arXiv:2512.17892v1 [cs.DS] 19 Dec 2025

前缀树降低大规模连续时间随机模型的内存消耗

Landon Taylor[1][0000-0002-4071-3625], JoshuaJeppson[1][0000-0002-4177-7604], Ahmed Irfan[2][0000-0001-7791-9021], Lukas Buecherl[1][0000-0002-4844-6605], Chris Myers[3],和 Zhen Zhang[1][0000-0002-8269-9489]

[1] 犹他州立大学，洛根，犹他州，美国[2] SRI International，门洛帕克，加利福尼亚州，美国[3] 科罗拉多大学博尔德分校，博尔德，科罗拉多州，美国

摘要。 具有巨大状态空间的高并发系统模型（例如用于建模生物和化学系统的化学反应网络（CRN）），给前沿形式化分析工具带来了严峻挑战。尽管已有许多符号化方法，但对建模为连续时间马尔可夫链（CTMC）的化学反应网络进行瞬态概率分析，仍需要显式状态表示。为此，当前的前沿方法采用哈希映射，其平均时间复杂度为常数，内存复杂度为线性。然而，面对状态空间极其庞大的模型，哈希映射往往会受到严重的内存限制。为解决这一问题，我们提出使用前缀树存储大规模高并发模型（尤其是化学反应网络）的状态，以节省内存。我们给出了理论分析和基准测试，表明对于极大的状态空间，前缀树优于哈希映射。此外，我们提出采用有界模型检测（BMC）预处理步骤来设定变量排序，以进一步改善内存使用量；初步评估结果表明该方法有效。我们指出，尽管本研究主要受化学反应网络带来的挑战所驱动，但其方法可推广至所有连续时间马尔可夫链模型。

关键词： 向量加法系统、概率模型检测、显式状态表示、数据结构

1 引言

现实世界中的连续时间概率系统，包括化学反应网络 （CRN）和其他随机向量加法系统 （VAS），会导出连续时间马尔可夫链 （CTMC）模型。对连续时间马尔可夫链进行形式化分析，是安全关键系统研究的重点，尤其是因为它们与合成生物设计 [11,23,35] 以及通信系统 [18], 、动态功率管理 [42],、自动驾驶车辆控制 [14], 和其他关键过程 [44]的建模密切相关。化学反应网络固有的复杂性

源于其状态空间往往无限或大到难以处理，这使分析和状态存储 [11]都变得复杂。概率模型检测 （PMC）能够为安全关键系统提供可证明的定量可靠性保证，并有望在设计早期揭示缺陷和错误。然而，模型庞大或无限的状态空间，加上缺乏先验模型信息，给最先进的概率模型检测工具带来了严峻挑战。

显式状态表示——在每个状态中存储每个状态变量的真实数值——是执行连续时间马尔可夫链有界时间瞬态数值分析所必需的。这意味着，诸如二元决策图（BDDs）之类的符号状态表示，必须先转换为显式状态表示，才能执行瞬态分析 [26]。包括 Storm 在内的前沿概率模型检测工具 [27]将显式状态存储在哈希映射中，从而平衡内存和时间复杂度。然而，许多看似简单的化学反应网络模型仍然需要大到无法接受的内存容量 [43,32,4]。

我们提出将前缀树数据结构创新性地应用于显式连续时间马尔可夫链状态存储，以大幅降低超大状态空间的内存消耗。通过减少存储数据的重复，前缀树在保持常数级查找和插入时间的同时，降低了超大状态空间所需的内存。从理论角度来看，前缀树与哈希映射具有相同的复杂度，并有望节省内存。本文所 प्रस्तुत的基准测试结果表明，前缀树在降低内存使用量方面非常有效，通常只需占用哈希映射约一半的内存。尽管本研究的动机来自化学反应网络，但该方法适用于所有连续时间马尔可夫链模型，并且非常适合所有向量加法系统。

由于前缀树带来的内存节省取决于变量的排序，我们进一步提出一种基于有界模型检测 (BMC) [8] 的预处理步骤，该步骤提供一种启发式方法，以实现额外的内存节省，尤其是在引导式状态空间探索期间。初步评估支持了这一方法。

论文结构。 第 2 节介绍改进显式状态表示内存使用量所面临的挑战，并阐述我们的目标。第 3 节介绍背景信息和相关工作。第 4 节提出一种前缀树状态表示方法，该方法克服了这些挑战，基准测试结果也证明了这一点。第 5 节介绍一种利用 BMC 在引导式状态空间探索期间提高内存节省效果的预处理方法。第 6 节总结全文。

2 问题陈述

连续时间马尔可夫链瞬态分析需要采用显式状态表示。在高度并发的模型中，包括化学反应网络和许多向量加法系统，即使使用最先进的工具，也可能需要耗费难以承受的内存。我们的首要目标是在保留现有方法的时间复杂度和显式状态表示的同时，减少存储大型状态空间所需的内存。

2.1 动机

分析化学反应网络的一大挑战是其状态空间往往无限大或大到难以处理。由于设计阶段缺乏相关知识，变量上界通常采用非常保守的估计，以避免排除关键行为。因此，化学反应网络的状态空间可能包含数十亿甚至数万亿个状态，轻易就会超出用于分析这些模型的计算机的内存容量。对于合成生物学模型而言，连续时间马尔可夫链的有界时间瞬态可达概率尤为重要，而这类模型往往给现有工具带来巨大挑战 [11,23,16,1]。因此，需要采用专门的方法，因为现有数据结构通常难以满足最先进方法的需求 [11,4]，包括状态空间截断 [32] 和引导式状态空间构造 [43]。

我们的工作专门针对大规模化学反应网络所面临的独特挑战，旨在提升其分析与设计的效率和效果。我们提出了一种显式状态表示方法，适用于广泛的模型，包括化学反应网络和其他向量加法系统，以及状态可表示为向量的概率模型和非概率模型。我们尤其关注化学反应网络在进行连续时间马尔可夫链瞬态分析时所面临的内存挑战。我们的数据结构经过专门设计，可改善高度并发的向量加法系统的内存使用量。

2.2 目标

要克服化学反应网络和向量加法系统状态存储所面临的内存瓶颈，需要满足若干要求。为了降低内存消耗，相较于前沿工具所用的哈希映射，我们确立了以下目标：

目标 1：降低内存使用量。为了能够分析规模极大的模型，我们的方法必须比前沿工具消耗更少的内存；这些工具目前在状态空间探索过程中使用哈希映射存储状态。这是最迫切的挑战，因为现有工具无法有效分析某些状态空间庞大的超大型化学反应网络模型 [29]。要实现这一目标，就必须在最坏情况下保持或改善哈希映射的内存复杂度，同时在实证评估中展现更低的内存使用量。

