软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(10):3009−3020 [doi: 10.13328/j.cnki.jos.005321]

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

良结构下推系统的可覆盖性问题的下界*

李春淼, 蔡小娟, 李国强

(上海交通大学 软件学院,上海 200240)

通讯作者: 李国强, E-mail: li.g@sjtu.edu.cn

摘 要: 良结构下推系统是下推系统和良结构迁移系统的结合,该系统允许状态和栈字符是向量的形式,因而它们是无限的.状态迁移的同时允许栈进行入栈出栈的操作.它“非常接近不可判定的边缘”.利用重置 0 操作,提出了一种模型可覆盖性问题复杂度下界的一般性证明方法,并且证明了状态是三维向量的子集和一般性的良结构下推系统的可覆盖性问题分别是 Tower 难和 Hyper-Ackermann 难的.

关键词: 良结构下推系统;可覆盖性;下界;重置 0;Hyper-Ackermann 难

中图法分类号: TP301

中文引用格式: 李春淼,蔡小娟,李国强.良结构下推系统的可覆盖性问题的下界.软件学报,2018,29(10):3009−3020. http://www.jos.org.cn/1000-9825/5321.htm

英文引用格式: Li CM, Cai XJ, Li GQ. Lower bound for coverability problem of well-structured pushdown systems. Ruan Jian Xue Bao/Journal of Software, 2018,29(10):3009−3020 (in Chinese). http://www.jos.org.cn/1000-9825/5321.htm

Lower Bound for Coverability Problem of Well-Structured Pushdown Systems

LI Chun-Miao, CAI Xiao-Juan, LI Guo-Qiang

(School of Software, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: Well-Structured pushdown systems (WSPDSs) combine pushdown systems and well-structured transition systems to allow locations and stack alphabets to be vectors, and thus they are unbounded. States change with the push and pop operations on the stack. The model has been said to be “very close to the border of undecidability”. This paper proposes a general framework to get the lower bounds for coverability complexity of a model by adopting the reset-zero method. The paper proves that the complexity is Tower-hard when a WSPDS is restricted with three dimensional state vectors, and Hyper-Ackermann hard for the general WSPDSs.

Key words: well-structured pushdown system; coverability; lower bound; reset-zero; Hyper-Ackermann hard

1 引 言

程序分析是软件工程和形式化方法重要的研究领域之一,人们在最初的程序分析中使用有限系统[1]作为程序模型,它是由有限的状态集和在状态上有限的迁移规则组成.其表达能力十分有限,无法对递归、并发等程序特征进行分析.对其扩展后可分别形成下推系统[2]和良结构迁移系统[3].前者给有限系统配备一个栈,状态迁移的同时进行入栈、出栈操作,方便建模单线程递归程序;后者将状态集拓展到无限的良拟序集,并保证迁移规则具有单调性.为了更好地描述和分析并发程序,Cai 和 Ogawa 将二者结合,提出良结构下推系统[4],可以很方便地建模并发递归程序,其具有强大的表达能力[5],已经“很接近不可判定的边缘”[6].

我们通过一个自然数程序例子来说明这个模型:只用一个局部变量计算 Ackermann 函数 $A(2,n)$的程序如图

* 基金项目: 国家自然科学基金(61472238, 61672340, 61872232)
Foundation item: National Natural Science Foundation of China (61472238, 61672340, 61472240, 61872232)
收稿时间: 2017-02-18; 修改时间: 2017-04-04, 2017-05-09; 采用时间: 2017-06-16; jos 在线出版时间: 2018-03-13
CNKI 网络优先出版: 2018-03-13 17:17:54, http://kns.cnki.net/kcms/detail/11.2560.TP.20180313.1717.001.html

1 所示.并发程序对应多栈,因此图灵等价,是不可判定的.很多研究都对并发程序进行了限制,比如在文献[7]中,要求程序在函数调用时不能进行进程切换,直到建模其行为的多重集下推系统栈空才行,该模型就可被编码为良结构下推系统;在文献[8]中,针对用于建模程序的模型,允许多栈但限制栈高,也允许栈元素是多重集,该模型也可以被编码成良结构下推系统.因此尽管并发后的程序相比单进程的程序,相对应的良结构下推系统的可覆盖性问题的难度可能会上升,但依旧满足我们提到的下界难度结论.由于我们后面证明的下界结论主要是针对单进程程序的执行,所以这里不考虑并发性对结果的影响.Ackermann 函数 $A(m,n)$是著名的非原始递归函数,其定义如下所示:

$$A(m,n)=\begin{cases} n+1, & \text{if } m=0 \\ A(m-1,1), & \text{if } n=0 \\ A(m-1mA(m,n-1)), & \text{otherwise} \end{cases}.$$

Ackermann 函数是严格单调的.

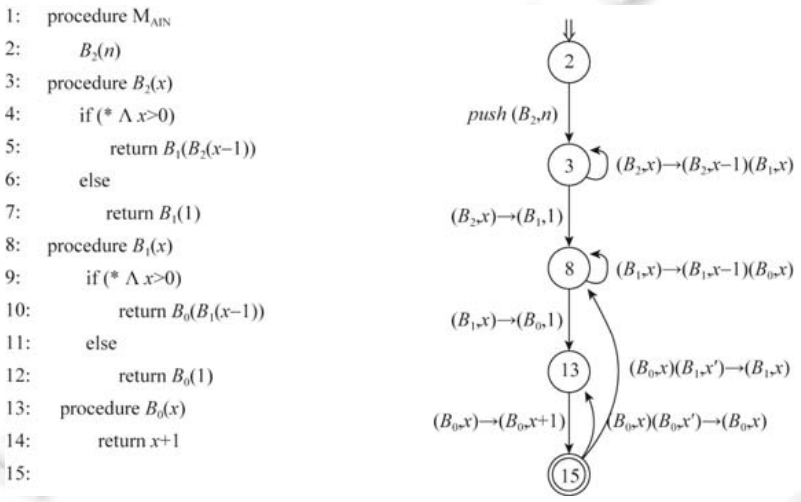

Fig.1 Computing Ackermann function $A(2,n)$ with only one local variable

图 1 只用 1 个局部变量计算 Ackermann 函数 $A(2,n)$

图 1 左侧为计算 $A(2,n)$的程序,*是布尔表达式,不确定地计算出 true 或 false,与 WSPDS 中不允许 0 测试(测试一个值是否为 0)相一致.这段程序只要求第 1 个参数为固定参数,主要是为了表明程序的局部变量可以被编码到相对应的良结构下推系统的栈字符集中,为第 4 节栈字符为向量集的良结构下推系统可用于建模的相关程序提供一个直觉上的例子.该程序不能精确地计算出 $A(2,n)$,但可以保证计算出的最大值是 $A(2,n)$.右侧为与程序对应的 WSPDS,状态对应于左侧程序的行号,栈字符是函数名 $x\in\{B_0,B_1,B_2\}$和局部变量自然数 x 组成的序对.状态间的迁移描述了栈上的操作,例如在状态 3 时的迁移$(B_2,x)\to(B_2,x-1)(B_1,x)$表示 $x\geqslant 1$ 时,将原栈顶(B_2,x)替换成(B_1,x)后压入$(B_2,x-1)$,这种规则称为入栈规则.状态 15 到状态 8 的迁移依次弹出两个元素$(B_0,x)(B_1,x')$后压入新栈顶(B_1,x),这种称为非标准的出栈规则.

