软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(3):1307−1320 [doi: 10.13328/j.cnki.jos.006817]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

单分支线性约束循环程序的终止性分析*

李 轶[1], 唐 桐[1,2]

[1](中国科学院 重庆绿色智能技术研究院 自动推理与认知中心, 重庆 400714)

[2](中国科学院大学, 北京 100049)

通信作者: 李轶, E-mail: zm_liyi@163.com

摘　要: 秩函数法是循环终止性分析的主要方法, 秩函数的存在表明了循环程序是可终止的. 针对单分支线性约束循环程序, 提出一种方法对此类循环的终止性进行分析. 基于增函数法向空间的计算, 该方法将原程序空间上的秩函数计算问题归结为其子空间上的秩函数计算问题. 实验结果表明, 该方法能有效验证现有文献中大部分循环程序的终止性.

关键词: 循环程序; 线性秩函数; 增函数; 终止性; 多阶段秩函数

中图法分类号: TP311

中文引用格式: 李轶, 唐桐. 单分支线性约束循环程序的终止性分析. 软件学报, 2024, 35(3): 1307–1320. http://www.jos.org.cn/1000-9825/6817.htm

英文引用格式: Li Y, Tang T. Termination Analysis of Single-path Linear Constraint Loops. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1307–1320 (in Chinese). http://www.jos.org.cn/1000-9825/6817.htm

Termination Analysis of Single-path Linear Constraint Loops

LI Yi[1], TANG Tong[1,2]

[1](Automated Reasoning and Cognition Center, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: The ranking function method is the main method for the termination analysis of loops, and it indicates that loop programs can be terminated. In view of single-path linear constraint loop programs, this study presents a method to analyze the termination of the loops. Based on the calculation of the normal space of the increasing function, this method considers the calculation of the ranking function in the original program space as that in the subspace. Experimental results show that the method can effectively verify the termination of most loop programs in the existing literature.

Key words: loop program; linear ranking function; increasing function; termination; multiphase ranking function

循环程序的终止性分析是程序验证的一个重要领域. 程序终止性是指程序在有限时间内会自动停止执行的属性. 在软件系统中, 循环程序普遍存在. 但即便是最简单的循环程序, 也可能会出现错误. 所以, 保证循环程序的可终止性是软件系统能可靠运行的必要条件. 尽管程序终止性问题早已经被证明是一个不可判定问题[1,2], 但人们发现, 具有某些特殊结构的循环程序, 其终止性是可判定的. 如, 2004 年, 利用赋值矩阵的 Jordan 标准型计算, Tiwari 将一类线性循环程序的终止性问题与赋值矩阵正实特征值对应的特征向量联系起来, 证明了该类循环在实数域上的终止性问题是可判定的[3]. 文献 [4] 进一步研究了这类循环的不可终止点集的构造问题. Xia 等人[5]证明了在一定条件下, 带线性赋值且循环条件为多项式不等式的一类循环程序在实域上的终止性是可判定的. Liu 等人[6]研究了循环条件为多项式等式的多分支多项式循环程序的终止性问题, 给出了该类循环程序可终止和不可终止的充分

* 基金项目: 重庆市自然科学基金 (cstc2019jcyj-msxmX0638); 国家自然科学基金 (11771421); 中国科学院“西部之光”人才培养计划
收稿时间: 2022-03-01; 修改时间: 2022-05-01, 2022-07-14; 采用时间: 2022-09-17; jos 在线出版时间: 2023-06-07
CNKI 网络首发时间: 2023-06-08

判准. 此外, Chen 等人[7]、He 等人[8]提出了基于 Büchi 自动机的终止性分析方法, 可将现实中的部分复杂程序的终止性问题归结为具更简单结构的循环程序的终止性问题. Chatterjee 等人[9]利用计算机代数中的正零点定理来研究概率程序的终止性. 针对 Tiwari 提出的线性循环程序[3], Xue 等人[10]分析了在浮点计算下这类循环的不可终止点集的下逼近问题.

目前大部分终止性研究多集中于循环的秩函数计算. 模板秩函数的存在为循环程序的可终止性提供了充分的判准. 由于秩函数的定义相对简洁, 秩函数法已成为当前终止性研究的主流方法. 秩函数法的本质思想是: 通过秩函数将程序变量映射到一个良序集上, 并要求函数值随着程序的运行而单调递减. 由于良序集具有最小元, 故具备秩函数的程序必定在有限时间内结束运行. 尽管程序终止性问题是不可判定的, 但是某些形式的秩函数的存在性问题是可判定的. 比如线性约束循环的线性秩函数. 因此, 给定一个循环程序, 如能找到它的秩函数, 则表明该循环程序是可终止的. 当前秩函数方面的工作多集中于线性约束循环程序的线性秩函数计算. 针对单分支线性约束循环 (SLC), 2001 年, Colón 等人利用多面体理论将线性不等式约束表示为多面体锥, 再将线性秩函数计算问题转化为对多面体锥投影、极锥和交点的计算[11]. 2004 年, Podelski 等人基于 Farkas' Lemma 引理, 提出了一种完备的方法来构造这类循环程序上的线性秩函数[12]. 2012 年, Bagnara 等人研究了 SLC 循环程序的线性秩函数计算的复杂度问题[13]. 2019, 针对 SLC 循环, 文献 [14] 考虑了其线性秩函数的存在性问题, 并建立了多项式时间算法去判定给定的 SLC 循环是否具有线性秩函数, 与现有方法不同, 该方法并不需要计算出具体秩函数. 此外, 一些研究者进一步提出将多个线性秩函数结合起来捕捉程序的终止性行为. 如, Bradley 等人分别在文献 [15–17] 提出了字典序线性秩函数 (LLRFs) 的概念, 扩大了秩函数外在形式. 2013 年, Bagnara 等人提出了 eventual 线性秩函数 (eventual linear ranking functions) 的概念, 进而通过合成 eventual 线性秩函数来证明单分支线性约束循环程序的可终止性[18]. 文献 [19] 对 eventual 线性秩函数的概念进行推广, 提出了 L 深度的 eventual 线性秩函数计算方法. Leike 等人首次提出了多阶段秩函数的概念[20], 在此基础上, Ben-Amram 等人在文献 [21] 中给出了固定阶数 L 下, L 阶多阶段线性秩函数 (MΦRFs) 的多项式复杂度计算方法, 并提出了 L 上界估计的公开问题. 他们在文献 [21] 中证明了, 对给定的一个 SLC 循环, 它存在多阶段秩函数等价于该循环存在嵌套秩函数 (nested ranking functions). 而嵌套秩函数的计算存在多项式时间算法. 因此, 文献 [21] 中均通过计算嵌套秩函数来证明程序的可终止性. 此外, 文献 [22] 证明了在一定条件下, L 上界是可估计的, 部分回答了文献 [21] 中的问题. 相较于线性秩函数, 非线性秩函数生成方面的工作较少. 文献 [23] 基于 CAD (柱形代数分解) 算法给出了一种非线性多项式秩函数的生成方法. 文献 [24,25] 使用半正定规划 SDP 工具来计算非线性多项式秩函数. 文献 [26,27] 将多项式秩函数问题归结为二分类问题, 并利用 SVM 算法来计算多项式嵌套秩函数.

本文主要研究非浮点型 SLC 循环程序在有理数域上的终止性问题. 相对于非浮点型程序, 浮点程序 (即程序输入为浮点数), 将涉及浮点数的误差控制问题, 而浮点程序的终止性问题则是另一更具挑战性的问题. 不同于上述文献中的方法, 本文将原程序空间中的线性秩函数存在性问题归结为一个新程序空间上的线性秩函数存在性问题. 具体地, 我们首先判断给定的 SLC 循环是否具有不动点. 若有, 则循环不终止. 否则, 通过文献 [14] 中的方法去判定该循环对应的程序空间上是否存在线性秩函数 (LRFs). 若其上存在线性秩函数, 则表明该 SLC 循环是终止的. 如若不然, 则基于文献 [18,19] 中的增函数概念计算原程序空间上的部分增函数空间, 并将部分增函数空间所对应的不等式约束加入到原程序空间的定义约束中, 从而将原 SLC 循环上的线性秩函数的存在性问题归结为一个新的 SLC 循环上的线性秩函数的存在性问题.

1 基本概念

本节将介绍 SLC 循环定义、秩函数、增函数、Farkas' Lemma 引理、生成表示相关的概念和定理.