目标 2：维持时间复杂度。哈希映射的查找和插入时间平均为常数级。我们的方法应保留这些特性，不过可能需要在时间和内存之间进行一定权衡。实现这一目标需要保留哈希映射的最坏情况时间复杂度。

目标 3：保留显式状态空间。必须能够从我们的表示中提取状态，以构建显式状态空间，从而保留执行连续时间马尔可夫链瞬态分析的能力。

3 背景与相关工作

现有文献和工具突显了探索这些状态空间所面临的挑战，尤其是在罕见事件发生概率较低的情况下。

例如，Prism [34], Storm [27],Modest 的 mcsta 工具 [25],和 Stamina [32] 都是通用的概率模型检查器，采用各种状态空间约简技术。Modest 工具集 [25] 针对罕见事件进行统计模型检验 [10], ，生成概率近似值，而非执行完整的连续时间马尔可夫链瞬态分析。

3.1 化学反应网络与向量加法系统

化学反应网络 （CRN）是一种用于模拟生物和化学过程中的化学反应动态的语言。形式上，化学反应网络 $\mathcal{M}$ 是一个元组 $\mathcal{M} = \langle \mathfrak{X}, \mathfrak{R}, \boldsymbol{s}_0 \rangle$，包含 n 个反应 $\mathfrak{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$，作用于 m个变量（化学物种） $\mathfrak{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$，其中反应 $\mathcal{R}_i$： $\alpha\mathcal{X}_1 \xrightarrow{k_i} \beta\mathcal{X}_2$ 可读作“反应 i 消耗 α 个 $\mathcal{X}_1$ 分子，并以反应速率常数 k_i生成 β 个 $\mathcal{X}_2$ 分子”。初始状态 $\boldsymbol{s}_0$ 定义为$|\boldsymbol{s}_0| = m$ 且 $\forall 0 \leqslant i < m \cdot 0 \leqslant \boldsymbol{s}_0[i] < \infty$。化学反应网络通常遵循随机化学动力学（SCK）模型假设，该假设将每个反应的反应物限制为两个，并要求每个反应几乎瞬时发生 [38]。

化学反应网络会导出一个连续时间马尔可夫链，其中状态变化（由反应引起）以离散量发生，但状态变化的概率由相对于实数时间定义的速率决定。连续时间马尔可夫链 $\mathcal{C}$是一个元组 $\mathcal{C} = \langle \mathbf{S}\ \boldsymbol{s}_0, \mathbf{R}, \mathbf{L} \rangle$，其中 $\mathbf{S}$ 是表示状态空间的一组状态， $\boldsymbol{s}_0 \in \mathbf{S}$ 是初始状态， $\mathfrak{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ 定义转移速率矩阵，而 $\mathbf{L} : \mathbf{S} \to 2^{AP}$是一个以原子命题 AP为标记的状态标记函数。转移速率$\mathbf{R}(\boldsymbol{s}, \boldsymbol{s}')$ 由 $\mathcal{R}_i$的倾向率决定，假设 $\mathcal{R}_i$ 是导致该状态变化的唯一反应。若反应 $\mathcal{R}_i$ 在特定状态 $\boldsymbol{s}$ 下发生的概率非零，则称其已启用。

向量加法系统 （VAS）也称为佩特里网，用于描述基于向量相加而改变状态的迁移系统。形式上，VAS $\mathcal{V}$是一个元组 $\mathcal{V} = \langle m, \boldsymbol{s_0}, R \rangle$，其中 m是变量的数量（即每个向量的长度）， $\boldsymbol{s_0}$是初始状态， $R = \{\boldsymbol{r_1}, \ldots \boldsymbol{r_n}\}$是转移集合，且满足 $\boldsymbol{s_0} \in \mathbb{Z}^m_{\geqslant 0}$ 和 $\forall i \in [1, n] \cdot r_i \in \mathbb{Z}^m$。

随机向量加法系统为每个转移的执行关联一个速率（对于连续时间）或概率（对于离散时间）。化学反应网络会诱导出一个随机向量加法系统，因为转移被限制为在一组可表示为向量 [13] 的状态之间进行加法。就我们的目的而言，将状态表示为非负整数向量即可，使得对于任意两个状态$\boldsymbol{s}_\alpha$， $\boldsymbol{s}_\beta$, $|\boldsymbol{s}_\alpha| = |\boldsymbol{s}_\beta|$ 和 $\forall i \in [0, m] \cdot 0 \leqslant \boldsymbol{s}_\alpha[i] < \infty \wedge 0 \leqslant \boldsymbol{s}_\beta[i] < \infty$。虽然我们针对化学反应网络对方法进行了专门设计，但该方法可推广到所有连续时间马尔可夫链。

3.2 现实生活中的化学反应网络

我们的示例是经过修改的酵母极化模型 [17], ，该模型改编自酵母属中的信息素诱导的G蛋白循环。

visia [19]，其中配体群体保持恒定，使其无法达到平衡 [41]。这是一个包含 7 种物种和 8 个化学反应的化学反应网络：

$$\begin{aligned}
&\mathcal{R}_1:\ \emptyset \xrightarrow{0.0038} \mathrm{R}, && \mathcal{R}_2:\ \mathrm{R} \xrightarrow{4.00\times10^{-4}} \emptyset, && \mathcal{R}_3:\ \mathrm{L}+\mathrm{R} \xrightarrow{0.042} \mathrm{RL}+\mathrm{L},\\
&\mathcal{R}_4:\ \mathrm{RL} \xrightarrow{0.010} \mathrm{R}, && \mathcal{R}_5:\ \mathrm{RL}+\mathrm{G} \xrightarrow{0.011} \mathrm{G_a}+\mathrm{G_{bg}}, && \mathcal{R}_6:\ \mathrm{G_a} \xrightarrow{0.100} \mathrm{G_d},\\
&\mathcal{R}_7:\ \mathrm{G_d}+\mathrm{G_{bg}} \xrightarrow{1.05\times10^{3}} \mathrm{G}, && \mathcal{R}_8:\ \emptyset \xrightarrow{3.21} \mathrm{RL}.
\end{aligned}$$

