现出来；还有一些操作（如，加宽）则对两种表示都需要。因此，在实现时，一般是把两种表示都维护起来，并在两种表示方法之间进行转化。这种对偶转化一般采用 Chernikova 算法 [160] 来实现。但是，Chernikova 算法可能产生指数级的输出 [160]，比如 n 维 cross-polytope 的生成子表示是线性的但是其约束表示则是指数级的。最近，作为双重描述法的一种替代候选，Simon 和 King[82] 通过把多面体域上求凸闭包操作转化成约束表示上的变量消除问题，使得多面体抽象域也可以只基于约束表示来实现，从而至少可以消除生成子表示可能带来的复杂度瓶颈。

为此，本文采用浮点数并且只基于约束表示，给出了多面体抽象域的一种新的实现方法，并且保证了所实现的浮点多面体域的可靠性。另外，考虑到基于约束的

多面体抽象域的处理能力主要受限于其高代价的接合操作，本文提出一系列低代价的弱接合操作来作为接合操作的可靠替代候选，以进一步提高多面体抽象域的计算效率和可扩展性。

本章的结构组织如下：3.2 节回顾只基于约束表示的有理数多面体抽象域的设计；3.3 节介绍基于约束的浮点多面体抽象域的设计，并讨论因为浮点实现所带来的精度和效率问题及其改进策略；3.4 节介绍一系列基于约束的多面体域的弱接合以及这些弱接合的启发式结合策略；3.5 节介绍浮点多面体域的原型系统实现，给出并分析实验结果；3.6 节是对本章的小结。

3.2 基于约束的有理数多面体域

3.2.1 域表示

在基于约束表示的多面体抽象域中，在有理数 $\mathbb{Q}$ 上，一个（凸）多面体 P 可通过一个线性不等式系统 $P = \{Ax \leq b\}$ 来描述，其中 $A \in \mathbb{Q}^{m\times n}$ 是一个有理数矩阵，$b \in \mathbb{Q}^m$ 是一个有理数向量，m 是不等式系统中约束的数目，n 是不等式系统中变量的个数。其语义在几何上对应位于该多面体内的点的集合，即 $\gamma(P) = \{x \in \mathbb{Q}^n \mid Ax \leq b\}$，其中每个点 x 代表一个可能的程序环境（即对所有变量 x 的一种可能赋值）。

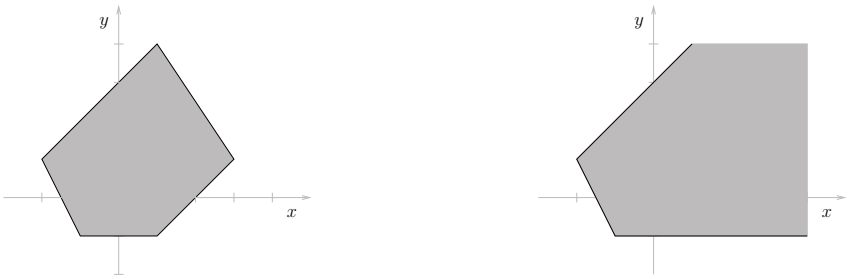
注意，多面体抽象域与具体域（程序环境集合）之间不存在抽象函数 α，因此也不存在 Galois 连接。比如，对于圆盘 $\{(x,y) \in \mathbb{Q}^2 \mid x^2 + y^2 \leq 1\}$，包含该圆盘的最小多面体不存在。多面体抽象域只构成格，而非完全格。另外，等式型线性约束 $\Sigma_i a_i x_i = b$ 可通过一对 $\leq$ 型约束来表示，即 $\Sigma_i a_i x_i \leq b \wedge -\Sigma_i a_i x_i \leq -b$，而形如 $\Sigma_i a_i x_i < b$ 的严格不等式则可以可靠地抽象为 $\Sigma_i a_i x_i \leq b$。

例 3.2.1 图 *3.1* 给出了一个有界多面体(又称多胞体)和一个无界多面体的示例。其中，*(a)* 中多面体对应的约束表示为 $\{-y \leq 1, -2x-y \leq 3, -x+y \leq 3, x-y \leq 2, 3x+2y \leq 11\}$；*(b)* 中多面体对应的约束表示为 $\{-y \leq 1, -2x - y \leq 3, -x + y \leq 3\}$。

3.2.2 域操作

首先，我们回顾线性不等式系统上的两个基本技术。

- Fourier-Motzkin 消除法：用来从定义多面体 P 的线性不等式系统中消除某个



(a) 有界多面体　　(b) 无界多面体

图 3.1 多面体几何示例

变量 x_i，即

$$FME(P,x_i) \stackrel{\text{def}}{=} \left\{ (-a_i^-)c^+ + a_i^+c^- \;\middle|\; \begin{array}{l} c^+ = (\sum_k a_k^+ x_k \le b^+) \in P, a_i^+ > 0 \\ c^- = (\sum_k a_k^- x_k \le b^-) \in P, a_i^- < 0 \end{array} \right\} \cup \{\ (\textstyle\sum_k a_k x_k \le b)\ \in P \mid a_i = 0\ \}.$$

不难看出，$FME(P,x_i)$ 中将不再涉及 x_i。注意，Fourier-Motzkin 变量消除算法可能产生冗余约束。

- 线性规划（LP）：是在一组给定有穷线性约束（称为可行空间）下寻找某线性函数（称为目标函数）的最优值的方法。线性规划的理论和方法都已发展得很成熟，在科学与工程领域应用非常广泛。目前，该领域研究人员已经开发了许多高效的、可扩展到成千上万规模变量数和约束数的线性规划算法和工具。

  求最小目标函数值的线性规划问题具有如下形式：

$$\min e \text{ subject to } P$$

  其中 P 是一个多面体的约束表示，称为可行空间，e 是一个线性表达式形如 $\Sigma_i a_i x_i$，称为目标函数。与之对应的是，求最大目标函数值的线性规划问题：$\max e$ subject to P。一个不等式约束 $\varphi : (\Sigma_i a_i x_i \le b)$ 被一个多面体 P 蕴含，记作 $P \models \varphi$，当且仅当 $\gamma(P)$ 中的所有点都满足不等式 φ。这可以通过线性规划来判定，即求解 LP 问题：$\mu = \max \Sigma_i a_i x_i$ subject to P，如果 $\mu \le b$，则 φ 被多面体 P 蕴含，即 $P \models \varphi$，否则 $P \not\models \varphi$。

下面，我们将详细描述多面体抽象域上用于静态分析所需要的常用域操作的实现方法。

(1) 冗余约束消除

一个多面体的约束表示不是惟一的，两个不同的约束表示可能在几何上对应同

一个多面体。比如，约束集合 $\{x = 0, y = 0\}$ 与 $\{x = 0, y = x\}$ 在几何上表示 x-y 平面上的同一个点 $(0, 0)$。在实现时，为了保证执行效率，一般希望约束越少越好，因而需要消除冗余约束。若一个不等式 $\varphi \in P$ 能够被 P 中其他约束蕴含，即 $P \setminus \{\varphi\} \models \varphi$，则称不等式 φ 在 P 中是冗余的。对于 P 中任意一个不等式约束 $\varphi : (\Sigma_i a_i x_i \leq b)$，可通过求解 LP 问题 $\mu = \max \Sigma_i a_i x_i \text{ subject to } P \setminus \{\varphi\}$，来检查 φ 是否是冗余的。如果 $\mu \leq b$，则 φ 是冗余的，需要从 P 中删除。该过程可重复直到 P 中没有更多的不等式可删除。

(2) 空多面体测试

空多面体对应了多面体抽象域中的 $\perp$ 元素。一个多面体是空的，当且仅当其约束集合是不可行的。事实上，在 LP 求解器计算目标函数最大（小）值时，LP 求解器会自动检查该约束系统的可行性。在程序分析过程中，约束往往是一条条增量式地加入，因此空多面体测试也可以通过一种增量式方法来实现。当添加一个新的约束 $\Sigma_i a_i x_i \leq b$ 到一个非空多面体 P 中时，我们求解 LP 问题：$\mu = \min \Sigma_i a_i x_i \text{ subject to } P$。如果 $b < \mu$，则说明新多面体是空的。

(3) 投影

多面体域中的一个重要操作是从多面体中消除所有关于变量 x_i 的信息，而不影响到其他变量之间的关系信息。为此，我们定义投影操作

$$\pi(P, x_i) \stackrel{\text{def}}{=} \{\, x[x_i/y] \,|\, x \in \gamma(P), y \in \mathbb{Q} \,\}$$

其中 $x[x_i/y]$ 表示向量 x 的第 i 个元素被 y 替代。这可以通过经典的 Fourier-Motzkin 变量消除算法 $FME(P, x_i)$ 来实现，即从定义 P 的系统中消除所有 x_i的出现。

投影可用来处理非确定性赋值语句，即

$$[\![x_i := \text{random}()]\!]^\sharp(P) \stackrel{\text{def}}{=} FME(P, x_i)$$

其中$[\![\cdot]\!]^\sharp(P)$ 表示一个程序语义动作（如赋值语句、条件测试语句等）作用在多面体 P 上的效果。另外，在程序分析中，投影操作也常用于对过程间分析的处理。

(4) 接合

在基于抽象解释的程序分析中，为了对程序中控制流接合（control-flow join 或control-flow merge）进行抽象，需要计算程序环境的并（union）。然而，多面体的（集合）并不是封闭的，即两多面体的并不一定是一个多面体。包含两多面体 P 和

P' 的并的最小多面体是这两个多面体的凸闭包（convex hull），记作 $P \sqcup_{CH} P'$。因此，在多面体抽象域中，接合（join）操作 $P \sqcup P'$ 就是通过凸闭包操作 $P \sqcup_{CH} P'$ 来实现的。注意，两多面体的凸闭包可能包含一些不在原来两个多面体内的点。

例如，对于图 3.2 中所示的程序，*brandom* 表示随机布尔值。在执行完 *if* 语句后，在程序点 ① 处，程序变量 x 和y 的可能取值为 $(x = 0 \wedge y = 0) \vee (x = 1 \wedge y = 1)$，几何上对应 x-y 平面上的点 $(0, 0)$ 和点 $(1, 1)$，即多面体 $P = \{x = 0, y = 0\}$ 和多面体 $P' = \{x = 1, y = 1\}$ 的并。这两个点的并不是一个凸的多面体，包含这两个点的并的最小多面体是连接这两个点的线段，即这两个多面体的凸闭包 $P \sqcup_{CH} P' = \{y = x, 0 \leq x \leq 1\}$。

```
if (brandom){
   x := 0;      y := 0;
} else {
   x := 1;      y := 1;
} ①
```

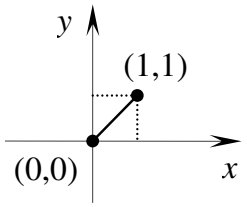


图 3.2 多面体抽象域上的接合操作

在传统基于对偶表示的多面体抽象域 [40] 中，两多面体的凸闭包通过生成子表示上的生成子（即顶点和射线）的集合并来计算，然后通过应用 Chernikova 算法把所得结果的生成子表示转化成约束表示，实现冗余生成子的消除。在基于约束的多面体域中，一般采用文献 [82] 中给出的方法来计算。其主要思想是：两个多面体 P 和 P' 的凸闭包可以通过考虑分别来自这两个多面体的任意两点 $z \in P$ 和 $z' \in P'$ 之间的凸组合（convex combination）来构造。两点 $z \in P$ 和 $z' \in P'$ 之间的凸组合定义为 $x = \sigma_1 z + \sigma_2 z'$，其中 $\sigma_1, \sigma_2 \in [0, 1]$ 且 $\sigma_1 + \sigma_2 = 1$，几何上 x 代表了 z 与 z' 之间连线线段上的任意一点，如图 3.3 所示。

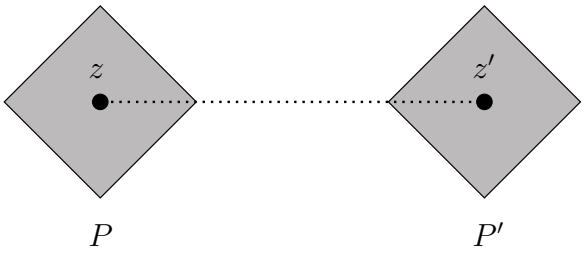


图 3.3 基于凸组合的多面体凸闭包计算

给定 $\gamma(P)=\{x\in\mathbb{Q}^n\,|\,Ax\leq b\}$ 和$\gamma(P')=\{x\in\mathbb{Q}^n\,|\,A'x\leq b'\}$，$P$ 和 P' 的凸闭包为

$$\left\{x\in\mathbb{Q}^n\;\middle|\;\begin{array}{l}\exists\sigma_1,\sigma_2\in\mathbb{Q},\ z,z'\in\mathbb{Q}^n.\\ x=\sigma_1 z+\sigma_2 z'\wedge\sigma_1+\sigma_2=1\wedge\sigma_1\geq 0\wedge\\ \qquad Az\leq b\quad\wedge\quad A'z'\leq b'\ \wedge\sigma_2\geq 0\end{array}\right\}$$

其中 $\sigma_1,\sigma_2\in\mathbb{Q}$ 和 $x,z,z'\in\mathbb{Q}^n$。为了消除非线性等式 $x=\sigma_1 z+\sigma_2 z'$，我们引入 $y=\sigma_1 z$ 和 $y'=\sigma_2 z'$，并把上述系统松弛成

$$\gamma(P_{CH})=\left\{x\in\mathbb{Q}^n\;\middle|\;\begin{array}{l}\exists\sigma_1,\sigma_2\in\mathbb{Q},\ y,y'\in\mathbb{Q}^n.\\ x=y+y'\ \wedge\sigma_1+\sigma_2=1\ \wedge\sigma_1\geq 0\wedge\\ Ay\leq\sigma_1 b\ \ \wedge\ A'y'\leq\sigma_2 b'\ \wedge\sigma_2\geq 0\end{array}\right\}\tag{3.1}$$

从约束系统 (3.1) 中投影掉 σ_1,σ_2,y,y' 即可得到 P 和 P' 的凸闭包（的拓扑闭包）。

例 3.2.2 给定多面体 $P_1=\{-4x-y\leq 34^{(1)},-2x+3y\leq 24^{(2)},x+3y\leq 6^{(3)},x+y\leq 0^{(4)},2x-3y\leq -5^{(5)},x-4y\leq 0^{(6)}\}$, $P_2=\{-x-2y\leq 2^{(7)},-2x-y\leq -5^{(8)},-4x+y\leq -7^{(9)},-3x+4y\leq 11^{(10)},-x+6y\leq 41^{(11)},2x+3y\leq 53^{(12)},2x+y\leq 39^{(13)},2x-y\leq 33^{(14)},3x-4y\leq 52^{(15)},x-4y\leq 28^{(16)}\}$，$P_1$ 与 P_2 的凸闭包 $P_1\sqcup_{CH}P_2=\{-4x-y\leq 34^{(1)},-2x+3y\leq 24^{(2)},-x+6y\leq 41^{(11)},2x+3y\leq 53^{(12)},2x+y\leq 39^{(13)},2x-y\leq 33^{(14)},3x-4y\leq 52^{(15)},x-4y\leq 28^{(16)},-4x+13y\leq 76^{(17)},-3x-16y\leq 56^{(18)}\}$，如图 *3.4* 所示。

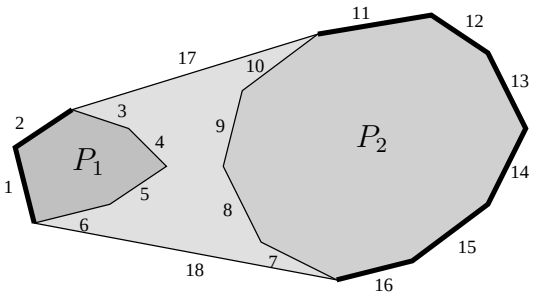


图 3.4 多面体凸闭包示例

(5) 迁移函数

条件测试迁移函数. 一个基于精确算术的线性条件测试可以转化成 $\Sigma_i a_i x_i\leq c$形式。此时，对于条件测试迁移函数 $[\![\Sigma_i a_i x_i\leq c]\!]^\sharp(P)$，可以简单地通过把约束 $\Sigma_i a_i x_i\leq c$ 加入到多面体 P 中来实现。注意，加入一条新的约束可能引入冗余约束（新加入的

约束可能是冗余的，或者可能导致原来非冗余约束变成冗余）或者使得多面体变得不可行。因此，在条件测试迁移函数执行后，需要进行冗余约束消除和空多面体测试。对于更为复杂的条件测试语句，比如涉及浮点算数或者非线性操作等，可以通过文 [42] 中提出的方法把条件测试约束可靠地抽象成 $\Sigma_i a_i x_i \leq c$ 形式。

两个多面体 P_1 和 P_2 的交 $P_1 \sqcap P_2$ 定义为这两个多面体的约束表示的集合并所对应的多面体，从而 $\gamma(P_1 \sqcap P_2) = \gamma(P_1) \cap \gamma(P_2)$。类似地，两个多面体的交的结果多面体可能不可行，或者可能引入冗余约束。

赋值迁移函数. 把某个表达式 e 赋值给变量 x_j 可以通过使用投影、条件测试及变量重命名来实现：

$$[\![x_j := e]\!]^\sharp(P) \stackrel{\text{def}}{=} (FME([\![x'_j - e = 0]\!]^\sharp(P), x_j))[x'_j/x_j] \ .$$

首先，引入一个新鲜变量 x'_j 来保存表达式 e 的值，然后我们通过 Fourier-Motzkin 变量消除方法把 x_j 投影掉，最终的线性系统可通过把 x'_j 重命名回 x_j 来得到。特别地，对于可逆赋值语句（即变量 x_j 也出现在表达式 e 中）而言，如 $x := x+1$，我们也可以通过替代的方式来实现其迁移函数。设表达式 e 为 $\Sigma_i a_i x_i$（其中 $a_j \neq 0$），那么我们可以在多面体 P 的约束表示中把所有变量 x_j 的出现通过表达式 $\frac{1}{a_j}(x_j - \Sigma_{i \neq j} a_i x_i)$ 来替换，化简后就可以得到迁移函数执行后的结果多面体。

(6) 包含测试

多面体域上的序关系可以定义为 $P_1 \sqsubseteq P_2$ 当且仅当 $\gamma(P_1) \subseteq \gamma(P_2)$，即 $\forall \varphi_2 \in P_2, P_1 \models \varphi_2$，并称 P_2 是 P_1 的一个上近似。因此，两个多面体 P_1 和 P_2 之间的包含测试，可以归结成检查 P_2 中每个不等式是否被 P_1 蕴含的问题，这可以通过 LP 来实现。对 P_2 中每个不等式 $\Sigma_i a_i x_i \leq b$，计算 $\mu = \max \Sigma_i a_i x_i$ subject to P_1。如果出现 $\mu > b$ 情形，则 $P_1 \sqsubseteq P_2$ 不成立。

(7) 加宽

多面体抽象域不满足递增链条件，因此为了处理循环结构，需要使用加宽算子加速不动点计算并保证不动点计算的终止性。多面体抽象域上的第一个加宽算子是在文 [40] 中提出的。

定义 3.2.1 (第一个加宽算子) 给定两个基于线性不等式约束表示的多面体 $P_1 \sqsubseteq P_2$，

$$P_1 \triangledown_{1st} P_2 \stackrel{\text{def}}{=} \mathcal{S}_1$$

其中

$$\mathcal{S}_1 = \{\, \varphi_1 \in P_1 \mid P_2 \models \varphi_1 \,\}$$

∇_{1st} 的主要思想是保留 P_1 中的那些稳定了的约束，即 P_1 中满足 P_2 的不等式约束集合 $\mathcal{S}_1$。其改进 [161]，现在已经成为多面体域上的标准加宽算子：

定义 3.2.2 (标准加宽算子) 给定两个基于线性不等式约束表示的多面体 $P_1 \sqsubseteq P_2$，

$$P_1 \ \nabla_{std}\ P_2 \stackrel{\text{def}}{=} \mathcal{S}_1 \cup \mathcal{S}_2$$

其中

$$\mathcal{S}_1 = \{\, \varphi_1 \in P_1 \mid P_2 \models \varphi_1 \,\},$$
$$\mathcal{S}_2 = \{\, \varphi_2 \in P_2 \mid \exists \varphi_1 \in P_1, \gamma(P_1) = \gamma((P_1 \setminus \{\, \varphi_1 \,\}) \cup \{\, \varphi_2 \,\}) \,\}.$$

标准加宽算子的关键点是不仅保留 P_1 中满足 P_2 的不等式约束集合 $\mathcal{S}_1$，还保留那些来自 P_2 的、且与 P_1 中某个不等式关于 P_1 相互冗余的不等式集合 $\mathcal{S}_2$。$\mathcal{S}_2$ 保证了加宽结果不依赖于 P_1 和 P_2 的表示，尤其是当 P_1 和 P_2 中存在仿射等式时（即仿射空间不为空）。注意，为了保证终止性，应用标准加宽算子时，对输入有如下限制：

- P_1 和 P_2 中每个仿射等式约束都必须拆分成两个线性不等式约束；
- P_1 中没有冗余约束。

在已有多面体域的实现库中，$P_1 \ \nabla_{std}\ P_2$ 都是基于双重描述法来实现的。本文下面给出一个基于约束的实现方法。首先，不难看出 $\mathcal{S}_1$ 可通过蕴含检查来计算。另外，下面的定理说明了 $\mathcal{S}_2$ 也可以归结到蕴含检查来计算。因此，标准加宽算子完全可以通过线性规划进行蕴含检查来实现。

定理 3.1 $\forall \varphi_1 \in P_1, \varphi_2 \in P_2,\ \gamma(P_1) = \gamma((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\})$ 当且仅当 $P_1 \models \varphi_2$ 且 $((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \models \varphi_1$.

证明. 1) 假设 $\gamma(P_1) = \gamma((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\})$：因为 $\varphi_1 \in P_1$，则有 $P_1 \models \varphi_1$，从而有 $(P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\} \models \varphi_1$；另一方面，因为 $\varphi_2 \in (P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}$，则有 $((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \models \varphi_2$，从而有 $P_1 \models \varphi_2$。因此，$P_1 \models \varphi_2$ 且 $((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \models \varphi_1$ 成立。

2) 假设 $((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \models \varphi_1$ 且 $P_1 \models \varphi_2$：因为 $\gamma(P_1) \subseteq \gamma(P_1 \setminus \{\varphi_1\})$ 且 $P_1 \models \varphi_2$，则有 $\gamma(P_1) \subseteq \gamma(P_1 \setminus \{\varphi_1\} \cup \{\varphi_2\})$；另一方面，因为 $\gamma((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \subseteq \gamma(P_1 \setminus \{\varphi_1\})$ 且 $(P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\} \models \varphi_1$，则有 $\gamma((P_1 \setminus \{\varphi_1\}) \cup \{\varphi_2\}) \subseteq \gamma(P_1)$。因此，$\gamma(P_1) = \gamma(P_1 \setminus \{\varphi_1\} \cup \{\varphi_2\})$ 成立。 □

3.3 基于约束的浮点多面体域

本节介绍基于约束的浮点多面体域的设计和实现。首先，我们给出浮点多面体域的域表示方法，然后给出浮点多面体域中域操作的可靠浮点实现方法。从第 3.2 节，我们不难看出，基于约束的多面体抽象域中域操作的实现其实可以归结到两个原子操作：

- Fourier-Motzkin 消除法
- 线性规划

因此，本节首先将上述两个原子操作可靠地适配到浮点计算上，然后基于这两个原子操作的可靠浮点版本来构造浮点多面体域上的域操作，并讨论浮点多面体域的可靠性。最后，讨论浮点实现给多面体域带来的精度和效率问题，并给出改进策略。

3.3.1 域表示

一个浮点多面体 P 可以表示成一个线性不等式系统 $Ax \leq b$，其中 A 和 b 中数都是浮点数 $\mathbb{F}$。注意，其语义仍然表示有理数值变量 x 上的程序环境集合，即 $\gamma(P) = \{x \in \mathbb{Q}^n \mid Ax \leq b\}$，其中$Ax \leq b$ 仍然是在数学精确算术意义下进行解释（而非在浮点算术语义上）。简单而言，这里与有理数多面体的区别，仅仅在于系数采用浮点数表示。

为了区分浮点算术操作和精确算术操作，我们使用如下表示方法：$\{+, -, \times, /\}$ 表示精确有理算术操作，对应的浮点操作表示成 $\{\oplus_r, \ominus_r, \otimes_r, \oslash_r\}$，其中浮点舍入模式 $r \in \{-\infty, +\infty\}$（$-\infty$：向下舍入；$+\infty$：向上舍入）。在浮点多面体的编程实现上，我们只需要这两种舍入模式。另外，浮点求相反数操作 $\ominus$ 是精确的，不会引入舍入误差。

为了方便地描述多面体域中变量的界信息，我们引入包围盒的概念。

定义 3.3.1 (包围盒) 如果一个多面体 P 的约束表示形如 $\pm x_i \leq c$（称为界约束），其中 c 是一个常量或 $+\infty$，则该多面体 P 是一个盒（*box*）。一个多面体 P 的包围盒（*bounding box*）是包含该多面体的最小的盒，记作 $BB(P)$。

下面，本节所有算法都将在浮点算术中实现。

3.3.2 浮点 Fourier-Motzkin 消除法

Fourier-Motzkin 消除法是基于约束的多面体抽象域实现的原子操作之一。在给出该算法的可靠浮点版本之前，我们先介绍一种线性化技术。

(1) 线性化

定义 3.3.2 给定一个区间线性不等式 $\varphi:(\sum_k [a_k,b_k]\times x_k \le c)$，一个向量 x 称为该区间线性不等式的一个**弱解**，当且仅当对于所有 k 存在某个 $d_k\in[a_k,b_k]$ 使得 $\Sigma_k d_k\times x_k\le c$ 成立。

并且，本文中，我们称一个向量 x 满足区间线性不等式 φ，记作 $x\in\gamma(\varphi)$，如果 x 是区间线性不等式 φ 的一个弱解。

通常意义上的线性不等式可以看作是一种特殊的区间线性不等式，其中所有的系数都是标量（即单点区间）。多面体的约束表示仅支持线性（非区间）不等式，而我们后面将看到浮点多面体域中的某些域操作可能产生区间线性不等式。因此，我们需要把一个区间线性不等式 φ 转化成一个线性不等式。为此，我们把文 [84] 中的（面向表达式的）线性化技术扩展到不等式约束上来。

定义 3.3.3 (线性化算子) 给定区间线性不等式 $\varphi:(\sum_k [a_k,b_k]\times x_k\le c)$，设 $\mathbf{x}:=[\underline{x},\overline{x}]$ 为变量 x 的包围盒。线性化算子定义为

$$\zeta(\varphi,\mathbf{x}) \stackrel{\text{def}}{=} \sum_k d_k\times x_k \le c \oplus_{+\infty} \bigoplus_k {}_{+\infty}\left(\max\{b_k\ominus_{+\infty} d_k,\ d_k\ominus_{+\infty} a_k\}\otimes_{+\infty}|x_k|\right)$$

其中，d_k 为区间 $[a_k,b_k]$ 内的任意浮点数且 $|x_k|=\max\{-\underline{x}_k,\overline{x}_k\}$。

理论上，d_k 可以是区间 $[a_k,b_k]$ 内的任意浮点数。在实际实现时，我们通常选择区间 $[a_k,b_k]$ 的中点 $d_k=(a_k\oplus_r b_k)\oslash_r 2$，因为中点导致最小程度的精度损失。

例 3.3.1 考虑区间线性不等式 $[0,2]x+[1,1]y\le 2$ 及包围盒 $x,y\in[-10,5]$。如果我们选择 $[a_k,b_k]$ 的中点作为 d_k，线性化的结果将是 $x+y\le 12$（因为 $2\oplus_{+\infty}\max\{2\ominus_{+\infty}1,1\ominus_{+\infty}0\}\otimes_{+\infty}10\oplus_{+\infty}\max\{1\ominus_{+\infty}1,1\ominus_{+\infty}1\}\otimes_{+\infty}10=12$）。注意，这里发生了精度损失，比如，点 $(0,12)$ 满足结果不等式 $x+y\le 12$ 但是不满足原来的区间线性不等式 $[0,2]x+[1,1]y\le 2$。

定理 3.2 (线性化算子的可靠性) 给定区间线性不等式 φ 及包围盒 $\mathbf{x}$，$\zeta(\varphi,\mathbf{x})$ 是 φ 的上近似，即 $\mathbf{x}$ 中的满足 φ 的任意点都将满足 $\zeta(\varphi,\mathbf{x})$:

$$\forall x\in\mathbf{x}, x\in\gamma(\varphi)\Rightarrow x\in\gamma(\zeta(\varphi,\mathbf{x}))$$

证明. 对于任意区间线性不等式 $\varphi:\sum_k[a_k,b_k]\times x_k\le c$，有

$$\begin{aligned}
&\textstyle\sum_k [a_k,b_k]\times x_k\le c\\
\iff\ &\textstyle\sum_k (d_k+[a_k-d_k,b_k-d_k])\times x_k\le c\\
\iff\ &\textstyle\sum_k d_k\times x_k\le c+\sum_k[d_k-b_k,d_k-a_k]\times x_k\\
\implies\ &\textstyle\sum_k d_k\times x_k\le (c\oplus_{+\infty}\bigoplus_k{}_{+\infty}(\max\{b_k\ominus_{+\infty}d_k,\ d_k\ominus_{+\infty}a_k\}\otimes_{+\infty}|x_k|)
\end{aligned}$$

因为 $a_k \le d_k \le b_k$ 且 $-|x_k| \le x_k \le |x_k|$ 其中 $|x_k| = \max\{-\underline{x}_k, \overline{x}_k\}$，因此对于每个项 $[d_k - b_k, d_k - a_k] \times x_k$ 有

$$[d_k - b_k, d_k - a_k] \times x_k \le \max\{b_k \ominus_{+\infty} d_k, d_k \ominus_{+\infty} a_k\} \otimes_{+\infty} |x_k|$$

从而有

$$c + \sum_k [d_k - b_k, d_k - a_k] x_k \le c \oplus_{+\infty} \bigoplus_k {}_{+\infty} \left(\max\{b_k \ominus_{+\infty} d_k,\ d_k \ominus_{+\infty} a_k\} \otimes_{+\infty} |x_k|\right)$$

□

注意，即使 $\zeta(\varphi, \mathbf{x})$ 结果不等式右端的值依赖于 $\bigoplus_{+\infty}$ 中求和的顺序，线性化算子仍是可靠的，因为实际上每个计算序给出的都是 $c + \sum_k [d_k - b_k, d_k - a_k] \times x_k$ 在精确算术上的值的上近似。

(2) Fourier-Motzkin 消除法的可靠浮点实现

构造可靠浮点 Fourier-Motzkin 消除法的核心思想是使用向外舍入的区间算术（即，求上界向上舍入、求下界向下舍入）。然后，使用上面引入的线性化算子 ζ，将区间运算结果中的区间线性不等式线性化为标量系数的线性不等式。

下面，考虑从如下两个线性不等式中消除变量 x_i

$$\begin{cases} a_i^+ x_i + \sum_{k \ne i} a_k^+ \times x_k \le c^+ & \text{where } a_i^+ > 0 \\ a_i^- x_i + \sum_{k \ne i} a_k^- \times x_k \le c^- & \text{where } a_i^- < 0 \end{cases} \tag{3.2}$$

使用向外舍入的区间算术把 (3.2) 中的不等式约束分别除以各自所对应的 x_i 系数的绝对值后，得到

$$\begin{cases} x_i + \sum_{k \ne i} [a_k^+ \oslash_{-\infty} a_i^+, a_k^+ \oslash_{+\infty} a_i^+] \times x_k \le c^+ \oslash_{+\infty} a_i^+ \\ -x_i + \sum_{k \ne i} [a_k^- \oslash_{-\infty} (\ominus a_i^-), a_k^- \oslash_{+\infty} (\ominus a_i^-)] \times x_k \le c^- \oslash_{+\infty} (\ominus a_i^-) \end{cases}$$

两不等式相加后得到

$$\begin{aligned} \Sigma_{k \ne i} [(a_k^+ \oslash_{-\infty} a_i^+) \oplus_{-\infty} (a_k^- \oslash_{-\infty} (\ominus a_i^-)), (a_k^+ \oslash_{+\infty} a_i^+) \oplus_{+\infty} (a_k^- \oslash_{+\infty} (\ominus a_i^-))] \times x_k \\ \le (c^+ \oslash_{+\infty} a_i^+) \oplus_{+\infty} (c^- \oslash_{+\infty} (\ominus a_i^-)) \end{aligned} \tag{3.3}$$

然后，对不等式 (3.3) 应用线性化算子 ζ 就可以抽象得到一个线性（非区间）不等式。我们用 $FME_f(P, x_i)$ 来表示用上述浮点算法从 P 中消除 x_i 所得到的结果线性不等式系统。

定理 3.3 (浮点 Fourier-Motzkin 消除法的可靠性) 给定一个多面体 P、一个待消除变量 x_i 及所有变量 x 的包围盒 $\mathbf{x}$，$\mathbf{x}$ 中的满足 $FME(P, x_i)$ 的所有点也会满足 $FME_f(P, x_i)$:

$$\forall x \in \mathbf{x}, x \in \gamma(FME(P, x_i)) \Rightarrow x \in \gamma(FME_f(P, x_i))$$

浮点 Fourier-Motzkin 消除法可行的关键点在于待消除变量的系数始终能够通过除法精确地归结为 1 或 -1。对于某些情形，也存在另外一种候选算法。假设 $a_i^+ \otimes_{-\infty} (\ominus a_i^-) = a_i^+ \otimes_{+\infty} (\ominus a_i^-)$ 成立，即 a_i^+ 和 $\ominus a_i^-$ 的浮点乘是精确的。此时，Fourier-Motzkin 消除法可以通过一种乘法方式来实现：

$$\begin{aligned}\Sigma_{k\neq i}[(a_k^+ \otimes_{-\infty} (\ominus a_i^-)) \oplus_{-\infty} (a_k^- \otimes_{-\infty} a_i^+), (a_k^+ \otimes_{+\infty} (\ominus a_i^-)) \oplus_{+\infty} (a_k^- \otimes_{+\infty} a_i^+)] \times x_k \\ \leq (c^+ \otimes_{+\infty} (\ominus a_i^-)) \oplus_{+\infty} (c^- \otimes_{+\infty} a_i^+)\end{aligned} \tag{3.4}$$

注意，条件 $a_i^+ \otimes_{-\infty} (\ominus a_i^-) = a_i^+ \otimes_{+\infty} (\ominus a_i^-)$ 保证了 (3.4) 中 x_i 的系数将精确地为 0。尤其，当 (3.2) 中所有的系数为小整数时，(3.4) 方法常常会给出精确的结果，但是 (3.3) 方法却往往不能。实际程序分析中，乘法方式的 Fourier-Motzkin 消除对于形成有着规整系数的不等式约束很有意义，特别是对于只包含整数变量的程序。

例 3.3.2 考虑两个线性不等式 $3x + y \leq 10$ 和 $-7x + y \leq 10$，并设变量的包围盒为 $x, y \in (-\infty, 10]$。消除变量 x 后，*(3.4)* 方法将得到结果 $y \leq 10$，然而 *(3.3)* 方法将得到结果 $y \leq +\infty$。

3.3.3 严格线性规划

线性规划（LP）是最优化问题中研究较早、发展较快、应用广泛、方法较成熟的一个重要分支。目前，已经有许多高效的、可扩展到上千上万规模变量和约束的线性规划算法和工具。但是，大部分 state-of-the-art LP 求解器都是使用浮点算术并只能给出近似解。而且，该近似解可能不是实际最优解，甚至可能在可行空间之外。为了得到正确的解，一种方法就是使用精确算术来计算精确解，但是这在实际中代价太高，可扩展性受到极大地限制。

本文中，线性规划是基于约束的多面体抽象域实现的原子操作之一。而且，本文希望所有算法均采用浮点实现，且需要保证实现的可靠性。为此，本文利用线性规划领域最近在严格线性规划技术方面取得的进展，采用计算代价不高、基于浮点计算的线性规划技术来计算 LP 目标函数的严格界。

最近，Neumaier 与 Shcherbina 在文 [107] 中给出了通过在对偶问题上应用浮点线性规划来求解原问题上目标函数的严格界的方法，从而使得仅使用浮点计算就可以求得线性规划问题目标值的可靠近似（不小于精确极大目标值或不大于精确极小目标值）。简而言之，浮点线性规划中目标函数的严格界可以通过对普通浮点 LP 求解器给出的近似结果做小代价的后处理来得到。

具体而言，对于如下形式的线性规划问题

$$\begin{aligned} min \quad & c^T x \\ s.t. \quad & Ax \le b \end{aligned}$$

其对偶问题为

$$\begin{aligned} max \quad & b^T y \\ s.t. \quad & A^T y = c, \ \ y \le 0 \end{aligned}$$

假设 y 是对偶问题的浮点近似解，我们使用向外舍入的区间算术计算 $A^T y$ 与 c 的差 r。注意，如果 y 是精确解且 $A^T y - c$ 采用精确算术来计算，则 r 必然为 0。但是，因为这里 y 是浮点近似解并且我们采用区间算术来计算 r，因此所得到的将是区间 $\mathbf{r}$：

$$r := A^T y - c \in \mathbf{r} = [\underline{r}, \overline{r}].$$

由于 $y \le 0$ 以及 $Ax \le b$，因此有 $y^T Ax \ge y^T b$。设所有变量 x 的包围盒为 $\mathbf{x} := [\underline{x}, \overline{x}]$，那么有

$$c^T x = (A^T y - r)^T x = y^T Ax - r^T x \ge y^T b - r^T x \in y^T b - \mathbf{r}^T \mathbf{x}$$

从而

$$\mu := \inf(y^T b - \mathbf{r}^T \mathbf{x}) \tag{3.5}$$

即是 $c^T x$ 的一个严格下界。换而言之，原线性规划问题的目标函数 $c^T x$ 的精确最小解，必然不小于 μ。(3.5) 式中 μ 的值可以通过如下浮点算法来计算：

roundown;
$\underline{r} = A^T y - c$;
$t = y^T b$;
roundup;
$\overline{r} = A^T y - c$;
$\mu = \max\{\ \underline{r}^T \underline{x},\ \underline{r}^T \overline{x},\ \overline{r}^T \underline{x},\ \overline{r}^T \overline{x}\ \} - t$;
$\mu = -\mu$;

注意，(3.5) 式给出的严格下界的精确程度依赖于包围盒 $\mathbf{x}$ 的范围。另外，为最大目标函数值寻找严格上界可以转换为求最小目标函数值的情形：

$$\max c^T x = -\min\ (-c)^T x$$

3.3.4 浮点多面体域的可靠性

从第 3.2 节，我们知道，基于约束的多面体抽象域中域操作的实现可以归结到两个原子操作：Fourier-Motzkin 消除法和线性规划。在第 3.3.2 节中，我们给出了 Fourier-Motzkin 消除法的一个可靠浮点实现方法。在第 3.3.3 节中，我们采用严格线性规划作为线性规划的一个可靠浮点实现方法。

因此，在基于约束的多面体抽象域中，把其两个原子操作，即 Fourier-Motzkin 消除法和线性规划，分别替换成对应的可靠浮点实现方法：

- 浮点 Fourier-Motzkin 消除法（第 3.3.2 节）
- 严格线性规划（第 3.3.3 节）

我们就可以构造出浮点多面体域上的域操作。注意，在实现时，这两个原子操作都可能产生浮点上溢或者 NaN (Not a Number) 值。在这种情况下，浮点 Fourier-Motzkin 消除法可以通过丢弃该约束来保证可靠性；对于严格线性规划，则可以通过返回 $+\infty$ $(-\infty)$ 作为严格最大（最小）目标值来保证可靠性。

整个浮点多面体抽象域的可靠性可通过每个域操作的可靠性得以保证：每个基于浮点实现的域操作应该总输出比基于多精度有理数实现的域操作更保守的结果。根据输出类型，多面体域上的域操作可以分为两大类：输出一个多面体（如投影、求交、求接合、冗余约束消除等）或者给出一个为真为假的判断（如包含测试、空多面体测试、蕴含测试等）。

首先，因为浮点 Fourier-Motzkin 消除法的可靠性（见定理 3.3），浮点投影操作将总是返回一个相对于精确投影操作的可靠的上近似多面体。这也保证了接合（凸闭包）操作和赋值迁移操作的可靠性。而冗余约束消除和条件测试迁移操作的可靠性则显而易见。

另一方面，对于一个不等式约束关于一个多面体的蕴含检查操作，是通过严格线性规划来实现的，肯定答案表示该不等式约束在实际上确实被这个多面体所蕴含，而否定答案则表示无法确定该不等式约束是否在实际上被这个多面体所蕴含。例如，如果一个不等式约束在实际上是被一个多面体蕴含，但是如果在几何上该不等式约束离多面体比较近甚至与该多面体有交点，严格线性规划可能因为给出一个太保守的目标值而不能断言其蕴含关系。因此，浮点多面体域中的包含测试实际上

会输出“true” 或“don’t know”。但是，这种近似不会改变基于浮点多面体域的静态分析的整体可靠性，只是可能增加迭代次数。对于基于浮点实现的空多面体测试的增量式实现方法，也有类似断言。另外，浮点多面体抽象域的加宽算子可能因为使用了严格线性规划技术，相对于基于精确实现的加宽算子保留更少的不等式约束，但这也是可靠的，因为得到的是一个上近似多面体。

3.3.5 精度和效率改进策略

正如第 3.3.4 节所说，浮点多面体域的每个域操作结果总是有理数多面体域上对应操作结果的上近似。从另外一个角度，这说明了每个浮点域操作都可能伴随着部分的精度损失。此外，严格线性规划所返回的保守值也可能会导致浮点多面体域计算效率的下降。比如，基于严格线性规划的冗余约束消除操作可能不能消除 Fourier-Motzkin 消除法所产生的冗余约束，使得多面体中的约束越来越多，从而降低计算性能。本节将从实践角度针对这些问题提出一些改进策略，以在保持可靠性的前提下提高浮点多面体域的分析精度和计算效率。

(1) 界缩紧

变量的界信息，即变量的包围盒，在浮点多面体抽象域中起着重要作用。比如，线性化和严格线性规划的结果都依赖于包围盒，因此变量的界越精确，这些操作所带来的精度损失将越少。变量的界会随着多面体的操作而改变，比如当向多面体加入新的约束时，变量的界就需要更新。本文采用多种策略来对变量的界进行缩紧以求得更为精确的界信息。

严格线性规划. 最简单的方法用来获得一个多面体 P 的界信息，是通过严格线性规划来计算

$$\max\,(\min)\ x_k \text{ subject to } P$$

从而得到变量 x_k 的严格上（下）界。然而，因为严格线性规划本身依赖于变量的包围盒的范围，严格线性规划给出的结果可能太保守，特别是当某些变量的界很大甚至在加宽操作后丢失（变成 $\pm\infty$）时。另外，在每次作用于 n 维多面体上的域操作之后都运行 $2n$ 个线性规划问题来计算所有变量的上下界，计算代价也较高。因此，我们需要某些轻量级方法来进行界缩紧。

界传播. 界传播是约束规划领域常用的一种约束传播技术。多面体的线性不等式系统中的每个不等式都可以用来缩紧那些在该不等式中出现的变量的界。给定一个不

等式 $\sum_i a_i x_i \leq b$，如果 $a_i > 0$，那么可以得到 x_i 的一个新的候选上界 ν：

$$x_i \leq \nu = (b - \sum_{j \neq i} a_j x_j)/a_i$$

在实际中，ν 的一个上近似可以通过向外舍入的区间算术来计算。同样地，如果 $a_i < 0$，我们可以找到一个新的候选下界。如果新的界更紧，x_i 的界就得以更新。

策略结合. 事实上，上述两种界缩紧方法是相互补充的。在某些情况下，每种方法都有可能比另一种方法找到更紧的界信息。

例 3.3.3 给定多面体$\{-x+3y \leq 0, x-6y \leq -3\}$及初始界信息 $x, y \in (-\infty, +\infty)$，此时界传播不能找到任何更紧的界信息，而严格线性规划则可为 x 找到一个更紧的界 $x \in [3, +\infty)$，但是不能为 y 找到更紧的界信息。接下来，如果在 $\{-x+3y \leq 0, x-6y \leq -3\}$ 及界 $x \in [3, +\infty), y \in (-\infty, +\infty)$ 上应用界传播技术，我们就可以得到 $x \in [3, +\infty), y \in [1, +\infty)$。而且，这就是该多面体的精确的界信息。

因此，我们需要将上述策略结合起来，并在计算代价和精度之间寻找合适的平衡点。例如，我们可以只使用严格线性规划来缩紧那些在约束系统中出现频率比较高的变量的界，然后使用界传播来缩紧其他变量的界。注意，严格线性规划和界传播对于所考虑的变量的序是敏感的。更为精确的方法是将严格线性规划和界传播结合起来并进行迭代，但是计算开销也会更大。

(2) 凸闭包缩紧

凸闭包计算是浮点多面体抽象域中计算复杂度最高的部分，也是精度损失较严重的地方。凸闭包计算内部都是通过浮点 Fourier-Motzkin 消除法来实现的，在连续消除多个变量的过程中，可能导致大量精度损失。在许多情况下，可以利用一些启发式信息比如凸包络和界信息等来提高精度。

凸包络. 从构成上看，两个多面体的凸闭包的约束表示可按来源分为两部分：

- 旧关系型约束：这些约束在凸闭包的输入参数多面体的约束表示中已出现过；
- 新关系型约束：这些约束在凸闭包的输入参数多面体的约束表示中并没有出现，而是在凸闭包计算过程中新产生的。

在图 3.4 中所示的凸闭包中，用粗线标识的 $\{-4x-y \leq 34^{(1)}, -2x+3y \leq 24^{(2)}, -x+6y \leq 41^{(11)}, 2x+3y \leq 53^{(12)}, 2x+y \leq 39^{(13)}, 2x-y \leq 33^{(14)}, 3x-4y \leq 52^{(15)}, x-4y \leq 28^{(16)}\}$ 是旧关系型约束，而 $\{-4x+13y \leq 76^{(17)}, -3x-16y \leq 56^{(18)}\}$ 则是新关系型约束。

定义 3.3.4 (凸包络) 给定两个多面体 P_1 和 P_2，其凸包络（*Envelope*）定义为

$$env(P_1, P_2) \stackrel{\text{def}}{=} \mathcal{S}_1 \cup \mathcal{S}_2$$

其中

$$\mathcal{S}_1 = \{\ \varphi_1 \in P_1 \mid P_2 \models \varphi_1\ \},$$
$$\mathcal{S}_2 = \{\ \varphi_2 \in P_2 \mid P_1 \models \varphi_2\ \}.$$

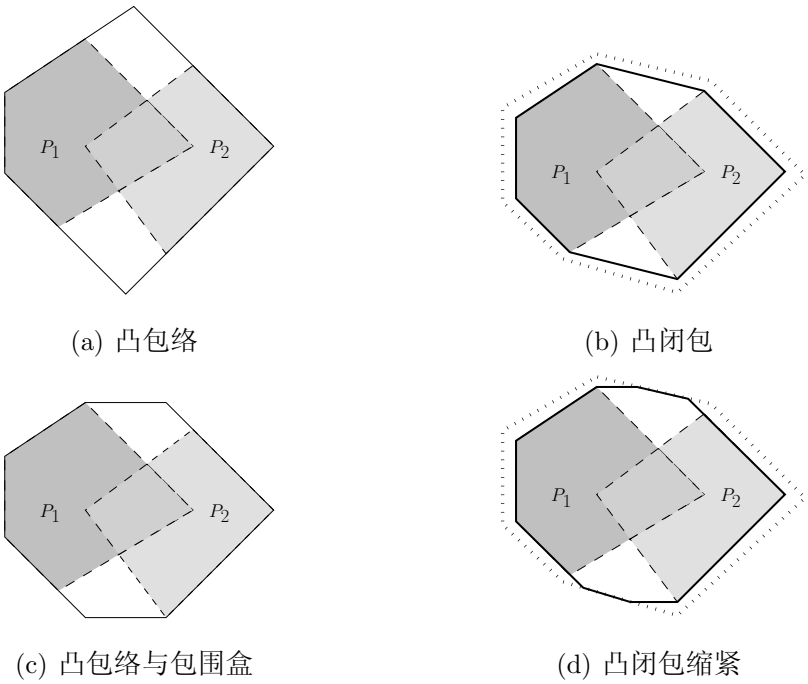
不难看出，凸包络是凸闭包的上近似，即 $P_1 \sqcup_{CH} P_2 \sqsubseteq env(P_1, P_2)$。事实上，两个多面体的凸包络构成了这两个多面体凸闭包中的旧关系型约束，因此凸包络中的约束必然是凸闭包的约束的一部分。在图 3.4 中所示的例子中，$env(P_1, P_2) = \{-4x - y \leq 34^{(1)}, -2x + 3y \leq 24^{(2)}, -x + 6y \leq 41^{(11)}, 2x + 3y \leq 53^{(12)}, 2x + y \leq 39^{(13)}, 2x - y \leq 33^{(14)}, 3x - 4y \leq 52^{(15)}, x - 4y \leq 28^{(16)}\}$（图 3.4 中用粗线标识）。

换而言之，凸包络中所有的不等式约束都可以可靠地加入到最终的凸闭包中。值得注意的是，两个多面体凸包络的计算只需简单的蕴含检查，可以在凸闭包计算之前低代价地确定下来，即通过 $(m_1 + m_2)$ 个线性规划求解的代价计算得到，其中 m_1 和 m_2 分别为 P_1 和 P_2 约束表示中不等式的数目。在浮点多面体域中，可以使用严格线性规划技术计算出凸包络，然后把凸包络中的约束添加入到浮点凸闭包的计算结果中来缩紧原来的浮点凸闭包结果。

包围盒. 给定两个盒 $B_1 = \{a \leq x \leq b\}$ 和 $B_2 = \{c \leq x \leq d\}$，则这两个盒在区间抽象域上的接合定义为 $B_1 \sqcup_i B_2 = \{\min(a, c) \leq x \leq \max(b, d)\}$。且有如下性质：两个多面体 P_1 和 P_2 的包围盒在区间抽象域上的接合所得到的盒，等价于这两个多面体凸闭包的包围盒，即 $BB(P_1 \sqcup_{CH} P_2) = BB(P_1) \sqcup_i BB(P_2)$。在图 3.4 中所示的例子中，$BB(P_1) = \{-9 \leq x \leq -1, -2 \leq y \leq 4\}$，$BB(P_2) = \{2 \leq x \leq 18, -5 \leq y \leq 9\}$，$BB(P_1 \sqcup_{CH} P_2) = BB(P_1) \sqcup_i BB(P_2) = \{-9 \leq x \leq 18, -5 \leq y \leq 9\}$。

根据上述性质，两个多面体凸闭包的包围盒的计算也可以在凸闭包计算之前以较低的代价确定下来，即通过计算 $BB(P_1) \sqcup_i BB(P_2)$ 得到。在浮点多面体抽象域中，会为每个变量维护并及时更新其上下界信息。因此，$BB(P_1) \sqcup_i BB(P_2)$ 可以很容易计算出来。并且，如果两个多面体的包围盒是精确的，那么两个多面体凸闭包的包围盒在浮点实现上也可以精确得到，因为 $\min / \max$ 操作对于浮点数也是精确的。因此，我们可以把这种方式计算得到的两多面体凸闭包包围盒中的约束添加到浮点凸闭包中，来缩紧原来的浮点凸闭包结果。

缩紧. 我们把所有来自凸包络和包围盒的约束都添加到浮点凸闭包计算结果中，来缩紧浮点凸闭包，以提高精度。这一过程如图 3.5 所示，其中 (a) 子图给出了P_1 与



(a) 凸包络　(b) 凸闭包

(c) 凸包络与包围盒　(d) 凸闭包缩紧

图 3.5 浮点凸闭包缩紧

P_2 的凸包络（实线）；(b) 子图给出了精确凸闭包（实粗线）和一个可能的近似浮点凸闭包（点线）；(c) 子图给出了由凸包络和包围盒所定义最小多面体（实线）；(d) 子图给出了浮点凸闭包（点线）和经凸包络和包围盒缩紧后的结果（实粗线）。

注意，缩紧后的浮点凸闭包依然保持可靠性。在实际程序分析中，把来自凸包络和包围盒的约束在浮点计算过程中保持起来具有很重要的实际意义，因为在加宽操作时这类约束往往在稳定约束中占较大比例。

(3) 启发式线性化

在基于多面体抽象域的分析中，变量之间的新关系往往是源自凸闭包计算。实际上，这些变量的系数在很大程度上取决于在线性化过程中所作出的选择，尤其是定义 3.3.3 中关于 d_k 的选择。在第 3.3.2 节中，我们建议使用区间中点 $d_k = (a_k \oplus_r b_k) \oslash_r 2$。这是一种局部贪婪策略，因为它使得结果约束中的常量项最小，即带来的精度损失最小。然而，考虑到程序分析的全局过程，选择一个更规整的数值如一个整数，对于提高后续计算（如线性规划求解）的效率和数值稳定性有着重要意义。此外，在同一个程序点处相邻两次的迭代过程中，如果采用精确计算则有可能得到同样的结果，但是采用浮点计算则有可能由于舍入误差的存在导致微小扰动从而得到稍微不同的结果。因此，在基于浮点实现的程序分析过程中，一般要求即使输入区间 $[a_k, b_k]$

上发生微小的抖动，线性化也应该尽可能选择同样的 d_k。这一点对于使用基于浮点实现的抽象域来寻找循环结构中稳定的不变式尤为重要。

在实践中，本文使用两种策略：把中点舍入到最接近的整数或者重用同一个约束中已为另一个变量做好选择的系数。当然，还可以设计其他策略。事实上，只需要对定义 3.3.3 中的公式稍作调整，还可以选择 $[a_k, b_k]$ 之外的 d_k。

(4) 有效冗余约束消除

在之前的浮点多面体抽象域中，多面体约束表示中的冗余约束是通过严格线性规划技术来实现的。但是，严格线性规划可能给出过于保守的目标值的严格界，使得无法检测出一些冗余约束。这可能大大削弱浮点多面体抽象域的处理能力。尤其凸闭包计算中长序列的 Fourier-Motzkin 变量消除可能引发冗余约束的组合爆炸问题，如果没有有效的冗余约束消除方法，多面体中的约束数可能呈指数级增长，使得抽象域的性能极大地降低。

但是，值得注意的是冗余约束消除操作始终是可靠的，即使一些非冗余的约束被误删了，这种情况下，将得到一个上近似的多面体。为了尽可能消除冗余约束，一种方法是采用普通线性规划，而不是严格线性规划。甚至可以更进一步，对于 P 中不等式 $\varphi : (\Sigma_i a_i x_i \leq b)$，采用普通浮点线性规划计算 $\mu = \max \Sigma_i a_i x_i \text{ subject to } P \setminus \{\varphi\}$，只要 $\mu < (1+\epsilon)b$ 就可以考虑从 P 中删除 φ，其中 $\epsilon > 0$ 为预设定的允许偏差。

为了使得冗余约束消除的计算效率更高，可以首先使用一些轻量级的冗余约束消除方法，然后只在需要的时候才使用高代价的基于线性规划的方法。首先，本文使用基于语法检查的方法：对于同一个多面体中的一对不等式 $\Sigma_i a_i x_i \leq b$ 和 $\Sigma_i a'_i x_i \leq b'$，如果 $\forall i. a'_i = a_i$，则只需要保留不等式右端常量项较小的那一个不等式。其二，我们在检测不等式在多面体中是否冗余之前，先检查该不等式与多面体的包围盒之间的关系，如果该不等式对于包围盒是冗余的，那么对于多面体更是冗余的。最后，本文还采用文 [162, 163] 中的方法来缓解 Fourier-Motzkin 变量消除序列（如，在凸闭包计算中）可能引发的冗余约束的组合爆炸问题。

3.4 基于约束的多面体域的弱接合

在基于约束的多面体抽象域中，其处理能力主要受限于其高代价的（强）接合操作，即两多面体的凸闭包计算。基于约束的凸闭包计算是通过转化为 Fourier-Motzkin 变量消除来实现的，而 Fourier-Motzkin 变量消除序列一般会产生大量甚至

指数级的冗余约束，从而可能引发冗余约束的组合爆炸问题 [163]。正如文 [82] 中所述，基于约束的凸闭包计算方法适合于低维稀疏约束系统。但对于高维的或稠密的约束系统，由于计算过程中会产生大量冗余约束，该方法的执行效率将受到很大影响。线性约束系统中的冗余约束通常需要采用线性规划技术逐一消除，而在凸闭包计算过程中这种可能出现的大规模的线性规划问题将会极大地降低线性规划求解器的执行效率，甚至导致线性规划求解器出现“数值不稳定性”（numerical instability）问题而不能找到最优解。因此，基于约束的接合操作成为基于约束的多面体域的主要计算瓶颈，极大地制约了其执行效率和可扩展性。

Sankaranarayanan 等人 [91, 164] 称其模版多面体域中的接合操作为弱接合（weak join），以区别传统通用多面体域中的（强）接合即凸闭包。但是，模版多面体域的主要缺陷在于该域只能发现预先设计好的固定形式的变量间线性关系（称为模版），而不能发现其他的、新的约束关系。与之不同的是，本文将面向通用多面体域。

本节为基于约束的多面体抽象域设计弱接合操作，在不损失太多精度的前提下提高接合操作的执行效率、可扩展性和易处理性。其基本思想是：结合多面体凸闭包的几何性质，利用包络和界等易于计算的启发式信息，把稠密的、复杂的多面体约束表示稀疏化、简单化，通过低代价的弱接合操作求得凸闭包的上近似。为了能在程序分析的计算效率与精度之间取得合理权衡，本节还提出一种启发式策略，把强、弱接合动态地、有机地结合起来进行程序分析。

3.4.1 基于模版的弱接合

类似模版多面体域 [91] 的思想，如果用户能够提供一些模版（template）约束，则可以把这些模版约束作为弱接合操作结果多面体中约束的候选，并通过线性规划求得凸闭包的上近似。对于多面体抽象域，其模版约束形如 $\Sigma_i a_i x_i \leq c$，其中变量 x_i 的系数 a_i 是一个固定的常数，常量项 c 是一个可变的参数，如 $2x + 3y \leq c$。给定两个多面体 P_1 和 P_2，对于模版 $\Sigma_i a_i x_i \leq c$，通过求解如下线性规划问题：$v_1 = \max \Sigma_i a_i x_i$ subject to P_1 及 $v_2 = \max \Sigma_i a_i x_i$ subject to P_2，我们可以把约束 $\Sigma_i a_i x_i \leq \max(v_1, v_2)$ 加入到弱接合结果中。

如果用户不能提供模版约束，本文的策略是选择两个多面体P_1、P_2 中约束较少的那个多面体中的约束作为模版约束。这里，不妨设 P_1 中的约束较少，不等式 $\Sigma_i a_i x_i \leq c_1$ 是 P_1 中的任意约束。求解线性规划问题：$v = \max \Sigma_i a_i x_i$ subject to P_2，如果 $v \neq +\infty$，则把约束 $\Sigma_i a_i x_i \leq \max(c_1, v)$ 加入到弱接合结果中。考虑到界约束的低计算代价及其在程序值范围分析中的重要性，本文把 $\pm x_i \leq c$ 也作为模版约

束，并通过 $BB(P_1) \sqcup_i BB(P_2)$ 计算得到。

基于模版的弱接合的优点在于其计算代价低，并能保证结果约束系统中的约束不会太多。尤其，当参与接合操作的两个多面体中，一个很简单、约束较少，而另一个很复杂、约束很多时，基于模版的弱接合能得到简单的结果，有利于后续分析。

对于图 3.4 中所示接合实例，图 3.6 给出了基于模版的弱接合的结果。

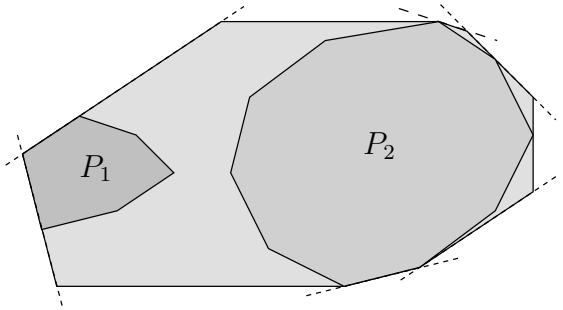


图 3.6 基于模版的弱接合

3.4.2 基于包络和界信息的弱接合

给定两个多面体 P_1 和 P_2，定义 $EEBB(P_1, P_2) \stackrel{\text{def}}{=} env(P_1, P_2) \sqcap (BB(P_1) \sqcup_i BB(P_2))$，则 $EEBB(P_1, P_2)$ 是由 P_1 和 P_2 的包络及其凸闭包之包围盒所能确定的最小多面体。不难看出，$EEBB(P_1, P_2)$ 是 P_1 和 P_2 凸闭包的上近似。$EEBB(P_1, P_2)$ 精确地保留了两个多面体的包络和界信息，但是不能发现新关系型约束。

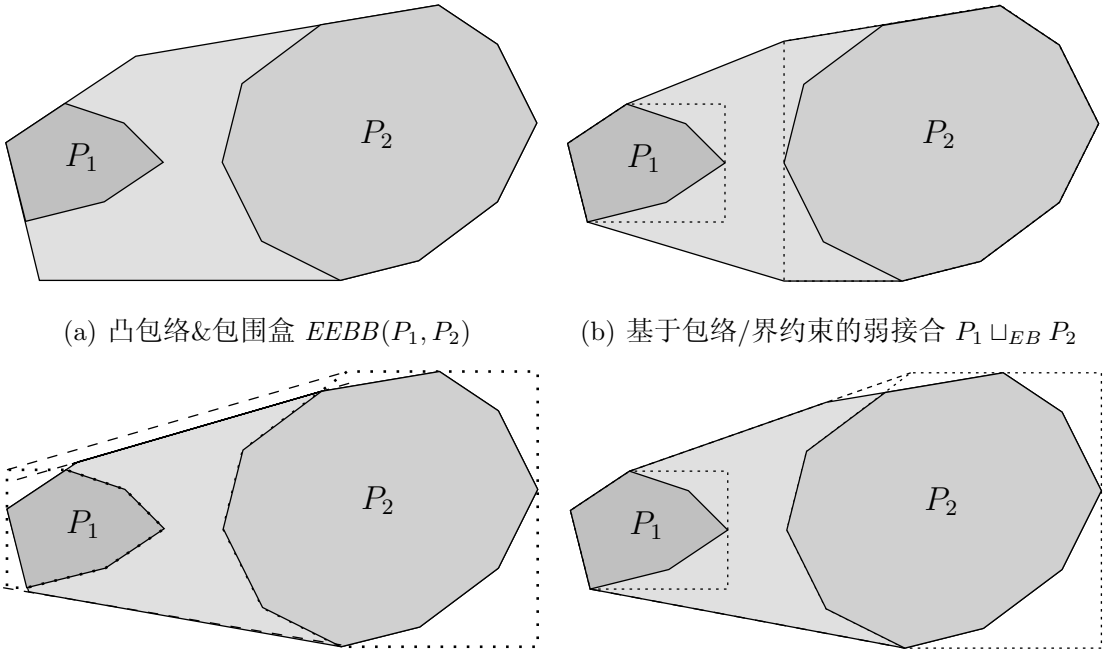
为了能够产生和逼近凸闭包中的新关系型约束，本文基于包络和界信息定义如下弱接合操作：

- 设 $i \in \{1, 2\}$，记 $EB(P_i)$ 为由 P_i 中的包络约束及 P_i 的包围盒所确定的最小多面体。定义弱接合操作：$P_1 \sqcup_{EB} P_2 \stackrel{\text{def}}{=} (EB(P_1) \sqcup_{CH} EB(P_2)) \sqcap EEBB(P_1, P_2)$。
- 对偶地，记 $NEB(P_i)$ 为由 P_i 中的非包络约束及 P_i 的包围盒所确定的最小多面体。定义弱接合操作：$P_1 \sqcup_{NEB} P_2 \stackrel{\text{def}}{=} (NEB(P_1) \sqcup_{CH} NEB(P_2)) \sqcap EEBB(P_1, P_2)$。
- 根据参与凸闭包计算的每个输入多面体中的包络约束数和非包络约束数，启发式地确定其上近似。如果该多面体的包络约束少于非包络约束，则使用包络约束及界约束来作为该多面体的上近似，否则使用非包络约束及界约束来作为该多面体的上近似。记使用该策略为输入多面体 P_i 确定的上近似为 $XB(P_i)$，定义弱接合操作：$P_1 \sqcup_{XB} P_2 = (XB(P_1) \sqcup_{CH} XB(P_2)) \sqcap EEBB(P_1, P_2)$。

显然，$EB(P_i), NEB(P_i), XB(P_i)$ 都是 P_i 的上近似。且 $P_1 \sqcup_{EB} P_2, P_1 \sqcup_{NEB} P_2, P_1 \sqcup_{XB} P_2$ 都是 P_1 与 P_2 的凸闭包 $P_1 \sqcup_{CH} P_2$ 的上近似。通常，由于 $EB(P_i)$,

$NEB(P_i), XB(P_i)$ 的约束数都比 P_i 少，也比 P_i 稀疏，因而 $P_1 \sqcup_{EB} P_2, P_1 \sqcup_{NEB} P_2, P_1 \sqcup_{XB} P_2$ 的计算效率和鲁棒性都会优于 $P_1 \sqcup_{CH} P_2$。为了进一步提高精度，我们把 $P_1 \sqcup_{EB} P_2, P_1 \sqcup_{NEB} P_2, P_1 \sqcup_{XB} P_2$ 执行产生的新关系型约束（未在 P_1, P_2 中出现的非界约束）作为模版约束，采用基于模版的弱接合的思想对弱接合结果进行缩紧。这些弱接合的优点在于他们能够产生新关系型约束，且结果的精度相对较高。

对于图 3.4 中所示接合实例，图 3.7 给出了 $EEBB(P_1, P_2)$ 及采用模版缩紧技术后 $P_1 \sqcup_{EB} P_2, P_1 \sqcup_{NEB} P_2, P_1 \sqcup_{XB} P_2$ 的结果。



(a) 凸包络&包围盒 $EEBB(P_1, P_2)$　(b) 基于包络/界约束的弱接合 $P_1 \sqcup_{EB} P_2$

(c) 基于非包络/界约束的弱接合$P_1 \sqcup_{NEB} P_2$　(d) 基于(非)包络/界约束的弱接合$P_1 \sqcup_{XB} P_2$

图 3.7 基于凸包络和界信息的弱接合

3.4.3 基于两变量约束的弱接合

类似于八边形抽象域 [41] 和每不等式两变量抽象域 [90] 的思想，本文采用两变量约束的动机基于如下观察：在实际程序分析中，最多只涉及两变量的不变式通常占有意义的、用户感兴趣的不变式中的大多数，在许多情况下这种不变式足以分析和验证用户所关注的程序性质。本文基于两变量约束的弱接合的主要目标是产生合适的两变量约束作为上近似来逼近凸闭包。

定理 3.4 给定两个盒 B 和 B', B 和 B' 的凸闭包多面体的约束表示中每个不等式约束最多只涉及两个变量, 即两个盒的凸闭包多面体可以精确地在每不等式两变量抽象域中计算得到。

该定理可以通过第 3.2.2 节所给出的多面体凸闭包计算公式 (3.1) 来证明。

根据定理 3.4，两个盒 B 和 B' 的凸闭包可以在每不等式两变量抽象域中计算得到。考虑任意两个变量 x_i 和 x_j，在 x_i-x_j 平面上，盒 B 与 B' 各自定义了一个矩形（可能不限界）。在 x_i-x_j 平面上，两个矩形的凸闭包将由一些形如 $\pm x_i \leq c$ 的单变量约束（即界约束，对应 x_i, x_j 变量的最小下界和最大上界），以及一些形如 $ax_i + bx_j \leq c$ 的两变量约束构成。几何上，这些两变量约束可以通过计算连接两矩形顶点的直线所确定的半平面得到。不管这两个矩形的布局如何，其凸闭包最多产生 4 个两变量约束，分别连接两个矩形各自的左上与左上、左下与左下、右上与右上、右下与右下顶点，如图 3.8 所示。

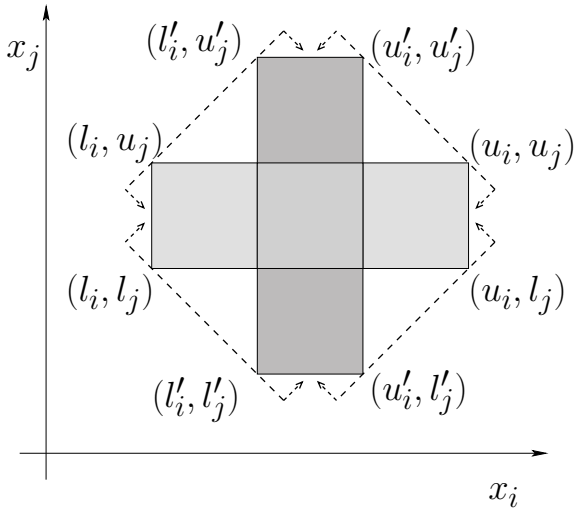


图 3.8 盒的多面体凸闭包

对于两个 n 维盒，其凸闭包最终将至多产生 $2n$ 个界约束（单变量约束）。由于在每个 x_i-x_j 平面，二维凸闭包算法至多产生 4 个两变量约束，因此在 n 维上总共至多产生 $2n(n-1)$ 个两变量约束。因此，盒的多面体凸闭包算法最终将至多产生 $2n^2$ 个不等式约束。

下面，定义基于两变量约束的弱接合操作：$P_1 \sqcup_{TVPI} P_2 \stackrel{\text{def}}{=} (BB(P_1) \sqcup_{CH} BB(P_2)) \sqcap EEBB(P_1, P_2)$。根据定理 3.4，两个多面体的包围盒的凸闭包即 $BB(P_1) \sqcup_{CH} BB(P_2)$，可以在每不等式两变量抽象域上计算得到，并且其计算代价低。显然，$P_1 \sqcup_{TVPI} P_2$ 是 $P_1 \sqcup_{CH} P_2$ 的上近似。

类似地，把 $BB(P_1) \sqcup_{CH} BB(P_2)$ 得到的两变量约束作为模版约束，采用基于模版的弱接合的思想对凸闭包结果进行缩紧可以提高精度。

基于两变量约束的弱接合操作的优点在于它能够避免凸闭包计算过程中的大

量的冗余约束消除计算以及凸闭包计算潜在的指数级输出，保证接合结果中的约束数在 $O(n^2)$ 数量级内，并能够产生程序分析中用户感兴趣的两变量约束。

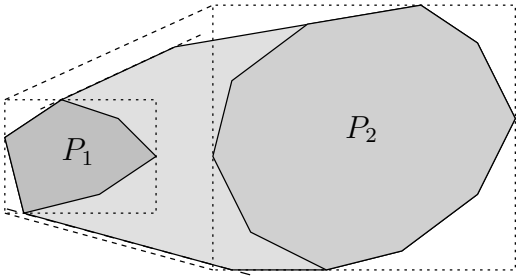


图 3.9 基于两变量约束的弱接合

对于图 3.4 中所示的接合实例，图 3.9 给出了采用模版缩紧技术后 $P_1 \sqcup_{TVPI} P_2$ 的结果。

3.4.4 启发式结合策略

第 3.4.1 节至第 3.4.3 节提出的弱接合操作，统一记作 $\sqcup_w$，均满足 $P_1 \sqcup_{CH} P_2 \sqsubseteq P_1 \sqcup_w P_2$，即弱接合所得多面体均是强接合（凸闭包）所得多面体的一个上近似，因此这些弱接合操作都是可靠的。相对强接合，弱接合执行效率高但存在一定程度的精度损失。同时，本文提出的弱接合操作也各有优缺点：基于模版的弱接合执行效率很高，但是不能产生新关系型约束；基于包络和界信息的弱接合可以产生任意形式的新关系型约束，但是依然存在凸闭包计算过程中产生指数级冗余约束的隐患；基于两变量约束的弱接合计算简单，但是只能产生至多涉及两个变量的新关系型约束。在实际程序分析中，为了在执行效率和精度之间取得权衡，需要把强接合与这些弱接合动态地结合起来进行程序分析。为此，本文提出一种启发式的结合策略用以在程序分析过程中根据输入多面体的参数特性动态地选择合适的接合操作。

给定输入多面体，可以对其复杂度和稀疏程度进行定量评价，评价指标包括：多面体的维数、多面体的约束数、系数矩阵中非 0 系数的个数、系数矩阵中非 0 系数的复杂度（小整数、大整数、还是浮点数）等。据此，给定两个输入多面体，本文按如下策略选择接合操作：① 若两者都很简单（如约束少、系数矩阵稀疏），结合策略选择强接合操作，以尽量不损失精度；② 若两者都很复杂（如约束多、系数矩阵稠密），结合策略选择基于两变量约束的弱接合操作，以提高效率；③ 若其中一个很简单而另一个非常复杂，结合策略选择基于模版的弱接合操作，其模版约束由较简单的输入多面体的约束组成；④ 若两者非包络约束之和远大于（小于）包络约束之

和，则使用基于包络（非包络）约束与界约束的弱接合；⑤ 对于一般情况，结合策略选择基于包络/非包络约束和界约束的弱接合：如果一个多面体中的包络（非包络）约束较少，则使用包络（非包络）约束和界约束来作为该多面体的上近似。

此外，结合策略还将根据程序分析的不同阶段来选择合适的接合操作。在基于抽象解释的程序分析中，一般采用“迭代+加宽”的方法来计算程序不动点。在第一次迭代完成之前，尽可能地选择强接合操作来产生约束，以尽量不丢失最终可能稳定的约束，即不变式。在之后的迭代中，经常会出现参与接合操作的两个多面体中存在许多相同约束的情况，此时使用弱接合可极大地提高分析效率。在加宽算子应用之后，包络约束和界约束往往在最终稳定的不变式中占很大比率，此时，使用弱接合操作可以提高分析效率，而又不会损失太多精度。

3.5 实现及实验

本文采用双精度浮点数为本章提出的浮点多面体抽象域开发了一个原型系统FPPol。FPPol 内部采用实现了单纯形算法的GNU 线性规划工具包 GLPK（GNU Linear Programming Kit）[165]。并且，本文把 FPPol 适配到为数值抽象域的开发提供了通用接口的数值抽象域库 Apron[86] 中。本文使用支持 Apron 库的静态分析工具 Interproc [87] 开展了一系列实验。

(1) 浮点多面体域实验

为了评估 FPPol 的分析精度和计算效率，我们把 FPPol 得到的不变式及其性能与基于多精度有理数实现的多面体抽象域库 NewPolka 进行了比较。

首先，我们测试了所有 Interproc 自带的测试程序。其中，大多数测试程序是基于精确算术的纯整数程序，只有 *numerical* 程序涉及了浮点算术。我们还分析了图 3.10 中所示的 *ratelimiter* 程序，该程序是飞行控制系统中软件速率限制器的一个浮点实现程序，其中 $\odot_{32,?}$ 表示任意舍入模式的 32 位单精度浮点运算。该程序与第 1.3.3 节中图 2.8 中的程序基本一样，但是这里我们对 y 的初始值设置参数 M。参数 M 取不同的值对应该程序的不同版本。下面，我们先从理论上分析该程序的不变式。从理论上讲，关于变量 y 范围的任何区间 $[-M, M]$，其中 $M = 128 + \epsilon$ 且 $\epsilon > \epsilon_0$（其中 ϵ_0 为某个很小的正数），在 ② 处都是稳定的。但是，由于该程序在循环中需要关系型不变式才能为 y 找到稳定的区间范围。非关系型区间抽象域不能为 y 找到任何稳定的区间，而弱关系型八边形抽象域能找到的最精确的稳定区间为

```
y ← [−M, M];
while random() {
    x ← [−128, 128];
    d ← [1, 16];
    s ← y;
 ① r ← x ⊖32,? s;
    y ← x;
    if r ≤ ⊖d { y ← s ⊖32,? d; } else
    if d ≤ r  { y ← s ⊕32,? d; }
} ②
```

图 3.10 浮点速率限制器程序 *ratelimiter*（带参数）

$[-M_0, M_0]$，其中 $M_0 \approx 144.00005$，而多面体抽象域可以找到的最精确的稳定区间为 $[-M_1, M_1]$，其中$M_1 \approx 128.000047684$。对于本文而言，这个例子很具代表性，因为把浮点表达式抽象成实数上的区间线性表达式时不可避免地会产生高阶数值（祥见第 1.3.3 节）。例如，在 ① 处，赋值语句 $r \leftarrow x \ominus_{32,?} s$ 可抽象成

$$r \leftarrow [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times x - [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}] \times s + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$$

其中 $\varepsilon_{\text{rel}} = 2^{-23}$ 且 $\varepsilon_{\text{abs}} = 2^{-149}$（分别对应 32 位单精度浮点格式下的相对误差和最小非零正值）。注意，抽象后的表达式包含了高阶数值 2^{-23} 和 2^{-149}，因此，如果采用多精度有理数实现来分析该程序将需要很高的代价。

表 3.1 给出并比较了基于 FPPol 和 NewPolka 两种抽象域的程序分析的实验结果。其中，“type”栏给出了测试程序的类型：“int” 指该程序只涉及整数变量和精确算术；“fp” 指该程序涉及浮点算术。“#∇ delay” 栏给出了 Interproc 的加宽算子延迟参数（加宽算子应用之前的迭代次数）。“#iter.” 栏给出了总的（递增）迭代次数。

不变式. “Res. Inv.” 栏比较了所获得的不变式：“=”表示 FPPol 输出的不变式与 NewPolka 一样；“≈”表示 FPPol 找到了与 NewPolka 几乎一样的不变式，只是由于舍入误差导致结果稍有不同，这种情况下， FPPol 计算得到的多面体比 NewPolka 得到的多面体稍大；“>” 表示 FPPol 找到了比 NewPolka 更强的不变式。

对于整数测试程序，FPPol 所得到的不变式与 NewPolka 产生的不变式一样。实际上，这些程序只涉及小整数值，此时如果使用乘法方式的浮点 Fourier-Motzkin

表 3.1 浮点多面体域的实验结果

Program		Analyzer	FPPol			NewPolka		Res.
type	name	#∇delay	#iter.	#lp	$t(ms)$	#iter.	$t(ms)$	Inv.
int	ackerman	1	6	1476	35	6	7	$=$
int	bubblesort	1	8	675	24	8	8	$=$
int	fact	1	9	2106	65	9	15	$=$
int	heapsort	1	4	1968	76	4	15	$=$
int	maccarthy91	1	4	418	13	4	3	$=$
int	symmetricalstairs	1	6	480	18	6	6	$=$
fp	numerical	1	1	250	17	1	31	$\approx$
fp	ratelimiter(M=128)	3	5	1777	125	5	394	$\approx$
fp	ratelimiter(M=128)	4	5	2555	227	6	809	$>$
fp	ratelimiter (M=128.000047683)	6	9	4522	510	8	1889	$\approx$
fp	ratelimiter (M=128.000047683)	7	8	3688	238	9	2435	$>$
fp	ratelimiter (M=128.000047684)	1	3	1068	57	3	116	$\approx$

消除法往往不会引入舍入误差，从而得到的结果是精确的。

对于浮点测试程序，*numerical* 程序涉及浮点算术但是不含循环，因此不具挑战性。对于 *ratelimiter*，FPPol 能够找到不变式 $-M_1 \leq x \leq M_1$，其中$M_1 \approx 128.000047684$，前提是加宽算子延迟参数设置足够大：参数 $M = 128$ 时至少 4；$M = 128.000047683$ 时至少 7；$M = 128.000047684$ 时至少 1。然而，NewPolka 只能在 $M = 128.000047684$ 时找到不变式，注意此时 $[-M, M]$ 本身已经是稳定的区间范围了。更有趣的是，对于 $M = 128.000047683$ 版 *ratelimiter*，NewPolka 不能在程序点 ② 处找到任何有意义的不变式，即使加宽算子延迟参数设置到 100 次迭代（分析时间为 413.2 秒），虽然 $[-128.000047683, 128.000047683]$ 实际上离稳定的区间范围 $[-128.000047684, 128.000047684]$ 只差一点点。因此，从某种程度上可以说，FPPol 中各个域操作因为浮点而带来的上近似，实际上在分析过程中加速了不动点计算（相当于大步计算），从而某些时候可以在 NewPolka（相当于小步计算）之前更快地到达稳定的区域。

性能. 表 3.1 在“t(ms)”栏以毫秒为单位给出了程序分析所花费的时间，其实验平台为：Fedora Linux 操作系统，2GB 物理内存，Intel(R) Core(TM)2 CPU 6600 2.4 GHz处理器。对于整数程序，NewPolka 优于 FPPol。因为这些整数测试程序只涉及小整数值，所产生的多面体也比较简单，这在 NewPolka 中计算代价很小，而 FPPol 则需要频繁地调用计算代价相对较高的线性规划求解。然而，对于浮点程序，FPPol 的计算效率大大优于 NewPolka。事实上，在进行浮点抽象后，程序中涉及高阶数值，采用有理数实现将导致时空开销很大，从而极大地降低了 NewPolka 的性能，而 FPPol 中采用浮点表示从而避免这样的问题。

LP 代价. 表 3.1 在“#lp”栏给出了 FPPol 中 LP 查询的统计次数。实验中，我们发现，对于浮点程序，超过 75% 的 LP 查询源自冗余约束消除操作，其中近 80% 的 LP 查询来自凸闭包计算。因此，FPPol 的性能极大地依赖于我们所使用的线性规划求解器。此外，我们还发现，对于浮点程序，在 LP 求解上花费的时间往往占用了总分析时间的 85% 以上；对于整数程序，则占用了 70%以上。值得注意的是，即使直接简单地采用浮点数来实现多面体域而不考虑实现的可靠性，也不能绕过这些 LP 计算。因次，可以说，本文在浮点多面体抽象域上的可靠性保证并不是通过额外的高代价的计算来换取的。

LP 数值不稳定性问题. 在浮点程序分析实验中，我们发现线性规划工具 GLPK 常碰到“数值不稳定”问题，原因在于很小的系数和很大的系数往往可能出现在同一条约束（导致约束对应的斜率很陡）。具体而言，在浮点程序分析中，因为舍入误差引入的小浮点数可能在整个系统中传播，比如，除以小浮点数就会产生大浮点数。在本文的实现中，我们采用与第 3.3.2 节线性化算子类似的想法来对含有小系数或者大系数的项进行规整化，比如，选择 $d_k = 0$ 来消除该项。当然，我们相信，一个更快的、更鲁棒的、扩展性更好的 LP 求解器，如 CPLEX LP 求解器，可以大大提高本文浮点多面体抽象域的精度、性能和可扩展性。

(2) 弱接合实验

同样地，本文采用双精度浮点数在 FPPol 中实现了 3.4 节提出的弱接合操作。为了评估应用弱接合在实际程序分析中的效果，本文进行了一系列实验，在结果不变式和性能等方面比较了基于强接合的多面体分析与基于弱接合的多面体分析。

本实验的测试程序源于 StInG [166] 和 LpInv [91]，这些测试程序都是控制流接合密集型程序，符合本实验的需求。表 3.2 给出了使用NewPolka、基于强接合的

FPPol、基于强/弱接合结合策略的 FPPol 的程序分析实验结果。其中，对于某些高维测试程序，NewPolka 由于其多精度有理数实现所带来的时空复杂度问题，不能在规定的时间（1 小时）内完成分析，本文使用“>1h”标记。对于浮点多面体域 FPPol，表 3.2 分别给出了基于强接合和基于强/弱接合结合策略的分析结果。

表 3.2 多面体抽象域的弱接合的实验结果

Program		Analyzer	NewPolka		FPPol				Res.
					强接合		强/弱接合		
name	#var.	#∇delay	#iter.	$t(s)$	#iter.	$t(s)$	#iter.	$t(s)$	Inv.
dragon	5	9	12	49.12	×	×	11	3.78	$\neq$
cars	7	5	>1h	>1h	×	×	6	3.9	$<$
barber	8	2	6	53.9	×	×	6	1.53	$>$
barberm4-2	8	1	6	53.61	×	×	5	3.13	$\neq$
swim-pool-1	9	9	11	1.55	11	17.8	11	15.47	$\neq$
csm	13	9	11	0.96	11	6.94	11	6.74	$\neq$
scheduler-2p	14	9	12	1.42	×	×	12	67.44	$=$
multipool	18	4	>1h	>1h	6	16.8	6	16.5	$<$
consprod	18	6	8	174.5	×	×	8	86.34	$\neq$
incdec	32	3	>1h	>1h	×	×	6	208.3	$<$
mesh2x2	32	5	>1h	>1h	×	×	7	88.03	$<$
bigjava	44	3	>1h	>1h	×	×	6	173.6	$<$

性能. 表 3.2 所采用的实验平台为：Fedora Linux 操作系统，2GB 物理内存，Intel P4 2.8GHz 单核 CPU 处理器。从表 3.2，可以看出，通常情况下，基于强/弱接合结合策略的多面体分析比基于强接合的多面体分析的效率要高，并且鲁棒性更好。基于强接合的多面体分析过程中，由于中间产生的某些约束系统的高复杂性，导致 GPLK 出现“数值不稳定”问题不能找到最优解，从而不能有效地消除约束系统的冗余约束，极大地削弱了分析的效率甚至导致分析无法继续下去（用“×”标注）。在这种基于传播的分析中，之前复杂（约数多、稠密）的结果将会影响后续的分析。基于强、弱接合结合策略的多面体分析中，当中间过程的某些约束系统比较复杂时，结合策略将选用弱接合操作，在提高单次接合操作效率的同时，还能保证输出结果的简单性，不会影响到后续分析。如第 3.4 节所述，每次接合操作执行之前，我们都会对输入多面体的复杂度和稀疏程度进行定量评估，以选择合适的接合操作。比如，

只要两个输入多面体的稠密度（系数矩阵 A 中非 0 系数个数与矩阵大小之比）之和超过 0.65 且系数矩阵中非 0 系数个数均大于 45 时，我们就会选用弱接合操作。而对于某些高维程序（如 bigjava），只要两输入多面体稠密度之和大于 0.1 且系数矩阵中非 0 系数个数均超过 120 时，我们就会选用弱接合操作。

精度. 由于对于某些测试程序，基于强接合的 FPPol 分析失效，因此本文把使用 NewPolka 的分析结果作为标准参考，并与基于强/弱接合结合策略的 FPPol 所得结果不变式进行了比较，比较结果如表 3.2 “精度比较”栏所示：“=”表示分析结果一样，“>”表示 NewPolka 所得不变式更强，“<”表示 FPPol 所得不变式更强，“≠”表示两种方法分析结果不可比。通过比较和分析所得不变式，我们发现对于某些测试程序，基于强/弱接合结合策略所产生的不变式不弱于甚至强于基于 NewPolka 产生的不变式。因为，在多面体分析中，分析的主要精度损失源于加宽算子。理论上，由于加宽算子的非单调性，基于强/弱接合结合策略所产生的不变式不一定弱于基于强接合的分析。在多面体分析中，一种常用的策略是“延迟加宽”策略，以尽量减少加宽算子在开始几次迭代中可能带来的精度损失。但是，这种延迟策略可能导致分析过程中中间产生的约束表示越来越复杂，接合操作执行代价越来越高，并且由于线性规划工具的使用，分析的鲁棒性可能越来越差。此时，弱接合的使用仍然将是一个很好的选择：虽然每次弱接合操作应用可能带来一些精度损失，但是至少可以保证分析的成功完成，而且最终分析结果的精度也不一定差。

3.6 小结

本文给出了多面体抽象域的一种可靠浮点实现方法，即基于约束的浮点多面体抽象域。据我们所知，本文工作第一次把多面体抽象域可靠地适配到浮点计算上。本文浮点多面体抽象域的关键点在于采用只基于约束的表示方法，而不是采用经典多面体域及其实现库所采用的双重描述法。其实现主要依赖于两个原子操作：浮点 Fourier-Motzkin 消除法和严格线性规划。进一步，我们还给出了一些实用性的策略来提高浮点多面体域的精度和效率。浮点多面体域的优势在于其很高的计算效率及其在存储空间上的紧致表示，并且能够充分发挥目前线性规划求解器的强大计算能力。本文在浮点多面体抽象域原型系统 FPPol 上的实验结果也令人鼓舞：针对涉及高阶数值的程序，比如浮点程序，浮点多面体抽象域相对于基于多精度有理数实现的多面体域在计算效率方面具有明显的优势。

另外，针对基于约束的多面体抽象域的计算瓶颈——接合（凸闭包）操作，本

章设计并实现了一系列可靠的、低计算代价的弱接合操作，作为计算代价昂贵的接合操作的替代候选，以提高多面体分析的执行效率和可扩展性。并且，给出了一种启发式的结合策略，能够在多面体分析过程中动态地选择合适的接合操作，把多种弱接合操作和强接合操作有机地结合起来，在分析的效率和精度之间取得权衡。实验结果表明，在基于多面体的程序分析中，这些弱接合操作的应用不仅能有效地提高分析的效率，还能提高基于约束的多面体抽象域的易处理性和鲁棒性。

第四章　区间多面体抽象域

4.1　引言

本章给出区间多面体抽象域，用来推导程序变量之间的区间线性不等式关系（形如$\sum_k [a_k, b_k]x_k \leq c$）。

区间系数问题. 实际软硬件系统的分析与验证中，考虑到不精确性或者不确定性因素的影响，抽象模型或程序中很可能会包含一些区间形式的输入数据（尤其是一些物理量）。特别地，在程序分析中，对程序进行抽象的过程很可能会引入区间系数。比如，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，常常需要使用一些抽象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 $\sum_k [a_k, b_k]x_k + [c, d]$）[84, 85]。另外，当使用数值抽象域的浮点实现（如第 3 章的浮点多面体抽象域）来分析程序时，为了保证可靠性，待分析程序中的实数或有理数需要抽象成由浮点数所构成的区间。简而言之，在待分析的模型或者程序中，区间系数很自然地出现了。

另一方面，区间运算为浮点运算提供了一个保证其可靠性的有效途径。从第 3 章，我们可以看到为抽象域设计浮点实现时，为了保证浮点实现的可靠性，我们往往会求助于区间运算。在此过程中，往往无法避免区间系数的出现，比如第 3.3.2 节中浮点 Fourier-Motzkin 消除法在中间计算过程中就产生了区间线性不等式。因此，在抽象域的浮点实现过程中，区间系数也很可能会出现。在第 3.3.2 节中，我们的处理策略是使用线性化技术把区间线性不等式抽象成常规线性不等式，但是这个过程可能导致大量精度损失。

但是，目前尚没有一种数值抽象域可以直接支持区间系数作为域表示约束中的变量系数。因此，设计一个数值抽象域使其能够直接支持区间系数将会很有实用价值。

凸性限制问题. 目前，大部分数值抽象域（包括区间域、八边形域、多面体域等）只能表示凸的性质。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。

本文使用图 4.1 中所示程序来说明已有数值抽象域所存在的凸性局限性。对于 4.1 中所示程序，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

```
1:    real x, y;
2:    x ← 1;
3:    y ← 1;
4:    while (true) {
5:        x ← −x;
6:        y ← 1/x;   ①
7:    }
```

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 4.1 数值抽象域凸性局限性示例程序

该程序，会出现同样的问题。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 6 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所表达不了的，需要非凸的抽象域才能表达。

为此，本章将区间线性代数引入到数值程序分析中，设计区间线性抽象域来支持区间系数。首先，本章基于区间线性不等式来设计区间多面体抽象域，用来推导多个变量间的区间线性不等式关系。作为一种简单情形，还给出单变量区间线性不等式抽象域，用来推导单个变量上的取值范围。并且，通过采用“弱解”作为区间线性约束的解语义，使得区间线性抽象域可以表达某类非凸（甚至非连通）性质。此外，区间线性抽象域还会尽可能地把 0 排除在值范围之外，因此对于消除“除零错”误报有着天然地优势。

本章的结构组织如下：4.2 节回顾区间线性代数上的一些概念和结果；4.3 节介绍区间多面体抽象域的设计，并讨论其可能应用；4.4 节介绍单变量区间线性不等式抽象域；4.5 节介绍区间多面体域及单变量区间线性不等式抽象域的原型系统实现，给出并分析实验结果；4.6 节是对本章的小结。

4.2 区间线性代数

本节简单介绍区间线性代数里的一些基本概念和表示，并回顾本文将用到的一些重要结果。其中，大部分内容都可以在文献 [128, 129] 中找到。

本文采用如下符号表示。$A \in \mathbb{R}^{m \times n}$ 表示一个大小为 $m \times n$ 的实数矩阵。区间使用粗体表示形如 $\mathbf{x}$，其上下界分别通过 $\underline{x}$ 和 $\overline{x}$ 来表示，其中 $\underline{x} \leq \overline{x}$。记所有实数区间的集合为 $\mathbb{IR}$。在本文中，区间及区间代数上的区间相关对象均采用粗体表示。

4.2.1 区间线性不等式系统

设 $\underline{A}, \overline{A} \in \mathbb{R}^{m\times n}$ 为两个实数矩阵，并满足 $\underline{A} \leq \overline{A}$，其中比较操作指逐元素比较，那么如下实数矩阵集合

$$\mathbf{A} = [\underline{A}, \overline{A}] = \{A \in \mathbb{R}^{m\times n} \colon \underline{A} \leq A \leq \overline{A}\}$$

称为一个**区间矩阵**，且矩阵 $\underline{A}, \overline{A}$ 称为其上下界。我们定义区间矩阵 $\mathbf{A}$ 的**中点矩阵**为 $A_c = \frac{1}{2}(\underline{A} + \overline{A})$，其**半径矩阵**为 $\triangle_A = \frac{1}{2}(\overline{A} - \underline{A})$。那么有，$\mathbf{A} = [\overline{A}, \underline{A}] = [A_c - \triangle_A, A_c + \triangle_A]$. 一个**区间向量**是一个单列区间矩阵 $\mathbf{d} = \{d \in \mathbb{R}^m \colon \underline{d} \leq d \leq \overline{d}\}$，其中 $\underline{d}, \overline{d} \in \mathbb{R}^m$ 并且 $\underline{d} \leq \overline{d}$。

设 $\mathbf{A} \in \mathbb{IR}^{m\times n}$ 为一个 $m \times n$ 区间矩阵，$b \in \mathbb{R}^m$ 为一个 m 实数向量。我们称如下系统

$$\mathbf{A}x \leq b$$

为一个**区间线性不等式系统**，表示所有线性不等式系统 $Ax \leq b$ 所构成的不等式系统族，其中 $A \in \mathbf{A}$。

定义 4.2.1 (弱解) *向量 $x \in \mathbb{R}^n$ 称为区间线性不等式系统 $\mathbf{A}x \leq b$ 的一个**弱解**，若存在某个 $A \in \mathbf{A}$ 使得 x 满足 $Ax \leq b$。并且，集合*

$$\Sigma_{\exists}(\mathbf{A}, b) = \{x \in \mathbb{R}^n \colon \exists A \in \mathbf{A}.\, Ax \leq b\}$$

*称为系统 $\mathbf{A}x \leq b$ 的**弱解集合**。*

如下定理从代数角度刻画了区间线性不等式系统的弱解集合 [129]。

定理 4.1 *向量 $x \in \mathbb{R}^n$ 是 $\mathbf{A}x \leq b$ 的弱解当且仅当 x 满足 $A_c x - \Delta_A |x| \leq b$。*

总体来说，弱解集合可以是非凸的，甚至非连通的。其非凸性质源于定理 4.1 中的非线性因子 $|x|$。一个(闭) 象限是 n 维欧几里得空间中 2^n 子集中的一个，通过限制每个笛卡尔坐标轴为非负或非正。注意，在一个给定的象限内，x 保持固定符号，从而可以去掉定理 4.1 中的 $|\cdot|$ 符号。因此，弱解集合与每个象限的交可通过一个凸多面体来表示。然而，对于一个 n 维区间线性不等式系统，其精确的描述需要 2^n 的不同的凸多面体来表达。

包含弱解集合 $\Sigma_{\exists}(\mathbf{A}, b)$ 的最小的区间向量 $\mathbf{x}^H$ 称为 $\Sigma_{\exists}(\mathbf{A}, b)$ 的区间包（interval hull），即 $\mathbf{x}_k^H = [\underline{x}_k^H, \overline{x}_k^H]$，其中 $\underline{x}_k^H = \min\{x_k \colon x \in \Sigma_{\exists}(\mathbf{A}, b)\}, \overline{x}_k^H = \max\{x_k \colon x \in \Sigma_{\exists}(\mathbf{A}, b)\}$, $k = 1, \ldots, n$。但是，计算 $\Sigma_{\exists}(\mathbf{A}, b)$ 解集合的区间包是一个 NP-hard 问题 [128]。

4.2.2 区间线性规划

设 $\mathbf{A} \in \mathbb{IR}^{m\times n}$ 为一个 $m \times n$ 区间矩阵，$b \in \mathbb{R}^m$ 为一个 m 实数向量，$\mathbf{c} \in \mathbb{IR}^n$ 为一个 n 维区间向量。我们称如下一族线性规划问题

$$f(A,b,c) = \min\{c^T x\colon\ Ax \le b\}$$

其中，

$$A \in \mathbf{A}, c \in \mathbf{c}$$

为一个**区间线性规划** ILP 问题

$$\mathbf{f}(\mathbf{A},b,\mathbf{c}) = \min\{\mathbf{c}^T x\colon \mathbf{A}x \le b\}.$$

并且，区间 $[\underline{f}(\mathbf{A},b,\mathbf{c}), \overline{f}(\mathbf{A},b,\mathbf{c})]$，其中 $\underline{f}(\mathbf{A},b,\mathbf{c}) = \inf\{f(A,b,c)\colon A \in \mathbf{A}, c \in \mathbf{c}\}, \overline{f}(\mathbf{A},b,\mathbf{c}) = \sup\{f(A,b,c)\colon A \in \mathbf{A}, c \in \mathbf{c}\}$，称为上述 ILP 问题的**最优值范围**。

本文，我们只关注下界 $\underline{f}(\mathbf{A},b,\mathbf{c})$。要计算精确的 $\underline{f}(\mathbf{A},b,\mathbf{c})$，最简单的方法就是把该问题分解成 2^n 个 LP 问题来求解，每个象限对应一个 LP 问题。实际中，文献 [131] 根据语法表示把 ILP 问题分成几个类型，然后为不同的类型设计不同的求解算法。最近，Jansson [108] 提出了一种迭代方法，通过求解一系列中点 LP 问题来计算 $\underline{f}(\mathbf{A},b,\mathbf{c})$ 的可靠下界，并且在许多情况下，该方法所需计算代价并不高。

本文只聚焦在抽象域的设计上，因此在后续章节我们将把区间线性规划当作一个黑盒来使用。

4.3 区间多面体域

本节提出一个新的数值抽象域—— **区间多面体抽象域**。其主要思想是使用区间线性不等式约束来作为域表示方法。该抽象域可用来推导程序中变量 $x_k (k = 1,\ldots,n)$ 间形如 $\Sigma_k[a_k,b_k] \times x_k \le b$ 的关系，其中常数 $a_k, b_k, c \in \mathbb{R}$ 由分析器自动推导出来。下面，本节将在实数 $\mathbb{R}$ 上来介绍区间多面域的域表示以及域操作的实现方法。后面，在第 4.5 节中我们再讨论区间多面体域的可靠浮点实现方法。

4.3.1 域表示

在域表示上，区间多面体域与已有数值抽象域的一个重要相似之处在于，它们都把程序环境看成是某类特定约束系统（即有穷个约束的合取）上的解。这里，区间多面体域选择区间线性不等式系统来作为其域表示。

(1) 区间多面体及其性质

首先，本文把区间线性不等式系统 $\mathbf{A}x \leq b$ 的弱解集合 $\Sigma_{\exists}(\mathbf{A}, b)$ 在几何上对应的图形区域称为一个**区间多面体**。那么，在实数 $\mathbb{R}$ 上，一个区间多面体 $\mathbf{P}$ 可通过一个区间线性不等式系统 $\mathbf{A}x \leq b$ 来描述，其中 $\mathbf{A} \in \mathbb{IR}^{m\times n}$ 是一个区间矩阵，$b \in \mathbb{R}^m$ 是一个实数向量，m 是不等式系统中约束的数目，n 是不等式系统中变量的个数。其语义在代数上对应区间线性不等式系统 $\mathbf{A}x \leq b$ 的弱解集合 $\Sigma_{\exists}(\mathbf{A}, b)$，在几何上对应位于该区间多面体 $\mathbf{P}$ 内的点的集合即 $\gamma(\mathbf{P}) = \{x \in \mathbb{R}^n \mid x \in \Sigma_{\exists}(\mathbf{A}, b)\}$，其中每个点$x \in \gamma(\mathbf{P})$ 代表一个可能的程序环境（即对所有变量 x 的一种可能赋值）。

从域表示方法上看，区间多面体域是经典凸多面体抽象域的区间扩展版本，因为凸多面体域仅支持标准的（非区间）线性不等式。类似于凸多面体域，区间多面体抽象域与具体域（程序环境集合）之间不存在抽象函数 α，因此也不存在 Galois 连接。比如，对于圆盘 $\{(x, y) \in \mathbb{Q}^2 \mid x^2 + y^2 \leq 1\}$，包含该圆盘的最小区间多面体也不存在。区间多面体抽象域只构成格，而非完全格。

在弱解语义下，区间线性等式 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k = [\underline{b}, \overline{b}]$ 可以表示成一对区间线性不等式，即 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq \overline{b}$ 和 $\sum_k -[\underline{a}_k, \overline{a}_k] \times x_k \leq -\underline{b}$ 的合取。而形如 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k < b$ 的严格不等式也可以可靠地抽象为 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b$。

不难看出，区间多面体具有如下性质:

- 非凸性：一个区间多面体可以是非凸的，但是它与 $\mathbb{R}^n$ 上每个象限的交必然是一个（可能为空的）凸多面体。甚至，一个区间多面体可以是非连通的（如图 4.2 (1.e)）。
- 关于交封闭：两个区间多面体的交仍是一个区间多面体。
- 关于并不封闭：两个区间多面体的并不一定是一个区间多面体。

总体而言，一个区间多面体在几何上有着复杂的形状。图 4.2 给出了二维平面上区间多面体 (1) 和非区间多面体 (2) 的一些例子。其中，第 (1) 组中的区间多面体分别对应如下区间线性不等式系统：(1.a) $\{[-1, 1]x + y = 0, [-1, 1]y = 1\}$, (1.b) $\{[-1, 0]x + y = [0, 1]\}$, (1.c) $\{[1, 2]x + [1, 2]y = [1, 2]\}$, (1.d) $\{[-1, 1]x + 2y = [-2, 2], 2x + [-2, 1]y = [-2, 2]\}$, (1.e) $\{[-1, 1]x = 1, [-1, 1]y = 1, x = [-2, 2], y = [-2, 2], x + y = [-1, 1]\}$。图 4.2(2) 给出了一些不能被区间多面体所表达的图形区域。具体而言，(2.a), (2.b), (2.d) 不能被区间多面体所表达是因为它们与某个象限（如 (+,+) 象限）的交不是凸的。根据 [130] 文中给出的性质，(2.c) 和 (2.e) 也不能被区间多面体所表达。

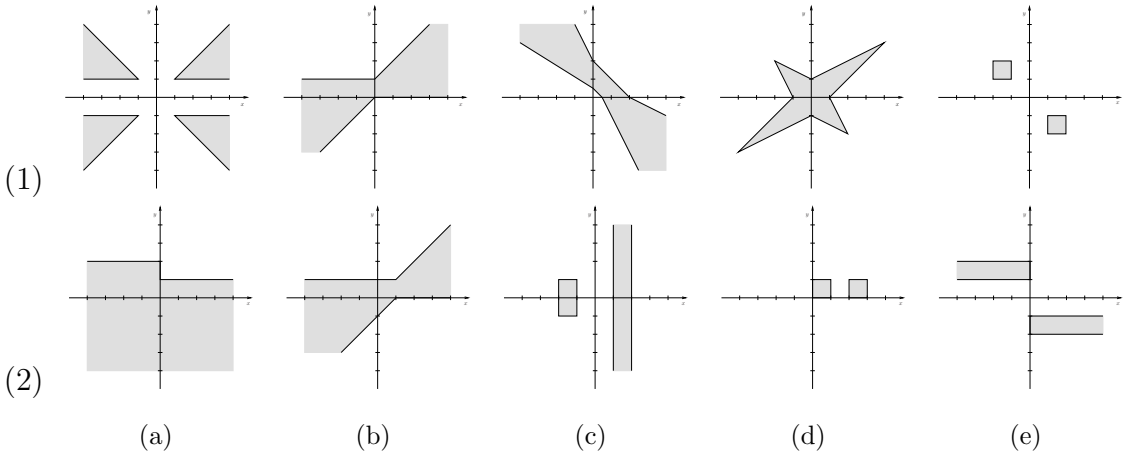


图 4.2 区间多面体与非区间多面体的几何示例

(2) 区间多面体：凸多面体的约简基数幂

本节，我们把区间多面体看作是关于凸多面体的约简基数幂（reduced cardinal power，见文 [80] 中第 10.2 节）：把每个象限映射到一个凸多面体。这主要是基于如下事实：一个区间多面体与一个象限的交必然是一个（可能为空的）凸多面体。从而，最差情形下，一个 n 维区间多面体可以看作 2^n 个凸多面体的集合并。更准确地说，给定区间多面体 $\mathbf{P}$，设 q 为 $\mathbf{P}$ 中那些无符号限制（可正可负）、且在区间矩阵 $\mathbf{A}$ 中有非单点区间作为其系数的变量的个数，则 $\mathbf{P}$ 可以表示成最多 2^q 个凸多面体的集合并。

下面，我们考虑通过凸多面体的约简基数幂来实现区间多面体域。一般情形下，我们需要为每个象限维护一个（可能为空的）凸多面体。凸多面体的约简基数幂域上的每个操作可以通过对凸多面体域上对应操作进行“提升”来得到。例如，两区间多面体的接合，可以通过在每个象限内分别计算对应凸多面体的凸闭包来实现。但是，赋值迁移函数则需要仔细考虑，因为某个象限内的凸多面体在应用赋值迁移函数后可能“进入”别的象限。这种情况下，每个象限的结果需要更新成迁移操作之后属于该象限的图形区域的凸闭包。因此，该域不是简单地等价于凸多面体域的有穷析取完全化（finite disjunctive completion）。从这个角度看，称之为凸多面体的象限划分域更为合适。

为了能利用区间多面体的紧致表示带来的优势，在每个域操作之后，可以把一组凸多面体（每个象限一个）构成的集合抽象成一个区间多面体。然而，能够精确地表示那些凸多面体的集合并的区间多面体可能并不存在。比如，图 4.2(2.c) 中所示的多面体集合。因此，需要设计算法来计算包含那些凸多面体的最小区间多面体。

但是，据我们所知，目前尚没有这样的算法能够计算出最小的区间多面体（本文第 6 章将给出一个算法）。而且，这样的算法如果存在，计算复杂度也会比较高。

例 4.3.1 给定图 *4.2(1.b)* 所示的区间多面体 $\mathbf{P} = \{[-1,0]x + y = [0,1]\}$，在凸多面体的约简基数幂域中，在 $\mathbf{P}$ 上执行赋值迁移函数 $[\![x := x + 1]\!]^\sharp$ 后，我们将得到图 *4.2(2.b)* 所示的图形区域，注意该区域不能被任何区间多面体精确表示。为此，我们首先在每个象限里计算位于其中的图形区域的凸闭包，将得到如下凸多面体: $(+,+)$ 象限 $\{x \geq 0, y \geq 0, -x + y \leq 1\}$; $(+,-)$ 象限 $\{x \geq 0, -1 \leq y \leq 0\}$。最后，通过采用第 *4.3.2* 节将提出的弱接合算法，可计算得到最小区间多面体 $\{[-1,0]x + y = [-1,1]\}$。

总体而言，通过凸多面体的约简基数幂来实现区间多面体域，将导致极高的复杂度，因为凸多面体域本身在最差情况下就已经是指数级复杂度。当然，在某些实际应用中，因为物理方面的限制，许多变量并不会改变其符号，从而使得区间多面体只与少数象限相交不为空。在这种情形下，凸多面体的约简基数幂域才比较可行。

4.3.2 域操作

前面已经提到，通过凸多面体的约简基数幂域来实现区间多面体域，将带来极高的计算复杂度，难以在实际中取得应用。为此，本节将设计更高效的近似算法来实现区间多面体域。类似于基于约束的凸多面体抽象域（见第 3.2 节），该构造主要基于如下两个原子操作：

- 区间 Fourier-Motzkin 消去法
- 区间线性规划 ILP

下面，我们将详细描述区间多面体抽象域上用于静态分析所需要的常用域操作的实现方法。

(1) 化简

类似凸多面体抽象域，区间多面体的约束表示不是惟一的。如，区间线性等式 $[-1,1]x = 1$ 和不等式 $[-1,1]x \leq -1$ 有着相同的弱界集合 $\{x \in [-\infty,-1] \cup [1,+\infty]\}$。考虑计算效率，一般希望约束越少越简单越好。

约简. 根据定理 4.1，一个区间线性不等式 φ: $(\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b)$ 可以约简成 φ': $(\sum_k [\underline{a}'_k, \overline{a}'_k] \times x_k \leq b)$ 其中 $x_k \in [\underline{x}_k^H, \overline{x}_k^H]$，

$$[\underline{a}'_k, \overline{a}'_k] = \begin{cases} [\underline{a}_k, \underline{a}_k] & \text{if } \underline{x}_k^H \geq 0 \\ [\overline{a}_k, \overline{a}_k] & \text{if } \overline{x}_k^H \leq 0 \\ [\underline{a}_k, \overline{a}_k] & \text{otherwise} \end{cases}$$

在程序分析过程中，对约束进行约简很有实际意义，因为相对于 φ，在后续计算中 φ' 将导致更少的精度损失，比如，在区间组合操作中（见第 4.3.2 节）。

冗余约束消除. 一个区间线性不等式 $\varphi \in \mathbf{P}$ 称为是冗余的，当且仅当 φ 被 $\mathbf{P}$ 中其他的约束蕴含，即 $\mathbf{P} \setminus \{\varphi\} \models \varphi$。给定 φ: $(\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b) \in \mathbf{P}$，我们可以通过求解如下 ILP 问题来检查 φ 是否冗余：$\mu = \max \sum_k [\underline{a}_k, \overline{a}_k] \times x_k$ subject to $\mathbf{P} \setminus \{\varphi\}$。若 $\mu \leq b$，则 φ 是冗余并可以从 $\mathbf{P}$ 中删除。重复应用该过程到 $\mathbf{P}$ 中所有不等式上，就可以删除 $\mathbf{P}$ 中所有冗余约束。

为了提高计算效率，可以首先使用一些轻量级的方法然后只在必要时才使用高代价的基于 ILP 的方法。例如，给定 φ: $(\Sigma_k[\underline{a}_k, \overline{a}_k] \times x_k \leq b) \in \mathbf{P}$，如果存在其他区间线性不等式 φ': $(\Sigma_k[\underline{a}'_k, \overline{a}'_k] \times x_k \leq b') \in \mathbf{P}$ 使得 $b \leq b'$ 且 $\forall k.[\underline{a}_k, \overline{a}_k] \subseteq [\underline{a}'_k, \overline{a}'_k]$，则 φ 在 $\mathbf{P}$ 中是冗余的。其次，给定 φ: $(\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b) \in \mathbf{P}$，如果 $\forall k.0 \in [\underline{a}_k, \overline{a}_k]$ 且 $b \geq 0$，则 φ 定义的是一个全局空间，可以从 $\mathbf{P}$ 中删除。

(2) 空区间多面体测试

空区间多面体对应了区间多面体域中的 $\perp$ 元素。一个区间多面体是空的即 $\gamma(\mathbf{P}) = \emptyset$，当且仅当其区间线性不等式系统不存在弱解即 $\Sigma_\exists(\mathbf{A}, b) = \emptyset$。由定理 4.1 可知，检查区间线性不等式系统是否存在弱解，理论上可以通过在每个象限上检查其对应线性系统的可行性来实现，比如，在每个象限上应用一次线性规划。当然，在 ILP 求解器计算目标函数最大（小）值时，ILP 求解器会自动检查该区间线性不等式系统的可行性。在程序分析过程中，约束往往是一条条增量式地加入，因此空区间多面体测试也可以通过一种增量式方法来实现。当添加一个新的约束 $\sum_k [a_k, b_k] \times x_k \leq b$ 到一个非空区间多面体 $\mathbf{P}$ 中时，我们求解 ILP 问题：$\min \sum_k [a_k, b_k] \times x_k$ subject to $\mathbf{P}$。如果 $b < \mu$，则说明新的区间多面体是空的。

(3) 投影

为了能够从定义 $\mathbf{P}$ 的区间线性不等式系统中消除所有 x_i 的出现，本文把经典 Fourier-Motzkin 消除法（FME）适配到区间算术上，从而得到 FME 的一个区间版本——区间 Fourier-Motzkin 消除法（IFME）。

设 $\mathbf{P} = \{\mathbf{A}x \leq b\}$ 为一个区间多面体，x_i 为待消除变量。当 $\mathbf{A}$ 中 x_i 的所有

非 0 区间系数均不含 0 时，IFME 算法可以较容易地通过区间算术来实现。然而，一般情形下，$\mathbf{P}$ 中可能存在这样的约束 φ: $(\sum_k[\underline{a}_k,\overline{a}_k]x_k \le b)$，其中 $0 \in [\underline{a}_i,\overline{a}_i]$ 且 $[\underline{a}_i,\overline{a}_i] \neq [0,0]$。此时，IFME 算法会因为除以一个包含 0 的区间而中断。为了避免这种情形的发生，在应用 IFME 算法之前，我们先应用部分线性化算子 $\zeta(\varphi,x_i)$ 把 $\mathbf{P}$ 中所有这样的含 0 非 0 区间系数线性化为标量系数。

定义 4.3.1 (部分线性化算子) 设 φ: $(\sum_k [\underline{a}_k,\overline{a}_k] \times x_k \le b)$ 为一个区间线性不等式，$x_i \in [\underline{x}_i^H,\overline{x}_i^H]$。部分线性化算子定义为 $\zeta(\varphi,x_i) \stackrel{\text{def}}{=}$

$$\begin{cases} \underline{a}_i \times x_i + \sum\limits_{k\neq i} [\underline{a}_k,\overline{a}_k] \times x_k \le b & \text{if } \underline{x}_i^H \ge 0 \\ \overline{a}_i \times x_i + \sum\limits_{k\neq i} [\underline{a}_k,\overline{a}_k] \times x_k \le b & \text{if } \overline{x}_i^H \le 0 \\ c \times x_i + \sum\limits_{k\neq i} [\underline{a}_k,\overline{a}_k] \times x_k \le \sup(b - [\underline{a}_i - c,\overline{a}_i - c] \times [\underline{x}_i^H,\overline{x}_i^H]) & \text{otherwise} \end{cases}$$

其中 c 可以是任意标量数值。

实践中，我们通常选择区间 $[\underline{a}_i,\overline{a}_i]$ 的中点 $c = (\underline{a}_i + \overline{a}_i)/2$，因为大部分情形下将得到较精确的结果以尽量减少精度损失。

例 4.3.2 考虑区间线性不等式 $[0,2]x + y \le 2$ 及变量 x 的取值范围 $x \in [-2,4]$。若选择 $[\underline{x},\overline{x}]$ 的中点作为 c，$\zeta(\varphi,x)$ 将得到 $x + y \le 6$。注意，这里发生了精度损失，比如，点$(0,4)$ 满足结果不等式 $x + y \le 6$ 但是并不满足原来的区间线性不等式 $[0,2]x + y \le 2$。

定理 4.2 (部分线性化算子的可靠性) 给定一个区间线性不等式 φ 及变量 $x_i \in [\underline{x}_i^H,\overline{x}_i^H]$，$\zeta(\varphi,x_i)$ 是 φ 的上近似，即 $\forall x.(x_i \in [\underline{x}_i^H,\overline{x}_i^H] \wedge x \in \gamma(\varphi)) \Rightarrow x \in \gamma(\zeta(\varphi,x_i))$。

借助于上述部分线性化技术，区间 Fourier-Motzkin 消除法就可以应用在任意形式的区间多面体上。给定一个不等式 φ: $(\sum_k[\underline{a}_k,\overline{a}_k]x_k \le b)$，我们定义 $\iota(\varphi,x_i)$ 为

$$\iota(\varphi,x_i) \stackrel{\text{def}}{=} \begin{cases} \zeta(\varphi,x_i) & \text{if } 0 \in [\underline{a}_i,\overline{a}_i] \wedge [\underline{a}_i,\overline{a}_i] \neq [0,0] \\ \varphi & \text{otherwise} \end{cases}$$

那么，给定一个区间多面体 $\mathbf{P}$ 及一个待消除变量 x_i，区间 Fourier-Motzkin 消除法（IFME）可以定义为

$$\begin{aligned} &\text{IFME}(\mathbf{P},x_i) \stackrel{\text{def}}{=} \{\ (\textstyle\sum_k[\underline{a}_k,\overline{a}_k]x_k \le b) \in \mathbf{P}' \mid [\underline{a}_i,\overline{a}_i] = [0,0]\ \} \\ &\cup \left\{ \textstyle\sum_{k\neq i} \left(\frac{[\underline{a}_k^+,\overline{a}_k^+]}{[\underline{a}_i^+,\overline{a}_i^+]} + \frac{[\underline{a}_k^-,\overline{a}_k^-]}{[-\overline{a}_i^-,-\underline{a}_i^-]} \right) x_k \le b' \;\middle|\; \begin{array}{l} (\sum_k[\underline{a}_k^+,\overline{a}_k^+]x_k \le b^+) \in \mathbf{P}', \underline{a}_i^+ > 0 \\ (\sum_k[\underline{a}_k^-,\overline{a}_k^-]x_k \le b^-) \in \mathbf{P}', \overline{a}_i^- < 0 \end{array} \right\} \end{aligned}$$

其中 $\mathbf{P}' = \{\iota(\varphi,x_i) \mid \varphi \in \mathbf{P}\}$，$b' = \sup\left(\frac{b^+}{[\underline{a}_i^+,\overline{a}_i^+]} + \frac{b^-}{[-\overline{a}_i^-,-\underline{a}_i^-]} \right)$。

定理 4.3 (区间 Fourier-Motzkin 消除法的可靠性) 给定一个区间多面体 $\mathbf{P}$ 和一个待消除变量 x_i，$\mathbf{P}$ 内的所有点都满足 $\mathrm{IFME}(\mathbf{P}, x_i)$，即 $\forall x \in \gamma(\mathbf{P}) \Rightarrow x \in \gamma(\mathrm{IFME}(\mathbf{P}, x_i))$。

(4) 接合

为了对程序中控制流接合进行抽象，我们需要计算程序环境的并。然而，据我们所知，目前还不存在已有算法可以计算出包含两个区间多面体集合并的最小区间多面体。即使有，计算代价也可能比较高。为此，本文采用一个较小计算代价的弱接合操作来构造一个区间多面体来包含两个输入区间多面体。

其主要思想如下：首先，我们在约束上定义操作 $\uplus$ 使得 $\gamma(\varphi') \cup \gamma(\varphi'') \subseteq \gamma(\varphi' \uplus \varphi'')$，以此来产生一个约束作为两输入约束的集合并的上近似。那么，给定两个区间多面体 $\mathbf{P}'$ 和 $\mathbf{P}''$，根据分配律有

$$\begin{aligned}\gamma(\mathbf{P}') \cup \gamma(\mathbf{P}'') &= \left(\bigcap_{\varphi' \in \mathbf{P}'} \gamma(\varphi')\right) \cup \left(\bigcap_{\varphi'' \in \mathbf{P}''} \gamma(\varphi'')\right) = \bigcap_{\substack{\varphi' \in \mathbf{P}' \\ \varphi'' \in \mathbf{P}''}} (\gamma(\varphi') \cup \gamma(\varphi'')) \\ &\subseteq \bigcap_{\substack{\varphi' \in \mathbf{P}' \\ \varphi'' \in \mathbf{P}''}} (\gamma(\varphi' \uplus \varphi''))\,.\end{aligned}$$

本文的弱接合就是通过使用 $\uplus$ 操作把 $\mathbf{P}_1$ 中的不等式约束与 $\mathbf{P}_2$ 中的不等式约束一对对组合起来。

区间组合.

定义 4.3.2 (区间组合) 给定两个区间线性不等式 $\varphi'\colon (\sum_k [\underline{a}'_k, \overline{a}'_k] \times x_k \leq b')$ 和 $\varphi''\colon \sum_k [\underline{a}''_k, \overline{a}''_k] \times x_k \leq b''$，$\varphi'$ 与 φ'' 的区间组合定义为

$$\varphi' \uplus \varphi'' \stackrel{\mathrm{def}}{=} \left(\textstyle\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b\right),$$

其中 $b = \max(b', b'')$，$[\underline{a}_k, \overline{a}_k] = [\min(\underline{a}'_k, \underline{a}''_k), \max(\overline{a}'_k, \overline{a}''_k)]$。

该定义可以直接提升到区间多面体上：给定两个区间多面体 $\mathbf{P}'$ 和 $\mathbf{P}''$，$\mathbf{P}' \uplus \mathbf{P}'' \stackrel{\mathrm{def}}{=} \{\varphi' \uplus \varphi'' \mid \varphi' \in \mathbf{P}' \wedge \varphi'' \in \mathbf{P}''\}$。

例 4.3.3 考虑两个区间多面体 $\mathbf{P}' = \{y \leq 1, -y \leq -1\}$ 和 $\mathbf{P}'' = \{-x+y \leq 0, x-y \leq 0\}$。通过区间组合，我们得到 $\mathbf{P} = \mathbf{P}' \uplus \mathbf{P}'' = \{[-1, 0]x + y = [0, 1]\}$，其弱解集合如图 *4.2(1.b)* 所示。注意，$\mathbf{P}$ 是包含了 $\mathbf{P}'$ 和 $\mathbf{P}''$ 的最小区间多面体。但是，这里发生了精度损失，比如，点 $(1, 0)$ 满足结果 $\mathbf{P}$ 但是既不满足 $\mathbf{P}'$ 也不满足 $\mathbf{P}''$。

定理 4.4 (区间组合的可靠性) 给定两个区间线性不等式 φ' 和 φ''，其区间组合 $\varphi' \uplus \varphi''$ 是 φ' 与 φ'' 的可靠上近似，即 $\gamma(\varphi') \cup \gamma(\varphi'') \subseteq \gamma(\varphi' \uplus \varphi'')$。

上述定理可以导出区间多面体上 $\uplus$ 操作的可靠性，即 $\gamma(\mathbf{P}' \cup \gamma(\mathbf{P}'') \subseteq \gamma(\mathbf{P}' \uplus \mathbf{P}'')$。然而，$\varphi' \uplus \varphi''$ 给出的结果可能不是包含了 φ' 与 φ'' 弱解集合的并的最紧（指弱解集

合最小）区间线性不等式。而且，区间组合的计算精度依赖于输入的语法表示。输入中的系数区间越小，得到的结果将越精确。因此，常常会在区间组合之前先执行一下第 4.3.2 节介绍的约简操作，来缩紧区间系数。

某些情形下，区间组合还可以进一步改进。设 $\mathbf{a}$，$\mathbf{b}$ 为两个区间。如果存在非 0 实数 λ 满足 $\lambda \times \mathbf{a} = \mathbf{b}$（即 $\lambda * \underline{a} = \underline{b}$ 且 $\lambda * \overline{a} = \overline{b}$），那么 λ 称为 $\mathbf{a}$ 关于 $\mathbf{b}$ 的乘子。给定 $\varphi'\colon (\sum_k [\underline{a}'_k, \overline{a}'_k] \times x_k \leq b')$ 与 $\varphi''\colon (\sum_k [\underline{a}''_k, \overline{a}''_k] \times x_k \leq b'')$，如果存在一个正乘子 λ^+ 使得对于某个 i，$\lambda^+ \times [\underline{a}'_i, \overline{a}'_i] = [\underline{a}''_i, \overline{a}''_i]$ 成立，那么 φ' 与 φ'' 的区间组合可以通过 $\varphi' \uplus \varphi'' = (\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b)$ 来计算，其中 $b = \max(\lambda^+ * b', b'')$，$[\underline{a}_k, \overline{a}_k] = [\min(\lambda^+ * \underline{a}'_k, \underline{a}''_k), \max(\lambda^+ * \overline{a}'_k, \overline{a}''_k)]$。大多数情况下，带乘子的区间组合会比通用形式的区间组合更精确。比如，给定 $\varphi_1\colon (x+y \leq 2)$ 与 $\varphi_2\colon (-x+2y \leq 2)$，$\varphi_1 \uplus \varphi_2$ 得到 $\varphi\colon ([-1,1]x + [1,2]y \leq 2)$。然而，如果我们使用乘子版本（即把 $\varphi_1\colon (x+y \leq 2)$ 重写为 $\varphi'_1\colon (2x + 2y \leq 4)$），$\varphi'_1 \uplus \varphi_2$ 将得到 $\varphi'\colon ([-1,2]x + 2y \leq 4)$，其结果 φ' 比之前的 φ 更精确。

弱接合.

定义 4.3.3 (包络) 给定两个区间多面体 $\mathbf{P}_1$ 和$\mathbf{P}_2$，$\mathbf{P}_1$ 与 $\mathbf{P}_2$ 的包络定义为

$$env(\mathbf{P}_1, \mathbf{P}_2) \stackrel{\text{def}}{=} \mathcal{S}_1 \cup \mathcal{S}_2$$

其中，$\mathcal{S}_1 = \{\ \varphi_1 \in \mathbf{P}_1 \mid \mathbf{P}_2 \models \varphi_1\ \}, \mathcal{S}_2 = \{\ \varphi_2 \in \mathbf{P}_2 \mid \mathbf{P}_1 \models \varphi_2\ \}$。

设 $i \in \{1,2\}$，那么对于任意 $\varphi \in \mathbf{P}_i$，若 $\varphi \in env(\mathbf{P}_1, \mathbf{P}_2)$，我们称 φ 为 $\mathbf{P}_i$ 中的一个包络约束，否则称 φ 是 $\mathbf{P}_i$ 中的非包络约束。我们记 $\mathbf{P}_i$ 中的非包络约束为 $\overline{env}(\mathbf{P}_i)$。

本文中，我们还使用区间多面体 $\mathbf{P}$ 的包围盒，记为 $BB(\mathbf{P})$，来表示 $\Sigma_\exists(\mathbf{A}, b)$ 的区间包 $\mathbf{x}^H$。$BB(\mathbf{P})$ 可通过区间线性规划 ILP 来计算，即计算 $\max(\min)\ x_k$ subject to $\mathbf{P}$。实际中，$BB(\mathbf{P})$ 可以通过一种 on-the-fly 方式来更新，并且一个可靠的上近似的界盒可以通过采用类似第 3.3.5 节中的一些低代价的方法来得到。给定两个盒 $B' = \{x \in [\underline{b}', \overline{b}']\}$ 及 $B'' = \{x \in [\underline{b}'', \overline{b}'']\}$，其在区间抽象域上的接合定义为 $B' \sqcup_I B'' = \{x \in [\min(\underline{b}', \underline{b}''), \max(\overline{b}', \overline{b}'')]\}$。给定两个区间多面体 $\mathbf{P}_1$ 与 $\mathbf{P}_2$，$BB(\gamma(\mathbf{P}_1) \cup \gamma(\mathbf{P}_2)) = BB(\mathbf{P}_1) \sqcup_I BB(\mathbf{P}_2)$。

定义 4.3.4 (弱接合) 给定两个区间多面体 $\mathbf{P}_1$ 与 $\mathbf{P}_2$，我们定义区间多面体域上的弱接合操作为

$$\mathbf{P}_1 \sqcup_w \mathbf{P}_2 \stackrel{\text{def}}{=} env(\mathbf{P}_1, \mathbf{P}_2) \sqcap (\overline{env}(\mathbf{P}_1) \uplus \overline{env}(\mathbf{P}_2)) \sqcap (BB(\mathbf{P}_1) \sqcup_I BB(\mathbf{P}_2)).$$

注意弱接合操作可能在结果中引入冗余约束，但是大部分冗余约束可以通过语法方法消除。

例 4.3.4 考虑两个区间多面体 $\mathbf{P}_1 = \{[-1,1]x+2y \leq 2, -2x-y \leq 2, x-y \leq 1, -y \leq 0\}$（对应图 *4.2(1.d)* 中 x 轴的上方区域）和 $\mathbf{P}_2 = \{[-1,1]x-2y \leq 2, 2x+y \leq 2, -x+y \leq 1, y \leq 0\}$（对应图 *4.2(1.d)* 中 x 轴的下方区域）。$env(\mathbf{P}_1, \mathbf{P}_2) = \{[-1,1]x+2y \leq 2, [-1,1]x-2y \leq 2\} = \{[-1,1]x+2y = [-2,2]\}$，$\overline{env}(\mathbf{P}_1) \uplus \overline{env}(\mathbf{P}_2) = \{-2x-y \leq 2, x-y \leq 1, -y \leq 0\} \uplus \{2x+y \leq 2, -x+y \leq 1, y \leq 0\} = \{2x+[-2,1]y = [-2,2]\}$（使用带乘子的区间组合和语法冗余消除）。因此，$\mathbf{P}_1 \sqcup_w \mathbf{P}_2 = \{[-1,1]x+2y = [-2,2], 2x+[-2,1]y = [-2,2]\}$，其弱解集合如图 *4.2(1.d)* 所示。

定理 4.5 (弱接合的可靠性) 给定两个区间多面体 $\mathbf{P}_1$ 与 $\mathbf{P}_2$，$\mathbf{P}_1 \sqcup_w \mathbf{P}_2$ 的结果区间多面体是 $\mathbf{P}_1$ 与 $\mathbf{P}_2$ 的上近似，即 $\forall x.(x \in \gamma(\mathbf{P}_1) \vee x \in \gamma(\mathbf{P}_2)) \Rightarrow x \in \gamma(\mathbf{P}_1 \sqcup_w \mathbf{P}_2)$。

上述弱接合可以构造出一些被输入区间多面体满足但是不被它们凸闭包满足的约束（如，例 4.3.4 中有 $\gamma(\mathbf{P}_1 \sqcup_w \mathbf{P}_2) = \gamma(\mathbf{P}_1) \cup \gamma(\mathbf{P}_2)$）。然而，当区间多面体 $\mathbf{P}_1$ 与 $\mathbf{P}_2$ 都是同一象限内的凸多面体时，$\mathbf{P}_1 \sqcup_w \mathbf{P}_2$ 就没有 $\mathbf{P}_1$ 与 $\mathbf{P}_2$ 的凸闭包那么精确。这种情况下，可以使用多面体凸闭包来作为接合操作。例如，给定 x-y 平面中 $(+,+)$ 象限上两个点 $(0,0)$ 与 $(1,1)$，$\sqcup_w$ 只能给出 $\{0 \leq x \leq 1, 0 \leq y \leq 1\}$, 该结果没有两者多面体凸闭包的结果（即 $\{0 \leq x \leq 1, y = x\}$）那么精确。

(5) 迁移函数

条件测试迁移函数. 条件测试 $[\![\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b]\!]^\sharp(\mathbf{P})$ 的结果是在区间多面体 $\mathbf{P}$ 中加入约束 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b$，其中 $[\![\cdot]\!]^\sharp(\mathbf{P})$ 表示一个程序语义动作在区间多面体 $\mathbf{P}$ 上的应用效果。更为复杂的情况，比如包含析取、非线性或浮点表达式的条件测试可以可靠地抽象成 $\sum_k [\underline{a}_k, \overline{a}_k] \times x_k \leq b$ 形式（将在第 4.3.3 节详细讨论）。注意，加入一条新的约束可能引入冗余约束（新加入的约束可能是冗余的，或者可能导致原来非冗余约束变成冗余）或者使得区间多面体变得不可行。因此，在条件测试迁移函数执行后，需要进行冗余约束消除和空区间多面体测试。

两区间多面体 $\mathbf{P}_1$ 与 $\mathbf{P}_2$ 的交，记作 $\mathbf{P}_1 \sqcap \mathbf{P}_2$，定义为 $\mathbf{P}_1$ 与 $\mathbf{P}_2$ 约束系统的合取所对应的区间多面体，因此有 $\gamma(\mathbf{P}_1 \sqcap \mathbf{P}_2) = \gamma(\mathbf{P}_1) \cap \gamma(\mathbf{P}_2)$。类似地，两个区间多面体的交的结果区间多面体可能不可行，或者可能引入冗余约束。

赋值迁移函数. 把某个表达式 e 赋给变量 x_j 可以通过条件测试、投影和重命名来建模：

$$[\![x_j := e]\!]^\sharp(\mathbf{P}) \stackrel{\text{def}}{=} (\text{IFME}([\![x'_j - e = 0]\!]^\sharp(\mathbf{P}), x_j))[x'_j/x_j] \tag{4.1}$$

其中，新鲜变量 x'_j 引入进来是用于保存表达式 e 的值，这尤其对于不可逆赋值语句如 $x := [-1,2]x + [2,3]$ 来说很有必要。

特别地，当 e 中 x_j 的系数不为 0 且不含 0 时，赋值迁移函数还可以通过一种替代的方式来实现。设 $e=\sum_k[\underline{d}_k,\overline{d}_k]x_k+[\underline{c},\overline{c}]$，其中 $0\notin[\underline{d}_j,\overline{d}_j]$。那么替代方式实现的赋值迁移函数可以定义为

$$[\![x_j:=e]\!]^\sharp(\mathbf{P})\stackrel{\text{def}}{=}\left\{[\underline{a}'_j,\overline{a}'_j]x_j+\sum_{k\neq j}[\underline{a}'_k,\overline{a}'_k]x_k\leq b'\mid(\sum_k[\underline{a}_k,\overline{a}_k]x_k\leq b)\in\mathbf{P}\right\}\quad(4.2)$$

其中 $[\underline{a}'_j,\overline{a}'_j]=\dfrac{[\underline{a}_j,\overline{a}_j]}{[\underline{d}_j,\overline{d}_j]}$, $[\underline{a}'_k,\overline{a}'_k]=\left([\underline{a}_k,\overline{a}_k]-\dfrac{[\underline{d}_k,\overline{d}_k]}{[\underline{d}_j,\overline{d}_j]}\right)$ 且 $b'=sup\left(b+\dfrac{[\underline{c},\overline{c}]}{[\underline{d}_j,\overline{d}_j]}\right)$。

在多数情况下，方法 (4.2) 比 (4.1) 得到的结果更精确。例如，给定一个区间多面体 $\mathbf{P}=\{[-1,1]x\leq-1\}$，对于赋值语句 $x:=-x$，方法 (4.1) 得到的结果是整个全局空间，而方法 4.2 将得到区间多面体 $\mathbf{P}'=\{[-1,1]x\leq-1\}$。

(6) 包含测试

区间多面体上的序关系 $\sqsubseteq$ 定义为 $\mathbf{P}_1\sqsubseteq\mathbf{P}_2$ 当且仅当 $\gamma(\mathbf{P}_1)\subseteq\gamma(\mathbf{P}_2)$，即 $\forall\varphi\in\mathbf{P}_2.\mathbf{P}_1\models\varphi$。这可以通过区间线性规划 ILP 来实现：对于所有 $(\sum_k[\underline{a}_k,\overline{a}_k]\times x_k\leq b)\in\mathbf{P}_2$ 检查 $\mu\leq b$ 是否成立，其中 $\mu=\max\sum_k[\underline{a}_k,\overline{a}_k]\times x_k$ subject to $\mathbf{P}_1$。但是，这种方法的计算代价较高。为此，我们基于语法表示为区间多面体定义一个近似序关系 $\sqsubseteq_s$。首先，给定一对约束 $\varphi\colon\Sigma_k[\underline{a}_k,\overline{a}_k]\times x_k\leq b$, $\varphi'\colon\Sigma_k[\underline{a}'_k,\overline{a}'_k]\times x_k\leq b'$，我们定义 $\varphi\sqsubseteq_s\varphi'$ 当且仅当 $b\leq b'$ 且 $\forall k.[\underline{a}_k,\overline{a}_k]\subseteq[\underline{a}'_k,\overline{a}'_k]$。接下来，给定两区间多面体 $\mathbf{P}_1$ 和 $\mathbf{P}_2$，我们定义 $\mathbf{P}_1\sqsubseteq_s\mathbf{P}_2$ 当且仅当对于所有 $\varphi_2\in\mathbf{P}_2$，存在 $\varphi_1\in\mathbf{P}_1$ 使得 $\varphi_1\sqsubseteq_s\varphi_2$。不难看出，$\mathbf{P}_1\sqsubseteq_s\mathbf{P}_2$ 蕴含 $\mathbf{P}_1\sqsubseteq\mathbf{P}_2$，反之则不一定成立。

(7) 加宽

类似凸多面体抽象域，区间多面体抽象域也不满足递增链条件。为此，需要引入加宽算子来保证不动点迭代的收敛性。我们定义区间多面体抽象域中的加宽算子如下:

定义 4.3.5 (加宽) *给定阈值 k 以及在第 i 次迭代中的两个区间多面体 $\mathbf{P}_1\sqsubseteq\mathbf{P}_2$，我们定义第 i 次迭代中的加宽为*

$$\mathbf{P}_1\triangledown_i^{[k]}\mathbf{P}_2\stackrel{\text{def}}{=}\begin{cases}\mathcal{S}_1\cup\mathcal{S}_2 & \text{if } i\leq k\\ \mathcal{S}_1 & \text{otherwise}\end{cases}$$

其中 $\mathcal{S}_1=\{\varphi_1\in\mathbf{P}_1\,|\,\mathbf{P}_2\models\varphi_1\}$, $\mathcal{S}_2=\{\varphi_2\in\mathbf{P}_2\,|\,\exists\varphi_1\in\mathbf{P}_1,\gamma(\mathbf{P}_1)=\gamma((\mathbf{P}_1\setminus\{\varphi_1\})\cup\{\varphi_2\})\}$。

$\mathcal{S}_1$ 保留 $\mathbf{P}_1$ 中稳定的不等式，而$\mathcal{S}_2$ 则用来添加 $\mathbf{P}_2$ 中的与 $\mathbf{P}_1$ 中某个不等式约束关于 $\mathbf{P}_1$ 相互冗余的不等式约束。注意，不像凸多面体抽象域那样把所有迭代中的加宽算子都定义成 $\mathcal{S}_1 \cup \mathcal{S}_2$，区间多面体抽象域上需要一个阈值 k 来保证上述加宽的收敛性。给定链 $(X_i)_{i\in N}$，定义递增链 $(Y_i)_{i\in N}$ 为 $Y_0 = X_0$ 且$Y_{i+1} = Y_i \triangledown_i^{[k]} X_{i+1}$，那么 $(Y_i)_{i\in N}$ 将在有穷次迭代后收敛。因为在第 k 次迭代后，Y_{j+1} 中的约束集合总会是 Y_j $(j > k)$ 中约束集合的子集。

4.3.3 应用

(1) 处理析取

本文将采用区间组合技术利用区间线性不等式来对线性约束的析取式进行抽象。一般情形下，给定一个 DNF 析取范式公式，每个 DNF 析取项可以看作是一个凸多面体。通过在区间多面体域上使用接合操作，这些凸多面体的析取可以抽象成一个区间多面体。另一方面，给定一个 CNF 合取范式公式，每个 CNF 合取项可以通过区间组合抽象成一个区间线性不等式，因此整个 CNF 公式就可以抽象成一个区间多面体。