1.1 SLC 循环

在程序分析中, 由于抽象和逼近技术的应用使得 SLC 循环大量出现, 对其终止性分析显得尤为必要[28]. 一般地, 含有 n 个变量的 SLC 循环形式如下:

$$\text{while}\,(Bx \geqslant b)\ \text{do}\ C\begin{pmatrix} x \\ x' \end{pmatrix} \geqslant c \tag{1}$$

其中, $x=(x_1,x_2,\ldots,x_n)^{\mathrm{T}} \in \mathbb{Q}^n$, $x'=(x'_1,x'_2,\ldots,x'_n)^{\mathrm{T}} \in \mathbb{Q}^n$均为 n 维列向量, $B \in \mathbb{Q}^{p\times n}$、 $b \in \mathbb{Q}^p$、 $C=(C_1,C_2) \in \mathbb{Q}^{q\times 2n}$、 $c \in \mathbb{Q}^q$. 约束 $Bx \geqslant b$ 称为循环条件, 约束 $C\begin{pmatrix} x \\ x' \end{pmatrix} \geqslant c$ 称为更新约束. 易见, SLC 循环程序可由一组线性不等式约束所刻画的集合来定义. 为方便, 用 Ω 表示 SLC 循环, 则公式 (1) 中的循环可被等价描述为: $\Omega=\{(x,x') \in \mathbb{Q}^{2n} : Ax+A'x' \geqslant a\}$. $x=(x_1,x_2,\ldots,x_n)^{\mathrm{T}} \in \mathbb{Q}^n$, $x'=(x'_1,x'_2,\ldots,x'_n)^{\mathrm{T}} \in \mathbb{Q}^n$, $A=\begin{pmatrix} B \\ C_1 \end{pmatrix} \in \mathbb{Q}^{s\times n}$, $A'=\begin{pmatrix} 0 \\ C_2 \end{pmatrix} \in \mathbb{Q}^{s\times n}$, $a=\begin{pmatrix} b \\ c \end{pmatrix} \in \mathbb{Q}^s$, $s=p+q$. 为方便, 下文中, 我们也沿用 Ω 去表示循环程序所对应的线性不等式约束, 即: $\Omega \triangleq Ax+A'x' \geqslant a$. 接下来, 我们通过文献 [28] 中的一个例子来阐述 SLC 循环.

$$\text{while}\ \ 4x_1 \geqslant x_2 \wedge x_2 \geqslant 1\ \ \text{do}\ \ \ x_1=\frac{2x_1+1}{5},\ x_2=x_2.$$

该循环可以等写成下面的形式:

$$\text{while}\ 4x_1 \geqslant x_2 \wedge x_2 \geqslant 1\ \ \text{do}\ \ \ x'_1 \leqslant \frac{2x_1+1}{5};\ x'_1 \geqslant \frac{2x_1+1}{5};\ x'_2 \geqslant x_2;\ x'_2 \leqslant x_2 \tag{2}$$

其中, 将 $x_1=(2x_1+1)/5$ 转换为 $\{x'_1 \leqslant (2x_1+1)/5; x'_1 \geqslant (2x_1+1)/5\}$ 的等价形式, 以便后面一般形式的书写. 这里, (x_1,x_2) 表示当前状态, 而 (x'_1,x'_2) 表示下一个状态. 故上述 SLC 循环 Ω 为: $\Omega=\{(x,x') \in \mathbb{Q}^{2\times 2} : 4x_1 \geqslant x_2, x_2 \geqslant 1, 5x'_1 \leqslant 2x_1+1, 5x'_1 \geqslant 2x'_1+1, x'_2 \geqslant x_2, x'_2 \leqslant x_2\}$.

不难看出, 更新约束中的等式约束总可以用不等式约束来刻画. 因为循环体内允许不等式更新约束使 SLC 较等式型赋值具有更丰富的表述能力.

1.2 线性秩函数

定义 1. 给定 SLC 循环程序 Ω. 令 $\rho(x)$ 为 n 元线性函数, 如果 $\rho(x)$ 满足下列条件, 则称 $\rho(x)$ 为循环程序 Ω 上的线性秩函数.

$$(1)\ \forall x,x' : \Omega \Rightarrow \rho(x) \geqslant \rho(x')+1\ ;\ (2)\ \forall x,x' : \Omega \Rightarrow \rho(x) \geqslant 0\ ,$$

其中, $x=(x_1,x_2,\ldots,x_n)^{\mathrm{T}} \in \mathbb{Q}^n, x'=(x'_1,x'_2,\ldots,x'_n)^{\mathrm{T}} \in \mathbb{Q}^n$.

1.3 增函数

定义 2. 给定 SLC 循环程序 Ω, 令 $f(x)$ 为 n 元线性函数. 如果 $f(x)$ 满足条件: $\forall x,x' : \Omega \Rightarrow f(x') \geqslant 1+f(x)$, 则称 $f(x)$ 是循环程序上的增函数.

若 $f(x)$ 是 Ω 上的一个增函数, 则对 $\forall k \in R$ 有 $f(x)-k$ 仍为 Ω 上的增函数. 不失一般性, 对任意给定的常数 k, 集合 $\Omega \cap \{(x,x') \in \mathbb{Q}^{2n} : f(x) \leqslant k\}$ 中的点 x 必然在有限次迭代后跳入到 $\{(x,x') \in \mathbb{Q}^{2n} : f(x) \geqslant k\}$ 中. 这里的 k 即为一个阈值. 因此, 上述增函数的性质在于其一般可以将 Ω 分为两个部分, 即 $\Omega^- = \Omega \cap \{(x,x') \in \mathbb{Q}^{2n} : f(x) < 0\}$ 和 $\Omega^+ = \Omega \cap \{(x,x') \in \mathbb{Q}^{2n} : f(x) \geqslant 0\}$. 且 Ω^- 中的点在经过有限次迭代后会落入 $\{(x,x') \in \mathbb{Q}^{2n} : f(x) \geqslant 0\}$ 中, 而 Ω^+ 中的点不迭代回 $\{(x,x') \in \mathbb{Q}^{2n} : f(x) \leqslant 0\}$ 中. 因此不失一般性, 任给定常数 k, 若 $f(x)$ 是 Ω 上的增函数, 那么对任意的点 $x \in \Omega \cap \{(x,x') \in \mathbb{Q}^{2n} : f(x) \leqslant k\}$, 必有点 x 在迭代有限次后必然跳入到 $\{(x,x') \in \mathbb{Q}^{2n} : f(x) > k\}$ 中. 上述性质将原 SLC 循环的终止性问题等价地归结为新 SLC 循环 Ω^+ 上的终止性问题[18]. 因此, 人们仅需考虑 Ω^+ 上的终止性 (或秩函数计算) 问题.

下面引用文献 [18] 中的例子来具体阐述增函数的应用. 给定一个循环程序 Ω: $\Omega=\{x \geqslant 0, y' \leqslant y-1, x' \leqslant x+y\}$. 用 Farkas' Lemma 引理计算出 Ω 上的一个增函数 $f_0(x,y)=-y$. 根据增函数定义不难看出, 对任意的 k, 有 $f(x,y)=f_0(x,y)-k=-y-k$ 均是 Ω 上的增函数. 选择合适的 k 使得超平面 $f(x,y)=-y-k=0$ 与 Ω 相交. 这样, 超平面 $f(x,y)=0$ 将 Ω 划分为两个部分 Ω^+ 和 Ω^-. 即:

$$\begin{cases} \Omega^+ = \{x \geqslant 0,\ y' \leqslant y-1,\ x' \leqslant x+y,\ -y-k \geqslant 0\} \\ \Omega^- = \{x \geqslant 0,\ y' \leqslant y-1,\ x' \leqslant x+y,\ -y-k < 0\} \end{cases}.$$

显然, $\Omega^+ \subseteq \Omega$ 定义了一个新的 SLC 循环. 一般地, 因 $\Omega^+ \subseteq \Omega$, 即: 划分后的程序空间是原始程序空间的子集,

故更容易找到 Ω^+ 上的秩函数. 同时, 由增函数性质可知: Ω^- 中的点不会迭代回 Ω^+ 中. 因此, 通过引入增函数, 文献 [18] 将原 SLC 循环程序 Ω 的终止性问题归结为新 SLC 循环 Ω^+ 上的终止性问题. 但文献 [18] 中仅将一个增函数加入到 Ω 中对其进行划分. 通常, Ω 上的增函数往往是无穷多个, 不同的增函数加入到 Ω 中将得到不同 Ω^+. 因此, 倘若加入的增函数无法成功划分原始空间 Ω, 那么该操作相当于没有起作用. 更重要的是, 当 Ω^+ 没有线性秩函数时, 文献 [18] 中并没有考虑 Ω^+ 的再次划分问题. 文献 [19] 中推广了文献 [18] 的工作, 即将文献 [18] 中的 1 次划分推广到多次划分. 但与文献 [18] 相同, 文献 [19] 在每次划分中仅仅考虑了用一个增函数 $f(x)$ 来对当前程序空间进行划分. 更重要的是, 程序空间上的增函数有无穷多个. 比如, 若 f 为 Ω 上增函数, 则对任意的 k, 均有 $f-k$ 仍为 Ω 上增函数. 同时, 不同增函数的选取有可能影响到划分后所得空间上是否存在线性秩函数. 因此, 为避免增函数的选取, 本文拟通过计算程序空间上的增函数法向空间来刻画全体增函数集合, 并利用所刻画的增函数集合对当前程序空间进行划分, 同时, 本文采用了文献 [19] 中 L-Depth 概念来对程序空间进行多次划分. 直观上, 这样得到的 Ω^+ 比起仅加入一个增函数约束所得到的 Ω^+ 要更小, 从而使得有较大可能性找到划分 所得 Ω^+ 上的线性秩函数. 具体地, 在实际计算过程中, 由于增函数空间中的元素往往有无穷多个, 故借助增函数的法向量空间为凸多面体这一性质, 我们采取将增函数法向量空间生成元对应的函数所形成的不等式约束加入到原始空间中这一策略来完成对 Ω 的划分.