所有反应倾向率均以每秒分子数表示。对于物种$[R, L, RL, G, G_a, G_{bg}, G_d]$，初始状态为 $\boldsymbol{s}_0 = [50, 2, 0, 50, 0, 0, 0]$。由于物种数量在理论上没有上限，这个简单模型面临严重的内存问题。该模型中用于瞬态分析的目标CSL性质为$\mathrm{P}_{=?}(\Diamond^{[0,20]}\ \mathrm{G_{bg}} = 50)$，它询问："在20秒内，物种$\mathrm{G_{bg}}$ 的数量达到50的概率是多少？"这是一个罕见事件，因为这种情况发生的概率极低，而且分析需要极其庞大的状态空间 [43]。因此，要准确计算概率，就必须枚举极其庞大的状态集合。

我们还将这种数据结构应用于多个规模庞大且复杂的化学反应网络，具体来说，是基因电路。基因电路是经过工程化设计的DNA序列，旨在细胞内执行用户定义的功能，其运作方式类似于计算机中的数字电子电路 [38]。这些电路通过调控基因表达来处理生物输入（例如某种化学物质或环境信号的存在），并生成输出（例如荧光蛋白、代谢变化或其他细胞行为）。

有若干模型基于遗传电路 *0x8E*，该电路最初由 Nielsen等人提出 [39]。基于该电路的七个模型均包含 18 种物种和 15 个反应。电路 0x8E 的另一种变体最早在 [21], 提出，它采用相同逻辑函数的另一种逻辑实现，旨在缓解实验室环境中观察到的错误行为 [39]。该模型包含 19 种物种和 16 个反应。

其他模型基于*Muller C-element* [35]的遗传实现。C 元件是异步电路设计中常用的重要状态保持门，用于协调并行进程。该电路采用速度无关设计，无论门状态变化快慢，都能确保行为正确。研究中使用了一个详细模型和一个抽象模型。每个模型均包含 20 种物种和 18 个反应。

3.3 前缀树

前缀树也称为字典树，是一种基于图的数据结构，已广泛用于自然语言处理。前缀树能够高效地存储和检索字符串，因为它利用存储数据中的公共前缀，实现快速查找和紧凑存储。在状态表示方面，我们倾向于使用前缀树，而不是相关的替代方案（包括R树、X树和基数树），这是因为前缀树简单、表示明确，且时间和内存开销较低。它们通常用于根据前缀快速搜索和检索单词、短语或其他文本元素 [22,33]。

前缀树还被用于压缩和存储数值数据集；在通用前缀树实现PTrie中，其性能优于哈希映射 [31]。然而，前缀树在随机系统验证和分析中的应用尚未得到研究。现有最先进的连续时间马尔可夫链分析工具主要使用哈希映射进行状态存储。我们将自己的前缀树实现与Storm进行基准测试 [27], ，后者使用3位 Murmur 哈希存储状态[4]。哈希映射之所以广受欢迎，是因为其时间和内存效率较高，但在处理内存需求高的状态空间时会面临挑战。在最坏情况下，前缀树的内存复杂度不会超过哈希映射；而在平均和最佳情况下，它能够显著节省内存。第4.6节中的结果证实了这一点。

我们的前缀树实现采用稀疏树结构；每个节点都包含一个小型哈希映射，以便快速查找后继状态。尽管前缀树的实现方式多种多样，但我们发现，此实现能够在运行时间与内存之间实现合理的权衡。

3.4 二元决策图

二元决策图 [9] 是一种高效的符号化数据结构，用于表示状态。二元决策图能够高效存储和探索模型 [5,12,37]的状态空间。二元决策图的一项主要优势是能够利用布尔函数中的冗余，从而与显式状态枚举相比减少内存使用量。

尽管二元决策图已成功解决多个领域中的状态空间爆炸问题，但其适用性往往受限于性能对变量排序的敏感性。寻找最优变量排序可能颇具挑战，而次优排序可能导致二元决策图的规模显著增大，从而抵消这种数据结构 [40]带来的优势。

为了解决二元决策图的局限性，研究人员提出了多终端二元决策图（ MTBDDs） [3,24,28]。MTBDDs 允许包含多个终端节点，因此能够表示取值范围而非仅取布尔值的函数。这使 MTBDDs 适用于对具有定量属性的系统进行建模和分析，例如转移奖励。现有的连续时间马尔可夫链分析工具（如 Prism 和 Storm）已采用基于二元决策图的技术，来表示和操作所研究模型的状态空间。默认情况下，这两种工具都会构建二元决策图，因为连续时间马尔可夫链稳态分析不需要显式状态表示。然而，连续时间马尔可夫链瞬态分析需要显式状态表示来进行概率计算。

前缀树利用化学反应网络的并发性来减少内存使用量。与二元决策图类似，我们发现前缀树中的变量排序会影响内存消耗。然而，二元决策图以针对布尔运算优化的方式符号化存储状态，而所提出的前缀树则保留显式状态存储，从而支持完整的连续时间马尔可夫链瞬态分析。

[4] Storm 的哈希映射功能在以下开源代码中实现：
https://github.com/moves-rwth/storm/blob/master/src/storm/storage/BitVectorHashMap.h

4 用于状态表示的前缀树

前缀树满足高效状态表示的设计目标。通过利用并发性减少数据重复，前缀树在保持时间复杂度和实用性的同时，相较于哈希映射还能节省内存。尽管所提出的方法是为化学反应网络设计的，但也可扩展到所有连续时间马尔可夫链模型。因此，本节采用第3.2节中所述的化学反应网络和向量加法系统的语法与语义。

4.1 数据结构与算法

在自然语言处理中，前缀树之所以高效，是因为许多单词共享前缀。通过只存储每个前缀一次，前缀树避免了重复存储字符。类似地，在高度并发的模型中，尤其是在化学反应网络中，每次执行的跃迁只会更新一小部分变量。因此，给定合适的变量排序，许多状态会共享一个“前缀”，从而避免重复存储这些前缀。

前缀树 $\mathcal{T}$是一种有向无环图，其根节点为 $\mathcal{T}$.root。对于具有 m 个变量的模型，该图的深度为 $m+1$，即根节点加上每个变量对应的一层。必须对模型的物种施加严格序 $<$，使得深度为 i 的所有节点都对应于 $\mathcal{X}_i$的取值。如果 $\mathcal{X}_i$ 的一个取值与 $\mathcal{X}_{i+1}$ 的一个取值之间存在一条边，其中 $\mathcal{X}_i < \mathcal{X}_{i+1}$，则表示存在一种状态，其中 $\mathcal{X}_i$ 和 $\mathcal{X}_{i+1}$ 的相连取值同时出现。

例如，在一个包含变量 $\mathfrak{X} = \{A, B, C\}$ 的模型中，若 $A < B < C$，则状态 $[2, 0, 5]$ 在前缀树中表示为 $\mathcal{T}$.root $\to 2 \to 0 \to 5$。图 1 进一步展示了这一示例，其中给出了状态空间（左侧）及其生成的前缀树（右侧）。图中，状态空间按每个状态一行表示，而前缀树采用的排序为 $A < B < C$。

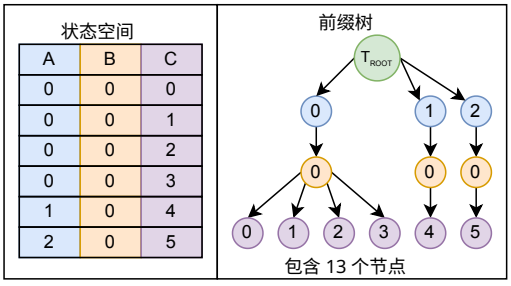

图 1. 从状态空间构建前缀树的示例

算法1形式化描述了所提出的数据结构中的插入、查找和状态空间提取操作。简而言之，该数据结构支持以下操作：

插入。 通过遍历树，沿着与状态变量有序取值对应的现有边前进，将新状态添加到树中。在第一个不存在的变量值之后创建节点和边。

查找。 遍历前缀树，沿着与状态变量的有序取值相对应的路径前进。如果从根节点到叶节点存在一条完整路径，则表示已找到该状态。

状态空间提取。 通过对该数据结构进行深度优先遍历，枚举前缀树中存储的所有状态。

以图1为例，插入第一个状态 $[0,0,0]$, 需要创建三个新节点。然而，插入下一个状态 $[0,0,1]$, 时，只需创建一个新节点。通过图遍历找到共享前缀 $[0,0]$ ，然后在前缀树的最低层添加节点 1。对每个状态重复这一过程，直到构建出完整的前缀树。

算法1前缀树操作

数据结构：
$\mathcal{T}$ =前缀树
$\mathcal{T}$.根节点 = 前缀树的根节点
$\mathcal{T}$.isTerminal(n) = 如果节点 n 是终端节点，则返回 true

过程 插入($\mathcal{T}$, s) ▷ s 是要插入的序列（状态）
  $n \leftarrow \mathcal{T}$.根节点
  对于 每个 $c \in s$ **执行**
    如果 n 没有标记为 c 的子节点**则**
      n.添加子节点(c)
    $n \leftarrow n$.获取子节点(c)
  n.标记为终端节点()
过程 搜索($\mathcal{T}, s$) ▷ s是要搜索的序列（状态）
  $n \leftarrow \mathcal{T}$.根节点
  对每个 $c \in s$ **执行**
    如果 n没有标签为 c 的子节点**则**
      返回 false
    $n \leftarrow n$.获取子节点(c)
  返回 $\mathcal{T}$.是终端节点(n)
过程 ExtractStates($\mathcal{T}$) ▷提取所有存储序列（状态）
  $\mathcal{S} \leftarrow \emptyset$ ▷ $\mathcal{S}$ 是生成的状态空间构建
  $\mathcal{S} \leftarrow$ ExtractStatesRecursive($\mathcal{T}$.根节点 [ ], $\mathcal{S}$)
  返回 $\mathcal{S}$
过程ExtractStatesRecursive(节点，序列， $\mathcal{S}$)
  如果 $\mathcal{T}$.isTerminal(node) **则**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{$ 序列$\}$
  对于 node 的每个子节点 c **执行**
    $\mathcal{S} \leftarrow$ ExtractStatesRecursive(c, 序列 $+ c$, $\mathcal{S}$)
  返回 $\mathcal{S}$

4.2 目标1评估：减少内存使用量

在最坏情况下（即没有状态共享公共前缀），内存复杂度与哈希映射相同：所有值都只存储一次。不过，前缀树通过将前缀分组并减少存储完整哈希表的需求，降低了平均内存占用。以图1为例，哈希映射会显式存储左侧表格中的数据，即总共 18 个值。此外，为避免冲突，哈希映射会预留额外内存，因此，追求时间效率的哈希映射可能会为 $next_prime(18 \times 2)$ 分配内存，即 37 个值。由于前缀树合并了共享前缀，因此只存储 13 个值。

前缀树的内存复杂度。算法1中的插入过程是唯一会导致内存消耗的过程。该过程的主循环恰好迭代 m 次（其中 m 是模型中的变量数量），最多可向前缀树中添加总计 m 个新节点。如果第4行的条件为假，则不会消耗额外内存；否则，就会创建一个新节点并消耗内存。如果该条件始终为真（即该操作消耗尽可能多的内存），则总共会添加 m 个新节点。因此，在最坏情况下，添加一个新状态需要 $O(m)$ 内存。由此，插入 n 个状态的最坏情况内存复杂度为 $O(n \times m)$。在最佳情况下，所有状态共享一个前缀，且每个状态只有最后一个变量不同，此时存储单个状态的内存复杂度可显著降低至最佳情况： $\Omega(n \times 1)$。实践中，第4.6节所示的基准测试表明，平均内存使用量介于这两个极端之间。

前缀树所实现的内存节省程度，取决于状态空间的结构和其中的冗余。状态之间相似度较高的模型（例如化学反应网络）非常适合采用基于前缀树的存储，因为这种结构能够利用共同的模式。

哈希映射的内存复杂度。相比之下，存储相同状态的哈希映射，其内存复杂度固定为 $O(n \times m)$。这是因为所有状态的所有 m值都会完整地存储在哈希映射中。实际上，哈希映射还会产生大量额外的内存开销，因为哈希表会预留过多内存，以避免冲突并维持恒定的平均时间复杂度。哈希映射为每个变量存储一个节点所需的等量内存，因此其最佳情况下的内存消耗也是 $\Omega(n \times m)$。

比较。在最坏情况下，哈希映射和前缀树的内存复杂度均为 $O(n \times m)$。然而，前缀树能够避免重复，因此其内存消耗可降至 $\Omega(n \times 1)$；而哈希映射的复杂度则固定为 $\Omega(n \times m)$。实际上，第4.6节中的基准测试证实了这一点：与哈希映射相比，内存消耗通常可减少 50% 以上。

4.3 目标 2 的评估：维持时间复杂度

哈希映射通常很有吸引力，因为其查找和插入时间是常数级的。为了与哈希映射竞争，前缀树必须提供常数级或

近似常数级的时间复杂度。以下分析展示了具有 m 变量和 n状态的模型的时间复杂度。

前缀树操作的时间复杂度。插入和搜索过程各包含一个循环，恰好迭代 m 次，因此每个过程的时间复杂度均为 $O(m)$。ExtractStates 过程采用深度优先遍历，时间复杂度为 $O(m \times n)$，因为它最多遍历 n 条深度为 m 的路径。在大多数实际模型中，我们观察到 $m \ll n$。
哈希映射操作的时间复杂度。考虑一个具有 m 个变量和 n 个状态的模型。在采用高效哈希算法的哈希映射中，插入和查找的平均时间复杂度均为 $\Theta(1)$，而最坏情况下的时间复杂度为 $O(n)$ [15]。提取状态空间需要枚举恰好 n 个状态，因此其复杂度为 $\Omega(n)$。
比较。前缀树和哈希映射的时间复杂度相当。对于每个模型，变量数量都是常数（且通常较少）。因此，我们认为前缀树的插入和查找复杂度仍可与哈希映射相竞争，不过前缀树会带来少量时间开销。虽然前缀树的最坏情况时间开销为 $O(m)$，而哈希映射的平均情况时间开销为 $\Theta(1)$，但前缀树带来的内存节省通常足以弥补额外的时间开销，尤其是对于非常大型的化学反应网络模型而言，因为内存消耗是其主要限制因素。

4.4 目标3的评估：保留显式状态空间

由于算法1给出了一个从前缀树中提取显式状态空间的过程（ExtractStates），该方法在节省内存的同时保留了显式状态表示的实用性。前缀树保留了连续时间马尔可夫链瞬态分析所需的显式状态表示。这种显式状态表示可由概率模型检查器解读，从而与现有验证工具和工作流程保持兼容。

4.5 前缀树实现

我们使用原生 C++ 实现了一个自定义前缀树，并链接 glibc，以与 Storm 保持一致。每个节点都存储一个小型哈希映射，将物种值映射到后继状态，以缓解分支因子过大的问题。两个基准测试程序执行完全相同的状态空间探索过程，分别使用 Storm 内置的哈希映射或所提出的前缀树来存储和搜索状态。为此，我们扩展了 Storm 的压缩位向量类（该类用于存储状态变量值），使其支持索引（即根据每个连续时间马尔可夫链变量在 BitVector 对象中的位置为其分配唯一索引）。我们利用该索引重载位向量类的 C++ [] 运算符和迭代功能，从而可以逐个变量地迭代解析状态，以便在前缀树中进行查找和插入。

基准测试程序会对指定步数进行广度优先状态空间探索，并测量插入和查找操作的耗时，

同时记录内存使用量。基准测试是在第3.2节所述的模型上进行的，测试机器配备 AMD Ryzen Threadripper 12 核 3.5 GHz 处理器和 132 吉字节随机存取存储器，运行 Ubuntu Linux 22.04.5。基准测试程序和模型已在 GitHub 上开源[5]。

将前缀树集成到兼容 Storm API 的工具中相对简单。此类集成可让用户选择降低检查具有庞大状态空间的模型所需的内存。将前缀树数据结构集成到 PMC 工具中用于状态存储，或可提升对庞大、高并发模型的易用性。

4.6 前缀树基准测试

我们针对第3.2节所述具有庞大状态空间、真实且具有挑战性的模型的不同变体，展示了基准测试结果，对每个模型在 10^i 步内逐步探索并构建状态空间，其中 $i \in [3, 8]$。此范围用于平衡状态空间大小与对大量模型进行基准测试的需求。我们分析了每种模型与步数组合的查找时间、插入时间和总内存消耗。

表1汇总了前缀树相较于 Storm 原生哈希映射的改进。每一行表示所有模型的平均改进。表中依次展示以下数据：

步数。 已探索的步数。一步是执行一次转移的结果。**查找**。 与哈希映射相比，前缀树在查找时间上的改进。**插入**。 与哈希映射相比，前缀树在插入时间上的改进。**哈希映射**。 使用哈希映射时消耗的内存。**前缀树**。 使用前缀树时消耗的内存。**内存百分比**。 与哈希映射相比，前缀树节省的内存百分比。

时间改进值为负，是因为前缀树的时间复杂度略高于平均哈希映射的时间复杂度。随着状态空间的大小（与步数相关）增加，内存节省也大幅增加。平均而言，前缀树的内存需求低于哈希映射，而更大的状态空间会带来更多利用共享前缀的机会。正如预期，前缀树的查找和插入时间略高于哈希映射。在需要数小时进行分析且设备内存受到严重限制的模型中，即使是观察到的最大时间开销（约15分钟），相较于由此带来的内存节省（约12吉字节）也很可能是值得的。我们观察到，对于大型状态空间，前缀树数据结构始终消耗不到一半的内存，少于 Storm 的原生哈希映射。

[5] 基准测试脚本和Storm接口的开源地址为
https://github.com/formal-verification-research/crn-prefix-tree

表1。按步数统计的平均改进

步数	查找	插入	哈希映射	前缀树	改进
10^{3}	0.00 秒	-0.01 秒	43.09 MB	42.77 MB	0.73%
10^{4}	-0.01 s	-0.09 s	45.79 MB	42.91 MB	6.29%
10^{5}	-0.12 秒	-0.93 秒	70.49 MB	50.99 MB	27.65%
10^{6}	-0.97 秒	-9.35 秒	291.59 MB	135.65 MB	53.48%
10^{7}	-6.63 秒	-94.79 秒	2.35 吉字节	0.99 吉字节	57.63%
10^{8}	-29.77 秒	-946.56 秒	21.61 吉字节	9.58 吉字节	55.67%

表2展示了内存节省表现最佳的10个模型（单位：字节）。我们观察到，正如预期，高并发模型往往能提供更好的内存节省。此外，随着状态空间的增长，内存节省也会增加，因此最大的状态空间能够实现最多的节省。

我们还观察到，在这些模型中，我们能够将内存消耗降低*50%*以上；也就是说，只需付出合理且较小的时间权衡，就能大幅节省内存。我们认为，这种数据结构不仅能提升内存使用效率，还能让此前无法存储的某些状态空间变得可行。例如，对于表2中的许多模型，配备32吉字节内存的机器使用哈希映射无法有效存储，但使用前缀树则能轻松存储。此外，对这些模型进行连续时间马尔可夫链瞬态分析时，构建完整状态空间可能需要数小时到数天 [29], ，因此我们认为，数秒到数分钟的时间开销并不过分。

表2。内存节省最多的前10个模型

模型	步数	哈希映射	前缀树改进	
Circuit0x8E _000至011	10^{8}	34.14 吉字节	12.86 吉字节	62.33%
Circuit0x8E _000到101	10^{8}	34.14 吉字节	12.86 吉字节	62.33%
Circuit0x8E _LHF_000至011	10^{8}	35.40 吉字节	14.23 吉字节	59.80%
Circuit0x8E _101至011	10^{8}	33.97 吉字节	12.89 吉字节	62.06%
Circuit0x8E _101to000	10^{8}	33.97 吉字节	12.89 吉字节	62.06%
Circuit0x8E _011to101	10^{8}	33.94 吉字节	12.88 吉字节	62.08%
Circuit0x8E _011to000	10^{8}	33.94 吉字节	12.88 吉字节	62.08%
速度 _独立 _10_10	10^{8}	27.68 吉字节	8.84 吉字节	68.06%
简化运动调节	10^{8}	21.86 吉字节	9.06 吉字节	58.55%
速度 _独立	10^{8}	19.68 GB	8.75 GB	55.54%

5 预处理以降低引导式状态空间探索的内存使用量

当对大型模型进行概率模型检测需要耗费难以承受的资源时，引导式状态空间探索是一种很有吸引力的替代方案。现有工具包括 Rag-timer [43], ，它从一组可达轨迹出发探索状态空间；以及 STAMINA [32], ，它通过截断状态空间来减轻模型的内存负担。这些工具依赖于节省内存的状态存储方法。

状态空间探索和截断很大程度上依赖于实时存储和查找状态的能力。由于引导式探索中希望探索尽可能大的状态空间子集，因此增加有限内存中可存储的状态数量是有益的。直观而言，状态空间越大，获得准确概率界限的可能性就越高。因此，减少每个状态消耗的内存，可以构建更大的状态空间，从而改进概率计算。

利用前缀树中的变量排序。 节省内存的一种方法是利用前缀树中的变量排序。为此，我们首先计算每个变量的有限步界限，以预测每个变量可达值的范围。然后，我们利用这些界限生成高效的变量排序，并将其用于构建前缀树。近期研究表明，有界模型检测 （BMC）是为化学反应网络（CRN）综合变量界限的有效方法 [2]。我们通过生成一系列界限来扩展这一方法。

迁移系统与有界模型检测。符号转换系统$\mathcal{S} = \langle X、 I、 T\rangle$是一个元组，其中 X是（状态）变量的有限集合， I（X）是表示系统初始状态的逻辑公式，而 T（X、 X'）是描述其迁移关系的公式，其中 X' 是将 X 中的每个元素 x替换为 x'后得到的集合。 $\mathcal{S}$ 的状态 s_i 是对 X中变量的赋值。长度为 k 的 $\mathcal{S}$的轨迹 $\sigma_k = s_0、 s_1、 s_2、 .....、 s_{k-1}$是状态序列，其中 s_0满足 I（X），并且对于所有 $0 \leqslant i < k-2$，将X重命名为 X'后的 $s_i \wedge s_{i+1}$满足 T（X、 X'）。将 x 替换为 x@i 后得到的集合记为 X@i。有界模型检测（BMC） [8,7]是一种符号模型检验技术，它借助SAT [36]或SMT [6] 求解器，检查系统从初始状态开始、长度不超过固定值的所有路径。对于迁移系统 $S = \langle X、 I、 T\rangle$ 和不变式性质P（X），即某个可达性性质的否定，BMC以如下形式向底层SAT/SMT求解器提交一系列证明义务： BMC（S、 P、 k） $=I$（X@0）$\wedge T$（X@0、 X@1）$\wedge \ldots \wedge T$（X@k-1、 X@k）$\wedge \neg P$（X@k），逐步增大 k 的值，直到发现一条轨迹。

5.1 变量排序示例

考虑图 2 中的简单示例。使用默认顺序 $A < B < C$ 时，前缀树包含 13 个节点。然而，将顺序改为 $B < A < C$ 后，状态空间大小减少到 11 个节点。使用最不理想的顺序 $C <$

$A < B$ 时，前缀树包含 19 个节点，消耗的内存几乎是原来的两倍。这正是前缀树内存效率的关键所在：哈希映射总是会存储全部 18 个变量值，而前缀树则可以通过避免重复存储，大幅减少内存占用。

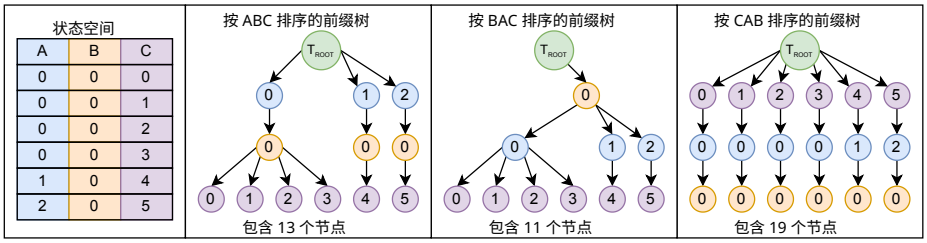

图2. 从状态空间构建前缀树示例

如果每个变量 $\mathcal{X}_i$ 的取值总数由 $|\mathcal{X}_i|$ 给出，则最优排序要求 $\mathcal{X}_i < \mathcal{X}_j \iff |\mathcal{X}_i| < |\mathcal{X}_j|$。这造成了一个两难局面：最优变量排序需要探索并存储整个状态空间，但在进行状态空间探索和存储之前，必须先确定变量排序。因此，我们提出使用基于BMC的启发式发现作为预处理步骤，以便在无需完整状态空间探索的情况下为变量界定上界。

5.2 变量界计算

与概率无关的预处理步骤可以生成合适的变量排序。该方法使用BMC来发现变量界。由于这些界可能过于严格，无法涵盖所有关键行为，因此我们不会将这些界强加于模型；而是仅用它们来构建一种启发式方法，再利用该启发式方法生成变量排序。

开始预处理时，我们按照 [29] 中的描述为模型构建依赖图。直观而言，依赖图会确定为达到目标状态而必须执行的跃迁，然后递归推导出启用这些跃迁所需执行的跃迁。

在第3.2节所述的示例中，$\mathcal{R}_5$ 是唯一能够增加目标变量 G_{bg} 计数的反应，因此将其添加为依赖项。由于 RL 不足，$\mathcal{R}_5$ 最初未启用，因此将 $\mathcal{R}_3$ 和 $\mathcal{R}_8$ 添加为 $\mathcal{R}_5$的依赖项。由于 $\mathcal{R}_3$ 和 $\mathcal{R}_8$ 均能够执行足够多次，从而启用 $\mathcal{R}_5$ 产生足量的 G_{bg} 以满足该性质，因此依赖图最终包含节点 $\mathcal{R}_5$、$\mathcal{R}_3$和 $\mathcal{R}_8$。

随后，我们将跃迁和初始条件编码为与概率无关的符号转换系统。尽管该迁移系统抽象掉了概率信息（纯粹关注可达性），但感兴趣的概率属性仍是整体分析的重点。由此得到一个 BMC

公式，长度为 k （在我们的案例中使用 Yices 2 [20]）。我们从 $k = 0$ 开始，并使用 SMT求解器检查 BMC公式。如果求解器返回不可满足，则表示目标在 k 步内不可达，因此我们将 k 的值加 1 并重复检查。当求解器返回 SAT 时，则表示目标在 k 步内可达，并且我们已找到一条从初始状态到达目标的路径。求解器提供的赋值给出了一条轨迹。为了找到界限，我们将 BMC公式与以下属性结合使用，从而为每个物种或变量获得两组界限。

第一组界限表示最紧界限，即允许执行至少一条反例轨迹的最严格界限，也就是从初始状态到目标状态的转换序列。我们迭代地断言可达性公式与每个变量的上界和下界同时成立，直到无法找到满足这些界限的轨迹。

第二组界限表示最宽松界限，即允许所有反例轨迹执行的限制最少的界限。我们迭代地断言可达性公式与每个变量的上界和下界至少有一个成立，直到界限之外不存在反例轨迹。这些界限用作确定前缀树变量排序的主要启发式方法。

5.3 基于界限的排序

对于变量 $\mathcal{X}_i$，其值的总数在此阶段尚未知，由 $|\mathcal{X}_i|$ 给出。通过上述过程得到的最紧下界和最紧上界分别为 Tight($\mathcal{X}_i$) $= [TL(\mathcal{X}_i)\ TU(\mathcal{X}_i)],$。同样，最宽松下界和最宽松上界分别为 Loose($\mathcal{X}_i$) $= [LL(\mathcal{X}_i)\ LU(\mathcal{X}_i)],$。例如，如果真实值为$|\mathcal{X}_i| = \{3, 4, 6, 19\}$，可能的紧界限和宽松界限分别为 Tight($\mathcal{X}_i$) $= [3, 6]$和 Loose($\mathcal{X}_i$) $= [3, 19]$。由此可知，至少存在一个转换序列可到达满足目标的状态，并导致 $\mathcal{X}_i \in [3, 6]$；而每一个通往满足目标的状态的转换序列都会产生 $\mathcal{X}_i \in [3, 19]$。

为了对一对变量进行排序，我们首先为变量 $\mathcal{X}_i$ 定义以下术语： $L(\mathcal{X}_i) = LU(\mathcal{X}_i) - LL(\mathcal{X}_i)$、 $T(\mathcal{X}_i) = TU(\mathcal{X}_i) - TL(\mathcal{X}_i)$。以下算法对一对物种（$\mathcal{X}_i$, $\mathcal{X}_j$）进行排序 $<$：

1. 如果 $L(\mathcal{X}_i) < L(\mathcal{X}_j)$，返回 $i < j$2.
如果 $T(\mathcal{X}_i) < T(\mathcal{X}_j)$，返回 $i < j$3. 返回 random($j < i$, $i < j$)

直观而言，最宽松界限的范围较大的变量或物种，其方差可能比范围较小的变量或物种更大。如果宽松界限相同，那么最严格界限的范围较大的变量或物种，其方差可能比范围较小的变量或物种更大。如果范围相同，则通过抛硬币决定顺序。

这种排序可以生成内存高效的前缀树。由于未能完整探索

模型，因此无法保证这种排序是最优的。不过，我们认为，使用有界模型检测生成这种排序所需的少量时间（通常不到五分钟），值得换取状态空间极大的模型在内存使用量上的改善。

5.4 预处理评估

我们通过 Yices 2 [20] 的 Python API，将 C++ 基准测试与其集成，以进行基于有界模型检测的预处理。预处理脚本实现了第 5 节中描述的技术，然后使用 pybind [30] 库将变量排序传递给 C++ 基准测试程序。

在修正版酵母极化模型中探索 10^8 步数时（见第 3.2 节），我们注意到，与使用哈希映射相比，采用预处理步骤并结合引导式状态空间探索可减少 10.2 吉字节的内存（约 55%）。此外，与采用随机变量排序和随机状态空间探索的前缀树方法相比，预处理步骤还节省了 287 MB。

预处理步骤的平均耗时稳定在约五分钟，与状态空间大小无关。我们认为，在分析内存限制是重要问题的模型时，这项时间投入是合理的，尤其是考虑到这些模型的状态空间构建可能需要数小时。

尽管与随机变量排序相比，预处理步骤带来的内存节省可能不如与哈希映射相比那么显著，但我们认为，这种方法能够让用户确信，其内存节省效果将显著优于最坏情况。对于超出现有内存限制、且连续时间马尔可夫链瞬态分析需要数小时乃至数天的状态空间，我们主张花少量时间进行预处理以找出高效的排序方式；对于内存效率至关重要的超大状态空间而言，这是值得的权衡。

6 结论

在本研究中，我们提出了一种新颖的应用：将前缀树数据结构用于改善大型、高并发连续时间马尔可夫链的内存消耗，其中包括化学反应网络和其他向量加法系统。与前沿工具所采用的哈希映射相比，前缀树在保持查找和插入时间为常数的同时，显著降低了内存消耗。理论复杂度分析表明，前缀树的表现可与哈希映射相媲美，且有望进一步节省内存。针对真实、复杂的化学反应网络模型进行的基准测试证实了这一分析，内存消耗通常改善了 50% 以上。我们还引入了一种基于有界模型检测的预处理步骤，该步骤提供了一种启发式方法，可进一步节省内存，初步评估也证实了这一点。采用前缀树并结合所提出的预处理方法，可显著提高连续时间马尔可夫链状态存储的内存效率。

未来的工作包括探索更多前缀树变量排序的启发式方法和技术，以及将前缀树数据结构集成到现有的概率模型检测工具中。

参考文献

1.Fluentverification/CaseStudies¯随机模型检验。（2024年6月），https://github.com/fluentverification/CaseStudies¯随机模型检验2. Ahmadi, M., Buecherl, L., Myers, C.J., Zhang, Z., Winstead, C., Zheng, H.：无限状态化学反应网络的罕见事件引导分析。载于：Hillston, J., Soudjani, S., Waga, M.（编）《系统定量评估与定时系统的形式化建模与分析》。第196–212页。施普林格自然瑞士，Cham（2024 10 1007 978 3 031 68416 6_12）。3 https://doi.org/./----. de Alfaro, L., Kwiatkowska, M., Norman, G., Parker, D., Segala, R.：使用多终端二叉决策图和克罗内克表示对概率过程进行符号模型检验。载于：Graf, S., Schwartzbach, M.（编）《系统构造与分析工具和算法》。第395–410页。施普林格，柏林，海德堡（2000 10 1007 3 540 46419 0_27）。https://doi.org/./---. 4 Andriushchenko, R., Bork, A., Budde, C.E., Češka, M., Grover, K., Hahn, E.M., Hartmanns, A., Israelsen, B., Jansen, N., Jeppson, J., Junges, S., Köhl, M.A., Könighofer, B., Křetínský, J., Meggendorfer, T., Parker, D., Pranger, S., Quatmann, T., Ruijters, E., Taylor, L., Volk, M., Weininger, M., Zhang, Z.：定量验证前沿工具：QComp 2023竞赛报告。载于：《TOOLympics 挑战赛 2023：形式化方法竞赛的最新进展、结果与成功经验》。第90–146页。施普林格出版社，柏林，海德堡（2024年11月）。https://doi.org/10.1007/978-3-031-67695-6_4，https://doi.org/10 1007 978 3 031 67695 6_4 ./----. Baier, C., Katoen, J.P.：《模型检验原理》。麻省5 理工学院出版社，剑桥，马萨诸塞州（2008）6. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.：理论可满足性。载于：《可满足性手册》，《人工智能与应用前沿》，第336卷，第1267–1329页。IOS出版社（2021）7. Biere, A.：有界模型检测。载于：《可满足性手册》，《人工智能与应用前沿》，第336卷，第739–764页。IOS出版社（2021）8. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.：不使用二元决策图的符号模型检验。载于：《系统构造与分析工具和算法：第5届国际会议，TACAS'99，作为软件理论与实践联合欧洲会议ETAPS'99的一部分，于1999年3月22日至28日在荷兰阿姆斯特丹举行，论文集5》。第193–207页。施普林格（1999）9. Bryant, R.E.：用于布尔函数操作的基于图的算法。电气电子工程师学会《计算机汇刊》**35**(8)，677–691（1986）10. Budde, C.E., D'Argenio, P.R., Hartmanns, A., Sedwards, S.：用于非确定性和罕见事件的统计模型检验器。载于：Beyer, D., Huisman, M.（编）《系统构造与分析工具和算法》。第340–358页。施普林格国际出版，Cham（2018）。https://doi.org/10.1007/978-3-319-89963-3 20 11¯. Buecherl, L., Roberts, R., Fontanarrosa, P., Thomas, P.J., Mante, J., Zhang, Z., Myers, C.J.：使用iBioSim和STAMINA对基因电路进行随机危害分析。ACS合成生物学**10**(10)，2532–2540（2021年10月）。https://doi. org/1000159 1210211./acssynbio.c. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.：符号模型检验：10^{20} 状态及其他。信息与计算**98**(2)，142–170（1992年6月）。https://doi.org/10.1016/0890-5401(92)90017-A

13. Češka, M., Křetínský, J.：化学反应网络的半定量抽象与分析。载于：Dillig, I., Tasiran, S.（编）《计算机辅助验证》，第11561卷，第475–496页。施普林格国际出版，Cham（2019）。https: //doi.org/ 10 1007 978 3 030 25540 4_28 . / - - - - . Cizelj, I., Ding, X.C.D., Lahijanian, M., Pinto, A., Belta, C.：敌对环境下的概率安全车辆控制。IFAC会议论文集44(1)，11803–11808（2011年1月）。https://doi.org/10.3182/20110828-6-IT-1002. 03248, 1474667016455126 15 https://www.sciencedirect.com/science/article/pii/S . Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.：《算法导论》第三版。MIT出版社（2009年7月）16. Cotner, M., Zhan, J., Zhang, Z.：大肠杆菌中白藜芦醇工程化生产的计算代谢模型。ACS合成生物学10(8)，1992–2001（2021）。https://doi.org/10.1021/acssynbio.1c00163, https://doi.org/10.1021/ acssynbio.1 34237218 17 00163, c pMID: . Daigle, B.J., Roh, M.K., Gillespie, D.T., Petzold, L.R.：生化系统中稀有事件概率的自动估计。《化学物理学杂志》134 3522769 18 4 044110 2011 10 1063 1 ( )，（2011年1月）。https://doi.org/ . / . . Daws, C., Kwiatkowska, M., Norman, G.：使用KRONOS和PRISM自动验证IEEE 1394根竞争协议。《软件工具与技术转移国际期刊》5(2)，221–236（2004年3月）。https://doi. org/10 5 19 1007 10009 003 0118 5, 10 1007 10009 003 0118 . /s - - - https://doi.org/ . /s - - - . Drawert, B., Lawson, M.J., Petzold, L., Khammash, M.：用于高效模拟随机反应扩散主方程的扩散有限状态投影算法。《化学物理学杂志》132(7)，074101（2010年2月）。https://doi.org/10 3310809 20 1063 1 2 2. / . . Dutertre, B.：Yices . . 载于：Biere, A., Bloem, R.（编）《计算机辅助验证》。第737–744页。施普林格国际出版，Cham（2014）。https://doi.org/10. 1007 978 3 319 08867 9_49 / - - - - . Fontanarrosa, P., Doosthosseini, H., Borujeni, A.E., Dorfan, Y., Voigt, C.A., Myers, C.：遗传电路动力学：冒险与毛刺分析。ACS合成生物学9(9)，2324–2338（2020）。https://doi.org/10.1021/acssynbio.0c00055, https://pubs. acs.org/doi/10 00055 22 1021 0 . /acssynbio. c . