软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(1):1−33 [doi: 10.13328/j.cnki.jos.007441] [CSTR: 32375.14.jos.007441]

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

向量加法系统可达性问题复杂性下界研究综述*

陈蔚骏[1], 傅育熙[2], 龙 环[2]

[1](上海交通大学 软件学院, 上海 200240)

[2](上海交通大学 计算机科学与工程系, 上海 200240)

通信作者: 傅育熙, E-mail: fu-yx@cs.sjtu.edu.cn

摘 要: 并发与可扩展性是绝大多数复杂系统的关键性质. 作为并发建模的常用语言, Petri 网被大量应用于众多领域. Petri 网的数学抽象, 即向量加法系统是计算机科学中的重要研究对象, 向量加法系统的可达性问题的算法与复杂性刻画是过去 50 年理论计算机科学中最重要的问题之一. 对向量加法系统可达性问题的复杂性下界研究进行系统而全面的总结与阐述, 主要内容包括: (1) 向量加法系统的定义、等价模型、向量加法系统的可达性问题; (2) 向量加法系统可达性问题复杂性的研究进展; (3) 固定维度的可达性问题的下界证明方法及其之间的联系; (4) 当前的研究瓶颈及有待解决的问题、未来的研究方向与挑战.

关键词: Petri 网; 向量加法系统; 可达性; 计算复杂性

中图法分类号: TP301

中文引用格式: 陈蔚骏, 傅育熙, 龙环. 向量加法系统可达性问题复杂性下界研究综述. 软件学报, 2026, 37(1): 1–33. http://www.jos.org.cn/1000-9825/7441.htm

英文引用格式: Chen WJ, Fu YX, Long H. Survey on Complexity Lower Bound Research for Reachability Problem in Vector Addition Systems. Ruan Jian Xue Bao/Journal of Software, 2026, 37(1): 1–33 (in Chinese). http://www.jos.org.cn/1000-9825/7441.htm

Survey on Complexity Lower Bound Research for Reachability Problem in Vector Addition Systems

CHEN Wei-Jun[1], FU Yu-Xi[2], LONG Huan[2]

[1](School of Software, Shanghai Jiao Tong University, Shanghai 200240, China)

[2](Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Concurrency and scalability are fundamental properties of most complex systems. As a widely used formalism for modeling concurrency, Petri nets have been applied across various fields. Their mathematical abstraction, the vector addition system (VAS), has become a central object of study in theoretical computer science. The reachability problem of VAS, along with its algorithmic and complexity characterizations, has been regarded as one of the most fundamental and long-standing challenges in the field over the last 50 years. This study presents a comprehensive survey of the research on the complexity lower bounds of the VAS reachability problem. The definitions of VAS, their equivalent models, and the core verification problem, the reachability problem, are introduced. Known completeness results concerning the complexity of the reachability problem are reviewed. For the fixed-dimension case, proof frameworks based on multiplication triples and amplifiers are outlined, and the state-of-the-art achievements as well as core proof techniques are summarized. Finally, the current research bottlenecks, major open problems, and future challenges are discussed.

Key words: Petri net; vector addition system (VAS); reachability; computational complexity

20 世纪中叶, 随着信息技术的进步, 通常的串行模型在为现实中各类复杂系统建模时常无法很好地满足可扩展的需求. 为了解决此类问题, 1962 年 Petri[1]提出了一种网络建模方法. 他放弃了全局的状态控制而是使用许多子

* 基金项目: 国家自然科学基金 (62072299); 上海市科委“科技创新行动计划” (24BC3200500, 24BC3200300)
收稿时间: 2024-07-23; 修改时间: 2024-11-02; 采用时间: 2025-03-31; jos 在线出版时间: 2025-11-03
CNKI 网络首发时间: 2025-11-04

模块配合局部操作来完成所有信息的传递和处理. 该方法指出了一条形式化研究异步系统的途径, 即如今为人所熟知的 Petri 网模型. Petri 网是一种图形化数学工具, 原始 Petri 网的描述非常简单: 一个带权重的有向二分图, 其中的两类节点分别是库所 (place) 和变迁 (transition), 库所和变迁之间的有向弧标示了有限个令牌 (token) 迁移规则. 任何时刻各库所的令牌数称为该时刻下 Petri 网的标识 (marking). 遵循迁移规则的标识变化体现了对应 Petri 网上计算的运行过程. 这一简单的模型可以用于刻画和研究各种类型数据的信息处理系统. 由于其形式简洁且建模能力强大, 除了作为描述和分析并发与分布式系统的工具以外[2,3], Petri 网理论还被广泛移植到生物、商务、化学、过程建模、软件设计、人工智能等多个领域中[4−10].

同样是出于研究并行程序与计算的目的, Karp 等人[11]于 1969 年提出了向量加法系统 (vector addition system, VAS) 这一概念, 并随后被 Hopcroft 与 Pansiot 扩展成为带状态的向量加法系统 (vector addition system with states, VASS)[12]. 尽管 Petri 网和 VASS 分别倾向于对实际问题的建模工作与对并行程序的形式化分析, 二者却是完全等价的. 两者的转换也是直接的: 用 VASS 中的向量来表示 Petri 网中的库所集合, 用向量的加法来表示 Petri 网上的变迁. 这种等价性使得实践中与 Petri 网相关的诸多问题最终都可归约为 VASS 模型上的验证问题. 相对于 Petri 网的图形化描述而言, VASS 模型在数学上更加简洁, 这在简化问题分析难度的同时也赋予了 VASS 重要的理论和应用价值. 为此本文将聚焦于讨论向量加法系统上的验证问题, 特别是其中可达性问题的下界.

对给定的模型, 形式化验证是要考察该模型是否满足某个或某些给定的性质. 一方面, 在形式化验证中, 常见的验证问题包括: 系统的正确性、安全性和可靠性等. 另一方面, 前面提到, Petri 网被广泛地应用在多个领域的建模中. 从而对领域中具体对象上的具体问题就可以用 Petri 网上的相应算法和结论. VASS 是对 Petri 网模型的进一步数学抽象. VASS 模型最主要的验证问题如下.

- 可达性问题 (reachability): 判定给定的初始格局和目标格局之间是否存在可达的路径.
- 有界性问题 (boundedness): 给定初始格局, 判定其可达的格局集合是否是有限的.
- 可覆盖性问题 (coverability): 给定初始格局和目标格局, 判定是否能够从初始格局迁移到某个比目标格局大的格局.

以上 3 个验证问题本身蕴含了很多具体、现实的算法问题. 比如通信系统的安全性可以归约到回答从初始格局出发能否到达某个“不安全”格局, 即可达性问题; 再比如计算系统的正确性, 可以归约到该系统能否达到或超越某些目标值, 其本质就是可覆盖性问题.

针对上述问题的算法复杂性研究主要涵盖如下两方面内容: 其一是上界研究, 即找到尽可能快的验证算法; 其二是下界研究, 即讨论该问题至少需要多少时间资源. 可达性、有界性和可覆盖性这 3 个问题间亦存在深刻的内在联系. 事实上, 有界性问题和可覆盖性问题都可以被有效归约到可达性问题. 换言之, 从计算复杂性的角度, 可达性问题是相关验证问题中最困难、最有挑战性的. 就研究历史而言, 早在 20 世纪 70 年代, 有界性问题与可覆盖性问题就被证明是 **EXPSPACE**-完备的[13], 但关于可达性问题的复杂性刻画却一度停滞了 40 余年, 直到在 2021 年 IEEE Symposium on Foundations of Computer Science (FOCS) 会议上, 来自华沙大学的 Czerwiński[14]和波尔多大学的 Leroux[15]两个研究团队才各自独立证明了一般 VASS 模型的可达性问题是 **Ackermann**-完备的. 即便如此, 人们对固定维度下的 VASS 以及关于维度的参数复杂性依然缺乏清晰的认识. 需要强调的是, 由于现实建模使用的 VASS 其维度往往是由具体应用所决定的, 是一个固定参数, 因此固定维加法向量模型在应用上有重要价值, 其精确的下界研究是相关形式化建模的理论基础. 相应地, 固定维加法向量模型可达性问题的下界研究也是近年理论计算机科学领域最活跃、最受关注的问题之一. 在国内, 上海交通大学的张文博等人[16]和杨启哲[17]曾做过此方面的调研, 但受当时领域内对该问题认识的限制, 在内容上前者主要围绕上界问题展开, 缺乏对下界的讨论, 后者虽有对下界的基本讨论, 但未包含最新的、在固定维模型上的突破性研究.

历年来, 相关研究者们发展并建立了许多精妙的证明技术和归约技巧, 积累了丰富的研究成果. 但这些研究工作往往比较分散, 缺乏对相关研究技术和思路的系统梳理和总结. 鉴于此, 本文对 VASS 可达性问题复杂性下界的关键证明技术进行一个全面的总结, 以期对后续研究有所启发. 特别地, 我们详细分析了若干核心技术之间的内在逻辑, 据此将近 50 年的研究成果联为一体, 并进一步揭示了这些技术的本质优势和局限. 本文旨在从更高的角度

去理解和分析相关研究, 在此基础上我们也指出了下界研究工作下一步最可能的突破方向.

本文第 1 节形式化定义向量加法系统、可达性问题及一些重要的等价模型. 第 2 节总结低维 VASS 可达性问题的完备性结论和核心研究技术. 第 3 节陈述 d-VASS 可达性问题关于维度的参数复杂性下界和相关技术. 第 4 节总结近年来固定维 VASS 可达性问题的研究进展. 第 5 节对未来研究方向进行展望和总结.

1 预备知识

本节给出加法向量系统及可达性问题的形式化定义, 定义向量加法系统的等价模型 (即计数器程序), 并介绍与本文相关的复杂性类. 注意, 本文涵盖大量数学证明, 故后续章节中使用了较多的数学符号, 其各自的含义在相关上下文中均有对应解释. 同时, 为了进一步帮助读者了解本文乃至 VASS 研究领域的符号使用惯例, 我们也在附录 A 中给出主要符号对照表, 供读者参考.

1.1 向量加法系统

本文用 $\mathbb{N}$, $\mathbb{Z}$ 分别表示自然数集与整数集, 定义 $\mathbb{N}_\omega := \mathbb{N}\cup\{\omega\}$, 其中, ω 是最小的极限序数. $\boldsymbol{u}$, $\boldsymbol{v}$ 通常用于表示向量, $d\in\mathbb{N}$ 通常用于表示向量维度, 例如, $\boldsymbol{u}\in\mathbb{N}^d$ 表示一个 d 维的自然数向量. 定义 $[d] := \{1,2,\ldots,d\}$, $[d]_0 := [d]\cup\{0\}$. 向量 $\boldsymbol{u}$ 的第 i 个元素 ($i\in[d]$) 表示为 $\boldsymbol{u}(i)$, 其无穷范数定义为 $\|\boldsymbol{u}\| = \max_{i\in[d]}|\boldsymbol{u}(i)|$. 用 $\mathbf{0}_d$ 表示全零向量, 常将下标 d 略去而直接写作 $\mathbf{0}$.

一个向量加法系统 (VAS) 是一个二元组 $V=(d,A)$, 其中, d 指定了该系统的维度, $A\subseteq\mathbb{Z}^d$ 为有限集, 表示系统的迁移规则 (transition). 系统 V 某时刻的格局 (configuration) 用 $\mathbb{N}^d$ 中的向量表示. 给定格局 $\boldsymbol{u}$, $\boldsymbol{v}$, 若存在 $\boldsymbol{a}\in A$ 使得 $\boldsymbol{u}+\boldsymbol{a}=\boldsymbol{v}$, 称 $\boldsymbol{u}$ 可通过规则 $\boldsymbol{a}$ 一步迁移到 $\boldsymbol{v}$, 记作 $\boldsymbol{u}\xrightarrow{\boldsymbol{a}}_V\boldsymbol{v}$, 在上下文清晰时也简写为 $\boldsymbol{u}\xrightarrow{\boldsymbol{a}}\boldsymbol{v}$. 有限长的迁移序列 $\boldsymbol{u}_0\xrightarrow{\boldsymbol{a}_1}\boldsymbol{u}_1\xrightarrow{\boldsymbol{a}_2}\ldots\xrightarrow{\boldsymbol{a}_n}\boldsymbol{u}_n$ 常称作运行 (run), 简记作 $\boldsymbol{u}_0\xrightarrow{\pi}\boldsymbol{u}_n$, 其中, $\pi=\boldsymbol{a}_1\ldots\boldsymbol{a}_n\in A^*$. 若有某个动作序列 π 使得 $\boldsymbol{u}_0\xrightarrow{\pi}\boldsymbol{u}_n$, 称 $\boldsymbol{u}_n$ 是从 $\boldsymbol{u}_0$ 出发可达的, 记作 $\boldsymbol{u}_0\to_V\boldsymbol{u}_n$, 此时 π 称作可达性证据 (witness).

带状态的向量加法系统 (VASS) 可用三元组 $V=(d,Q,T)$ 表示. 其中, d 指定了系统的维度, Q 为有限的状态集, 有限集 $T\subseteq Q\times\mathbb{Z}^d\times Q$ 是系统迁移规则. VASS 的格局 $\alpha=(p,\boldsymbol{u})\in Q\times\mathbb{N}^d$ 记录系统的状态和向量值, 有时也写作 $p(\boldsymbol{u})$. 给定格局 $p(\boldsymbol{u})$, $q(\boldsymbol{v})$, 若存在 $t=(p,\boldsymbol{a},q)\in T$ 使得 $\boldsymbol{u}+\boldsymbol{a}=\boldsymbol{v}$, 称 $p(\boldsymbol{u})$ 可通过规则 t 迁移到 $q(\boldsymbol{v})$, 记作 $p(\boldsymbol{u})\xrightarrow{t}q(\boldsymbol{v})$. 同理称 $\alpha_0\xrightarrow{t_1}\alpha_1\xrightarrow{t_2}\ldots\xrightarrow{t_n}\alpha_n$ 为 VASS 的一次运行, 并简写作 $\alpha_0\xrightarrow{\pi}\alpha_n$, 其中, $\pi=t_1\ldots t_n\in T^*$. 用 $\alpha_0\to_V\alpha_n$ 表示 V 中 α_n 从 α_0 出发可达. (带状态的) d 维向量加法系统常记作 d-VAS (d-VASS).

例 1: 给定非负整数 k, 定义 2-VAS $V_1=(2,\{\boldsymbol{a}_1,\boldsymbol{a}_2\})$, 其中, $\boldsymbol{a}_1=(-1,1)$, $\boldsymbol{a}_2=(k,-1)$. 给定格局 $\boldsymbol{u}=(m,0)$, 容易验证 $\boldsymbol{u}\to_{V_1}(km,0)$, 对应的证据为 $\pi=\boldsymbol{a}_1^m\boldsymbol{a}_2^m$. 需注意, $\boldsymbol{u}\xrightarrow{\boldsymbol{a}_2}(m+k,-1)$ 不是一个合法迁移, 因为所有格局都必须是 $\mathbb{N}^2$ 中的向量. 本例说明, 2 维 VAS 可完成数乘计算.

例 2: 定义 3-VASS $V_2=(3,\{p,q\},\{t_1,t_2,t_3,t_4\})$, 其中,

$$t_1=(p,(-1,1,0),p),\ t_2=(p,(0,0,0),q),\ t_3=(q,(k,-1,0),q),\ t_4=(q,(0,0,-1),p).$$

为了更直观地表示迁移关系, 我们将 V_2 表示为图 1 中带标签的多重有向图. 给定初始格局 $\alpha_0=p(1,0,n)$, 考虑形如 $\pi_m=t_1^mt_2t_3^mt_4$ 的规则序列, 易知从 α_0 出发有如下运行:

$$\alpha_0\xrightarrow{\pi_{k^0}}p(k,0,n-1)\xrightarrow{\pi_{k^1}}p\left(k^2,0,n-2\right)\xrightarrow{\pi_{k^2}}\ldots\xrightarrow{\pi_{k^{n-1}}}p(k^n,0,0).$$

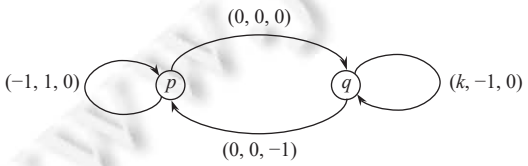


图 1 V_2 的有向图表示

此 VASS 是取自文献 [12] 的经典例子, 说明了 VASS 在可达性意义下能完成指数运算.

显然 VAS 可被看作状态集 Q 是单元集的特殊 VASS. 因此所有 VAS 的行为都可以使用一个对应的 VASS 来

模拟. 关于 VAS 与 VASS 两大模型的关系, Hopcroft 等人[12]在 1979 年证明了如下命题.

命题 1[12]. 任意 d-VASS 都可以被一个 $(d+3)$-VAS 模拟.

此命题的证明思路是为原来 VAS 中的向量额外增加 3 个维度用于编码状态信息, 构造过程是高效的. 基于此命题, 后文将集中讨论 VASS 上的可达性问题.

1.2 VASS 可达性问题

一般 VASS 可达性问题的形式化定义如下.

一般 VASS 可达性问题 (general VASS reachability)

输入: 一个 VASS $V=(d,Q,T)$, 两个格局 $p(\boldsymbol{u}),q(\boldsymbol{v})\in Q\times\mathbb{N}^d$;
输出: 是否有 $p(\boldsymbol{u})\to_V q(\boldsymbol{v})$?

此处“一般 (general)”指的是不对输入的 VASS 模型的维度加任何限制. 此问题的复杂性已有完备性结论. 出于应用及理论两方面的动机, 目前本领域主要聚焦于固定维度 VASS 的可达性问题, 其表述如下.

---

d-VASS 可达性问题 (d-VASS reachability)

输入: 一个 d-VASS $V=(d,Q,T)$, 两个格局 $p(\boldsymbol{u}),q(\boldsymbol{v})\in Q\times\mathbb{N}^d$;
输出: 是否有 $p(\boldsymbol{u})\to_V q(\boldsymbol{v})$?

注意, 此处 d 是一个固定的参数. 例如研究 2-VASS 可达性问题时, 所有输入的 VASS 实例都必须是 2 维的.

可以进一步限制初始和目标格局的向量都为 0, 从而将以上两个问题的表述简化为: 仅输入 $V=(d,Q,T)$ 与状态 $p,q\in Q$, 判断是否有 $p(0)\to_V q(0)$. 显然这是一般可达性问题的特例, 但其复杂性与原问题相等: 给定一个原可达性问题实例 V、$p(\boldsymbol{u})$、$q(\boldsymbol{v})$, 我们可以高效地构造 $V'=(d,Q\cup\{q_s,q_f\},T')$, 其中, q_s、q_f 是不在 Q 中的新状态, 且 $T'=T\cup\{(q_s,\boldsymbol{u},p),(q,-\boldsymbol{v},q_f)\}$. 这使得 $p(\boldsymbol{u})\to_V q(\boldsymbol{v})$ 当且仅当 $q_s(0)\to_{V'} q_f(0)$. 为此, 在讨论 VASS 可达性问题复杂性下界时, 我们约定使用简化版的表述.

研究可达性问题的复杂性时, 我们需要分析算法运行时间或空间与问题规模的关系. 这需要严格定义 VASS 向量常用的两类编码方案, 它们将直接影响复杂性结论. 给定 $V=(d,Q,T)$, 令 $|Q|$、$|T|$ 分别表示 Q、T 集合的大小. 对于 $t=(p,\boldsymbol{a},q)\in T$, 定义它的范数为其中向量的范数, 即 $\|t\|:=\|\boldsymbol{a}\|$, 同时定义 $\|T\|:=\max\limits_{t\in T}\|t\|$. 根据不同进制, VASS V 的规模分别表示如下.

- 在一进制编码 (unary encoding) 下, V 的规模 $|V|_1:=|Q|+d\cdot|T|\cdot\|T\|$.
- 在二进制编码 (binary encoding) 下, V 的规模 $|V|_2:=|Q|+d\cdot|T|\cdot\lceil\log(\|T\|+1)\rceil$.

直观上, 二进制编码在各类算法研究中都是常用方案; 而在一进制编码下 VASS 可以在多项式时间内转换为对应的迁移规则只允许对某一维度 +1 或 −1 的受限 VASS, 后者则与自动机 (automaton) 联系紧密[18], 因此其也是研究的主要对象之一. 以例 2 中的 V_2 为例, 有 $|Q|=2$, $|T|=4$, $\|T\|=k$, 从而 $|V_2|_1=12k+2$, $|V_2|_2=12\lceil\log(k+1)\rceil+2$. 由此可见, 同一个 VASS 在从二进制转换为一进制编码后长度有一个指数量级的放大. 对低维度 VASS 而言这样的放大将会对复杂性产生较大影响, 特别是导致二进制编码下的可达性问题严格难于一进制编码下的可达性问题; 而高维度和一般 VASS 的复杂性远高于指数复杂性谱系, 此时编码方式对复杂性没有影响, 不必再加以区分.

1.3 计数器自动机、计数器程序

除了 VASS 和 Petri 网模型外, 在理论研究中人们也提出了诸多和 VASS 等价的、更方便研究的模型. 本节介绍最常用的一种, 其特点是可读性强. 其他一些等价模型将会在相关章节再分别讨论.

计数器机器 (counter machine, CM) 或计数器自动机 (counter automaton, CA) 本质上是一个指令的有限序列 $P=(I_1,\ldots,I_n)$, 其中, $I_1,\ldots,I_n$ 只能为下面 5 类指令 (可带标签).

1) 加法: “x += a;”. 这里的 a 指代某一常量, x 是某个变量, 一般称为计数器.

2) 减法: "$x{-}{=}a$;". 同上.

3) 非确定跳转: "**goto** L (or L' or …);". 其中, L, L' 等是有限个指令的标签.

4) 测零: "**zero**? $x,y,z,\ldots$;" 对单个计数器 x 或一组计数器"$x,y,z,\ldots$"测零.

5) 停机: "**halt**;". 表示程序运行至此停机, 只在程序末出现一次.

若某计数器机器 $P=(I_1,\ldots,I_n)$ 使用的计数器集合为 X, 则它的格局形如 $\alpha=I(\boldsymbol{u})$, 描述了下一条执行的指令 I 和当前所有计数器的取值函数 $\boldsymbol{u}:X\to\mathbb{N}$ (本质上也是一个有限维向量). 给定计数器 $x\in X$, α 中 x 的取值可直接表示为 $\alpha(x):=\boldsymbol{u}(x)$, 这种写法还可以拓展到函数 $f(x_1,\ldots,x_n)$ 上, 定义 $\alpha(f):=f(\alpha(x_1),\ldots,\alpha(x_n))$. 计数器机器的语义如下: 初始时 $\alpha_0=I_1(\mathbf{0}_{|X|})$, 顺序执行 P 中指令; 若某次减法操作对应的计数器值 <0 或测零时指定的计数器值不为 0, 则终止执行; 若遇到 **goto** 语句, 则非确定地选择一个分支执行. 最终得到一个极大的有限长的迁移序列 $R:\alpha_0\xrightarrow{I_{i_1}}\alpha_1\ldots\xrightarrow{I_{i_k}}\alpha_k=I_{i_{k+1}}(u)\nrightarrow$, 称作 P 的一次执行 (execution), 若 $I_{i_{k+1}}=$ **halt**, 则称 R 为完全的 (complete), 记作 R: $\alpha_0\to_P\alpha_k$; 否则称其为部分的 (partial). 通常我们只会针对程序 (或片段) 的某个执行讨论其初始格局和终止格局, 此时对应指令已被隐式地确定, 后文中均直接省略. 不包含任何测零指令的计数器机器又称作计数器程序 (counter program, CP).

若某次完全执行的终止格局为 **0**, 则称这是一个 **0**-执行. 可以在计数器程序模型上定义可达性问题 (CP reachability): 给定某计数器程序 P, 判定其是否存在 **0**-执行. 类比 d-VASS 的概念, 我们将使用 d 个计数器的机器 (程序) 分别称作 d-计数器机器 (程序), 在这些模型上同样可以类似地定义对应的可达性问题. 下面的命题阐述了 d-VASS 与 d-计数器程序的等价性.