1.4 Farkas' Lemma 引理

定义实数上的下列线性不等式组所确定的集合为 S:

$$c_{1,1}x_1+\ldots+c_{1,n}x_n+c_{1,n+1}\geqslant 0,\ c_{2,1}x_1+\ldots+c_{2,n}x_n+c_{2,n+1}\geqslant 0,\ \ldots,\ c_{t,1}x_1+\ldots+c_{t,n}x_n+c_{t,n+1}\geqslant 0,$$

其中, $c_{i,j}$ 是对应变量 x_j 的系数, $i,j=1,\ldots,n$. $c_{i,n+1}$ 是常数项, $i=1,\ldots,n$. 其取值范围均是有理数域 $\mathbb{Q}$. 假设上述不等式组有解 (即 S 非空), 则有: $\forall x=(x_1,\ldots,x_n)\in S\Rightarrow(d_1x_1+\ldots+d_nx_n+d_{n+1}\geqslant 0)$ 成立的充分必要条件是:

$$\exists\lambda_1\geqslant 0,\ldots,\lambda_t\geqslant 0,\left(d_{n+1}\geqslant\sum\nolimits_{i=1}^{t}\lambda_i c_{i,n+1}\right)\wedge\bigwedge_{j=1}^{n}\left(d_j=\sum\nolimits_{i=1}^{t}\lambda_i c_{i,j}\right).$$

1.5 生成表示

文献 [14,21] 中的定理表明凸多面体可以用其顶点和射线表示为:

$$\Phi=conv(x_1,\ldots,x_t)+cone(y_1,\ldots,y_m),$$

其中, 顶点 $x_i\in\mathbb{Q}^n, i=1,\ldots,t$. 射线 $y_j\in\mathbb{Q}^n, j=1,\ldots,m$. 对任意点 $z\in\Phi$, z 具有生成表示: $z=\sum_{i=1}^{t}r_ix_i+\sum_{j=1}^{m}u_jy_j$, $r_1,\ldots,r_t,u_1,\ldots,u_m\geqslant 0$, $\sum_{i=1}^{t}r_i=1$ 且 $r_1,\ldots,r_t,u_1,\ldots,u_m$ 取值范围为 $\mathbb{Q}$.

2 主要结果

给定 SLC 循环 Ω, 设 Ω 上的增函数法向量空间 $\Phi=\{a^{\mathrm{T}}:a^{\mathrm{T}}(x'-x)\geqslant 1,\forall(x,x')\in\Omega\}$, 则:

引理 1. Ω 上的增函数法向量空间 Φ 是一个凸多面体.

证明: 不失一般性, 设 SLC 程序 Ω 的不等式约束为:

$$\begin{cases}c_{1,1}x_1+\ldots+c_{1,n}x_n+c_{1,n+1}x'_1+\ldots+c_{1,2n}x'_n+c_{1,2n+1}\geqslant 0\\ c_{2,1}x_1+\ldots+c_{2,n}x_n+c_{2,n+1}x'_1+\ldots+c_{2,2n}x'_n+c_{2,2n+1}\geqslant 0\\ \vdots\\ c_{m,1}x_1+\ldots+c_{m,n}x_n+c_{m,n+1}x'_1+\ldots+c_{m,2n}x'_n+c_{m,2n+1}\geqslant 0\end{cases}.$$

Ω 上的增函数设为 $f(x)=a^{\mathrm{T}}x+b$, $a^{\mathrm{T}}=(a_1,a_2,\ldots,a_n)\in\mathbb{Q}^n$, $x=(x_1,x_2,\ldots,x_n)^{\mathrm{T}}\in\mathbb{Q}^n$, $b\in\mathbb{Q}$, 使用 Farkas' Lemma, 有:

$$\forall x,x':\Omega\Rightarrow f(x')\geqslant 1+f(x)\tag{3}$$

等价于:

$$\begin{cases}\exists\lambda_1\geqslant 0,\ldots,\lambda_m\geqslant 0\\ M\triangleq\left(-1\geqslant\sum\nolimits_{i=1}^{m}\lambda_i c_{i,2n+1}\right)\wedge\bigwedge\limits_{j=1}^{n}\left(-a_j=\sum\nolimits_{i=1}^{m}\lambda_i c_{i,j}\right)\wedge\bigwedge\limits_{j=1}^{n}\left(a_j=\sum\nolimits_{i=1}^{m}\lambda_i c_{i,n+j}\right)\end{cases}\tag{4}$$

因为公式 (4) 中均为线性约束, 所以 M 的解集是一个凸多面体. 将该凸多面体投影到只含有 a^{T} 的方向上, 即可得增函数法向量空间 $\Phi=\{a^{\mathrm{T}}:a^{\mathrm{T}}(x'-x)\geqslant 1,\forall(x,x')\in\Omega\}$. 既然 M 是凸多面体, 那么 Φ 作为 M 在 a^{T} 上的投影也是一个凸多面体[14], 故 Φ 具有生成表示为: $\Phi=conv(v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}})+cone(d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}})$. 因此, 对 $\forall a^{\mathrm{T}}\in\Phi$, 有 $a^{\mathrm{T}}=\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}},r_1,\ldots,r_t,u_1,\ldots,u_m\geqslant 0,\sum_{i=1}^{t}r_i=1$.

定理 1. 记 Ω 为 SLC 循环. 令 Φ 为 Ω 上的增函数法向量空间, 其具有生成表示 $\Phi=conv(v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}})+cone(d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}})$. 则对任意固定的常数 $k^*\in\mathbb{R}$, 有:

$$\bigwedge_{a^{\mathrm{T}}\in\Phi}a^{\mathrm{T}}x\geqslant k^*\Leftrightarrow\bigwedge_{i=1}^{t}v_i^{\mathrm{T}}x\geqslant k^*\wedge\bigwedge_{j=1}^{m}d_j^{\mathrm{T}}x\geqslant 0 \tag{5}$$

证明: 由于 Φ 为 Ω 上的增函数法向量空间, 故 $\Phi=\{a^{\mathrm{T}}:a^{\mathrm{T}}(x'-x)\geqslant 1,\forall(x,x')\in\Omega\}$, 且具有生成表示 $\Phi=conv(v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}})+cone(d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}})$, 即对 $\forall a^{\mathrm{T}}\in\Phi$, 有 $a^{\mathrm{T}}=\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}},r_1,\ldots,r_t,u_1,\ldots,u_m\geqslant 0,\sum_{i=1}^{t}r_i=1$.

(1) "右端" $\Rightarrow$ "左端". 由右端得知:

$$\bigwedge_{i=1}^{t}v_i^{\mathrm{T}}x\geqslant k^*\wedge\bigwedge_{j=1}^{m}d_j^{\mathrm{T}}x\geqslant 0 \tag{6}$$

根据 Φ 的生成表示, 对 $\forall a^{\mathrm{T}}\in\Phi$, 有:

$$a^{\mathrm{T}}=\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}} \tag{7}$$

其中, $r_1,\ldots,r_t$, $u_1,\ldots,u_m\geqslant 0$, $\sum_{i=1}^{t}r_i=1$, 且 $\{v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}}\}$ 是增函数法向量空间 Φ 的顶点, $\{d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}}\}$ 是增函数法向量空间 Φ 的射线. 由公式 (6) 得:

$$\bigwedge_{i=1}^{t}v_i^{\mathrm{T}}x\geqslant k^*\wedge\bigwedge_{j=1}^{m}d_j^{\mathrm{T}}x\geqslant 0\Rightarrow\bigwedge_{i=1}^{t}r_iv_i^{\mathrm{T}}x\geqslant r_ik^*\wedge\bigwedge_{j=1}^{m}u_jd_j^{\mathrm{T}}x\geqslant 0 \tag{8}$$

将公式 (8) 右端的所有不等式相加得:

$$\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}x+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}x\geqslant\sum_{i=1}^{t}r_ik^* \tag{9}$$

既然 $\sum_{i=1}^{t}r_i=1$, 故有: $\left(\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}\right)x\geqslant k^*$.

由公式 (7) 可得:

$$a^{\mathrm{T}}x\geqslant k^*.$$

既然 $a^{\mathrm{T}}\in\Phi$ 是任意选取的, 故对 $\forall a^{\mathrm{T}}=\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}\in\Phi$, $r_1,\ldots,r_t$, $u_1,\ldots,u_m\geqslant 0$, $\sum_{i=1}^{t}r_i=1$, 都有:

$$\bigwedge_{i=1}^{t}v_i^{\mathrm{T}}x\geqslant k^*\wedge\bigwedge_{j=1}^{m}d_j^{\mathrm{T}}x\geqslant 0\Rightarrow a^{\mathrm{T}}x\geqslant k^*.$$

进而有:

$$\bigwedge_{i=1}^{t}v_i^{\mathrm{T}}x\geqslant k^*\wedge\bigwedge_{j=1}^{m}d_j^{\mathrm{T}}x\geqslant 0\Rightarrow\bigwedge_{a^{\mathrm{T}}\in\Phi}a^{\mathrm{T}}x\geqslant k^*.$$

(2) "左端" $\Rightarrow$ "右端". 由左端得知:

$$\bigwedge_{a^{\mathrm{T}}\in\Phi}a^{\mathrm{T}}x\geqslant k^* \tag{10}$$

下面先证明: $\bigwedge_{a^{\mathrm{T}}\in\Phi}a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_j^{\mathrm{T}}x\geqslant 0$ 对 $\forall j=1,\ldots,m$ 均成立. 既然 Φ 是凸多面体, 故对 $\forall a^{\mathrm{T}}\in\Phi$, 必存在相应的 $r_1,\ldots,r_t$, $u_1,\ldots,u_m\geqslant 0$, $\sum_{i=1}^{t}r_i=1$ 使得:

$$a^{\mathrm{T}}=\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}.$$

进而有:

$$\bigwedge_{a^{\mathrm{T}}\in\Phi}a^{\mathrm{T}}x\geqslant k^*=\bigwedge_{\substack{r_i\geqslant 0\\ u_j\geqslant 0\\ \sum_{i=1}^{t}r_i=1}}\left[\sum_{i=1}^{t}r_iv_i^{\mathrm{T}}x+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}x\geqslant k^*\right] \tag{11}$$