递归的自然数程序可以被建模成良结构下推系统,可将局部的自然数变量编码到栈向量,可将全局的自然数变量编码到状态向量.用良结构下推系统的栈字符向量编码并发的递归程序的活动的进程数[9].因此,递归程序的某一点是否可达可以转换成良结构下推系统的可覆盖性问题.

本文展示了基于重置 0 的证明模型的可覆盖性问题难度下界的通用方法,所谓重置 0 是指直接对变量赋值为 0.我们证明了一旦一个模型可以正向弱计算和反向弱计算一个函数 f,则其可覆盖性问题的难度的下界就是 f 难的.另外,还证明了具有栈字符集为 n 维向量集的良结构下推系统的可覆盖性问题的下界是 Hyper-Ackermann 难的.并且利用本文提出的方法重新证明了具有状态集为三维自然数向量集的良结构下推系统的

可覆盖性问题的下界是 TOWER 难的.

对于可覆盖性和可达性的复杂性证明有如下一些工作:Lipton 给出最有名的向量加法系统的下界[10].证明了其可覆盖性问题和可达性问题是 EXPSPACE 的,主要思想是用并行程序设置固定上界的计数器设置“精确地”0 测试.Lazic[11]使用与 Lipton 相同的方法,证明了状态为向量集的下推系统的可覆盖性问题的下界是 TOWER 难的,Schnoebelen 等人的工作[12,13]使用重置 0 的技巧,证明了重置 Petri 网的可覆盖性问题是 Ackermann 难的,也是本文借鉴使用的方法.在 Schnoebelen 等人的证明中,重置 Petri 网的维度必须与要得到的其可覆盖性问题的难度 *Ackermann*(*m*,*n*)中的参数 *m* 相关,因此整个规约过程相对输入大小不是多项式的.但在良结构下推系统中,因为有栈的存在,保证了其可覆盖性问题下界的证明是多项式的,且独立于参数的大小.Demri 等人[14]证明了分支向量加法系统的可覆盖性问题是 2EXPTIME 完备的.Lazic[15]证明了分支向量加法系统的可达性问题是 2EXPSPACE 难的.Bonnet[16]证明了允许一个计数器用于 0 测试的向量加法系统的可达性问题是可判定的.不对称的带状态的向量加法系统的可达性问题是不可判定的[17−19],但其自底向上的可覆盖性问题是 Ackermann 完备的[20].

本文第 2 节介绍后文将用到的一些基础符号和良结构下推系统、Hyper-Ackermann 函数和 2-计数机器的定义.第 3 节介绍证明可覆盖性问题的难度的一般性方法.第 4 节给出证明,得出栈字符集为 *n* 维向量集的良结构下推系统的可覆盖性问题的下界是 Hyper-Ackermann 的.第 5 节采用本文提出的方法重新证明状态为向量集的下推系统的可覆盖性问题的下界是 TOWER 难的,与 Lazic[11]的结果一致.第 6 节总结全文,并展望未来工作.

2 预备知识

令$\mathbb{N}$为自然数集,$\mathbb{Z}$为整数集,$\mathbb{N}^k$(或$\mathbb{Z}^k$)表示 k 维的自然数(或整数)向量集.$\mathbf{n},\mathbf{m}$ 表示 k 维的自然数向量,$\mathbf{a},\mathbf{a}'$表示 k 维的整数向量,$\mathbf{n}_k$ 表示向量 $\mathbf{n}$ 的第 k 维分量.令 p,q 等表示状态,α,β等表示栈字符,w,v 表示栈上的字,其长度分别为$|w|$和$|v|$.用 c,d 表示格局.$|\Gamma|<\infty$表示Γ是有限集.

在迁移规则中使用$\rightarrow$表示迁移关系.用$\mapsto$表示格局之间的单步迁移,$\mapsto^*$表示格局之间的多步迁移.

2.1 良结构下推系统

定义 2.1. 一个良结构下推系统是一个三元组,其中,

- P 是有限的状态集,或者给定一个具体的维度 k,$P=\mathbb{N}^k$;
- Γ是有限的栈字符集,或者给定一个具体的维度 d,$\Gamma=\mathbb{N}^d$;
- Δ是有限的单调偏函数的集合:$P\times\Gamma^{\leqslant 2}\rightarrow P\times\Gamma^{\leqslant 2}$.

单调性是指对 $p,q\in P,w,v\in\Gamma^{\leqslant 2}$,若$(p,w\rightarrow q,v)\in\Delta$,则对任意 $p'\geqslant p,w'\gg w$,有$(p',w'\rightarrow q',v')\in\Delta$且 $q'\geqslant q,v'\gg v$.这里,$\gg$是$\geqslant$在Γ^*上的按位扩展,即:$a_1a_2...a_m\gg b_1b_2\ldots b_n$,当且仅当对任意的 $i,a_i\geqslant b_i$ 且 $m=n$.如果移除了单调性,良结构下推系统就会变得图灵完备.

下推向量加法系统是栈字符集有限且状态集为 k 维自然数向量集的良结构下推系统,即 $P=\mathbb{N}^k$,$|\Gamma|<\infty$.

良结构下推系统的格局$\langle p,w\rangle$是由状态 p 和栈上的字 w 组成的序对.格局间的单步迁移$\mapsto$是格局的重写,$\mapsto^*$是$\mapsto$的自反传递闭包.

$$\frac{(p,w\rightarrow q,v)\in\Delta}{\langle p,ww'\rangle\mapsto\langle q,vw'\rangle}.$$

给定初始格局 $c_0=\langle p,w\rangle$和目标格局 $c_f=\langle q,v\rangle$,良结构下推系统的可达性问题是判断 $c_0\mapsto^* c_f$ 是否成立,可覆盖性问题是对任意 $c_g=\langle q',v'\rangle$且 $q'\geqslant q,v'\gg v$,判断 $c_0\mapsto^* c_g$ 是否成立.

例 2.1:一个良结构下推系统 $M=\langle(\mathbb{N},\leqslant),(\mathbb{N},\leqslant),\Delta\rangle$,迁移规则$\Delta$如下:

$$\Delta=\begin{cases}r_1:p,\alpha\rightarrow p+6,(\alpha-6)(\alpha-6), & \text{if } \alpha\geqslant 6\\ r_2:p,\varepsilon\rightarrow p-6,6, & \text{if } p\geqslant 6\end{cases}.$$

给定初始格局 $c_0=\langle 6,0\rangle$,则存在一个如下的格局迁移序列:

$$\langle 6,0\rangle \mapsto \langle 0,60\rangle \mapsto \langle 6,000\rangle \mapsto \langle 0,6000\rangle \mapsto \langle 6,00000\rangle \mapsto ...$$

可见,格局⟨6,000⟩是从初始格局可达的,而格局⟨0,5000⟩虽然不可达,但可被覆盖.