例如，考虑图 4.3 中程序。在 ② 处，$-1 \le x \le 1$ 在整数域上的非为 $x \le -2 \vee -x \le -2$，可以被区间线性不等式 $[-1,1]x \le -2$ 精确表示。并且，使用区间多面体抽象域，我们可以证明在 ⑤ 处 $y = -1$ 成立，因此 $y \neq 0$。然而，凸多面体域只能得出在 ⑤ 处 $-1 \le y \le 0$ 成立，从而不能证明 $y \neq 0$。

```
int x, y;
if ( x ≥ −1 and x ≤ 1 ){
      y := x − 1; ①
} else {  ②
      y := x; ③
}; ④
if( x == 0 ) {
⑤   assert(y ≠ 0);
}
```

位置	凸多面体	区间多面体
①	$x - y = 1 \wedge -1 \le x \le 1$	$x - y = 1 \wedge -1 \le x \le 1$
②	$\top$	$[-1,1]x \le -2$
③	$y = x$	$y = x \wedge [-1,1]x \le -2$
④	$0 \le x - y \le 1$	$0 \le x - y \le 1$ $\wedge[-1,1]x + [0,1]y \le -1$ $\wedge x + [-1,0]y \le 1$
⑤	$x = 0 \wedge -1 \le y \le 0$	$x = 0 \wedge y = -1$

图 4.3 析取示例程序 program1 及产生的不变式

(2) 处理非线性表达式

Miné 建议采用一种区间线性化抽象技术来把任意表达式抽象成区间线性形式 $\Sigma_k[\underline{a}_k, \overline{a}_k] \times x_k + [\underline{c}, \overline{c}]$ [85]。然而，目前大部分数值抽象域如凸多面体抽象域，不能直接处理区间线性形式。因此，需要采用一个称为拟线性化的技术来把区间线性形式 $\Sigma_k[\underline{a}_k, \overline{a}_k] \times x_k + [\underline{c}, \overline{c}]$ 转化成拟线性形式 $\Sigma_k a_k \times x_k + [\underline{c}, \overline{c}]$ [42]。拟线性化过程可能导致精度损失。然而，使用区间多面体域，可以避免（至少延迟）这样的精度损失，因为区间多面体域直接支持区间线性形式的表示。

对于图 4.4 中所示程序，在对非线性表达式 $z * x + 1$ 进行抽象后，我们在 ① 处得到 $[-5, 5]x + y = 1$。如果使用凸多面体域，我们需要对 $[-5, 5]x + y = 1$ 进行拟线性化。其最佳拟线性化结果为 $5x - y \geq -21 \wedge 5x + y \geq -19$。注意，这里发生了一些精度损失，比如，点 $(0, 0)$ 满足 $5x - y \geq -21 \wedge 5x + y \geq -19$ 但是不满足 $[-5, 5]x + y = 1$。最后，使用区间多面体域，我们可以证明 ② 处满足 $x \geq 3$，而凸多面体域只能证明 ② 处满足 $x \geq -1$。

int x, y, z;
assume $-5 \leq z \leq 5$;
assume $x \geq -2$;
$y := z * x + 1$; ①
assume $y == -14$; ②

位置	凸多面体	区间多面体
①	$-5 \leq z \leq 5 \wedge x \geq -2$ $\wedge 5x - y \geq -21$ $\wedge 5x + y \geq -19$	$-5 \leq z \leq 5 \wedge x \geq -2$ $\wedge [-5, 5]x + y = 1$
②	$y = -14 \wedge -5 \leq z \leq 5$ $\wedge x \geq -1$	$y = -14 \wedge -5 \leq z \leq 5$ $\wedge x \geq 3$

图 4.4 非线性示例程序 program2 及产生的不变式

(3) 处理浮点算术

实际程序语言不是操作有理数或实数而是浮点数，但是浮点算术难以抽象。一个解决方法是把浮点表达式抽象为实数域上的带区间系数的线性表达式，并把舍入误差显式地表示出来 [84]。舍入误差是非线性的，但是能够通过区间来抽象。例如，浮点加法 $x + y$ 可以抽象成 $[1 - \varepsilon_{\text{rel}}, 1 + \varepsilon_{\text{rel}}] \times x + [1 - \varepsilon_{\text{rel}}, 1 + \varepsilon_{\text{rel}}] \times y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$，其中 ε_{rel} 表示相对误差，ε_{abs} 表示绝对误差，即对应浮点格式下的最小非零正数。例如，在 32 位单精度浮点格式中 $\varepsilon_{\text{rel}} = 2^{-23}, \varepsilon_{\text{abs}} = 2^{-149}$。这种浮点抽象方法也可以纳入到区间线性化框架下。因此，区间多面体域可用来直接分析进行了抽象后的浮点程序。

考虑图 4.5 中程序，由于变量 x 的取值范围没有限制，对程序中的浮点赋值语句 $y := -2 * x \oplus_{32,?} 1$ 与 $y := x \oplus_{32,?} 1$ 进行拟线性化后将得到 $y \leftarrow [-\infty, +\infty]$。因此，凸多面体域不能得到任何有用信息，而区间多面体域可以证明 ② 处满足 $0.49999988 \leq x \leq 1.00000024$，从而说明了程序中最后一条语句 $x := x \oplus_{32,?} 1$ 不会发生浮点上溢。

real x, y;
if ($brandom$) {
  $y := -2 * x \oplus_{32,?} 1$;
} else {
  $y := x \oplus_{32,?} 1$;
} ①
assume $y == 0$; ②
$x := x \oplus_{32,?} 1$;

位置	凸多面体	区间多面体
①	$\top$	$[0.99999988, 2.00000024]x + y \leq 1.00000012$ $\wedge [0.99999988, 2.00000024]x + y \geq 0.99999988$
②	$y = 0$	$y = 0 \wedge 0.49999988 \leq x \leq 1.00000024$

图 4.5 浮点示例程序 program3 及产生的不变式

4.4 单变量区间线性不等式域

区间抽象域 [38] 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。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。相对于后来出现的、能够表示变量间关系的抽象域而言，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。但是，通过某些策略（比如带阀值的加宽策略 [21]、策略迭代 [138] 等），区间抽象域的分析精度可以得到改进。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 [21, 22] 等。

同样地，区间抽象域只能表示单个变量上凸的值范围信息，因此也存在凸性局限性问题，可能影响到分析的精度。尤其，在程序分析中，区间对于除法表达式的处理至关重要。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：使用一个区间来对分母表

达式的值进行上近似，然后通过除以该区间将整个表达式抽象为（区间）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”误报（如图 4.1 中所给出的例子），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 $\frac{2*z}{x*y}$，如果分母表达式 $x*y$ 的值范围是 $[2,4]$，则除法表达式 $\frac{2*z}{x*y}$ 将抽象成 $\frac{2*z}{[2,4]}$，即 $[0.5,1]*z$，从而成为（区间）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 $\frac{2*z}{x*y}$ 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，需要设计数值抽象域尽可能地把 0 排除在值范围之外。

另外，在实际程序分析中，区间作为变量系数的情形很可能会出现。但是，传统区间抽象域（形如 $a \le x \le b$，即 $x \in [a,b]$）尚不支持使用区间作为变量的系数。注意，区间抽象域只允许区间出现在常量项。

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 上形如 $[a,b]x \le c$ 的关系，其中常数 a,b,c 将由静态分析器自动推导出来。直观上讲，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。通过采用弱解语义，该域也可以表示某类非凸甚至非连通性质。而且，该域只需要通过简单的区间算术就可以实现，并具有线性的时空复杂度。

(1) 域表示

根据弱解的定义，一个单变量区间线性不等式

$$\varphi : ([a,b]x \le c)$$

可以按如下方式进行约简：

$$\varrho(\varphi) \stackrel{\text{def}}{=} \begin{cases} 0 \le 1 & \text{if } 0 \in [a,b] \wedge c \ge 0 \\ ax \le c & \text{else if } x \ge 0 \vee (a \ge 0 \wedge c \ge 0) \vee (b \le 0 \wedge c \le 0) \\ bx \le c & \text{else if } x \le 0 \vee (a \ge 0 \wedge c \le 0) \vee (b \le 0 \wedge c \ge 0) \\ [a,b]x \le c & \text{otherwise} \end{cases}$$

其中，$0 \le 1$ 表示不等式 φ 是恒成立的，可以删除；$ax \le c$ 和 $bx \le c$ 通过分别除以 a 和 b 的绝对值可以进一步约简成 $x \le c$ 或 $-x \le c$ 形式；$[a,b]x \le c$（其中 $a < 0 < b$ 且 $c < 0$）通过除以 c 的绝对值可以进一步约简成 $[a',b']x \le -1$ 形式，其中 $a' < 0 < b'$。注意 $\varrho(\varphi)$ 是精确的。

对于任意变量 x，我们使用关于 x 的单变量区间线性系统 $\vec{\mathbf{a}} \cdot x \le \vec{c}$ 来表示变量 x 的可能取值范围，其中 $\vec{\mathbf{a}}$ 是一个区间向量，$\vec{c}$ 是一个实数向量。其弱解集合

$\Sigma_{\exists}(\vec{\mathbf{a}} \cdot x \leq \vec{c}) = \{x \in \mathbb{R} \mid \exists \vec{a} \in \vec{\mathbf{a}}.\vec{a} \cdot x \leq \vec{c}\}$ 称为一个扩展区间。通过约简操作 ϱ，一个扩展区间的约束表示（可以有任意多个约束）可以转化成如下正规型（至多三个约束）：

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

其中，$c_1 \in \mathbb{R} \cup \{-\infty\}, c_2 \in \mathbb{R} \cup \{+\infty\}, a, b \in \mathbb{R}, a < 0 < b$。若 $c_1 = -\infty$（$c_2 = +\infty$）则表示 $-x \leq -c_1$（$x \leq c_2$）恒成立，可以从约束系统中删除；若 $[a,b] = [-\infty, +\infty]$，本文表示 $[a,b]x \leq -1$ 恒成立，可以从约束系统中删除。给定一个扩展区间 $\mathbf{I}$，$\mathbf{I}$ 的上述正规型扩展区间记为 $\varrho(\mathbf{I})$。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。

不难看出，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甚至非连通的，但是它与数轴上每个半轴的交必然是一个（可能为空的凸的）区间。实际上，一个扩展区间可以看作是两个（可能为空）区间的集合并（注意本文使用集合并 $\cup$ 来表示析取 $\vee$），但是这两个区间必须分布在原点的两侧。比如，扩展区间 $[[-3,2],[-1,1]]$ 其实就是表示 $[-3,-1] \cup [1,2]$。

本文把单变量区间线性不等式抽象域简称为扩展区间域。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 $\{[[c_1,c_2],[a,b]] \mid c_1 \in \mathbb{R} \cup \{-\infty\}, c_2 \in \mathbb{R} \cup \{+\infty\}, a, b \in \mathbb{R}, c_1 \leq c_2, a < 0 < b\} \cup \{[[c_1,c_2],[-\infty,+\infty]] \mid c_1 \in \mathbb{R} \cup \{-\infty\}, c_2 \in \mathbb{R} \cup \{+\infty\}, c_1 \leq c_2\} \cup \{\bot_{ei}\}$。$ExtItvs$ 构成了一个完全格 $(ExtItvs, \sqsubseteq_{ei}, \sqcap_{ei}, \sqcup_{ei}, \bot_{ei}, \top_{ei})$。其中，$\top_{ei} = [[-\infty,+\infty],[-\infty,+\infty]]$ 表示整个实数 $\mathbb{R}$；$\bot_{ei}$ 表示空扩展区间。同时，定义 $[[c_1,c_2],[a,b]] = \bot_{ei}$，其中 $[c_1,c_2] = \bot_i$ 或 $c_1 > -\frac{1}{b} \wedge \frac{1}{a} > c_2$。具体函数 $\gamma_{ei} \in [ExtItvs \to \wp(\mathbb{R})]$ 定义为：$\gamma_{ei}(\bot_{ei}) = \emptyset, \gamma_{ei}([[c_1,c_2],[a,b]]) \overset{\text{def}}{=} \{x \in \mathbb{R} \mid -x \leq c_1, x \leq c_2, [a,b]x \leq -1\}$。抽象函数 $\alpha_{ei} \in [\wp(\mathbb{R}) \to ExtItvs]$ 定义为：$\alpha_{ei}(\emptyset) \overset{\text{def}}{=} \bot_{ei}$，$\alpha_{ei}(S) \overset{\text{def}}{=} [[c_1,c_2],[a,b]]$ 其中 $S \subseteq \mathbb{R}$ 是一

个实数集合，$[c_1, c_2] = [\min S, \max S]$ 且

$$[a,b] = \begin{cases} [-\infty, +\infty] & \text{if } 0 \in S \vee \{x \in S \mid x > 0\} = \emptyset \\ & \quad \vee \{x \in S \mid x < 0\} = \emptyset \\ \left[\frac{-1}{\min\{x \in S | x > 0\}}, \frac{-1}{\max\{x \in S | x < 0\}}\right] & \text{otherwise} \end{cases}$$

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 $[c_1, c_2] \subseteq [c_1', c_2'] \wedge [a, b] \subseteq [a', b']$。

(2) 域操作

由于一个扩展区间本质上是一个区间或者是分布在原点两侧的两个区间的集合并。因此，扩展区间抽象域上的域操作可以通过分情况讨论采用区间抽象域上对应的域操作来构造。并且，所构造出来的域操作也同样具有线性的时空复杂度 $O(n)$。这一构造过程并不复杂，为此，本文下面只给出扩展区间抽象域上的加宽和变窄操作，该域上其他域操作的实现方法请参见文 [145]。

类似于区间完全格，扩展区间所构成的完全格上也存在无穷递增链。因此，为了处理循环，我们需要设计加宽算子来保证不动点计算的终止性。并且可以设计变窄算子来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 $\triangledown_{ei}$ 定义为

$$\perp_{ei} \triangledown_{ei} \mathbf{I} = \mathbf{I} \triangledown_{ei} \perp_{ei} \stackrel{\text{def}}{=} \mathbf{I}$$

$$\mathbf{I} = \mathbf{I}' \triangledown_{ei} \mathbf{I}'' \stackrel{\text{def}}{=} [[c_1, c_2], [a, b]]$$

其中，

$$[c_1, c_2] = [c_1' \leq c_1'' \,?\, c_1' : -\infty,\ c_2' \geq c_2'' \,?\, c_2' : +\infty]$$

$$[a,b] = \begin{cases} [a', b'] & \text{if } [a', b'] \supseteq [a'', b''] \\ [-\infty, +\infty] & \text{otherwise} \end{cases}$$

扩展区间抽象域上的变窄 $\triangle_{ei}$ 定义为

$$\mathbf{I} = \mathbf{I}' \triangle_{ei} \mathbf{I}'' \stackrel{\text{def}}{=} [[c_1, c_2], [a, b]]$$

其中，

$$[c_1, c_2] = [c_1' = -\infty \,?\, c_1'' : c_1',\ c_2' = +\infty \,?\, c_2'' : c_2']$$

$$[a,b] = \begin{cases} [a'', b''] & \text{if } [a', b'] = [-\infty, +\infty] \\ [a', b'] & \text{otherwise} \end{cases}$$

4.5 实现及实验

(1) 区间多面体域的可靠浮点实现

本文采用双精度浮点数为本章提出的区间多面体抽象域开发了一个原型系统itvPol。整个 itvPol 原型系统都是通过向外舍入的区间算术来实现的（即，求上界向上舍入，求下界向下舍入）。这种向外舍入的区间算术保证了浮点实现的可靠性。并且，本文按照文献 [131, 108] 中的方法，基于 GNU 线性规划工具包 GLPK（GNU Linear Programming Kit）[165] 实现了一个基于浮点的区间线性规划 ILP 求解器。类似于第 3.3.3 节，浮点 ILP 求解器的可靠性也是通过严格线性规划技术 [107, 108] 来保证的。

同样地，本文也把 itvPol 适配到为数值抽象域的开发提供了通用接口的数值抽象域库 Apron[86] 中。并使用支持 Apron 库的静态分析工具 Interproc [87] 开展了一系列实验。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等）。

(2) 区间多面体域实验

为了评估 itvPol 的分析精度和计算效率，我们把 itvPol 得到的不变式及其性能和前面第 3 章的工作 FPPol（凸多面体域的可靠浮点实现）进行了比较。

首先，为了展示 itvpol 的表达能力，我们给出两个简单而又具有代表性的循环程序，如图 4.6 与图 4.7 所示，以及分析器产生的不变式。program4 是一个反复对变量 x 求相反数的循环，而 program5 的程序行为包括两个阶段：首先在内部循环中增加 y，然后在外部循环中增加 x。对于图 4.6 中程序 program4，itvpol 可以证明在 ① 处满足 $x = -1 \vee x = 1$，这比 FPPol 得到的结果不变式 $-1 \leq x \leq 1$ 要精确。对于图 4.7 中 program5，itvpol 可以证明在 ② 处满足 $-20 \leq y \leq -10 \vee y \geq 10$，而 FPPol 只能证明 $y \geq -20$。

real x, y;
$x := -1$;
while ($true$) do
① $x := -x$;
done;

位置	凸多面体	区间多面体
①	$-1 \leq x \leq 1$	$-1 \leq x \leq 1 \wedge [-1,1]x \leq -1$

图 4.6 取相反数循环程序 program4 及产生的不变式

```
int x, y;
x := 1;
y := −20;
while (x ≤ 9) do
①    x := x + 1;
     while (y ≤ 9) do
         y := y + 1;
     done;
done; ②
```

位置	凸多面体	区间多面体
①	$y \geq -20$	$y \geq -20$
	$\wedge 1 \leq x \leq 9$	$\wedge 1 \leq x \leq 9$
		$\wedge -x + [0,1]y \leq -2$
		$\wedge [-1,1]y \leq -10$
②	$y \geq -20 \wedge x \geq 10$	$y \geq -20 \wedge x \geq 10$
		$\wedge [-1,1]y \leq -10$

图 4.7 嵌套循环程序 program5 及产生的不变式

表 4.1 给出并比较了基于 itvPol 和 FPPol 两种抽象域的程序分析的实验结果。其中，测试程序中的 program1-5，对应于图 4.3-4.7 所示的例子，其他测试程序来自第 3.5 节采用的测试用例。

对于每个程序，“#vars” 表示程序中变量的总数，而“#±” 表示程序中无符号限制的变量的个数。对于同一个程序，INTERPROC 的加宽算子延迟参数（加宽算子应用之前的迭代次数）设置成相同的，并且本文实验中区间多面体域上加宽算子的定义 4.3.5 中的阈值 k 设置为 10。“#iter.” 栏给出了总的（递增）迭代次数。

不变式. “Res. Inv.” 栏比较了所获得的不变式：“>”（“<”）表示 itvpol 找到的不变式比 FPPol 更强（弱）。如果程序中存在无符号限制的变量，itvpol 常能够找到一些有意义的非凸不变式。如果程序中所有变量的符号都是固定的，那么大部分情形下 itvpol 产生的不变式不会比 FPPol 好，因为 itvpol 使用了弱接合 $\sqcup_w$，而符号固定的变量所构成的区间多面体实际上是凸多面体，此时 $\sqcup_w$ 比凸多面体域中的多面体凸闭包要弱。

性能. 表 4.1 在“t(ms)” 栏以毫秒为单位给出了程序分析所花费的时间，其实验平台为：Fedora Linux 操作系统，768MB 物理内存，Intel(R) Pentium(R) M 1.6 GHz 单核 CPU 处理器。从表 4.1 可以看到，相比 FPPol 而言，itvpol 的计算代价不是显得太高。其主要原因在于 itvpol 使用了弱接合 $\sqcup_w$。在某些情况下，如对于测试程序 ratelimiter_f 与 symmetricalstairs，itvpol 的性能优于 FPPol。“#lp”栏给出了 LP 查询（调用 GLPK）的统计次数。实验中，我们还发现使用 itvpol 时，在 LP 求解上花费的时间往往占用了总分析时间的的一半以上。

表 4.1 区间多面体域的实验结果

Program		*itvPol*			FPPol			Res.
name	#vars(#±)	#iter.	#lp	$t(ms)$	#iter.	#lp	$t(ms)$	Inv.
program1	2(2)	1	256	31	1	138	24	$>$
program2	3(3)	1	78	12	1	54	11	$>$
program3	2(2)	1	68	13	1	0	6	$>$
program4	1(1)	4	19	10	4	8	7	$>$
program5	2(1)	5	270	49	6	187	35	$>$
sequencewhiles	3(1)	9	368	61	9	237	46	$>$
ratelimiter_f	5(4)	4	5846	792	5	2966	1425	$>$
bubblesort	4(4)	8	845	123	8	646	101	$>$
maccarthy91	3(2)	4	609	83	4	442	63	$>$
heapsort	7(7)	4	1534	273	4	1929	374	$<$
symmetricalstairs	2(1)	5	245	45	6	469	78	$<$
ackerman	4(2)	4	883	127	6	1477	298	$<$

4.6 小结

本章把区间线性代数引入到静态分析中，提出了一个新的数值抽象域——区间多面体抽象域。该抽象域能够表示和操作带区间系数的线性约束，可用来推导程序中变量间的区间线性不等式关系（形如 $\Sigma_k[a_k, b_k] \times x_k \leq b$）。该域的表达能力比经典的凸多面体抽象域更强。并且，通过采用区间线性约束的弱解语义，使得该域具有一些良好的特征：能够天然地表达某类非凸（甚至非连通）性质，而不需要显式的析取表示。区间多面体域的域操作可以通过两个原子操作，即区间 Fourier-Motzkin 消除法和区间线性规划来构造。此外，我们还给出区间多面域的一些可能应用，比如，可以用来处理包含析取、非线性或浮点表达式的程序，以及受不精确性或不确定性影响的含区间输入数据的程序或模型。本文在区间多面体抽象域原型系统 itvPol 上的实验结果也说明了：itvPol 可以比经典的凸多面体抽象域发现更好的不变式，而不需要太高的计算代价；对于包含了无符号限制变量的程序，itvPol 常常能够发现一些有意义的非凸性质。

并作为一种简单形式，给出了单变量区间线性不等式抽象域，用来推导单个变量上的单变量区间线性不等式关系（形如 $[a, b] \times x \leq c$）。该域可以看成是经典区间

抽象域的区间系数扩展版本，并且也可以表达某类非凸（甚至非连通）性质。而且，该域只需要通过简单的区间算术就可以实现，是一种轻量级的抽象域，有着极高的计算效率。

第五章　行阶梯形区间线性等式抽象域

5.1　引言

早在 1976 年，Karr [39] 提出了一个多项式时间算法来发现程序变量间的仿射关系（$\sum_k a_k x_k = b$）。抽象解释框架下，该算法被理解成一个仿射等式抽象域。仿射等式抽象域的一个重要特征在于仿射等式所构成的格的高度是有穷的，因此不需要加宽算子来保证分析的终止性。这使得该抽象域特别适合于某些分析，例如关于仿射程序的精确的过程间分析 [167]。至今，仿射等式抽象域仍然是计算效率最高的关系型数值抽象域之一。

最近的一些相关工作 [168, 167, 169]，观察到仿射等式抽象域在实现方面所存在的一个问题：该域的有理数实现可能导致大系数问题。为了缓解该问题，本文试图使用浮点数来实现仿射等式域，正如第 3 章我们采用浮点数来实现凸多面体抽象域一样。然而，简单地把仿射等式域适配到浮点算术上，由于浮点舍入误差的存在，抽象域实现的可靠性及其分析精度都很难得到保证。例如，对仿射等式的首项系数规格化为 1 是仿射等式域上的一个常用操作。如果把等式 $3x + y = 1$ 中变量 x 的系数规格化为 1，在有理数上将得到 $x + \frac{1}{3}y = \frac{1}{3}$，但是其中的新系数 $\frac{1}{3}$ 不能被浮点数精确表达。为了保证可靠性，一种方法就是把该等式删掉。但是，这可能导致极大的精度损失。为此，一个很自然的方式就是是把仿射等式域扩展成带区间系数的域，通过使用区间来包罗那些不能通过浮点数精确表达的数。

另一方面，本文第 3 章的浮点多面体抽象域与第 4 章的区间多面体抽象域都具有最差情况下指数级的计算复杂度。并且，这两个域都依赖于线性规划求解器，但是在程序分析过程中当前的浮点线性规划求解器可能存在第 3.5 节所讨论的“数值不稳定性问题”，导致求解给定的线性规划问题，从而降低抽象域的性能。为此，本章将设计一个具有多项式时空复杂度的抽象域，并且其域操作的实现不依赖于线性规划。

此外，本文第 4 章的区间多面体抽象域虽然支持区间变量系数，但是并不支持无穷区间系数。然而，在实际程序分析中，无穷区间很可能作为变量的系数出现。比如，如果仅知道变量 x, y 的取值范围是 $x \in [1, +\infty], y \in [-\infty, +\infty]$，那么此时对乘法表达式 $x * y$ 进行抽象的最好的结果是 $[1, +\infty] \times y$。而且，在加宽操作执行后，某些变量的取值范围很有可能是有无穷上界或无穷下界。为此，本章将设计一个抽

象域来支持带无穷区间系数的区间线性约束。

本章提出一个关于区间线性等式的抽象域—— 行阶梯形区间线性等式抽象域（简称为 itvLinEqs 域），用来推导变量 $x_k(k=1,\ldots,n)$ 间形如 $\sum_k[a_k,b_k]\times x_k=[c,d]$ 的关系，其中常量 $a_k,b_k,c,d\in\mathbb{R}\cup\{-\infty,+\infty\}$ 将通过静态分析器自动推导得到。为了保证计算效率，本文不采用任意形式的区间线性等式系统，而是采用一种受限形式的区间线性等式系统，即行阶梯形区间线性等式系统。直观上讲，itvLinEqs 域是经典仿射等式抽象域（$\sum_k a_k x_k=b$）的区间系数扩展版本，是本文第 4 章的区间多面体抽象域（$\sum_k[a_k,b_k]x_k\le c$）的等式版本。

同样地，通过采用弱解语义， itvLinEqs 域也可以表达某类非凸甚至非连通性质。下面，我们通过图 5.1 中的示例程序来展示 itvLinEqs 在不变式产生方面的优势。在程序点 ① 处，传统的仿射等式域和凸多面体域都不会得到任何信息，而 itvLinEqs 在 ① 处将得到 $x+[-2,-1]y=1$，并证明 ② 处满足 $y=[-1,-0.5]$，从而表明语句 $y:=1/y+1$ 中既不会发生除零错又不会发生上溢出。

real x,y;
if *random*() {
  $x:=y+1$;
} else {
  $x:=2y+1$;
} ①
assume $x==0$; ②
$y:=1/y+1$;

位置	仿射等式/凸多面体域	itvLinEqs 域
①	$\top$(无信息)	$x+[-2,-1]y=1$
②	$x=0$	$x=0\wedge y=[-1,-0.5]$

图 5.1 行阶梯形区间线性等式抽象域的动机示例程序

本章的结构组织如下：5.2 节介绍带无穷区间系数的区间线性等式系统；5.3 节介绍行阶梯形区间线性等式抽象域的设计；5.4 节介绍行阶梯形区间线性等式抽象域的原型系统实现，给出并分析实验结果；5.5 节是对本章的小结。

5.2 带无穷区间系数的区间线性等式系统

在第 4.2 节中我们介绍了区间线性不等式系统。与之不同的是，本节将采用区间线性等式系统，且支持无穷区间（即含无穷端点的区间）作为变量系数。本章使用如下表示方法。设 $\mathbf{x}=[\underline{x},\overline{x}]$ 为一个由其端点 $\underline{x}\le\overline{x}$ 所构成的区间。设 $\mathbb{IR}$ 为所

有实数区间所构成的集合 $[\underline{a},\overline{a}]$，其中 $\underline{a},\overline{a}\in\mathbb{R}$。设 $\mathbb{IR}$ 为所有区间 $[\underline{a},\overline{a}]$ 所构成的集合，其中 $\underline{a}\in\mathbb{R}\cup\{-\infty\},\overline{a}\in\mathbb{R}\cup\{+\infty\}$。

设 $\underline{A}\in(\mathbb{R}\cup\{-\infty\})^{m\times n},\overline{A}\in(\mathbb{R}\cup\{+\infty\})^{m\times n}$ 为两个矩阵 $\underline{A}\le\overline{A}$，其中比较操作指逐元素比较。那么如下实数矩阵集合

$$\mathbf{A}=[\underline{A},\overline{A}]=\{A\in\mathbb{R}^{m\times n}\colon \underline{A}\le A\le\overline{A}\}$$

称为一个（扩展）**区间矩阵**，且矩阵 $\underline{A},\overline{A}$ 称为其上下界。一个**区间向量**是一个单列区间矩阵 $\mathbf{b}=\{b\in\mathbb{R}^m\colon \underline{b}\le b\le\overline{b}\}$，其中 $\underline{b}\in(\mathbb{R}\cup\{-\infty\})^m$，$\overline{b}\in(\mathbb{R}\cup\{+\infty\})^m$ 且 $\underline{b}\le\overline{b}$。

设 $\mathbf{A}$ 为一个大小为 $m\times n$ 的区间矩阵，$\mathbf{b}$ 为一个大小为 m 的区间向量。本文称如下系统

$$\mathbf{A}x=\mathbf{b}$$

为一个（扩展）**区间线性等式系统**，来表示所有线性不等式系统 $Ax=b$ 所构成的等式系统族，其中 $A\in\mathbf{A}$, $b\in\mathbf{b}$。

定义 5.2.1 (弱解) *向量 $x\in\mathbb{R}^n$ 称为区间线性等式系统 $\mathbf{A}x=\mathbf{b}$ 的一个**弱解**，若存在某个 $A\in\mathbf{A}$, $b\in\mathbf{b}$ 使得 x 满足 $Ax=b$。且集合*

$$\Sigma_{\exists\exists}(\mathbf{A},\mathbf{b})=\{x\in\mathbb{R}^n:\exists A\in\mathbf{A},\exists b\in\mathbf{b}.\,Ax=b\}$$

*称为系统 $\mathbf{A}x=\mathbf{b}$ 的**弱解集合**。*

如下定理从代数角度刻画了区间线性等式系统的弱解集合 $\Sigma_{\exists\exists}(\mathbf{A},\mathbf{b})$。

定理 5.1 *设 $\sum_{j=1}^n[\underline{A}_{ij},\overline{A}_{ij}]x_j=[\underline{b}_i,\overline{b}_i]$ 为 $\mathbf{A}x=\mathbf{b}$ 的第 i 行。向量 $x\in\mathbb{R}^n$ 是 $\mathbf{A}x=\mathbf{b}$ 的一个弱解，当且仅当如下线性不等式*

$$\begin{cases}\quad\sum_{j=1}^n A'_{ij}x_j\le\quad\overline{b}_i\\ -\sum_{j=1}^n A''_{ij}x_j\le-\underline{b}_i\end{cases}$$

对于所有的 $i=1,\ldots,m$ 成立，其中 A'_{ij},A''_{ij} 定义如下

$$A'_{ij}=\begin{cases}\underline{A}_{ij}, & \text{if } x_j>0\\ 0, & \text{if } x_j=0\\ \overline{A}_{ij}, & \text{if } x_j<0\end{cases}\qquad A''_{ij}=\begin{cases}\overline{A}_{ij}, & \text{if } x_j>0\\ 0, & \text{if } x_j=0\\ \underline{A}_{ij}, & \text{if } x_j<0\end{cases}$$

定理 5.1 可以从文 [129]（中的 Theorem 2.1）得到，这里我们进行了扩展使其支持带无穷区间系数的情形。注意，对于定理 5.1 中的线性不等式 $\sum_{j=1}^n A'_{ij}x_j\le\overline{b}_i$，其中的每个项 $A'_{ij}x_j$ 不会是 $+\infty$，因为 $A'_{ij}=-\infty$ 只会在 $x_j>0$ 时才可能成立而 $A'_{ij}=+\infty$ 只会在 $x_j<0$ 时才可能成立。因此，$+\infty$ 不会出现在 $\le$ 的左边，从而不会出现矛盾。当其中任意一项 $A'_{ij}x_j$ 的结果为 $-\infty$ 时，线性不等式 $\sum_{j=1}^n A'_{ij}x_j\le\overline{b}_i$ 表示全局空间，可以从系统中忽略掉。对于 $-\sum_{j=1}^n A''_{ij}x_j\le-\underline{b}_i$，也有同样的结论。

一个(闭) 象限是 n 维欧几里得空间中 2^n 子集中的一个，通过限制每个笛卡尔坐标轴为非负或非正。注意，在一个给定的象限内，x 保持固定符号，这样弱解集合 $\Sigma_{\exists\exists}(\mathbf{A},\mathbf{b})$ 与每个象限的交可通过一个未必封闭的凸多面体来表示。事实上，这种可能的非封闭性以一种很受限的形式存在，使其封闭只会增加一些满足某些 $x_j=0$ 的点。特别地，若 $\mathbf{A}\in\mathbb{IR}^{m\times n}$，则每个闭象限内的区域是一个封闭的凸多面体。总体来说，弱解集合 $\Sigma_{\exists\exists}(\mathbf{A},\mathbf{b})$ 可以是非凸的 甚至非连通的。比如，$[-1,1]x=1$ 描述集合 $\{x\mid x\in[-\infty,-1]\cup[1,+\infty]\}$。

例 5.2.1 给定 $[-\infty,+\infty]x=1$，根据定理 *5.1*，

$$[-\infty,+\infty]x=1\Leftrightarrow\begin{cases}\{(-\infty)x\le 1,-(+\infty)x\le -1\}\Leftrightarrow\{-\infty\le 1,-\infty\le -1\}, & \text{if } x>0\\ \{(+\infty)x\le 1,-(-\infty)x\le -1\}\Leftrightarrow\{-\infty\le 1,-\infty\le -1\}, & \text{if } x<0\\ \{0x\le 1,-0x\le -1\}\Leftrightarrow\{0\le 1,0\le -1\}, & \text{if } x=0\end{cases}$$

在 $x>0$ 或 $x<0$ 时，$[-\infty,+\infty]x=1$ 定义了全局空间，而当 $x=0$ 时，$[-\infty,+\infty]x=1$ 定义了空集。因此，总体上，$[-\infty,+\infty]x=1$ 表示 $x\neq 0$ 。

5.3 行阶梯形区间线性等式域

本节提出一个新的数值抽象域 —— **行阶梯形区间线性等式域**，简记为 itv-LinEqs。其主要思想是使用行阶梯形的区间线性等式系统来作为域表示方法。抽象域中每个域元素的语义通过对应约束系统的弱解集合来定义。下面，本节将在实数 $\mathbb{R}$ 上来介绍行阶梯形区间线性等式域的域表示以及域操作的实现方法。后面，在第 5.4 节中我们再讨论该抽象域的可靠浮点实现方法。

5.3.1 域表示

约束的规范化. 本文中，我们固定如下变量序 $x_1\prec x_2\prec\ldots\prec x_n$。区间线性等式约束 $\Sigma_k[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}]$ 称为是一个恒成立的约束，若 $[\underline{b},\overline{b}]$=$[-\infty,+\infty]$ 或 $0\in[\underline{b},\overline{b}]\wedge\forall k.\,0\in[\underline{a}_k,\overline{a}_k]$。方便起见，我们使用 $\Sigma_k[0,0]x_k=[0,0]$ 作为恒成立约束的规范型。设 φ 为一个非恒成立的约束 $\Sigma_k[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}]$。其首项变量 x_i 是一个使得 $[\underline{a}_i,\overline{a}_i]\neq[0,0]$ 成立的具有最小下标 i 的变量。φ 称为是规范化的，若其首项变量 x_i 的区间系数满足 $[\underline{a}_i,\overline{a}_i]\in\{[0,1],[0,+\infty],[1,c],\ [-1,c'],[-\infty,+\infty]\}$，其中 $c,c'\in\mathbb{R}\cup\{+\infty\},c\ge 1,c'>0$。给定非规范化约束 φ，其规范形式可以通过把整个约束 φ 除以 -1（若 $[\underline{a}_i,\overline{a}_i]=[-\infty,0]$），$\pm\underline{a}_i$（若 $\underline{a}_i\notin\{0,-\infty\}$），或 $\pm\overline{a}_i$（若 $\overline{a}_i\notin\{0,+\infty\}$）来得到。

行阶梯形式. 设 $\mathbf{A}x=\mathbf{b}$ 为一个区间线性等式系统，其中 $\mathbf{A}\in\mathbb{IE}^{m\times n},\mathbf{b}\in\mathbb{IE}^m$。系

统 $\mathbf{A}x = \mathbf{b}$ 满足**行阶梯形**，若

- $m = n$，且
- 要么 x_i 是系统第 i 行的首项变量，要么第 i 行全是 0。

itvLinEqs 元素. itvLinEqs 域中的每个域元素 $\mathbf{P}$ 通过一个行阶梯形式的区间线性等式系统 $\mathbf{A}x = \mathbf{b}$ 来描述，其中 $\mathbf{A} \in \mathbb{IE}^{n\times n}, \mathbf{b} \in \mathbb{IE}^n$。其语义对应集合 $\gamma(\mathbf{P}) = \Sigma_{\exists\exists}(\mathbf{A}, \mathbf{b}) = \{x \in \mathbb{R}^n : \exists A \in \mathbf{A}, \exists b \in \mathbf{b}.\, Ax = b\}$，其中每个点 x 代表一个可能的程序环境（即对所有变量 x 的一种可能赋值）。图 5.2 给出了二维平面上 itvLinEqs 元素的一些例子。这些元素分别对应如下区间线性等式系统：(a) $\{[-1,1]x + [1,2]y = [2,4]\}$；(b) $\{[1,2]x + [1,2]y = [2,4], [1,1]y = [-3,3]\}$；(c) $\{[1,+\infty]x + [1,1]y = [1,1]\}$；(d) $\{[-1,1]x + [1,1]y = [0,1], [-1,1]y = [-0.5,0.5]\}$；(e) $\{[1,2]x + [1,2]y = [2,4], [-1,1]y = [-0.5,0.5]\}$；(f) $\{-1,1]x = [-1,-1], [1,1]y = [1,1]\}$。

itvLinEqs 域元素的几何拓扑性质与第 4 章介绍的区间多面体非常类似，都可以表达某类非凸甚至非连通性质，而且与每个闭象限的交都是一个（可能为空的）凸多面体。比如，图 4.2 中子图 (2.a)-(2.e) 所示的非区间多面体区域也同样不能被 itvLinEqs 域表达。但是，两者表达能力也有所区别。所有 itvLinEqs 元素可表达的拓扑封闭集合都可以被区间多面体所表达，如图 5.2 中除 (c) 子图之外的所有子图。但是，区间多面体的约束系统可以是任意形式，从而可以表达一些 itvLinEqs 元素不能表达的拓扑封闭集合，如图 4.2 中的子图 (1.d) 和 (1.e)。另一方面，itvLinEqs 元素可表达的拓扑非封闭集合则不能被区间多面体所精确表达，如图 5.2 中的 (c) 子图。

行阶梯抽象. 一个仿射等式系统可以通过基本的矩阵变换等价地转化为行阶梯形式。然而，并非所有的区间线性等式系统都可以精确地表达成行阶梯形式。设 $\mathbf{P}$ 为任意形式的区间线性等式系统。我们考虑找一个处于行阶梯形的系统 $\rho(\mathbf{P})$ 使得 $\gamma(\mathbf{P}) \subseteq \gamma(\rho(\mathbf{P}))$。但是，行阶梯抽象 $\rho(\mathbf{P})$ 不一定是惟一的，而且最佳抽象可能不存在。$\mathbf{P}$ 的行阶梯抽象 $\rho(\mathbf{P})$ 可以通过第 5.3.2 节将介绍的“约束添加”操作来构造，即通过把 $\mathbf{P}$ 中每个约束一条一条地“添加”到一个初始化为空的行阶梯系统。

不难看出，行阶梯抽象可能导致精度损失。尽管如此，在 itvLinEqs 域中我们仍然要求每个域元素满足行阶梯形。因为行阶梯形可以使得在 itvLinEqs 域中构造出具有多项式时间复杂度的域操作，并且可以避免“指数级增长”问题（例如，不会产生指数级输出）[82]。而且，行阶梯形式仍然足以精确地表示程序中的仿射空间（即仿射等式系统）。此外，行阶梯形式也使得我们可以通过借鉴已有仿射等式抽象域

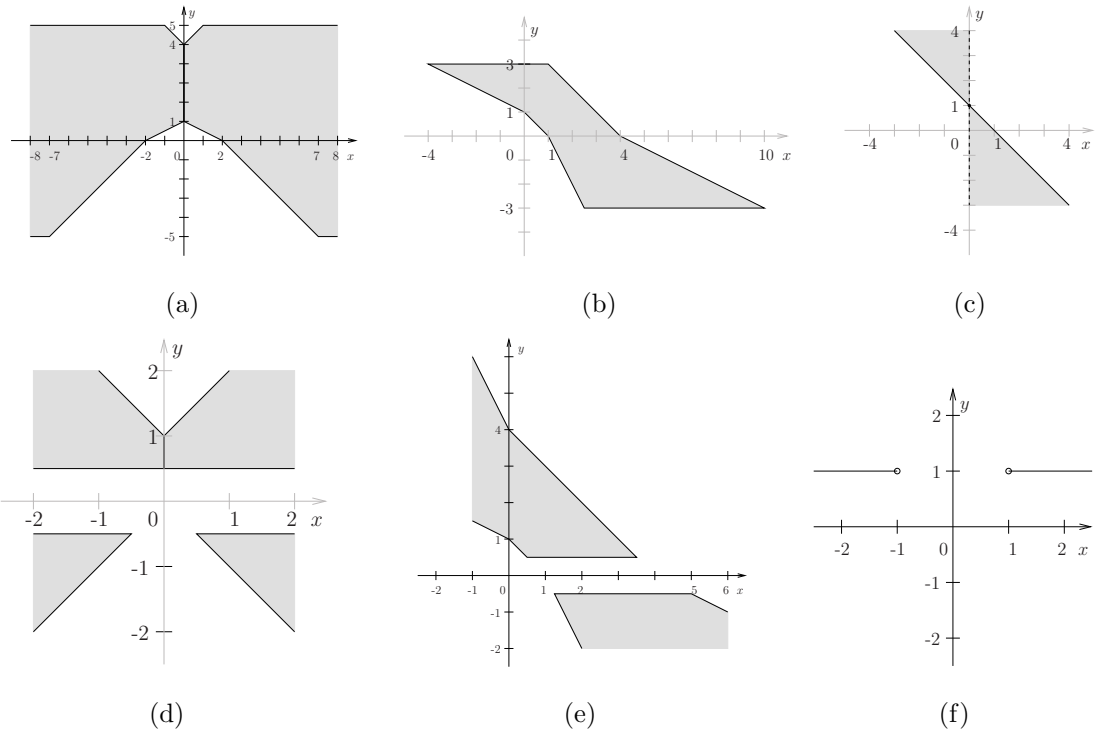


图 5.2 行阶梯形区间线性等式域中域元素的几何示例

的实现框架，就可以比较容易地构造出 itvLinEqs 域的实现算法。

5.3.2 域操作

itvLinEqs 维护了一个行阶梯形等式系统，从而其域操作可以通过借鉴已有仿射等式抽象域的实现框架来构造。下面，我们将详细描述行阶梯形区间线性等式抽象域上用于静态分析所需要的常用域操作的实现方法。

(1) 约束比较

为了在抽象域中维护行阶梯形式，对于行阶梯形的每一行比如第 i 行，我们需要对若干个候选约束进行比较并从中选择最好的一个来作为行阶梯形的第 i 行。我们首先使用一些启发式的度量来评估规范化约束 φ 中所含信息的精度，从而可以对多个约束进行排序。

定义 5.3.1 给定规格化约束 $\varphi : (\sum_k [\underline{a}_k, \overline{a}_k] x_k = [\underline{b}, \overline{b}])$ 及变量 x 的包围盒 $x \in [\underline{x}, \overline{x}]$，度量 $f_{weight}(\varphi), f_{width}(\varphi) \in \mathbb{R} \cup \{+\infty\}, f_{mark}(\varphi) \in \mathbb{R}$ 分别定义如下:

1. $f_{weight}(\varphi) \stackrel{\text{def}}{=} \sum_k (\overline{a}_k - \underline{a}_k) \times (\overline{x}_k - \underline{x}_k) + (\overline{b} - \underline{b})$,
2. $f_{width}(\varphi) \stackrel{\text{def}}{=} \sum_k (\overline{a}_k - \underline{a}_k) + (\overline{b} - \underline{b})$,
3. $f_{mark}(\varphi) \stackrel{\text{def}}{=} \sum_k \delta(\underline{a}_k, \overline{a}_k) + \delta(\underline{b}, \overline{b})$, 其中

$$\delta(\underline{d},\overline{d}) \stackrel{\text{def}}{=} \begin{cases} -1 & \text{if } \underline{d}=\overline{d} \\ +200 & \text{else if } \underline{d}=-\infty \wedge \overline{d}=+\infty \\ +100 & \text{else if } \underline{d}=-\infty \vee \overline{d}=+\infty \\ 0 & \text{otherwise} \end{cases}$$

定义 5.3.2 (约束比较) 给定两个规格化约束 φ 和 φ'，我们定义 $\varphi \preceq \varphi'$，如果 $(f_{weight}(\varphi), f_{width}(\varphi), f_{mark}(\varphi)) \leq (f_{weight}(\varphi'), f_{width}(\varphi'), f_{mark}(\varphi'))$ 在字典序意义下成立。

具体而言，f_{weight} 同时考虑了区间系数的宽度信息和变量的界信息；f_{width} 只考虑了区间系数的宽度信息，因为某些变量的界可能是无穷的（此时 $f_{weight} = +\infty$）；f_{mark} 为那些有无穷区间系数的约束打分（此时 $f_{weight} = f_{width} = +\infty$）。在这种意义下，可以保证每个仿射等式总比其他类型的约束 $\preceq$ 要小。例如，$(x+y=1) \preceq (x+y=[1,2]) \preceq (x+y=[1,+\infty])$。方便起见，我们称一个非恒成立的约束 φ 比 φ' 较好，若 φ' 是一个恒成立约束或者 $\varphi \preceq \varphi'$ 成立。（若 $\varphi \preceq \varphi'$ 且 $\varphi' \preceq \varphi$，我们根据上下文从 φ, φ' 中选择其中之一作为两者中较好的那一个。）约束比较操作的时间复杂度为 $O(n)$。

(2) 投影

在程序分析中，投影是一个重要的操作，对于构造赋值迁移函数和过程间分析都很重要。在 itvLinEqs 中，投影操作对于约束添加操作和接合操作的实现也非常有用。下面，我们使用 $\{\boxplus, \boxminus, \boxtimes, \boxslash\}$ 来表示区间算术操作。