因此, 要证明 $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_j^{\mathrm{T}}x\geqslant 0$ 等价于证明: $\underset{\substack{r_i\geqslant 0\\ u_j\geqslant 0\\ \sum_{i=1}^{t}r_i=1}}{\wedge}\left[\sum_{i=1}^{t}r_iv_i^jx+\sum_{j=1}^{m}u_jd_j^{\mathrm{T}}x\geqslant k^*\right]\Rightarrow\underset{j=1}{\overset{m}{\wedge}}d_j^{\mathrm{T}}x\geqslant 0$. 不失一般性, 下面仅证明 $\underset{a\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_1^{\mathrm{T}}x\geqslant 0$, 而对 $j=2,\ldots,m$ 的情况可采用类似方法得证. 首先, 取 $r_i=r_i^*,i=1,\ldots,t$, 且 $\sum_{i=1}^{t}r_i^*=1$, 令 $u_2=\ldots=u_m=0$. 将 $r_i=r_i^*,i=1,\ldots,t$ 以及 $u_2=\ldots=u_m=0$ 代入 (11) 右端, 得到公式 (12):

$$\underset{u_1\geqslant 0}{\wedge}\left[\sum_{i=1}^{t}r_i^*v_i^{\mathrm{T}}x+u_1\left(d_1^{\mathrm{T}}x\right)\geqslant k^*\right] \tag{12}$$

易见公式 (12) 中的不等式均来自于公式 (11). 下面采用反证法来证明待证的结论. 假设 $\underset{a\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\not\Rightarrow d_1^{\mathrm{T}}x\geqslant 0$. 那么, 必 $\exists x^*\in\left\{x:\underset{a\in\Phi}{\wedge}a^{\mathrm{T}}x\geqslant k^*\right\}$, s.t. $d_1^{\mathrm{T}}x^*<0$. 故在公式 (12) 中, 因 $\sum_{i=1}^{t}r_i^*v_i^{\mathrm{T}}x^*$ 是一个固定值, 且 $d_1^{\mathrm{T}}x^*<0$, 则当 $u_1\to+\infty$ 时, 必 $\exists N>0$, 当 $u_1>N$ 时, 有 $\sum_{i=1}^{t}r_i^*v_i^{\mathrm{T}}x^*+u_1d_1^{\mathrm{T}}x^*<k^*$. 不妨取 $u_1=u_1^*>N$, 进而 $\exists(a^*)^{\mathrm{T}}=\left(\sum_{i=1}^{t}r_i^*v_i^{\mathrm{T}}+u_1^*d_1^{\mathrm{T}}\right)\in\Phi$, 使得:

$$(a^*)^{\mathrm{T}}x^*<k^* \tag{13}$$

又 $a^{*\mathrm{T}}\in\Phi$, 故有: $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow(a^*)^{\mathrm{T}}x\geqslant k^*$.

这表明, 对 $\forall x\in\left\{x:\underset{a^{\mathrm{T}}\in\Phi}{\wedge}a^{\mathrm{T}}x\geqslant k^*\right\}$, 有:

$$a^{*\mathrm{T}}x\geqslant k^* \tag{14}$$

根据假设, 既然 $x^*\in\left\{x:\underset{a^{\mathrm{T}}\in\Phi}{\wedge}a^{\mathrm{T}}x\geqslant k^*\right\}$, 那么由公式 (14) 可得:

$$a^{*\mathrm{T}}x^*\geqslant k^* \tag{15}$$

这显然与公式 (13) 矛盾, 故假设不成立. 因此可得: $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_1^{\mathrm{T}}x\geqslant 0$.

上述证明方法可类似应用于证明: $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_j^{\mathrm{T}}x\geqslant 0\ (\forall j=1,\ldots,m)$, 故在此省略其证明. 综上所述可证得:

$$\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow d_j^{\mathrm{T}}x\geqslant 0\ (\forall j=1,\ldots,m) \tag{16}$$

又因为 $v_i^{\mathrm{T}},i=1,\ldots,t$ 为多面体 Φ 的顶点, 故 $\{v_i^{\mathrm{T}}\}\subseteq\Phi$, 所以有 $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow\underset{i=1}{\overset{t}{\wedge}}v_i^{\mathrm{T}}x\geqslant k^*$. 结合公式 (16) 可得:

$$\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*\Rightarrow\underset{i=1}{\overset{t}{\wedge}}v_i^{\mathrm{T}}x\geqslant k^*\wedge\underset{j=1}{\overset{m}{\wedge}}d_j^{\mathrm{T}}x\geqslant 0 \tag{17}$$

定理 1 表明, 当 $k=k^*$ 被固定时, Ω 上型如 $a^{\mathrm{T}}x-k^*$ 的增函数构成的不等式组 $\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k^*$ 所刻画的解集合 $\{x\in\mathbb{Q}^n:a^{\mathrm{T}}x\geqslant k^*,a^{\mathrm{T}}\in\Phi\}$ 具有有限个边界, 其为一个凸多面体. 更重要的是, 定理 1 表明, 公式 (5) 左端的无穷多个不等式所构成的不等式组可以等价地用其中的有限个不等式来表述, 这为后面部分增函数空间的构造带来便利. 由定理 1 可得定理 2.

定理 2. 记号同上. 令 Φ 为 Ω 上的增函数法向量空间, 则:

$$\underset{k\in\mathbb{R}}{\wedge}\underset{a^{\mathrm{T}}\in\Phi}{\wedge} a^{\mathrm{T}}x\geqslant k\iff\underset{k\in\mathbb{R}}{\wedge}\left[\underset{i=1}{\overset{t}{\wedge}}v_i^{\mathrm{T}}x\geqslant k\wedge\underset{j=1}{\overset{m}{\wedge}}d_j^{\mathrm{T}}x\geqslant 0\right] \tag{18}$$

其中, v_i^{T} 和 d_j^{T} 分别为增函数法向量空间 Φ 的顶点和射线.

定理 2 中, 公式 (18) 左边即为 Ω 上的增函数空间 *Inc* (由 Ω 上的所有增函数 $f(x)=a^{\mathrm{T}}x-k$ 构成的集合). 但由于公式 (18) 左端中不等式组中的 k 有无穷取值, 为后续计算带来困难. 因此, 在实际计算中, 我们提前将 k 的值确定, 比如令 $k=10$. 从而保证了形如 $f(x)=a^{\mathrm{T}}x-10\geqslant 0$ 的所有约束均可以由有限个不等式等价刻画, 为后续计算提供便利. 一般地, 当 k 被固定为 k_0 时, 公式 (18) 右端就具有有限个不等式, 这有限个不等式刻画了由形如 $f(x)=a^{\mathrm{T}}x-k_0$ 的增函数所构成的增函数空间 Inc_{k_0}, 显然, $Inc_{k_0}\subseteq Inc$, 即 Inc_{k_0} 是增函数空间 *Inc* 中的部分增函数构成的集合 (即称为部分增函数空间). 实验表明, 对于现有文献中实例, 提前取定 k 值的方式能较好地解决问题.

当 k 值被固定时, 将公式 (18) 右端不等式加入到原 SLC 循环的不等式约束中, 得到新的 SLC 循环 Ω^+. 然后, 我们需进一步判定 Ω^+ 上是否具有线性秩函数. 为此, 我们将采用文献 [14] 中定理 4 来解决这一问题. 记 SLC 循环程序 $\Omega=\{(x,x')\in\mathbb{Q}^{2n}:Ax+A'x'\geqslant a\}$. 记 $\Omega_H=\{(dx,dx')\in\mathbb{Q}^{2n}:Adx+A'dx'\geqslant 0\}$ 为 Ω 的 recession 方向. 这里, $dx=$

$(dx_1,\ldots,dx_n)$, $dx'=(dx'_1,\ldots,dx'_n)$. 为方便, Ω_H 也用来表示 recession 方向 Ω_H 对应的约束, 即 $\Omega_H \triangleq Adx+A'dx' \geqslant 0$. 下面引入文献 [14] 中的定理 4, 即本文定理 3.

定理 3. 给定如上定义的 SLC 循环程序 Ω [14]. 令 Ω_H 为 Ω 上的 recession 方向, 令:

$$\Delta \triangleq \Omega \wedge \Omega_H \wedge \{x'=x+dx\} = \{Ax+A'x' \geqslant a \wedge Adx+A'dx' \geqslant 0 \wedge x'=x+dx\},$$

则 Δ 无解当且仅当 Ω 上存在线性秩函数.

定理 3 表明在不计算出线性秩函数的情况下, 就可以通过判定一个不等式组 Δ 是否有解来确定 Ω 上是否存在线性秩函数.