2.2 Hyper-Ackermann函数

康托尔在 1883 年提出了序数的概念[21],引入了无限的序列.ω是最小的无限序数.按照康托尔正规式的表示方法,每一个 limit 序数λ都可以被唯一地分解成 $\omega^{d_1}.c_1+\omega^{d_2}.c_2+...+\omega^{d_k}.c_k$,其中,$k$ 是自然数,$c_1,c_2,\ldots,c_k$ 是正整数,$d_1>d_2>\ldots>d_k\geqslant 0$ 是序数.每一个 limit 序数λ都有一个唯一的基础序列$(\lambda_n)_{n\in\omega}$.对小于ω^ω的 limit 序数,其基础序列为

$$\left(\omega^{d_1}.c_1+\omega^{d_2}.c_2+...+\omega^{d_k}.c_k\right)_n=\omega^{d_1}.c_1+...+\omega^{d_{k-1}}.c_{k-1}+\omega^{d_k}.(c_k-1)+\omega^{d_{k+1}}.(n+1).$$

定义 2.2. Fast-Growing 函数族按下标序数为$\alpha\leqslant\omega^\omega$索引定义如下[6]:

$$F_0(n)=n+1,\qquad F_{\alpha+1}(n)=F_\alpha^{n+1}(n)=\overbrace{F_\alpha(F_\alpha(...F_\alpha(n)...))}^{n+1},$$

$$F_\lambda(n)=F_{\lambda_n}(n),\lambda<\omega^\omega,\qquad F_{\omega^\omega}(n)=F_{\omega^{n+1}}(n).$$

$F_1(n)$是 n 的线性函数,$F_2(n)$是 n 的指数函数,$F_{\omega^\omega}(n)$ 是 Hyper-Ackermann 函数.

例 2.2:

$$\begin{aligned}F_{\omega^\omega}(3)&=F_{\omega^4}(3)\\&=F_{\omega^3.4}(3)\\&=F_{(\omega^3.3+\omega^2.4)}(3)=F_{(\omega^3.3+\omega^2.3+\omega.4)}(3)=F_{(\omega^3.3+\omega^2.3+\omega.3+4)}(3)\\&=F_{\left(\omega^3.3+\omega^2.3+\omega.3+3\right)}\left(F_{\left(\omega^3.3+\omega^2.3+\omega.3+3\right)}\left(F_{\left(\omega^3.3+\omega^2.3+\omega.3+3\right)}(3)\right)\right)\\&=...\end{aligned}$$

引理 2.2. Hyper-Ackermann 函数是严格单调的,即对任何 $n'>n$, $F_{\omega^\omega}(n')>F_{\omega^\omega}(n)$.

2.3 2-计数机器

定义 2.3. 一个计数机器是一个三元组 $M=\langle P,C,\Delta\rangle$,其中,$P$ 是有限的状态集,C 是有限的计数器的集合,$\Delta\subseteq P\times op(C)\times P$ 是有限的迁移规则.指令集 $op(C)$是按如下方式定义的规则集(C_i 是 M 的第 i 个计数器):

$$op::=C_i=0?\mid C_i++\mid C_i>0?C_i--$$

计数机器的格局是一个序对$\langle p,\mathbf{n}\rangle$,其中,$p\in P$ 表示当前的状态,$\mathbf{n}\in\mathbb{N}^{|C|}$表示每个计数器当前的内容.格局之间的迁移 $\langle p,\mathbf{n}\rangle\mapsto\langle q,\mathbf{m}\rangle$ 成立当且仅当如下情况之一成立.

- 若$(p,C_i=0?,q)\in\Delta$且 $\mathbf{n}_i=0$,则 $\mathbf{m}=\mathbf{n}$;
- 若$(p,C_i++,q)\in\Delta$,则 $\mathbf{m}_i=\mathbf{n}_i+1$ 且对 $j\neq i$ 有 $\mathbf{m}_j=\mathbf{n}_j$;
- 若$(p,C_i>0?C_i--,q)\in\Delta$且 $\mathbf{n}_i>0$,则 $\mathbf{m}_i=\mathbf{n}_i-1$ 且对 $j\neq i$ 有 $\mathbf{m}_j=\mathbf{n}_j$.

给定初始格局$\langle p_0,\mathbf{0}\rangle$和目标格局$\langle p_f,\mathbf{0}\rangle$,计数机器的可达性问题是判断格局的执行序列 $\langle p_0,\mathbf{0}\rangle\mapsto^*\langle p_f,\mathbf{0}\rangle$ 是否成立.

2-计数机器(只有 2 个计数器)是图灵完备的[22],其可达性问题是不可判定的.然而,如果给每个计数器设定一个上界,即每个计数器的值不超过这个上界值,则 2-计数机器的计算能力就得到了限制,其可达性问题就是可判定的.例如,输入的大小为 n,上界值是 $\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}$,可达性问题是 TOWER 难的;上界值为 Hyper-Ackermann 函数 $F_{\omega^\omega}(n)$, 可达性问题是 Hyper-Ackermann 难的[23].

这里给定 2-计数机器设定一个上界 B,使得在格局迁移的过程中所有的计数器的值的总和不超过上界,记为 $\langle p,\mathbf{n}\rangle\mapsto^B\langle q,\mathbf{m}\rangle$. $\mapsto^{B^*}$ 是 $\mapsto^B$ 的自反传递闭包.

3 可覆盖性问题的下界

我们的证明技巧基于 Schnoebelen 等人在文献[12,13]中提出的将 Ackermann-bound 的 2-计数机器的可达

性问题规约到重置 Petri 网的可覆盖性问题.本节不针对任何具体的模型,只是给出具体的规约的思路:将以 $f(x)$ 为界的 2-计数机器的可达性问题规约到某个特定模型 M 的可覆盖性问题,注意,这里的 $f(x)$应该是严格单调的函数.

简单来说,规约由 3 部分构成.

- 输入为 $\mathbf{n}$,M_H 计算 $f(\mathbf{n})$的值;
- M^b 模拟以 $f(x)$为界的 2-计数机器;
- M_H^{-1} 逆向计算 $f(\mathbf{n})$.

M_H 根据输入向量 $\mathbf{n}$ 计算一个上界值 B,最大为 $f(\mathbf{n})$,将这个值传递给 M^b,触发 M^b 的执行.M^b 执行后,B 的值可能会变小(错误的模拟)或保持不变(正确的模拟).然后将值传递给第 3 部分 M_H^{-1} 进行逆向计算,即根据 B 的值求对应的向量 $\mathbf{n}'$.函数 $f(\mathbf{n})$严格的单调性,保证如果 $B<f(\mathbf{n})$,$\mathbf{n}'\succ\mathbf{n}$,即 $\mathbf{n}'$不能覆盖 $\mathbf{n}$.只有正确的模拟才能保证最后的可覆盖性成立.当可覆盖性成立时,M^b 精确地模拟了以 $f(x)$为界的 2-计数机器的执行,所以 M 的可覆盖性问题的下界为 f-难的.相关思想如图 2 所示(M_{CM}^b 是以 $f(x)$为界的 2-计数机器,下文将具体介绍其构造).

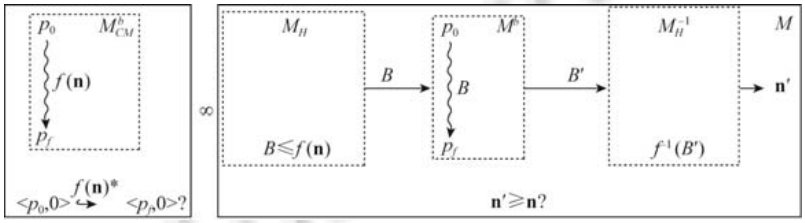

Fig.2 Reduction from the reachability problem of 2-counter machine with $f(x)$ bounds to the coverability problem of M

图 2 从以 $f(x)$为界的 2-计数机器的可达性问题到 M 的可覆盖性问题的规约

3.1 有界的 2-计数机器的模拟

给定 2-计数机器 $M=\langle P,C,\Delta\rangle$,有界的 2-计数机器 $M_{CM}^b=\langle P',C\cup\{b\},\Delta'\rangle$, 初始化 $P'=P$,$\Delta'=\varnothing$,根据Δ中的每一条规则$(p,op(C),q)$按如下方式构造 M_{CM}^b.

- 若 $op(C)=C_i{+}{+}$,则对新状态 $s\notin P'$,$P'=P'\cup\{s\}$,
$$\Delta'=\Delta'\cup\{(p,b>0?b--,s),(s,C_i++,q)\};$$
- 若 $op(C)=C_i=0?$,则$\Delta'=\Delta'\cup\{p,C_i=0?q\}$;
- 若 $op(C)=C_i>0?C_i--$,则对新状态 $s\notin P'$,$P'=P'\cup\{s\}$,
$$\Delta'=\Delta'\cup\{(p,C_i>0?C_i--,s),(s,b++,q)\}.$$

M_{CM}^b 的构造具体如图 3 所示.

M_{CM}^b 给 M 添加了一个上界计数器,记为 b,保证在对应的 2-计数机器的格局迁移的过程中所有计数器的和不发生改变(在添加的新状态上可能总和会暂时地变大或变小,但在原始对应的 2-计数机器中状态上的计数器的和是固定的).根据 i_1+i_2+b 的大小,M_{CM}^b 的可达性问题具有相应的难度.

但是由于很多模型都没有测 0 能力,因此本文使用重置 0 操作代替测 0,即将图 3 中的 $i_1=0?$用 $i_1=0$ 来代替,构造相应的模型 M^b.M^b 的构造如图 4 所示.

注意,在 M^b 中,i_1+i_2+b 的和可能会变小,虽然一旦对一个非 0 的计数器置 0,但却并未对有界计数器增加相应的数量.在 M^b 的格局迁移过程中,如果 $i_1=0$ 从未执行或者仅仅执行在对一个值已经为 0 的变量上,则所有计数器的和保持不变,M^b 精确地模仿了 M_{CM}^b 的格局迁移;如果对一个非 0 的计数器执行了重置 0 操作,所有计数器的和就会变小,这种情况下,M^b 与 M_{CM}^b 会产生不同的格局序列,即 M^b 并未精确地模仿 M_{CM}^b. 反之,如果在 M^b 的格局迁

移中,所有计数器的和始终保持不变(注意,这里的始终也是对应于原始的 2-计数机器的状态上),则每一次可能存在的重置 0 操作都精确地模仿了有界的 2-计数机器上的测 0 操作.

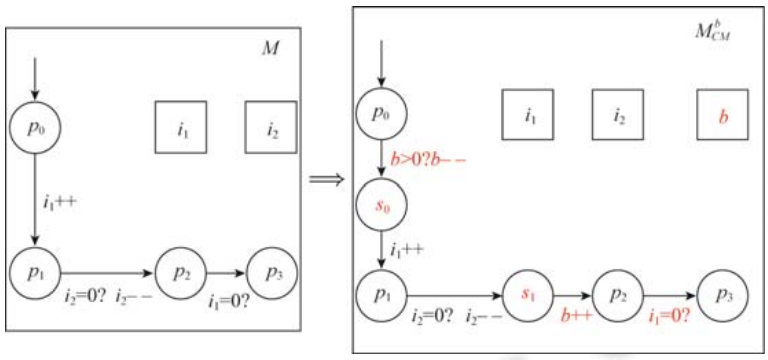

Fig.3 Construct bounded 2-counter machine from 2-counter machine

图 3 从 2-计数机器构造相应的有界的 2-计数机器

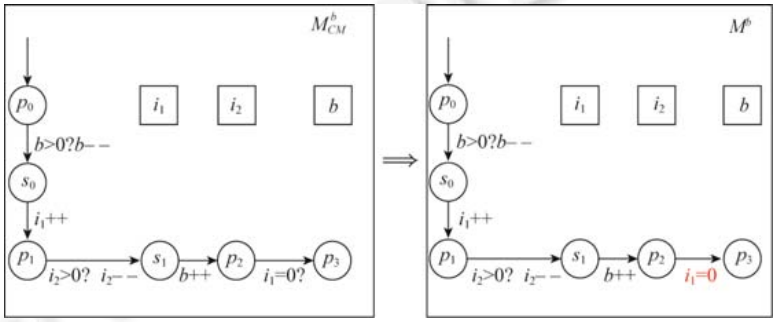

Fig.4 M^b simulate bounded 2-counter machine

图 4 M^b 模拟有界的 2-计数机器

引理 3.1. M^b_{CM} 是如图 3 构造的有界的 2-计数机器,M^b 是如图 4 构造的模型(以下的状态 p,q 均对应于图 3 中原始的 2-计数机器 M 中的状态).

- 如果 M^b_{CM} 中存在 $\langle p,(i_1,i_2,b)\rangle \mapsto^* \langle q,(i_1',i_2',b')\rangle$ 的格局迁移序列,则 M^b 中也存在相同的格局迁移序列,且 $i_1+i_2+b=i_1'+i_2'+b'$.
- 如果 M^b 中存在 $\langle p,(i_1,i_2,b)\rangle \mapsto^* \langle q,(i_1',i_2',b')\rangle$ 的格局迁移序列,则 $i_1+i_2+b \geqslant i_1'+i_2'+b'$. 但是,如果 $i_1+i_2+b=i_1'+i_2'+b'$, M^b_{CM} 中必然存在 $\langle p,(i_1,i_2,b)\rangle \mapsto^* \langle q,(i_1',i_2',b')\rangle$ 的格局迁移序列.

3.2 函数 *f*(**n**)的正向计算和逆向计算

函数 $f(\mathbf{n})$的定义域为 k 维的自然数向量 $\mathbf{n}$,且严格单调,即对任意 $i\in[1...k]$,若 $n_i' > n_i$, 则有

$$f(n_1,...,n_i',...,n_k) > f(n_1,...,n_i,...,n_k).$$

因为我们要研究的模型不是图灵完备的,$f(\mathbf{n})$的正向和逆向计算可能不精确,但这并不影响我们的结论,因为 M^b 的格局迁移中不会增大 $f(\mathbf{n})$的值.因此,如果 M_H 计算出的上界值 B 不大于 $f(\mathbf{n})$,则经过 M^b 后,B 依旧不大于 $f(\mathbf{n})$, M_H^{-1} 逆向计算后得到的 $\mathbf{n}'\succ\mathbf{n}$.

定义 3.2. 给定输入 $\mathbf{n}$,如果 M_H 计算出的最大的值为 $f(\mathbf{n})$,则称 M_H 弱计算函数 $f(\mathbf{n})$.

给定输入 B',设 M_H^{-1} 逆向计算出来的值的集合为 S,满足:

- 若 $f(\mathbf{n})=B'$,则 $\mathbf{n}\in S$;

- 若 $\mathbf{n}\in S$,则 $f(\mathbf{n})\leqslant B'$,

则称 M_H^{-1} 逆向弱计算函数 $f(\mathbf{n})$.

如图 2 所示,模型 M 的格局的迁移由 M_H、M^b、M_H^{-1} 组成.M_H 完成计算后,触发 M^b 中的 p_0,并把计算得到的 B 传递给 M^b.M^b 中格局迁移到状态为 p_f 的格局时触发 M_H^{-1} 执行.3 部分共享相同的计数器.M_H 的最终状态连接 M^b 中的 p_0,p_f 连接到 M_H^{-1} 的初始状态.

定理 3. f 是严格单调的 k 元函数.M_H 弱计算函数 $f(\mathbf{n})$ M_H^{-1} 逆向弱计算函数 $f(\mathbf{n})$. M_{CM}^b 是如图 3 构造的有界的 2-计数机器,M^b 是图 4 构造的模拟有界的 2-计数机器的模型.M_H 的输入是 k 维向量 $\mathbf{n}$.

- 若在 M_{CM}^b 中存在 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b')\rangle$ 的格局迁移序列,则 M_H^{-1} 逆向计算得到的最大值是 $\mathbf{n}$;
- 若 M_H^{-1} 逆向计算得到的 $\mathbf{n}'\geqslant\mathbf{n}$,则在 M_{CM}^b 中必然存在 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b)\rangle$ 的格局迁移序列.

证明:

- 在 M_{CM}^b 中存在 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b')\rangle$ 的格局迁移序列,由引理 3.1 可知,M^b 中也存在相同的格局迁移序列 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b')\rangle$, 且 $b=b'\leqslant f(\mathbf{n})$. 又因为 M_H^{-1} 逆向弱计算 $f(\mathbf{n})$,所以 M_H^{-1} 输出的最大值是 $\mathbf{n}$;
- 若 M_H^{-1} 弱逆向计算得到的 $\mathbf{n}'\geqslant\mathbf{n}$,则传递给 M_H^{-1} 的 $b'\geqslant f(\mathbf{n})$. 又由图 2 中 3 部分的连接情况可得,$M^b$ 中存在格局迁移序列 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(i_1',i_2',b')\rangle$, 又有 M_H 弱计算函数 $f(\mathbf{n})$,则 $f(\mathbf{n})\geqslant b\geqslant b'+i_1'+i_2'\geqslant f(\mathbf{n})$, 所以,$b=b'=f(\mathbf{n})$,即 M^b 中存在格局迁移序列 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b)\rangle$ (M^b 的格局迁移中所有计数器 s 的和保持不变,表明 M^b 中的每次重置 0 操作都精确地模拟了测 0),由引理 3.1 可得, M_{CM}^b 中必然存在 $\langle p_0,(0,0,b)\rangle\mapsto^*\langle p_f,(0,0,b)\rangle$ 的格局迁移序列.证毕. □

4 良结构下推系统的可覆盖性问题下界

本节考虑良结构下推系统的子模型:状态为有限集而栈字符为向量,即 $M=\langle|P|<\infty,\Gamma=\mathbb{N}^{n+3},\Delta\rangle$ (其中,n 为要计算的函数给定的参数值).该模型可正向和逆向弱计算 Hyper-Ackermann 函数的值.由定理 3 可得,其可覆盖性问题是 Hyper-Ackermann 难的.

4.1 (正向)弱计算 Hyper-Ackermann 函数

给定一个良结构下推系统 $M_r=\langle\{\bullet\},\mathbb{N}^{n+3},\Delta\rangle$,其中,状态集为单字符集,栈字符集为 n+3 维自然数向量的集合(因为只有一个状态,所以迁移规则中省去了状态,仅描述栈上的迁移),迁移规则为

$$\Delta=\begin{cases}r_1:(k_n,k_{n-1},...,k_0,t,x)\to(k_n,k_{n-1},...,k_0-1,x,x)\\ r_2:(k_n,k_{n-1},...,k_0,t,x)\to(k_n,k_{n-1},...,k_0,0,x)(k_n,k_{n-1},...,k_0,t-1,x)\\ r_3:\left(\underbrace{k_n,k_{n-1},...,k_i,...,0}_{n+1},0,x\right)\to\left(\underbrace{k_n,k_{n-1},...,k_i-1,x+1,...,0}_{n+1},0,x\right),i\in[1,n].\\ r_4:\left(\underbrace{0,0,...,0}_{n+1},0,x\right)(k_n',k_{n-1}',...,k_0',t',x')\to(k',k_{n-1}',...,k',t',x+1)\end{cases}$$

按照迁移规则中所示,栈向量的前 n+1 个分量都标记了下标,例如 k_0 就表示该向量的从左向右第 n+1 个分量.以下描述都直接使用这一表示方法.规则 r_1 对栈顶元素进行在某些分量上值的修改;r_2 将栈顶元素的第 n+1 个分量进行修改并压入新向量($k_n,k_{n-1},\ldots,0,x$);r_3 表示当 i 在[1,n]区间内(含两端)时,都可执行 r_3 的迁移;r_4 表示当栈顶的前 n+2 个分量都是 0 时,可执行出栈操作,并将最后一个分量值传递给栈顶的下一个元素,形成新的栈顶.注意,每条迁移规则都代表单调函数,例如对 r_4,任何向量只要其所有分量值均不小于 0,都有此迁移,即

$$(p_n,p_{n-1},...,p_0,te,xp)(k_n',k_{n-1}',...,k_0',t',x')\to(k_n',k_{n-1}',...,k_0',t',xp+1).$$

这里写 r_4 只是方便理解.

引理 4.1. M_r 可弱计算 Hyper-Ackermann 函数,且

$$hyper(n)=\max\left\{x+1 \mid \left\langle \bullet, \left(\underbrace{n+1,0,...,0}_{n+1},0,n\right)\perp\right\rangle \mapsto^{*} \left\langle \bullet, \left(\underbrace{0,0,...,0}_{n+1},0,x\right)\perp\right\rangle\right\},$$

其中,⊥为栈底元素.

证明:首先,因为如定义 2.2 所示,给定参数 n 后,Hyper-Ackermann 函数没有升幂操作(除了一开始的 $F_{\omega^{\omega}}(n)=F_{\omega^{n+1}}(n)$, $F_{\omega^{\omega}}(n)=F_{\omega^{n+1}}(n)$, 可直接应用 $F_{\lambda}(n)=F_{\lambda_n}(n),\lambda<\omega^{\omega}$ 降幂),$\omega^n.k_n+\omega^{n-1}.k_{n-1}+\ldots+\omega^0.k_0$ 的形式固定,即最高幂次是 n,栈向量的前 n+1 个分量$(k_n,k_{n-1},\ldots,k_0)$对应于$\omega^n.k_n+\omega^{n-1}.k_{n-1}+\ldots+\omega^0.k_0$ 的 n+1 个系数.简单来说,迁移规则 r_1 和 r_2 描述 $F_{\alpha+1}(n)=F_{\alpha}^{n+1}(n)=\overbrace{F_{\alpha}\left(F_{\alpha}\left(...F_{\alpha}(n)...\right)\right)}^{n+1}$, 求出内层函数值(栈顶向量的前 n+1 个分量都是 0,对应 $F_0(n)$=n+1)之后应用 r_4 出栈,并将内层结果值保存在最后一个分量上传递给下一个向量,构成新栈顶进行计算.迁移规则 r_3 对应 Hyper-Ackermann 函数计算规则 $F_{\lambda}(n)=F_{\lambda_n}(n),\lambda<\omega^{\omega}$.