命题 2. d-VASS 可达性问题与 d-计数器程序可达性问题可以在对数空间内互相归约.

此命题的证明较为简单: 直接为 d-VASS 构造对应的 d-计数器程序, 反之亦然. 例如, 例 2 中的 V_2 可以通过此方法转换为图 2 左侧的计数器程序 P_1 使得 $p(1,0,n)\to_{V_2}p(k^n,0,0)$ 当且仅当 P_1 存在 **0**-执行.

```
      a +=1;   c += n;
p:    goto t1 or t2 or end;
t1:   a -=1;   b +=1;  goto p;
t2:   goto q;
t3:   a += k;  b -=1;  goto q;
t4:   c -=1;   goto p;
q:    goto t3 or t4;
end:  a -= k^n;  halt;
```

P_1

```
a +=1;   c += n;
loop
    loop a -=1,   b +=1;
    loop a += k,  b -=1;
    c -=1;
a -= k^n;
halt;
```

P_2

图 2 V_2 的等价计数器程序

为了提高程序的易读性, 文献中常引入一些语法糖.

● **loop** 指令: 非确定地将循环体重复执行若干次. 它可以使用 **goto** 语句实现, 因此并不影响计数器机器或程序的计算能力. 使用这一语法糖, 我们可以将 P_1 改写成图 2 右侧的程序 P_2.

● "**do** s_1 **or** s_2;" 语句: 非确定地执行语句 s_1 或 s_2, 这里, s_1, s_2 可以是 **skip**, 即什么操作都不做.

● "**for** i:= 1 **to** n M;" 语句: 将 M 重复 n 次, 本质上这是一段 $O(n|M|)$ 长的程序.

最后, VASS 还有一种带测零操作的变体, 它们的迁移规则集合中允许形如 $t=(p,\textbf{zero-test}_i,q)$ 的规则. 给定格局 $p(\boldsymbol{u})$, $q(\boldsymbol{v})$, $p(\boldsymbol{u})\xrightarrow{t}q(\boldsymbol{v})$ 当且仅当 $\boldsymbol{v}=\boldsymbol{u}$ 且 $\boldsymbol{u}(i)=0$. 显然, 此模型和计数器机器是等价的.

1.4 归约、复杂性类、完备性

本节主要介绍后文中涉及的复杂性理论的相关前置知识.

1.4.1 归约与完备性

我们假定读者对图灵机 (Turing machine)、时间函数、空间函数等传统复杂性理论的重要概念均有基础的了解, 本文仅做一些记号上的约定. 给定时间函数 $t(n)$, 我们用 TIME$(t(n))$ 和 NTIME$(t(n))$ 分别表示为所有确定、非确定图灵机在 $O(t(n))$ 时间内可判定的判定问题集合, FTIME$(t(n))$ 和 FNTIME$(t(n))$ 则分别对应它们的计算版本, 即所有确定 (非确定) 图灵机在 $O(t(n))$ 时间内可计算的函数集合. 给定空间函数 $s(n)$, 我们用 SPACE$(s(n))$ 和

NSPACE($s(n)$) 表示所有确定、非确定图灵机在 O($s(n)$) 空间内可判定的问题集合. 在此基础上, 一些经典的复杂性类定义如下:

$$\mathbf{NL}=\mathrm{NSPACE}(\log n),\ \mathbf{P}=\bigcup_{c\in\mathbb{N}}\mathrm{TIME}(n^c),\ \mathbf{NP}=\bigcup_{c\in\mathbb{N}}\mathrm{NTIME}(n^c),\ \mathbf{PSPACE}=\bigcup_{c\in\mathbb{N}}\mathrm{SPACE}(n^c) \tag{1}$$

若考虑形如 $k\text{-}\exp(n):=2^{\cdot^{\cdot^{2^n}}}\}k$ times 的函数, 则还可以定义如下的复杂性谱系:

$$k\text{-}\mathbf{EXP}=\mathrm{TIME}(k\text{-}\exp(n^c)),\ k\text{-}\mathbf{EXPSPACE}=\bigcup_{c\in\mathbb{N}}\mathrm{SPACE}(k\text{-}\exp(n^c)) \tag{2}$$

在复杂性理论中, 我们通常用多项式时间归约 (reduction) 比较问题的相对难度, 其形式化定义如下.

定义 1. 给定问题 A,B, 若存在一个多项式时间可计算函数 (polynomial-time computable function) f 使得 $x\in A$ 当且仅当 $f(x)\in B$, 则称 A 能多项式时间归约到 B, 记作 $A\leqslant_{\mathrm{p}}B$.

将归约函数限制为多项式时间可计算函数可以保证构造过程的高效性, 但有时我们也会考虑其他形式的归约函数, 届时将以不同的下标加以区别, 例如 $A\leqslant_{\mathrm{L}}B$ 表示可以使用一个对数空间可计算函数将问题 A 归约到问题 B. 在给定某归约 $\leqslant$ 的基础上, 我们给出完备性 (completeness) 的定义.

定义 2. 给定复杂性类 C 和问题 A, 若对任意 $B\in C$, 都有 $B\leqslant A$, 则称问题 A 是 C-难的 (C-hard); 若进一步有 $A\in C$, 则称问题 A 是 C-完备的 (C-complete).

直观上, 完备问题刻画了一个复杂性类中“最难的问题”. 一般而言, 定义完备性时需要指定归约的类型: 在定义 **P**、**NP**、**PSPACE**、k-**EXP**、k-**EXPSPACE** 的完备问题时通常使用多项式时间归约 $\leqslant_{\mathrm{p}}$, 在定义 **NL**-完备问题时则使用对数空间归约 $\leqslant_{\mathrm{L}}$. 若问题 $A\in C$, 我们称 C 是 A 的复杂性上界; 若 A 是 C-难的, 我们称 C 是 A 的复杂性下界. 由于本文主要聚焦于 VASS 的复杂性下界, 因此会涉及较多归约相关的证明. 值得注意的是, 复杂性下界具有传递性: 若问题 A 是 C-难 (包括 C-完备), 且在对应归约下 $A\leqslant B$, 则问题 B 也是 C-难. 因此, 我们无须考虑 C 中的所有问题, 只需从 C-难或 C-完备问题出发进行归约即可.

下面列出几个经典复杂性类上的完备问题, 便于后文归约.

- 子集和问题. 子集和问题是经典的 **NP**-完备问题, 其定义如下.

子集和问题 (subset sum)

输入: 集合 $S=\{a_1,\ldots,a_n\}\subseteq\mathbb{N}$, 目标 $c\in\mathbb{N}$;
输出: 是否存在子集 $T\subseteq S$ 使得 $\sum_{a\in T}a=c$?

- 子集和博弈 (subset sum game) 问题. 子集和博弈问题本质上是一个双人博弈: 两个玩家 ∀, ∃ 依次选数, ∃ 玩家的目标是使得所选数之和为某个给定的自然数 C. 问题是: ∃ 玩家是否有必胜策略? 该问题形式化描述如下.

子集和博弈问题 (subset sum game)

输入: 自然数 $A_1,B_1,E_1,F_1,\ldots,A_n,B_n,E_n,F_n,C\in\mathbb{N}$;
输出: $\forall x_1\in\{A_1,B_1\}\exists y_1\in\{E_1,F_1\}\ldots\forall x_n\in\{A_n,B_n\}\exists y_n\in\{E_n,F_n\}\sum_{i=1}^{n}(x_i+y_i)=C$. 是否为真?

Travers 证明了 **PSPACE**-完备问题 TQBF $\leqslant_{\mathrm{L}}$ subset sum game, 同时该问题自身也在 **PSPACE** 中, 从而有如下引理.

引理 1[19]. 子集和博弈问题是 **PSPACE**-完备的.

- 图灵机 $s(n)$-空间接收问题 (($s(n)$)-space TMSAT). 这是一类平凡的问题, 其难度与 $s(n)$ 相关.

图灵机 $s(n)$-空间接收问题

输入: 图灵机 M, 输入 x, 参数 1^n;
输出: M 在输入 x 上是否能够在 $s(n)$ 空间内接收?

任何 $\bigcup_{c\in\mathbb{N}}\mathrm{SPACE}(s(n^c))$ 中的问题 L 都可以多项式时间归约到对应的图灵机 $s(n)$-空间接收问题, 这说明了图灵机 $s(n)$-空间接收问题在多项式归约下是 $\bigcup_{c\in\mathbb{N}}\mathrm{SPACE}(s(n^c))$-难的. 若 $s(n)$ 是空间可构造的, 图灵机 $s(n)$-有界问题还有一个变体, 即 $s(n)$-有界图灵机问题 ($s(n)$-Bounded Automata): 给定输入 $\langle M,x,1^n\rangle$, 其中, M 是一个能且仅能使用 $s(n)$ 空间的图灵机, 判定 M 在输入 x 上是否能输出 1. 该问题同样是 $\bigcup_{c\in\mathbb{N}}\mathrm{SPACE}(s(n^c))$-难的. 例如, 当 $s(n)=n$ 时, 该问题即是著名的 **PSPACE**-完备问题: 线性有界自动机接受问题 (LBA).

● $f(n)$-有界计数器机器可达性问题. 此问题为 $s(n)$-有界图灵机问题在计数器机器中的推广. 易知, 3 个计数器可以模拟一个图灵机: 其中, 两个计数器分别以数的形式记录图灵机磁头左右两端的, 剩下一个用于辅助模拟图灵机的迁移. 特别的对于一个 $s(n)$-有界图灵机, 模拟过程中计数器的值永远不会超过 $c^{s(n)}$, 常数 c 和图灵机使用的符号表有关. 常把执行过程中所有计数器之和不超过 $f(n)$ 的计数器机器称作 $f(n)$-有界的 ($f(n)$-bounded). 故可以使用 3 个计数器 x、y、z 的 $c^{s(n)}$-有界计数器机器 M' 模拟一个 $s(n)$-有界图灵机 M. 不妨设终止后 a 给出了 M 的输出, 令 $a-$=1, 其他计数器循环递减, 最终执行"**zero**? a, b, c;", 则 M 在 x 上输出 1 当且仅当 M' 存在完全执行. 定义如下.

$f(n)$-有界计数器机器可达性问题 ($f(n)$-BCM reachability)

输入: 1^n 和仅使用 3 个计数器的 $f(n)$-有界计数器机器 M;
输出: M 是否存在完全执行?

上述讨论中令 $s(n)=2^n$, $f(n)=2^{2^n}$, 可以得到如下引理.

引理 2. 2^{2^n}-有界计数器机器可达性问题是 **EXPSPACE**-难的.

此外, 图灵机 $s(n)$-空间接收问题也可以推广到计数器机器模型上: 给定参数 n, 函数 $f(n)$ 和计数器机器 M, 若 M 的某完全执行中所有格局的计数器值之和不超过 $f(n)$, 则此次执行称为 $f(n)$-有界执行. 定义如下计数器机器 $f(n)$-有界可达性问题 (CM $f(n)$-bounded reachability).

计数器机器 $f(n)$-有界可达性问题

输入: 1^n 和仅使用 3 个计数器的计数器机器 M;
输出: M 是否存在 $f(n)$-有界 **0**-执行?

该问题难度与图灵机 $s(n)$-空间接收问题一致.

● 计数器机器 $f(n)$-时间可达性问题 (CM $f(n)$-reachability). 给定参数 n 后, 若计数器机器 M 的某次执行恰好使用了 $f(n)$ 次测零语句, 我们将此次执行称作 $f(n)$-时间执行, 定义如下新问题.

计数器机器 $f(n)$-时间可达性问题

输入: 1^n 和仅使用 3 个计数器的计数器机器 M;
输出: M 是否存在 $f(n)$-时间执行?

计数器机器 $f(n)$-时间可达性问题和 $f(n)$-有界可达性问题分别对应图灵机的时间和空间复杂性. 类似地, 我们可以证明该问题在多项式归约下是 $\bigcup_{c\in\mathbb{N}}\mathrm{TIME}(f(n^c))$-难的.

1.4.2 快速增长复杂性类

回顾 VASS 的研究历史, 可达性问题之所以在很长一段时间处于开放状态, 其难度不仅在于算法设计、问题归约等方面, 还在于对基本复杂度类认识的不足. 事实上, 经典的算法问题一般位于人们认识比较充分的复杂性类, 如 **NL**、**P** 等; 其次是指数时间复杂性谱系 k-**EXP** 以及初等复杂性类 **ELEM** = $\bigcup_{k\in\mathbb{N}}k$-**EXP**. 过去对 VASS 的研究工作表明, 可达性问题不太可能位于指数时间谱系中[14]. 然而 **ELEM** 没有完备问题, 无法直接进行归约; 人们又对该谱系之外的问题和复杂性类缺乏了解. 为了解决这一状况, Schmitz[20]提出了一族基于序数 (ordinal) 的快速增长函数 $\{F_t\}_t$ 和对应的复杂性谱系 $\{\mathbf{F}_t\}_t$. 该谱系成为研究 d-VASS 可达性下界的基础. 本节简单给出这类函数和谱系

的形式化定义.

由于已知一般 VASS 可达性问题是 **Ackermann**-完备的, 这里只给出基于 $\mathbb{N}_\omega$ 的快速增长函数 (fast-growing function) $\{F_\iota\}_{\iota\in\mathbb{N}_\omega}$. 首先对于自然数 d, 归纳定义 $F_d:\mathbb{N}\to\mathbb{N}$ 如下:

$$F_0(x)=x+1 \tag{3}$$

$$F_{d+1}(x)=F_d^{(x+1)}(x) \tag{4}$$

其中, $f^{(k)}$ 表示将函数 f 复合 k 次. 对于极限序数 ω, 定义 $F_\omega:\mathbb{N}\to\mathbb{N}$ 为 $F_\omega(x)=F_{x+1}(x)$. 例如, $F_1(x)=2x+1$; $F_2(x)=2^{x+1}(x+1)-1$ 是一个指数量级的函数; $F_3(x)$ 的表达式已经十分复杂了, 是一个达到 $\mathrm{tower}(x)=2^{\cdot^{\cdot^{\cdot^{2^x}}}}\Big\}x\text{ times}$ 量级的非初等函数 (elementary function); $F_\omega(x)$ 是非原始递归 (primitive-recursive function) 的 Ackermann 函数.

在上述定义的基础上, 可以定义如下两种函数类:

$$\mathcal{F}_\iota=\bigcup_{c\in\mathbb{N}}\mathrm{FTIME}(F_\iota^{(c)}(n)),\ \mathcal{F}_{<\iota}=\bigcup_{\iota'<\iota}\mathcal{F}_{\iota'} \tag{5}$$

易知, $\mathcal{F}_1=\mathcal{F}_0$ 是线性时间可计算的函数; $\mathcal{F}_2=\mathcal{F}_{<3}$ 表示初等函数类 **FELEM** (前文中的 **ELEM** 是对应的判定问题类); $\mathcal{F}_{<\omega}$ 表示原始递归函数类 **FPR**. 此谱系 $\{\mathcal{F}_\iota\}_\iota$ 一般称为扩展 Grzegorczyk 谱系 (extended Grzegorczyk hierarchy), 由于谱系中的层级均无对应的完备问题, 它依然不是刻画复杂性的良好选择, 仅作为快速增长复杂性谱系的中间定义引入.

Schmitz 的关键贡献在于定义了快速增长复杂性谱系 $\{\mathbf{F}_\iota\}_\iota$, 其中, $\mathbf{F}_\iota=\bigcup_{p\in\mathcal{F}_{<\iota}}\mathrm{TIME}(F_\iota(p(n)))$. 函数 p 在谱系 $\{\mathcal{F}_\iota\}_\iota$ 中层级低于 ι. 注意, 当 $\iota\geqslant 3$ 时将定义中的 TIME 替换成 NTIME 或 SPACE 不会改变所定义的复杂性类, 因为此时 p 已覆盖所有初等函数, 而对应选项之间的转换却只涉及至多指数量级的放大. 一般令 **Tower** $=\mathbf{F}_3$, **Ackermann** $=\mathbf{F}_\omega$, 这也解释了前文中相关记号的含义. 引入复杂性类 $\{\mathbf{F}_\iota\}_\iota$ 的主要原因是这些类均有完备问题. 在定义 $\mathbf{F}_\iota$-难问题时通常使用 $\mathcal{F}_{<\iota}$ 时间归约, 结合第 1.4.1 节的内容, 有如下引理.

引理 3. 当 $\iota\geqslant 3$ 时, 计数器机器 $\iota(n)$-有界可达性问题、计数器机器 $F_\iota(n)$-时间可达性问题是 $\mathbf{F}_\iota$-难的.

当 $\iota\geqslant 3$ 时, 可以将上述问题的输入 M 限定为只使用 2 个计数器的机器, 因为我们可以将原来的 x, y 编码为 2^x3^y, 这是一个 $2^{2^{F_\iota(n)}}$ 量级的数值. 因此图灵机 $F_\iota(n)$-空间接收问题可以初等归约到计数器机器 $2^{2^{F_\iota(n)}}$-有界可达性问题上, 且 $2^{2^{F_\iota(n)}}$ 远小于 $F_\iota(\mathrm{poly}(n))$. 计数器机器 $F_\iota(n)$-时间可达性同理, 故有如下推论.

推论 1. 当 $\iota\geqslant 3$ 时, 只使用 2 个计数器的计数器机器 $F_\iota(n)$-有界可达性问题、计数器机器 $F_\iota(n)$-时间可达性问题也均为 $\mathbf{F}_\iota$-难的.

2 1-VASS 和 2-VASS 的完备性结论

本节梳理加法向量模型上已有的完备性结论, 重点聚焦于下界证明技术. 需要指出的是, 迄今 VASS 模型上关于可达性问题复杂性的完备性结论屈指可数: 除了前文提到的 **Ackermann**-完备性, 只有关于 1 维 VASS 和 2 维 VASS 的结论. 表 1 总结了已有的完备性结论.

表 1 VASS 可达性问题的完备性结论

编码方案	1-VASS	2-VASS	一般情形
一进制编码	**NL**-完备	**NL**-完备	**Ackermann**-完备
二进制编码	**NP**-完备	**PSPACE**-完备	

关于一般 VASS 可达性问题 (**Ackermann**-完备) 可以基于维度得到更精细的参数复杂性结果, 更多细节将在第 3 节讨论. 参考第 1.2 节的分类方式, 我们按一进制和二进制编码方案对 1 维 VASS 和 2 维 VASS 的可达性问题复杂性分别进行讨论.

2.1 一进制编码下的可达性问题

1 维 VASS 和 2 维 VASS 可达性下界相对容易. 例 2 中给出了如何将 VASS 转换为带向量标签的多重图. 事实上传统有向图可以看作是 0-VASS 模型实例 (即只有状态迁移关系); 类似地, d-VASS 模型亦可看作 (d+1)-VASS 模型的实例. 如前所述, 我们用 $\leqslant_{\mathrm{L}}$、$\leqslant_{\mathrm{p}}$ 分别表示对数空间归约和多项式时间归约, 用 PATH 表示图上的可达性问题. 则有: PATH $\leqslant_{\mathrm{L}}$ 0-VASS reachability $\leqslant_{\mathrm{L}}$ 1-VASS reachability $\leqslant_{\mathrm{L}}$ 2-VASS reachability. 其中, PATH 是著名的 **NL**-完备问题, 它给出了这些问题的下界.

在上界方面, Blondin 等人[21]证明了如下引理.

引理 4[21]. 对任意 2-VASS $V=(2,Q,T)$, 若 $p(\boldsymbol{u})\to_V q(\boldsymbol{v})$, 则存在长度为 $(|V|_1+\boldsymbol{u}+\boldsymbol{v})^{\mathrm{O}(1)}$ 的证据 $\pi\in T^*$ 使得 $p(\boldsymbol{u})\xrightarrow{\pi} q(\boldsymbol{v})$.

此引理表明在 2-VASS V 中, 若从初始状态 $p(\boldsymbol{u})$ 可以到达某个状态 $q(\boldsymbol{v})$, 则一定存在一个“短”的可达证据 π, 其长度是关于 V、$\boldsymbol{u}$、$\boldsymbol{v}$ 的一进制编码长度的多项式. 这个结论使得我们可以用非确定图灵机猜测一条长度受限的运行, 并检验该路径是否是合法的可达路径, 并且此过程至多需要使用对数量级的空间. 因此一进制编码下的 2-VASS 可达性问题在 **NL** 中. 这也是 Blondin 等人在该项工作中的主要贡献. 这些结论总结如下.

定理 1[18]. 一进制编码下, 1-VASS 和 2-VASS 的可达性问题都是 **NL**-完备的.

2.2 二进制编码下的可达性问题

二进制编码下, 低维 VASS 可达性证据长度及用于归约的问题如表 2 所示. 其中, 1-VASS 的证据长度数据来自文献 [22], 2-VASS 的证据长度数据来自文献 [21].

表 2 二进制编码下 ($\leqslant 2$)-VASS 可达性证据长度及用于归约的问题

关键算法性质	(普通或测零) 1-VASS	2-VASS
证据长度	$2^{\mathrm{poly}(n)}$	$2^{\mathrm{poly}(n)}$
归约问题	子集和问题	子集和博弈

2.2.1 上界证明技术

此部分完备性结论的上界证明较为复杂精细, 由于本文主要聚焦于 VASS 可达性问题复杂性下界, 故在此仅对证明思路做简要概述, 读者可以参考原文[23]及一些综述[16]进一步了解相关证明细节.

引理 4 本质上是一个与编码方案无关的命题. 若我们将 2-VASS 可达性问题中的所有向量编码为二进制, 则证据 π 的长度是问题输入规模的指数量级. 同理使用非确定图灵机逐步猜测-验证方法, 可证该编码方案下 2-VASS 可达性问题在 **PSPACE** 中. 然而, 这一简单的思路不适用于 1-VASS 的可达问题上界的推导. 这是因为 1-VASS 并不能得到比指数更好的可达证据长度上界. 要证明该问题在 **NP** 中, 需要寻找更简洁的证据表示方法. Haase 等人的思路[23]是将 VASS 看作流图 (flow graph) 并使用 Parikh 像 (Parikh image) 来表示一条路径. 该方法仅记录每条边出现的次数 (而不记录具体的出现次序). 要注意的是, 一个满足欧拉条件的 Parikh 像可以导出一段迁移序列, 但并不保证一定能导出一条“合法”的 VASS 的运行: 还需要判断该过程中向量值是否始终落在自然数集合 $\mathbb{N}$ 中. 仅有满足某些特定条件的 Parikh 像可以高效地验证是否存在一条对应的证据, 这类 Parikh 像被称作可达证书 (reachability certificate).

直观上, 1-VASS 有独特的算法性质: 向量仅有一个维度, 这使得各类操作无须考虑维度之间的相互影响. Haase 等人[23]借此证明了 1-VASS 具有可泵性 (pumpability), 并进一步说明每个可达的问题实例都存在形如 $\pi_1\circ\pi_2\circ\pi_3$ 的证据. 其中, 各 π_i 的长度至多为指数, 并均可以通过可达证书表征. 因此最终非确定算法只需猜测 3 段对应的可达证书并进行高效验证即可, 这证明了 1-VASS 可达性问题在 **NP** 中. 值得注意的是, Haase 等人的证明是基于 1-计数器机器进行的. 而由抽屉原理, 测零 1-VASS 中一次没有重复格局的运行至多进行 $|Q|$ 次测零. 即测零 1-VASS 可达性问题可以转换为线性个 (无测零) 1-VASS 的可达性问题. 故在只有一个计数器的情形下, 能否测零不影响可达性问题的复杂性.

2.2.2 二进制编码下 1-VASS 可达性问题的 **NP**-难归约

本节介绍下述引理的证明技巧, 同时解释了为何引理中二进制编码这一条件是必要的.

引理 5[23]. 二进制编码下的 1-VASS 可达性问题是 **NP**-难的.