在本文方法中, 当根据 Ω 所构造出的 Δ 有解时, 则说明 Ω 上不存在线性秩函数. 因此, 为判定 Ω 上是否可终止, 我们将计算 Ω 上的部分增函数空间, 并利用部分增函数空间对应的不等式约束去对 Ω 进行划分得到 Ω^+. 根据前述分析, 要计算部分增函数空间, 就需要首先计算 Ω 的增函数法向空间 Φ. 为方便, 记 Ω 上增函数法向量空间为 $\Phi_\Omega=\{a^{\mathrm{T}}:a^{\mathrm{T}}(x'-x)\geqslant 1,\forall(x,x')\in\Omega\}$, 其生成表示为: $\Phi_\Omega=conv(v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}})+cone(d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}})$. 根据定理 2, 在公式 (18) 的右端中, 我们令 $k=k_0$. 由定理 2 易知, 此刻, 公式 (18) 右端的不等式约束: $\bigwedge_{i=1}^{t} v_i^{\mathrm{T}}x \geqslant k_0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}}x \geqslant 0$ 的系数向量定义了部分的增函数空间. 接下来, 将不等式约束 $\bigwedge_{i=1}^{t} v_i^{\mathrm{T}}x \geqslant k_0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}}x \geqslant 0$ 加入到 Ω 中, 那么 Ω 被部分增函数空间所确定的约束划分后即得到新的 SLC 循环程序, 记为 $\Omega_1^+(k=k_0)$:

$$\Omega_1^+(k=k_0) \triangleq \left\{Ax+A'x' \geqslant a \wedge \bigwedge_{i=1}^{t} v_i^{\mathrm{T}}x \geqslant k_0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}}x \geqslant 0\right\}.$$

同理, 根据定理 3, 在 Ω_1^+ 上构造:

$$\begin{aligned}&\Delta_{\Omega_1^+}(k=k_0) \triangleq \Omega_1^+ \wedge (\Omega_1^+)_H \wedge \{x'=x+dx\}\\ &=\left\{Ax+A'x' \geqslant a \wedge \bigwedge_{i=1}^{t} v_i^{\mathrm{T}}x \geqslant k_0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}}x \geqslant 0 \wedge Adx+A'dx' \geqslant 0 \wedge \bigwedge_{i=1}^{t} v_i^{\mathrm{T}}dx \geqslant 0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}}dx \geqslant 0 \wedge x'=x+dx\right\}.\end{aligned} \tag{19}$$

由定理 3 知 $\Delta_{\Omega_1^+}(k=k_0)$ 无解当且仅当循环程序空间 $\Omega_1^+(k=k_0)$ 存在线性秩函数.

3 算法与实验

下面通过文献 [19] 中的一个例子来说明算法流程. 给出一个 SLC 循环程序: while$(x-z\geqslant 0)$ $x'\leqslant x+y$; $y'\leqslant y-z$; $z'\geqslant z+1$. 故有: $\Omega \triangleq \{x-z\geqslant 0,\ x'\leqslant x+y,\ y'\leqslant y-z,\ z'\geqslant z+1\}$.

第 1 步: 验证 Ω 上是否存在不动点. 定义: $Q \triangleq \Omega \wedge \{x'=x,y'=y,z'=z\}$. 调用 *maple* 命令中的 *LinearSolve* 计算可知 Q 无解, 表示 Ω 上不存在不动点.

第 2 步: 根据定理 3, 构造 Ω 上的 Δ_Ω: $\Delta_\Omega=\Omega\wedge\Omega_H\wedge\{x'=x+dx,y'=y+dy,z'=z+dz\}$. 经过 *LinearSolve* 计算, Δ_Ω 有解, 根据定理 3 可知 Ω 上不存在线性秩函数, 故需要继续计算 Ω 上的部分增函数空间, 并利用部分增函数空间所确定的不等式约束对 Ω 划分. 此时, 为便于计算, 我们取划分常数 $k=k_0=10$, 并设置最大划分深度 $depth=3$. 此处的 k_0, *depth* 的取值是随机的. 一般地, 为便于计算, k 的取值可以是取值于有理数, 整数, 但无法事先确定 k 取值的上界. 但本文的实验结果表明, k 越大, 划分所得子空间就越小, 其上存在线性秩函数的可能性就越大. 因此, k 是可调参数. 在本文实验中, 我们将 k 的值固定为 10. 另外, 本文的方法是一个不断利用增函数集合对当前空间进行划分的过程. 该过程仅当划分所得的新空间上存在线性秩函数则停止. 但目前还无法事先得知划分多少次能找到线性秩函数. 因此, 为保证本文算法能停机, 需要事先固定划分的次数 (即划分深度 *depth*) 的上界 (即最大划分深度) *MaxDepth* 的值. 本文实验中, 我们将 *MaxDepth* 的值设置为 3. 当给定的 SLC 循环在当前的 k_0,*depth* 的取值下无法判断其是否终止时, 可适当增大 k_0, *depth* 的取值.

第 3 步: 构造 Ω 上的增函数法向量空间: $\Phi_\Omega=\{a^{\mathrm{T}}=(a_1,a_2,a_3)\in\mathbb{Q}^3:a^{\mathrm{T}}(x'-x)\geqslant 1,\forall(x,x')\in\Omega\}$. 对 Ω 使用 Farkas' Lemma 可以得到: $\Phi_\Omega=\{a^{\mathrm{T}}=(a_1,a_2,a_3)\in\mathbb{Q}^3:a_1=0,a_2=0,a_3\geqslant 1\}$.

第 4 步: 用 *PPL* [29]计算 Φ_Ω 的生成表示: $\Phi_\Omega=conv\{(0,0,1)\}+cone\{(0,0,1)\}$.

第 5 步: 根据定理 2, 用 Φ_Ω 构造出 $\{z\geqslant k,z\geqslant 0\}$ 并加入到 Ω 中, 得到新的 SLC 循环:

$$\Omega_1^+(k=10)=\Omega\wedge\{z\geqslant k,z\geqslant 0\}=\{x-z\geqslant 0,x'\leqslant x+y,y'\leqslant y-z,z'\geqslant z+1,z\geqslant k,z\geqslant 0\}.$$

第 6 步: 利用定理 3 判断 $\Omega_1^+(k=10)$ 上是否存在线性秩函数, 故根据公式 (19) 构造:

$$\Phi_{\Omega_1^+}(k=10)=\{x-z\geqslant 0,x'\leqslant x+y,y'\leqslant y-z,z'\geqslant z+1,z\geqslant k,z\geqslant 0\}$$
$$\wedge\{dx-dz\geqslant 0,dx'\leqslant dx+dy,dy'\leqslant dy-dz,dz'\geqslant dz,dz\geqslant 0\}\wedge\{x'=x+dx,y'=y+dy,z'=z+dz\}.$$

第 7 步: 验证 $\Delta_{\Omega_1^+}(k=10)$ 是否有解. 代入 $k=10$, 使用 *LinearSolve* 计算得知, $\Delta_{\Omega_1^+}(k=10)$ 的确有解. 因此, 根据定理 3 可知 $\Omega_1^+(k=10)$ 上不存在线性秩函数. 因而需继续计算 $\Omega_1^+(k=10)$ 上的部分增函数空间, 并用该增函数空间所确定的不等式约束对 $\Omega_1^+(k=10)$ 划分, 再判断计算划分后循环程序空间上是否存在线性秩函数. 重复第 3–7 步的计算.

首先, 第 3 步计算出 $\Omega_1^+(k=10)$ 上的增函数法向量空间: $\Phi_{\Omega_1^+(k=10)}=\{a^{\mathrm{T}}=(a_1,a_2,a_3)\in\mathbb{Q}^3:a_1=0,a_2\leqslant 0,a_3\geqslant 0,a_2\leqslant\frac{1}{10}a_3-\frac{1}{10}\}$.

第 4 步计算其生成表示: $\Phi_{\Omega_1^+(k=10)}=conv\left\{\left(0,-\frac{1}{10},0\right),(0,0,1)\right\}+cone\{(0,-1,0),(0,0,1)\}$.

第 5 步根据定理 2 构造:

$$\Omega_2^+(k=10)=\Omega_1^+(k=10)\wedge\left\{-\frac{1}{10}y\geqslant k,z\geqslant k,-y\geqslant 0,z\geqslant 0\right\}$$
$$=\left\{x-z\geqslant 0,x'\leqslant x+y,y'\leqslant y-z,z'\geqslant z+1,z\geqslant k,z\geqslant 0,-\frac{1}{10}y\geqslant k,-y\geqslant 0\right\}.$$

第 6 步构造:

$$\Delta_{\Omega_2^+}(k=10)\triangleq\Omega_2^+(k=10)\wedge(\Omega_2^+(k=10))_H\wedge\{x'=x+dx,y'=y+dy,z'=z+dz\}$$
$$=\left\{x-z\geqslant 0,x'\leqslant x+y,y'\leqslant y-z,z'\geqslant z+1,z\geqslant k,z\geqslant 0,-\frac{1}{10}y\geqslant k,-y\geqslant 0\right\}$$
$$\wedge\{dx-dz\geqslant 0,dx'\leqslant dx+dy,dy'\leqslant dy-dz,dz'$$
$$\geqslant dz,dz\geqslant 0,-\frac{1}{10}y\geqslant 0,-dy\geqslant 0\}\wedge\{x'=x+dx,y'=y+dy,z'=z+dz\}.$$

然后判断 $\Delta_{\Omega_2^+}(k=10)$ 是否有解. 代入 $k=10$ 计算发现该不等式组无解. 因此, 根据定理 3 可知, $\Omega_2^+(k=10)$ 上存在线性秩函数, 故循环程序 Ω 是终止的. 计算结束.

算法 1 给出基于上述理论的 SLC 循环程序终止性分析的具体算法 TASLC (termination analysis of single-path linear constraint loops).

算法 1. TASLC.

输入: $Ax+A'x'\geqslant a$: 循环程序的约束; x: 循环程序中的变量; k_o: 划分常数; *MaxDepth*: 最大划分深度;
输出: T (True, 存在), F (False, 不存在), U (Undefined, 未知).