其次,栈顶元素可分为以下 3 类.

(1) 栈顶为$(k_n,k_{n-1},\ldots,k_0,t,x)$形式,对应接下来需要计算的 Hyper-Ackermann 函数为 $F^{t}_{\omega^n.k_n+\omega^{n-1}.k_{n-1}...+\omega^0.k_0}(x)$, 首先计算内层函数,直接应用 r_2 入栈新元素$(k_n,k_{n-1},\ldots,k_0,0,x)$表明接下来要计算 $F_{\omega^n.k_n+\omega^{n-1}.k_{n-1}...+\omega^0.k_0}(x)$, 并将$(k_n,k_{n-1},\ldots,k_0,t-1,x)$(对应 $F^{t-1}_{\omega^n.k_n+\omega^{n-1}.k_{n-1}+...+\omega^0.k_0}(x)$) 变为栈顶的下一个元素.根据归纳假设,栈顶会变为 $\left(\underbrace{0,0,...,0}_{n+1},0,x\right)$, 对应 $F_0(x)$,即 $F_{\omega^n.k_n+\omega^{n-1}.k_{n-1}+...+\omega^0.k_0}(x)$ 的结果为 x+1.再应用 r_4 将结果作为下一个函数的参数进行后续计算.

(2) 栈顶为 $\left(\underbrace{0,0,...,0}_{n+1},0,x\right)$ 形式,已在(1)中说明.

(3) 栈顶为 $\left(\underbrace{k_n,k_{n-1},...,k_i,...,0}_{n+1},0,x\right)$ 形式,接下来需要计算的 Hyper-Ackermann 函数为 $F_{\omega^n.k_n+\omega^{n-1}.k_{n-1}+...+\omega^i.k_i}(x)$ 且 i≠0.应用 r_3 生成新栈顶 $\left(\underbrace{k_n,k_{n-1},...,k_i-1,x+1,...,0}_{n+1},0,x\right)$, 即 $F_{\omega^n.k_n+\omega^{n-1}.k_{n-1}+...+\omega^i.(k_i-1)+\omega^{i-1}.(x+1)}(x)$, 与 Hyper 函数计算规则 $F_{\lambda}(n)=F_{\lambda_n}(n),\lambda<\omega^{\omega}$ 保持一致.

但是,因为 M_r 的迁移规则具有单调性,相比正确的格局迁移,可能存在错误的格局迁移,如应使用 r_3 却使用了 r_4,表明 $F_{\omega^n.k_n+\omega^{n-1}.k_{n-1}+...+\omega^i.k_i}(x)$ 且 i≠0 的值直接被看作 x 进行后续计算,造成数据丢失.如果在格局迁移过程中没有任何数据丢失,则当格局处于 $\left\langle \bullet, \left(\underbrace{0,0,...,0}_{n+1},0,x\right)\perp\right\rangle$ 时 x+1 就是正确的结果.证毕. □

例 4.1:利用引理 4.1 计算 $hyper(1)=F_{\omega^{\omega}}(1)=F_{\omega^2}(1)$, 以验证其正确性.相关的格局迁移序列如下(这里,仅列出栈上的迁移,因只有一个状态):

$$\begin{aligned}
&(2,0,0,1)\perp \overset{r_3}{\mapsto}(1,2,0,1)\perp \overset{r_1}{\mapsto}(1,1,1,1)\perp \overset{r_2}{\mapsto}(1,1,0,1)(1,1,0,1)\perp \overset{r_1}{\mapsto}(1,0,1,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(1,0,0,1)(1,0,0,1)(1,1,0,1)\perp \overset{r_3}{\mapsto}(0,2,0,1)(1,0,0,1)(1,1,0,1)\perp \overset{r_1}{\mapsto}(0,1,1,1)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(0,1,0,1)(0,1,0,1)(1,0,0,1)(1,1,0,1)\perp \overset{r_1}{\mapsto}(0,0,1,1)(0,1,0,1)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(0,0,0,1)(0,0,0,1)(0,1,0,1)(1,0,0,1)(1,1,0,1)\perp \overset{r_4}{\mapsto}(0,0,0,2)(0,1,0,1)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_4}{\mapsto}(0,1,0,3)(1,0,0,1)(1,1,0,1)\perp \overset{r_1}{\mapsto}(0,0,3,3)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(0,0,0,3)(0,0,2,3)(1,0,0,1)(1,1,0,1)\perp \overset{r_4}{\mapsto}(0,0,2,4)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(0,0,0,4)(0,0,1,4)(1,0,0,1)(1,1,0,1)\perp \overset{r_4}{\mapsto}(0,0,1,5)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_2}{\mapsto}(0,0,0,5)(0,0,0,5)(1,0,0,1)(1,1,0,1)\perp \overset{r_4}{\mapsto}(0,0,0,6)(1,0,0,1)(1,1,0,1)\perp\\
&\quad\overset{r_4}{\mapsto}(1,0,0,7)(1,1,0,1)\perp \overset{r_3}{\mapsto}(0,8,0,7)(1,1,0,1)\perp \overset{r_1}{\mapsto}(0,7,7,7)(1,1,0,1)\perp \overset{r_2}{\mapsto}\ldots
\end{aligned}$$

最后的(0,7,7,7)(1,1,0,1)表示 $F_{\omega+1}(F_7^8(7))$, 之后继续利用迁移规则计算,直至计算至栈中为$(0,0,0,x)\perp$时,x+1

为 *hyper*(1)的值.

4.2 逆向弱计算 Hyper-Ackermann 函数

如果仅仅把 M_r 的迁移规则逆过来,就构造了逆向弱计算 Hyper-Ackermann 值对应参数的良结构下推系统 $M_{r^{-1}}$. 给定一个 Hyper-Ackermann 的值,初始格局是 $\left\langle \bullet,\left(\underbrace{0,0,...,0}_{n+1},0,x-1\right)\perp\right\rangle$ (这里的 n 可任意,但是一旦给定一个 n,之后的栈上的元素都是固定维度的),只要可到达 $\left\langle \bullet,\left(\underbrace{n+1,0,...,0}_{n+1},0,n\right)\perp\right\rangle$,最后的 n 就是对应 $hyper(n)=x$ 的参数 n.虽然 $M_{r^{-1}}$ 的规则是单调的,但每次错误的计算都会造成最后得到比正确的参数 n 还要小的参数值.

引理 4.2. $M_{r^{-1}}$ 可逆向弱计算 Hyper-Ackermann 函数.

定理 4. 给定特定的良结构下推系统,其状态集有限,栈字符集为 n+3 维的良拟序集,则其可覆盖性问题的下界是 Hyper-Ackerman 难的.