证明: 思路是归约经典 **NP**-完备问题“子集和问题”, 其定义参见第 1.4.1 节. 子集和问题中所有自然数用二进制编码. 文献 [19] 中给出了从子集和问题到 1-VASS 可达性问题的对数空间归约. 这里我们展示一种更便捷的归约到 1-计数器程序的构造方法.

给定子集和问题实例 $\langle S,c\rangle$, 构造对应的程序 $f(\langle S,c\rangle)$ 如图 3 所示.

```
for i := 1 to n
    do x += a_i; or skip;
x -= c; halt;
```

图 3 从子集和问题到 1-计数器程序的归约 P

程序中的 $a_1, a_2, \ldots, a_n$, c 即原子集和问题中的参数. 注意, 为了保证归约的高效性, 此构造中 1-计数器程序也应采用二进制编码方案 (否则原问题中的 a_i 转换为对应的一进制编码需要花费指数时间).

P 仅使用了一个计数器 x, 故是一个 1-计数器程序. 该程序的任意一次执行等价于非确定选择一个子集 $T\subseteq S$, 若 $\sum_{a\in T}a-c\geqslant 0$ 则该执行是完全的且终止格局 β_T 满足 $\beta_T(x)=\sum_{a\in T}a-c$, 否则该执行是部分的. 显然, 若实例 $\langle S,c\rangle$ 为真, 则存在子集 $T_0\subseteq S$ 使得 $\sum_{a\in T_0}a-c=0$, 程序中对应的执行路径是完全的且最终 $\beta_{T_0}(x)=0$, 即 P 存在一个 **0**-执行; 若该问题实例为假, 则所有子集 $T\subseteq S$ 都有 $\sum_{a\in T}a-c\neq 0$, 即 P 不存在 **0**-执行. 综上, $\langle S,c\rangle\in$ subset sum 当且仅当 $f(\langle S,c\rangle)\in$ 1-CP reachability. 此程序的长度为 $\mathrm{O}\left(\sum_{i=1}^{n}\log(a_i)+\log(c)\right)$, 归约 f 可在对数空间完成, 即有 subset sum$\leqslant_{\mathrm{L}}$1-CP reachability. 根据计数器程序和 VASS 的等价性即知: 1-VASS 可达性问题是 **NP**-难的.

需要指出的是, 经典的 **NP**-完备问题有很多, 但子集和问题与 VASS 模型都与自然数以及加减法运算有关, 二者之间的关联较为紧密, 因此归约较简单. 结合第 2.2.1 节给出的上界可得如下定理.

定理 2[19]. 二进制编码下普通和测零 1-VASS 可达性问题都是 **NP**-完备的.

2.2.3 二进制编码下 2-VASS 可达性问题的 **PSPACE**-难归约

正如子集和问题对应于 1-VASS 可达性问题的复杂性下界, 2-VASS 的 **PSPACE**-难证明源自归约子集和博弈问题[19]. 然而子集和博弈到 2-VASS 可达性问题的归约是非平凡的, 主要困难在于对 $\forall$ 玩家, 需要使用仅有的两个计数器模拟其所有可能选择的分支. 为了完成归约, 研究者引入了两个中间模型, 即有界计数器自动机 (bounded counter automaton, BCA) 和计数器栈自动机 (counter-stack automaton, CSA). 这些模型都有各自等价的数学描述和程序表示, 后续会根据具体情况选择合适的表示方式. Fearnley 等人[24]和 Blondin 等人[21]的证明路线如下:

$$\text{subset sum game}\leqslant_{\mathrm{L}} b\text{-Safe CSA reachability}\leqslant_{\mathrm{L}}\text{1-BCA reachability}\leqslant_{\mathrm{L}}\text{2-VASS reachability} \tag{6}$$

2.2.3.1 有界 1-计数器自动机

有界 1-计数器自动机 (1-BCA) 是一个 3-元组 $M=(Q,b,\delta)$, 其中, Q 是一个有限状态集, $b\in\mathbb{N}$ 是一个计数器的全局上界, $\delta\subseteq Q\times\mathbb{Z}\times[b]_0^2\times Q$ 是有限的迁移规则集合. M 的格局形如 $p(x)\in Q\times[b]_0$, 若迁移 $p(x)\xrightarrow{t}q(y)$ 成立, 则一定存在 $t=(p,y-x,lb,ub,q)\in\delta$ 使得 $lb\leqslant x\leqslant ub$. 直观上, 1-BCA 可以判断计数器的值是否大于 lb 或小于 ub, 自然也可以完成测零 (令 $lb=ub=0$), 故其可达性问题至少是 **NP**-难的.

实际上, 1-BCA 可以直接看作一个格局在 $Q\times(\mathbb{N}\cap[0,b])$ 中的 1-VASS (有界 1-VASS). 换言之, 其迁移规则中的局部上下界参数 lb、ub 不是必须的. 给定 $M=(Q,b,\delta)$, 若有规则 $t=(p,a,lb,ub,q)\in\delta$, 我们可以用如下的有界 1-VASS V 的运行替代:

$$\xrightarrow{\pi}=\xrightarrow{(p,-lb,p_0)}\xrightarrow{(p_0,lb,p_1)}\xrightarrow{(p_1,b-ub,p_2)}\xrightarrow{(p_2,ub-b,p_3)}\xrightarrow{(p_3,a,q)} \tag{7}$$

若 $p(x)\xrightarrow{t}_M q(y)$, 则 t 满足 $x-lb\geqslant 0$, $x+b-ub\leqslant b$, $x+a=y$, 于是 $p(x)\xrightarrow{\pi}_V q(y)$; 反之若 $p(x)\xrightarrow{\pi}_V q(y)$, 则一定有 $0\leqslant x-lb\leqslant b$, $0\leqslant x+b-ub\leqslant b$, $x+a=y$, 于是 $p(x)\xrightarrow{t}_M q(y)$. 此过程为每条迁移引入 4 个中间状态, 最终 V 状态数

是 $|Q|+4|\delta|$, 迁移规则有 $5|\delta|$ 条, 总体是一个线性的放大. 因此二者可对数空间互相归约, 故有以下引理.

引理 6[24]. 1-BCA reachability$\leqslant_{\mathrm{L}}$2-VASS reachability.

证明: 给定 1-BCA 可达性实例 $\langle M=(Q,b,\delta),p_s,p_t\rangle$, 由上述讨论知 δ 可以直接定义为 $Q\times\mathbb{Z}\times Q$ 的有限子集. 构造 2 -VASS 可达性实例 $\langle V=(2,Q,T),p_s(0,b),p_t(0,b)\rangle$, 其中, $T=\{(p,(a,-a),q):(p,a,q)\in\delta\}$. 显然, $p(x)\xrightarrow{(p,a,q)}_M q(y)$ 当且仅当 $p(x,b-x)\xrightarrow{(p,(a,-a),q)}_V q(y,b-y)$, 如果原问题实例是可达的, 对该可达的证据逐步应用这一观察即可知有 $p_s(0,b)\to_V p_t(0,b)$. 反之, 若 $p_s(0,b)\to_V p_t(0,b)$, 将每一步向量投影到第 1 维即可得到 $p_s(0)\to_M p_t(0)$. 若两问题采用的编码方案相同, 此归约可在对数空间完成. 证毕.

2.2.3.2 计数器栈自动机

计数器栈自动机是一个 3-元组 $M=(Q,d,\delta)$, 其中, Q 是有限状态集, $d\in\mathbb{N}$ 是计数器数量, δ 是有限迁移规则集. M 的格局形如 $p(\boldsymbol{u})\in Q\times\mathbb{N}^d$, 迁移规则形如 $t=(p,E,\boldsymbol{I},R,q)\in\delta$. 给定格局 $p(\boldsymbol{u})$, $q(\boldsymbol{v})$, $p(\boldsymbol{u})\xrightarrow{t}_M q(\boldsymbol{v})$ 当且仅当下述条件成立.

- $E:[d]\to\mathbb{N}$ 是偏函数, 指定了对某些维度的测量, 对于每个 $i\in\mathrm{dom}(E)$ 都有 $\boldsymbol{u}(i)=E(i)$; 若 $E(j)$ 有定义, 则对任意 $j\leqslant i\leqslant d$, $E(i)$ 也有定义.
- $R\subseteq\mathrm{dom}(E)$ 指定了需要置为 0 的计数器, 对于每个 $i\in R$ 都有 $\boldsymbol{v}(i)=0$.
- $\boldsymbol{I}\in\mathbb{N}^d$ 指定了 d 个计数器的增量, 对于每个 $i\notin R$ 都有 $\boldsymbol{v}(i)=\boldsymbol{u}(i)+\boldsymbol{I}(i)$.

上述规则本质上是为了模拟 1-BCA 的行为. 事实上, 给定 d 个计数器构成的栈 $\boldsymbol{u}$, 可以选择一个充分大的 k 按照 k 进制用单个计数器记录下 $C(\boldsymbol{u})=\sum_{i=1}^{d}\boldsymbol{u}(i)k^{i-1}$. 图 4 以 $k=16$, $d=3$ 为例展示了这一编码过程的直观解释.

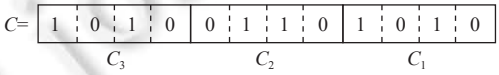


图 4 CSA 的直观解释

给定规则 $t=(p,E,\boldsymbol{I},R,q)$, 若 $p(\boldsymbol{u})\xrightarrow{t}q(\boldsymbol{v})$ 成立, 易知 $\mathrm{dom}(E)$ 一定形如 $\{j,j+1,\ldots,d\}$, 定义:

$$lb=\sum_{j\leqslant i\leqslant d}E(i)k^{i-1},\ ub=lb+k^{j-1}-1 \tag{8}$$

则一定有 $lb\leqslant C(\boldsymbol{u})=\sum_{j\leqslant i\leqslant d}E(i)k^{i-1}+\sum_{i\in[j-1]}c_ik^{i-1}\leqslant ub$. 令 $a=\sum_{i\notin R}\boldsymbol{I}(i)k^{i-1}-\sum_{i\in R}E(i)k^{i-1}$, 则有 $C(\boldsymbol{u})+a=C(\boldsymbol{v})$. 最终可知, 可用 1-BCA 中的迁移 $p(C(\boldsymbol{u}))\xrightarrow{(p,a,lb,ub,q)}q(C(\boldsymbol{v}))$ 模拟原 CSA 中的迁移. 对所有 $t\in\delta$ 构造对应的迁移构成集合 δ', 并令 $B=k^d$, 我们可以得到一个 1-BCA $M'=(Q,B,\delta')$. 需要注意的是, k 的选取十分重要, 否则计数器栈有上溢的风险, 导致编码不是一一对应的. 因此, 在归约中我们仅考虑一类特殊的 CSA, 它运行中任意可达的格局 $p(\boldsymbol{u})$ 都满足对任意 $i\in[d]$, $\boldsymbol{u}(i)\leqslant b$, 此性质称为 b-安全的 (b-safe). 这样只需在上述证明中取 $k=b$ 即可保证这样的 CSA 都能够被 M' 模拟. M' 的构造过程可在对数空间完成, 即有如下引理.

引理 7[24]. b-Safe CSA reachability$\leqslant_{\mathrm{L}}$1-BCA reachability.

综合引理 1、引理 6、引理 7, 为了证明 2-VASS 可达性问题是 **PSPACE**-难的, 还需证明下述引理.

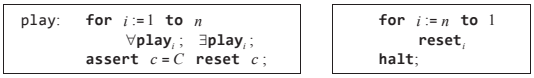
引理 8[24]. subset sum game$\leqslant_{\mathrm{L}}b$-Safe CSA reachability.

证明: 这里使用 CSA 的程序表示, 我们在计数器程序中额外引入形如"**assert** $x=A$ **reset** x;"的断言语句, 表示若计数器 x 值为常量 A 则把 x 置零, 否则终止该次执行, **reset** 动作也可省略不做. 给定子集和博弈问题实例 A_1, B_1, E_1, F_1,…, A_n, B_n, E_n, F_n, C, 我们需要遍历博弈中 x_i ($i\in[n]$) 的所有可能取值, 随后判定是否存在对应的一组 y_i 满足 $\sum_{i=1}^{n}(x_i+y_i)=C$. 为此我们可以使用一组计数器 a_i, b_i, e_i, f_i ($i\in[n]$) 记录遍历过程中选择的分支, c 用于累加选择的数, 即构造下面形式的语句.

$\forall$**play**$_i$ = **do** a_i +=1, c += A_i; **or** b_i +=1, c += B_i;

$\exists$**play**$_i$ = **do** e_i +=1, c += E_i; **or** f_i +=1, c += F_i;

最终博弈阶段的程序如图 5 中的 P_{play} 所示.



P_{play} P_{reset}

图 5 子集和博弈问题对应的 CSA 程序

对于每个 $i\in[n]$, a_i, b_i 仅有一个非零, 例如若 $a_i=1$ 则 ∀ 玩家该步选择了 A_i; e_i, f_i 的情况类似. 若 P_{play} 能执行完毕, 则寻找到了使得 $\sum_{i=1}^{n}(x_i+y_i)=C$ 成立的一组数, 即 ∃ 玩家通过一轮博弈.

随后, 我们需要合理重置进程使得最终恰好能够遍历 ∀ 玩家的 2^n 种不同选择. 定义如下程序模块.

reset$_i$=**do assert** $e_i=2^{n-i}$, f_i=0 **reset** e_i, f_i;

or assert e_i=0, $f_i=2^{n-i}$ **reset** e_i, f_i;

do assert $a_i=2^{n-i}$, b_i=0; **goto** play;

or assert $a_i=2^{n-i}$, $b_i=2^{n-i}$ **reset** a_i, b_i;

重置阶段的程序如图 5 中的 P_{reset} 所示. 最终完整的程序 P 由 P_{play} 顺序复合上 P_{reset} 构成.

接着证明该构造的正确性: 易知 P 的任意一次执行 R 总会因为重置阶段的 **goto** play 语句不断地回到程序的起点, 我们可以以此为断点将 R 划分为若干段从 play 处开始的执行"$R_1, R_2, \ldots$". 注意以下事实.

● 若 $j=0\,(\mathrm{mod}\,2^{n-i})$, 则 R_j 一定成功执行了 **reset**$_i$ 中的语句, 否则 R_j 会因为不满足断言而中断.借此可以推断若 R 为完全的执行, 则 R 恰好可以分割为 2^n 段.

● 对于一次完全的执行 $R=R_1,\ldots,R_{2^n}$, 可以将每段执行的下标 $j\in[2^n]$ 编码为 n-比特二进制数 $j_{(2)}$. 若 $j_{(2)}$ 从左起第 i 位为 0, 则说明博弈阶段 R_j 增加了 a_i; 若 $j_{(2)}$ 从左起第 i 位为 1, 则说明博弈阶段 R_j 增加了 b_i. 结合 a_i, b_i 的含义可知, R 恰好遍历博弈树的所有分支.

● 若 R_j 的执行没有中断, 则 ∃ 玩家能够赢得 j 对应的分支. 若 R 是完全的, 则 ∃ 玩家有必胜策略.

上述事实均可通过对 n 归纳进行证明, 其证明细节在此不再赘述, 感兴趣的读者可以参考文献 [19].

综上, $\forall x_1\in\{A_1,B_1\}\ \exists y_1\in\{E_1,F_1\}\ldots\forall x_n\in\{A_n,B_n\}\ \exists y_n\in\{E_n,F_n\}\sum_{i=1}^{n}(x_i+y_i)=C \iff P$ 存在 **0**-执行. 将计数器按照 a_1, b_1, e_1, f_1,…, a_n, b_n, e_n, f_n, c 的顺序从小到大编号, 使得 P 中对某一计数器进行断言时, 编号更大的计数器值都已知 (为 0), 这满足 CSA 的迁移规则要求, 从而 P 等价于一个 CSA. 由于所有计数器的值都不超过 $b=2^n+\sum_{i\in[n]}(a_i+b_i+e_i+f_i)$, 因此 P 是 b-安全的. 若 P 使用二进制编码, 则此构造能在对数空间完成. 证毕.

现公式 (6) 中的所有归约已完成证明, 注意, 引理 8 的证明要求 CSA 必须是二进制编码的, 故后续归约使用的 BCA 及 2-VASS 也必须是二进制编码的. 由引理 1 可得如下引理.

引理 9[21]. 二进制编码下的 2-VASS 可达性问题是 **PSPACE**-难的.

结合第 2.2.1 节给出的上界, 最终得到如下定理.

定理 3[21]. 二进制编码下的 2-VASS 可达性问题是 **PSPACE**-完备的.

3 *d*-VASS 的下界证明技术

本节先对一般 VASS 可达性问题的 **Ackermann**-完备性及其关于维度的参数复杂性结论做一个简要的回顾, 随后分节讨论几个重要的下界证明技术及各自的启发和局限.

20 世纪 80–90 年代, Mayr 等人[25]、Kosaraju 等人[26]、Lambert 等人[27]、Sacerdote 等人[28]就此问题进行了大量深入的研究, 提出并简化了判定 VASS 可达性问题的算法, 即著名的 KLMST 分解算法. 2015 年, Leroux 等人[29,30]给出了 KLMST 算法的首个上界结论, 证明了一般 VASS 可达性问题在 $\mathbf{F}_{\omega^3}$ 中. 通过对原算法的不断优化, 在 2017 年和 2019 年, 上界先后被降到了 $\mathbf{F}_{\omega^2}$ [31]和 $\mathbf{F}_{\omega}$ [32]. 其中, 文献 [32] 的主定理是一个关于维度的精细结论, 即: 对于任意的 $d\geqslant 3$, d-VASS 可达性问题在 $\mathbf{F}_{d+4}$ 中. 相较于最初的 $\mathbf{F}_{\omega^3}$ 上界, 此结论是一个巨大的提升, 但仍远未达到最优. 2024 年, 国内研究者 Fu 等人[33]基于几何维度 (geometric dimension) 的概念进一步改进了 KLMST 算法, 并得到了目前关于 d-VASS 可达性问题的最优上界.

定理 4[33]. 对于任意的 $d \geqslant 3$, d-VASS 可达性问题在 $\mathbf{F}_d$ 中.

此结论对复杂性上界的刻画十分优美, 但遗憾的是目前没有相匹配的下界. 事实上, 理论计算机科学界对 VASS 可达性问题的复杂性下界的研究由来已久. 一方面, 早在 1976 年该问题可判定性尚且未知时, Lipton[34]就给出了 **EXPSPACE**-难的下界归约. 但由于问题复杂, 下界研究随后沉寂了 40 余年. 新的突破始于 2020 年, Czerwiński 等人[35]借鉴 Lipton 的证明方法开创了一套普适性强的归约流程, 并给出了首个非初等的 **Tower**-难下界. 该工作发表于 Journal of the ACM. 接着在 2021 年, Czerwiński 等人[14]和 Leroux[15]通过巧妙地改进流程细节, 为一般 VASS 可达性问题提供了匹配的 **Ackermann** 下界, 一个开放多年的复杂性问题至此得到解决. 另一方面, d-VASS 可达性问题的参数化下界更具挑战性. 经过研究者[36−38]的不断努力, 目前最先进的结果如下.

定理 5[36]. 对于任意的 $d \geqslant 3$, $(2d+3)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

除了结论之外, 改进过程本身亦有重要的参考价值, 接下来我们将总结自 1976 年以来人们为解决下界问题所提出的研究方法, 指出不同方法的使用范围和局限性, 以期对未来的相关工作有所启发.

3.1 Lipton 证明技术

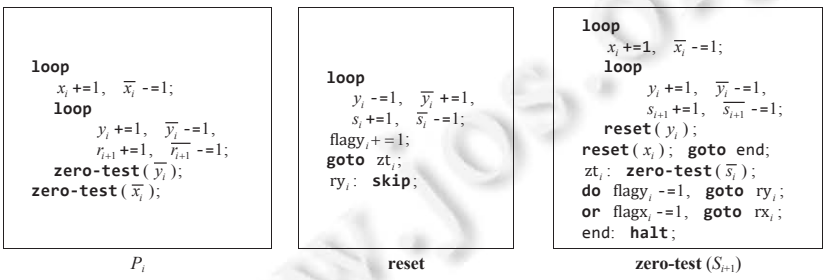
Lipton 给出了 VASS 可达性问题的第 1 个非平凡复杂性下界——**EXPSPACE**-难. 其原方法是基于并行程序 (parallel program) 进行的, 文献 [16] 中对此有详尽的描述, 在此不再赘述. 在第 3.1.1 节中, 我们将基于更为现代的计数器程序方法给出一个全新的证明, 并在第 3.1.2 节分析 Lipton 方法的局限性及其对后续研究工作的启发.

3.1.1 **EXPSPACE**-难归约

本节介绍 Lipton 的证明思路. 首先, 我们需要选择合适的问题用于归约. 引理 2 表明 2^{2^n}-有界计数器机器可达性问题是 **EXPSPACE**-难的, 本节正是从此问题出发进行归约. 给定一个 2^{2^n}-BCM M, 我们接着将其归约到一个计数器程序 P. 由于计数器机器和计数器程序唯一的区别在于前者可以测零, 但后者不具备此能力, 故归约的关键在于: 如何使用一个程序模拟 M 中的每一次测零? 事实上, 后续每一次下界提高的背后都蕴含着模拟测零技术的改进.

在第 2.2.3.1 节我们曾提到一种简单的测零技术: 当已知计数器 x 满足 $0 \leqslant x \leqslant b$ 时, 只需执行"x+= b; x−= b;"程序. 若该语句顺利执行, 定有 $x \leqslant 0$, 从而 $x = 0$. 当上界 b 充分小 (关于输入规模成指数量级) 时, 可以高效构造出这条语句. 但显然 2^{2^n} 不符合此情形. 最终, Lipton 巧妙地通过递归构造出了上界序列 2^{2^0}, 2^{2^1},…, 2^{2^n}.

定义 $A_i = 2^{2^i}$, 构造过程中为了保证某个计数器 x_i 是 A_i-有界的, 只需定义辅助计数器 $\overline{x_i}$ 使得 $x_i + \overline{x_i} = A_i$ 即可. 假设已有 $x_i + \overline{x_i} = A_i$, $y_i + \overline{y_i} = A_i$, $s_{i+1} + \overline{s_{i+1}} = A_{i+1}$, 需要对 s_{i+1} 测零, 又观察到有 $A_{i+1} = A_i^2$. 为了实现 s_{i+1} += A_{i+1} 的效果只需用 x_i, y_i 构造图 6 中程序 P_i 所示的双层循环.



P_i reset zero-test (S_{i+1})

图 6 Lipton 的模拟测零技术

在初始 $x_i = y_i = s_{i+1} = 0$ 的情形下, 此段程序可以执行完全, 最终 $x_i = y_i = A_i$, $s_{i+1} = A_{i+1}$. 但此段程序有两个问题: 其一, 它没有恢复 x_i、y_i、s_i 的状态, 无法用 x_i、y_i 进行下一次测零; 其二, 它是递归构造的, **zero-test**($\overline{y_i}$) 和 **zero-test**($\overline{x_i}$) 各需要使用两对辅助计数器, 使得最终整个程序需要使用 4^i 个计数器, 这显然是无法高效完成的. 为此, 可以将 **zero-test**($\overline{y_i}$) 替换成图 6 中的 **reset**(y_i), 将 y_i、$\overline{y_i}$ 的值记录到 s_i、$\overline{s_i}$ 的同时重置为 $y_i = 0$, $\overline{y_i} = A_i$, 随后通过 **zero-**

test($\overline{s_i}$) 等效实现对 $\overline{y_i}$ 的测零; **zero-test**($\overline{x_i}$) 同理. 因为有两处会跳转到 **zero-test**($\overline{s_i}$) 的入口 zt_i 处, 这里分别用 flagx_i 和 flagy_i 作为信号, 确保完成测零后准确跳转回对应的位置 (rx_i 或 ry_i). 最终完整的程序如图 6 中 **zero-test**(s_{i+1}) 所示.

注意到, 若我们将 **zero-test**(s_{i+1}) 中的"s_{i+1} +=1; $\overline{s_{i+1}}$ −=1;"这一句替换成"x_{i+1} +=1;", 最终可以构造出 $x_{i+1}=A_{i+1}$. 至此, 我们足以完成测零的模拟和对 $x_n+\overline{x_n}=A_n$ 的递归构造. 如下定理成立.