首先，我们引入一个部分线性化操作 $\zeta(\varphi, x_j)$ 来把约束 φ 中变量 x_j 的区间系数线性化为标量。

定义 5.3.3 (部分线性化算子) 设 φ: $(\sum_k [\underline{a}_k, \overline{a}_k] \times x_k = [\underline{b}, \overline{b}])$ 为一个区间线性等式，变量 x_j 的界为 $[\underline{x}_j, \overline{x}_j]$。我们定义部分线性化算子为

$$\zeta(\varphi, x_j) \stackrel{\text{def}}{=} \left(c \times x_j + \sum_{k \neq j} [\underline{a}_k, \overline{a}_k] \times x_k = [\underline{b}, \overline{b}] \boxminus [\underline{a}_j - c, \overline{a}_j - c] \boxtimes [\underline{x}_j, \overline{x}_j]\right)$$

其中 c 为任意实数常量。

虽然理论上 c 可以为任意实数常量，但是选择一个合适的 c 可以导致较少的精度损失，但是如何选择一个合适的 c 依赖于 $\underline{a}_j, \overline{a}_j, \underline{x}_j, \overline{x}_j$ 的值。实践中，当 $[\underline{a}_j, \overline{a}_j]$ 为有穷时，我们常选择区间 $[\underline{a}_j, \overline{a}_j]$ 的中点 $c = (\underline{a}_j + \overline{a}_j)/2$，大部分情形下这可以使得精度损失较少。若区间 $[\underline{a}_j, \overline{a}_j]$ 只有一个端点是无穷的时，我们选择另一个有穷端点作为 c。若 x_j 有无穷界时，我们在 $c = \underline{a}_j$ 与 $c = \overline{a}_j$ 所得结果中选择关于 $\preceq$ 较好的那个约束。

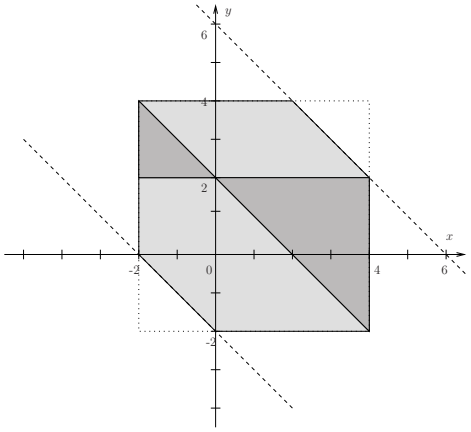


图 5.3 部分线性化的几何示例

例 5.3.1 考虑区间线性等式 $\varphi:([0,2]x+y=2)$ 及变量的包围盒 $x,y\in[-2,4]$。若选择 $[0,2]$ 的中点作为 c，$\zeta(\varphi,x)$ 将得到 $x+y=[-2,6]$。注意，这里发生了精度损失，比如，点$(0,0)$ 满足结果约束 $x+y=[-2,6]$ 但是并不满足原区间线性等式 $[0,2]x+y=2$。图 *5.3* 给出了包围盒内，原区间线性等式 $[0,2]x+y=2$ 弱解集合所对应的图形区域（深灰色表示）及部分线性化 $\zeta(\varphi,x)$ 所得结果 $x+y=[-2,6]$ 所对应的图形区域（淡灰色表示）。

定理 5.2 (部分线性化算子的可靠性) 给定一个区间线性等式 φ 以及一个变量 $x_i\in[\underline{x}_i,\overline{x}_i]$，$\zeta(\varphi,x_i)$ 是 φ 的上近似，即 $\forall x.(x_i\in[\underline{x}_i,\overline{x}_i]\wedge x\in\gamma(\varphi))\Rightarrow x\in\gamma(\zeta(\varphi,x_i))$。

接下来，我们考虑如何从一个约束或一对约束中投影掉某个变量 x_j。

基于界的投影. 从约束 φ: $(\sum_k[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}])$ 中投影掉 x_j，我们可以简单地在 $\zeta(\varphi,x_j)$ 中选择 $c=0$。所得结果 $\sum_{k\neq j}[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}]\boxminus[\underline{a}_j,\overline{a}_j]\boxtimes[\underline{x}_j,\overline{x}_j]$ 将是 φ 的上近似，且不再含 x_j。

基于组合的投影. 设 φ: $(\sum_k[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}])$ 和 φ': $(\sum_k[\underline{a}'_k,\overline{a}'_k]x_k=[\underline{b}',\overline{b}'])$ 为两个分别满足 $[\underline{a}_j,\overline{a}_j]\neq[0,0]$ 和 $[\underline{a}'_j,\overline{a}'_j]\neq[0,0]$ 的约束。为了计算一个不含 x_j 并满足 $\gamma(\varphi)\cup\gamma(\varphi')\subseteq\gamma(\phi)$ 的约束 ϕ，我们采用一种类似于高斯消去法的方法来实现，并通过如下两个步骤来实现：

- 首先，我们把 φ 中变量 x_j 的区间系数转化为 1 （如，通过 $\zeta(\varphi,x_j)$ 并选择 $c=1$）。假设我们得到 $x_j+\sum_{k\neq j}[\underline{a}''_k,\overline{a}''_k]x_k=[\underline{b}'',\overline{b}'']$；
- 然后，通过在 φ' 中把 x_j 用（$[\underline{b}'',\overline{b}'']-\sum_{k\neq j}[\underline{a}''_k,\overline{a}''_k]x_k$）来替代，从而可以得到如下不含变量 x_j 的约束

$$\phi:\left(\sum_{k\neq j}([\underline{a}'_k,\overline{a}'_k]\boxminus[\underline{a}'_j,\overline{a}'_j]\boxtimes[\underline{a}''_k,\overline{a}''_k])x_k=[\underline{b}',\overline{b}']\boxminus[\underline{a}'_j,\overline{a}'_j]\boxtimes[\underline{b}'',\overline{b}'']\right)。$$

特别地，当 $0 \notin [\underline{a}_j, \overline{a}_j]$ 时，把 φ 中的 $[\underline{a}_j, \overline{a}_j]$ 转换为 1 的步骤还可以通过如下定理来实现。

定理 5.3 设 φ 为 $(\sum_k [\underline{a}_k, \overline{a}_k] x_k = [\underline{b}, \overline{b}])$，其中 $0 \notin [\underline{a}_j, \overline{a}_j]$。那么有

$$\varphi'' : x_j + \sum_{k \neq j} ([\underline{a}_k, \overline{a}_k] \boxslash [\underline{a}_j, \overline{a}_j]) x_k = [\underline{b}, \overline{b}] \boxslash [\underline{a}_j, \overline{a}_j]$$

是 φ 的一个上近似，即 $\gamma(\varphi) \subseteq \gamma(\varphi'')$。

证明. 因为 $0 \notin [\underline{a}_j, \overline{a}_j]$，那么有

x^* 是区间线性等式 $\varphi : (\sum_k [\underline{a}_k, \overline{a}_k] x_k = [\underline{b}, \overline{b}])$ 的一个弱解

$\iff \exists a_k^* \in [\underline{a}_k, \overline{a}_k], b^* \in [\underline{b}, \overline{b}].\ \sum_k a_k^* x_k^* = b^*$

$\iff \exists a_k^* \in [\underline{a}_k, \overline{a}_k], b^* \in [\underline{b}, \overline{b}].\ x_j^* + \sum_{k \neq j} (a_k^*/a_j^*) x_k^* = b^*/a_j^*$ 〈因为 $a_j^* \neq 0$〉

$\implies x^*$ 是 $\varphi'' : (x_j + \sum_{k \neq j} ([\underline{a}_k, \overline{a}_k] \boxslash [\underline{a}_j, \overline{a}_j]) x_k = [\underline{b}, \overline{b}] \boxslash [\underline{a}_j, \overline{a}_j])$ 的一个弱解

〈因为 $(a_k^*/a_j^*) \in ([\underline{a}_k, \overline{a}_k] \boxslash [\underline{a}_j, \overline{a}_j])$ 且 $(b^*/a_j^*) \in ([\underline{b}, \overline{b}] \boxslash [\underline{a}_j, \overline{a}_j])$〉 □

把 φ 中 x_j 的区间系数 $[\underline{a}_j, \overline{a}_j]$ 转换为 1，定理 5.3 中的“除”方法不依赖于 x_j 的界但是要求 $0 \notin [\underline{a}_j, \overline{a}_j]$，而 $\zeta(\varphi, x_j)$ 中选择 $c = 1$ 的方法更通用但是依赖于 x_j 的界。这两种方法都可能导致精度损失。实践中，大部分情况下，定理 5.3 得到的结果更精确，尤其当 x_j 的界不是很精确甚至无穷时。例如，给定 $\varphi : ([1, 2]x + y = 2)$ 及界信息 $x, y \in [-\infty, +\infty]$，为了把 x 的系数转化为 1，$\zeta(\varphi, x_j)$ 中选择 $c = 1$ 的方法将得到恒成立约束，但是定理 5.3 将得到 $\varphi'' : x + [0.5, 1]y = [1, 2]$。注意这里发生了精度损失，例如，点 $(0, 1)$ 满足 φ'' 但是不满足 φ。

itvLinEqs 上的投影. 上面，我们已经给出了方法以从一个约束或一对约束中投影掉某个变量。接下来，我们采用算法 1 从 itvLinEqs 域元素 $\mathbf{P}$（即整个约束系统）中投影掉变量 x_j。我们记 $\mathbf{P}$ 的第 i 行对应的约束为 $\mathbf{P}_i$。对于每一行，我们尝试各种消除方法（基于界的或者基于组合的），最终只保留关于 $\preceq$ 的最佳约束作为该行约束。例如，从 $\mathbf{P} = \{x - y = 0, [-1, 1]y = [2, +\infty]\}$ 中投影掉 y 将得到 $\mathbf{P}' = \{[-1, 1]x = [2, +\infty]\}$。算法 1 的最差时间复杂度为 $O(n^3)$。

(3) 约束添加

现在，我们考虑如何把一个新的约束 $\varphi : (\sum_k [\underline{a}_k, \overline{a}_k] x_k = [\underline{b}, \overline{b}])$ “添加”到一个 itvLinEqs 元素 $\mathbf{P}$ 中，记作 $[\![\varphi]\!]^{\#}(\mathbf{P})$，即导出一个行阶梯抽象 $\mathbf{P}'$ 使得 $\gamma(\mathbf{P}) \cap \gamma(\varphi) \subseteq \gamma(\mathbf{P}')$。设 x_i 为 φ 的首项变量。首先，我们把 $\mathbf{P}'$ 初始化为 $\mathbf{P}$。然后，把新约束 φ 与 $\mathbf{P}$ 的第 i 行 φ'（即 $\varphi' = \mathbf{P}_i$）进行比较。

算法 1: PROJECT$(\mathbf{P}, x_j)$

输入: $\mathbf{P}$：一个 itvLinEqs 域元素 $\mathbf{A}x = \mathbf{b}$

x_j：待投影变量

输出: $\mathbf{P}'$: 一个不含 x_j 且满足 $\gamma(\mathbf{P}) \subseteq \gamma(\mathbf{P}')$ 的 itvLinEqs 域元素

```
1  begin
2      P' ← P;
3      for i ← 1 to j − 1 do
4          if [A_ij, Ā_ij] ≠ [0, 0] then
5              φ ← ζ(P'_i, x_j) with c = 0;   // 基于界的投影
6              for k ← i + 1 to j do
7                  if [A_kj, Ā_kj] ≠ [0, 0] then
8                      φ' ← 组合 P'_i 与 P'_k 消除掉 x_j 所得最佳约束;
9                      if φ' ⪯ φ then φ ← φ';
10             P'_i ← φ;   // φ 是首项变量为 x_i 且不含 x_j 的最佳约束
11     P'_j ← [0, 0]^{1×(n+1)};
12     return P';
13 end
```

- 若 φ 与 φ' 都是线性条纹且两者是互相平行的，即 $\forall k.\underline{a}_k = \overline{a}_k = \underline{a}'_k = \overline{a}'_k$，$\mathbf{P}_i$ 将被更新为 $\sum_k [\underline{a}'_k, \overline{a}'_k] x_k = [\max\{\underline{b}, \underline{b}'\}, \min\{\overline{b}, \overline{b}'\}]$。若 $\max\{\underline{b}, \underline{b}'\} > \min\{\overline{b}, \overline{b}'\}$，则 $\mathbf{P}'$ 是不可行的。
- 否则，我们选择 φ 和 φ' 中关于 $\preceq$ 最好的约束来替代 $\mathbf{P}'_i$。然后，我们把 φ 与 φ' 进行组合以消除 x_i，并把结果约束 φ'' 递归地加入到更新后的 $\mathbf{P}'$。由于被加入的约束的首项变量的下标将严格增加，当 φ'' 变成恒成立约束时，该递归过程将终止。

约束添加操作可以用来构造一些操作，比如条件测试迁移函数、行阶梯抽象、求交等。两个 itvLinEqs 元素 $\mathbf{P}$ 与 $\mathbf{P}'$ 的交，记作 $\mathbf{P} \sqcap_w \mathbf{P}'$，可以通过把 $\mathbf{P}'$ 中的约束（从第一行到最后一行）逐个添加到 $\mathbf{P}$ 中来实现。注意 $\gamma(\mathbf{P}) \cap \gamma(\mathbf{P}') \subseteq \gamma(\mathbf{P} \sqcap_w \mathbf{P}')$ 总成立，反之未必成立。约束添加操作的时间复杂度为 $O(n^2)$，而 $\mathbf{P} \sqcap_w \mathbf{P}'$ 的时间复杂度为 $O(n^3)$。

(4) 接合

为了对控制流接合进行抽象，我们需要设计一个接合操作来返回一个 itvLinEqs 域元素，使得该元素在几何上包含了两输入 itvLinEqs 元素。然而，itvLinEqs 域上并不存在最佳接合操作来计算包含了输入参数的最小 itvLinEqs 元素。为此，我们设计一个轻量级的接合操作，称为弱接合，来作为 itvLinEqs 域上的接合操作。特别地，对于仿射等式而言，该弱接合操作可以精确地计算出输入元素所在仿射空间的仿射包（affine hull），从而不会在仿射等式关系上造成精度损失。

近似凸组合.

首先，我们回顾线性代数上的两个概念：给定一组向量 $x_1,\ldots,x_n$，

$$\Sigma_{i=1}^{n}\lambda_i x_i \text{ 其中 } \Sigma_{i=1}^{n}\lambda_i = 1$$

称为这些向量的仿射组合（affine combination）；

$$\Sigma_{i=1}^{n}\lambda_i x_i \text{ 其中 } \Sigma_{i=1}^{n}\lambda_i = 1, \forall i.\lambda_i \geq 0$$

称为这些向量的凸组合（convex combination）。

在仿射等式域 [39] 中，其接合操作（仿射包）可以通过仿射组合来构造；在凸多面体域 [40] 中，其接合操作（凸闭包）可以通过凸组合来构造。基于上述观察，我们按照类似的思想来为 itvLinEqs 域设计接合操作。

给定两个 itvLinEqs 元素 $\gamma(\mathbf{P}) = \{x \,|\, \mathbf{A}x = \mathbf{b}\}$ 和 $\gamma(\mathbf{P}') = \{x \,|\, \mathbf{A}'x = \mathbf{b}'\}$，基于分别来自 $\mathbf{P}$ 的任意点（满足 $\mathbf{A}z = \mathbf{b}$ 的 z）与来自 $\mathbf{P}'$ 的任意点（满足 $\mathbf{A}'z' = \mathbf{b}'$ 的 z'）的凸组合（$x = \sigma_1 z + \sigma_2 z'$ 其中 $\sigma_1 + \sigma_2 = 1, \sigma_1 \geq 0, \sigma_2 \geq 0$），我们定义如下点集合：

$$\gamma(\mathbf{P}) \uplus \gamma(\mathbf{P}') = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \exists \sigma_1, \sigma_2 \in \mathbb{R}, z, z' \in \mathbb{R}^n. \\ x = \sigma_1 z + \sigma_2 z' \wedge \sigma_1 + \sigma_2 = 1 \wedge \sigma_1 \geq 0 \wedge \\ \quad \mathbf{A}z = \mathbf{b} \quad \wedge \quad \mathbf{A}'z' = \mathbf{b}' \quad \wedge \sigma_2 \geq 0 \end{array} \right\}$$

不难看出，$\gamma(\mathbf{P}) \uplus \gamma(\mathbf{P}')$ 是 $\mathbf{P}$（$\sigma_1 = 1$ 时）与 $\mathbf{P}'$（$\sigma_2 = 1$ 时）并的上近似。为了消除非线性项 $\sigma_1 z$ 与 $\sigma_2 z'$，我们引入新的向量 $y = \sigma_1 z$ 及 $y' = \sigma_2 z'$，并把上述系统松弛成

$$\left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \exists \sigma_1, \sigma_2 \in \mathbb{R}, y, y' \in \mathbb{R}^n. \\ \quad x = y + y' \;\wedge \sigma_1 + \sigma_2 = 1 \;\wedge \sigma_1 \geq 0 \wedge \\ \mathbf{A}y = \sigma_1 \mathbf{b} \;\wedge\; \mathbf{A}'y' = \sigma_2 \mathbf{b}' \;\wedge \sigma_2 \geq 0 \end{array} \right\}$$

并进一步重写为

$$\left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \exists \sigma_1 \in \mathbb{R}, y \in \mathbb{R}^n. \\ \mathbf{A}'x - \mathbf{A}'y + \mathbf{b}'\sigma_1 = \mathbf{b}' \quad \wedge \\ \mathbf{A}\,y - \mathbf{b}\,\sigma_1 = 0 \quad \wedge \\ \sigma_1 = [0,1] \end{array} \right\} \tag{5.1}$$

从而，得到了关于变量序 $x_1 \prec \ldots \prec x_n \prec y_1 \prec \ldots \prec y_n \prec \sigma_1$ 的行阶梯形式区间线性等式系统。采用 itvLinEqs 域上的投影操作 Project（见算法 1），从系统 (5.1) 中按 $y_1, \ldots, y_n, \sigma_1$ 顺序投影掉 y, σ_1 就可以得到一个 itvLinEqs 元素，记作 $\mathbf{P} \uplus_w \mathbf{P}'$。从而，有

$$\gamma(\mathbf{P}) \cup \gamma(\mathbf{P}') \subseteq \gamma(\mathbf{P}) \uplus \gamma(\mathbf{P}') \subseteq \gamma(\mathbf{P} \uplus_w \mathbf{P}').$$

注意，$\mathbf{P} \uplus_w \mathbf{P}'$ 是通过 itvLinEqs 域上的投影操作 Project计算得到的，从而一定是一个 itvLinEqs 域元素，而 $\gamma(\mathbf{P}) \uplus \gamma(\mathbf{P}')$ 是一个通过精确存在量词定义的点集，但是该点集未必是一个 itvLinEqs 域元素。

$\mathbf{P} \uplus_w \mathbf{P}'$ 可以精确地计算出输入参数所在仿射空间的仿射包，因为根据约束比较 $\preceq$ 的定义，仿射等式与其他类型的约束比较时总是会优先保留仿射等式。此外，$\mathbf{P} \uplus_w \mathbf{P}'$ 也能产生其他类型的有意义的区间线性约束，比如线性条纹（形如 $\sum_k a_k x_k = [\underline{b}, \overline{b}]$），来填充那些没有仿射关系成立的行。与多面体凸闭包不同的是，凸闭包得到总是凸的结果而且具有最差情况下指数级的时间复杂度，而 $\mathbf{P} \uplus_w \mathbf{P}'$ 可以在多项时间 $O(n^4)$ 内计算出来，并且能够产生非凸约束（当然也可能丢失一些凸闭包所能产生的线性不等式）。

区间组合.

类似于第 4.3.2 节，我们在区间线性等式约束上定义如下区间组合操作。

定义 5.3.4 (区间组合) 给定两个区间线性等式约束 $\varphi'\colon (\sum_k [\underline{a}'_k, \overline{a}'_k] \times_k = [\underline{b}', \overline{b}'])$ 与 $\varphi''\colon (\sum_k [\underline{a}''_k, \overline{a}''_k] \times x_k = [\underline{b}'', \overline{b}''])$, φ' 与 φ'' 的区间组合定义为

$$\varphi' \uplus \varphi'' \stackrel{\text{def}}{=} \left(\sum_k [\min(\underline{a}'_k, \underline{a}''_k), \max(\overline{a}'_k, \overline{a}''_k)] \times x_k = [\min(\underline{b}', \underline{b}''), \max(\overline{b}', \overline{b}'')] \right)$$

该定义可以直接提升到 itvLinEqs 域元素上。给定两个 itvLinEqs 域元素 $\mathbf{P}'$ 与 $\mathbf{P}''$，其区间组合定义为 $\mathbf{P} = \mathbf{P}' \uplus \mathbf{P}''$ 其中 $\mathbf{P}_i = \mathbf{P}'_i \uplus \mathbf{P}''_i$ $(i = 1, \ldots, n)$。

定理 5.4 (区间组合的可靠性) 给定两个区间线性等式 φ' 与 φ''，其区间组合 $\varphi' \uplus \varphi''$ 是 φ' 和 φ'' 的可靠上近似，即 $\gamma(\varphi') \cup \gamma(\varphi'') \subseteq \gamma(\varphi' \uplus \varphi'')$。

定理 5.4 蕴含了 itvLinEqs 域元素上 $\uplus$ 的可靠性，即 $\gamma(\mathbf{P}')\cup\gamma(\mathbf{P}'')\subseteq\gamma(\mathbf{P}'\uplus\mathbf{P}'')$。区间组合的精度依赖于输入的语法表示。为了得到一个经验上可预计的且更精确的结果，我们在区间组合之前先把输入规范化。

弱接合.

下面，我们为 itvLinEqs 域定义如下弱接合操作。

定义 5.3.5 (弱接合) 给定两个 itvLinEqs 元素 $\mathbf{P}$ 和 $\mathbf{P}'$，我们定义 itvLinEqs 域上的弱接合操作为

$$\mathbf{P}\sqcup_w\mathbf{P}'\stackrel{\text{def}}{=}(\mathbf{P}\Cup_w\mathbf{P}')\sqcap_w(\mathbf{P}\uplus\mathbf{P}').$$

注意，$\mathbf{P}\uplus\mathbf{P}'$ 的时间复杂度为 $O(n^2)$，$\mathbf{P}\sqcup_w\mathbf{P}'$ 的时间复杂度为 $O(n^4)$。

例 5.3.2 给定两个 itvLinEqs 域元素 $\mathbf{P}=\{I=2,J-K=5,[-1,1]K=1\}$ 和 $\mathbf{P}'=\{I=3,J-K=8,[-1,4]K=2\}$，那么有 $\mathbf{P}\Cup_w\mathbf{P}'=\{3I-J+K=1,J-K=[5,8]\},\mathbf{P}\uplus\mathbf{P}'=\{I=[2,3],J-K=[5,8],[-1,4]K=[1,2]\}$，从而有 $\mathbf{P}\sqcup_w\mathbf{P}'=\{3I-J+K=1,J-K=[5,8],[-1,4]K=[1,2]\}$。然而，如果只考虑输入参数所在仿射空间（即 $\{I=2,J-K=5\}$ 与 $\{I=3,J-K=8\}$）的接合操作，仿射等式域中仿射包的计算结果为 $\{3I-J+K=1\}$，而凸多面体域中凸闭包的计算结果为 $\{3I-J+K=1,J-K=[5,8]\}$。从而，在本例中，$\mathbf{P}\sqcup_w\mathbf{P}'$ 的结果更精确。

定理 5.5 (弱接合的可靠性) 给定两个 itvLinEqs 域元素 $\mathbf{P}$ 与 $\mathbf{P}'$，其弱接合 $\mathbf{P}\sqcup_w\mathbf{P}'$ 是 $\mathbf{P}$ 与 $\mathbf{P}'$ 的上近似，即 $\gamma(\mathbf{P})\cup\gamma(\mathbf{P}')\subseteq\gamma(\mathbf{P}\sqcup_w\mathbf{P}')$。

(5) 赋值迁移函数

把某个区间线性表达式 e 赋给变量 x_j 可以通过条件测试、投影和重命名来在时间 $O(n^3)$ 内建模：

$$[\![x_j:=e]\!]^{\#}(\mathbf{P})\ \stackrel{\text{def}}{=}(\textsc{Project}([\![x'_j-e=0]\!]^{\#}(\mathbf{P}),x_j))[x'_j/x_j]\ .$$

其中，新鲜变量 x'_j 引入进来是用于保存表达式 e 的值，这尤其对于不可逆赋值语句如 $x:=[-1,1]x+1$ 来说很有必要。赋值迁移函数 $[\![x_j:=e]\!]^{\#}(\mathbf{P})$ 可以在 $O(n^3)$ 时间复杂度内计算得到。

(6) 蕴含测试

itvLinEqs 域上的最佳序关系 $\sqsubseteq$ 可以定义为 $\mathbf{P}\sqsubseteq\mathbf{P}'$ 当且仅当 $\gamma(\mathbf{P})\subseteq\gamma(\mathbf{P}')$。但是，$\sqsubseteq$ 计算代价很高且需要区间线性规划技术。为此，我们基于语法表示为 itvLinEqs 域定义一个近似序关系 $\sqsubseteq_s$。首先，给定一对约束 φ: $\Sigma_k[\underline{a}_k,\overline{a}_k]x_k=[\underline{b},\overline{b}]$ 与

$\varphi'\colon \Sigma_k[\underline{a}'_k,\overline{a}'_k]x_k=[\underline{b}',\overline{b}']$，我们定义 $\varphi\sqsubseteq_s\varphi'$ 当且仅当 $[\underline{b},\overline{b}]\subseteq[\underline{b}',\overline{b}']$ 且 $\forall k.[\underline{a}_k,\overline{a}_k]\subseteq[\underline{a}'_k,\overline{a}'_k]$。接下来，给定两个 itvLinEqs 域元素 $\mathbf{P}$ 与 $\mathbf{P}'$，我们定义 $\mathbf{P}\sqsubseteq_s\mathbf{P}'$ 当且仅当对于 $\mathbf{P}'$ 中的每行 $\mathbf{P}'_i$，要么 $\mathbf{P}'_i$ 是一个恒成立的约束或者 $\mathbf{P}_i\sqsubseteq_s\mathbf{P}'_i$。不难看出，$\mathbf{P}\sqsubseteq_s\mathbf{P}'$ 蕴含 $\mathbf{P}\sqsubseteq\mathbf{P}'$，反之未必成立。$\mathbf{P}\sqsubseteq_s\mathbf{P}'$ 的时间复杂度为 $O(n^2)$。

(7) 加宽

不像仿射等式域，itvLinEqs 域不满足递增链条件。因此，为了处理循环，需要引入加宽算子来保证不动点迭代的收敛性。

为此，我们首先定义区间线性等式约束上的加宽算子。

定义 5.3.6 给定两个区间线性等式 $\varphi':(\sum_k[\underline{a}'_k,\overline{a}'_k]x_k=[\underline{b}',\overline{b}'])$ 和 $\varphi'':(\sum_k[\underline{a}''_k,\overline{a}''_k]x_k=[\underline{b}'',\overline{b}''])$，我们定义约束 φ' 和 φ'' 上的加宽为

$$\varphi'\triangledown_{row}\varphi''\stackrel{\text{def}}{=}\left(\sum_k([\underline{a}'_k,\overline{a}'_k]\triangledown_{itv}[\underline{a}''_k,\overline{a}''_k])x_k=([\underline{b}',\overline{b}]\triangledown_{itv}[\underline{b}'',\overline{b}''])\right)$$

其中 $\triangledown_{itv}$ 是区间抽象域 [38] 上的加宽操作，如:

$$[\underline{a},\overline{a}]\triangledown_{itv}[\underline{b},\overline{b}]=[\underline{a}\leq\underline{b}\,?\,\underline{a}:-\infty,\ \overline{a}\geq\overline{b}\,?\,\overline{a}:+\infty]$$

接下来，我们定义 itvLinEqs 域上的加宽操作。

定义 5.3.7 (itvLinEqs 域上的加宽) 给定两个 itvLinEqs 域元素 $\mathbf{P}'\sqsubseteq\mathbf{P}''$，我们定义 itvLinEqs 域上的加宽为 $\mathbf{P}'\triangledown_{ile}\mathbf{P}''\stackrel{\text{def}}{=}\mathbf{P}$ 其中

$$\mathbf{P}_i=\begin{cases}\mathbf{P}''_i & \text{if } \mathbf{P}''_i \text{ is an affine equality}\\ \mathbf{P}'_i\triangledown_{row}\mathbf{P}''_i & \text{otherwise}\end{cases}$$

注意，在 $\mathbf{P}'\sqsubseteq\mathbf{P}''$ 不成立时，我们可以使用 $\mathbf{P}'\triangledown_{ile}(\mathbf{P}'\sqcup_w\mathbf{P}'')$。加宽 $\triangledown_{ile}$ 保留了 $\mathbf{P}''$ 中所有的仿射等式，因此不会在加宽过程中导致在仿射关系上发生精度损失。当第 i 行不存在仿射关系时，$\mathbf{P}'_i\triangledown_{row}\mathbf{P}''_i$ 通过捕获一对演进约束 $\mathbf{P}'_i$ 与 $\mathbf{P}''_i$ 之间的稳定信息来改进加宽算子的精度。加宽算子 $\triangledown_{ile}$ 的时间复杂度为 $O(n^2)$。

不难看出，加宽算子 $\triangledown_{ile}$ 满足 $\mathbf{P}'\sqsubseteq(\mathbf{P}'\triangledown_{ile}\mathbf{P}'')$ 且 $\mathbf{P}''\sqsubseteq(\mathbf{P}'\triangledown_{ile}\mathbf{P}'')$。直观上讲，加宽算子 $\triangledown_{ile}$ 的收敛性可以通过如下两个事实来保证：

1. 仿射等式构成的格的高度是有穷的，$\mathbf{P}''$ 中仿射等式的数目将一直减少直到等于程序中仿射空间的维数；
2. itvLinEqs 元素中区间系数（包括变量系数和常量系数）的数目是有穷的（至多 $\frac{1}{2}n(n+3)$），而在这些区间系数所对应位置上的区间加宽算子 $\triangledown_{itv}$ 可以保证非仿射等式约束部分的收敛性。

定理 5.6 (itvLinEqs 域上加宽算子 $\triangledown_{ile}$ 的终止性) 对于每个链 $(X^i)_{i\in\mathbb{N}}$，递增链 $(Y^i)_{i\in\mathbb{N}}$ 其中 $Y^0\stackrel{\text{def}}{=}X^0, Y^{i+1}\stackrel{\text{def}}{=}Y^i\triangledown_{ile}(Y^i\sqcup_w X^{i+1})$ 将在有穷时间内收敛。

证明. 首先，根据 $\triangledown_{ile}$ 的定义可以知道 $Y^i \sqsubseteq Y^{i+1}$。因此，序列 $(Y^i)_{i\in\mathbb{N}}$ 中的仿射空间在增长，从而 $(Y^i)_{i\in\mathbb{N}}$ 中的仿射等式数目在减少。因为仿射等式所构成的格的高度是有穷的，$(Y^i)_{i\in\mathbb{N}}$ 中仿射空间将在某个第 k 次（k 是有穷的）迭代中稳定下来，且 $(Y^i)_{i\geq k}$ 中仿射等式的数目即为程序所在的仿射空间的维数。注意，$(Y^i)_{i>k}$ 中的仿射等式在语法表示上可能跟 Y^k 中的仿射等式不一样，但是他们在语义上表示了相同的仿射空间且在 $(Y^i)_{i>k}$ 和 Y^k 中占据着相同的行。事实上，如果我们为仿射等式维护一个约简行阶梯形，那么 $(Y^i)_{i>k}$ 中的仿射等式在语法上也将与 Y^k 中的仿射等式一样。记 $\mathcal{I}$ 为 Y^k 中仿射等式所在行的下标集合。

在第 k 次迭代后，$\triangledown_{ile}$ 只会通过 $\triangledown_{row}$ 来改变那些下标不在 $\mathcal{I}$ 中的行。设 $m \in \{1,\ldots,n\}$ 且 $m \notin \mathcal{I}$。下面，考虑 $(Y^i)_{i>k}$ 中第 m 行所构成的序列 $(Y^i_m)_{i>k}$。设 $(\sum_{j=m}^{n}[\underline{A}^i_{mj},\overline{A}^i_{mj}]x_j = [\underline{b}^i_m,\overline{b}^i_m])_{i>k}$ 为 $(Y^i_m)_{i>k}$ 的第 m 行。那么，应用在 $([\underline{A}^i_{mj},\overline{A}^i_{mj}])_{i>k}$ 和 $([\underline{b}^i_m,\overline{b}^i_m])_{i>k}$ 序列上的区间域加宽算子 $\triangledown_{itv}$ 的收敛性，将保证 $(Y^i_m)_{i>k}$ 上加宽算子 $\triangledown_{row}$ 的收敛性。简而言之，对于每一行 $m \in \{1,\ldots,n\}$ 且 $m \notin \mathcal{I}$，序列 $(Y^i_m)_{i>k}$ 将在有穷时间内收敛。

因此，整个序列 $(Y^i)_{i\in\mathbb{N}}$ 将在有穷时间内收敛。 □

带阈值的加宽. 带阈值的加宽 [21] $\triangledown^T$ 是一个带参数的加宽，参数是一个由包含了 $-\infty$ 和 $+\infty$ 在内的有穷多个阈值所组成的有穷集合 T。区间抽象域上带阈值的加宽定义为：

$$\begin{aligned}[\underline{a},\overline{a}]\,\triangledown^T_{itv}\,[\underline{b},\overline{b}] = \;& [\underline{a}\leq\underline{b}\,?\,\underline{a} : \max\{\ell\in T \mid \ell\leq\underline{b}\},\\ & \overline{a}\geq\overline{b}\,?\,\overline{a} : \min\{h\in T \mid h\geq\overline{b}\}]\end{aligned}$$

但是，$\triangledown^T_{itv}$ 只是作用在单个变量上，通过阈值来导引单个变量上的加宽操作。

把 $\triangledown_{row}$ 中的 $\triangledown_{itv}$ 替换为 $\triangledown^T_{itv}$，我们就可以得到 itvLinEqs 域上的带阈值的加宽算子 $\triangledown^T_{ile}$。这一过程很自然地把带阈值的加宽策略从单个变量的取值范围提升到多个变量间的关系上。值得注意的是，$\triangledown^T_{ile}$ 不仅可以用来猜测常量项的上下界（如同在模版多面体域 [91] 的常量项上应用阈值策略），还可以用来猜测将来可能稳定的不变式的斜率（即 $\sum_k[\underline{a}_k,\overline{a}_k]x_k$ 部分）。

例 5.3.3 对于图 *5.4* 所给程序，在第一次迭代后，在程序点 ① 处加宽算子的输入参数为 $\varphi : ([1,1]x+[-0.75,-0.75]y=[1,1])$ 和 $\varphi' : ([1,1]x+[-1,-0.6875]y=[1,1.25])$。那么，$\varphi\triangledown_{row}\varphi'$ 结果为 $[1,1]x+[-\infty,+\infty]y=[1,+\infty]$。但是，如果我们使用 $\pm n\pm 0.5 (n\in\mathbb{N}, n\leq 2)$ 作为阈值集合 T，那么 $\varphi\triangledown^T_{row}\varphi'$ 将得到 $[1,1]x+[-1,-0.5]y=[1,1.5]$，而且该等式将在后续的迭代中稳定，从而是该程序的一个不变式。

```
real x, y;
x := 0.75 * y + 1;
while (true) {
① if random() {
        x := y + 1;
    } else {
        x := 0.25 * x + 0.5 * y + 1;
    }
}
```

图 5.4 带阈值的加宽策略示例程序

5.4 实现及实验

5.4.1 实现

与区间域的约简. 变量的界信息在 itvLinEqs 域中起着重要作用。例如，部分线性化（见定义 5.3.3）和约束比较（见第 5.3.2 节）都依赖于变量界。然而，itvLinEqs 域本身推导变量界信息的能力有限。为此，我们使用区间抽象域来维持界信息。

由于约简操作与加宽操作的交迭可能导致不收敛性问题 [41]，我们在区间域和 itvLinEqs 域之间只进行一个方向的约简，即从 itvLinEqs 域到区间域的约简。在某些域操作（如条件测试/赋值迁移函数、求交等）执行后，我们把信息从 itvLinEqs 域传播到区间域上来对变量的界进行缩紧。类似第 3.3.5 节，我们也可以通过约束传播技术来对界进行缩紧：每个约束都可以用来缩紧其中所含变量的界。

可靠浮点实现. 前面所介绍的 itvLinEqs 域都是在实数 $\mathbb{R}$ 和精确算术上考虑的。下面，我们考虑使用浮点数来实现 itvLinEqs 域。itvLinEqs 域上的域操作都是基于区间算术的，而而区间算术可以很容易地通过向外舍入的浮点区间算术来实现（即，求上界向上舍入，求下界向下舍入），并保证浮点实现的可靠性。

然而，itvLinEqs 域的浮点实现可能带来一些其他问题。首先，浮点 itvLinEqs 可能因为舍入误差而丢失一些仿射等式，从而浮点 itvLinEqs 所产生的不变式不再比精确的（有理数）仿射等式域所产生的不变式要强。在浮点计算里，把区间线性等式进行规范化也不再是精确的，比如对 $3x + y = 5$ 进行规范化。另外，基于浮点 itvLinEqs 的分析可能面临文 [21] 中所提到的关于浮点迭代的不稳定性问题。

然而，带阀值的加宽可以减轻该问题。比如，我们可以选择一些比较规整的阈值如$\pm 2^{\pm n}(n \in \mathbb{N})$，这样，乘以/除以该阀值就是简单的二进制位移操作，这在大多数情况下是精确的。

5.4.2 Interproc 实验

本文采用双精度浮点数为本章提出的行阶梯形区间线性等式抽象域开发了一个原型系统 FP-itvLinEqs。同样地，本文也把 FP-itvLinEqs 适配到为数值抽象域的开发提供了通用接口的数值抽象域库 Apron[86] 中。并使用支持 Apron 库的静态分析工具 Interproc [87] 开展了一系列实验。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等）。

为了评估 FP-itvLinEqs 的分析精度和计算效率，我们把 FP-itvLinEqs 得到的不变式及其性能和以下抽象域库进行了比较：

- NewPolka：基于多精度有理数的凸多面体域的实现库；
- polkaeq：基于多精度有理数的仿射等式域的实现库（事实上， polkaeq 是基于 NewPolka 来实现，而不是采用 Karr 算法 [39]，但是 polkaeq 与 Karr 算法的推导能力相同）；
- itvPol：本文第 4 章介绍的基于浮点数的区间多面体域的实现。

表 5.1 行阶梯形区间线性等式域关于仿射等式的实验结果

Program		FP-itvLinEqs				polkaeq			Res.
name	#var	#iter.	#=	#≃	t(ms)	#iter.	#=	t(ms)	Inv.
Karr1	3	4	1	1	13	4	1	8	>
Karr2	4	1	2	1	10	1	2	7	>
GS1	4	1	2	3	19	1	2	13	>
GS2	4	1	2	0	9	1	2	7	=
MOS1	6	8	1	1	66	8	1	33	>
MOS2	1	1	1	0	3	1	1	5	=
policy1	2	4	1	1	12	4	1	10	>

我们通过三个方面的实验来评价 FP-itvLinEqs。实验结果如表 5.1-5.3 所示。对于同一个程序，Interproc 的加宽算子延迟参数（加宽算子应用之前的迭代次数）设置成相同的。其中，“#iter.” 栏给出了总的（递增）迭代次数。栏“Res. Inv.”

比较了所得到的不变式: “>”（“<”，或“≠”）表示左边的分析得到的不变式比右边的分析要强（弱，或不可比）；“time” 或“t(ms)” 栏给出了程序分析所花费的时间，其实验平台为：Fedora Linux 操作系统，768MB 物理内存，Intel(R) Pentium(R) M 1.6 GHz 单核 CPU 处理器。

仿射等式实验. 首先，本文在一些用于发现仿射等式型不变式的测试用例上比较了 FP-itvLinEqs 和 polkaeq，其中大部分例子来自文 [39, 167, 168, 139]。表 5.1 总结了这些测试用例上的实验结果。其中，“#=”表示程序分析所发现的仿射等式的数目，“#≃”表示程序分析所发现的其他类型的约束的数目。对于这些测试程序，FP-itvLinEqs 可以发现所有 polkaeq 能够发现的仿射等式。因为这些程序只涉及小整数值，此时浮点计算在大部分情形下都是精确的，从而能够发现这些程序中所有的仿射等式。

表 5.2 行阶梯形区间线性等式域关于不等式的实验结果

Program		FP-itvLinEqs			NewPolka		itvPol			Res. Inv.	
name	#var	#≤	#≃	time	#≤	time	#≤	#≃	time		
policy2	2	3	1	20ms	2	22ms	3	0	46ms	>	>
policy3	2	2	2	18ms	2	20ms	2	2	49ms	>	<
policy4	2	3	1	19ms	1	24ms	2	1	59ms	>	≠
symmetrical stairs	2	3	0	33ms	3	31ms	2	0	45ms	<	>
maccarthy91	3	1	2	28ms	2	15ms	2	3	83ms	≠	<
bubblesort	4	3	3	87ms	2	58ms	1	3	123ms	>	≠
incdec	32	26	12	32s	×	>1h	×	×	>1h	>	>
mesh2X2	32	24	18	20s	×	>1h	5	3	190s	>	≠
bigjava	44	18	16	43s	×	>1h	6	4	1206s	>	≠

不等式实验. 第二部分测试用例用于发现不等式型不变式，主要来自本文第 3.5 节所采用的测试用例及文 [139, 91]，实验结果如表 5.2 所示。其中，“#≤” 表示程序分析所发现的线性不等式的数目（包括仿射等式和线性条纹在内，并按两个线性不等式来计算），“#≃”表示程序分析所发现的其他类型的约束。“Res. Inv.”的左子栏对 FP-itvLinEqs 和 NewPolka 进行了比较，而右子栏对 FP-itvLinEqs 和 itvPol 进行了比较。与 NewPolka 相比，大部分情形下，FP-itvLinEqs 得到了更精确的

结果，因为 FP-itvLinEqs 可以找到一些非凸区间线性不变式，使得每个程序点处所发现的不变式的可行空间比 NewPolka 更小。尤其，对于某些高维测试程序（如 incdec、mesh2X2、bigjava 等），NewPolka 不能在规定的时间（1 小时）内完成分析，本文使用“>1h”标记，而 FP-itvLinEqs 仍然可以在较短的时间内完成分析。与 itvPol 相比，FP-itvLinEqs 的计算效率明显较高。事实上，随着程序中变量数目的增多，两者在计算效率上的差别也越来越明显。另外，FP-itvLinEqs 还可以产生一些带无穷区间系数的不变式（例如，policy3 与 bubblesort 各产生了 2 条，incdec 与 mesh2X2 各产生了 5 条，bigjava 产生了 12 条），而这些不变式是 itvPol 所不能表达的。

阀值实验. 在表 5.3 中，我们比较了不带阀值加宽的 FP-itvLinEqs 和带阀值加宽的 FP-itvLinEqs。其中，Example5.3.3 对应于本文第 5.3.2 节的例 5.3.3。ratelimiter_f 对应于本文第 3.5 节图 3.10 所示的浮点速率限制器程序（其中，$M = 128$）。nonlinear 是一个包含了非线性表达式的程序。当使用带阈值的加宽时（其中阈值为 $\{\pm n \pm 0.5 \mid n \in \mathbb{N}, n \leq 150\}$），FP-itvLinEqs 可以发现更精确的或新的不变式，其数目在表 5.3 中“$\#newinv.$”栏列出。

表 5.3 行阶梯形区间线性等式域关于带阈值加宽策略的实验结果

Program		FP-itvLinEqs					Res.
		without thresholds		with thresholds			
name	#var	#iter.	t(ms)	#iter.	#newinv.	t(ms)	Inv.
Example5.3.3	2	4	12	4	1	18	$<$
ratelimiter_f	5	5	88	5	2	91	$<$
nonlinear	3	5	29	7	1	56	$<$

5.4.3 Astrée 实验

基于双精度浮点数实现的行阶梯形区间线性等式域原型系统 FP-itvLinEqs，已适配到商业化工具 Astrée 中，并在工业界航空领域嵌入式命令控制程序上开展了一系列实验 [1]。

[1]这部分实验是 Antoine Miné 使用 Astrée 在工业界代码上开展的，在此表示感谢。

(1) 静态分析工具 Astrée

Astrée [21, 22] 是一个基于抽象解释的静态分析工具，主要用于检查程序运行时错误。目前，Astrée 已在数值密集型嵌入式安全攸关软件的分析与验证中取得了成功应用。比如，Astrée 成功地验证了空客 A340（约 13.2 万行代码）、A380（约 35 万行代码）等系列大规模飞行控制软件，并实现了零误报。这些程序往往包含很多布尔、整数、浮点等相关的数值计算，从而数值抽象域在分析中起着重要作用。

Astrée 中使用到的数值抽象域包括

- 通用型抽象域：包括区间域、同余域等非关系型抽象域，以及八边形域等弱关系型抽象域；
- 领域相关抽象域：数字滤波器域、等差等比数列域等关系型抽象域，主要用来处理一些特定的非线性程序行为，比如二阶滤波器等。

为了保证分析的可扩展性，Astrée 把变量集合划分成多个组，每个组称为一个包（pack）。同一个变量可以在多个包中出现。在分析时只推导每个变量包内的变量之间的关系。

为了处理浮点程序，Astrée 把浮点表达式抽象为实数表达式。但是与本文第 1.3.3 节所介绍的技术不同的是，Astrée 以一种“绝对误差”的形式来对浮点表达式进行抽象。比如，$x \oplus y$ 将被抽象为 $x+y+[-\epsilon_{\text{abs}}, \epsilon_{\text{abs}}]$，而非 $[1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}]x + [1-\varepsilon_{\text{rel}}, 1+\varepsilon_{\text{rel}}]y + [-\varepsilon_{\text{abs}}, \varepsilon_{\text{abs}}]$。然而，区间变量系数依然可能在抽象后的程序中出现，比如对非线性表达式（如两表达式的乘/除）进行线性化时会产生区间变量系数。

(2) 实验相关全局信息

本文下面使用如下符号来表示实验中所用到数值抽象域集合：

- all：Astrée 中原有抽象域（不包括本文的 FP-itvLinEqs）；
- all+itveq：Astrée 中原有抽象域，以及 FP-itvLinEqs；
- all-rel：Astrée 中原有的非关系型抽象域（主要包括区间域和同余域）；
- all-rel+itveq：Astrée 中原有的非关系型抽象域，以及 FP-itvLinEqs；
- all-oct：Astrée 中原有抽象域除八边形域外；
- all-oct+itveq：Astrée 中原有抽象域除八边形域外，另加 FP-itvLinEqs。

本实验中，itvLinEqs 域所使用的变量分组信息重用 Astrée 中八边形域的变量分组信息。平均每个包的变量数大概是 3 到 4 个，同一个变量最多在 2 个包中出现。下面，表 5.4-5.6 将给出 itvLinEqs 域在 Astrée 上的实验结果。对于每个

测试程序，分别给出了其代码行数、全局变量个数以及分析器所使用的变量包的个数。“#alarms” 给出了分析器所报的错误（有可能误报，但是不会漏报）的个数。

(3) 实验结果

第一类嵌入式命令控制程序. Application1 源于航空领域的一个安全攸关嵌入式命令控制 C 程序。该 C 程序是从图形化的同步数据流语言（如 LUSTRE、SCADE 等）程序中自动生成得到的，其中包含了许多小规模的 C 宏。所生成的代码的最外层是一个很大规模的死循环。内部循环的规模都很小，分析时将被自动完全展开。最外层的循环在分析将被自动展开 3 次。表 5.4 给出的迭代次数 “#iter.”指最外层的循环（展开后）的带加宽的迭代次数。

表 5.4 行阶梯形区间线性等式域在 Astrée 上的实验结果 1

Program	Domains	#iter.	time	#alarms
Application1_part • 368 lines • 127 global variables • 20 packs	**all**	10	5s	0
	all+itveq	17	10s	0
	all-rel	5	1s	0
	all-rel+itveq	17	6s	0
	all-oct	10	4s	0
	all-oct+itveq	17	15s	0
Application1 • 37 000 lines • 19 013 global variables • 2 435 packs	**all**	49	2h 46m	2
	all+itveq	69	7h 20m	2
	all-rel	31	22m	1469
	all-rel+itveq	116	5h 13m	1168
	all-oct	87	3h 31m	6
	all-oct+itveq	126	12h 15m	3

其中 Application1_part 是整个应用程序 Application1 中一个片断。该片断比较简单，区间抽象域就可以实现零误报。但是，加入关系型抽象域可以得到一些更精确的区间范围：与 all-rel 相比，all 精化了 35 个区间；all+itveq 在此基础上进一步精化了另外 4 个不能被其他关系型抽象域精化的区间；但是八边形域不能精化任何区间。对于该程序，FP-itvLinEqs 所找到的不变式形如：① $X - Y = 0$；② $X - e = 0$（主要源于赋值语句，其中 e 为区间线性表达式）；③ $[0, 1]X - e = 0$（主

要源于分支语句的接合操作)。

对于完整的应用程序 Application1 而言，虽然添加 itveq 到 all 不会消除任何误报，但是可以精化 329 个区间。注意，all-oct+itveq 几乎与 all 一样精确，这意味着对于该程序 itvLinEqs 域几乎可以替代八边形域。与简单的区间分析 all-rel 相比，all-rel+itveq 可以消除 301 个误报。我们从这些误报中随机挑选了 30 个误报，发现这些误报主要源于速率限制器代码，或者使用了速率限制器输出结果的代码。注意，Application1 中包含了一些（区间）线性抽象域不能处理的计算，比如二阶滤波器或布尔计算，从而可以解释 all-rel+itveq 中的高误报数。对于 Application1，itvLinEqs 域找到了很多形如 $[0,1]X-e=0$ 或 $[-1,0]X-e=0$ 的不变式，这主要源于对分支语句的接合操作。并且，itvLinEqs 域发现了许多形如 $\Sigma_i a_i X_i+[c,d]=0$ 的线性条纹，其中 a_i 常常属于 $\{-1,1\}$，这些不变式主要源于一些简单赋值语句所构成的序列，如 X_3:=X_1+X_2； X_5:=X_3-X_4； ...

第二类嵌入式命令控制程序. 下面考虑另一个类型的安全攸关嵌入式命令控制 C 程序，实验结果如表 5.5 所示。其计算与第一类程序类似，但是代码生成方法不太一样，并且所涉及的 C 宏更加复杂。对于该类程序，最外层的循环将在分析过程中自动展开 24 次。

表 5.5 行阶梯形区间线性等式域在 Astrée 上的实验结果 2

Program	Domains	#iter.	time	#alarms
Application1_V1 • 103 000 lines • 13 392 global variables • 2 230 packs	**all**	26	31mn	2
	all+itveq	30	57mn	2
	all-rel	19	6mn	711
	all-rel+itveq	31	30mn	545
	all-oct	30	20mn	2
	all-oct+itveq	27	47mn	2
Application1_V2 • 494 000 lines • 53 664 global variables • 7 303 packs	**all**	62	7h 22mn	3
	all+itveq	61	25h 28mn	3
	all-rel	41	1h 12mn	4736
	all-rel+itveq	86	16h 4mn	2710
	all-oct	50	3h 37mn	3
	all-oct+itveq	57	21h 7mn	3

Application1_V1 是该应用程序的第一个版本，只实现了部分功能。把 itveq 添加到 all 不能消除任何误报，但是可以精化 22 个区间。与 all-rel 相比，all-rel+itveq 可以消除 166 个误报。然而，剩下的 545 个误报需要非线性不变式才能消除，从而需要其他面向领域的抽象域。对于该程序，itveq 所产生的大部分区间线性不变式，都源于数字滤波器相关代码，如二阶数字滤波器（S:=$\alpha*S_1$+$\beta*S_2$+$\gamma*E$+$\delta*E_1$+$\iota*E_2$; E_2:=E_1; E_1:=E; S_2:=S_1; S_1:=S;）。然而，这些区间线性不变式并没有增加有意义的信息，因为二阶数字滤波器需要非线性不变式才能刻画。

Application1_V2 是该应用程序的第二个版本，程序规模更大，实现的功能更多。all+itveq 不能在 all 基础上消除任何误报，但是精化了 82 个区间。与 all-rel 相比，all-rel+itveq 消除了 2026 个误报。

硬件自动测试程序. 这是一个用来在硬件切换启动时对硬件进行测试的嵌入式软件。整个程序由很多小片断组成，每个片断测试一个功能点。每个测试的大致过程是先向存储映射 I/O 写一些数据，然后读回。该程序只包含一些整数计算。该程序内部包含了很多小规模的循环，但是分析时不会被展开。本文在表 5.6 中给出的迭代次数指程序中所有循环的迭代次数之和。另外，该程序涉及很多数组变量，本实验中 Astrée 把所有数组打散为多个标量变量，把每个标量变量当作单个变量来处理，并算入全局变量个数中。但是，这些标量变量并不加入到变量包中，从而可以解释表 5.6 中全局变量个数超过 10 万个，而变量包的个数只有 62 个。

表 5.6 行阶梯形区间线性等式域在 Astrée 上的实验结果 3

Program	Domains	#iter.	time	#alarms
Auto-test • 50 000 lines • 17 032 global variables • 62 packs	**all**	1137	12s	101
	all+itveq	2261	26s	101
	all-rel	948	3s	115
	all-rel+itveq	2164	8s	109
	all-oct	1048	17s	115
	all-oct+itveq	2256	8s	109

对于 Auto-test 程序，对比行 all、all-rel、all-oct 所报的错误个数可以看出：只使用非关系型抽象域将导致出现 14 个误报。注意，消除这些误报，只需要线性抽象域（如八边形域）就可以，并不需要面向领域的高阶抽象域。与 all-rel 相比，all-rel+itveq 可以消除 6 个误报，而八边形域（对比 all 和 all-oct）则可以消除 14 个

误报（包括 itveq 所能消除的误报，另加其他 8 个误报）。这些误报与循环计数器之间关系的发现相关，区间抽象域不能表达这种关系，从而需要线性抽象域。然而，itvLinEqs 域有时足以发现这种关系，但有时不能。

5.5 小结

本章提出了一个新的数值抽象域 —— 行阶梯形区间线性等式域。该域对经典的仿射等式抽象域进行了一般化，使其支持区间系数，可用来推导程序中变量间的区间线性等式关系（形如 $\Sigma_k[a_k, b_k] \times x_k = [c, d]$）。通过采用区间线性约束的弱解语义，该域能够天然地表达某类非凸（甚至非连通、非封闭）性质。通过维持约束系统的一个行阶梯形式，该域具有多项式的时空复杂度：在域表示上的空间复杂度为 $O(n^2)$，在域操作算法上的时间复杂度为 $O(n^4)$。本文在行阶梯形区间线性等式域原型系统 FP-itvLinEqs 上的实验结果说明了：FP-itvLinEqs 可以发现程序中一些有意义的区间线性不变式，包括常用的仿射等式、线性条纹、线性不等式等。在商业化工具 Astrée 和工业界实际代码（规模达几十万行）上的实验结果说明了：FP-itvLinEqs 具有很强的可扩展性，可应用于实际大规模的程序分析中。从而，在实际中，行阶梯形区间线性等式域为多面体类的抽象域提供了一个时空高效的替代候选。

第六章　面向线性绝对值及广义线性互补关系的抽象域

6.1　引言

本章给出线性绝对值不等式抽象域和线性绝对值等式抽象域，用来推导程序变量之间的线性绝对值关系（形如 $\Sigma_k a_k x_k + \Sigma_k b_k |x_k| \leq c, \Sigma_k a_k x_k + \Sigma_k b_k |x_k| = c$）。

绝对值关系. 在数学中，绝对值是一个很基本的概念，用来表示距离或数量的大小。几何意义下，数的绝对值就是数轴上表示这个数的点到原点的距离。比如，$|a|$ 表示数轴上数 a 所对应的点到原点的距离，$|x-a|$ 表示数轴上数 x 所对应的点到数 a 所对应的点之间的距离。代数意义下，正数和 0 的绝对值是它本身，负数的绝对值是它的相反数，即

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

物理意义下，绝对值可以用来刻画分段物理特征，比如绝对值函数是一种常见的分段函数 [1]，如图 6.1 所示。

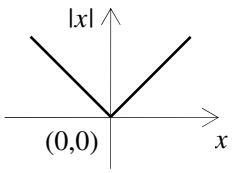


图 6.1　绝对值函数示意图

实际中，绝对值也有着广泛的应用。例如，许多实际程序语言提供了有关绝对值的标准库函数。比如，C 语言（在 C99 中）提供了 $abs(), labs(), llabs()$ 等用来计算整数对象绝对值的函数，以及 $fabs(), fabsf(), fabsl()$ 等用来计算浮点对象绝对值的函数。另外，C 语言（在 C99 中）还提供了 $fmin()$（求最小值）、$fmax()$（求最大值）、$fdim()$（求正差值）等函数，这些函数都是分段函数，并且实际上都可以通过绝对值函数来表达（比如 $\max(x,y) = \frac{1}{2}(|x-y|+x+y)$）。在第 2.5 节，我们已看到浮点程序中浮点舍入误差也可以通过绝对值函数来抽象：$|R_{\mathbf{f},r}(x) - x| \leq \varepsilon_{\text{rel}} \cdot |x| + \varepsilon_{\text{abs}}$。但是，绝对值函数 $|x|$ 本质上是非线性函数。更准确地说，绝对值函数是一种分段线性函数（即分段函数的每一段都是线性的）。正是因为这种非线性特征，使得目前程序分析与验证很少考虑绝对值相关程序行为和相关库函数。

[1] 分段函数是指自变量在不同的取值范围内，其对应法则也不相同的函数。

另一方面，从图 6.1 不难看出，绝对值函数也可以描述某种非凸性质。因此，如果在抽象域的域表示中支持绝对值函数，就可以构造出非凸数值抽象域。这是本章的主要研究动机之一。

广义线性互补问题. 线性互补问题是数学优化领域的一个重要优化问题，在工程设计、最优控制、计算力学、交通与经济平衡等领域有着广泛应用。线性互补问题涵盖了很多数学优化问题，比如线性规划、二次规划、双矩阵博弈等问题 [170]。正是因为其重要性，该问题在优化领域得到了广泛关注。并且，根据应用需求，许多研究从不同角度对线性互补问题进行了扩展和一般化，衍生出了多种广义线性互补问题[170, 171]。本文关注的正是其中的一种。本章将利用广义线性互补关系来构造抽象域，以用来验证实际中通过广义线性互补问题所描述的一些系统和模型 [172, 173, 174]，比如一些混成系统的模型。

区间线性关系. 本文第 4 章介绍了区间多面体抽象域，用来推导程序中变量间的区间线性关系。但是，在第 4 章，区间多面体域中的许多域操作都不是最佳抽象。比如，区间多面体域中所给出的接合操作只是一个弱的接合操作。进一步还需考虑如下问题：给定两个区间多面体，包含了这两个区间多面体的最小区间多面体是否存在？若存在，如何设计算法构造出这样的最小区间多面体？本章将回答这些问题。

首先，本章将给出并证明线性绝对值不等式系统、区间线性不等式系统、广义线性互补问题（系统）三者之间的等价性，以及线性绝对值等式系统、水平线性互补问题（系统）两者之间的等价性。然后，本章将提出两种新的数值抽象域，即线性绝对值不等式抽象域和线性绝对值等式抽象域。其中，线性绝对值不等式抽象域是经典凸多面体抽象域的一般化，用来推导变量 $x_k(k=1,\ldots,n)$ 间的线性绝对值不等式关系（$\Sigma_k a_k x_k + \Sigma_k b_k|x_k| \leq c$ ）、区间线性不等式关系（$\Sigma_k [a_k, b_k] \times x_k \leq c$ ）、广义线性互补不等式关系（$\Sigma_k a_k x_k^+ + \Sigma_k b_k x_k^- \leq c$，其中 $x^+, x^- \geq 0 \wedge (x^+)^T x^- = 0 \wedge x = x^+ - x^-$）。而线性绝对值等式抽象域是经典仿射等式抽象域的一般化，用来推导变量 $x_k(k=1,\ldots,n)$ 间的线性绝对值等式关系（$\Sigma_k a_k x_k + \Sigma_k b_k|x_k| = c$ ）、水平线性互补等式关系（$\Sigma_k a_k x_k^+ + \Sigma_k b_k x_k^- = c$，其中 $x^+, x^- \geq 0 \wedge (x^+)^T x^- = 0 \wedge x = x^+ - x^-$）。并且，这两个新的抽象域都是非凸的且域操作都是具体域上对应操作的最佳抽象。

本章的结构组织如下：6.2 节介绍线性绝对值系统及其等价刻画；6.3 节介绍广义线性互补问题（系统）的双重描述法；6.4 节介绍线性绝对值不等式抽象域的设计；6.5 节介绍线性绝对值等式抽象域的设计；6.6 节介绍这两个抽象域的原型系统实现，给出并分析实验结果；6.7 节是对本章的小结。

6.2 线性绝对值系统及其等价刻画

(1) 线性绝对值系统

给定 $A, B \in \mathbb{R}^{m\times n}$ 和 $c \in \mathbb{R}^m$，我们定义如下线性绝对值等式（简称 AVE）系统：

$$Ax + B|x| = c \tag{6.1}$$

该形式是由 Rohn 在文 [175] 中提出的，最近引起了许多研究的关注 [176, 177, 178, 179]。注意，求解 AVE (6.1) 是 NP-完全问题 [179]。

给定 $A, B \in \mathbb{R}^{m\times n}$ 和 $c \in \mathbb{R}^m$，我们定义如下线性绝对值不等式（简称 AVI）系统

$$Ax + B|x| \leq c \tag{6.2}$$

该形式最初在文 [177] 中出现。对于上述 AVI 系统，本文下面给出一个等价的受限形式并证明其等价性。

引理 6.1 *任意 AVI 系统 $Ax + B|x| \leq c$ 可以等价地转换为如下 AVI 系统 $A'x + B'|x| \leq c'$ 其中 $B' \leq 0$。*

证明. 任意 AVI 不等式

$$\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p|x_p| \leq c \tag{6.3}$$

其中 $b_p > 0$，等价于如下两个 AVI 不等式的合取

$$\begin{cases} \sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p x_p \leq c \\ \sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| - b_p x_p \leq c \end{cases} \tag{6.4}$$

这可以通过如下两种情况来证明：

- 当 $x_p \geq 0$ 时：因为

$$\begin{aligned} &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p|x_p| \leq c \\ \Leftrightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p x_p \leq c \\ \Leftrightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| - b_p x_p \leq c - 2b_p x_p \\ \Rightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| - b_p x_p \leq c \end{aligned}$$

从而，此时 $\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| - b_p x_p \leq c$ 在系统 (6.4) 中是冗余的，因此有不等式 (6.3) 与系统 (6.4) 是等价的。

- 当 $x_p \leq 0$ 时，因为

$$\begin{aligned}
&\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p|x_p| \leq c \\
\Leftrightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| - b_p x_p \leq c \\
\Leftrightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p x_p \leq c + 2b_p x_p \\
\Rightarrow\ &\textstyle\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p x_p \leq c
\end{aligned}$$

从而，此时 $\sum_i a_i x_i + \sum_{i\neq p} b_i|x_i| + b_p x_p \leq c$ 在系统 (6.4) 中是冗余的，因此有不等式 (6.3) 与系统 (6.4) 是等价的。

因此，原 AVI 系统 $Ax + B|x| \leq c$ 中的任意 AVI 不等式

$$\sum_i a_i x_i + \sum_i b_i|x_i| \leq c$$

（通过必要的展开）都可以等价地转换成一个 AVI 系统

$$A''x + B''|x| \leq c''$$

其中 $B'' \leq 0$。那么，整体上，原 AVI 系统 $Ax + B|x| \leq c$ 也可以等价地转换成如下形式的 AVI 系统

$$A'x + B'|x| \leq c'$$

其中 $B' \leq 0$。 □

接下来，我们给出 AVI 系统与 AVE 系统之间的关系。首先，AVE (6.1) 等价于如下 AVI 系统（大小为 $\mathbb{R}^{2m\times(2n+1)}$）

$$\begin{pmatrix} A \\ -A \end{pmatrix} x + \begin{pmatrix} B \\ -B \end{pmatrix} |x| \leq \begin{pmatrix} c \\ -c \end{pmatrix}$$

另一方面，通过引入 m 个松弛变量所构成的向量 $y \in \mathbb{R}^m$ 并基于如下观察：

$$y \geq 0 \Leftrightarrow |y| = y$$

可知 AVI 系统 (6.2) 等价于如下 AVE 系统（大小为 $\mathbb{R}^{2m\times(2n+2m+1)}$）

$$\begin{pmatrix} A \\ 0 \end{pmatrix} x + \begin{pmatrix} B \\ 0 \end{pmatrix} |x| + \begin{pmatrix} I \\ I \end{pmatrix} y + \begin{pmatrix} 0 \\ -I \end{pmatrix} |y| = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

其中 I 为 $\mathbb{R}^{m\times m}$ 上的单位矩阵。

(2) 线性互补问题及其扩展

给定矩阵 $M \in \mathbb{R}^{n\times n}$ 和向量 $q \in \mathbb{R}^n$，如下问题称为（标准）线性互补问题（Linear Complementarity Problem，简称 LCP）：寻找向量 x^+ 和 x^- 使得

$$x^+ = Mx^- + q \tag{6.5}$$

$$x^+, x^- \geq 0 \tag{6.6}$$

$$(x^+)^T x^- = 0. \tag{6.7}$$

我们记上述 LCP 为 (q, M)。从 (6.6) 和 (6.7) 可知，如果 x^+ 和 x^- 是 LCP (q, M) 的解，则有

$$x_i^+ x_i^- = 0 \qquad \text{for } i = 1, \ldots, n$$

即对于每个 i 皆有：若 $x_i^+ > 0$ 则 $x_i^- = 0$，若 $x_i^- > 0$ 则 $x_i^+ = 0$。从而，x_i^+ 与 x_i^- 为 0 的模式是互补。因此，条件 (6.7) 通常称为 LCP (q, M) 的互补条件。

下面，我们给出本文所关心的 LCP 的几种扩展：

- 水平线性互补问题（Horizontal LCP，简称 HLCP）[170]:
  给定 $M, N \in \mathbb{R}^{m\times n}$ 和 $q \in \mathbb{R}^n$，寻找 $x^+, x^- \in \mathbb{R}^n$ 使得

$$Mx^+ + Nx^- = q \tag{6.8}$$

$$x^+, x^- \geq 0 \tag{6.9}$$

$$(x^+)^T x^- = 0. \tag{6.10}$$

  HLCP 最初是在电机工程领域研究分段线性电路时引入的，但是后来发现 HLCP 与 LCP 其实是等价的 [180]，且与分段线性等式系统是等价的 [181]。

- 由 Mangasarian 与 Pang 所提出的广义线性互补问题（eXtended LCP，简称 XLCP）[182, 183]:
  给定 $M, N \in \mathbb{R}^{m\times n}$ 和一个多面体 $\mathcal{P} \subseteq \mathbb{R}^m$，寻找 $x^+, x^- \in \mathbb{R}^n$ 使得

$$Mx^+ + Nx^- \in \mathcal{P}$$

$$x^+, x^- \geq 0$$

$$(x^+)^T x^- = 0.$$

  XLCP 是 HLCP 的一般化（当 $\mathcal{P}$ 通过一个向量 $q \in \mathbb{R}^m$ 来定义时，XLCP 即为 HLCP）。不失一般性，$\mathcal{P}$ 可以通过如下集合来表示 $\mathcal{P} = \{y \in \mathbb{R}^m : Ay \leq b\}$，

其中 $A \in \mathbb{R}^{k\times m}$ 为某矩阵而 $b \in \mathbb{R}^k$ 为某向量。因此，条件 $Mx^+ + Nx^- \in \mathcal{P}$ 等价于 $AMx^+ + ANx^- \leq b$，从而可以进一步写成 $M'x^+ + N'x^- \leq b$ 形式（其中 $M' = AM, N' = AN$）。根据这一观察，我们下面引入另一种表述形式。

- 广义线性互补问题（称为 Inequality HLCP，简称 IHLCP）：
  给定 $M, N \in \mathbb{R}^{m\times n}$ 和向量 $q \in \mathbb{R}^m$，寻找 $x^+, x^- \in \mathbb{R}^n$ 使得

$$Mx^+ + Nx^- \leq q \tag{6.11}$$

$$x^+, x^- \geq 0 \tag{6.12}$$

$$(x^+)^T x^- = 0. \tag{6.13}$$

  直观上，IHLCP 可以看成是 HLCP 的不等式版本，但是 IHLCP 比 HLCP 更具一般性。不难看出，IHLCP 与 XLCP 是等价的：前面我们已经看到 XLCP 可以重写成 IHLCP 形式；另一方面，IHLCP 也可以归结成 XLCP，其中 $\mathcal{P} = \{y \in \mathbb{R}^m : y \leq q\}$。

下面，本文只关心 HLCP 和 IHLCP。为了方便描述，本文有时称水平线性互补问题 HLCP 所对应的约束系统为*水平线性互补系统*，称广义线性互补问题 IHLCP 所对应的约束系统为*广义线性互补系统*。

(3) 区间线性不等式系统

下面，简单回顾第 4.2 节所介绍的区间线性系统的相关定义和结果。设 $\mathbf{A} \in \mathbb{IR}^{m\times n}$ 为一个 $m \times n$ 区间矩阵，其中点矩阵为 $A_c = \frac{1}{2}(\underline{A} + \overline{A})$，其半径矩阵为 $\triangle_A = \frac{1}{2}(\overline{A} - \underline{A})$。并设 $b \in \mathbb{R}^m$ 为一个 m 实数向量。那么，对于区间线性不等式系统

$$\mathbf{A}x \leq b$$

我们有如下定理（参考文 [129]）：

定理 6.2 *向量 $x \in \mathbb{R}^n$ 是 $\mathbf{A}x \leq b$ 的弱解当且仅当 x 满足 $A_c x - \Delta_A|x| \leq b$。*

注意，根据 Δ_A 的定义，总有 $\Delta_A \geq 0$。换而言之，在上述定理中，$|x|$ 的系数总是非正的。

6.2.1 线性绝对值不等式系统及其等价刻画

(1) 线性绝对值不等式系统与广义线性互补系统 IHLCP 之间的等价性

给定向量 $x = (x_i)_{i=1}^n$，我们定义向量 x^+ 和 x^- 为

$$x^+ \stackrel{\text{def}}{=} (\max(x_i, 0))_{i=1}^n, \quad x^- \stackrel{\text{def}}{=} (\max(-x_i, 0))_{i=1}^n$$

从而有

$$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0 \tag{6.14}$$

且

$$x = x^+ - x^-, \qquad |x| = x^+ + x^- \tag{6.15}$$

$$x^+ = \frac{1}{2}(x + |x|), \qquad x^- = \frac{1}{2}(|x| - x) \tag{6.16}$$

根据公式 (6.15)，AVI 系统 (6.2) 可以等价地重写为如下 IHLCP 系统：

$$\begin{aligned}(A+B)x^+ + (B-A)x^- &\leq c\\ x^+, x^- &\geq 0\\ (x^+)^T x^- &= 0\end{aligned}$$

类似地，根据公式 (6.16)，IHLCP 系统 (6.11)-(6.13) 可以等价地重写为如下 AVI 系统：

$$\frac{1}{2}(M-N)x + \frac{1}{2}(M+N)|x| \leq q$$

(2) 线性绝对值不等式系统与区间线性不等式系统之间的等价性

定理 6.3 任意 AVI 系统

$$Ax + B|x| \leq b \tag{6.2}$$

其中 $A, B \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ 可以等价地重写为一个区间线性不等式系统

$$\mathbf{A}'x \leq b' \tag{6.17}$$

其中 $\mathbf{A}' \in \mathbb{IR}^{k\times n}, b' \in \mathbb{R}^k (k \in \mathbb{N})$。反之亦然。

证明. i) (6.17) → (6.2)：根据定理 6.2，系统 (6.17) 等价于如下 AVI 系统

$$A'_c x - \triangle_{A'}|x| \leq b'$$

ii) (6.2) → (6.17)：

- 若 $B \leq 0$，那么根据定理 6.2，系统 (6.2) 可以等价地重写为

$$[A+B, A-B]x \leq b$$

- 否则，根据引理 6.1 可知，系统 (6.2) 可以等价地重写为另一个 AVI 系统

$$A''x + B''|x| \leq b''$$

其中 $B'' \leq 0$。然后，根据定理 6.2，上述系统可以进一步重写为如下区间线性不等式系统：

$$[A'' + B'', A'' - B'']x \leq b''$$

□

6.2.2 线性绝对值等式系统及其等价刻画

根据公式 (6.15)，AVE 系统 (6.1) 可以重写为如下 HLCP 系统：

$$\begin{aligned}(A+B)x^+ + (B-A)x^- &= c\\ x^+, x^- &\geq 0\\ (x^+)^T x^- &= 0\end{aligned}$$

类似地，根据公式 (6.16)，HLCP 系统 (6.11)-(6.11) 可以重写为如下 AVE 系统：

$$\frac{1}{2}(M-N)x + \frac{1}{2}(M+N)|x| = q$$

注意，在相同变量集合上，线性绝对值等式系统与如下两种形式的区间线性等式系统都不具有等价关系：

$$\mathbf{A}'x = \mathbf{b}' \quad 或 \quad \mathbf{A}''x = b''$$

其中，$\mathbf{b}'$ 表示一个区间向量，b'' 表示一个实数向量。

6.3 广义线性互补问题的双重描述法

在最优化、计算几何、程序分析等领域，多面体经常都会采用 Motzkin 等人于 1953 年提出 [184] 的双重描述法来表示。

根据 Minkowski-Weyl 表示定理（见文献 [185, 186]），一个集合 $P \subseteq \mathbb{R}^n$ 是凸多面体当且仅当它是有限生成的，即存在有穷集合 $V, R \in \mathbb{R}^n$ 使得集合 P 可以通过 (V, R) 生成：

$$P = \left\{ \sum_{i=1}^{|V|} \lambda_i V_i + \sum_{j=1}^{|R|} \mu_j R_j \mid \forall i, \lambda_i \geq 0, \forall j, \mu_j \geq 0, \sum_{i=1}^{|V|} \lambda_i = 1 \right\}$$

其中，V 中的元素称为顶点（或称 extreme point），R 中的元素称为极射线（extreme ray），(V,R) 统称为多面体的生成子。我们使用 $|V|$ 和 $|R|$ 分别表示集合 V 和 R 中的元素个数。直观上，凸多面体中的任意一点都可以通过顶点的凸组合和极射线的非负组合（也称锥组合）来表示。

在多面体的多重描述法中，一个多面体可以通过线性不等式约束系统 $\{Ax \leq b\}$ 来表示，也可以通过生成子集合 (V,R) 来表示。实际上，约束系统刻画了多面体的代数形式，生成子则刻画了多面体的关键几何特征。多面体的约束表示和生成子表示是对偶的：每种表示方法都可以等价地转化成另外一种表示方法。这种对偶转化一般采用 Chernikova 算法 [187, 188, 189] 来实现。目前，多面体抽象域的公开实现库都是基于多面体的双重描述法并采用 Chernikova 算法开发出来 [160]，包括 Polylib[157]、NewPolka [158]、Parma Polyhedra Library（PPL）库 [159] 等。

本节将基于凸多面体的双重描述法来构造广义线性互补系统 IHLCP 的双重描述法，并基于 Chernikova 算法来构造广义线性互补系统 IHLCP 的约束表示和生成子表示之间的对偶转换。

6.3.1 从约束表示到生成子表示的转化

(1) 计算多面体尖锥的所有非负解

首先，我们回顾经典的 Chernikova 算法，并采用 Chernikova 算法来计算多面体尖锥（pointed polyhedral cone，形如 $\{x \mid Ax \geq 0, x \geq 0\}$）的生成子表示。注意，对于多面体尖锥而言，除顶点 $\{\mathbf{0}\}$ 外生成子皆为极射线。

设 $C = \{x \in \mathbb{R}^n \mid Ax \geq 0, x \geq 0\}$ 为一个多面体尖锥，$H = \{x \in \mathbb{R}^n \mid cx \geq 0\}$ 为 $\mathbb{R}^n$ 上的一个半空间。并设 $Q = \{y_1, \ldots, y_r\}$ 为 C 的极射线的非冗余集合。那么，根据 cy_i 的计算结果，我们可以将 Q 划分成如下三个集合：

$$Q^{=} = \{y \mid y \in Q, cy = 0\}$$
$$Q^{>} = \{y \mid y \in Q, cy > 0\}$$
$$Q^{<} = \{y \mid y \in Q, cy < 0\}$$

那么，新锥 $C \cap H$ 的极射线的非冗余集合 Q' 为

$$Q' = Q^{=} \cup Q^{>} \cup \overline{Q}$$

其中 $\overline{Q}$ 定义为

$$\{y \mid cy = 0, y = \lambda y_1 + \mu y_2, (y_1, y_2) \in Q^{>} \times Q^{<}, adjacent(y_1, y_2), \lambda > 0\}$$

其中 $adjacent(y_1, y_2) = true$ 表示 y_1 与 y_2 在集合 Q 中是相邻的。

集合 $Q^=$ 与 $Q^>$ 包括了 Q 中那些满足 H 从而也属于 $C \cap H$ 的极射线。集合 $\overline{Q}$ 通过对分别来自 $Q^>$ 与 $Q^<$ 的极射线进行正组合来构造位于超平面 $cx = 0$ 上的新的极射线。注意，满足 $cy = 0 \wedge y = \lambda y_1 + \mu y_2 \wedge \lambda > 0$ 条件的 y 总可以定义出来，比如令 $\lambda = -cy_2$ 且 $\mu = cy_1$。而相邻条件 $adjacent(y_1, y_2)$ 则保证了集合 $\overline{Q}$ 中新产生的射线也都是极射线，即不能通过对 Q' 中其他极射线进行正组合来得到，从而不是冗余的。不难看出，集合 $\overline{Q}$ 的构造方式是影响 Chernikova 算法计算效率的关键因素，而其中关于 $adjacent(y_1, y_2)$ 的判断又最为重要。

设 $C' = \{x \mid Ax \geq 0\}$ 为一个多面体锥，对于某个 $y \in C'$（几何上对应 C' 内的某一点），我们使用集合 $S(y, A)$ 来表示如下集合

$$S(y, A) \stackrel{\text{def}}{=} \{a \mid a \text{ 是 } A \text{ 中某一行且使得 } ay = 0 \text{ 成立 }\}$$

实现时，$S(y, A)$ 可以通过这些行在 A 中的下标集合 $\mathcal{I}(y, A)$ 来表示。

引理 6.4 ([160]) 给定 $y_1, y_2 \in \{x \mid Ax \geq 0\}$，设 y 是 y_1 与 y_2 的正组合。那么有 $S(y, A) = S(y_1, A) \cap S(y_2, A)$。

设 D 为 $C = \{x \mid Ax \geq 0, x \geq 0\}$ 的约束矩阵，即

$$D = \begin{bmatrix} A \\ I \end{bmatrix}$$

其中 $I \in \mathbb{R}^{n \times n}$ 为单位矩阵。那么，两个极射线 y_1 与 y_2 称为在集合 Q 中是相邻的，即 $adjacent(y_1, y_2) = true$，当且仅当 Q 中不存在其他极射线 y' 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$。

就整个过程而言，为了计算 $\{x \mid Ax \geq 0, x \geq 0\}$ 的生成子表示，我们设置初始集合 $C_0 = \{x \mid x \geq 0\}$，$Q_0 = \{e_1, \ldots, e_n\}$ 其中 e_i 为单位基向量。然后，逐行考虑 $Ax \geq 0$ 中每一行所对应的约束 $H = \{x \in \mathbb{R}^n \mid cx \geq 0\}$，从而可以基于约束表示 C 所对应的生成子表示 Q 构造出 $C \cap H$ 所对应的生成子表示 Q'。此外，实现时还有很多实用性的启发式规则来避免计算 $\overline{Q}$ 过程中不必要的枚举（参见文 [160]）。

(2) 计算满足互补条件的多面体尖锥的所有非负解

互补生成子集合.

为了方便描述，这里我们记 $x = (x^+, x^-)^T$，其中向量 $x \in \mathbb{R}^{2n}, x^+, x^- \in \mathbb{R}^n$。下面，我们考虑如下形式

$$\{x \mid Ax \geq 0, x \geq 0, (x^+)^T x^- = 0\}$$

注意，其中 $\{x \mid Ax \geq 0, x \geq 0\}$ 部分表示的是一个多面体尖锥。因此，我们可以采用标准的 Chernikova 算法先计算出多面体尖锥 $\{x \mid Ax \geq 0, x \geq 0\}$ 的所有生成子，然后从该生成子集合中删除那些不满足互补条件 $(x^+)^T x^- = 0\}$ 的生成子，就可以得到 $\{x \mid Ax \geq 0, x \geq 0, (x^+)^T x^- = 0\}$ 的生成子表示。但是，在计算出 $\{x \mid Ax \geq 0, x \geq 0\}$ 的所有生成子后再删除那些不满足互补条件的生成子，会导致许多不必要的计算，甚至产生组合爆炸问题。

因此，本文采用一种增量式的方式来计算满足互补条件的多面体尖锥的生成子表示。换而言之，本文将在计算过程中把互补条件考虑进来，只产生那些满足互补条件的生成子，本文称之为互补生成子。

设 $C_\pm = \{x \mid Ax \geq 0, x \geq 0, x^+x^- = 0\}$ 为一个满足互补条件的多面体尖锥，$H = \{x \mid cx \geq 0\}$ 为 $\mathbb{R}^{2n}$ 上的一个半空间。并设 $Q_\pm = \{y_1, \ldots, y_r\}$ 为 $C_\pm$ 的互补极射线的非冗余集合。根据 cy_i 的计算结果，我们可以将 $Q_\pm$ 划分成如下三个集合：

$$Q_\pm^= = \{y \mid y \in Q_\pm, cy = 0\}$$
$$Q_\pm^> = \{y \mid y \in Q_\pm, cy > 0\}$$
$$Q_\pm^< = \{y \mid y \in Q_\pm, cy < 0\}$$

那么，新锥 $C_\pm \cap H$ 的互补极射线的非冗余集合 $Q'_\pm$ 即为

$$Q'_\pm = Q_\pm^= \cup Q_\pm^> \cup \overline{Q}_\pm$$

其中 $\overline{Q}_\pm$ 定义为

$$\{y \mid cy = 0, y = \lambda y_1 + \mu y_2, (y_1, y_2) \in Q_\pm^> \times Q_\pm^<, adjacent^c(y_1, y_2), \lambda > 0, y^+y^- = 0\}$$

其中互补相邻条件 $adjacent^c(y_1, y_2) = true$ 表示 y_1 与 y_2 在集合 $Q_\pm$ 中是相邻的，即 $Q_\pm$ 中不存在其他互补极射线 y' 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$ 成立。

前面我们看到，对于多面体尖锥 C 而言，Q 中不存在其他极射线 y' 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$ 成立，是相邻条件 $adjacent(y_1, y_2)$ 成立的充分必要条件。但是，对于满足互补关系的多面体尖锥 $C_\pm$ 而言，判断互补相邻条件 $adjacent^c(y_1, y_2)$ 是否为真时，我们实际上只检查了 Q 的一个子集 $Q_\pm$ 中是否存在极射线 $y' \in Q_\pm$ 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$ 成立。而 Q 中的某些元素可能不满足互补条件，从而不会出现在 $Q_\pm$ 中。因此，可能会出现 $adjacent^c(y_1, y_2) = true$

但 $adjacent(y_1, y_2) = false$ 的情形。换而言之，在构造 $\overline{Q}_{\pm}$ 时，我们多考虑了一些不必要的组合，即 $adjacent^c(y_1, y_2) = true$ 但 $adjacent(y_1, y_2) = false$ 的情形。

为此，我们添加一些检查 $adjacent(y_1, y_2)$ 的规则，并通过如下方式来计算 $\overline{Q}_{\pm}$:

① 若 $|S(y_1, A) \cap S(y_2, A)| \leq n - 3$，则必有 $adjacent(y_1, y_2) = false$ [160]，此时无需构造 y;

② 否则，若 $adjacent^c(y_1, y_2) = false$，则必有 $adjacent(y_1, y_2) = false$，此时无需构造 y;

③ 否则，若 $adjacent^c(y_1, y_2) = true$，则检查由 $cy = 0 \wedge y = \lambda y_1 + \mu y_2 \wedge \lambda > 0$ 所构造出来的 y 是否满足 $(y^+)^T y^- = 0$。只有当 $(y^+)^T y^- = 0$ 时，才将 y 添加到 $\overline{Q}_{\pm}$ 中。

定理 6.5 $\overline{Q}_{\pm}$ 生成了且仅生成了所有满足互补条件的极射线，即 $\overline{Q}_{\pm} = \overline{Q} \cap \{y \mid y^+ y^- = 0\}$。

证明. 我们回顾 $\overline{Q}$ 的定义

$$\overline{Q} = \{y \mid cy = 0, y = \lambda y_1 + \mu y_2, (y_1, y_2) \in Q^> \times Q^<, adjacent(y_1, y_2), \lambda > 0\}$$

首先，因为 $c(\lambda y_1 + \mu y_2) = 0$ 且 $\lambda > 0$，从而有 $\mu > 0$。对于任意 $(y_1, y_2) \in Q^> \times Q^< \wedge (y_1, y_2) \notin Q^{\geq}_{\pm} \times Q^{\leq}_{\pm}$，$y_1$ 与 y_2 的正组合不会生成任何满足互补条件 $(y^+)^T y^- = 0$ 的 y。接下来，我们定义

$$\overline{Q'} = \{y \mid cy = 0, y = \lambda y_1 + \mu y_2, (y_1, y_2) \in Q^{\geq}_{\pm} \times Q^{\leq}_{\pm}, adjacent(y_1, y_2), \lambda > 0\}$$

则有 $\overline{Q} \cap \{y \mid y^+ y^- = 0\} = \overline{Q'} \cap \{y \mid y^+ y^- = 0\}$。同时，因为 $adjacent(y_1, y_2) = true$ 蕴含 $adjacent^c(y_1, y_2) = true$，从而有 $\overline{Q'} \cap \{y \mid (y^+)^T y^- = 0\} \subseteq \overline{Q}_{\pm}$。因此，有 $\overline{Q} \cap \{y \mid y^+ y^- = 0\} \subseteq \overline{Q}_{\pm}$。

下面，设 $(y_1, y_2) \in Q^{\geq}_{\pm} \times Q^{\leq}_{\pm}$，$adjacent(y_1, y_2) = false$，$adjacent^c(y_1, y_2) = true$。那么，$\exists y' \in Q \wedge y' \notin Q_{\pm}$ 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$。接下来，我们将证明 y_1 与 y_2 的任意正组合都不满足互补条件，即 $\{y \mid cy = 0, y = \lambda y_1 + \mu y_2, \lambda > 0\} \cap \{y \mid (y+)^T y^- = 0\} = \emptyset$。假设 y_1 与 y_2 的某个正组合结果 z 满足互补条件，即 $\exists \lambda, \mu. cz = 0 \wedge z = \lambda y_1 + \mu y_2 \wedge \lambda > 0 \wedge \mu > 0 \wedge (z+)^T z^- = 0$。那么，根据引理 6.4，有 $S(z, D) = S(y_1, D) \cap S(y_2, D)$。因为 $adjacent(y_1, y_2) = true \wedge adjacent^c(y_1, y_2) = false$，从而存在 $y' \in Q \wedge y' \notin Q_{\pm}$ 使得 $S(y_1, D) \cap S(y_2, D) \subseteq S(y', D)$。从而有 $S(z, D) \subseteq S(y', D)$。因为 $(z^+)^T z^- = 0$，则有 $(y'^+)^T y'^- = 0$，这与 $y' \notin Q_{\pm}$ 矛盾。因此，前面关于 z 存在的假设不成立，由此可知 $\overline{Q}_{\pm} \subseteq \overline{Q} \cap \{y \mid y^+ y^- = 0\}$。

总而言之，$\overline{Q}_{\pm} = \overline{Q} \cap \{y \mid y^+y^- = 0\}$。 □

互补生成子分组.

前面，我们给出了算法可以计算出满足互补条件的多面体尖锥 $C_{\pm} = \{x \mid Ax \geq 0, x \geq 0, (x^+)^T x^- = 0\}$ 的所有互补生成子，即互补极射线所构成的集合 R^c。显然，R^c 内互补极射线之间的任意非负组合的结果未必满足互补条件 $(x^+)^T x^- = 0$，从而未必是 $C_{\pm}$ 中的元素。

为了精确地刻画满足互补条件的多面体尖锥的所有非负解，我们还需要根据互补条件，对 R^c 进行分组，并使得分组结果 $R^{cc} = \langle R^c_{s_1}, \ldots, R^c_{s_i}, \ldots, R^c_{s_m} \rangle$ 满足如下条件：

1. $R^c_{s_i} \subseteq R^c$，$\cup_{i=1}^m R^c_{s_i} = R^c$，且
2. 在每个组 $R^c_{s_i}$ 内，$R^c_{s_i}$ 中互补极射线之间的任意非负组合的结果都满足互补条件 $(x^+)^T x^- = 0$。

为了构造满足上述条件的分组，我们采用如下基于无向图的方法。首先，构造一个无向图 Θ，其中每个互补生成子 $r^c_i \in R^c$ 对应图 Θ 中的一个节点，并且两个节点 r^c_i, r^c_j 之间存在一条边，如果 $bin(r^c_i) \vee bin(r^c_j)$ 的结果向量 s 满足互补条件 $(s^+)^T s^- = 0$，其中函数 bin 定义如下：

给定向量 $x \in \mathbb{R}^n$，我们定义 $bin(x) = b \in \mathbb{R}^n$ 其中 $b_i = \begin{cases} 0 & \text{if } x_i = 0 \\ 1 & \text{if } x_i > 0 \end{cases}$

接下来，我们的目标就是寻找图 Θ 中的所有极大完全子图，每个极大完全子图实际上对应了互补生成子 R^{cc} 中的一个组 $R^c_{s_i}$。我们知道，生成无向图的所有极大完全子图是图论里的一个基本问题，也是最早发现的 NP-完全问题之一 [190]。所幸的是，后面我们将看到对于构造抽象域而言，并不需要对互补生成子进行分组。

定理 6.6 *设满足互补条件的多面体尖锥 $C_{\pm} = \{x \mid Ax \geq 0, x \geq 0, (x^+)^T x^- = 0\}$ 的互补生成子(这里即为互补极射线)集合的互补分组为 $R^{cc} = \langle R^c_{s_1}, \ldots, R^c_{s_i}, \ldots, R^c_{s_m} \rangle$。那么，$x \in C_{\pm}$ 当且仅当存在某个 i $(i \in \mathbb{N}, 1 \leq i \leq m)$，使得*

$$x = \sum_{r^c_k \in R^c_{s_i}} \mu_k r^c_k$$

其中 $\mu_k \geq 0$。

(3) 计算（满足互补条件的）尖多面体的所有非负解

尖多面体的生成子集合. 我们称一个多面体是尖多面体（pointed polyhedron），如果该多面体的线性空间是 $\{0\}$，即不含任何线（line）生成子。显然，$\{x \in \mathbb{R}^{2n} \mid Ax \geq$

$b, x \geq 0\}$ 是一个尖多面体。通过引入一个新的变量 $h \in \mathbb{R}$，非齐次线性不等式系统

$$\begin{aligned} Ax &\geq b \\ x &\geq 0 \end{aligned} \tag{6.18}$$

可以等价地转化为如下齐次线性不等式系统：

$$\begin{aligned} [A - b]y &\geq 0 \\ y &\geq 0 \end{aligned} \tag{6.19}$$

其中 $y = \begin{pmatrix} x \\ h \end{pmatrix}$ 是一个大小为 $(2n+1)$ 的列向量，$h \in \mathbb{R}$ 且 $h \geq 0$。

齐次线性不等式系统 (6.19) 的极射线生成子 r 都形如 $r = (x \ h)^T$ 其中 $h \geq 0$。我们使用 r_h 来表示向量 r 中的 h 元素。对于每个极射线 r，只有两种可能性：$r_h = 0$ 或 $r_h > 0$。如果 $r_h > 0$，我们把 r 向量中的每个元素都除以 r_h。不难看出，新的向量（其中 $r_h = 1$）仍然是齐次线性不等式系统 (6.19) 的极射线生成子。经过上述处理后，齐次线性不等式系统 (6.19) 的极射线生成子集合 R^h 可以分为两组：$R^0 = \{r \in R^h \mid r_h = 0\}$ 和 $R^1 = \{r \in R^h \mid r_h = 1\}$。

我们从 R^0 和 R^1 的向量中提取 x 部分，设分别得到集合 $R = \{x \mid (x \ 0)^T \in R^0\}$ 和 $V = \{x \mid (x \ 1)^T \in R^1\}$。那么，$V$ 中的生成子称为非齐次线性不等式系统 (6.18) 的顶点，而 R 中的生成子则称为非齐次线性不等式系统 (6.18) 的极射线。从而，我们得到了非齐次线性不等式系统 (6.18) 的生成子表示，即 $G = \langle V, R \rangle$。

满足互补条件的尖多面体的生成子集合. 类似地，从满足互补关系的多面体尖锥 $\{y \mid Ay \geq 0, y \geq 0, (x^+)^T x^- = 0, y = (x^+ \ x^- \ h)^T\}$ 的互补生成子集合，可以导出满足互补关系的尖多面体

$$\{x \mid Ax \geq b, x \geq 0, (x^+)^T x^- = 0\}$$

的互补生成子集合 $G^c = (V^c, R^c)$。

进一步，为了精确地刻画满足互补条件的尖多面体的所有非负解，我们还需要根据互补条件，对 G^c 进行分组，并使得分组结果 $G^{cc} = \langle G^c_{s_1}, \ldots, G^c_{s_i}, \ldots, G^c_{s_m} \rangle$ 其中 $G^c_{s_i} = (V^c_{s_i}, R^c_{s_i})$ 满足如下条件：

1. $V^c_{s_i} \subseteq V^c$, $R^c_{s_i} \subseteq R^c$, $\cup_{i=1}^m V^c_{s_i} = V^c$, $\cup_{i=1}^m R^c_{s_i} = R^c$，且
2. 在每个组 $G^c_{s_i}$ 内，互补顶点 $V^c_{s_i}$ 上的任意凸组合与互补极射线 $R^c_{s_i}$ 上的任意非负组合之和都满足互补条件 $(x^+)^T x^- = 0$。

同样的，我们也可以通过生成无向图的所有极大完全子图来构造满足上述条件的分组。设采用该算法得到的分组结果为 $\tilde{G}^{cc} = \langle \tilde{G}^c_{s_1}, \ldots, \tilde{G}^c_{s_i}, \ldots, \tilde{G}^c_{s_m} \rangle$。对于其中任意某个组 $\tilde{G}^c_{s_i} = (\tilde{V}^c_{s_i}, \tilde{R}^c_{s_i})$，如果 $\tilde{V}^c_{s_i} = \emptyset$，则可以从 $\tilde{G}^{cc}$ 中删掉该组 $\tilde{G}^c_{s_i}$。接下来，对于某个互补极射线 $r^c_k \in R^c$，如果 r^c_k 不在经过上述处理后的 $\tilde{G}^{cc}$ 中出现，那么我们需要从 R^c 中删除 r^c_k。

定理 6.7 设满足互补条件的尖多面体 $P_\pm = \{x \in \mathbb{R}^{2n} \mid Ax \geq b, x \geq 0, (x^+)^T x^- = 0\}$ 的互补生成子集合的互补分组为 $G^{cc} = \langle G^c_{s_1}, \ldots, G^c_{s_i}, \ldots, G^c_{s_m} \rangle$ 其中 $G^c_{s_i} = (V^c_{s_i}, R^c_{s_i})$。那么，$x \in P_\pm$ 当且仅当存在某个 i $(i \in \mathbb{N}, 1 \leq i \leq m)$，使得

$$x = \sum_{v^c_j \in V^c_{s_i}} \lambda_j v^c_j + \sum_{r^c_k \in R^c_{s_i}} \mu_k r^c_k$$

其中 $\lambda_j, \mu_k \geq 0, \Sigma_j \lambda_j = 1$。

6.3.2 从生成子表示到约束表示的转化

给定某个满足互补条件的尖多面体 $P_\pm$ 的互补生成子集合 $G^c = (V^c, R^c)$，下面我们考虑如何构造出 $P_\pm$ 的约束表示，即 $P_\pm$ 所对应的 IHLCP 约束系统。这可以通过如下步骤计算得到：

1. 把 $G^c = (V^c, R^c)$ 看成是某个凸多面体的生成子表示，从而可以采用标准的 Chernikova 算法计算出其所对应的凸多面体约束表示，即一个线性不等式系统

$$M'x^+ + N'x^- \leq b'$$

2. 把 $x^+, x^- \geq 0$ 添加到上述线性不等式系统，并删除 $M'x^+ + N'x^- \leq b'$ 中那些因为添加了 $x^+, x^- \geq 0$ 而变成冗余的约束，设得到

$$\begin{aligned} Mx^+ + Nx^- &\leq b \\ x^+, x^- &\geq 0 \end{aligned}$$

3. 把 $(x^+)^T x^- = 0$ 添加到上述系统，从而得到

$$\begin{aligned} Mx^+ + Nx^- &\leq b \\ x^+, x^- &\geq 0 \\ (x^+)^T x^- &= 0 \end{aligned} \tag{6.20}$$

系统 (6.20) 即为 $P_\pm$ 所对应的 IHLCP 约束系统。

6.3.3 应用：绝对值线性规划

关于在线性规划中考虑绝对值的问题，早在上世纪 70 年代就有研究了 [191, 192]。但是，大部分研究仅考虑在目标函数中使用绝对值，即形如 $\max\{\Sigma_i c_i|x_i| : Ax \le b\}$，而且一般都是转换成多个标准线性规划问题。直到最近，Mangasarian 在文 [177] 中考虑了可行空间和目标函数中都出现绝对值的情形，给出了一些对偶性结论以及与原问题等价的其他形式，但最终还是转换成一系列标准线性规划问题来求解。下面，本文将考虑线性规划问题中可行空间和目标函数中都出现绝对值的情形，并给出一种新的方法来解绝对值线性规划问题。

根据第 6.2 节的结果，我们知道线性绝对值不等式系统与广义线性互补系统 IHLCP 之间可以进行等价转换。从而，如下绝对值线性规划问题：

$$\max\{cx + d|x| \colon Ax + B|x| \le b\}$$

等价于

$$\max\{\tfrac{c+d}{2}x^+ + \tfrac{d-c}{2}x^- \colon \tfrac{A+B}{2}x^+ + \tfrac{B-A}{2}x^- \le b, x^+ \ge 0, x^- \ge 0, (x^+)^T x^- = 0\}$$

其中，$x = x^+ - x^-, |x| = x^+ + x^-$。

从而，我们只需要考虑如下形式的优化问题

$$\max\{c'x^{\pm} : A'x^{\pm} \le b', x^{\pm} = (x^+\ x^-)^T, x^+ \ge 0, x^- \ge 0, (x^+)^T x^- = 0\} \qquad (6.21)$$

设 IHLCP 系统 $\{A'x^{\pm} \le b', x^{\pm} = (x^+\ x^-)^T, x^+ \ge 0, x^- \ge 0, (x^+)^T x^- = 0\}$ 的互补生成子表示为 $G^c = (V^c, R^c)$。那么，只存在如下三种情形：

1. 若 $G^c = \emptyset$，那么优化问题 (6.21) 没有可行解；
2. 若存在互补极射线 $r \in R^c$ 使得 $c'r > 0$，那么优化问题 (6.21) 有无界解；
3. 若对于 R^c 中的所有互补极射线 $r \in R^c$ 均有 $c'r \le 0$，那么优化问题 (6.21) 目标函数的最大值为

   $$\max\{c'v \colon v \in V^c\}$$

   并且，若 G^c 的互补分组为 $G^{cc} = \langle G^c_{s_1}, \ldots, G^c_{s_i}, \ldots, G^c_{s_m} \rangle$ 其中 $G^c_{s_i} = (V^c_{s_i}, R^c_{s_i})$，那么此时的最优解为

   $$S = \bigcup_{i=1}^{m} \{x \mid\ x = \Sigma_j \lambda_j \check{v}_j + \Sigma_k \mu_k \check{r}_k, \lambda_j \ge 0, \mu_k \ge 0, \Sigma_j \lambda_j = 1, \check{v}_j \in \check{V}^c_{s_i}, \check{r}_k \in \check{R}^c_{s_i}\}$$

   其中，$\check{V}^c_{s_i}$ 是 $V^c_{s_i}$ 中那些使得目标函数达到最大值的顶点集合，$\check{R}^c_{s_i}$ 是 $R^c_{s_i}$ 中那些使得目标函数为 0（即使得 $c'\check{r}_k = 0$）的极射线集合。

6.4 线性绝对值不等式域

本节提出一个新的数值抽象域—— **线性绝对值不等式抽象域**，简称 AVI 域。其主要思想是使用线性绝对值不等式约束来作为域表示方法。该抽象域可用来推导程序中变量 $x_k(k = 1, \ldots, n)$ 间形如 $\Sigma_k a_k x_k + \Sigma_k b_k |x_k| \leq c$ 的关系，其中常数 $a_k, b_k, c \in \mathbb{R}$ 由分析器自动推导出来。下面，本节将在实数 $\mathbb{R}$ 上来介绍 AVI 域的域表示以及域操作的实现方法。

6.4.1 域表示

在域表示上，类似于其他数值抽象域，AVI 域把程序环境看成是某类特定约束系统（即有穷个约束的合取）的解。这里，AVI 域选择线性绝对值不等式系统来作为其域表示。

在实数 $\mathbb{R}$ 上，一个 AVI 域元素 $\mathbf{P}$ 可通过一个线性绝对值不等式系统 $Ax + B|x| \leq c$ 来描述，其中实数矩阵 $A, B \in \mathbb{R}^{m \times n}$，实数向量 $c \in \mathbb{R}^m$，m 是不等式系统中约束的数目，n 是不等式系统中变量的个数。因为线性绝对值不等式系统与（弱解语义下的）区间线性不等式系统是等价的（见第 6.2.1 节），从而线性绝对值不等式系统的解集合在几何上也对应一个区间多面体 $\mathbf{P}$，其中每个点 $x \in \gamma(\mathbf{P})$ 代表一个可能的程序环境（即对所有变量 x 的一种可能赋值）。AVI 域元素和区间多面体具有相同的表达能力和性质，比如可以表示某种非凸甚至非连通性质且与 $\mathbb{R}^n$ 上每个象限的交必然是一个（可能为空的）凸多面体。

6.4.2 域操作

下面，我们详细描述 AVI 域上用于静态分析所需要的常用域操作的实现方法。

我们知道，经典凸多面体域上的域操作都是基于凸多面体的双重描述法来实现的。采用类似的思想，本文也基于绝对值线性不等式系统（即广义线性互补系统 IHLCP）的双重描述法来实现 AVI 域上的域操作。在实现时，我们维护变量 x 上抽象环境与变量 x^+, x^- 上抽象环境之间的映射关系，即

$$\begin{aligned} x &= x^+ - x^-, & |x| &= x^+ + x^- \\ x^+ &= \frac{1}{2}(x + |x|), & x^- &= \frac{1}{2}(|x| - x) \end{aligned}$$

其中 x^+, x^- 满足

$$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

从而，AVI 域上的域操作可以在所对应的广义线性互补系统 IHLCP 上基于双重描述法来实现。并且，在构造抽象域时，在 AVI 域元素 $\mathbf{P}$ 的互补生成子表示上，我们只需要 IHLCP 上的互补生成子集合 $G^c = (V^c, R^c)$，而不需要互补分组信息 G^{cc}。

下面，设 AVI 域元素 $\mathbf{P}$ 所对应的 IHLCP 约束系统为

$$Mx^+ + Nx^- \leq b$$
$$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

并记其所对应的互补生成子集合为

$$G^c = (V^c, R^c)$$

(1) 格相关操作

- 空元素测试：$\mathbf{P}$ 为空，当且仅当 $V^c = \emptyset$。

  下面，我们假设 AVI 域元素 $\mathbf{P}, \mathbf{P}'$ 非空，那么有

- 包含测试：$\mathbf{P} \sqsubseteq \mathbf{P}'$ 即 $\gamma(\mathbf{P}) \subseteq \gamma(\mathbf{P}')$，当且仅当

  $$\forall v \in V^c, M'v^+ + N'v^- \leq b' \quad \wedge \quad \forall r \in R^c, M'r^+ + N'r^- \leq 0$$

- 相等测试：$\gamma(\mathbf{P}) = \gamma(\mathbf{P}')$ 当且仅当 $\mathbf{P} \sqsubseteq \mathbf{P}' \wedge \mathbf{P}' \sqsubseteq \mathbf{P}$。
- 求交：$\mathbf{P} \sqcap \mathbf{P}'$ 定义为输入 $\mathbf{P}$ 与 $\mathbf{P}'$ 的约束系统的合取，即

  $$Mx^+ + Nx^- \leq b$$
  $$M'x^+ + N'x^- \leq b'$$
  $$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

- 接合：$\mathbf{P} \sqcup \mathbf{P}'$ 定义为输入 $\mathbf{P}$ 与 $\mathbf{P}'$ 的互补生成子的集合并，即 $(V^c \cup V'^c, R^c \cup R'^c)$。

(2) 迁移操作

- 条件测试迁移函数：$\tau[\![cx + d|x| \leq e]\!]^\#(\mathbf{P})$ 的约束系统定义为

  $$Mx^+ + Nx^- \leq b$$
  $$(c+d)x^+ + (d-c)x^- \leq e$$
  $$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

- 投影操作：$\tau[\![x_j := random()]\!]^\#(\mathbf{P})$ 的互补生成子集合定义为 $(V^c, R^c \cup \{e_j^+, e_j^-, -e_j^+, -e_j^-\})$，其中 e_j^+ 表示一个向量其中 x_j^+ 所对应的项为 1 而其他项为 0，e_j^- 表示一个向量其中 x_j^- 所对应的项为 1 而其他项为 0。
- 赋值迁移函数：$\tau[\![x_j := \Sigma_i a_i x_i + \Sigma_i b_i |x_i| + c]\!]^\#(\mathbf{P})$ 通过条件测试、投影、重命名来建模：

  $$\left(\tau[\![x_j := random()]\!]^\# \circ \tau[\![\Sigma_i a_i x_i + \Sigma_i b_i |x_i| + c - x'_j = 0]\!]^\#(\mathbf{P})\right)[x'_j/x_j]$$

(3) 外推操作

- 加宽算子：类似于凸多面体域上的标准加宽算子，给定两个基于 IHLCP 约束系统表示的 AVI 域元素 $\mathbf{P} \sqsubseteq \mathbf{P}'$，我们定义

$$\mathbf{P} \triangledown \mathbf{P}' \stackrel{\text{def}}{=} \mathcal{S}_1 \cup \mathcal{S}_2 \cup \{x^+, x^- \geq 0, (x^+)^T x^- = 0\}$$

其中

$$\mathcal{S}_1 = \{\, \varphi_1 \in (Mx^+ + Nx^- \leq b) \mid \mathbf{P}' \models \varphi_1 \,\},$$
$$\mathcal{S}_2 = \left\{ \varphi_2 \in (M'x^+ + N'x^- \leq b') \;\middle|\; \begin{array}{l} \exists \varphi_1 \in (Mx^+ + Nx^- \leq b), \\ \gamma(\mathbf{P}) = \gamma((\mathbf{P} \setminus \{\, \varphi_1 \,\}) \cup \{\, \varphi_2 \,\}) \end{array} \right\}$$

其中，$\varphi_1 \in (Mx^+ + Nx^- \leq b)$ 表示 φ_1 是 $Mx^+ + Nx^- \leq b$ 中的一个约束，不妨设 φ_1 为 $(cx^+ + dx^- \leq e)$，那么蕴涵关系 $\mathbf{P}' \models \varphi_1$ 可以通过检查

$$\forall v' \in V'^c, c\,v'^+ + d\,v'^- \leq e \quad \wedge \quad \forall r' \in R'^c, c\,r'^+ + d\,r'^- \leq 0$$

来实现。

下面，我们重点考虑 AVI 域上的接合操作 $\mathbf{P} \sqcup \mathbf{P}'$，并证明该接合操作是 AVI 域上的强接合（即计算得到的是包含两输入 AVI 域元素的最小 AVI 域元素）。并据此，进一步探讨 AVI 域的表达能力。

定理 6.8 给定 AVI 域元素 $\mathbf{P}$ 与 $\mathbf{P}'$，对于任意满足 $\gamma(\mathbf{P}) \subseteq \gamma(\mathbf{Q})$ 且 $\gamma(\mathbf{P}') \subseteq \gamma(\mathbf{Q})$ 的 AVI 域元素 $\mathbf{Q}$，皆有 $\gamma(\mathbf{P} \sqcup \mathbf{P}') \subseteq \gamma(\mathbf{Q})$。

证明. 设 AVI 域元素 $\mathbf{P}, \mathbf{P}', \mathbf{Q}$ 的互补生成子集合分别为 $(V^c, R^c), (V'^c, R'^c), (\hat{V}^c, \hat{R}^c)$。方便起见，我们记集合 S 的大小（或势）为 $|S|$。

因为 $\gamma(\mathbf{P}) \subseteq \gamma(\mathbf{Q})$，那么有

- 对于 $\forall v \in V^c$，v 可以通过 $\hat{V}^c, \hat{R}^c$ 来生成，即存在 λ_j, μ_k $(j = 1, \ldots, |\hat{V}^c|,\ k = 1, \ldots, |\hat{R}^c|)$ 满足

$$\lambda_j, \mu_k \geq 0, \Sigma_j \lambda_j = 1$$

使得

$$v = \sum_{j=1}^{|\hat{V}^c|} \lambda_j \hat{v}_j + \sum_{k=1}^{|\hat{R}^c|} \mu_k \hat{r}_k$$

- 对于 $\forall r \in R^c$，r 可以通过 $\hat{R}^c$ 来生成，即存在 μ_k $(k = 1, \ldots, |\hat{R}^c|)$ 满足

$$\mu_k \geq 0$$

使得

$$r = \sum_{k=1}^{|\hat{R}^c|} \mu_k \hat{r}_k$$

同理，因为 $\gamma(\mathbf{P}') \subseteq \gamma(\mathbf{Q})$，那么 V'^c, R'^c 中的元素也都可以通过 $(\hat{V}^c, \hat{R}^c)$ 来生成。

从而，$V^c \cup V'^c, R^c \cup R'^c$ 中的元素都可以通过 $(\hat{V}^c, \hat{R}^c)$ 来生成。因此有 $\gamma(\mathbf{P} \sqcup \mathbf{P}') \subseteq \gamma(\mathbf{Q})$。 □

从上述定理可知，$(V^c \cup V'^c, R^c \cup R'^c)$ 所对应的 AVI 域元素是包含了 $\mathbf{P}$ 和 $\mathbf{P}'$ 的最小 AVI 域元素。从而，$\mathbf{P} \sqcup \mathbf{P}'$ 是 AVI 域上的最佳接合操作。

定理 6.9 给定一组凸多胞体(即有界凸多面体)所构成的集合且每个象限内最多一个，这些凸多胞体的(集合)并可以通过一个相同变量集上的 AVI 域元素来精确描述。

证明. 首先，我们考虑 $\mathbb{R}^n$ 空间中某个象限 $\mathbf{O}$ 内的一个凸多胞体 $P = \{Ax \leq b, x \in \mathbf{O}\}$，其对应的 IHLCP 系统为

$$\begin{aligned} Ax^+ - Ax^- &\leq b \\ x^+ \geq 0, x^- &\geq 0 \\ x^+ - x^- &\in \mathbf{O} \\ (x^+)^T x^- &= 0 \end{aligned}$$

并设该 IHLCP 系统的互补生成子集合为 (V^c, R^c)。

下面，我们采用反证法来证明 $R^c = \emptyset$。假设存在某个互补极射线 $r \in R^c$，那么对于上述 IHLCP 的任意解 $(y^+\ y^-)^T$（由此可知 $y = y^+ - y^-$ 是凸多胞体 P 中的点），$(y^+\ y^-)^T + \mu r$（其中$\mu \geq 0$）也是上述 IHLCP 的解，从而 $y + \mu(r^+ - r^-)$ 也总是凸多胞体 P 中的点。因此，$r^+ - r^-$ 是 P 中的极射线。但是，多胞体不可能含极射线生成子。从而假设不成立，因此有 $R^c = \emptyset$。

设 $\{P_1, \ldots, P_i, \ldots P_{2^n}\}$ 为 $\mathbb{R}^n$ 空间上的一组多胞体且每个象限对应一个，并设其中的任意多胞体 P_i 所对应的 IHLCP 上的互补生成子集合为 $(V_i^c, \emptyset)$。我们可以构造一个 AVI 域元素 $\mathbf{Q}$，使得其互补生成子的互补分组为 $\langle (V_1^c, \emptyset), \ldots, (V_i^c, \emptyset), \ldots, (V_{2^n}, \emptyset) \rangle$，即其互补生成子集合为 $(V_1^c \cup \ldots \cup V_i^c \cup \ldots \cup V_{2^n}, \emptyset)$。根据互补分组的定义，可以知道 $\mathbf{Q}$ 精确地描述了 $\{P_1, \ldots, P_i, \ldots P_{2^n}\}$ 的集合并，即 $\gamma(\mathbf{Q}) = \gamma(P_1) \cup \ldots \cup \gamma(P_i) \cup \ldots \cup \gamma(P_{2^n})$。 □

推论 6.10 给定两个 AVI 域元素 $\mathbf{P}$ 与 $\mathbf{P}'$，如果两者与每个象限的交都分别是一个（可能为空的）凸多胞体，那么 $\mathbf{P} \sqcup \mathbf{P}'$ 计算结果与每个象限的交等价于 $\mathbf{P}$ 和 $\mathbf{P}'$ 位于该象限内的凸多胞体的多面体凸闭包结果。

然而，定理 6.9 和推论 6.10 对于无界多面体情形不一定成立。

例 6.4.1 考虑两个 AVI 域元素 $\mathbf{P}' = \{(x\ y)^T \mid 1 \le x \le 2, -1 \le y \le 1\} = \{(x^+\ x^-\ y^+\ y^-)^T \mid 1 \le x^+ - x^- \le 2, -1 \le y^+ - y^- \le 1, x^+ \ge 0, x^- \ge 0, y^+ \ge 0, y^- \ge 0, x^+x^- = 0, y^+y^- = 0\}$ 和 $\mathbf{P}'' = \{(x\ y)^T \mid -2 \le x \le -1\} = \{(x^+\ x^-\ y^+\ y^-)^T \mid -2 \le x^+ - x^- \le -1, x^+ \ge 0, x^- \ge 0, y^+ \ge 0, y^- \ge 0, x^+x^- = 0, y^+y^- = 0\}$，如图 *6.2(1)* 所示。注意，$\mathbf{P}'$ 是无界凸多面体，而 $\mathbf{P}''$ 是凸多胞体。

首先，如果我们不考虑互补条件 $x^+x^- = 0 \wedge y^+y^- = 0$，那么 $\mathbf{P}'$ 与 $\mathbf{P}''$ 在 $(x^+, x^-, y^+, y^-)^T$ 上的生成子为:

$$V_{\mathbf{P}'} = \left\{ \begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

$$R_{\mathbf{P}'} = \left\{ \begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \right\}$$

$$(V_{\mathbf{P}''}, R_{\mathbf{P}''}) = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \quad \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix} \right\}, \quad \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right)$$

考虑互补条件 $x^+x^- = 0 \wedge y^+y^- = 0$ 后，得到 $\mathbf{P}'$ 与 $\mathbf{P}''$ 在 $(x^+, x^-, y^+, y^-)^T$ 上的互补生成子如下:

$$(V^c_{\mathbf{P}'}, R^c_{\mathbf{P}'}) = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}, \emptyset \right)$$

$$(V^c_{\mathbf{P}''}, R^c_{\mathbf{P}''}) = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right)$$

因此，$\mathbf{P}' \sqcup \mathbf{P}''$ 的互补生成子集合为 $(V^c_{\mathbf{P}'} \cup V^c_{\mathbf{P}''}, R^c_{\mathbf{P}'} \cup R^c_{\mathbf{P}''})$，其中

$$V^c_{\mathbf{P}'} \cup V^c_{\mathbf{P}''} = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix} \right\} \right)$$

$$R^c_{\mathbf{P}'} \cup R^c_{\mathbf{P}''} = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right)$$

在删除冗余生成子后，我们将得到 $\mathbf{P}' \sqcup \mathbf{P}''$ 的互补生成子集合为:

$$\left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right)$$

转化成约束表示后，得到 $\mathbf{P}' \sqcup \mathbf{P}''$ 的 IHLCP 约束系统为

$$\{1 \le x^+ \le 2, 1 \le x^- \le 2, 1 \le x^+ + x^- \le 2, x^+ \ge 0, x^- \ge 0, x^+ x^- = 0\}$$

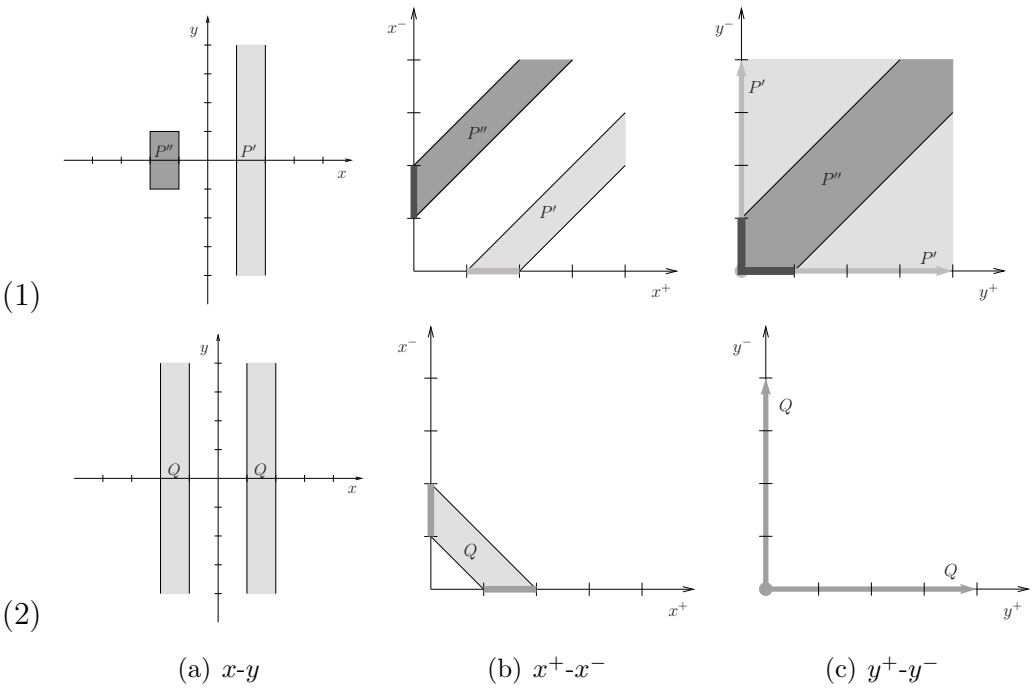
删除冗余约束后得到

$$\{1 \le x^+ + x^- \le 2, \quad x^+ \ge 0, x^- \ge 0, x^+ x^- = 0\}$$

进一步可得到 $\mathbf{P}' \sqcup \mathbf{P}''$ 的线性绝对值不等式表示:

$$\mathbf{P}' \sqcup \mathbf{P}'' = \{(x\ y)^T \mid 1 \le |x| \le 2\}$$

在图 *6.2* 中，(1) 子图刻画了接合操作的输入元素 $\mathbf{P}'$ 和 $\mathbf{P}''$，而 (2) 子图刻画了 AVI 域上的接合结果 $\mathbf{Q} = \mathbf{P}' \sqcup \mathbf{P}''$。并且，分别在 (a) x-y，(b) x^+-x^-，(c) y^+-y^- 平面上给出了这些 AVI 域元素所对应的图形区域。



(a) x-y　　(b) x^+-x^-　　(c) y^+-y^-

图 6.2 线性绝对值不等式域上的接合操作

6.5 线性绝对值等式域

本节提出一个新的数值抽象域—— **线性绝对值等式抽象域**，简称 AVE 域。其主要思想是使用线性绝对值等式约束来作为域表示方法。该抽象域可用来推导程序中变量 $x_k(k = 1, \ldots, n)$ 间形如 $\Sigma_k a_k x_k + \Sigma_k b_k |x_k| = c$ 的关系，其中常数 $a_k, b_k, c \in \mathbb{R}$ 由分析器自动推导出来。下面，本节将在实数 $\mathbb{R}$ 上来介绍 AVE 域的域表示以及域操作的实现方法。

6.5.1 域表示

在域表示上，AVE 域选择线性绝对值等式系统来作为其域表示。AVE 域把程序环境看成是有穷线性绝对值等式系统上的解。

在实数 $\mathbb{R}$ 上，一个 AVE 域元素 $\mathbf{P}$ 可通过一个线性绝对值等式系统 $Ax + B|x| = c$ 来描述，其中实数矩阵 $A, B \in \mathbb{R}^{m \times n}$，实数向量 $c \in \mathbb{R}^m$，m 是不等

式系统中约束的数目，n 是不等式系统中变量的个数。且总有 $m \leq 2n$，因为任意线性绝对值等式系统 $Ax + B|x| = c$ 都可以采用高斯消去法化简成关于变量序 $x_1 \prec |x_1| \prec x_2 \prec |x_2| \prec \ldots \prec x_n \prec |x_n|$ 的行阶梯形。每个 AVE 域元素都是 AVI 元素，即都是区间多面体。在域表示的表达能力上看，AVE 域是 AVI 域的等式限制版本，是经典仿射等式域 [39] 的绝对值扩展版本。AVE 域元素也可以表示某种非凸甚至非连通性质，且与 $\mathbb{R}^n$ 上每个象限的交必然是一个（可能为空的）仿射空间，即一个仿射等式域上的域元素。

6.5.2 域操作

下面，我们详细描述 AVE 域上用于静态分析所需要的常用域操作的实现方法。

基于 Chernikova 于 1964 年在文 [187] 中所提出的用来描述仿射等式系统所有非负解的生成子构造算法，并采用与第 6.3 节同样的思路，我们可以构造出水平线性互补系统 HLCP 的双重描述法。下面，本文基于 HLCP 的双重描述法来实现 AVE 域上的域操作。在实现时，我们维护变量 x 上抽象环境与变量 x^+, x^- 上抽象环境之间的映射关系，即

$$\begin{aligned} x &= x^+ - x^-, & |x| &= x^+ + x^- \\ x^+ &= \frac{1}{2}(x + |x|), & x^- &= \frac{1}{2}(|x| - x) \end{aligned}$$

其中 x^+, x^- 满足

$$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

从而，AVE 域上的域操作可以在所对应的水平线性互补系统 HLCP 上基于双重描述法来实现。并且，构造抽象域时，在 AVE 域元素 $\mathbf{P}$ 的互补生成子表示上，我们只需要 HLCP 上的互补生成子集合 $G^c = (V^c, R^c)$，而不需要其互补分组信息 G^{cc}。

下面，设 AVE 域元素 $\mathbf{P}$ 所对应的 HLCP 约束系统为

$$\begin{aligned} Mx^+ + Nx^- &= b \\ x^+ \geq 0, x^- \geq 0, (x^+)^T x^- &= 0 \end{aligned}$$

并记其所对应的互补生成子集合为

$$G^c = (V^c, R^c)$$

(1) 格相关操作

- 空元素测试：$\mathbf{P}$ 为空，当且仅当 $V^c = \emptyset$。

  下面，我们假设 AVE 域元素 $\mathbf{P}, \mathbf{P}'$ 非空，那么有

- 包含测试：$\mathbf{P} \sqsubseteq \mathbf{P}'$ 即 $\gamma(\mathbf{P}) \subseteq \gamma(\mathbf{P}')$，当且仅当

$$\forall v \in V^c, M'v^+ + N'v^- = b' \quad \wedge \quad \forall r \in R^c, M'r^+ + N'r^- = 0$$

- 相等测试：$\gamma(\mathbf{P}) = \gamma(\mathbf{P}')$ 当且仅当 $\mathbf{P} \sqsubseteq \mathbf{P}' \wedge \mathbf{P}' \sqsubseteq \mathbf{P}$。
- 求交：$\mathbf{P} \sqcap \mathbf{P}'$ 定义为输入 $\mathbf{P}$ 与 $\mathbf{P}'$ 的约束系统的合取，即

$$\begin{aligned} Mx^+ + Nx^- &= b \\ M'x^+ + N'x^- &= b' \\ x^+ \geq 0, x^- \geq 0, (x^+)^T x^- &= 0 \end{aligned}$$

  其中，$\{Mx^+ + Nx^- = b, M'x^+ + N'x^- = b'\}$ 可以进一步通过高斯消去法化简成关于变量序 $x_1 \prec |x_1| \prec x_2 \prec |x_2| \prec \ldots \prec x_n \prec |x_n|$ 的行阶梯形，从而消除部分冗余约束。

- 接合：AVE 域上的接合操作 $\mathbf{P} \sqcup_{\text{AVE}} \mathbf{P}'$ 通过如下步骤来计算：
  1. 首先，计算输入 $\mathbf{P}$ 与 $\mathbf{P}'$ 的互补生成子的集合并，即 $(V^c \cup V'^c, R^c \cup R'^c)$。不妨设 $V^c \cup V'^c = \{v_1, \ldots, v_p\}, R^c \cup R'^c = \{r_1, \ldots, r_q\}$；
  2. 采用高斯消去法从如下等式系统中投影掉变量 $\lambda_j (j = 1, \ldots, p), \mu_k (k = 1, \ldots, q)$

$$\begin{cases} (x^+ \ x^-)^T = \sum_{j=1}^{p} (\lambda_j v_j) + \sum_{k=1}^{q} (\mu_k r_k) \\ \sum_{j=1}^{p} \lambda_j = 1 \end{cases}$$

  就可以得到一个关于变量 x^+, x^- 的线性等式系统

$$\hat{M}x^+ + \hat{N}x^- = \hat{b}$$

接合操作 $\mathbf{P} \sqcup_{\text{AVE}} \mathbf{P}'$ 结果所对应的 HLCP 系统定义为：

$$\begin{aligned} \hat{M}x^+ + \hat{N}x^- &= \hat{b} \\ x^+ \geq 0, x^- \geq 0, (x^+)^T x^- &= 0 \end{aligned}$$

(2) 迁移操作

- 条件测试迁移函数：
  - 等式型条件测试：$\tau[\![cx + d|x| = e]\!]^{\#}(\mathbf{P})$ 的 HLCP 约束系统定义为

$$\begin{aligned} Mx^+ + Nx^- &= b \\ (c+d)x^+ + (d-c)x^- &= e \\ x^+ \geq 0, x^- \geq 0, (x^+)^T x^- &= 0 \end{aligned}$$

    其中，$\{Mx^+ + Nx^- = b, (c+d)x^+ + (d-c)x^- = e\}$ 可以进一步通过高斯

消去法化简成关于变量序 $x_1 \prec |x_1| \prec x_2 \prec |x_2| \prec \ldots \prec x_n \prec |x_n|$ 的行阶梯形，从而消除部分冗余约束。

– 非等式型条件测试：我们定义

$$\tau[\![cx + d|x| \leq e]\!]^{\sharp}(\mathbf{P}) = \begin{cases} \bot^{\sharp} & \text{if } \min\{cx + d|x| : \mathbf{P}\} > e \\ \mathbf{P} & \text{otherwise} \end{cases}$$

- 投影操作：$\tau[\![x_j := random()]\!]^{\sharp}(\mathbf{P})$ 可通过如下方式来实现：采用高斯消去法从

$$Mx^+ + Nx^- = b$$

中消除掉 x_j^+, x_j^-，设得到 $M'x^+ + N'x^- = b'$。那么，$\tau[\![x_j := random()]\!]^{\sharp}(\mathbf{P})$ 的结果 HLCP 约束系统即为

$$M'x^+ + N'x^- = b'$$

$$x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0$$

- 赋值迁移函数：$\tau[\![x_j := \Sigma_i a_i x_i + \Sigma_i b_i |x_i| + c]\!]^{\sharp}(\mathbf{P})$ 通过条件测试、投影、重命名来建模：

$$\left(\tau[\![x_j := random()]\!]^{\sharp} \circ \tau[\![\Sigma_i a_i x_i + \Sigma_i b_i |x_i| + c - x'_j = 0]\!]^{\sharp}(\mathbf{P})\right)[x'_j/x_j]$$

(3) 外推操作

- 加宽算子：我们知道，仿射等式抽象域 [39] 满足递增链条件（因为程序中变量的数目是有穷的，从而程序中仿射等式所构成的格的高度是有穷的）。而 AVE 域元素与每个象限的交必然是一个（可能为空的）仿射空间，即一个仿射等式抽象域中的域元素，并且象限的个数是有穷的，从而 AVE 域也满足递增链条件。所以，在 AVE 域中，我们并不需要加宽算子来保证程序不动点迭代的终止性。在程序加宽点，我们使用接合操作 $\sqcup_{\text{AVE}}$ 来代替加宽算子。

例 6.5.1 给定两个 AVE 域元素 $\mathbf{P}' = \{(x\ y)^T \mid x - y = 0\} = \{(x^+\ x^-\ y^+\ y^-)^T \mid x^+ - x^- - y^+ + y^- = 0, x^+ \geq 0, x^- \geq 0, y^+ \geq 0, y^- \geq 0, x^+x^- = 0, y^+y^- = 0\}$ 和 $\mathbf{P}'' = \{(x\ y)^T \mid x + y = 0\} = \{(x^+\ x^-\ y^+\ y^-)^T \mid x^+ - x^- + y^+ - y^- = 0, x^+ \geq 0, x^- \geq 0, y^+ \geq 0, y^- \geq 0, x^+x^- = 0, y^+y^- = 0\}$。注意，$\mathbf{P}'$ 和 $\mathbf{P}''$ 其实也都是仿射等式抽象域中的域元素，但是其仿射包结果和多面体凸闭包结果都将是全局空间。

$\mathbf{P}', \mathbf{P}''$ 在 $(x^+, x^-, y^+, y^-)^T$ 上的互补生成子集合分别为

$$(V^c_{\mathbf{P}'}, R^c_{\mathbf{P}'}) = \left(\begin{pmatrix} x^+ \\ x^- \\ y^+ \\ y^- \end{pmatrix} : \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \right)$$

$$(V^c_{\mathbf{P}''}, R^c_{\mathbf{P}''}) = \left(\left(\begin{array}{c} x^+ \\ x^- \\ y^+ \\ y^- \end{array} \right) : \left\{ \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array} \right) \right\}, \left\{ \left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \right), \left(\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \end{array} \right) \right\} \right)$$

那么从系统

$$\left(\begin{array}{c} x^+ \\ x^- \\ y^+ \\ y^- \end{array} \right) = \mu_1 \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ 1 \end{array} \right) + \mu_2 \left(\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \end{array} \right) + \mu_3 \left(\begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \end{array} \right) + \mu_4 \left(\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \end{array} \right)$$

中消除 $\mu_1, \mu_2, \mu_3, \mu_4$ 后将得到

$$x^+ + x^- - y^+ - y^- = 0$$

从而，可知 $\mathbf{P} \sqcup_{\text{AVE}} \mathbf{P}' = \{(x\ y)^T \mid |x| - |y| = 0\}$。

6.6 实现及实验

(1) 线性绝对值抽象域的有理数实现

本文采用多精度有理数，为本章提出的线性绝对值不等式抽象域开发了原型系统 rAVI，并为本章提出的线性绝对值等式抽象域开发了原型系统 rAVE。这些原型系统都是基于 GMP 库 [81] 中的多精度有理数实现的。

同样地，本文也把 rAVI 和 rAVE 适配到为数值抽象域的开发提供了通用接口的数值抽象域库 Apron[86] 中。并使用支持 Apron 库的静态分析工具 Interproc[87] 开展了一系列实验。

(2) 线性绝对值抽象域实验

由于 rAVE 域可以看作是 rAVI 域的特殊子域，本文下面主要给出 rAVI 域的实验结果。为了评估 rAVI 的分析精度和计算效率，我们把 rAVI 得到的不变式及其性能和前面第 4 章的工作 itvPol（区间多面体域的可靠浮点实现）进行了比较。

首先，为了展示 rAVI 的表达能力，我们给出三个简单而又具有代表性的程序以及分析器所产生的不变式，如图 6.3-6.5 所示。其中，图 6.3 所示的 AVtest1 程序的初始状态为 x-y 平面上的四个点 $(1,1), (-1,1), (-1,-1), (1,-1)$，分别分布在四

real x, y;
assume $x = 1$ or $x = -1$;
assume $y = 1$ or $y = -1$;
while ($true$) {
① if ($x \geq 0$) { $x := x + 1$; }
  else { $x := x - 1$; }
  if ($y \geq 0$) { $y := y + 1$; }
  else { $y := y - 1$; }
}

位置	区间多面体域	线性绝对值不等式域
①	$[-1,1]x \leq -1$ $\wedge[-1,1]y \leq -1$	$\|x\| = \|y\| \wedge \|x\| \geq 1$

图 6.3 绝对值相关程序 AVtest1 及产生的不变式

个不同象限，程序中循环过程实际是对每个象限里的点沿 $y = x$ 或 $y = -x$ 方向同步增长。在程序点 ① 处，rAVI 可以证明 $|y| = |x| \wedge |x| \geq 1$，但是 itvPol 只能证明 $[-1,1]x \leq -1 \wedge [-1,1]y \leq -1$（即 $|x| \geq 1 \wedge |y| \geq 1$）而不能发现 x 与 y 之间的任何关系。

Simple(int $x0$, n)
  $x := x0; i := 0$;
  $t := n - x0$;
  while ($x < n$) {
    $i := i + 1$;
    $x := x + 1$;
  } ①

位置	区间多面体域	线性绝对值不等式域
①	关于 i 的不变式： $i \geq 0$	关于 i 的不变式： $i \geq 0$ $\wedge i \leq \frac{1}{2}(\|t\| + t)$ $\wedge i = x - x0$ $\wedge \ldots$

图 6.4 时间复杂度分析相关程序 CmplxTest1 及产生的不变式

图 6.4 所示的 CmplxTest1 程序源于文 [50] 中作为分析程序时间复杂度的例子，分析该循环的时间复杂度主要在于分析循环计数器 i 的上界。但是，itvPol 域不能为 i 找到任何上界，而 rAVI 域可以证明 $i \leq \frac{1}{2}(|t| + t)$，即 $i \leq \max(0, n - x0)$，从而可以证明该程序关于输入参数 $x0, n$ 的时间复杂度为 $\max(0, n - x0)$。

图 6.5 所示的 Synergy1 程序源于文 [193]，该程序经常被 SLAM [12]，BLAST[194] 和 SYNERGY [193] 等路径敏感的工具用来展示基于谓词抽象的反例制导抽象精化过程。本文对原程序稍微做了修改，我们使用 $lock = -1$ 来表示 Unlocked 状态（而不是通常使用的 $lock = 0$）。并且，我们引入了两个新的变量 t, s 来分别表示 $x - y$

```
int x, y, lock, t, s;
lock := −1;    /* -1 means Unlocked */
do {
    lock := 1;    /* 1 means Locked */
    x := y;
    if (brandom) { lock := −1;   y := y + 1; }
    t = x − y;
} while (t ≠ 0);
s = lock − 1;
if (s ≠ 0) { ①: error; }
```

图 6.5 反例制导抽象精化相关程序 Synergy1

和 $lock - 1$。rAVI 域可以证明在循环执行后 $lock = 1$，从而说明程序点 ① 处的错误不可达，即程序是正确的。但是 itvPol 域不能证明程序点 ① 处不可达。

表 6.1 给出并比较了基于 rAVI 和 itvPol 两种抽象域的程序分析的实验结果。其中，测试程序中的 AVtest1、CmplxTest1、Synergy1 分别对应于图 6.3-6.5 中所示程序。CmplxTest1-3 源自文 [50] 中的用来分析程序时间复杂度的示例程序。program4 和 program5 分别对应第 4.5 节的图 4.6 和图 4.7 中所示程序。对于每个程序，“#vars” 表示程序中变量个数。对于同一个程序，INTERPROC 的加宽算子延迟参数（加宽算子应用之前的迭代次数）设置成相同的。“#iter.” 栏给出了总的（递增）迭代次数。

不变式. “Res. Inv.” 栏比较了所获得的不变式：“>”（“=”）表示 rAVI 找到的不变式比 itvPol 更强（相等）。从表 6.1 不难看出，大部分情形下 rAVI 产生的不变式都比 itvPol 强，因为两者虽然在域表示的表达能力上相同，但是 rAVI 的域操作都是最佳抽象而 itvPol 中大部分域操作都是弱的操作（比如弱接合）。从而，在不变式产生的能力上，rAVI 强于 itvPol。对于 program4 而言，两者产生的不变式相同，是因为该程序只涉及一个变量，此时 itvPol 域中大部分域操作也都是最佳抽象。通过对实验所得不变式进行分析，我们发现 rAVI 产生的许多线性绝对值不变式本质上都源于程序的分段线性语义，比如循环内有分支语句。对于时间复杂度分析示例程序 CmplxTest1-3，分段线性语义主要源于对循环变量与输入参数（或初始值）之间差值所进行的分情况讨论，比如差值大于 0 还是小于 0 等。注意，对于表 6.1 中

表 6.1 线性绝对值不等式域的实验结果

Program		rAVI		itvPol			Res.
name	#vars	#iter.	$t(ms)$	#iter.	$t(ms)$	#lp	Inv.
AVtest1	2	4	48	4	45	732	$>$
AVtest2	2	4	31	3	14	100	$>$
AVtest3	2	5	73	4	16	197	$>$
CmplxTest1	5	4	57	4	26	399	$>$
CmplxTest2	5	7	150	6	34	493	$>$
CmplxTest3	8	5	369	4	328	4278	$>$
Synergy1	5	4	109	4	123	1958	$>$
program4	1	4	10	4	4	30	$=$
program5	2	8	45	5	20	270	$>$

的测试程序，传统的凸抽象域（如凸多面体域）的分析能力都比较弱，因为这些测试程序涉及了凸抽象域所不能刻画的非凸性质。

性能. 表 6.1 在“t(ms)” 栏以毫秒为单位给出了程序分析所花费的时间，其实验平台为：Fedora Linux 操作系统，2GB 物理内存，Intel(R) Core(TM)2 CPU 6600 2.4 GHz处理器。从表 6.1 可以看到，相比 itvPol 而言，rAVI 的计算代价相对较高。其主要原因在于 rAVI 域是基于多精度有理数实现的，而 itvPol 域是基于浮点数实现的且 itvPol 域上的操作都是计算代价相对较低的弱的操作。“#lp”栏还给出了 itvPol 域中 LP 查询（调用 GLPK）的统计次数。

6.7 小结

本章把绝对值引入到抽象域的设计中，提出了两种线性绝对值抽象域：线性绝对值不等式域和线性绝对值等式域。首先，从数学上，本章给出并证明了线性绝对值关系、广义线性互补关系、区间线性关系三者之间的某种等价关系，这些关系本质上都可以用来描述分段线性特征。然后，基于经典凸多面体的双重描述法，给出了广义线性互补问题（系统）的双重描述法，并作为其应用还给出了一种求解绝对值线性规划问题的新方法。在此基础上，本章设计了线性绝对值不等式抽象域（用来推导线性绝对值不等式/广义线性互补不等式/区间线性不等式关系）和线性绝对值

等式抽象域（用来推导线性绝对值等式/水平线性互补等式关系）。其中，线性绝对值不等式抽象域是经典多面体抽象域 [40] 的一般化，在域表示上其表达能力与第 4 章介绍的区间多面体域相同；而线性绝对值等式抽象域则是经典仿射等式抽象域[39]的一般化。这两个线性绝对值抽象域都可以表达某种非凸甚至非连通性质，且域操作都是具体域上对应操作的最佳抽象。最后，本章采用多精度有理数为这两个抽象域开发了原型系统，并开展了实验。实验结果说明：实际中，线性绝对值抽象域可以发现程序中一些有意义的线性绝对值不变式，尤其对于包含分段线性语义程序的分析有着重要意义。

第七章　结束语

7.1　工作总结

本文主要研究了抽象解释框架下数值抽象域的设计和实现，从改进分析精度和提高计算效率两方面对已有数值抽象域进行了拓展，旨在进一步推动数值抽象域在实际程序分析与验证中的应用。

本文工作主要包括如下三个方面：

1. 数值抽象域的可靠浮点实现：针对多精度有理数实现所存在的计算效率和可扩展性问题，本文基于浮点数来实现数值抽象域并保证浮点实现的可靠性；
2. 非凸数值抽象域的设计和实现：针对大部分已有数值抽象域在表达能力方面所存在的凸性局限性，本文设计并实现了一系列可表达非凸性质的数值抽象域；
3. 支持区间系数数值抽象域的设计和实现：已有数值抽象域尚不支持实际程序分析中常出现的区间系数，为此，本文设计并实现了一系列在域表示上直接支持区间系数的数值抽象域。

本文的主要贡献具体总结如下：

- 给出了经典凸多面体抽象域的可靠浮点实现方法

  凸多面体抽象域是目前表达能力最强、应用最广泛的数值抽象域之一。本文提出了浮点多面体域（$\Sigma_i a_i x_i \leq c$，其中 a_i, c 为浮点数），作为凸多面体域的一种可靠浮点实现方法，以通过浮点实现来提高凸多面体域的计算效率。另外，考虑到凸多面体域的处理能力主要受限于其高代价的接合操作，本文提出了一系列低代价的弱接合操作以进一步提高凸多面体域的计算效率和可扩展性。
- 提出了区间多面体抽象域用以推导区间线性不等式关系

  本文把区间线性代数引入到程序分析中，提出了区间多面体抽象域（$\Sigma_i [a_i, b_i] x_i \leq c$），用以推导程序中变量间的区间线性不等式关系。该域是经典凸多面体域的一般化，且能够表达某类拓扑非凸（甚至非连通）性质。本文采用浮点数可靠地实现了区间多面体域。此外，作为一种简单情形，本文提出了单变量区间线性不等式抽象域，用以推导单个变量的（非连通）取值范围。
- 提出了具有多项式时空复杂度的行阶梯形区间线性等式抽象域

  本文提出了行阶梯形区间线性等式抽象域（$\Sigma_i [a_i, b_i] x_i = [c, d]$），用以推导变量间的区间线性等式关系。该域是经典仿射等式域的一般化，其表示方法基于（可

能带无穷区间系数的）区间线性等式的行阶梯形系统，也可以表示某类拓扑非凸（甚至非连通、非封闭）性质。并且，其域操作存在多项式时间的可靠浮点实现算法。该域已经适配到静态分析工具 Astrée 中，并对工业界航空领域嵌入式命令控制程序开展了分析实验，程序规模达 49 万行代码共涉及 5 万多个（全局）变量。

- 给出了广义线性互补问题（所对应约束系统）的双重描述法

  本文给出了广义线性互补问题 IHLCP（或称 XLCP，形如 $\{Mx^+ + Nx^- \leq q, x^+ \geq 0, x^- \geq 0, (x^+)^T x^- = 0\}$）所对应约束系统的双重描述法（包括约束表示和生成子表示）。还给出并证明了线性绝对值不等式系统、广义线性互补问题（所对应约束系统）、区间线性不等式系统三者之间的等价性，以及线性绝对值等式系统、水平线性互补问题 HLCP（所对应约束系统）两者之间的等价性。从而，该双重描述法可以推广到其他等价的约束系统中。由此，本文还给出了一种求解绝对值线性规划问题的新方法。

- 提出了线性绝对值不等式/等式抽象域用以推导线性绝对值关系

  本文提出了线性绝对值不等式抽象域（用来推导线性绝对值不等式 $\Sigma_i a_i x_i + b_i|x_i| \leq c$ /广义线性互补不等式/区间线性不等式关系）和线性绝对值等式抽象域（用来推导线性绝对值等式 $\Sigma_i a_i x_i + b_i|x_i| = c$ /水平线性互补等式关系）。本文基于广义线性互补问题的双重描述法并采用多精度有理数实现了这两个抽象域。线性绝对值不等式域是经典凸多面体域的一般化且表达能力与区间多面体域相同，而线性绝对值等式域是经典仿射等式域的一般化，并且这两个域都是非凸的且域操作都是具体域上对应操作的最佳抽象。

7.2 研究展望

进一步的工作包括如下几个方面。

- 基于浮点抽象域的程序分析

  本文主要关注基于浮点实现的抽象域的可靠性。但是，浮点舍入误差可能影响到浮点抽象域的分析精度，并可能给不动点计算带来扰动。因此，下一步将研究一些通用性策略来提高浮点抽象域的分析精度和稳定性，比如在接合、加宽等操作中考虑一些具有规整系数（如整数、$2^{\pm k}$ 等）的模版约束。在第 3.5 节，我们看到当前浮点线性规划工具可能出现“数值不稳定性问题”从而影响到分析的性能甚至导致分析无法完成，因此还将研究鲁棒性更好的浮点线性规划技术。

- 基于区间线性代数其他解语义的抽象域设计

  本文基于区间线性代数设计了一些数值抽象域，但是目前采用的都是区间线性系统的弱解语义。而实际上，在区间线性代数领域，还存在其他关于区间线性系统的解的语义解释，包括强解、容限解、控制解、代数解等。不同的解语义可以表达不同的性质，面向不同的应用。因此，下一步将研究基于区间线性代数其他解语义来设计一些面向特定应用的抽象域。

- 非凸抽象域与凸抽象域的有机结合

  本文关注非凸抽象域。非凸抽象域在表达能力方面具有一定优势，但是在计算效率方面可能存在不足。因此，下一步将考虑以一种灵活的方式把非凸抽象域和凸抽象域结合起来，在计算效率和分析精度之间进行权衡。比如，凸多面体是区间多面体的一个特殊类型，如果区间多面体域中的计算比较复杂时，可以考虑把区间多面体转化成凸多面体，然后在凸多面体域上进行对应的操作。

- 非凸抽象域表达能力的进一步提升

  本文提出的非凸抽象域实际上只能表达某类非凸性质，而非任意形式的非凸性质。比如，线性绝对值不等式域要求域表示中的绝对值仅应用在单个变量上，从而不能直接表达类似 $-|x+2y| \leq -1$ 的性质。但是，通过引入新变量 $z = x+2y$ 并结合 $-|z| \leq -1$，就可以表达 $-|x+2y| \leq -1$。因此，下一步将研究在程序分析过程中动态地引入一些新变量来提升非凸抽象域的表达能力和分析能力。此外，还将考虑对行阶梯形区间线性等式域中的行阶梯形式进行松弛以提升表达能力，比如，为每个首项变量维护两条约束。

- 本文抽象域实现的进一步完善和应用

  目前，本文仅把行阶梯形区间线性等式域适配到 Astrée 中并开展了一些大规模的实验。下一步将完善其他抽象域的实现，进行更为详尽的测试，并在实际大规模的程序分析中开展一些实验。另外，本文目前仅考虑了本文抽象域在程序分析中的应用，实际上这些抽象域还可以应用到混成系统的验证、程序自动并行化等其他领域。比如，线性绝对值不等式域本质上可以表达分段线性性质，因此可以考虑在分段线性混成系统验证中的应用。

致　谢

深切缅怀我的导师陈火旺教授。陈老师一生治学严谨、学养丰厚、造诣深湛。陈老师晚年虽然由于身体原因不适合长时间工作和谈话，但依然坚持定期组织师门进行学术讨论。讨论时，陈老师常常不能自已地想把发现的问题和想法详详细细地表达出来。那种教人不诲的殷切之情溢于言表，但是伴随的阵阵咳嗽声让学生心中既感动又倍感心痛。忘不了，陈老师在病榻上为师兄们修改论文，在病房里和学生们讨论学术问题的情形。依然记得在我赴法留学之前和陈老师的谈话，其中饱含师长教诲和鼓励，让我终生难忘。不曾想，那次谈话竟成了永别。如今陈老师虽然远逝，但是老师的谆谆教诲和常青风范，将永远清晰地留在我的记忆深处，激励我奋然前行。同时，感谢我的师母祖沛南女士一直以来对我的关心和鼓舞。

感谢我的导师王戟教授。记得刚上硕士的时候，王老师给我的第一篇文献是《Specifying systems》，指引我从大师之作中汲取营养，扎实基础。在后面接近一年的研读和讨论中，让我对系统的建模和验证等基础理论有了深刻的理解。王老师学术洞察力强，对问题往往有独到深刻的见解。每次和王老师的学术讨论都能使我得到新的启发和思路，让我受益匪浅。一直以来，王老师总是能够从一个更高的角度、更开阔的视野、更长远的眼光来启发我，鼓励我，让我充满憧憬和力量继续前行。王老师坚持勤奋工作、追求学术卓越的品质，在潜移默化中深深地影响着我。同时，博士期间，王老师对我在生活等其他方面也给予了很多关心，让我倍感温暖。

感谢法国巴黎高师 Patrick Cousot 教授。能在抽象解释创立者的指导下学习抽象解释并开展相关研究，是我的莫大荣幸。感谢教授为了照顾我而改用英语讲授抽象解释的两门课程，让我有幸能系统性地学到整个理论框架。课程结束时 50 页的 90 道考题也让我深深领略到教授严谨的治学态度。感谢教授对我的指导，他的指引让我在研究中少走了很多弯路。尤其在我博士课题开始阶段，与教授每周一次的讨论，让我受益匪浅。感谢教授的鼓励和对我学术方面的训练，让我树立自信。教授年过 60 依然坚持在研究的第一线，亲自编代码写文章；博士毕业 30 多年，依然坚持完善自己博士论文相关工作。这种学无止境、奋斗不息的精神将值得我学习一生。

感谢法国巴黎高师 Antoine Miné 博士。在我博士课题开展期间，博士对我的工作给予了全方位的帮助，使我的工作进展顺利。与博士的讨论都是在白板上进行的，博士总能把我不合理的想法擦除，把我粗浅的想法进行提升和完善。博士常能把我描述不清的问题和思路，从另外一个角度用漂亮的数学公式清楚地展现出来。

博士对我的工作倾注了大量的心血和时间，我的进展和进步与他的帮助密不可分。

感谢德国慕尼黑工业大学 Axel Simon 博士。在法学习期间，我们在同一个办公室共同奋斗了两年。他总是耐心热情地给我答疑解惑，对我的研究提出了很多中肯的意见和建议。留法的日子里，他给予了我如大哥般的关心和照顾。

感谢董威老师、易晓东师兄在我硕博阶段对我的引领和指导。感谢 602 教研室的毛晓光、毛新军、谭庆平、王挺、李暾、文艳军等各位老师对我的指导和帮助。感谢计算机系的宁洪、唐玉华、罗宇、姜新文、姜晶菲等各位老师对我的关心和鼓励。

感谢刘万伟师兄对我的悉心指导和帮助，师兄闻道在先，学业有成，一直是我学习的榜样。并感谢师兄所提供的 LaTeX 模板，本文就是在其模版上修改得到的。

感谢颜炯、邓欣、颜跃进、于洋、陈波、王涛、张丽娟、马晓东、沈锐、陈振邦、张晓艳、陈耀东、虞蕾等师兄、师姐对我的帮助；感谢王昭飞、张圣栋、李仁见、陈杰、樊沛、董龙明、张羽丰、樊华、徐厚峰、黄海、沈思淇、侯苏宁、傅先进、李婧、余杰、李莎莎等师弟、师妹对我的支持。感谢刘万伟、李仁见、陈杰等师兄、师弟帮我仔细阅读本论文，并给出了很多细致的修改意见。

感谢多年朝夕相处的室友周倜、晋刚、王睿伯对我的关心和帮助。感谢吴国福、赵常智、张伟、刘峰、董孟高、唐锦涛、肖枫涛、吴俊杰、朱俊、宋伟等同学，作为真挚的朋友，他们给予了我大量的关心和帮助。

感谢汪长江、吉炳安、郑光辉、鲍资富、王新国等学员队领导的关怀。感谢学院教务办潘晓辉等老师，研究生院培养处黄楠、侯艺华等老师，感谢他们在为我办理赴法手续所付出的辛勤劳动以及对我的关心。

感谢我在法国期间巴黎高师“Abstract interpretation and semantics”研究组的 Julien Bertrane, Bruno Blanchet, David Cadé, Ferdinanda Camporesi, Radhia Cousot, Jérôme Feret, Pietro Ferrara, Vincent Laviron, Jérémy Leconte, Laurent Mauborgne, David Monniaux, Miriam Paiola, Xavier Rival, Élodie-Jane Sims, Matteo Zanioli 等人对我的大力支持和帮助。

感谢我的母亲和妹妹对我的支持和关爱。感谢我的舅舅、舅母，他们对我的关心和期望是我前进的动力。感谢我的女友魏登萍，她给予了我深深的理解、支持、鼓励和照顾，是我精神上的坚实后盾。在此特将此文献给他们。

参考文献

[1] 刘克, 单志广, 王戟, 何积丰, 张兆田, 秦玉文. “可信软件基础研究”重大研究计划综述. 中国科学基金. 2008, 22(3):145–151.

[2] The Economist. Software that makes software better. The Economist Technology Quarterly. 8th March 2008:20–21.

[3] S. McConnell. Code Complete. Microsoft Press, 1993.

[4] NIST. Software errors costs U.S. economy $59.5 billion annually. Tech. Rep. NIST 2002–10, National Institute of Standards and Technology, June 2002.

[5] R. Skeel. Roundoff error and the Patriot missile. SIAM News. July 1992, 25(4):11.

[6] J.L. Lions et al. Ariane 5 flight 501 failure report by the inquiry board. Tech. rep., European Space Agency, Paris, France, 1996.

[7] 梅宏, 王千祥, 张路, 王戟. 软件分析技术进展. 计算机学报. 2009, 32(9):1697–1710.

[8] R.W. Floyd. Assigning Meaning to Programs. Proc. of Symposium on Applied Mathematics. American Mathematical Society, 1967, vol. 19, 19–32.

[9] C. A. R. Hoare. An Axiomatic Basis for Computer Programming. Communications of the ACM. 1969, 12(10):576–580.

[10] P. Naur. Proof of algorithms by general snapshots. BIT. 1966, 6(4).

[11] R. Gerth. Model Checking if Your Life Depends on It a View from Intel’s Trenches. SPIN’01. Springer, 2001, vol. 2057 of LNCS, 15.

[12] T. Ball, S. K. Rajamani. The SLAM project: debugging system software via static analysis. ACM POPL’02. ACM Press, 2002, 1–3.

[13] T. Ball, B. Cook, V. Levin, S. K. Rajamani. SLAM and Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft. IFM’04. Springer, 2004, vol. 2999 of LNCS, 1–20.

[14] B. Cook, A. Podelski, A. Rybalchenko. Terminator: Beyond Safety. CAV’06. Springer, 2006, vol. 4144 of LNCS, 415–418.

[15] T. Hoare, J. Misra. Verified Software: Theories, Tools, Experiments Vision of a Grand Challenge Project. Verified Software: Theories, Tools, Experiments (VSTTE’05). Springer, 2008, vol. 4171 of LNCS, 1–18.

[16] C. B. Jones, P. W. O’Hearn, J. Woodcock. Verified Software: A Grand Challenge. IEEE Computer. 2006, 39(4):93–95.

[17] D. Jackson. Dependable Software by Design. Scientific American. June, 294(6).

[18] G. Stix. Send in the Terminator. Scientific American. December 2006, 294(12).

[19] H.G. Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. Transactions of the American Mathematical Society. 1953, 74(2):358–366.

[20] P. Cousot, R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. ACM POPL'77. ACM Press, New York, 1977, 238–252.

[21] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, X. Rival. A Static Analyzer for Large Safety-Critical Software. ACM PLDI'03). ACM Press, 2003, 196–207.

[22] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, X. Rival. The ASTRÉE Analyzer. ESOP'05. Springer, 2005, vol. 3444 of LNCS, 21–30.

[23] The MathWorks. Polyspace Products for Embedded Software Verification. `http://www.mathworks.com/products/polyspace/`.

[24] E. Goubault, M. Martel, S. Putot. Asserting the Precision of Floating-Point Computations: A Simple Abstract Interpreter. ESOP'02. Springer, 2002, vol. 2305 of LNCS, 209–212.

[25] D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, F. Védrine. Towards an Industrial Use of FLUCTUAT on Safety-Critical Avionics Software. FMICS'09. Springer, 2009, vol. 5825 of LNCS, 53–69.

[26] O. Bouissou, E. Conquet, P. Cousot, R. Cousot, J. Feret, K. Ghorbal, E. Goubault, D. Lesens, L. Mauborgne, A. Miné, S. Putot, X. Rival, M. Turin. Space Software Validation using Abstract Interpretation. DASIA'09. ESA, 2009, vol. SP-669, 1–7.

[27] D. Delmas, J. Souyris. Astrée: From Research to Industry. SAS'07. Springer, 2007, vol. 4634 of LNCS, 437–451.

[28] J. Souyris, D. Delmas. Experimental Assessment of Astrée on Safety-Critical Avionics Software. SAFECOMP'07. Springer, 2007, vol. 4680 of LNCS, 479–490.

[29] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, X. Rival. Why does Astrée scale up? Formal Methods in System Design. 2010, 35(3):229–264.

[30] P. Cousot. The Verification Grand Challenge and Abstract Interpretation. Verified Software: Theories, Tools, Experiments (VSTTE'05). Springer, 2008, vol. 4171 of LNCS, 189–201.

[31] D. Wagner, J. S. Foster, E. A. Brewer, A. Aiken. A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities. NDSS'00. The Internet Society, 2000.

[32] D. Brumley, T. Chiueh, R. Johnson, H. Lin, D. Song. RICH: Automatically Protecting Against Integer-Based Vulnerabilities. NDSS'07. The Internet Society, 2007.

[33] N. Dor, M. Rodeh, S. Sagiv. CSSV: towards a realistic tool for statically detecting all buffer overflows in C. PLDI'03. ACM Press, 2003, 155–167.

[34] V. Ganapathy, S. Jha, D. Chandler, D. Melski, D. Vitek. Buffer overrun detection using linear programming and static analysis. CCS'03. ACM Press, 2003.

[35] R. Rugina, M. C. Rinard. Symbolic bounds analysis of pointers, array indices, and accessed memory regions. ACM Transactions on Programming Languages and Systems (TOPLAS). 2005, 27(2):185–235.

[36] X. Allamigeon, W. Godard, C. Hymans. Static Analysis of String Manipulations in Critical Embedded C Programs. SAS'06. Springer, 2006, vol. 4134 of LNCS, 35–51.

[37] G. Kildall. A Unified Approach to Global Program Optimization. ACM POPL'73. ACM Press, 1973, 194–206.

[38] P. Cousot, R. Cousot. Static determination of dynamic properties of programs. Proc. of the 2nd International Symposium on Programming. Dunod, Paris, 1976, 106–130.

[39] M. Karr. Affine Relationships Among Variables of a Program. Acta Inf. 1976, 6:133–151.

[40] P. Cousot, N. Halbwachs. Automatic discovery of linear restraints among variables of a program. ACM POPL'78. ACM Press, New York, 1978, 84–96.

[41] A. Miné. The Octagon Abstract Domain. Higher-Order and Symbolic Computation. 2006, 19(1):31–100.

[42] A. Miné. Weakly Relational Numerical Abstract Domains. Ph.D. thesis, École Polytechnique, Palaiseau, France, December 2004.

[43] S. Sankaranarayanan. Mathematical Analysis of Programs. Ph.D. thesis, Stanford University, Stanford, CA, USA, September 2005.

[44] D. Gopan. Numeric program analysis techniques with applications to array analysis and library summarization. Ph.D. thesis, University of Wisconsin, Madison, WI, USA, August 2007.

[45] M. Colón, H. Sipma. Synthesis of Linear Ranking Functions. TACAS'01. Springer, 2001, vol. 2031 of LNCS, 67–81.

[46] M. Colón, H. Sipma. Practical Methods for Proving Program Termination. CAV'02. Springer, 2002, vol. 2404 of LNCS, 442–454.

[47] Y. Chen, B. Xia, L. Yang, N. Zhan, C. Zhou. Discovering Non-linear Ranking

Functions by Solving Semi-algebraic Systems. ICTAC'07. Springer, 2007, vol. 4711 of LNCS, 34–49.

[48] J. Berdine, A. Chawdhary, B. Cook, D. Distefano, P. W. O'Hearn. Variance analyses from invariance analyses. 34th ACM Symposium on Principles of Programming Languages (POPL'07). ACM Press, New York, 2007, 211–224.

[49] A. Chawdhary, B. Cook, S. Gulwani, M. Sagiv, H. Yang. Ranking Abstractions. the 17th European Symposium on Programming (ESOP'08). Springer-Verlag, 2008, vol. 4960 of LNCS, 148–162.

[50] B. S. Gulavani, S. Gulwani. A Numerical Abstract Domain based on Expression Abstraction and Max Operator with Application in Timing Analysis. CAV'08. Springer-Verlag, 2008, vol. 5123 of LNCS, 370–384.

[51] S. Gulwani, S. Jain, E. Koskinen. Control-flow refinement and progress invariants for bound analysis. ACM PLDI'09. ACM Press, 2009, 375–385.

[52] S. Gulwani, T. Lev-Ami, M. Sagiv. A combination framework for tracking partition sizes. ACM POPL'09. ACM Press, 2009, 239–251.

[53] S. Gulwani, K. K. Mehra, T. M. Chilimbi. SPEED: precise and efficient static estimation of program computational complexity. ACM POPL'09. ACM Press, 2009, 127–139.

[54] S. Gulwani. SPEED: Symbolic Complexity Bound Analysis. CAV'09. Springer, 2009, vol. 5643 of LNCS, 51–62.

[55] R. Rugina. Quantitative Shape Analysis. SAS'04. Springer, 2004, vol. 3148 of LNCS, 228–245.

[56] S. Magill, J. Berdine, E. M. Clarke, B. Cook. Arithmetic Strengthening for Shape Analysis. SAS'07. Springer, 2007, vol. 4634 of LNCS, 419–436.

[57] S. Magill, M. Tsai, P. Lee, Y. Tsay. Automatic numeric abstractions for heap-manipulating programs. ACM POPL'10. ACM Press, 2010, 211–222.

[58] B.-Y. E. Chang, X. Rival. Relational inductive shape analysis. ACM POPL'08. ACM Press, 2008.

[59] V. D'Silva, D. Kroening, G. Weissenbacher. A Survey of Automated Techniques for Formal Software Verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). 2008, 27(7):1165–1178.

[60] R. Jhala, R. Majumdar. Software model checking. ACM Computing Surveys. 2009, 41(4).

[61] David A. Schmidt. Data Flow Analysis is Model Checking of Abstract Interpreta-

tions. ACM POPL'98. ACM Press, 1998, 38–48.

[62] D. A. Schmidt, B. Steffen. Program Analysis *s* Model Checking of Abstract Interpretations. SAS'98. Springer, 1998, vol. 1503 of LNCS, 351–380.

[63] A. Gurfinkel, S. Chaki. Combining Predicate and Numeric Abstraction for Software Model Checking. FMCAD'08. IEEE, 2008, 1–9.

[64] S. Chaki, A. Gurfinkel, O. Strichman. Decision diagrams for linear arithmetic. FMCAD'09. IEEE, 2009, 53–60.

[65] H. Jain, F. Ivancic, A. Gupta, I. Shlyakhter, C. Wang. Using Statically Computed Invariants Inside the Predicate Abstraction and Refinement Loop. CAV'06. Springer, 2006, vol. 4144 of LNCS, 137–151.

[66] G. Goth. Exploring new frontiers. Communications of the ACM. 2009, 52(11):21–23.

[67] Dr. Dobb's Journal. Computing Luminaries Receive NSF Grant To Develop Modeling Tools for Complex Systems. August 19, 2009.

[68] A. Miné. A New Numerical Abstract Domain Based on Difference-Bound Matrices. Proc. of the 2d Symp. on Programs as Data Objects (PADO II). Springer, 2001, vol. 2053 of LNCS, 155–172.

[69] K. G. Larsen, F. Larsson, P. Pettersson, W. Yi. Efficient verification of real-time systems: compact data structure and state-space reduction. IEEE Real-Time Systems Symposium. IEEE Computer Society, 1997, 14–24.

[70] S. Yovine. Model-checking timed automata. Lectures on Embedded Systems. The Internet Society, 1998, vol. 1494 of LNCS, 25–41.

[71] K. G. Larsen, P. Pettersson, Y. Wang. UPPAAL in a Nutshell. Int J Softw Tools Technol Transf (STTT). 1997, 1(1-2):134–152.

[72] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, S. Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. CAV'98. Springer, 1998, vol. 1427 of LNCS, 546–550.

[73] T. A. Henzinger, P.-H. Ho, H. Wong-Toi. HYTECH: A Model Checker for Hybrid Systems. Int J Softw Tools Technol Transf (STTT). 1997, 1(1-2):110–122.

[74] N. Halbwachs, Y.-E. Proy, P. Roumanoff. Verification of Real-Time Systems using Linear Relation Analysis. Formal Methods in System Design. 1997, 11(2):157–185.

[75] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. Int J Softw Tools Technol Transf (STTT). 2008, 10(3):263–279.

[76] O. Bournez, O. Maler, A. Pnueli. Orthogonal Polyhedra: Representation and Com-

putation. HSCC'99. Springer, 1999, vol. 1569 of LNCS, 46–60.

[77] E. Asarin, T. Dang, O. Maler. The d/dt Tool for Verification of Hybrid Systems. CAV'02. Springer, 2002, vol. 2404 of LNCS, 365–370.

[78] A. Girard. Reachability of Uncertain Linear Systems Using Zonotopes. HSCC. Springer, 2005, vol. 3414 of LNCS, 291–305.

[79] S. Sankaranarayanan, T. Dang, F. Ivancic. Symbolic Model Checking of Hybrid Systems Using Template Polyhedra. TACAS'08. Springer, 2008, vol. 4963 of LNCS, 188–202.

[80] P. Cousot, R. Cousot. Systematic design of program analysis frameworks. ACM POPL'79. ACM Press, New York, 1979, 269–282.

[81] GNU Multiple Precision arithmetic library. `http://gmplib.org/`.

[82] A. Simon, A. King. Exploiting Sparsity in Polyhedral Analysis. SAS'05. Springer Verlag, 2005, vol. 3672 of LNCS, 336–351.

[83] D. N. Que. Robust and Generic Abstract Domain for Static Program Analysis: the Polyhedral Case. Tech. rep., École des Mines de Paris, July 2006.

[84] A. Miné. Relational Abstract Domains for the Detection of Floating-Point Run-Time Errors. ESOP'04. Springer, 2004, vol. 2986 of LNCS, 3–17.

[85] A. Miné. Symbolic Methods to Enhance the Precision of Numerical Abstract Domains. VMCAI'06. Springer, 2006, vol. 3855 of LNCS, 348–363.

[86] APRON numerical abstract domain library. Http://apron.cri.ensmp.fr/library/.

[87] G. Lalire, M. Argoud, B. Jeannet. Interproc. `http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/interproc/`.

[88] R. Bagnara, P. M. Hill, E. Zaffanella. Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems. Theoretical Computer Science. 2009, 410(46):4672–4691.

[89] R. Clarisó, J. Cortadellab. The octahedron abstract domain. Science of Computer Programming. 2007, 64(1):115–139.

[90] A. Simon, A. King, J. M. Howe. Two Variables per Linear Inequality as an Abstract Domain. LOPSTR'03. Springer, 2003, vol. 2664 of LNCS, 71–89.

[91] S. Sankaranarayanan, H. Sipma, Z. Manna. Scalable Analysis of Linear Systems using Mathematical Programming. VMCAI'05. Springer Verlag, 2005, vol. 3385 of LNCS, 25–41.

[92] S. Sankaranarayanan, F. Ivancic, A. Gupta. Program Analysis Using Symbolic Ranges. SAS'07. Springer, 2007, vol. 4634 of LNCS, 366–383.

[93] F. Logozzo, M. Fähndrich. Pentagons: a weakly relational abstract domain for the efficient validation of array accesses. ACM SAC'08. ACM Press, 2008, 184–188.

[94] V. Laviron, F. Logozzo. SubPolyhedra: A (more) scalable approach to infer linear inequalities. VMCAI'09. Springer Verlag, 2009, vol. 5403 of LNCS, 229–244.

[95] J. Feret. Static Analysis of Digital Filters. ESOP'04. No. 2986 in LNCS, Springer-Verlag, 2004.

[96] J. Feret. The Arithmetic-Geometric Progression Abstract Domain. VMCAI'05. No. 3385 in LNCS, Springer-Verlag, 2005, 42–58.

[97] P. Cousot, R. Cousot. Higher-Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection and PER Analysis of Functional Languages). ICCL'94. IEEE Computer Society Press, 1994, 95–112.

[98] R. Giacobazzi, F. Ranzato. Optimal domains for disjunctive abstract interpretation. Sci Comput Program. 1998, 32(1-3):177–210.

[99] R. Bagnara, P. M. Hill, E. Zaffanella. Widening operators for powerset domains. VMCAI'04. Springer Verlag, 2004, vol. 2937 of LNCS, 135–148.

[100] X. Rival, L. Mauborgne. The Trace Partitioning Abstract Domain. ACM Transactions on Programming Languages and Systems (TOPLAS). 2007, 29(5).

[101] S. Sankaranarayanan, F. Ivancic, I. Shlyakhter, A. Gupta. Static Analysis in Disjunctive Numerical Domains. SAS'06. Springer, 2006, vol. 4134 of LNCS, 3–17.

[102] B. Jeannet. Dynamic Partitioning in Linear Relation Analysis: Application to the Verification of Reactive Systems. Formal Methods in System Design. 2003, 23(1):5–37.

[103] A. Simon. Splitting the Control Flow with Boolean Flags. SAS'08. Springer, 2008, vol. 5079 of LNCS, 315–331.

[104] P. Granger. Static analysis of arithmetical congruences. International Journal of Computer Mathematics. 1989, 30:165–199.

[105] X. Allamigeon, S. Gaubert, E. Goubault. Inferring Min and Max Invariants Using Max-plus Polyhedra. SAS'08. Springer Verlag, 2008, vol. 5079 of LNCS, 189–204.

[106] J. ZHANG, Y. FENG. Obtaining exact value by approximate computations. Science in China Series A: Mathematics. 2007, 50(9):1361–1368.

[107] A. Neumaier, O. Shcherbina. Safe bounds in linear and mixed-integer linear programming. Math Program. 2004, 99(2):283–296.

[108] C. Jansson. Rigorous Lower and Upper Bounds in Linear Programming. SIAM

Journal on Optimization. 2004, 14(3):914–935.

[109] C. Jansson, D. Chaykin, C. Keil. Rigorous Error Bounds for the Optimal Value in Semidefinite Programming. SIAM Journal on Numerical Analysis. 2007, 46(1):180–200.

[110] C. Jansson. On Verified Numerical Computations in Convex Programming. Japan J Indust Appl Math. 2009, 26:337–363.

[111] D. Monniaux. On using floating-point computations to help an exact linear arithmetic decision procedure. CAV'09. Springer Verlag, 2009, vol. 5643 of LNCS, 570–583.

[112] David Goldberg. What Every Computer Scientist Should Know About Floating-Point Arithmetic. ACM Computing Surveys. 1991, 23(1):5–48.

[113] D. Monniaux. The pitfalls of verifying floating-point computations. ACM Transactions on Programming Languages and Systems (TOPLAS). 2008, 30(3):1–41.

[114] V.R. Pratt. Anatomy of the Pentium Bug. TAPSOFT'95. Springer, 1995, vol. 915 of LNCS, 97–107.

[115] J. Harrison. Floating-Point Verification Using Theorem Proving. SFM'06. Springer, 2006, vol. 3965 of LNCS, 211–242.

[116] J. Harrison. Floating-Point Verification. Journal of Universal Computer Science. 2007, 13(5):629–638.

[117] E. Goubault. Static Analyses of Floating-Point Operations. SAS'01. Springer-Verlag, 2001, vol. 2126 of LNCS, 234–259.

[118] S. Boldo, J.-C. Filliâtre. Formal Verification of Floating-Point Programs. 18th IEEE Symposium on Computer Arithmetic. IEEE Computer Society, 2007, 187–194.

[119] S. Boldo, J.-C. Filliâtre, G. Melquiond. Combining Coq and Gappa for Certifying Floating-Point Programs. 16th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning. Springer, 2009, vol. 5625 of LNAI, 59–74.

[120] R. Moore. Interval Analysis. Prentice-Hall, 1966.

[121] B. Hayes. A lucid interval. American Scientist. 2003, 91(6):484–488.

[122] E. R. Hansen. Global Optimization Using Interval Analysis. New York, NY: Marcel Dekker, 1992.

[123] L. Jaulin, M. Kieffer, O. Didrit, È. Walter. Applied Interval Analysis. London: Springer-Verlag, 2001.

[124] R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. Euromath Bulletin. 1994, 1(2).

[125] A. Neumaier. Interval Methods for Systems of Equations. Cambridge University Press, 1990.

[126] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, K. Zimmermann. Linear optimization problems with inexact data. New York: Springer, 2006.

[127] W. Oettli, W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. Numer Math. 1964, 6:405–409.

[128] J. Rohn. A Handbook of Results on Interval Linear Problems. Tech. rep., Czech Academy of Sciences, Prague, Czech Republic, April 2005.

[129] J. Rohn. Solvability of systems of interval linear equations and inequalities. Linear Optimization Problems with Inexact Data. Springer, 2006, 35–77.

[130] C. Jansson. Calculation of exact bounds for the solution set of linear interval systems. Linear Algebra and Its Applications. 1997, 251:321–340.

[131] J.W. Chineck, K. Ramadan. Linear Programming with interval coefficients. Journal of the Operational Research Society. 2000, 51(2):209–220.

[132] P. Cousot. Abstract interpretation. MIT course 16.399, Feb.–May 2005. `http://web.mit.edu/16.399/www/`.

[133] A. Tarski. A lattice-theoretical fixpoint theorem and its application. Pacific Journal of Mathematics. 1955, 5:285–309.

[134] P. Cousot, R. Cousot. Constructive Versions of Tarski's Fixed Point Theorems. Pacific Journal of Mathematics. 1979, 81(1):43–57.

[135] S.C. Kleene. Introduction to metamathematics. Amsterdam: North-Holland.

[136] Z. Su, D. Wagner. A class of polynomially solvable range constraints for interval analysis without widenings. Theor Comput Sci. 2005, 345(1):122–138.

[137] T. Gawlitza, J. Leroux, J. Reineke, H. Seidl, G. Sutre, R. Wilhelm. Polynomial Precise Interval Analysis Revisited. Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday. Springer, 2009, vol. 5760 of LNCS, 422–437.

[138] A. Costan, S.Gaubert, E.Goubault, M.Martel, S.Putot. A Policy iteration algorithm for computing fixed points in static analysis of programs. CAV'05. Springer Verlag, 2005, vol. 3576 of LNCS, 462–475.

[139] S. Gaubert, E. Goubault, A. Taly, S. Zennou. Static Analysis by Policy Iteration on Relational Domains. ESOP'07. Springer, 2007, vol. 4421 of LNCS, 237–252.

[140] T. Gawlitza, H. Seidl. Precise Relational Invariants Through Strategy Iteration.

CSL'07. Springer, 2007, vol. 4646 of LNCS, 23–40.

[141] P. Cousot, R. Cousot. Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. PLILP'92. Springer-Verlag, 1992, vol. 631 of LNCS, 269–295.

[142] F. Bourdoncle. Efficient Chaotic Iteration Strategies with Widenings. FMPA'93. Springer Verlag, 1993, vol. 735 of LNCS, 128–141.

[143] F. Masdupuy. Array abstractions using semantic analysis of trapezoid congruences. ICS'92. ACM Press, 1992, 226–235.

[144] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, E. Zaffanella. Grids: A Domain for Analyzing the Distribution of Numerical Values. LOPSTR'06. Springer-Verlag, 2007, vol. 4407 of LNCS, 219–235.

[145] 陈立前, 王戟, 侯苏宁. 单变量区间线性不等式抽象域. 计算机学报. 2010, 33(3):427–439.

[146] J.M. Howe, A. King. Logahedra: a New Weakly Relational Domain. ATVA'09. Springer-Verlag, 2009, vol. 5799 of LNCS, 306–320.

[147] L. Chen, A. Miné, J. Wang, P. Cousot. An Abstract Domain to Discover Interval Linear Equalities. VMCAI'10. Springer, 2010, vol. 5944 of LNCS, 112–128.

[148] P. Granger. Static analysis of linear congruence equalities among variables of a program. TAPSOFT'91. Springer-Verlag, 1991, vol. 493 of LNCS, 169–192.

[149] L. Chen, A. Miné, P. Cousot. A Sound Floating-Point Polyhedra Abstract Domain. APLAS'08. Springer Verlag, 2008, vol. 5356 of LNCS, 3–18.

[150] L. Chen, A. Miné, J. Wang, P. Cousot. Interval Polyhedra: An Abstract Domain to Infer Interval Linear Relationships. SAS'09. Springer Verlag, 2009, vol. 5673 of LNCS, 309–325.

[151] C. Bartzis, T. Bultan. Efficient Symbolic Representations for Arithmetic Constraints in Verification. International Journal of Foundations of Computer Science. 2003, 14(4):605–624.

[152] C. Bartzis, T. Bultan. Widening Arithmetic Automata. CAV'04. Springer, 2004, vol. 3114 of LNCS, 321–333.

[153] J. Leroux. Convex Hull of Arithmetic Automata. SAS'08. Springer, 2008, vol. 5079 of LNCS, 47–61.

[154] R. Bagnara, P. M. Hill, E. Zaffanella. Not necessarily closed convex polyhedra and the double description method. Formal Aspects of Computing. 2005, 17(2):222–257.

[155] IEEE Computer Society. IEEE standard for binary floating point arithmetic. Tech.

rep., ANSI/IEEE Std 745-1985, 1985.

[156] P. Cousot. Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation. Electronic Notes in Theoretical Computer Science. 1997, 6.

[157] V. Loechner. *PolyLib*: A Library for Manipulating Parameterized Polyhedra. Available at `http://icps.u-strasbg.fr/polylib/`, mar 1999.

[158] B. Jeannet, A. Miné. Apron: A Library of Numerical Abstract Domains for Static Analysis. CAV'09. Springer, 2009, vol. 5643 of LNCS, 661–667.

[159] R. Bagnara, P. M. Hill, E. Zaffanella. The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. Science of Computer Programming. 2008, 72(1–2):3–21.

[160] H. LeVerge. A note on Chernikova's algorithm. Tech. Rep. 635, IRISA, France, 1992.

[161] N. Halbwachs. Détermination automatique de relations linéaires vérifiées par les variables d'un programme. Ph.D. thesis, Thèse de 3ème cycle d'informatique, Université scientifique et médicale de Grenoble, Grenoble, France, March 1979.

[162] T. Huynh, C. Lassez, J.-L. Lassez. Practical Issues on the Projection of Polyhedral Sets. Annals of Mathematics and Artificial Intelligence. 1992, 6(4):295–315.

[163] J.-L. Imbert. Fourier's Elimination: Which to Choose? PCPP'93. 1993, 117–129.

[164] S. Sankaranarayanan, M. Colón, H. B. Sipma, Z. Manna. Efficient Strongly Relational Polyhedral Analysis. VMCAI'06. Springer, 2006, vol. 3855 of LNCS, 111–125.

[165] A. Makhorin. The GNU Linear Programming Kit, 2000. Http://www.gnu.org/software/glpk/.

[166] S. Sankaranarayanan, H. B. Sipma, Z. Manna. Constraint-Based Linear-Relations Analysis. SAS'04. Springer, 2004, vol. 3148 of LNCS, 53–68.

[167] M. Müller-Olm, H. Seidl. Precise interprocedural analysis through linear algebra. ACM POPL'04. ACM Press, 2004, 330–341.

[168] S. Gulwani, G. Necula. Discovering Affine Equalities Using Random Interpretation. ACM POPL'03. ACM Press, 2003, 74–84.

[169] M. Müller-Olm, H. Seidl. A Note on Karr's Algorithm. ICALP'04. Springer, 2004, vol. 3142 of LNCS, 1016–1028.

[170] R.W. Cottle, J.-S. Pang, R.E. Stone. The Linear Complementarity Problem. New York: Academic Press, 1992.

[171] B. De Schutter, B. De Moor. The extended linear complementarity problem. Math-

ematical Programming. 1995, 71(3):289–325.

[172] W. P. M. H. Heemels, J. M. Schumacher, S. Weiland. Linear Complementarity Systems. SIAM Journal on Applied Mathematics. 2000, 60(4):1234–1269.

[173] B. De Schutter, B. De Moor. The extended linear complementarity problem and its applications in the analysis and control of discrete event systems and hybrid systems. SISCTA'97. 1997, 394–398.

[174] B. De Schutter. The extended linear complementarity problem and its applications in analysis and control of discrete-event systems. Pareto Optimality, Game Theory and Equilibria. Springer, 2008, vol. 17 of Springer Optimization and Its Applications, 541–570.

[175] J. Rohn. A theorem of the alternatives for the equation $Ax + B|x| = b$. Linear and Multilinear Algebra. 2004, 52(6):421–426.

[176] L. Mangasarian, R.R. Meyer. Absolute Value Equations. Linear Algebra and Its Applications. 2006, 419:359–367.

[177] L. Mangasarian. Absolute Value Programming. Computational Optimization and Applications. 2007, 36(1):43–53.

[178] J. Rohn. An Algorithm for Solving the Absolute Value Equation. Electronic Journal of Linear Algebra. 2009, 18:589–599.

[179] O.A. Prokopyev. On equivalent reformulations for absolute vlaue equations. Computational Optimization and Applications. 2009, 44(3):363–372.

[180] M. Anitescu, G. Lesaja, F.A. Potra. Equivaence between different formulations of the linear complementarity promblem. Optimization Methods and Software. 1997, 7(3):265–290.

[181] B.C. Eaves, C.E. Lemke. Equivalence of LCP and PLS. MATHEMATICS OF OPERATIONS RESEARCH. 1981, 6(4):475–484.

[182] O. L. Mangasarian, J. S. Pang. The Extended Linear Complementarity Problem. SIAM J Matrix Anal Appl. 1995, 16(2):359–368.

[183] M.S. Gowda. On the extended linear complementarity problem. Math Program. 1996, 72(1):33–50.

[184] T. S. Motzkin, H. Raiffa, G. L. Thompson, R. M. Thrall. The double description method. Contributions to the Theory of Games. Princeton University Press, 1953, vol. 17 of Annals of Mathematics Studies, 51–73.

[185] A. Schrijver. Theory of linear and integer programming. John Wiley & Sons, Inc., 1986.

[186] K. Fukuda, A. Prodon. Double Description Method Revisited. Combinatorics and Computer Science. Springer, 1996, vol. 1120 of LNCS, 91–111.

[187] N.V. Chernikova. Algorithm for Finding a General Formula for the Non-Negative Solutions of a System of Linear equations. USSR Computational Mathematics and Mathematical Physics. 1964, 4(4):151–158.

[188] N.V. Chernikova. Algorithm for Finding a General Formula for the Non-Negative Solutions of a System of Linear Inequalities. USSR Computational Mathematics and Mathematical Physics. 1965, 5(2):228–233.

[189] N.V. Chernikova. Algorithm for discovering the set of all the solutions of a linear programming problem. USSR Computational Mathematics and Mathematical Physics. 1968, 8(6):282–293.

[190] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo. The maximum clique problem. Handbook of Combinatorial Optimization (Supplement Volume A). Kluwer Academic Publishers, 1999, 1–74.

[191] D. F. Shanno, R. L. Weil. 'Linear' Programming with Absolute-Value Functionals. Operations Research. 1971, 19(1):120–124.

[192] G. B. Dantzig, M. N. Thapa. Linear programming 1: introduction. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[193] B. S. Gulavani, T. A. Henzinger, Y. Kannan, A. V. Nori, S. K. Rajamani. SYNERGY: a new algorithm for property checking. SIGSOFT FSE'06. ACM Press, 2006, 117–127.

[194] T. A. Henzinger, R. Jhala, R. Majumdar, G. Sutre. Lazy abstraction. ACM POPL'02. ACM Press, 2002, 58–70.

作者在学期间取得的学术成果

[1] Liqian Chen, Antoine Miné, Patrick Cousot. A Sound Floating-Point Polyhedra Abstract Domain. In Proc. of the 6th Asian Symposium on Programming Languages and Systems (APLAS 2008), Springer, 2008, Bangalore, India, vol. 5356 of LNCS, 3–18.

[2] Liqian Chen, Antoine Miné, Ji Wang, Patrick Cousot. Interval Polyhedra: An Abstract Domain to Infer Interval Linear Relationships. In Proc. of the 16th International Static Analysis Symposium (SAS 2009), Springer, 2009, Los Angeles, CA, USA, vol. 5673 of LNCS, 309–325.

[3] Liqian Chen, Antoine Miné, Ji Wang, Patrick Cousot. An Abstract Domain to Discover Interval Linear Equalities. In Proc. of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2010), Springer, 2010, Madrid, Spain, vol. 5944 of LNCS, 112–128.

[4] 陈立前, 王戟, 侯苏宁. 单变量区间线性不等式抽象域. 计算机学报, 2010, 33(3): 427–439.

[5] 陈立前, 王戟, 刘万伟. 基于约束的多面体抽象域的弱结合. 软件学报（已录用）.

[6] 陈立前, 王戟, 陈火旺. 基于 TLA+ 的 BRP 协议规约及验证. 计算机工程与科学, 2006, 28(1): 12–15.

[7] 朱俊, 陈立前, 杨学军. 基于 RTEMS 的 ATM 网卡设备驱动程序以及 CIPOA 的研究与实现. 计算机应用, 2006, 26(9): 2211–2214.

[8] 朱俊, 陈立前, 杨学军. 一种星载嵌入式计算机系统的 ATM 网络适配器及其设备驱动程序的研究与实现. 第 17 届中国过程控制会议（CPCC 2006）, 无锡, 2006.

[9] 李仁见, 陈立前, 王戟. IP 及其路由技术在基于 ATM 网络的星载计算机中的研究与实现. 计算机工程, 2007, 33(14): 110–112.

[10] 侯苏宁, 陈立前, 王昭飞, 王戟. 一个面向 C 和 Fortran 数值程序的静态分析工具. 计算机工程与科学（已录用）.