1. **function** TASLC($Ax+A'x'\geqslant a,x,k_o,MaxDepth$)
2. **if** $Ax+A'x'\geqslant a$ 上存在不动点 **then**
3. **return** F;
4. **end if**
5. $\Delta=Ax+A'x'\geqslant a\wedge Adx+A'dx'\geqslant 0\wedge x'=x+dx$;
6. **if** Δ 无解 **then**
7. **return** T;
8. **end if**
9. $\Omega=\{Ax+A'x'\geqslant a\}$ and $k=k_o$ and $depth=1$;
10. **while** $depth\leqslant MaxDepth$ **do**
11. $\varphi_\Omega=\{a^{\mathrm{T}}:a^{\mathrm{T}}(x'-x)\geqslant 1\}\leftarrow$ Farkas' Lemma(Ω,x);
12. $\varphi_\Omega=\{v_1^{\mathrm{T}},\ldots,v_t^{\mathrm{T}}\}+\{d_1^{\mathrm{T}},\ldots,d_m^{\mathrm{T}}\}\leftarrow PPL(\varphi_\Omega)$;

13. $\Omega_1^+ = \Omega \wedge \bigwedge_{i=1}^{t} v_i^{\mathrm{T}} x \geqslant k \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}} x \geqslant 0$;
14. $\Delta(k) = \Omega_1^+ \wedge Adx + A'dx' \geqslant 0 \wedge \bigwedge_{i=1}^{t} v_i^{\mathrm{T}} x \geqslant 0 \wedge \bigwedge_{j=1}^{m} d_j^{\mathrm{T}} x \geqslant 0 \wedge x' = x + dx$;
15. **if** $\Delta(k)$ 无解 **then**
16. **return** T;
17. **end if**
18. $\Omega = \Omega_1^+$ and $depth = depth + 1$;
19. **end while**
20. **return** U;
21. **end function**

实验是在一台 7.85 GB 可用内存、处理器实际频率为 1.99 GHz、操作系统是 VM VirtualBox 上安装的 Ubuntu 18 的计算机上进行的, 主要用到 *PPL* 和 *maple* 工具包 *RegularChains* 中的 *LinearSolve* .

对表 1 中的循环程序进行实验. 循环程序 1–4 来自文献 [18], 循环程序 5–41 来自文献 [21], 循环程序 42–43 来自文献 [19]. 当 $k_0 = 10$, $MaxDepth = 3$. 实验结果如表 1 所示. 表 1 中 *depth* 为计算出结果时的划分深度. 当返回 T 时, $depth = 0$ 表明原循环程序 Ω 上存在线性秩函数; $depth = 1$ 表明对原循环程序 Ω 进行了一次划分, 划分后的程序空间 Ω_1^+ 中存在线性秩函数, 以此类推. $depth = 4$ 表明当前循环程序划分次数超过了最大划分深度 $MaxDepth$, 返回 U , 此时认为本算法无法判定该循环的终止性. 返回 F 时, 表明该循环程序上存在不动点, 故不终止. 表 1 中的 44 个循环, 1–4 以及 17–43 是终止的, 5–16 是非终止的. 对于严格不等式如 $x > y$, 实验中会将其转换为 $x \geqslant y + 1$ 的非严格不等式形式 (原文献中, 出现在严格不等式中的程序变量都是整数变量), 再对后者调用本文算法进行处理. TASLC 算法计算出的结果中, 循环 1–4 返回 T , 循环 5–16 结果均为 F 或 U , 循环 16–43 均返回 T . 以第 5–16 个循环为例, 这 12 个循环中有的返回 F, 有的返回 U. 根据我们的算法, 返回 F, 表明该循环是不可终止的. 因此, F 是在算法第 3 步返回; 若返回 U, 表明当达到最大划分深度 (最大划分次数 3) 时, 当前的程序空间上仍不存在线性秩函数. 此时, 我们的算法将不再往下进行, 因而原始程序到底是否终止无法确定, 故返回 U. 在我们的算法中, U 是在算法的第 20 步返回.

表 1 循环实例

编号	loop	*depth*	Termination
1	while $(x \geqslant 0, y \leqslant -1)$ $y' \leqslant y - 1$; $x' \leqslant x + y$;	0	T
2	while $(x \geqslant 0)$ $x' \leqslant x + y$; $y' \leqslant -y - 1$;	1	T
3	while $(x \geqslant -1)$ $y' \leqslant y - 1$; $x' \leqslant x + y$;	1	T
4	while $(x \geqslant 1)$ $x' = y$; $y' = y - 1$;	1	T
5	while $(x \geqslant 0)$ $x' = -2x + 10$;	0	F
6	while $(x > 0)$ $x' = x + y$; $y' = y + z$;	0	F
7	while $(x < 0)$ $x' = x + y$; $y' = y - 1$;	4	U
8	while $(x > 0)$ $x' = x + y$; $y' = -2y$;	0	F
9	while $(x < y)$ $x' = x + y$; $y' = -2y$;	0	F
10	while $(x < y)$ $x' = x + y$; $2y' = y$;	0	F
11	while $(4x - 5y > 0)$ $x' = 2x + 4y$; $y' = 4x$;	4	U
12	while $(x < 5)$ $x' = x - y$; $y' = x + y$;	0	F
13	while $(x > 0, y > 0)$ $x' = -2x + 10y$;	0	F
14	while $(x > 0)$ $x' = x + y$;	0	F
15	while $(x < 10)$ $x' = -y$; $y' = y + 1$;	4	U

表 1 循环实例 (续)

编号	loop	*depth*	Termination
16	while $(x<0)$ $x'=x+z;\ y'=y+1;\ z'=-2y;$	4	U
17	while $(x>0, x<100)$ $x'\geqslant 2x+10;$	0	T
18	while $(x>1)$ $-2x'=x;$	0	T
19	while $(x>1)$ $2x'\leqslant x;$	0	T
20	while $(x>0)$ $2x'\leqslant x;$	0	T
21	while $(x>0)$ $x'=x+y;\ y'=y-1;$	1	T
22	while $(x>0, x<y)$ $x'=2x;\ y'=y+1;$	1	T
23	while $(x>0)$ $x'=x-2y;\ y'=y+1;$	1	T
24	while $(x>0, x<n)$ $x'=-x+y-5;\ y'=2y;\ n'=n;$	0	T
25	while $(x>0, y<0)$ $x'=x+y;\ y'=y-1;$	0	T
26	while $(x-y>0)$ $x'=-x+y;\ y'=y+1;$	1	T
27	while $(x>0)$ $x'=y;\ y'=y-1;$	1	T
28	while $(x>0)$ $x'=x+y-5;\ y'=-2y;$	1	T
29	while $(x+y>0)$ $x'=x-1;\ y'=-2y;$	1	T
30	while $(x>y)$ $x'=x-y;\ 1\leq y'\leq 2;$	0	T
31	while $(x>0)$ $x'=x+y;\ y'=-y-1;$	1	T
32	while $(x>0)$ $x'=y;\ y'\leqslant -y;$	1	T
33	while $(x<y)$ $x'=x+1;\ y'=z;\ z'=z;$	1	T
34	while $(x>0)$ $x'=x+y;\ y'=y+z;\ z'=z-1;$	2	T
35	while $(x+y\geqslant 0, x\leqslant z)$ $x'=2x+y;\ y'=y+1;\ z'=z;$	1	T
36	while $(x>0, x\leqslant z)$ $x'=2x+y;\ y'=y+1;\ z'=z;$	1	T
37	while $(x\geqslant 0)$ $x'=x+y;\ y'=z;\ z'=-z-1;$	2	T
38	while $(x-y>0)$ $x'=-x+y;\ y'=z;\ z'=z+1;$	1	T
39	while $(x>0, x<y)$ $x'>2x;\ y'=z;\ z'=z;$	1	T
40	while $(x\geqslant 0, x+y\geqslant 0)$ $x'=x+y+z;\ y'=-z-1;\ z'=z;$	0	T
41	while $(x+y\geqslant 0, x\leqslant n)$ $x'=2x+y;\ y'=z;\ z'=z+1;\ n'=n;$	2	T
42	while $(x\geqslant 0)$ $x'\leqslant x+y;\ y'\leqslant y-1;$	1	T
43	while $(x-z\geqslant 0)$ $x'\leqslant x+y;\ y'\leqslant y-z;\ z'\geqslant z+1;$	2	T

当算法返回 U 时, 可能是因为循环本身并不终止导致的, 也可能因为当前设置的划分常数 k_0、最大划分深度 *MaxDepth* 不够大导致没有计算出正确的结果. 但从实验结果来看, 当返回结果为 U 时, 循环程序均是非终止的. 上述 43 个循环程序的实验结果与文献 [18,19,21] 计算出的实验结果是一致的. 与文献 [18,19] 不同, 由于本文的方法仅涉及线性约束求解, 其时间复杂度相比于文献 [18,19] 中的非线性约束求解较低.

表 2 针对返回为 T 的循环程序, 用不同论文中的方法进行了测试. 可以看到, 文献 [19] 方法由于采用非线性约束求解, 其终止性判定时间开销超过了 6 608 s. 文献 [21] 与本文方法均采用线性约束求解, 时间消耗较低, 但由于本文方法不需要计算具体秩函数, 时间消耗更低. 另外, 对 k 设不同值时, 所得到的结果也是不一样的, 如表 3.

表 3 中, 当 $k=-10$ 时, 有 25 个循环判定结果为 T, 8 个判定为 F, 11 个判定为 U. 与 $k=10$ 比较后可以看出, 直观地, 当 k 取较大值时, 表明利用部分增函数空间在原循环程序空间上划分得到的子集越小, 因而越容易找到子集上的线性秩函数.

除此之外, 由于本文主要研究单分支线性循环的终止性, 故我们抽取了 SVCOMP 中的 84 个循环程序, 并将本文方法与工具 iRankFinder[30]针对这些程序做了对比. iRankFinder 是目前进行这类循环程序终止性分析颇为高效的工具. 实验结果如表 4, 其中的记号 T (终止), F (不可终止), U (不确定) 含义同上.