定理 6[34]. 一般 VASS 可达性问题是 **EXPSPACE**-难的.

证明: 给定 2^{2^n}-有界计数器机器可达性实例 $\langle M,1^n\rangle$, 不妨设其使用的计数器为 a、b、c. 用前文阐述的方法构造出 $\overline{a}=\overline{b}=\overline{c}=A_n$, 此过程需要使用辅助计数器 x_i、$\overline{x_i}$、y_i、$\overline{y_i}$、s_i、$\overline{s_i}$、flagx_i 和 flagy_i ($i\in[n-1]$). 随后用计数器程序 P 模拟 M 的每条语句, 并将测零语句替换成对应的模块, 例如"**zero**? a" 应该替换为"**zero-test**(a); **zero-test**($\overline{a}$);" (只使用 **zero-test**(a) 会对调 a、$\overline{a}$ 的值, 需要使用 **zero-test**($\overline{a}$) 对调回来). 易知若 M 有完全运行则 P 也有完全运行, 否则 P 会在某次模拟测零的过程中终止执行. 同时, 最终的程序只使用 $8n-2$ 个计数器, 整体长度至多存在一个多项式的放大, 因此可以在多项式时间内构造完毕, 从而有 2^{2^n}-BCM reachability $\leqslant_{\mathrm{p}}$ CP reachability. 最终由引理 2 得出结论: 一般 VASS 可达性问题是 **EXPSPACE**-难的. 证毕.

3.1.2 Lipton 方法的局限性与启发

Lipton 给出了一般 VASS 可达性问题复杂性的首个非平凡下界, 他的证明方法是开创性的. 但后继研究表明, 这种方法有较大的局限性. 本节将详尽分析其局限性与相应改进思路.

● 技术局限. 回顾定理 6 证明中所有模拟测零模块的行为都是完全确定的, 以至于最终并不需要判定"P 存在 **0**-执行"这么强的条件, 只需判定 P 存在完全执行即可. 换而言之, 我们实际上证明的是: 一般 VASS 的可覆盖性问题是 **EXPSPACE**-难的. 另外, 在程序 P 末尾添加"**loop** a +=1;" 语句可以使得若 M 存在 **0**-执行当且仅当 P 停机后终止格局的取值情况有无限种, 于是一般 VASS 的有界性问题也是 **EXPSPACE**-难的. 1978 年, Rackoff[13]给出了可覆盖性问题和有界性问题的上界, 故有如下命题.

命题 3[13,34]. 一般 VASS 的可覆盖性问题和有界性问题都是 **EXPSPACE**-完备的.

换言之, Lipton 的证明思路存在 **EXPSPACE**"壁垒": 无法使用类似的归约方法得到任何严格更难的复杂性下界 (如 **Tower**-难). 但由于可达性问题上下界之间差距巨大, 人们倾向于相信其一定具有严格更高的下界. 为了得到这样的突破, 我们须在归约时构造一类行为并非完全确定的程序, 使其执行完全后至少有一个计数器的值无法确定. 换言之, 我们须通过判定这些计数器最终的取值是 0 来保证程序中所有模拟测零是成功的; 若我们的程序不满足此性质, 则此次构造能够证明出的下界一定不比 Lipton 的高.

● 正面启发. 从 Lipton 的证明中还可以提炼出一种模拟测零的思路.

1) 构造不变量: 若需要对 x 进行模拟测零, 维护形如 $x+\overline{x}=f(n)$ 的不变量是有帮助的, 其中, $f(n)$ 与用于归约的问题有关.

2) 复用计数器: 多次模拟测零时, 不能为每一次模拟单独写一个独立的程序片段, 否则可能会造成使用的计数器数量超指数量级. 我们更应当思考如何去复用计数器, 例如对 x、y 测零时, 可以通过类似 reset 的模块将它们的值统一记录到某个计数器 s 上, 通过对 s 的模拟测零间接实现原目的.

结合上述两点, 我们可以观察到: 若已构造出 $x+\overline{x}=f(n)$, 每次对 x 测零时我们不妨将 $\overline{x}$ 的值转移到 s 上. 注意, 此过程会对调 x、$\overline{x}$ 的值, 为了恢复原值我们需要再操作一次. 假设待模拟机器 M 一共进行了 m 次测零且 s 初始化为 0, 则完全执行后都应当满足 $s\leqslant 2mf(n)$, 同时若 $s=2mf(n)$ 则说明每次测零时 $\overline{x}=f(n)$ 都成立, 即原机器 M 的 m 次测零都成功了. 最终, 通过判定停机后 $s=2mf(n)$ 是否成立即可说明 M 是否存在 **0**-执行.

上述方法依然存在问题: 首先, $2mf(n)$ 往往是个大数, 无法在归约的时候直接构造出来; 其次, m 的精确值大部分情况下都是未知的. 针对第 1 点的解决方案是: 既然可以构造 $x+\overline{x}=f(n)$, 我们不妨初始化 $s=2mf(n)$, 每次转移不是增加 s 而是减少 s, 最终判定 $s=0$ 是否成立即可. 而后者要求我们必须能够针对无限个 $m\in\mathbb{N}$ 构造出 $s=2mf(n)$, 这样总会有充分大的 m 可以覆盖 M 的测零次数, 随后我们在 M 停机前添加若干次测零使得最终总测

零次数恰好是 m 即可, 其代价是需要引入一个辅助计数器记录剩余可用测零次数. 但即便是在最新的工作里, 常数个计数器的代价也并不算高.

至此, 我们得到了一套相当完善的测零方法. 归约中的关键步骤仅剩以下两步.

1) 构造一对计数器 x、$\bar{x}$, 使其总和等于某给定函数 $f(n)$.

2) 对任意 $m\in\mathbb{N}$ 和给定的 $f(n)$, 构造一对值分别等于 m、$2mf(n)$ 的计数器.

事实上, 后续所有工作中几乎都涉及了此测零方法和上述步骤的变体. 在接下来的几节, 我们将会看到此方法的具体应用以及研究者是如何通过不断改进这一技巧得到更好的下界的.

3.2 下界证明框架

Czerwiński 等人[35]形式化了第 3.1.2 节中的想法, 他们将实现步骤 1) 的程序模块称作放大器 (amplifier); 放大器构造出的计数器 $x=f(n)$ 和步骤 2) 中值分别为 m、$2mf(n)$ 的计数器共同称为乘法三元组 (multiplication triple). 后续工作[14,15,35−38]均沿用了相同的思路, 但不同研究者的定义方式不同, 这对该方法未来的改进与推广十分不利. 我们将在第 3.2.1 节中严格给出三元组和放大器的一般定义, 随后在第 3.2.2 节介绍了基于二者的通用模拟测零技巧与下界证明框架, 并在后文中以此框架统一地梳理近些年的进展.

3.2.1 乘法三元组与放大器

本节给出乘法三元组和放大器的定义, 相关文献中具体使用的不同定义均与此等价. 乘法三元组可以推广到更一般的情形: 任给一个三元不变量 inv, 我们均可按照如下方式定义一个 inv-三元组.

定义 3. 给定函数 $\mathrm{inv}:\mathbb{N}^3\to\mathbb{N}$, 一个 inv-三元组形如 $\tau=(a,b,c,A,B,X)$, 其中, $A,B\in\mathbb{N}$; X 是计数器的有限集; $a,b,c\in X$ 是 3 个计数器, 它们满足 $a=A$, $b=B$, $\mathrm{inv}(A,B,c)=0$ 且对任意 $x\in X\setminus\{a,b,c\}$ 都有 $x=0$.

直观上, A 代表 τ 允许的测零次数的 2 倍, 需为偶数; B 代表待测零计数器值应当满足的上界, 通常是某个增长超过指数量级的函数; c 用于在每次模拟中记录下待测零计数器的信息, 故应和 a、b 保持某些不变量关系 inv. 若 inv 定义为 $\mathrm{inv}(A,B,C)=C-AB$, 对应的 inv-三元组即乘法三元组, 本节仅讨论乘法三元组.

考虑某个使用计数器 $X=\{x_1,\ldots,x_k\}$ 的程序 P, 由于乘法三元组确定了 X 中所有计数器的值, 后文有时直接用 $\tau=(a,b,c,A,B,X)$ 来指定初始或终止格局. 此外, 若某次完全执行后对任意计数器 $x\in Z\subseteq X$ 都有 $x=0$, 则此次执行称为 Z-执行. 若某次 Z-执行初始格局为 α, 终止格局为 β, 则称 P 可以从 α 出发 Z-计算出 β.

定义 4. 给定 $f(n)\geqslant n$, 一个 $f(n)$-放大器是一个 5-元组 $\mathcal{A}mpl=(P,X,(a',b',c'),(a,b,c),Z)$.

- P 是一段计数器程序, 它使用的所有计数器集合为有限集 $X\supseteq Z\cup\{a,b,c\}\cup\{a',b',c'\}$.
- $a,b,c,a',b',c'\in X$ 是计数器; a'、b'、c' 称为放大器的输入; a、b、c 称为放大器的输出; $Z\neq\varnothing$ 是一组控制计数器.
- 若存在 $A_s,B\in\mathbb{N}$, 使得 (a',b',c',A_s,B,X) 构成一个乘法三元组, 则任意 Z-执行后存在 $A_t\in\mathbb{N}$ 使得 $(a,b,c,A_t,f(B),X)$ 构成一个乘法三元组.
- 对任何 $A,B\in\mathbb{N}$, 存在 $A_s\in\mathbb{N}$, 使得: 若 $\tau_s=(a',b',c',A_s,B,X)$ 构成一个乘法三元组, 则从该格局出发可以 Z-计算得到某个 $\tau_t=(a,b,c,A_t,f(B),X)$ 使得 $A_t\geqslant A$.

引入控制计数器集合 Z 可弥补 Lipton 方法的局限: 将通过判定 Z 中计数器的取值是否为 0 来“控制”程序终止于某个希望的格局. 为此 $Z\neq\varnothing$ 是必须的. 注意, $\{a,b,c\}$、$\{a',b',c'\}$、Z 不必互斥, 一般而言至少有 $c'\in Z$. 事实上这 3 组计数器的复用程度往往对 d-VASS 复杂性下界有显著影响.

3.2.2 通用模拟测零技术

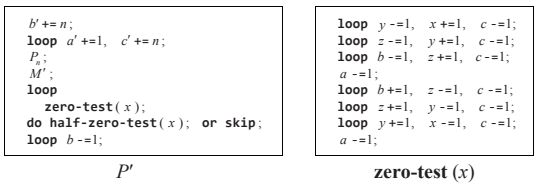
第 1.4.1 节定义了计数器机器 $f(n)$-有界可达性问题, 基于乘法三元组和放大器, 我们有如下引理.

引理 10. 给定 $f(n)$, 若存在一族 $f(n)$-放大器 $\mathcal{A}mpl_n=(P_n,X_n,(a',b',c'),(a,b,c),Z_n)$ 使得其中程序 P_n 长度为 $\mathrm{poly}(n)$, 则有如下结论.

1) CM $f(n)$-bounded reachability $\leqslant_{\mathrm{p}}$ VASS reachability.

2) 当 $|X_n|$ 是个与 n 无关的常数 d 时, CM $f(n)$-bounded reachability $\leqslant_{\mathrm{p}}(d+3)$-VASS reachability.

证明: 给定计数器机器 $f(n)$-有界可达性问题实例 M、1^n, 不妨假定 M 使用的计数器是 $x,y,z \notin X$ (我们总能通过重命名达到这个目的), 构造使用计数器 $X'_n = X_n \cup \{x,y,z\}$ 的程序 P', 如图 7 所示.



P' **zero-test** (x)

图 7 使用放大器和乘法三元组模拟 M 的执行

P' 可以划分为 3 个阶段.

1) 初始化阶段. 第 1 个 **loop** 语句构造了一个平凡的乘法三元组, 若其完全执行后 $a' = A_s$, 则一定有 $c' = A_s n$, 因此执行 P 前的初始格局是 $\tau_s = (a',b',c',A_s,n,X'_n)$. $\mathcal{A}mpl_n$ 是一个 $f(n)$-放大器, 它的程序 P_n 不改变 x、y、z 的值, 可以看作计数器集合为 X_n' 的 $f(n)$-放大器, 因此从 τ_s 出发可以 Z_n-计算出某个 $\tau_t = (a,b,c,A_t,f(n),X'_n)$.

2) 模拟阶段. M' 由 M 变换而来. 首先对于 M 中形如 x += k 的语句, 则 M' 中需转换为 x += k; b −= k; 形如 x −= k 的语句, 则转换为 x −= k; b += k, 即借用计数器 b 辅助维持 $x+y+z+b$ 总量不变. 注意, 执行 M' 前的格局 τ_t 满足 $x=y=z=0$, $b=f(n)t$, 因此每次执行完模拟后都有 $x+y+z+b=f(n)$. 其次, 对跳转语句的模拟是平凡的, **halt** 语句直接忽略. 最后, 考虑 M 中形如"**zero**? x"的语句, 在 M' 中需转换为图 7 中 **zero-test**(x) 所示的程序模块. 设 $\alpha \xrightarrow{\textbf{zero-test}(x)} \beta$, 则计数器 c、a 的减少量分别满足 $0 \leqslant \Delta c \leqslant 2(\alpha(y)+\alpha(z)+\alpha(b)) = 2(f(n)-\alpha(x))$, $\Delta a = 2$. 注意, $\Delta c = 2f(n)$ 当且仅当 $\alpha(x)=0$, 并且此时 α、β 在除了 a、c 的计数器上取值是一致的, 此情况称作一次完全的模拟.

3) 结束阶段. 乘法三元组中 a 的值指定了对应的模拟测零次数, 若 M' 中未完成对应次数的模拟测零, 我们则需要在结束阶段非确定地循环若干次测零模块来完成对 a 的清零. 随后非确定地选择执行 1 次或 0 次模块 **half-zero-test**(x), 即 **zero-test**(x) 的前 4 行, 此处是为了使得即便 a 的值为奇数也能被置零. 最终, 使用循环语句将 x、y、z、b 清零.

正确性分析. 假设 M 存在一个使用了 k 次测零的 $f(n)$-有界 0-执行 R, 令 $A=2k$, 由 $\mathcal{A}mpl_n$ 的性质可知: 存在 A_s 使得 P_n 能够从 $\tau_s = (a',b',c',A_s,n,X'_n)$ 出发 Z-计算出 $\tau_t = (a,b,c,A_t,f(n),X'_n)$ 使得 $A_t \geqslant A$. 考虑 P' 的执行 R', 它在初始化阶段恰好循环 A_s 次, 使得执行 P_n 前初始格局恰好是 τ_s, Z-执行完程序 P_n 可以构造出 τ_t. 随后用 R' 模拟 R, 因为 R 中每个格局都满足 $x+y+z \leqslant f(n)$, 这是可以做到的. **zero-test** 模块的性质保证存在一条路径使得最终执行完 M' 后的格局 β 满足 $\beta(a) = A_t - 2k$, $\beta(c) = (A_t-2k)f(n)$, 让结束阶段循环恰好 A_t-2k 次且每次模拟测零都是完全的, 则最终 $a=c=0$. 由于执行完 P_n 后, a、b、c 外计数器的值为 0 且不再改变, 计数器 a、b、c 也被清零了, 可以确定终止格局为 0.

反之, 假设 P' 存在一个 0-执行 R', 则在初始化阶段, R' 对某个 A_s 构造出了 $\tau_s = (a',b',c',A_s,n,X'_n)$. 由于模拟阶段和结束阶段不改变 Z 中计数器的值, 因此这些计数器在执行完 P_n 后就为 0. 由此可知, R' 中对应程序 P_n 的部分是 Z 执行, 存在 A_t 使得执行后格局为 $\tau_t = (a,b,c,A_t,f(n),X'_n)$. 最终 $a,c=0$, 因此模拟阶段和结束阶段共进行了 $A_t/2$ 次模拟测零和至多 1 次半模拟测零, 得到满足 $\beta(c)=0$ 的格局 β, 从而每一次模拟测零都是完全的, 且对应的待测零计数器确实是 0. 考虑 R' 对应程序 M' 的部分, 可以诱导出一个 M 的完全执行 R; 因为 R' 维持了 $x+y+z \leqslant f(n)$, 因此 R 是 $f(n)$-有界的.

综上, M 存在 $f(n)$-有界 **0**-执行当且仅当 P' 存在 **0**-执行. P' 使用了 $|X_n|+3$ 个计数器, 整体上是多项式长的, 因此 CM $f(n)$-bounded reachability $\leqslant_{\mathrm{p}}$ VASS reachability. 当 $|X_n|$ 是个与 n 无关的常数 d 时, 该结论可以更精细地表述为 CM $f(n)$-bounded reachability $\leqslant_{\mathrm{p}} (d+3)$-VASS reachability. 证毕.

此引理为本节核心内容, 其主要贡献如下.

1) 引入了基于乘法三元组的通用模拟测零技术.

2) 介绍了给定一族 $f(n)$-放大器 $\{\mathcal{A}mpl_n\}$ 后, 模拟 $f(n)$-有界计数器机器的一般方法.

这使得在后续的证明中, 我们无须再关注繁琐的模拟步骤, 只需对给定的 $f(n)$ 构造计数器尽可能少的 $f(n)$-放大器即可. 对于 $f(n)=\text{tower}(n)$ 的情形, Czerwiński 等人[35]构造出了一族 tower(n)-放大器, 但 $|X_n|$ 和 n 呈线性关系, 因此只能证明一般 VASS 可达性问题是 **Tower**-难的. 但如果考虑 $f_k(n)=k\text{-exp}(n)=2^{\cdot^{\cdot^{\cdot^{2^n}}}}\Big\}k$ times 且固定 k, 他们实际上构造了一族一致 (计数器集合 X 与 n 无关) 的 k-exp(n)-放大器. 和引理 2 同理可证, 计数器机器 $(k+1)$-exp(n)-可达性问题是 k-**EXPSPACE**-难的, 最终有如下定理.

定理 7[35]. $(k+13)$-VASS 可达性问题是 k-**EXPSPACE**-难的.

但文献 [35] 的构造方式过于繁琐, 且在最新技术下完全可以使用常数个计数器完成[37], 因此我们认为其技术参考价值不大, 故省略证明细节.

3.2.3 d-VASS 相关非初等下界总览

由于一般 VASS 可达性问题下界是非初等的, 上界甚至是非原始递归的, 我们有理由相信 $\mathbf{F}_d$-难问题可以归约到某个维度的 VASS 上. 根据引理 10 和推论 1, 有如下推论.

推论 2. $d\geqslant 3$ 时, 若存在一族 $F_d(n)$-放大器 $\mathcal{A}mpl_d=(P_d,X_d,(a',b',c'),(a,b,c),Z_d)$ 使得程序 P_d 的长度为 $p(n)\in\mathcal{F}_{<d}$, 令 $D=\max(|X_d|,2+|\{a,b,c\}\cup Z_d|)$, 则 D-VASS 可达性问题是 $\mathbf{F}_d$-难的.

证明: 只需给出 D 的计算, 其他和引理 10 同理. 事实上, 在初始化阶段结束后 X_d 中的计数器可分为 3 类: 输出 a, b, c; 控制计数器 Z_d; 剩余的 $|X_d|-|\{a,b,c\}\cup Z_d|$ 个可复用计数器. 模拟阶段的两个计数器完全可以从可复用计数器里选择 (如有), 因此整体只需 $D=|X_d|+\max(0,2-|X_d|+|\{a,b,c\}\cup Z_d|)$ 个计数器. 证毕.

若能构造一族 $F_d(n)$-放大器 $\mathcal{A}mpl_d=(P_d,X_d,(a',b',c'),(a,b,c),Z_d)$ 并考察其计数器的个数 $|X_d|$, 根据上述推论便可得到 d-VASS 可达性问题的相关下界. 当然, 后续研究证明这是可以做到的, 同时人们还探讨了使得 $D(d)$-VASS 可达性是 $\mathbf{F}_d$-难的函数 $D(d)$ 的形式. 显然 $D(d)$ 增长速度越慢, 对应的下界与上界越接近, 目前最新研究成果已将此优化为线性函数的形式, 相关结论可以总结如图 8 所示. 此图中的箭头表示技术之间的依赖关系. 总体而言, $F_d(n)$-放大器构造技术可以分为两条路线.

路线 1. Czerwiński 等人基于递归的构造方法.

路线 2. Leroux 的基于向量编码的构造方法.

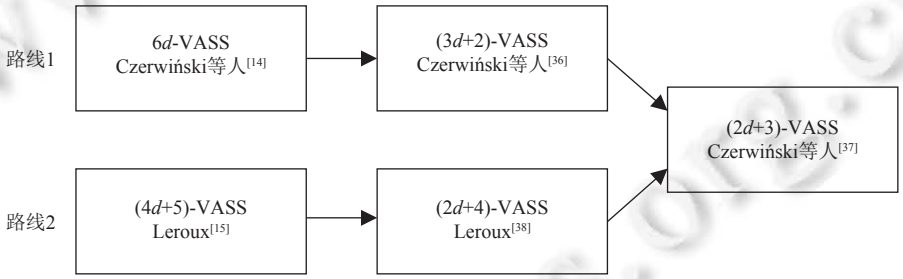


图 8 d-VASS 可达性问题复杂性下界

目前最好的构造集两者所长, 最终证明了 $(2d+3)$-VASS 可达性是 $\mathbf{F}_d$-难的. 我们将在第 3.3 和 3.4 节分别介绍这两条技术路线, 并在第 3.5 节给出最优下界的证明方法.

3.3 递归构造 $F_d(n)$-放大器

Czerwiński 等人[14]和 Leroux[15]的思路是根据 $F_d(n)$ 的定义式 (3)、(4) 递归地构造出 $F_d(n)$-放大器. 但为了方便构造, 可以考虑重新定义 $F_1(n)=2n$ 以及当 $k\geqslant 1$ 时 $F_{k+1}(n)=F_k^{(n)}(1)$, 这样得到的一族新的快速增长函数 $\{F_d(n)\}_{d\in\mathbb{N}}$ 和前文中的定义在量级上是一致的. 更具体地说, 构造过程有以下两步.

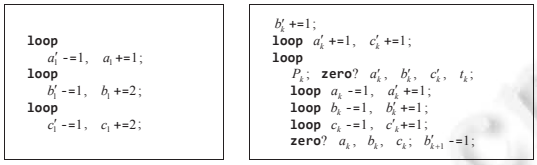
1) 构造一个 $F_1(n)$-放大器 $\mathcal{A}mpl_1$.

2) 假设已经得到了 $F_k(n)$-放大器 $\mathcal{A}mpl_k$, 通过 $\mathcal{A}mpl_k$ 构造出 $\mathcal{A}mpl_{k+1}$.

根据数学归纳法可知, 只要完成上述两步, 即可构造出一族 $F_d(n)$-放大器 $\{\mathcal{A}mpl_d\}_{d\in\mathbb{N}}$.

$F_1(n)$-放大器的构造是平凡的, 例如图 9 中程序 P_1 给出了 $\mathcal{A}mpl_1=(P_1,X_1,(a_1',b_1',c_1'),(a_1,b_1,c_1),Z_1)$, 其中, $X_1=\{a_1,b_1,c_1,a_1',b_1',c_1',t_1\}$, $Z_1=\{a_1',b_1',c_1',t_1\}$. 容易验证其满足放大器的性质, 因为 $F_1(n)=2n$, $\mathcal{A}mpl_1$ 是 $F_1(n)$-放大器. 这里根本没有使用到辅助计数器 t_1, 但为了后文叙述方便, 还是假设存在这样一个计数器.