证明:因为重置 0 操作是单调的,n+3 维的良结构下推系统可以作为 M^b.M^b,M_r 和 $M_{r^{-1}}$ 共同构成良结构下推系统.由定理 3 可得,当其可覆盖性问题得到解决时, M^b_{CM} 中必然存在 $\langle p_0,(0,0,hyper(n))\rangle \mapsto^* \langle p_f,(0,0,hyper(n))\rangle$ 的格局迁移序列.因后者是 Hyper-Ackermann 难的,所以该良结构下推系统的可覆盖性问题的下界是 Hyper-Ackermann 难的.证毕. □

5 下推向量加法系统的可覆盖性问题的下界的重新证明

本节考虑良结构下推系统的子模型:状态为向量而栈字符为有限集,即 $M=\langle P=\mathbb{N}^3,|\Gamma|<\infty,\Delta\rangle$, 即下推向量加法系统.该模型可正向和逆向弱计算 $\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}$ 的值.由定理 3 可得,其可覆盖性问题是 TOWER 难的.

5.1 (正向)弱计算 $\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}$ 函数

给定一个良结构下推系统 $M_p=\langle \mathbb{N}^3,\left(\{a,\perp\},\ll\right),\Delta\rangle$, 其中,状态集为 3 维自然数向量的集合,栈字符集为单字符集,迁移规则 $\Delta=\begin{cases} r_1:\left((n,0,x),\varepsilon\right)\to\left((n-1,x,x),\varepsilon\right)\\ r_2:\left((n,t,x),\varepsilon\right)\to\left((n,t-1,x+1),\varepsilon\right)\\ r_3:\left((0,0,x),a\right)\to\left((x-1,0,2),\varepsilon\right)\end{cases}$.

引理 5.1. M_p 可弱计算 $\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}$ 函数,且

$$\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n}=\begin{cases}\max\left\{x\middle|\left\langle(1,0,2),\underbrace{aa...a}_{n-2}\perp\right\rangle\mapsto^*\left\langle(0,0,x),\perp\right\rangle\right\}, & n\geqslant 2\\ 2, & n=1\end{cases},$$

其中,⊥为栈底元素.

证明:对格局$\langle(n,t,x),\omega\perp\rangle$有:$t$=0 时表示需要对 x 进行 n 次翻倍操作,即求 $x\times 2^n$,需使用规则 r_1;$t\neq 0$ 时表示需要把 t 的值加到 x 上,需连续使用 r_2.以下运用数学归纳法证明.

- 当 n=1 时显然成立;
- 当 n=2 时,$tower(2)=2^2=4$.对 M_p 有如下格局迁移序列: $\langle(1,0,2),\perp\rangle\overset{r_1}{\mapsto}\langle(0,2,2),\perp\rangle\overset{r_2}{\mapsto}\overset{r_2}{\mapsto}\langle(0,0,4),\perp\rangle$.
- 假设 $n=k$ 时题设成立,则有 $\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{k}=\max\left\{x\middle|\left\langle(1,0,2),\underbrace{aa...a}_{k-2}\perp\right\rangle\mapsto^*\left\langle(0,0,x),\perp\right\rangle\right\}$ 成立,则当 $n=k+1$ 时,起始格局为 $\left\langle(1,0,2),\underbrace{aa...a}_{k-1}\perp\right\rangle$, 由 3 条迁移规则可得,在格局迁移的过程中,会迁移到格局为 $\langle(0,0,x),a\perp\rangle$ 且 $x=\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{k}$. 那么,之后的格局迁移为

$$\langle(0,0,x),a\perp\rangle\overset{r_3}{\mapsto}\langle(x-1,0,2),\perp\rangle\overset{r_1}{\mapsto}\langle(x-2,2,2),\perp\rangle\overset{r_2\ r_2}{\mapsto\mapsto}\langle(x-2,0,4),\perp\rangle\overset{r_1\ r_2\ r_2\ r_2\ r_2}{\mapsto\mapsto\mapsto\mapsto\mapsto}\langle(x-3,0,8),\perp\rangle$$
$$\overset{r_1}{\mapsto}\underbrace{\overset{r_2}{\mapsto}\ldots\overset{r_2}{\mapsto}}_{8}\langle(x-4,0,16),\perp\rangle\overset{r_1}{\mapsto}\underbrace{\overset{r_2}{\mapsto}\ldots\overset{r_2}{\mapsto}}_{16}\ldots\overset{r_1}{\mapsto}\underbrace{\overset{r_2}{\mapsto}\ldots\overset{r_2}{\mapsto}}_{2^{(x-1)}}\langle(0,0,2\times2^{(x-1)}),\perp\rangle=\langle(0,0,2^x),\perp\rangle,$$

对应的 *tower*(*k*+1)的值是 $2^x=2^{\overbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}^{k}}=\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{k+1}$, 即 *n*=*k*+1 时成立.

但是,因为 M_p 的迁移规则具有单调性,相比正确的格局迁移,可能存在如下情况的错误的格局迁移,造成数据丢失.

- 当 *n*>0,*t*>0 时,用了 r_1 而不是 r_2,即 *x*+*t* 被认为是 *x* 进行后续计算.
- 当 *n*>0,*t*>0 时,用了 r_3 而不是 r_2,即(*x*+*t*)×2^n 被认为是 *x* 进行后续计算.
- 当 *n*=0,*t*>0 时,用了 r_3 而不是 r_2,即 *x*+*t* 被认为是 *x* 进行后续计算.
- 当 *n*>0,*t*=0 时,用了 r_3 而不是 r_1,即 *x*×2^n 被认为是 *x* 进行后续计算.

如果在格局迁移过程中没有任何的数据丢失,则当格局处于⟨(0,0,*x*),⊥⟩时 *x* 就是正确的结果.证毕. □

例 5.1:利用引理 5.1 计算 2^{2^2}, 以验证其正确性.相关的格局迁移序列如下:

$$\langle(1,0,2),a\perp\rangle\overset{r_1}{\mapsto}\langle(0,2,2),a\perp\rangle\overset{r_2\ r_2}{\mapsto\mapsto}\langle(0,0,4),a\perp\rangle\overset{r_3}{\mapsto}\langle(3,0,2),\perp\rangle$$
$$\overset{r_1}{\mapsto}\langle(2,2,2),\perp\rangle\overset{r_2\ r_2}{\mapsto\mapsto}\langle(2,0,4),\perp\rangle\overset{r_1}{\mapsto}\langle(1,4,4),\perp\rangle$$
$$\overset{r_2\ r_2\ r_2\ r_2}{\mapsto\mapsto\mapsto\mapsto}\langle(1,0,8),\perp\rangle\overset{r_1}{\mapsto}\langle(0,8,8),\perp\rangle\underbrace{\overset{r_2\ r_2}{\mapsto\mapsto}\ldots\overset{r_2}{\mapsto}}_{8}\langle(0,0,16),\perp\rangle.$$

因此, $2^{2^2}=16$.

5.2 逆向弱计算 $\underbrace{2^{2^{\cdot^{\cdot^2}}}}_{n}$ 函数

如果仅仅把 M_p 的迁移规则逆过来,就构造了逆向弱计算 $\underbrace{2^{2^{\cdot^{\cdot^2}}}}_{n}$ 值对应参数的良结构下推系统 $M_{p^{-1}}$. 给定一个 $\underbrace{2^{2^{\cdot^{\cdot^2}}}}_{n}$ 值 *x*,初始格局是⟨(0,0,*x*),⊥⟩,只要可到达 $\left\langle(1,0,2),\underbrace{aa\ldots a}_{n}\perp\right\rangle$, 最后得到的最大的 *n*(即栈中除去栈底元素外元素的个数)就是对应 *x* 的参数 *n*(每次错误的计算都会造成最后得到比正确的参数 *n* 还要小的参数值).