表 2 时间对比 (s)

编号	loop	文献[19]方法	文献[21]方法	Ours
1	while $(x\geqslant 0, y\leqslant -1)$ $y'\leqslant y-1$; $x'\leqslant x+y$;	8.014	0.031	0.033
2	while $(x\geqslant 0)$ $x'\leqslant x+y$; $y'\leqslant -y-1$;	19.362	0.14	0.032
3	while $(x\geqslant -1)$ $y'\leqslant y-1$; $x'\leqslant x+y$;	13.895	0.031	0.023
4	while $(x\geqslant 1)$ $x'=y$; $y'=y-1$;	12.261	0.046	0.025
17	while $(x>0, x<100)$ $x'\geqslant 2x+10$;	0.512	0.046	0.001
18	while $(x>1)-2x'=x$;	0.358	0.031	0.008
19	while $(x>1)$ $2x'\leqslant x$;	0.702	0.031	0.008
20	while $(x>0)$ $2x'\leqslant x$;	0.312	0.14	0.008
21	while $(x>0)$ $x'=x+y$; $y'=y-1$;	12.542	0.031	0.025
22	while $(x>0, x<y)$ $x'=2x$; $y'=y+1$;	⩾600	0.046	0.029
23	while $(x>0)$ $x'=x-2y$; $y'=y+1$;	31.231	0.109	0.026
24	while $(x>0, x<n)$ $x'=-x+y-5$; $y'=2y$; $n'=n$;	⩾600	0.062	0.063
25	while $(x>0, y<0)$ $x'=x+y$; $y'=y-1$;	1.357	0.125	0.015
26	while $(x-y>0)$ $x'=-x+y$; $y'=y+1$;	162.631	0.062	0.037
27	while $(x>0)$ $x'=y$; $y'=y-1$;	55.271	0.031	0.025
28	while $(x>0)$ $x'=x+y-5$; $y'=-2y$;	237.667	0.109	0.029
29	while $(x+y>0)$ $x'=x-1$; $y'=-2y$;	156.718	0.046	0.027
30	while $(x>y)$ $x'=x-y$; $1\leqslant y'\leqslant 2$;	170.290	0.046	0.016
31	while $(x>0)$ $x'=x+y$; $y'=-y-1$;	233.783	0.109	0.034
32	while $(x>0)$ $x'=y$; $y'\leqslant -y$;	580.292	0.046	0.074
33	while $(x<y)$ $x'=x+1$; $y'=z$; $z'=z$;	415.274	0.046	0.035
34	while $(x>0)$ $x'=x+y$; $y'=y+z$; $z'=z-1$;	142.116	0.062	0.057
35	while $(x+y\geqslant 0, x\leqslant z)$ $x'=2x+y$; $y'=y+1$; $z'=z$;	184.352	0.046	0.033
36	while $(x>0, x\leqslant z)$ $x'=2x+y$; $y'=y+1$; $z'=z$;	174.326	0.062	0.048
37	while $(x\geqslant 0)$ $x'=x+y$; $y'=z$; $z'=-z-1$;	124.301	0.093	0.070
38	while $(x-y>0)$ $x'=-x+y$; $y'=z$; $z'=z+1$;	⩾600	0.046	0.035
39	while $(x>0, x<y)$ $x'>2x$; $y'=z$; $z'=z$;	⩾600	0.046	0.041
40	while $(x\geqslant 0, x+y\geqslant 0)$ $x'=x+y+z$; $y'=-z-1$; $z'=z$;	240.257	0.031	0.065
41	while $(x+y\geqslant 0, x\leqslant n)$ $x'=2x+y$; $y'=z$; $z'=z+1$; $n'=n$;	⩾600	0.062	0.079
42	while $(x\geqslant 0)$ $x'\leqslant x+y$; $y'\leqslant y-1$;	31.028	0.046	0.017
43	while $(x-z\geqslant 0)$ $x'\leqslant x+y$; $y'\leqslant y-z$; $z'\geqslant z+1$;	⩾600	0.062	0.045
	Sum	⩾6608	1.92	1.063

表 3 不同 k 值的实验

返回结果	−10	0	10
T	25	25	31
F	8	8	8
U	11	11	5

表 4 TASLC 与 iRankFinder 的比较

TASLC	F	T	U	U	F	F	T	T	U
iRankFinder	U	U	U	F	F	T	F	T	T
循环个数	6	3	11	23	20	0	0	21	0

表 4 中记录了针对 84 个循环, TASLC 方法与 iRankFinder 对同一个循环的判定结果. 比如在表 4 第 2 列中, 对 84 个循环中的其中 6 个循环, TASLC 判断它们是不终止的, 而 iRankFinder 无法确定它们是否不终止; 第 5 列, 对其中 23 个循环, 工具 iRankFinder 能判定它们都是不终止的, 而 TASLC 无法判定其是否不终止; 但第 3 列表明, 对其中 3 个循环, TASLC 能判定它们都是终止的, 但 iRankFinder 无法判定其是否是终止的. 综上所述, 针对不可

终止循环的判定上, 由于 iRankFinder 采用不变集探测技术, 故 iRankFinder 表现得更优异; 但在可终止循环的判定上, TASLC 则稍占优势. 上述 84 个循环被记录在如下网站: https://github.com/MinghaiQingtong/PPL_Project/blob/master/Experiments/NestTASLCCom.md.

4 相关工作

单分支线性约束循环程序的终止性问题在国内外已被广泛研究[11−21]. 早期研究集中于这类循环的全局线性秩函数合成[11−13]. 之后字典序秩函数的概念被提出用以证明这类循环的终止性[15−17]. 2013 年, 基于增函数划分的思想, 文献 [18] 提出了 eventual 秩函数的概念. 具体地, 给定一个线性程序, 其程序空间记为 Ω. 若 Ω 上没有线性秩函数, 则文献 [18] 的方法计算 Ω 上的其中一个增函数 f_1, 并用 f_1 来划分 Ω 得到 Ω^1. 若 Ω^1 上有线性秩函数, 则表明原始程序是可终止; 但当 Ω^1 上没有线性秩函数, 文献 [18] 则没有进一步处理. 但重要的是, 增函数的性质——将原空间 Ω 上的终止性问题等级地归结为 Ω^1 上的终止性问题, 为终止性分析提供了新的思路. 由于文献 [18] 仅利用增函数进行了一次划分, 故文献 [19] 拓展了文献 [18] 中的工作, 将利用增函数进行一次划分推广到进行多次划分. 最近, 多阶段秩函数的定义在文献 [20] 中被提出. 文献 [21] 进一步证明在线性约束循环的终止性分析方面, 多阶段秩函数与字典序秩函数具有同等能力. 由于多阶段秩函数的定义中涉及诸如 $\forall(x,x')\exists i$ 的混合量词消去问题使得并不便于计算, 故文献 [21] 在证明了多阶段秩函数与嵌套秩函数的等价性基础上, 将多阶段秩函数的计算归结为嵌套秩函数的计算. 因此, 文献 [21] 实验表格中所得到的秩函数均为 nested 秩函数. 由于文献 [11−18,20,21] 考虑的是线性约束循环程序的线性 (全局, 字典序, eventual, nested) 秩函数的计算问题, 故 Farkas' Lemma 能够被应用于上述各类线性秩函数的计算. 但与上述基于 Farkas' Lemma 的方法不同, 针对线性约束循环程序, 文献 [14] 构造了相应的线性约束系统, 并证明了给定的线性循环程序存在线性秩函数当且仅当其对应的线性约束系统无解. 该结果的建立可以让我们无需真正计算出线性秩函数就能判断给定的线性循环是否具有线性秩函数. 文献 [14] 中方法的特点在于: 当循环程序中的约束增多时, 所构造的线性约束系统中的变量个数仍保持不变, 而传统基于 Farkas' Lemma 的方法此刻将会随着约束的增多而引入更多的变量, 增加计算开销.

本文中, 我们将文献 [14] 中提出的线性秩函数存在性判定方法与文献 [19] 中增函数的"多次划分"的思路相结合, 提出了用于单分支线性约束循环终止分析的方法. 但, 不同于文献 [19] 中的方法, 本文方法中, 每次都计算当前空间 Ω_{j-1} 的一个部分增函数空间 f_j^{cone} 来对 Ω_{j-1} 进行划分 (而文献 [19] 中每次仅计算一个增函数), 这样划分所得的新空间比仅仅用一个增函数划得的新空间更小, 提高了新空间上线性秩函数存在的可能性. 更重要的是, 程序空间上的增函数有无穷多个, 而不同增函数的选取有可能影响到划分后所得空间上是否存在线性秩函数. 因此, 本文方法利用部分增函数空间对当前程序空间进行划分, 一定程度上避免了最优的一个增函数的选取. 本文方法也与文献 [21] 中 nested 秩函数 (或多阶段秩函数方法) 方法不同. 本文的方法是一种渐进式的方法, 即: 不断利用所计算的增函数集合 (部分增函数空间) 将原始程序空间划分得更小, 从而有利于在更小的空间上找到可能的线性秩函数. 若更小的空间没有线性秩函数, 则又计算新空间上的增函数集合, 并用该增函数集合去划分新空间. 显然, 倘若在更小的空间上没有找到线性秩函数, 那么这个划分的过程是可以一直往下进行的, 直到达到本文设置的划分上界 *MaxDepth* 为止. 一般地, 程序空间上的线性增函数有无穷多个 (即增函数集合中的元素个数为无穷), 不同的线性增函数划分所得的新空间的形状会有所不同. 线性增函数的"无穷性"将导致无法通过计算机来判定一个包含无穷多个线性约束的系统是否无解. 为避免"无穷性"给计算带来的不便, 本文证明了增函数空间具有凸多面体结构, 因而利用其生成表示, 可以将无穷增函数问题归结为有限增函数问题, 将"无穷"变为"有穷", 为后续计算提供便利, 这也是本文方法区别于其他方法的一个特点. 此外, 在判定当前空间上有无线性秩函数时, 我们并没有直接计算出一个秩函数, 而是调用了文献 [14] 中的方法, 即: 通过判定某个线性约束系统无解来判定线性秩函数的存在.