假设对某个 $k\geqslant 1$, 我们已经得到了一个 $F_d(n)$-放大器 $\mathcal{A}mpl_k=(P_k,X_k,(a_k',b_k',c_k'),(a_k,b_k,c_k),Z_k)$, 其中, $Z_k=\{a_k',b_k',c',t_k'\}$, 需构造出一个 $F_{k+1}(n)$-放大器. 由于计数器程序表达能力有限, 直接构造难度较大, 不妨先构造一个测零计数器机器. 考察图 9 中的计数器机器 P'_{k+1}, 令 $Z_{k+1}=\{b'_{k+1}\}$, $X_{k+1}=X_k\cup\{a'_{k+1},b'_{k+1},c'_{k+1}\}$.



P_1 P'_{k+1}

图 9 $F_d(n)$-放大器的递归构造

● 假设某 Z_{k+1}-执行从格局乘法三元组 $\tau_s=(a'_{k+1},b'_{k+1},c'_{k+1},A_s,B,X_{k+1})$ 出发, 则前两行可以对任意 $A_0\in\mathbb{N}$ 构造乘法三元组 $\tau_0=(a_k',b_k',c_k',A_0,1,X_k)$; 因为 $\mathcal{A}mpl_k$ 是 $F_k(n)$-放大器, 对 Z_k 的测零保证执行 P_k 后得到 $\tau_0'=(a_k,b_k,c_k,A_0',F_k(1),X_k)$, 随后的 3 个循环配合对 a_k、b_k、c_k 的测零使得下一次执行 P_k 前格局为 $\tau_1=(a_k',b_k',c_k',A_1,F_k(1),X_k)$, 其中, $A_1=A_0'$; Z_{k+1}-执行保证了最终 P_k 被准确执行 B 次, 从而终止格局为 $\tau_B=(a_k',b_k',c_k',A_B,F_{k+1}(B),X_k)$.

● 相反, 给定 $A_B\in\mathbb{N}$, 由 $\mathcal{A}mpl_k$ 的性质知: 存在 $A_{B-1}\in\mathbb{N}$ 使得 Z_k-执行一次第 2 个 **loop** 语句的循环体可以从 $\tau_{B-1}=\left(a_k',b_k',c_k',A_{B-1},F_k^{(B-1)}(1),X_k\right)$ 计算出 $\tau_B=\left(a_k',b_k',c_k',A_B',F_k^{(B)}(1),X_k\right)$, 其中, $A_B'\geqslant A_B$. 不断向前推导可知, 最终存在 $A_0\in\mathbb{N}$ 使得 Z_k-执行 B 次相关语句可以实现从 $\tau_0=(a_k',b_k',c_k',A_0,1,X_k)$ 计算出 $\tau_B=(a_k',b_k',c_k',A_B',F_{k+1}(B))$ 并使得 $A_B'\geqslant A_B$. 若第 1 个 **loop** 恰好循环 A_0 次, 我们可以构造出 τ_0, 从而使得 P'_{k+1} 可以 Z_{k+1}-计算出 τ_B.

目前 P'_{k+1} 可以完成 $F_{k+1}(n)$-放大器的功能, 但因其使用了测零操作故不是计数器程序. 接下来的关键是构造计数器程序去模拟实现 P'_{k+1} 中的两处测零语句. 这需要引入更多新计数器, 也需要更多控制计数器 (注意, 目前 $Z_{k+1}=\{b'_{k+1}\}$). 接下来两节分别阐述基于控制计数器的方法和后续的优化.

3.3.1 控制计数器方法

本节介绍 Czerwiński 等人[14]是如何通过引入额外的控制计数器将图 9 中的计数器机器转换为计数器程序的. 此方法看似复杂, 其本质是基于一个简单的事实: 若干个非负整数之和为 0 当且仅当每个整数都为 0. 在第 4 节中, 我们还能看到该方法的其他相关应用.

考虑图 9 中的 P'_{k+1}, 其中对 a_k 等计数器进行了多次测零, 故考虑: 若对单计数器 a 进行了 k 次测零, 至多需要引入多少控制计数器才能完成模拟? 假设测零前格局分别为 $\alpha_0,\ldots,\alpha_{k-1}$, 则这次执行可以表示为:

$$\alpha_0\xrightarrow{\textbf{zero}?,+d_1}\alpha_1\xrightarrow{\textbf{zero}?,+d_2}\cdots\xrightarrow{\textbf{zero}?,+d_{k-1}}\alpha_{k-1}\xrightarrow{\textbf{zero}?}0 \tag{9}$$

平凡的方式是每次测零后不再修改 a 并引入一个副本 a', 将剩下对 a 的操作都转移到 a' 上. 但这样需要线性个计数器, 对下界的证明十分不利. 注意, 所有计数器的值都是非负的, 因此 $\sum_{i=0}^{k-1}\alpha_i(a)=0$ 当且仅当 $\alpha_i(a)=0$ 对 $0\leqslant i\leqslant k-1$ 都成立, 我们只需使用辅助计数器 t 记录 $\sum_{i=0}^{k-1}\alpha_i(a)$, 最终对 t 测零即可. 应用阿贝尔变换得到:

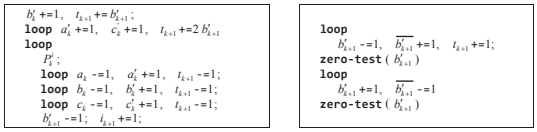
$$\sum_{i=0}^{k-1}\alpha_i(a)=k\alpha_0(a)+(k-1)d_1+(k-2)d_2+\ldots+d_{k-1} \tag{10}$$

因此可以仅使用一个额外的维度按如下方式完成记录:

$$\begin{pmatrix}\alpha_0\\ k\alpha_0(a)\end{pmatrix}\xrightarrow[+(k-1)d_1]{\textbf{zero}?,+d_1}\begin{pmatrix}\alpha_1\\ k\alpha_0(a)+(k-1)d_1\end{pmatrix}\xrightarrow[+(k-2)d_2]{\textbf{zero}?,+d_2}\cdots\xrightarrow[+d_{k-1}]{\textbf{zero}?,+d_{k-1}}\begin{pmatrix}\alpha_{k-1}\\ \sum_{i=0}^{k-1}\alpha_i(a)\end{pmatrix}\xrightarrow[\textbf{zero}?]{}\mathbf{0} \tag{11}$$

此方法被称作控制计数器方法 (controlling counter technique)[14], 并且可以推广到多个计数器测零次数已知的情形上.

对于 P'_{k+1}, 我们引入辅助计数器 t_{k+1}, 用 i_{k+1} 从 0 到 $B-1$ 记录当且循环的次数. 以 a_k 为例, 它一共进行了 B 次测零, 第 i 次循环时的 b_k −=1 位于第 i_{k+1} 次测零前, 因此需要修改 t_{k+1} 如下: t_{k+1} −= $B-i_{k+1}$, 其他计数器同理. P_k 中对 a_k 等计数器也有增减, 因此也需要在其中对 t_{k+1} 做出调整, 记第 i 次循环中调整后的程序为 P'^i_k, 最终可以初步得到图 10 中没有测零语句的程序 P_{k+1}.



P_{k+1} t_{k+1}+=b'_{k+1}

图 10 基于控制计数器方法完成 $\mathcal{A}mpl_{k+1}$ 的构造

观察可知, 执行 P'^i_k 时需要引入形如 t_{k+1} +=$d(B-i_{k+1})$ 的修改, 而此时有 b'_{k+1} 的值恰为 $B-i_{k+1}$, 只需令 t_{k+1} += db'_{k+1} 即可. 事实上, 常量 d 不会超过 2, 因此可以只通过重复 t_{k+1} += b'_{k+1} 操作 d 次实现. 但这显然不是计数器程序允许的语句格式, 需要替换成图 10 右侧的程序片段. 该片段中引入了计数器 $\overline{b'_{k+1}}$ 使得 $b'_{k+1}+\overline{b'_{k+1}}$ 之和不变. 当 P_{k+1} 的初始格局是 $\tau_s=(a'_{k+1},b'_{k+1},c'_{k+1},A_s,B,X_{k+1})$ 时, 因为不变量 $b'_{k+1}+\overline{b'_{k+1}}+i_{k+1}=B$ 始终成立, 符合使用 τ_s 模拟测零的前提, 因此 **zero-test** 可以和图 7 同理实现.

最终我们构造了一个 $F_{k+1}(n)$-放大器 $\mathcal{A}mpl_{k+1}=(P_{k+1},X_{k+1},(a'_{k+1},b'_{k+1},c'_{k+1}),(a'_k,b'_k,c'_k),Z_k)$. 其中, 输入计数器是一组新的计数器; 输出计数器复用了 $\mathcal{A}mpl_k$ 的输入计数器; $Z_{k+1}=\{a'_{k+1},b'_{k+1},c'_{k+1},t'_{k+1}\}$; 整体上使用了 $X_{k+1}=X_k\cup Z_{k+1}\cup\{i_{k+1},\overline{b'_{k+1}}\}$. P_{k+1} 的 Z_{k+1}-执行相当于 P'_{k+1} 的 $\{b'_{k+1}\}$-执行, 此处不再重新论证为何 $\mathcal{A}mpl_{k+1}$ 符合 $F_{k+1}(n)$-放大器的定义. 易知计数器总数符合递推关系式 $g(1)=|X_1|=6$ 和当 $k\geqslant 1$ 时 $g(k)=|X_k|=g(k-1)+6$, 解得对任意 $d\geqslant 3$ 都有 $g(d)=6d$. 程序的长度大致为 $\exp(d)$ 量级, 与 n 无关. 因此我们得到如下引理.

引理 11[14]. 当 $d\geqslant 3$ 时, 存在一族 $F_d(n)$-放大器 $\mathcal{A}mpl_d=(P_d,X_d,(a'_d,b'_d,c'_d),(a_d,b_d,c_d),Z_d)$ 使得其中程序 P_d 长度为 $2^{O(d)}$, $|X_d|=6d$, $|Z_d|\leqslant 4$.

结合推论 2 知, 当 $d\geqslant 3$ 时, $D=\max(|X_d|,2+|\{a_d,b_d,c_d\}\cup Z_d|)=\max(6d,7)=6d$, 因此有如下定理.

定理 8[14]. 当 $d\geqslant 3$ 时, $(6d)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

● 技术局限. 上述方法给出了一般 VASS 可达性问题的 **Ackermann** 完备性, 其重要性不言而喻. 但从维度的角度思考, 每当 d 增加 1, 我们便需要引入额外 6 个计数器用于计算, 直观上, 该方法仍有优化空间. 注意, 在使用放大器时, 我们保证了其初始格局中输入计数器构成乘法三元组, 例如图 9 中的 P'_{k+1}, 其初始格局是某个 $\tau_s=(a'_{k+1},b'_{k+1},c'_{k+1},A_s,B,X_{k+1})$. 但后续程序并未直接应用该三元组模拟测零, 事实上使用 τ_s 测零的前提如下.

1) 初始格局中 A_s 可以设定得充分大, 使之能够覆盖 P'_{k+1} 中测零次数的 2 倍.

2) 给定计数器上界 B 后, 后续执行所有格局中待测零计数器值之和都不超过 B.

而 P'_{k+1} 中待测零的计数器 b_{k+1} 最终会增长到 $F_{k+1}(B)$ 量级, 不可能满足条件 2). 文献 [14] 中采取的策略是使用控制计数器方法将测零转移到一些“小”计数器 b'_{k+1}, $\overline{b'_{k+1}}$ 上, 它们满足 $b'_{k+1}+\overline{b'_{k+1}}\leqslant B$, 最终用三元组 τ_s 配合 **zero-test** 模块即可完成模拟测零, 但代价是引入 3 个额外的控制计数器. 若能够想办法直接使用 τ_s 测零而不引入多余的计数器, 则最终可以将定理 8 的结论优化到 $3d$ 量级.

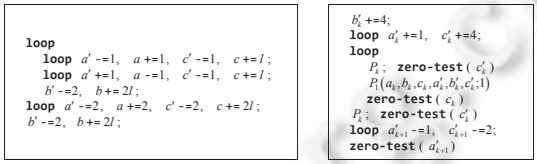
3.3.2 优化: 测零次数与计数器上界的交换

Czerwiński 等人[36]注意到任意三元组 $\tau_s=(a',b',c',A_s,B,X)$ 中 a', b' 的角色完全是对称的, 完全可以将 A_s 看作计数器值上界, B 用于覆盖测零次数的 2 倍. 观察程序 P'_{k+1}, 它一共进行了 $7B$ 次测零, 尽管没有被 B 覆盖, 但已经是线性关系, 只需对 P'_{k+1} 稍加改进减少测零次数即可; 另一方面待测零计数器的值都是有上界的, 可以通过将 A 设定得充分大使得所有待测零计数器值之和都不超过 A. 最终一个 $f(n)$-放大器可以将 τ_s 放大为 $\tau_t=(a,b,c,A_t,f(B),X)$. 在这种对乘法三元组新的解释下, τ_t 无法用于 $f(n)$-有界计数器机器的模拟, 因为 $f(B)$ 现在是对测零次数的限制. 我们转而考虑计数器机器 $f(n)$-时间可达性问题, 有如下引理.

引理 12. $d\geqslant 3$ 时, 若存在一族 $F_d(n)$-器 $\mathcal{A}mpl_d=(P_d,X_d,(a',b',c'),(a,b,c),Z_d)$ 使得其中程序 P_d 长度为 $p(n)\in\mathcal{F}_{<d}$, 令 $D=\max(|X_d|,2+|\{a,b,c\}\cup Z_d|)$, 则 D-VASS 可达性问题是 $\mathbf{F}_d$-难的.

证明: 与引理 10、推论 2 同理, 只需将归约问题改为计数器机器 $F_d(n)/2$-时间可达性问题, 并对调模拟和结束阶段中三元组内 a、b 的角色: a 用于维持不变量 $x+y+a=A$; b 用于记录测零次数, 每次减 2.

为了解决三元组只能进行 $B/2$ 次测零, 但 P'_{k+1} 中却需要进行 $\geqslant B$ 次测零的问题, Czerwiński 等人[36]使用了快速增长函数的另一等价定义 (等价性证明见原文): $F_1(n)=2n$ 以及 $k\geqslant 1$ 时 $F_{k+1}(n)=F_k^{(n/4)}(4)$. 最终得到一族 $F_d:\mathbb{N}_4\to\mathbb{N}_4$, 这里, $\mathbb{N}_4$ 表示 4 的自然数倍数. 函数族的定义域限制并不影响引理 12 的正确性, 因为将 n 替换成为稍大的 $\mathbb{N}_4$ 中的数是容易的. 本节后续证明中仅考虑满足 $B\in\mathbb{N}_4$ 的乘法三元组. 同时, 为了减少 Z_k 中的计数器数量, 我们必须从 Z_1 开始优化. 定义计数器程序 $P_1(a',b',c',a,b,c;l)$ 如图 11 所示.



$P_1(a',b',c',a,b,c;l)$ P_{k+1}

图 11 优化后的 $F_d(n)$-放大器

设 $X=\{a,b,c,a',b',c'\}$, 容易验证 $l=2$ 时即可得到一个 $F_1(n)$-放大器 $\mathcal{A}mpl_1=(P_1,X_1,(a'_1,b'_1,c'_1),(a_1,b_1,c_1),\{c'_1\})$, 其中, $P_1=P_1(a'_1,b'_1,c',a,b,c;2)$, $X_1=\{a_1,b_1,c_1,a'_1,b'_1,c'\}$.

假设我们已经得到了一个 $F_k(n)$-放大器 $\mathcal{A}mpl_k=(P_k,X_k,(a'_k,b'_k,c'_k),(a_k,b_k,c_k),Z_k)$, 其中, $Z_k=\{c'_k\}$, 构造 P_{k+1} 如图 11 所示. 程序中的测零使用 a'_{k+1}, b'_{k+1}, c'_{k+1} 构成的三元组模拟, 需要对其中的加法语句进行适当的互补操作, 此处隐去了这些操作. 令 $X_{k+1}=X_k\cup\{a'_{k+1},b'_{k+1},c'_{k+1}\}$, $Z_{k+1}=\{c'_{k+1}\}$.

- 假设初始格局为 $\tau_s=(a'_{k+1},b'_{k+1},c'_{k+1},A_s,B,X_{k+1})$, 将任意从 τ_s 出发的 Z_{k+1}-执行限制在计数器集合 X_k 上, 第 1 个 **loop** 语句对某个 A_1 构造了三元组 $\tau_0=(a'_k,b'_k,c'_k,A_0,4,X_k)$. 第 2 个 **loop** 语句对 c'_k 的模拟测零保证了该执行对应 P_k 的一次 Z_k-执行, 因此将 τ_0 放大到了 $\tau'_0=(a_k,b_k,c_k,A_1,F_k(4),X_k)$, 其中, A_1 是某个自然数; 对 c_k 的模拟测零保证了该执行对应 $P_1(a_k,b_k,c_k,a'_k,b'_k,c';1)$ 的一次 $\{c_k\}$-执行, 得到 $\tau_1=(a'_k,b'_k,c',A_1,F_k(4),X_k)$. 由于 $B=0\ (\mathrm{mod}\ 4)$, 乘法三元组 τ_s 保证一次 Z_{k+1}-执行中恰好能够进行 $B/2$ 次模拟测零, 从而 P_k 可以循环执行 $B/4$ 次, 最终得到 $\tau_t=(a_k,b_k,c_k,A_{B/4},F_k^{(B/4)}(4),X_{k+1})$.
- 若给定 $A\in\mathbb{N}$, 其中, $B\in\mathbb{N}_4$, 借用 $\mathcal{A}mpl_k$ 的性质不断倒推可知: 存在 $A_s\in\mathbb{N}$, 使得 P_{k+1} 能从 $\tau_s=(a'_{k+1},b'_{k+1},c'_{k+1},A_s,B,X_{k+1})$ Z_{k+1}-计算出某个 $\tau_t=(a_k,b_k,c_k,A,F_{k+1}(B),X_{k+1})$ 使得 $A_t\geqslant A$.

综上, $\mathcal{A}mpl_{k+1}=(P_{k+1},X_{k+1},(a'_{k+1},b'_{k+1},c'_{k+1}),(a_k,b_k,c_k),Z_{k+1})$ 是一个 $F_{k+1}(n)$-放大器. 由构造过程可知: $|X_d|=3d+3$, $|Z_d|=1$, 程序长度为 $2^{\mathrm{O}(d)}$, 从而有如下引理.

引理 13[36]. $d\geqslant 3$ 时, 存在一族 $F_d(n)$-放大器 $\mathcal{A}mpl_d=(P_d,X_d,(a'_d,b'_d,c'_d),(a_d,b_d,c_d),Z_d)$ 使得其中程序 P_d 长度为 $2^{\mathrm{O}(d)}$, $|X_d|=3d+3$, $|Z_d|=1$.

当 $d\geqslant 3$ 时, $D=\max(|X_d|,2+|\{a_d,b_d,c_d\}\cup Z_d|)=\max(3d+3,4)=3d+3$. 注意, b'_d 只会减少, 因此在图 7 的 P' 中可以使用 $n,n-1,\ldots,0$ 替换每次 b'_d 的出现, 从而减少一个计数器, 代价是程序长度变为 $n2^{\mathrm{O}(d)}$. 因此有如下定理.

定理 9[36]. $d\geqslant 3$ 时, $(3d+2)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

相较于定理 8, 这是一个更好的下界. 事实上, 在不复用任何一组三元组中的计数器的情况下, 计数器数量必然有 $3d+\Omega(1)$ 的下界, 此方法已经几乎对应此时的最优下界了.

3.4 快速增长函数的向量编码

本节介绍 $F_d(n)$-放大器的构造路线 2: Leroux[15,30]的基于向量编码的构造方法. 与第 3.3 节中阐述的思路不同, 2021 年 Leroux[15]在构造 $F_d(n)$-放大器的过程中巧妙地使用向量对一般形式的快速增长函数进行编码, 同时用程序

模拟函数求值的过程. 此方法虽然较为复杂, 但提供了更广阔的思路, 同时得到的下界相对而言更好. 本节将对这一方法做详细介绍.

对于向量 $\boldsymbol{u}\in\mathbb{N}^d$, 定义如下一般形式的快速增长函数:

$$F_{\boldsymbol{u}}=F_d^{(\boldsymbol{u}(d))}\circ F_{d-1}^{(\boldsymbol{u}(d-1))}\circ\ldots\circ F_1^{(\boldsymbol{u}(1))} \tag{12}$$

即 $\boldsymbol{u}$ 的第 i 个元素对应公式 (12) 中 F_i 复合的次数. 于是, 元组 $(\boldsymbol{u},n)$ 可以用于编码唯一的函数值 $F_{\boldsymbol{u}}(n)$, 我们将这个映射关系记作 $\mathcal{V}al$:$\mathbb{N}^d\times\mathbb{N}\to\mathbb{N},(\boldsymbol{u},n)\mapsto F_{\boldsymbol{u}}(n)$. 显然 $\mathcal{V}al$ 不是单射, 例如 $\mathcal{V}al(\boldsymbol{e}_d,n)=F_{\boldsymbol{e}_d}(n)=F_d(n)$, $\mathcal{V}al(\boldsymbol{0}_d,F_d(n))=\mathcal{I}d(F_d(n))=F_d(n)$, 其中, $\mathcal{I}d:\mathbb{N}\to\mathbb{N}$ 是恒等函数. 因此 $(\boldsymbol{e}_d,n)$ 和 $(\boldsymbol{0}_d,F_d(n))$ 都是值 $F_d(n)$ 的有效编码, 二者的区别在于 $(\boldsymbol{e}_d,n)=n$, 因此是可以高效构造的编码; $(\boldsymbol{0}_d,F_d(n))=F_d(n)$ 是个大数, 无法直接构造但却可以在乘法三元组中直接使用. 下面我们将这种编码记作 $\lambda=(\boldsymbol{u},n):[d]_0\to\mathbb{N}$, 并且定义 $\lambda(0)=n$ 以及当 $i\in[d]$ 时 $\lambda(i)=\boldsymbol{u}(i)$, 这相当于将 n 看作 λ 的第 0 维. 利用这种编码, 我们可以构造 $F_d(n)$-放大器如下.

1) 假设初始格局为 $\tau_s=(a',b',c',A_s,B,X)$, 构造编码 $\lambda_0=(\boldsymbol{e}_d,B)$.

2) 构造一段计数器程序将一个给定编码 λ 转换为等价的编码 λ' 使得 $\lambda'<_{\text{lex}}\lambda$.

3) 不断执行上述程序直到无法继续, 完成转换: $(\boldsymbol{e}_d,B)=\lambda_0\to\lambda_1\to\ldots\to\lambda_k=(\boldsymbol{0}_d,F_d(B))$, 根据最终得到的 λ_k 构造三元组 $\tau_t=(a,b,c,A_t,F_d(B),X)$.

步骤 2) 中使用的序关系 $\leqslant_{\text{lex}}$ 是字典序, $\lambda'\leqslant_{\text{lex}}\lambda$ 当且仅当最大的使得 $\lambda'(i)\neq\lambda(i)$ 的下标 i 满足 $\lambda'(i)\leqslant\lambda(i)$. 显然 0_d 是所有 d 维向量中字典序最小的, 这也是步骤 3) 的转换能够终止的原因. 整个过程的关键是如何找到合适的变换并且构造对应的程序.

给定 $\lambda=(\boldsymbol{u},x)$, 若 $\boldsymbol{u}$ 的元素全为 0, 则 $\mathcal{V}al(\lambda)=x$, 无法也不需要继续进行变换, 这种情况我们称作是平凡的. 若 λ 非平凡, 则可定义 $\text{index}(\lambda)=\min\{i\in[d]:\boldsymbol{u}(i)\neq 0\}$, 不妨设 $\text{index}(\lambda)=i_0$, 则 $\mathcal{V}al(\boldsymbol{u},x)=F_d^{(\boldsymbol{u}(d))}\circ F_{d-1}^{(\boldsymbol{u}(d-1))}\circ\ldots\circ F_{i_0}^{(\boldsymbol{u}(i_0))}(x)$, 对复合序列的末端分离出一个 F_{i_0} 使用快速增长函数的定义有:

$$\mathcal{V}al(\boldsymbol{u},x)=F_d^{(\boldsymbol{u}(d))}\circ F_{d-1}^{(\boldsymbol{u}(d-1))}\circ\ldots\circ F_{i_0}^{(\boldsymbol{u}(i_0)-1)}\left(F_{i_0-1}^{(x+1)}(x)\right) \tag{13}$$

当 $i_0\geqslant 2$ 时, 上述结果说明了 $\mathcal{V}al(\boldsymbol{u},x)=\mathcal{V}al(\boldsymbol{u}-\boldsymbol{e}_{i_0}+(x+1)\boldsymbol{e}_{i_0-1},x)$. 定义 $\varepsilon_0=(\boldsymbol{0}_d,1)$; 对 $i\in[d]$, 定义 $\varepsilon_i=(\boldsymbol{e}_i,0)$, 则上述结果可以统一写作:

$$\mathcal{V}al(\lambda)=\mathcal{V}al(\lambda-\varepsilon_{i_0}+(\lambda(0)+1)\varepsilon_{i_0-1}) \tag{14}$$

上述讨论定义了一个确定的变换过程 $\mathcal{E}val:\mathbb{N}^d\times\mathbb{N}\to\mathbb{N}^d\times\mathbb{N}$, $\lambda\mapsto\lambda-\varepsilon_{i_0}+(\lambda(0)+1)\varepsilon_{i_0-1}$, 其中, $i_0=\text{index}(\lambda)$. 观察可知: 首先对 $i>i_0$ 都有 $\mathcal{E}val(\lambda)(i)=\lambda(i)$ 但 $\mathcal{E}val(\lambda)(i_0)<\lambda(i_0)$, 从而 $\mathcal{E}val(\lambda)<_{\text{lex}}\lambda$; 其次, $\mathcal{V}al(\lambda)=\mathcal{V}al(\mathcal{E}val(\lambda))$, 从而变换 $\mathcal{E}val$ 保持编码的含义不变. 对 $\lambda_0=(\boldsymbol{e}_d,B)$ 不断应用该变换即可得到迁移: $\lambda_0\xrightarrow{\mathcal{E}val}{}^{*}\ \mathcal{E}val^{(k)}(\lambda_0)=(\boldsymbol{0}_d,F_d(B))$, 其中, k 是一个由 λ_0 唯一确定的自然数. 我们无须关心 k 的真实值, 只需使用程序 **eval** 实现变换 $\mathcal{E}val$, 并执行 **loop eval** 从 λ_0 出发得到 $\lambda'=(\boldsymbol{v},y)$, 最终检查是否有 $\boldsymbol{v}=\boldsymbol{0}_d$ 即可.

3.4.1 $\mathcal{E}val$ 的计数器程序实现

从上述讨论中可以看出, $\mathcal{E}val$ 的计数器程序实现是此方法的核心. 本节将介绍 Leroux[15]基于乘法形式不变量的构造方法, 并简要分析该方法的局限性与改进方向. 另外, Leroux 原证明因缺少对特殊情况的讨论而有些许瑕疵, 在后文中我们予以纠正.

首先, 我们需要使用若干个计数器保存一个给定编码 λ. 我们显然不能只令 $b_i=\lambda(i)$, 否则在非确定地修改 b_i 时很容易丢失原有的 $\lambda(i)$ 值. 根据上文介绍的相关技术, 比较容易想到的方法是使用一对计数器维护 $b_i+\overline{b_i}=\lambda(i)$. 同时因为三元组 (a,b,c,A,B,X) 中需要满足 $c=ab$, 因此对合适的 A, 我们需要记录 $A\lambda(j)$ 的值. Leroux[15]的方法是: 对 $i\in[d]_0$ 构造 $b_i+\overline{b_i}=\lambda(i)$, 对 $i\in[d+1]_0$ 构造 $a_i+\overline{a_i}=A_i$ 且保持对任何 $i\in[d]_0$ 都有 $A_{i+1}=(1+\lambda(i))A_i$. 这里因子设定为 $\lambda(i)+1$ 而非 $\lambda(i)$ 的用意在于: 若某个 $\lambda(i)$ 为 0, 则无论 A_i 是多少, 使用关系式 $A_{i+1}=\lambda(i)A_i$ 将导致对任意 $j>i$ 都有 $A_j=0$, 因此一组 $\{A_i\}_{i\in[d+1]_0}$ 即便满足 $A_0\neq 0$ 也可能对应多组 λ, 这造成了信息的丢失; 使用关系式 $A_{i+1}=(1+\lambda(i))A_i$ 保证了若 A_0 不为 0, 则对任意 λ 和 $j\in[d+1]$ 都有 $A_j\neq 0$, 因此一组满足 $A_0\neq 0$ 的 $\{A_i\}_{i\in[d+1]_0}$ 至多只能对

应一个 λ.

给定一个上述计数器的格局 α, 用 $\alpha(A_i)$ 表示 $\alpha(a_i+\overline{a_i})$, $\alpha(B_j)$ 表示 $\alpha\left(b_j+\overline{b_i}\right)$. 若 α 对任何 $i\leqslant j\leqslant d$ 都有 $\alpha(A_{j+1})=(1+\alpha(B_j))\alpha(A_j)$, 则称其为 i-好的; 若 α 是 $(i+1)$-好的, 但是 $\alpha(A_{i+1})\geqslant(1+\alpha(B_i))\alpha(A_i)$, 则称其为 i-坏的; 若 α 是 i-好的或对于某个 $j\geqslant i$ 是 j-坏的, 则称 α 是 i-合法的; 若 α 满足对 $i\in[d]_0$ 都有 $\alpha(B_i)=\lambda(i)$, 则称其为对应 λ 的格局. 若 α 是 0-好的或 0-合法的, 则可以简称为好的或合法的.

假设是对应非平凡编码 λ 的好格局, 令 $i_0=\mathrm{index}(\lambda)$, 为了从 α 得到一个新的好格局 β 对应编码 $\mathcal{E}val(\lambda)$, 我们先考察 β 在 $\{B_i\}_{i\in[d]_0}$ 上应当满足的条件:

$$\beta(B_i)=\alpha(B_i)+\begin{cases}-1, & i=i_0\\ 1+\alpha(B_0), & i=i_0-1\\ 0, & \text{其他情形}\end{cases}\tag{15}$$

公式 (15) 是根据公式 (14) 唯一确定的. β 在 $\{A_i\}_{i\in[d+1]_0}$ 上的行为不是唯一的, 不妨令 $\beta(A_{i_0})=\alpha(A_{i_0})$, 即不对 A_{i_0} 的值做修改, 其他值可以通过不变量推导唯一确定:

$$\beta(A_i)=\begin{cases}\dfrac{\alpha(A_0)(1+\alpha(B_{i_0-1}))}{2+\alpha(B_{i_0-1})+\alpha(B_0)}, & i=0\\ \dfrac{\alpha(A_1)}{2+\alpha(B_0)}, & 1\leqslant i\leqslant i_0-1\\ \alpha(A_1), & i=i_0\\ \alpha(A_i)-\alpha(A_1)\prod_{j=i_0+1}^{i-1}(1+\alpha(B_j)), & i_0+1\leqslant i\leqslant d+1\end{cases}\tag{16}$$

注意, 根据 i_0 的定义, $\alpha(B_1)=\ldots=\alpha(B_{i_0-1})=0$, 因此 $\alpha(A_{i_0})=\ldots=\alpha(A_1)$, 公式 (16) 中统一用 $\alpha(A_1)$ 表示. 显然计算 β 不是一件容易的事, 在正式构造这样的程序前, 我们需要设计一些基本的程序模块.

● 若计数器 a、$\overline{a}$ 满足 $a+\overline{a}=A$, 需实现操作 A −=1, 只需替换为模块“**dec**(A): **do** a −=1 **or** $\overline{a}$ −=1”.

对应地, 我们也可以定义“**inc**(A): **do** a +=1 **or** $\overline{a}$ +=1;”. 对多个计数器“ $a_1,a_2,\ldots$”加减 1 也可以简写作 **inc**$(a_1,a_2,\ldots)$ 和 **dec**$(a_1,a_2,\ldots)$.

● 若计数器 a、$\overline{a}$ 满足 $a+\overline{a}=A$, 有时我们需要将 A 完全转移到计数器 a 上, 这时可以使用模块“**reset**(A): **loop** a +=1, $\overline{a}$ −=1;”.

● 有时我们需要控制 **loop** 循环体被执行的最大次数, 可以使用如图 12 所示的程序实现.

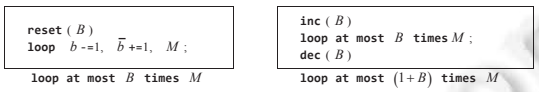


图 12 控制循环的最大次数

若 $b+\overline{b}=B$, 则左侧程序可控制 M 执行不超过 B 次, 同时不影响其他计数器; 右侧程序则可控制 M 执行不超过 $(1+B)$ 次. 因为不变量中有形如 $(1+B_j)$ 的因子, 这段程序在后续构造中使用频繁.

● 若修改了某个 A_i 的值, 如 A_i −=1, 仅使用模块 **dec**(A_i) 替换是不够的. 为了维持不变量 $A_{i+1}=(1+B_i)A_i$, 我们至少还需要同时实现 A_{i+1} −=$(1+B_i)$. 对于 $i\in[d+1]_0$, 令 **rdec**$_{d,i}$ 表示这一程序模块, 我们可以归纳地给出定义: **rdec**$_{d,d+1}$ 只需定义为 **dec**(A_{d+1}) 即可; 对于其他 $i\in[d]_0$, 对应的程序如图 13 所示. 实现 **rdec**$_{d,i}$ 的过程中调用了 **rdec**$_{d,i+1}$, 最终 **rdec**$_{d,0}$ 是 $\mathrm{O}(d)$ 长的.

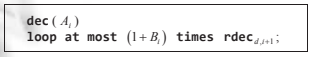
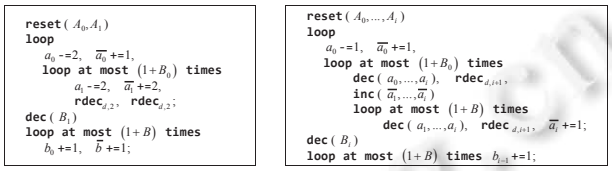


图 13 **rdec**$_{d,j}$

定义程序 **eval**$_d$ 用于实现变换 $\mathcal{E}val$, 由上述分析可知, 我们需要根据当前格局 α 的 $\mathrm{index}(\lambda)$ 确定应该进行什么操作, 为此我们可以构造一族 **eval**$_{d,i}$ 分别对应 $\mathrm{index}(\lambda)=1,\ldots,d$ 的情形, 最后令:

$$\mathbf{eval}_d = \mathbf{do}\ \mathbf{eval}_{d,1}\ \mathbf{or}\ \mathbf{eval}_{d,2}\ \mathbf{or}\ \ldots\ \mathbf{or}\ \mathbf{eval}_{d,d};$$

即可. 当然这里要求每次从 α 出发执行 $\mathbf{eval}_d$ 都能选择到对应的 $\mathbf{eval}_{d,\mathrm{index}(\lambda)}$, 这需要由 $\{\mathbf{eval}_{d,i}\}_{i\in[d]}$ 这一族程序模块的性质来保证. 回顾公式 (15) 和 (16) 可知, 在 $\mathrm{index}(\lambda)=1$ 和 $\mathrm{index}(\lambda)>1$ 两种情况下, $\beta(B_0)$ 分别为 $\alpha(B_0)$ 和 $1+2\alpha(B_0)$, 因此构造 $\mathbf{eval}_{d,i}$ 时我们将这两种情况区分开来. 注意, $i=1$ 时有 $\beta(B_0)=1+2\alpha(B_0)$ 而非保持不变, 因此文献 [15] 中对此情况的构造是错误的. 幸运的是, 我们可以通过重新设计程序弥补这一缺陷. 最终完整程序如图 14 所示. 这里因为需要两重嵌套同时实现 **loop at most** $(1+B_0)$ **times**, 因此额外引入了计数器 b、$\overline{b}$, 使得 $b+\overline{b}=B=B_0$, 另一层使用 **loop at most** $(1+B)$ **times** 替代. 但每次对 B_0 进行增减操作时, 都需要对 B 同步.



i=1 $\quad$ $i\geqslant 2$

图 14 $\quad$ $\mathbf{eval}_{d,i}$ 的两种不同情形

若 $\mathbf{eval}_{d,i_0}$ 的某次执行从对应非平凡 λ 的好格局 α 出发且每次循环都执行到了允许的最大次数后得到 β, 令 $i_0=\mathrm{index}(\lambda)$. 当 $i_0\geqslant 2$ 时, 若 $2+\alpha(B_0)$ 整除 $\alpha(A_0)$, 则有如下结论.

- 最外层循环每次将 a_0 减少 $2+\alpha(B_0)$, 将 $\overline{a_0}$ 加 1, 最多执行 $\dfrac{\alpha(A_0)}{2+\alpha(B_0)}$ 次, 且最终 $\beta(A_0)=\dfrac{\alpha(A_0)}{2+\alpha(B_0)}$.
- 第 2 重循环最多执行 $\dfrac{\alpha(A_1)}{2+\alpha(B_0)}$ 次, 每次执行 $2+\alpha(B_0)$ 次 $\mathbf{rdec}_{d,i_0+1}$, 最终 $\beta(A_{i_0+1})=\alpha(A_{i_0+1}-A_1)$.
- 最内层循环最多执行 $\dfrac{\alpha(A_1)(1+\alpha(B_0))}{2+\alpha(B_0)}$ 次, 每次 $a_1,\ldots,a_{i_0-1}$ 减 1, 从而 $\beta(A_i)=\dfrac{\alpha(A_1)}{2+\alpha(B_0)}$, $1\leqslant i<i_0$.
- $\beta(B_{i_0})=\alpha(B_{i_0})-1$, $\beta(B_{i_0-1})=1+\alpha(B_0)$.

显然, $\beta(A_{i_0})=\alpha(A_{i_0})$, 因此 $i\leqslant i_0+1$ 时, 计算结果与公式 (16) 一致, $i\geqslant i_0+1$ 时, $\mathbf{rdec}_{d,i_0+1}$ 可以保证 β 是 (i_0+1)-好的, 因此 β 是对应编码 $\mathcal{E}val(\lambda)$ 的好格局.

当 $i_0=1$ 时, 同理可知, 最终 $\beta(A_2)=\alpha(A_2-A_1)$; $\beta(B_1)=\alpha(B_1)-1$, $\beta(B_0)=1+2\alpha(B_0)$. 若 2 整除 $\alpha(A_0)$, 则 β 是对应编码 $\mathcal{E}val(\lambda)$ 的好格局. 另外, Leroux[15]证明了如下引理.

引理 14[15]. 若 α 是一个对应非平凡编码 λ 的合法格局, 执行 $\mathbf{eval}_{d,i}$ 后得到的 β 也是合法的; 若 β 是好的, 则 α 也是好的, 并且 $i=\mathrm{index}(\lambda)$, β 对应编码 $\mathcal{E}val(\lambda)$.

此引理的证明只需假设从上到下 4 个循环分别执行了 l、m、n、r 次, 根据程序推导出如下关系:

$$m\leqslant(1+\alpha(B_0))l,\ n\leqslant(1+\alpha(B_0))m,\ l+m\leqslant\alpha(N_0),\ m+n\leqslant\min_{j\in[i]}\alpha(X_j),\ r\leqslant 1+\alpha(B_0) \tag{17}$$

随后对 α 和 β 进行分析即可, 由于步骤比较繁杂, 在此不再赘述. 值得注意的是, 关系 $m+n\leqslant\min_{j\in[i]}\alpha(X_j)$ 是必要的, 这也解释了为何程序中总是出现形如 a_i −=1, $\overline{a_i}$ +=1 的语句: 尽管它们对 A_i 没有作用效果, 但却可以限制循环的次数. 上述分析可以总结成如下引理.

引理 15. 程序 $\mathbf{eval}_d$ 满足如下两条性质.

1) 若 α 是对应非平凡编码 λ 的好格局且 $\alpha(A_0)$ 满足某些整除性条件时, 则存在 $\alpha\xrightarrow{\mathbf{eval}_d}\beta$ 使得 β 是对应 $\mathcal{E}val(\lambda)$ 的好格局.

2) 若 α 是对应非平凡编码 λ 的合法格局且 $\alpha\xrightarrow{\mathbf{eval}_d}\gamma$, 则 γ 也是合法格局; 若 γ 是好的, 则 α 也是好的且 γ 对应编码 $\mathcal{E}val(\lambda)$.

Leroux[38]并没有严格给出一个放大器, 而是构造了一个 $(1+F_{d+1}(n))$-生成器 $\mathcal{G}en_d=(P_d,X_d,(a,b,c),Z_d)$, 其功能相当于一个无须输入计数器, 而是将 n 直接写进程序 P_d 中的放大器. 其中, X_d 包括上述所有的计数器 $\{b_i,\overline{b_i}\}_{i\in[d]_0}$、$\{a_i,\overline{a_i}\}_{i\in[d+1]_0}$、$b$、$\overline{b}$ 和新计数器 a、c; $Z_d=\{b_i,\overline{b_i}\}_{i\in[d]_0}\cup\left\{a_{d+1},\overline{a_{d+1}},\overline{b}\right\}$. a、c 分别用于记录 A_0、A_1, 即对 A_0、A_1 的每次

增减都需要同时作用在 a、c 上. 记 $\mathbf{eval}'_d$ 为引入对 a、c 同步操作后的 $\mathbf{eval}_d$, 程序 P_d 如图 15 所示.

考虑 P_d 的某个 Z_d -执行, 观察可知: 执行完前 3 行之后得到了一个对应 $\lambda_0=((n+1)\boldsymbol{e}_d,n)$ 的好格局 α_0, 且 $\alpha_0(A_0)\neq 0$. 设执行 $\mathbf{loop}\ \mathbf{eval}'_d$ 后得到 β, 执行 $\mathbf{loop}\ \mathbf{rdec}_{d,0}$ 后得到 γ. Z_d -执行保证了 $\gamma(B_1)=\ldots=\gamma(B_d)=\gamma(A_{d+1})=0$, 因此 γ 是个好格局, 根据 $\mathbf{rdec}_{d,0}$ 的性质可推断出 β 也是好的. 应用引理 15 得: β 对应 $\lambda_k=\mathcal{E}val^{(k)}(\lambda_0)=(\mathbf{0}_d,\beta(B_0))$, 其中, k 是 $\mathbf{eval}'_d$ 执行的次数. 根据变换 $\mathcal{E}val$ 的定义可知: $\beta(B_0)=\mathcal{V}al(\lambda_k)=\mathcal{V}al(\lambda_0)=F_d^{(n+1)}(n)=F_{d+1}(n)$. 最终 a, b, c 分别记录了 $\beta(A_0)$, $\beta(B_0)+1$, $\beta(A_1)$, 终止格局为一个乘法三元组 $\tau_t=(a,b,c,\beta(A_0),1+F_{d+1}(n))$.

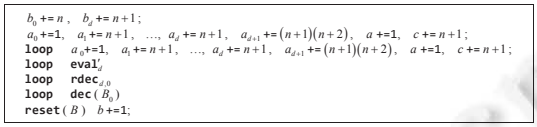


图 15 使用 $\mathbf{eval}_d$ 构造 P_d

由于 $|X_d|=4d+10$, $|Z_d|=2d+5$, 根据推论 2 可知: $(4d+10)$ -VASS 的可达性问题是 $\mathbf{F}_{d+1}$-难的. 注意到 $\overline{a_{d+1}}$ 是个全程没有使用过的计数器, 可以去掉, 最终有如下定理.

定理 10. $d\geqslant 3$ 时, $(4d+5)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

● 技术局限. 上述方法的瓶颈在于需要构造两组量 $\{B_i\}_{i\in[d]_0}$, $\{A_i\}_{i\in[d+1]_0}$, 每个量需要两个计数器维持, 因此整体需要 $4d+\mathrm{O}(1)$ 个计数器. 但核心程序 $\mathbf{eval}_d$ 只使用了 $\mathbf{loop\ at\ most}\ (1+B_0)\ \mathbf{times}$ 语句, 并没有要求对 $i\geqslant 1$ 维持 $b_i+\overline{b_i}=B_i$. 事实上, 后者是 $\mathbf{rdec}_{d,i}$ 程序的要求, 而 $\mathbf{rdec}_{d,i}$ 用于维持不变量 $A_{i+1}=(1+B_i)A_i$, 若能找到新不变量使得 B_i −=1 后对 A_{i+1} 的副作用与 B_i 无关, 则 $i\geqslant 1$ 时便不再需要 $\overline{b_i}$, 从而节省 $d+\Theta(1)$ 个计数器.

3.4.2 优化: 指数形式不变量

为了尽可能地复用计数器, Leroux 在文献 [38] 中使用了新不变量: 对 $i\in[d]_0$, 维持 $A_{i+1}=2^{B_i}A_i$. 每次 B_i −=1 后只需令 A_{i+1} /=2 即可, 不再与 B_i 具体的值有直接关联. 另外, 他还使用了一个全局计数器 c 来维持 $a_i+c=A_i$, 使得原来所有的 $\overline{a_i}$ 均可以舍去, 最终只需使用 $2d+\mathrm{O}(1)$ 个计数器. 本节将对此做详细介绍.

首先, 我们引入必要的计数器: 对 $i\in[d]$, 令 $b_i=B_i$; 令 $b_0+\overline{b_0}=B_0$; 对 $i\in[d+1]$, 令 $a_i+c=A_i$; 令 $a_0+\overline{a_0}=A_0$; 额外引入 $b+\overline{b}=B$. 令 α 是这些计数器的一个格局, 给定 λ, 若 α 满足对任意 $i\in[d]_0$ 都有 $\alpha(A_{i+1})=2^{\alpha(B_i)}\alpha(A_i)$, 且 $\alpha(B_i)=\lambda$, $\alpha(B)=2^{\alpha(B_0)}$, $\alpha\left(\overline{b_0}\right)=\alpha(c)=\alpha(\overline{a_0})=\overline{b}=0$, 则称 α 是对应 λ 的好格局. 同理, 可以定义什么样的格局是坏的或合法的. 若 λ 非平凡, $i_0=\mathrm{index}(\lambda)$, 设 β 是一个从 α 得到的对应 $\mathcal{E}val(\lambda)$ 的好格局, 则 β 依然需要满足公式 (15). 至于 β 在 $\{A_j\}_{j\in[d+1]_0}$ 上, 为使改动尽量少, 我们不妨令 $\beta(X_{d+1})=\alpha(X_{d+1})$, 计算可知:

$$\beta(A_i)=\begin{cases}A_0 2^{-\alpha(B_0)}, & i=0\\ \alpha(A_0), & 1\leqslant i\leqslant i_0-1\\ 2\alpha(A_1), & i=i_0\\ \alpha(A_i), & i_0+1\leqslant i\leqslant d+1\end{cases}\tag{18}$$

根据公式 (18) 构造 $\mathbf{eval}_{d,i}$, 如图 16 所示.

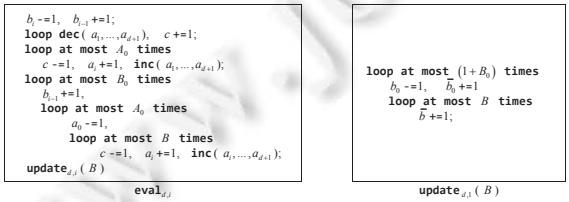


图 16 新不变量下 $\mathbf{eval}_{d,i}$ 的构造

图 16 中, $\mathbf{loop\ at\ most}\ A_0\ \mathbf{times}$ 等模块与第 3.4.1 节定义一致. 在构造 $\mathbf{eval}_{d,i}$ 时使用了 $\mathbf{update}_{d,i}(B)$, 作用是更新 B 的值, 但易知只有当 $i_0=1$ 时需要修改 B, 因此对 $i>1$, $\mathbf{update}_{d,i}(B_0)$ 是一段空程序; 对 $i=1$, $\mathbf{update}_{d,1}(B_0)$ 如

图 16 右侧所示.