引理 5.2. $M_{p^{-1}}$ 可逆向弱计算 $\underbrace{2^{2^{\cdot^{\cdot^2}}}}_{n}$ 函数.

定理 5. 给定特定的良结构下推系统,其状态集为三维的良拟序集,栈字符集有限,即三维的下推向量加法系统,其可覆盖性问题的下界是 TOWER 难的.

证明:类似定理 4 的证明.因三维下推向量加法系统可以模拟 M^b.所以,由定理 3 可得,其可覆盖性问题的下界是 TOWER 难的.证毕. □

6 总结以及未来的工作

本文基于良结构下推系统,采用重置 0 操作,提出了一种证明模型的下界的通用方法,证明了 *n* 维的良结构下推系统的可覆盖性问题的下界是Hyper-Ackermann难的,还将其与程序中的全局变量或局部变量的个数相对应,表明了有 *n* 个局部变量的程序的可达性分析问题是很难的,为分析含无限定义域的变量(如整数域)的程序时采取重置 0 操作持否定态度,鼓励将无限定义域限制在有限定义域处理.本文还重新证明了下推向量加法系统在三维情况下其可覆盖性问题就可达到 TOWER 难.

未来希望能够推广这种证明模型可覆盖性难度下界的方法,探索更多向量加法系统的拓展模型的下界或比已知下界更高的下界.同时,寻求证明良结构下推系统的可覆盖性问题难度的上界,形成证明上界的一般性方法并进行对其他模型的应用推广.再进一步探究良结构下推系统的可覆盖性问题的判定性.

References:

[1] Derick WD. Theory of Computation: A Primer. Boston: Addison-Wesley Longman Publishing Co., 1987.

[2] Autebert JM, Berstel J, Boasson L. Context-Free languages and pushdown automata. Handbook of Formal Languages, 1997,6(3): 111–174. [doi: 10.1007/978-3-642-59136-5_3]

[3] Abdulla PA, Čerāns K, Jonsson B, Tsay YK. Algorithmic analysis of programs with well quasi-ordered domains. Information and Computation, 2000,160(1-2):109–127. [doi: 10.1006/inco.1999.2843]

[4] Cai X, Ogawa M. Well-Structured pushdown systems. In: D'Argenio PR, Melgratti H, eds. Proc. of the Int'l Conf. on Concurrency Theory. Berlin: Springer-Verlag, 2013. 121–136. [doi: 10.1007/978-3-642-40184-8_10]

[5] Jin Y, Cai XJ, Li GQ. Expressiveness of well-structured pushdown systems. Journal of Shanghai Jiaotong University, 2015,49(8):1084–1089 (in Chinese with English abstract). [doi: 10.16183/j.cnki.jsjtu.2015.08.002]

[6] Leroux J, Praveen M, Sutre G. Hyper-Ackermannian bounds for pushdown vector addition systems. In: Henzinger T, Miller D, eds. Proc. of the Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS). New York: ACM, 2014. Article 63. [doi: 10.1145/2603088.2603146]

[7] Sen K, Viswanathan M. Model checking multithreaded programs with asynchronous atomic methods. In: Ball T, Jones RB, eds. Proc. of the Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2006. 300–314. [doi: 10.1007/11817963_29]

[8] Kochems J, Luke Ong CH. Safety verification of asynchronous pushdown systems with shaped stacks. In: D'Argenio PR, Melgratti H, eds. Proc. of the Int'l Conf. on Concurrency Theory. Berlin: Springer-Verlag, 2013. 288–302. [doi: 10.1007/978-3-642-40184-8_21]

[9] Bouajjani A, Emmi M. Analysis of recursively parallel programs. In: Field J, ed. Proc. of the 39th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. New York: ACM, 2012. 203–214. [doi: 10.1145/2103621.2103681]

[10] Lipton RJ. The reachability problem requires exponential space [Ph.D. Thesis]. Department of Computer Science, Yale University, 1976.

[11] Lazic R. The reachability problem for vector addition systems with a stack is not elementary. Computer Science, 2013.

[12] Schnoebelen P. Revisiting Ackermann-Hardness for lossy counter machines and reset Petri nets. Lecture Notes in Computer Science, 2010,6281:616–628. [doi: 10.1007/978-3-642-15155-2_54]

[13] Haddad S, Schmitz S, Schnoebelen P. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In: Logic in Computer Science. IEEE, 2012. 355–364. [doi: 10.1109/LICS.2012.46]

[14] Demri S, Jurdziński M, Lachish O, Lazić R. The covering and boundedness problems for branching vector addition systems. Journal of Computer and System Sciences, 2013,79(1):23–38. [doi: 10.1016/j.jcss.2012.04.002]

[15] Lazić R. The reachability problem for branching vector addition systems requires doubly-exponential space. Information Processing Letters, 2010,110(17):740–745. [doi: 10.1016/j.ipl.2010.06.008]

[16] Bonnet R. The reachability problem for vector addition system with one zero-test. In: Murlak F, Sankowski P, eds. Mathematical Foundations of Computer Science. Berlin: Springer-Verlag, 2011. 145–157. [doi: 10.1007/978-3-642-22993-0_16]

[17] Lincoln P, Mitchell J, Scedrov A, Shankar N. Decision problems for propositional linear logic. Annals of Pure and Applied Logic, 1992,56(1-3):239–311. [doi: 10.1016/0168-0072(92)90075-B]

[18] Kanovich MI. Petri nets, Horn programs, linear logic, and vector games. Annals of Pure and Applied Logic, 1995,75(1-2):107–135. [doi: 10.1016/0168-0072(94)00060-G]

[19] Raskin JF, Samuelides M, Begin LV. Games for counting abstractions. Electronic Notes in Theoretical Computer Science, 2005,128(6):69–85. [doi: 10.1016/j.entcs.2005.04.005]

[20] Urquhart A. The complexity of decision procedures in relevance logic II. The Journal of Symbolic Logic, 1999,64(4):1774–1802. [doi: 10.2307/2586811]

[21] Cantor G. Ueber unendliche, lineare Punktmannichfaltigkeiten. Mathematische Annalen, 1883,21(4):545–591. [doi: 10.1007/BF01446819]

[22] Minsky ML. Computation: Finite and Infinite Machines. Prentice-Hall, Inc., 1967.

[23] Schmitz S, Schnoebelen P. The power of well-structured systems. In: D'Argenio PR, Melgratti H, eds. Proc. of the Int'l Conf. on Concurrency Theory. Berlin: Springer-Verlag, 2013. 5–24. [doi: 10.1007/978-3-642-40184-8_2]

李春淼(1993－),女,陕西商洛人,硕士,主要研究领域为形式化方法,程序语言理论.

蔡小娟(1982－),女,博士,助理研究员,主要研究领域为形式化方法,程序语言理论.

李国强(1979－),男,博士,副教授,CCF 专业会员,主要研究领域为形式化方法,理论计算机科学,程序语言理论.