5 总 结

本文对文献 [18,19] 中的增函数方法进行了推广, 在 SLC 循环程序空间 Ω 上利用部分增函数空间对其进行多

次划分, 并对划分后的新程序空间进行线性秩函数存在性 (而非实际计算) 验证, 据此设计了新的算法 TASLC 来分析 SLC 循环的终止性问题. 此外, 文献 [19] 的方法涉及非线性约束求解因而效率不高. 本文方法所涉及的约束均是线性不等式约束, 可利用线性规划算法在较快时间内给出解答, 因而较文献 [19] 中方法, 本文方法效率更高. 此外, 与文献 [21] 求出具体秩函数这一思路不同, 本文方法基于增函数的概念, 将原程序空间上的终止性判定问题等价归结为更小的程序子空间上的终止性判定问题, 仅需判定子空间上的线性秩函数存在性问题, 无需求出具体的线性秩函数. 我们使用新算法对文献 [18,19,21] 中的循环程序进行了测试. 实验结果与这 3 篇文献中的实验结果完全相同, 证明本算法是可行有效的.

但本算法同时也存在一定局限性. 对一个本身并不终止的循环程序, 无论使用增函数空间对该循环程序空间划分多少次, 其所得子空间上仍然是非终止的 (永远不存在线性秩函数), 这将造成上述划分过程无法停止. 为了尽量避免本算法对程序空间进行无限次划分, 一方面, 我们在算法初期就利用不动点来判定循环程序是否不可终止 (但基于不动点的不可终止性探测是不完备的); 另一方面, 我们设定了最大划分深度 *MaxDepth* 来保证整个划分算法的终止性. 此外, 本文方法依赖于多面体生成元的计算, 而这一计算在最糟糕情形下是指数复杂度的. 因此, 程序变量太多将增加本文算法的时间开销. 在未来工作中, 我们将一方面引入不变集的概念来加强原算法对此类循环不可终止性的探测; 另一方面, 寻找更高效的凸多面体生成元计算方法来进一步提高本文算法的效率.

References:

[1] Turing AM. On computable numbers, with an application to the Entscheidungsproblem. Proc. of the London Mathematical Society, 1937, s2-42(1): 230–265. [doi: 10.1112/plms/s2-42.1.230]

[2] Turing AM. On computable numbers, with an application to the Entscheidungsproblem—A correction. Proc. of the London Mathematical Society, 1938, s2-43(1): 544–546. [doi: 10.1112/plms/s2-43.6.544]

[3] Tiwari A. Termination of linear programs. In: Proc. of the 16th Int'l Conf. on Computer Aided Verification. Boston: Springer, 2004. 70–82. [doi: 10.1007/978-3-540-27813-9_6]

[4] Li Y. Witness to non-termination of linear programs. Theoretical Computer Science, 2017, 681: 75–100. [doi: 10.1016/j.tcs.2017.03.036]

[5] Xia BC, Zhang ZH. Termination of linear programs with nonlinear constraints. Journal of Symbolic Computation, 2010, 45(11): 1234–1249. [doi: 10.1016/j.jsc.2010.06.006]

[6] Liu J, Xu M, Zhan NJ, Zhao HJ. Discovering non-terminating inputs for multi-path polynomial programs. Journal of Systems Science and Complexity, 2014, 27(6): 1286–1304. [doi: 10.1007/s11424-014-2145-6]

[7] Chen YF, Heizmann M, Lengál O, Li Y, Tsai MH, Turrini A, Zhang LJ. Advanced automata-based algorithms for program termination checking. ACM SIGPLAN Notices, 2018, 53(4): 135–150. [doi: 10.1145/3296979.3192405]

[8] He F, Han JT. Termination analysis for evolving programs: An incremental approach by reusing certified modules. Proc. of the ACM on Programming Languages, 2020, 4(OOPSLA): 199. [doi: 10.1145/3428267]

[9] Chatterjee K, Fu HF, Goharshady AK. Termination analysis of probabilistic programs through positivstellensatz's. In: Proc. of the 28th Int'l Conf. on Computer Aided Verification. Toronto: Springer, 2016. 3–22. [doi: 10.1007/978-3-319-41528-4_1]

[10] Xue B, Zhan NJ, Li YJ, Wang QY. Robust non-termination analysis of numerical software. In: Proc. of the 4th Int'l Symp. on Dependable Software Engineering. Theories, Tools, and Applications. Beijing: Springer, 2018. 69–88. [doi: 10.1007/978-3-319-99933-3_5]

[11] Colón M, Sipma HB. Synthesis of linear ranking functions. In: Proc. of the 7th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Genova: Springer, 2001. 67–81. [doi: 10.1007/3-540-45319-9_6]

[12] Podelski A, Rybalchenko A. A complete method for the synthesis of linear ranking functions. In: Proc. of the 5th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Venice: Springer, 2004. 239–251. [doi: 10.1007/978-3-540-24622-0_20]

[13] Bagnara R, Mesnard F, Pescetti A, Zaffanella E. A new look at the automatic synthesis of linear ranking functions. Information and Computation, 2012, 215: 47–67. [doi: 10.1016/j.ic.2012.03.003]

[14] Li Y, Wu WY, Feng Y. On ranking functions for single-path linear-constraint loops. Int'l Journal on Software Tools for Technology Transfer, 2020, 22(6): 655–666. [doi: 10.1007/s10009-019-00549-9]

[15] Bradley AR, Manna Z, Sipma HB. Linear ranking with reachability. In: Proc. of the 17th Int'l Conf. on Computer Aided Verification. Edinburgh: Springer, 2005. 491–504. [doi: 10.1007/11513988_48]

[16] Bradley AR, Manna Z, Sipma HB. The polyranking principle. In: Proc. of the 32nd Int'l Colloquium on Automata, Languages and

Programming. Lisbon: Springer, 2005. 1349–1361. [doi: 10.1007/11523468_109]

[17] Alias C, Darte A, Feautrier P, Gonnord L. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: Proc. of the 17th Int'l Symp. on Static Analysis. Perpignan: Springer, 2010. 117–133. [doi: 10.1007/978-3-642-15769-1_8]

[18] Bagnara R, Mesnard F. Eventual linear ranking functions. In: Proc. of the 15th Symp. on Principles and Practice of Declarative Programming. Madrid: ACM. 2013. 229–238. [doi: 10.1145/2505879.2505884]

[19] Li Y, Zhu G, Feng Y. The *L*-depth eventual linear ranking functions for single-path linear constraint loops. In: Proc. of the 10th Int'l Symp. on Theoretical Aspects of Software Engineering. Shanghai: IEEE, 2016. 30–37. [doi: 10.1109/TASE.2016.8]

[20] Leike J, Heizmann M. Ranking templates for linear loops. Logical Methods in Computer Science, 2015, 11(1): 1–27.

[21] Ben-Amram AM, Genaim S. On multiphase-linear ranking functions. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification. Heidelberg: Springer, 2017. 601–620. [doi: 10.1007/978-3-319-63390-9_32]

[22] Yuan Y, Li Y, Shi WC. Detecting multiphase linear ranking functions for single-path linear-constraint loops. Int'l Journal on Software Tools for Technology Transfer, 2021, 23(1): 55–67. [doi: 10.1007/s10009-019-00527-1]

[23] Chen YH, Xia BC, Yang L, Zhan NJ, Zhou CC. Discovering non-linear ranking functions by solving semi-algebraic systems. In: Proc. of the 4th Int'l Colloquium on Theoretical Aspects of Computing. Macao: Springer, 2007. 34–49. [doi: 10.1007/978-3-540-75292-9_3]

[24] Cousot P. Proving program invariance and termination by parametric abstraction, Lagrangian relaxation and semidefinite programming. In: Proc. of the 6th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. Paris: Springer, 2005. 1–24. [doi: 10.1007/978-3-540-30579-8_1]

[25] Shen LY, Wu M, Yang ZF, Zeng ZB. Generating exact nonlinear ranking functions by symbolic-numeric hybrid method. Journal of Systems Science and Complexity, 2013, 26(2): 291–301. [doi: 10.1007/s11424-013-1004-1]

[26] Li Y, Sun XC, Li Y, Turrini A, Zhang LJ. Synthesizing nested ranking functions for loop programs via SVM. In: Proc. of the 21st Int'l Conf. on Formal Engineering Methods. Shenzhen: Springer, 2019. 438–454. [doi: 10.1007/978-3-030-32409-4_27]

[27] Sun XC. Proving the termination of loop programs based on SVM [MS. Thesis]. Beijing: University of Chinese Academy of Sciences, 2020 (in Chinese with English abstract).

[28] Ben-Amram AM, Genaim S. Ranking functions for linear-constraint loops. Journal of the ACM, 2014, 61(4): 1–55. [doi: 10.1145/2629488]

[29] Bagnara R, Hill PM, Zaffanella E. 2008. https://www.bugseng.com/pplhttp://www.bugseng.com/ppl

[30] Domenech JJ, Genaim S. 2018. https://github.com/costa-group/iRankFinder

附中文参考文献:

[27] 孙学超. 基于SVM算法的循环程序的终止性证明 [硕士学位论文]. 北京: 中国科学院大学, 2020.

李轶(1980－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为程序验证, 符号计算.

唐桐(1995－), 男, 硕士, 主要研究领域为程序验证, 机器学习.