给定 λ 和 i_0, 若 $\mathbf{eval}_{d,i_0}$ 的某次执行从对应 λ 的好格局 α 出发且每次循环都执行到了允许的最大次数后得到 β, 且 $2^{\alpha(B_0)}$ 整除 $\alpha(A_0)$ 时, 不难得到如下结论.

- $i > i_0$ 时, 每次对 y 的修改都伴随着对 a_i 的修改, 因此 $\beta(A_i) = \alpha(A_i)$.
- 第 1 个循环最多执行 $\alpha(A_1)$ 次后得到中间格局 $\alpha_1(c) = \alpha(A_1)$.
- 第 2 个循环至多执行 $\alpha(A_0)$ 次后得到中间格局 $\alpha_2(c) = \alpha(A_1 - A_0)$, $\alpha_2(A_i) = \alpha(A_1 + A_0)$.
- 第 3 个循环至多执行 $\alpha(B_0)$ 次, 因此 $\beta(B_{i-1}) = \alpha(B_{i-1}) + \alpha(B_0) + 1$, $\beta(B_i) = \alpha(B_i) - 1$, 符合公式 (15).
- 第 4 个循环依次执行 $\alpha(A_0)2^{-1}, \ldots, A_0 2^{-\alpha(B_0)}$ 次, 共 $\alpha(A_0)(1 - 2^{-\alpha(B_0)})$ 次且最终 $\beta(A_0) = A_0 2^{-\alpha(B_0)}$.
- 第 5 个循环执行 $\alpha(A_1 - A_0)$ 次, 使得最终 $\beta(A_i) = \alpha_2(A_i) + \alpha(A_1 - A_0) = 2\alpha(A_1)$, $\beta(A_1) = \ldots = \beta(A_{i-1}) = \alpha(A_1) - \alpha(A_1 - A_0) = \alpha(A_0)$, 符合公式 (17).

$\mathbf{update}_{d,1}(B_0)$ 中的外层循环至多执行 $(\beta(B_0)+1)/2 = 1 + \alpha(B_0)$ 次, 内层循环每次将 B 翻倍, 最终 $\beta(B) = 2^{1+\alpha(B_0)}\alpha(B_0) = 2^{\beta(B_0)}$, 因此, β 是对应 $\mathcal{E}val(\lambda)$ 的好格局. Leroux[38]还证明了如下引理.

引理 16[38]. 若 α 是一个对应非平凡编码 λ 的合法格局, 执行 $\mathbf{eval}_{d,i}$ 后得到的 β 也是合法的; 若 β 是好的, 则 α 也是好的, 并且 $i = i_\lambda$, β 对应编码 $\mathcal{E}val(\lambda)$.

最终 $\mathbf{eval}_d$ 的定义与第 3.4.1 节一致, 并且引理 15 对新的 $\mathbf{eval}_d$ 也成立. 借此我们可以构造一个 $2^{F_{d+1}(n)}$-生成器 $\mathcal{G}en_d = (P_d, X_d, (a,b,c), Z_d)$, 其中, X_d 即上述所有计数器, $Z_d = \{a_{d+1}, \overline{b}, \overline{b_0}\} \cup \{b_i\}_{i\in[d]_0}$, P_d 如图 17 所示.

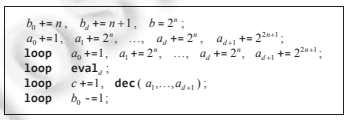


图 17 优化后的 P_d

考虑 P_d 的某个 Z_d-执行, 易知: 前 3 行构造了一个对应 $\lambda_0 = ((n+1)\boldsymbol{e}_d, n)$ 的好格局 α_0, 且 $\alpha_0(A_0) \neq 0$. 执行完若干次 $\mathbf{eval}_d$ 后得到一个合法格局 β, 终止后 a_{d+1} 和 $\{b_i\}_{i\in[d]}$ 为 0 表明 $\beta(A_{d+1}) = \beta(A_1)$, 且 A_1 的值都转移到了 c 上, 因此一定有 $\beta(c) = \beta(A_1) \geqslant \beta(A_0)2^{\beta(B_0)} \geqslant \beta(A_0)\beta(B)$. 至少存在一种执行 $\beta(c) = \beta(A_0)\beta(B)$, 则 β 是个对应 $(0_d, F_{d+1}(n))$ 的好格局, 终止后 $\overline{b} = 0$, $\overline{a_0} = 0$, 从而得到了一个三元组 $\tau_t = (a_0, b, c, \beta(A_0), 2^{F_{d+1}(n)})$; 其他分支满足 $\beta(c) > \beta(A_0)\beta(B)$, 虽然不是三元组, 但不影响引理 10 的正确性, 因为后续程序使用 a_0、b、c 测零时永远有 $c > a_0 b$, 无法得到任何 $\{c\}$-执行.

最后, $|X_d| = 2d+8$, $|Z_d| = d+4$, 根据推论 2 可知, $(2d+8)$-VASS 的可达性问题是 $\mathbf{F}_{d+1}$-难的. 最终观察到 b_d 只会减少, 因此可以用具体数值替代; 由于 x_d 只会增加, 因此可以省略. 最终有如下定理.

定理 11[38]. $d \geqslant 3$ 时, $(2d+4)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

此方法首次突破了因三元组技术带来的 $3d + \Omega(1)$ 的下界, 但某种程度上也具有程序可读性较差的缺点. 同时, 我们目前还无法判断, 这 $d + \Theta(1)$ 的提升究竟是源于指数形式的不变量, 还是与向量编码方法本身有关. 事实上在下一节中, 我们将会看出此次优化完全是新不变量带来的, 因为在 Czerwiński 等人[14,36,37]的递归构造框架下, 我们依然能够基于指数形式的不变量得到一致的结论.

3.5 (2*d*+3)-VASS 可达性问题是 $\mathbf{F}_d$-难的

Czerwiński 等人在文献 [37] 中提出了目前最先进的下界归约技术. 该方法结合了第 3.3 及 3.4 节中介绍的各方法的优点.

1) 选择基于递归的方法构造 $F_d(n)$-放大器, 使得构造的程序与相关证明可读性强.

2) 继承了第 3.3.2 节中 Czerwiński 等人[14,36]对三元组的解释, 将 B 理解为测零次数的上界, 使得递归过程可以直接使用输入三元组进行测零.

3) 放弃了乘法三元组形如 $C = AB$ 的不变量形式, 而是选择继承第 3.4.2 节中 Leroux[38]提出的指数形式的不变量, 通过复用减少了 $d + \Theta(1)$ 个计数器, 突破因三元组形式带来的 $3d$ 的计数器数量下界.

令 $\mathrm{inv}: A,B,C \mapsto (A+C)-A\cdot 4^B$, 本节均考虑这种形式的 inv-三元组 (见定义 3). 该不变量也可以理解为 $C=A(4^B-1)$, 之所以使用 4^B-1 而非 4^B 是为了使得 $B=0$ 当且仅当 $C=0$, 最终判定 $c=0$ 即可, 否则需要判定 $c=a$, 但这是非平凡的, 还需额外的程序实现. 除了这一点改动之外, Czerwiński 等人[37]的方法本质上还是递归构造 $F_d(n)$-放大器, 在有了前文技术的基础上, 本节仅作简单介绍.

首先, 使用新三元组模拟测零的方式与前文中不同. 不妨设待测零计数器为 x、y, 一次成功的测零不应当对 x、y 产生影响. 直观上, b 记录了允许模拟测零的次数, 因此每次模拟测零只需减 1 即可. 前文中我们总是维持 $a+x+y=A$ 不变, 但此时这并不适用, 因为这样测零后必须令 $b,c \mapsto b-1,\ c(4^{b-1}-1)(4^b-1)^{-1}$, 这显然不是容易的. 计算可知, 单独维持 C 不变也不容易, 最佳选择是保持 $A+C=a+x+y+c$ 始终不变. 假设模拟测零前格局 α 满足 $\alpha(a+x+y)4^{\alpha(b)}=\alpha(a+x+y+c)$, 简单计算可知, 模拟测零过程只需要完成变换:

$$(a,b,c,x,y) \mapsto (4a+3x+3y, b-1, c-3a-3x-3y, x, y) \tag{19}$$

c 的作用是和 a、x、y 互补, 它的行为由其他计数器确定, 设计程序时确保 a、x、y 计算正确即可. 这需要额外引入辅助计数器 t, 新的 **zero-test**(x) 模块如图 18 所示, **zero-test**(y) 同理.

```
loop a -=1,  c -=1,  t +=2;
loop y -=1,  x +=1,  c -=1,  t +=1;
loop t -=1,  c -=1,  a +=2;
loop x -=1,  y +=1,  c -=1,  a +=1;
b -=1;
```

图 18 **zero-test**(x)

若格局 α 满足 $\alpha(a+x+y+t)4^{\alpha(b)} \leqslant \alpha(a+x+y+c)$ 则称为合法的; 若该式子取等且 $\alpha(t)=0$, 则称其为好的. 考虑从合法格局出发的执行 $\alpha \xrightarrow{\textbf{zero-test}(x)} \beta$, 易知 β 是合法的; 若 β 是好的, 则 α 也是好的, 且 $\beta(x)=\alpha(x)=0$, $\beta(y)=\alpha(y)$, 说明待测零计数器确实是 0 且没有副作用; 若 α 是好的并且满足 $\alpha(x)=0$, 则存在 $\alpha \xrightarrow{\textbf{zero-test}(x)} \gamma$ 使得 γ 是好的, 且 $\gamma(y)=\alpha(y)$, $\gamma(x)=0$, 说明本次模拟测零正常运行且没有副作用. 最终通过检验 $c=0$ 是否成立, 可以推断出终止格局是否是好的, 进一步确定程序中 B 次模拟测零是否成功.

接着考虑 $F_d(n)$-放大器的构造. 若初始格局为 $\tau_s=(a,b',c',A,B,X_1)$, 计算 $\tau_t=(a,b,c,A',F_d(B))$ 的过程需维持 $a+c'+c=A\cdot 4^B$, 这样可以充分利用 τ_s 进行一些模拟测零等行为, 直至 $c'=0$. 最终根据 $A'\cdot 4^{F_d(B)}=A\cdot 4^B$ 可以确定 $A'=A\cdot 4^{B-F_d(B)}$. 考虑快速增长函数的等价定义: $F_1(n)=2n-1$, $F_{k+1}(n)=F_k^{(n-1)}(1)$, 首先构造 $F_1(n)$-放大器 $\mathcal{A}mpl_1=(P_1,X_1,(a,b',c'),(a,b,c),Z_1)$, 其中, $X_1=\{a,b,c,b',c',t\}$, $Z_1=\{c'\}$, P_1 如图 19 所示.

```
loop a -=1,  t +=1;
loop t -=1,  a +=1,  c' -=3,  c +=3;
b' -=1,  b +=1;
loop
  loop a -=1,  t +=2,  c' -=1;
  loop c -=1,  a +=2,  c' -=1;
  loop a -=1,  c +=2,  c' -=1;
  loop t -=8,  c +=15,  c' -=8,  a +=1;
  b' -=1,  b +=2;
```

P_1

```
loop a -=1,  t +=1;
loop t -=1,  a +=1,  c'_{k+1} -=3,  c'_k +=3;
b'_{k+1} -=1,  b_k +=1;
loop
  loop a -=1,  t +=1;
  loop t -=1,  a +=4,  c'_{k+1} -=3;
  loop c_k -=1,  c'_k +=4,  c'_{k+1} -=3;
  loop b_k -=1,  b'_k +=1;
  P_k;  b'_{k+1} -=1;
```

P_{k+1}

图 19 $F_d(n)$-放大器的构造

P_1 的核心框架是“b' −=1, b +=1; **loop** b' −=1, b +=2;”, 这一部分可以 $\{b'\}$-计算出正确的 $F_1(B)$. 而最终需要得到 $A'=A\cdot 4^{1-B}$, 因此在第 1 处 b' −=1 无须修改 a, 后续循环中每次令 $a \mapsto a/4$ 即可. 为了成功地执行对 a 的变换, 可以借助初始格局给出的三元组 τ_s, 因此 P_1 中除核心框架外的部分本质上是在利用三元组保证程序的任意 $\{c'\}$-执行 (也是一个$\{b'\}$-执行) 中对 a 的变换达到了预期要求.

若已经得到了 $\mathcal{A}mpl_k=(P_k,X_k,(a,b'_k,c'_k),(a,b_k,c_k),Z_k)$, 定义 $X_{k+1}=X_k\cup\{b'_{k+1},c'_{k+1}\}$, $Z_{k+1}=Z_k\cup\{c'_{k+1}\}$, 则我们可以得到 $F_{k+1}(n)$-放大器 $\mathcal{A}mpl_{k+1}=(P_{k+1},X_{k+1},(a,b'_{k+1},c'_{k+1}),(a,b_k,c_k),Z_{k+1})$, 其中, P_{k+1} 如图 19 所示. 构造框架完全与图 11 一致, 在细节上则和 P_1 相似, 即使用 $\tau_s=(a,b'_{k+1},c'_{k+1},A,B,X_{k+1})$ 确保程序中的操作 **loop** b_k −=1, b'_k +=1; 执行成功, 便于利用 P_k 进行下一次放大.

最终, 我们构造出了一族 $F_d(n)$-放大器 $\{\mathcal{A}mpl_d\}_{d\in\mathbb{N}}$. 易知 $|X_d|=2d+4$, $|Z_d|=d$, 由于 b'_d 只会减少, 因此在后续使

用 P_d 时可以使用具体的数值替代, 则有如下定理.

定理 12[37]. 当 $d \geqslant 3$ 时, $(2d+3)$-VASS 可达性问题是 $\mathbf{F}_d$-难的.

此定理虽然相较于定理 11 并无本质上的提升, 但给我们如下启发.

1) 在构造 $F_d(n)$-放大器时, 基于快速增加的向量编码的构造方法与基于递归的构造方法很可能互相等价. 在没有提出全新构造方案的情况下, 后续研究者可以自由选择其一进行深入, 而没必要从二者的角度都进行思考.

2) 不变量的形式是关键的. 为了构造使用更少计数器的 $F_d(n)$-放大器, 我们很可能需要寄希望于寻找一些合适的不变量, 至少不应局限于目前已有的乘法与指数形式.

4 固定维 VASS 的现有下界

本节主要介绍固定维 VASS 的最新研究进展. 与第 2 节相比, 本节的研究对象大都是尚未定论的开放问题, 值得后续研究者深入探索. 同时在 d-VASS 可达性问题下界的研究中, 1–2 个的计数器差异往往不会对结果造成本质上的差异, 但在低维 VASS 的可达性问题中并非如此, 因此人们需要更加注重归约的细节. 另外, 基于乘法三元组和放大器的证明框架需要 3 个向量充当三元组、至少两个向量用于模拟计数器机器的行为, 难以得到 5 维以下的相关结果. 因此到目前为止, 除了第 2 节介绍的关于 1 维 VASS 和 2 维 VASS 的完备性结论外, 3 维–8 维 VASS 的可达性问题仅有一些难 (hardness) 结论[39]. 表 3 总结了最新结论.

表 3 固定维 VASS 可达性问题相关结论

维度	3	4	5	6	7	⩾8
一进制编码	**NL**-难	**NP**-难	**PSPACE**-难			**Tower**-难
二进制编码	**PSPACE**-难			**EXPSPACE**-难		

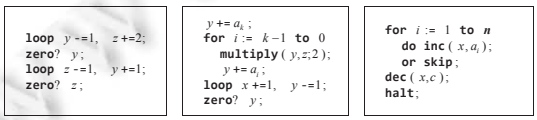
表 3 中大部分结果都是继承自更低维的 VASS, 真正非平凡的结论主要有 4 条, 以下我们分节介绍.

4.1 一进制编码下 4-VASS 的 NP-难归约

我们可以将一进制程序理解为只使用 $x \pm = d$ 形式语句的程序, 其中, d 是一个与输入无关常量. 图 3 给出的归约使用了 $x \mathrel{+}= a$ 形式的程序, 因此不能直接推广到一进制 1-VASS 上. 但如果允许程序使用更多计数器, 我们则可以等价实现该功能. 本节的核心是用 3 个额外的维度来实现二进制到一进制的转换.

给定二进制表示 $a = \sum_{i=0}^{k} a_k 2^k$, 现考虑下面带测零的计数器机器 **inc**$(x, a)$, 该机器使用了辅助计数器 y,z, 将 a 逐位累加到计数器 y 上, 随后再将 y 转移到 x 上, 从而实现了 $x \mathrel{+}= a$. 同理, 若将最后一个循环更改为"**loop** x −=1, y −=1;", 则可以得到 **dec**(x,a).

给定子集和问题实例 $\langle\{a_1,\ldots,a_n\},c\rangle$, 可以高效归约到图 20 中的 3-计数器机器 P, 其正确性同引理 5. 最后, 为了得到 VASS 相关结论, 我们需要去掉程序 P' 中的测零 (来自模块 **multiply**). 令 $k_m = \max_{i\in[n]} |a_i|$ 表示这些数在二进制下的最大长度, 则 **inc** 或 **dec** 至多使用 $2k_m - 1$ 次测零, P' 至多使用 $(2k_m - 1)(n+1)$ 次测零, 整体上与输入规模成多项式关系, 只需利用第 3.2.2.1 节介绍的控制计数器方法即可解决. 更具体地说, P' 中待测零计数器包括 z, y, 我们需要引入控制计数器 t, 以 y 为例, 每次进行操作 $y \pm = d$ 时, 我们修改考察剩余程序中还需对 y 进行多少次测零, 不妨设为 m 次, 则需同时令 $c \pm = dm$. 最终得到计数器程序 P, 其长度是 n 和 k_m 的多项式, 且 P 有 **0**-执行当且仅当 $\langle\{a_1,\ldots,a_n\},c\rangle \in$ subset sum.



multiply $(y, z; k)$ **inc** (x, a) P'

图 20 从子集和问题到 3-计数器机器的归约

值得注意的是, 程序 P 是没有嵌套循环的, 这种程序对应的 VASS 中也不存在嵌套的环 (cycle), 因此被称作平坦的 (flat). 这类 VASS 在上界的证明中扮演着重要的角色, 也有研究者[40,41]专门从事过平坦 VASS 的研究, 因此我们有必要在下面的定理中特地强调 flat 这一性质.

定理 13[39]. 一进制编码下平坦 4-VASS 可达性问题是 **NP**-难的.

4.2 一进制编码下 5-VASS 的 PSPACE-难归约

针对 **PSPACE**-难下界的证明, 前文已经提供了两个思路: 其一是利用子集和博弈问题进行归约; 其二是利用 3 个计数器的 2^n-有界计数器机器可达性问题. 受限于编码方案, 目前暂无人使用子集和博弈问题进行证明; 后者的难点主要在于可用计数器过少上. 为此, 我们至少需要从两个方面进行优化: 首先, 是否可以证明 2 个计数器的 2^n 有界计数器机器可达性问题也是 **PSPACE**-难的? 其次, 能否使用更少的计数器进行模拟测零? 解决这两个问题是证明 5 维以下的一进制 VASS 是 **PSPACE**-难的前提.

4.2.1 2 个计数器的 2^n -有界计数器机器可达性问题

前文中, 我们使用了较为平凡的方法论证 **Tower** 以上的下界可以使用 2 个计数器的 $f(n)$-有界计数器机器进行归约, 即利用 2^x3^y 编码两个计数器 x、y, 但这涉及指数量级的放大, 不能应用于需要多项式归约的情景. Czerwiński 等人[39]针对这种特殊情况给出了一种更精妙的证明方法, 使得如下引理成立.

引理 17[39]. 在一进制编码下, 只使用 2 个计数器的 2^n-有界计数器机器可达性问题是 **PSPACE**-难的.

证明: 给定 LBA 实例 $\langle M,x\rangle$, 其纸带长度为 $n=|x|$, 不妨设 $x=x_1\circ\ldots\circ x_n$. 令 p_i 为第 i 个素数, 则我们可以使用 $a=p_1^{x_1}p_2^{x_2}\ldots p_n^{x_n}$ 编码输入, 后续迁移过程使用 a *= p_i 和 a /= p_i 修改纸带上第 i 位的内容. 易知这可以使用辅助计数器 t 实现. 最终我们得到了一个有界计数器机器 M', 它一共具有两个计数器 a, t, 且 $a+t\leqslant 2p_n^{nl}$, 其中, l 是一个和 M 的字符表有关的常数. 根据素数定理 $p_n=\mathrm{O}(n\ln n)$, 因此 $a+t=2^{\mathrm{poly}(n)}$. 注意, M' 的所有操作中涉及的常数都是 $\mathrm{poly}(n)$ 量级的, 可以采用一进制编码. 由此可知, LBA 问题可以多项式时间归约到一进制编码的只有 2 个计数器的 2^n-有界计数器机器可达性问题上. 证毕.

4.2.2 平方二元组

Czerwiński 等人[39]基于三元组的思想提出了平方二元组 (quadratic pair) 的概念, 所谓平方二元组是指元组 $\tau=(b,c,B,X)$, 其中, $b,c\in X$ 是两个计数器, 满足 $b=B$, $c=B^2$, 且对任意 $x\in X\setminus\{b,c\}$ 都有 $x=0$. 使用过程中需要维持待测零计数器 x, y 满足 $b+x+y=B$, 并将所有 **zero**? x 都替换为下面的 **zero-test**(x), y 同理.

设某次执行该段程序的初始和终止格局分别为 α、β, 则 $\beta(c)\geqslant\alpha(c)-2\alpha(B)+1$, $\beta(B)=\alpha(B)-1$. 若初始格局有 $\alpha(c)\geqslant\alpha(B)^2$, 则 $\beta(c)\geqslant\beta(B)^2$, 且 $\beta(c)=\beta(B)^2$ 可推导出 $\alpha(x)=\beta(x)=0$. 模拟测零的过程中 B 也会减少, 经过 l 次测零后有 $b+x+y=B-l$, 因此 x、y 必须满足 $x+y+l\leqslant B$. 我们不妨只考虑 $x+y\leqslant B/2$ 与 $l\leqslant B/2$ 的情形, 由此可知平方二元组可为 $(B/2)$-有界计数器机器进行 $B/2$ 次模拟测零.

给定使用 2 个计数器的 2^n-有界计数器机器 M, 易知其测零次数不会超过 $2^{n+1}s$, 其中, s 是 M 的状态数. 而 M 同时也是 $2^{n+1}s$ 有界的 (这也是我们只能从有界计数器机器可达性问题进行归约, 而不能从有界可达性问题进行归约的原因), 为了模拟 M 的执行, 我们只需构造平方二元组 $\tau=(b,c,2^{n+2}s,X)$. 注意, $2^{n+2}s$ 和 $2^{2n+4}s^2$ 是输入规模的指数, 无法直接构造, 而应当分别从 $4s$、$16s^2$ 出发进行 n 次乘法, 对应图 21 中的程序 P. 该程序使用计数器 b,c,d, 并重复了 **multiply**($b,d,2$); **multiply**($c,d,2$); n 次, 每次需要进行 4 次测零, 最终共进行 $4n$ 次测零. 引入 t 后使用控制计数器方法后, 最终得到 $\mathrm{poly}(n)$ 长的程序, 可以 $\{t\}$-计算出 τ.

图 21 平方二元组的模拟测零和构造

构造平方二元组的过程中使用了 4 个计数器, 模拟 M 的过程需要两个计数器, 总共 6 个. 但注意, 计数器 d 可以复用, 因此最终只需要 5 个计数器. 如下定理成立.

定理 14[39]. 一进制编码下 5-VASS 可达性问题是 **PSPACE**-难的.

4.3 二进制编码下 6-VASS 的 EXPSPACE-难归约

根据引理 2, 我们考虑给出 2^{2^n}-有界计数器机器可达性问题到 6-VASS 可达性问题的归约. 给定使用 3 个计数器的 2^{2^n} 有界计数器机器 M, 其测零次数不超过 $3s4^{2^n}$, 只需构造出平方二元组 $\tau=(b,c,6s4^{2^n},X)$. 虽然此形式与第 4.2 节形式相似, 但 $B=6s4^{2^n}$ 需要从 $6s$ 出发做 2^n 次乘法, 不能直接采用上面的方法, 否则会构造出一个指数长的程序. 如果转而将 **for** 语句替换成普通的 **loop**, 则需要设法保证该循环恰好执行 2^n 次. 注意, 每次循环会进行 4 次测零, 因此只需保证进行 4×2^n 次测零即可. 这可以通过构造三元组实现.

如图 22 所示, 在 P 中, 我们构造了三元组 $\tau'=(a',b',c',A,8\times 2^n+2,X)$, 并在得到 τ' 后执行了 P'_τ, 即将 P_τ 中所有测零使用三元组 τ' 模拟, 最终可以 $\{c'\}$-计算出需要的二元组 τ. 最终我们希望复用 a', 因此对其进行了一次模拟测零, 这也是 τ' 测零次数为 $4\times 2^n+1$ 而非恰好 4×2^n 的原因. 整个程序中 $X=\{a',b',c',b,c,d\}$ 共使用 6 个计数器, 注意, a'、b'、d 均可以复用. 可得到如下定理.

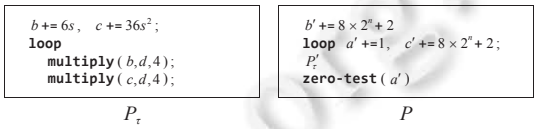


图 22 构造平方二元组 $\tau=(b,c,6s4^{2^n},X)$

定理 15[39]. 二进制编码下 6-VASS 可达性问题是 **EXPSPACE**-难的.

4.4 一进制编码下 8-VASS 的 Tower-难归约

证明 **Tower**-难的核心在于构造一个 F_3-放大器, 在第 3 节介绍的结论中, 给出的最好构造方法需要使用 $2d+3=9$ 个计数器, 注意到该方法是从 F_1-放大器开始递归构造的, 但事实上直接构造 F_2-放大器并不困难, 并且借此可以减少一次递推, 从而省去一个计数器. 图 23 中的程序 P_2 给出了一个 F_2-放大器 $\mathcal{A}mpl_2=(P_2,X_2,(a',b',c'),(a,b,c),Z_2)$, 其中, $X_2=\{a',b',c',a,b,c,d\}$, $Z_2=\{a',c'\}$.

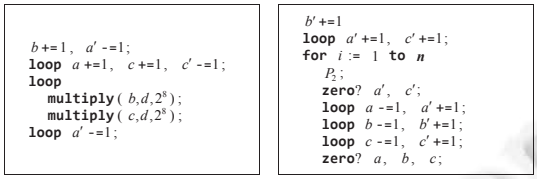


图 23 构造 $\mathcal{A}mpl_3$

若该程序从 $\tau_s=(a',b',c',A_s,B,X_2)$ 出发, 循环中 **multiply** 内的 4 次测零均使用 τ_s 模拟, 则一共可以进行 $B/8$ 次循环, 因此最终可以 Z_2-计算出 $\tau_t=(a,b,c,A_t,2^B,X_2)$. 基于 P_2 可以进一步构造出计数器机器 P'_3, 该程序一进制编码为 $O(n)$ 长, 且只需进行 $5n$ 次测零. 设引入 t 并使用控制计数器方法变换后的程序为 P_3, 我们高效构造出了一个一进制编码、8 计数器的 $F_3(n)$-生成器 $\mathcal{G}en_3=\{X_2\cup\{t\},P_3,(a',b',c'),Z_2\cup\{c\}\}$. 根据推论 2 有如下定理.

定理 16[39]. 一进制编码下 8-VASS 可达性问题是 **Tower**-难的.

本节有一个细节问题, 即为何此处放弃了平方二元组思想, 以及平方二元组能否推广到 F_3 以上的情形. 直观上, 三元组使用 A 限制计数器的值, B 限制计数器的测零次数, 二元组中用 B 同时限制这二者, 因此少使用了一个计数器. 但前者 A、B 不必相同, 从而对待测零计数器的限制较少; 使用二元组则要求该计数器的值和测零次数都不能超过 $B/2$, 这导致了二元组无法使用 $f(n)$-放大器构造: 若给定 $\tau_s=(b,c,B,X)$, 我们希望通过一个程序 P 构造出 $\tau_t=(b',c,f(B),X)$, 则此过程中需要有 $b'\leqslant B$, 否则无法使用 τ_s 模拟测零; 从而有 $f(B)\leqslant B$, 但这样的放大器是平凡的.

4.5 固定维平坦 VASS

所谓平坦 VASS, 是指那些迁移函数无法构成嵌套环的 VASS. 这是一个很强的性质, 因此对平坦 VASS 的研究相对简单, 并且有机会对一般 VASS 的研究带来启发. 目前相关的结论可以总结如表 4 所示.

表 4 平坦 VASS 可达性问题相关结论

维度	$\leqslant 2$	3	$\geqslant 4$
一进制编码	**NL**-完备[21]	**NL**-难	**NP**-完备[39]
二进制编码	**NP**-完备[21]		

由此可见, 平坦 VASS 可达性问题的复杂性仅剩下唯一一块“拼图”: 一进制编码下 3-VASS 可达性问题究竟应当如何刻画? 已知它是 **NL**-难的, 并且在 **NP** 内, 这几乎是所有 VASS 可达性问题中, 已知上界与下界最为接近的一种情形. 在更高维的情形下, 平坦 d-VASS 是一个严格简单于普通 d-VASS 的模型, 我们有如下猜想.

猜想 1. 一进制编码下平坦 3-VASS 可达性问题是 **NP**-完备的.

此外, 一进制编码下固定维 VASS 可覆盖性问题是 **NL**-完备的; 二进制编码下固定维 VASS 可覆盖性问题在 **NP** 中, 暂时不能证明其严格比可达性问题简单[42].

4.6 短证据

在证明 VASS 可达性问题上界时, 常需要说明若任意两个格局可达, 则一定存在一条对应的“短”证据 π. 同理, 给定 d_0, 若存在一族 d_0-VASS 实例 $\{(V_n, p_n, q_n)\}_{n\in\mathbb{N}}$, 使得 $p_n(0)\to_V q_n(0)$ 的任何证据都是 $\Omega(f(n))$, 则一定程度上能够说明 d_0-VASS 比 SPACE$(\log f(n))$ 中的问题都难. Czerwiński 等人[43]证明了如下命题.

猜想 2[43]. 存在一族 $O(n^3)$ 长、具有 $2^{2^{\Omega(n)}}$ 长可达证据的二进制 4-VASS 可达性实例 $\{(V_n, p_n, q_n)\}_{n\in\mathbb{N}}$.

目前最好的相关结论是定理 15: 6-VASS 可达性问题是 **EXPSPACE**-难的. 根据此命题, 我们有希望将其提升至 4 维或 5 维. 同一篇文献中还证明了如下猜想.

猜想 3[43]. 存在一族 $O(n^2)$ 长、具有 $2^{\Omega(n)}$ 长可达证据的一进制平坦 3-VASS 可达性实例 $\{(V_n, p_n, q_n)\}_{n\in\mathbb{N}}$.

这说明我们有希望证明 3 维或 4 维一进制 VASS 可达性问题是 **PSPACE**-难的. 目前最优结果是定理 14 (仍然需要 5 个计数器). 第 4.2 节中曾提到过受限于编码方案, 暂无人考虑过将子集和博弈归约到一进制 VASS 上, 但第 4.1 节中关于 **NP**-难的归约启发我们可以在线地构造出每一个数, 这也是一种未来可以考虑的证明思路.

5 总结与展望

Petri 网凭借其简洁的建模形式和强大的表达能力, 被普遍应用于工程优化和理论分析等诸多领域. 向量加法系统作为其等价模型, 也相应地受到了学术界的广泛关注. 在前文中, 我们主要从理论研究的角度切入, 综述了向量加法系统可达性问题复杂性下界的相关模型和概念、最新进展和核心技术. 特别地, 对 d-VASS 可达性问题梳理了基于三元组和放大器方法的证明框架, 并以此串联起了自 1976 年以来该领域中的重要结论, 揭示了不同研究者提出的方法之间的联系及演变规律. 接下来, 将根据我们对该问题的深入理解, 给出一系列重要开放的问题, 以及相关可能的突破方向, 以期为有志研究者提供支持与启发.

1) d-VASS 可达性问题. d-VASS 可达性问题在快速增长函数类中的层级目前还没有精确的刻画, 上界结论表明该问题复杂性不高于 $\mathbf{F}_d$, 下界结论则大致说明其复杂性至少是 $\mathbf{F}_{d/2}$. 不妨猜想该问题落在 $Cd+O(1)$ 层中, 当下的主要工作仍在确定 C 的范围, 由定理 12 可知, $1/2\leqslant C\leqslant 1$. 然而上界的工作已很难推进, 下界方面最大的短板则在于 F_d-放大器的构造, 在现有测零技术 (三元组) 下, 递归构造放大器时每次至少增加两个新计数器, 导致人们无法得到任何 $C>1/2$ 的结论. 同时, 上文中断言放大器不适用于构造平方二元组, 因此暂时也无法将平方二元组推广到更高维. 除非未来有研究者能够提出类似平方二元组的方法对模拟测零做出改进, 同时还能配合放大器进行递归构造, 或是提出更简洁的编码形式, 使得目前 $O(2d)$ 个计数器所能记录的信息可以使用更少的计数器覆盖, 否则 d-VASS 可达性问题的下界研究仍会处于僵局.

2) 其他开放问题. 除此之外, 我们认为如下开放问题也是重要的.

(1) 是否存在一族二进制 3-VASS 可达性问题实例, 使得它们的证据长度至少是 2-exp(n)?

(2) 是否存在一族二进制 7-VASS 可达性问题实例, 使得它们的证据长度至少是 3-exp(n)?

(3) 一进制 7-VASS 是否是 **EXPSPACE**-难的?

(4) 1 -PVASS 可覆盖性问题是否是 **EXPSPACE**-难的?

关于问题 2(1)、2(2), 我们倾向于相信二进制 3-VASS 可达性问题是 **Tower**-完备的, 但是目前却找不到任何严格长于 $2^{\Omega(n)}$ 的实例. 一进制 7-VASS 目前已知是 **PSPACE**-难的, 但是 8-VASS 即是 **Tower**-难, 二者差距较大, 因此 7-VASS 大概率有更好的下界.

VASS 模型有不少变种, 这些变种上的可达性问题难度更大, 有些问题的可判定性目前都无从知晓. PVASS (pushdown VASS) 是 VASS 的一个常见拓展模型, 它比普通 VASS 增加了一个栈, 目前已知 1-PVASS 可覆盖性问题是 **PSPACE**-难的, 且在 **EXPSPACE** 中, 也是一个上下界较为接近但仍旧没有定论的开放问题. 关于更多限制或拓展模型的相关验证问题, 可以参考文献 [16].

References

[1] Petri CA. Kommunikation mit Automaten [Ph.D. Thesis]. Bonn: Institute für Instrumentelle Mathematik, 1962.

[2] van Glabbeek R, Vaandrager F. Petri net models for algebraic theories of concurrency. In: Proc. of the 1987 Int'l Conf. on Parallel Architectures and Languages Europe (PARLE). Eindhoven: Springer, 1987. 224–242. [doi: 10.1007/3-540-17945-3_13]

[3] Reisig W. Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets. Berlin, Heidelberg: Springer, 2013. [doi: 10.1007/978-3-662-03687-7]

[4] Zurawski R, Zhou MC. Petri nets and industrial applications: A tutorial. IEEE Trans. on Industrial Electronics, 1994, 41(6): 567–583. [doi: 10.1109/41.334574]

[5] Chaouiya C. Petri net modelling of biological networks. Briefings in Bioinformatics, 2007, 8(4): 210–219. [doi: 10.1093/bib/bbm029]

[6] Hamadi R, Benatallah B. A Petri net-based model for Web service composition. In: Proc. of the 14th Australasian Database Conf. Adelaide: Australian Computer Society Inc., 2003. 191–200.

[7] Angeli D, De Leenheer P, Sontag ED. A Petri net approach to the study of persistence in chemical reaction networks. Mathematical Biosciences, 2007, 210(2): 598–618. [doi: 10.1016/j.mbs.2007.07.003]

[8] Lohmann N, Verbeek E, Dijkman R. Petri net transformations for business processes—A survey. In: Trans. on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-aware Information Systems. Berlin, Heidelberg: Springer, 2009. 46–63. [doi:10.1007/978-3-642-00899-3_3]

[9] van der Aalst WMP. Petri-net-based workflow management software. In: Proc. of the 1996 NFS Workshop on Workflow and Process Automation in Information Systems. 1996. 114–118.

[10] Sun J, Qin SY, Song YH. Fault diagnosis of electric power systems based on fuzzy Petri nets. IEEE Trans. on Power Systems, 2004, 19(4): 2053–2059. [doi: 10.1109/TPWRS.2004.836256]

[11] Karp RM, Miller RE. Parallel program schemata. Journal of Computer and System Sciences, 1969, 3(2): 147–195. [doi: 10.1016/S0022-0000(69)80011-5]

[12] Hopcroft J, Pansiot JJ. On the reachability problem for 5-dimensional vector addition systems. Theoretical Computer Science, 1979, 8(2): 135–159. [doi: 10.1016/0304-3975(79)90041-0]

[13] Rackoff C. The covering and boundedness problems for vector addition systems. Theoretical Computer Science, 1978, 6(2): 223–231. [doi: 10.1016/0304-3975(78)90036-1]

[14] Czerwiński W, Orlikowski Ł. Reachability in vector addition systems is Ackermann-complete. In: Proc. of the 62nd IEEE Annual Symp. on Foundations of Computer Science (FOCS 2021). Denver: IEEE, 2022. 1229–1240. [doi: 10.1109/FOCS52979.2021.00120]

[15] Leroux J. The reachability problem for Petri nets is not primitive recursive. In: Proc. of the 62nd IEEE Annual Symp. on Foundations of Computer Science (FOCS 2021). Denver: IEEE, 2022. 1241–1252. [doi: 10.1109/FOCS52979.2021.00121]

[16] Zhang WB, Long H. State-of-the-art survey on verification of vector addition systems. Ruan Jian Xue Bao/Journal of Software, 2018, 29(6): 1566–1581 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5465.htm [doi: 10.13328/j.cnki.jos.005465]

[17] Yang QZ. The study of vector addition system [Ph.D. Thesis]. Shanghai: Shanghai Jiao Tong University, 2022 (in Chinese with English abstract). [doi: 10.27307/d.cnki.gsjtu.2022.000038]

[18] Valiant L G, Paterson MS. Deterministic one-counter automata. Journal of Computer and System Sciences, 1975, 10(3): 340–350. [doi: 10.1016/S0022-0000(75)80005-5]

[19] Travers S. The complexity of membership problems for circuits over sets of integers. Theoretical Computer Science, 2006, 369(1–3): 211–229. [doi: 10.1016/j.tcs.2006.08.017]

[20] Schmitz S. Complexity hierarchies beyond elementary. ACM Trans. on Computation Theory (TOCT), 2016, 8(1): 3. [doi: 10.1145/2858784]

[21] Blondin M, Englert M, Finkel A, Göller S, Haase C, Lazić R, Mckenzie P, Totzke P. The reachability problem for two-dimensional vector addition systems with states. Journal of the ACM (JACM), 2021, 68(5): 34. [doi: 10.1145/3464794]

[22] Lafourcade P, Lugiez D, Treinen R. Intruder deduction for *AC*-like equational theories with homomorphisms. In: Proc. of the 16th Int'l Conf. on Term Rewriting and Applications. Nara: Springer, 2005. 308–322. [doi: 10.1007/978-3-540-32033-3_23]

[23] Haase C, Kreutzer S, Ouaknine J, Worrell J. Reachability in succinct and parametric one-counter automata. In: Proc. of the 20th Int'l Conf. on Concurrency Theory. Bologna: Springer, 2009. 369–383. [doi: 10.1007/978-3-642-04081-8_25]

[24] Fearnley J, Jurdziński M. Reachability in two-clock timed automata is PSPACE-complete. Information and Computation, 2015, 243: 26–36. [doi: 10.1016/j.ic.2014.12.004]

[25] Mayr EW. An algorithm for the general Petri net reachability problem. In: Proc. of the 13th Annual ACM Symp. on Theory of Computing. Milwaukee: ACM, 1981. 238–246. [doi: 10.1145/800076.802477]

[26] Kosaraju SR. Decidability of reachability in vector addition systems (Preliminary Version). In: Proc. of the 14th Annual ACM Symp. on Theory of Computing. San Francisco: ACM, 1982. 267–281. [doi: 10.1145/800070.802201]

[27] Lambert JL. A structure to decide reachability in Petri nets. Theoretical Computer Science, 1992, 99(1): 79–104. [doi: 10.1016/0304-3975(92)90173-D]

[28] Sacerdote GS, Tenney RL. The decidability of the reachability problem for vector addition systems (Preliminary Version). In: Proc. of the 9th Annual ACM Symp. on Theory of Computing. Boulder: ACM, 1977. 61–76. [doi: 10.1145/800105.803396]

[29] Leroux J, Schmitz S. Demystifying reachability in vector addition systems. In: Proc. of the 30th Annual ACM/IEEE Symp. on Logic in Computer Science. Kyoto: IEEE, 2015. 56–67. [doi: 10.1109/LICS.2015.16]

[30] Leroux J, Schmitz S. Ideal decompositions for vector addition systems. In: Proc. of the 33rd Symp. on Theoretical Aspects of Computer Science (STACS 2016). Orléans: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2016. 1. [doi: 10.4230/LIPIcs.STACS.2016.1]

[31] Schmitz S. Algorithmic Complexity of Well-quasi-orders. Paris: École Normale Supérieure, 2017.

[32] Leroux J, Schmitz S. Reachability in vector addition systems is primitive-recursive in fixed dimension. In: Proc. of the 34th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS 2019). Vancouver: IEEE, 2019. 1–13. [doi: 10.1109/LICS.2019.8785796]

[33] Fu YX, Yang QZ, Zheng YL. Improved algorithm for reachability in *d*-VASS. In: Proc. of the 51st Int'l Colloquium on Automata, Languages, and Programming (ICALP 2024). Tallinn: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2024. 136. [doi: 10.4230/LIPIcs.ICALP.2024.136]

[34] Lipton R. The reachability problem requires exponential space. Research Report, #63, New Haven: Department of Computer Science, Yale University, 1976.

[35] Czerwiński W, Lasota S, Lazić R, Leroux J, Mazowiecki F. The reachability problem for Petri nets is not elementary. Journal of the ACM (JACM), 2020, 68(1): 7. [doi: 10.1145/3422822]

[36] Czerwiński W, Lasota S, OrlikowskiŁ. Improved lower bounds for reachability in vector addition systems. In: Proc. of the 48th Int'l Colloquium on Automata, Languages, and Programming (ICALP 2021). Dagstuhl: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2021. 128. [doi: 10.4230/LIPIcs.ICALP.2021.128]

[37] Czerwiński W, Jecker I, Lasota S, Leroux J, Orlikowski Ł. New lower bounds for reachability in vector addition systems. In: Proc. of the 43rd IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023). Telangana: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2023. 35. [doi: 10.4230/LIPIcs.FSTTCS.2023.35]

[38] Leroux J. The reachability problem for Petri nets is not primitive recursive. CoRR, abs/2104.12695, 2021.

[39] Czerwiński W, Orlikowski L. Lower bounds for the reachability problem in fixed dimensional vasses. In: Proc. of the 37th Annual ACM/IEEE Symp. on Logic in Computer Science. Haifa: ACM, 2022. 40. [doi: 10.1145/3531130.3533357]

[40] Leroux J, Sutre G. On flatness for 2-dimensional vector addition systems with states. In: Proc. of the 15th Int'l Conf. on Concurrency Theory. London: Springer, 2004. 402–416. [doi: 10.1007/978-3-540-28644-8_26]

[41] Leroux J. Flat Petri nets. In: Proc. of the 2021 Int'l Conf. on Applications and Theory of Petri Nets and Concurrency. Cham: Springer, 2021. 17–30. [doi: 10.1007/978-3-030-76983-3_2]

[42] Rosier LE, Yen HC. A multiparameter analysis of the boundedness problem for vector addition systems. Journal of Computer and System

Sciences, 1986, 32(1): 105–135. [doi: 10.1016/0022-0000(86)90006-1]

[43] Czerwiński W, Lasota S, Lazić R, Leroux J, Mazowiecki F. Reachability in fixed dimension vector addition systems with states. In: Proc. of the 31st Int'l Conf. on Concurrency Theory (CONCUR 2020). Dagstuhl Publishing, 2020. 48. [doi: 10.4230/LIPIcs.CONCUR.2020.48]

附中文参考文献

[16] 张文博, 龙环. 向量加法系统验证问题研究综述. 软件学报, 2018, 29(6): 1566–1581. http://www.jos.org.cn/1000-9825/5465.htm [doi: 10.13328/j.cnki.jos.005465]

[17] 杨启哲. 向量加法系统模型研究 [博士学位论文]. 上海: 上海交通大学, 2022. [doi: 10.27307/d.cnki.gsjtu.2022.000038]

附录 A

表 A1 本文数学符号对照表

符号	含义	符号	含义
$\mathbb{N}$	自然数集	$a,b,c,x,y,z \in X$	计数器
$\mathbb{Z}$	整数集	$\overline{x}$, $\overline{y}$, $\overline{z}$	与 x,y,z 互补的计数器
ω	最小的极限序数	i, j, k	常用于表示自然数
$\mathbb{N}_\omega$	自然数集的扩充 $\mathbb{N}\cup\{\omega\}$	$\{x_i\}_i$, $\{y_i\}_i$, $\{z_i\}_i$	表示一族作用相似的计数器
d, D	维度	n	输入规模
$\boldsymbol{u},\boldsymbol{v} \in \mathbb{N}^d,\mathbb{Z}^d$	d维自然数/整数向量	$f,g,h:\mathbb{N}\to\mathbb{N}$	自然数上的函数
$\|\cdot\|$	向量的无穷范数	$t(n)$, $s(n)$	时间函数、空间函数
$\|\cdot\|$	某个集合的基数	**NL**, **NP**,…	各种经典复杂性类
$\mathbf{0}_d$, $\mathbf{0}$	全零向量	$\leqslant_{\mathrm{p}}$, $\leqslant_{\mathrm{L}}$	多项式时间归约, 对数空间归约
V	某个VAS或VASS	ι	序数
$A \subseteq \mathbb{Z}^d$	VAS迁移规则集合	$\{F_\iota\}_\iota$	快速增长函数谱系
$\boldsymbol{a} \in A$	某条VAS迁移规则	$\{\mathcal{F}_\iota\}_\iota$	快速增长函数类谱系
Q	状态集	$\{\mathbf{F}_\iota\}_\iota$	快速增长复杂性类谱系
$p,q,r \in Q$	状态	O, Ω, Θ	渐进复杂性记号
$T \subseteq Q\times\mathbb{Z}^d\times Q$	VASS迁移规则集合	$\mathrm{poly}(n)$	多项式函数
$t \in T$	某条VASS迁移规则	inv	不变量
$\alpha,\beta,\ldots \in Q\times\mathbb{N}^d$	VASS格局	τ	乘法三元组或 inv-三元组
$\pi \in T^*$	可达性证据	$\mathcal{A}mpl$	放大器
$\alpha \to_V \beta$	在 V 中 α 到 β 可达	$\mathcal{G}en$	生成器
$\|V\|_1$, $\|V\|_2$	V的一进制和二进制编码长度	Z	控制计数器集合
M	各类自动机	B	常用于表示上界
I_1, I_2, I_3,…	机器指令	λ	快速增长函数的向量编码
P	计数器程序	$\mathcal{V}al$	解码函数
X	计数器集合	$\mathcal{E}val(\lambda)$	编码 λ 的求值变换

作者简介

陈蔚骏, 博士生, 主要研究领域为程序语言, 进程演算, 并发模型.

傅育熙, 博士, 特聘教授, CCF 杰出会员, 主要研究领域为理论计算机科学.

龙环, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为理论计算机科学.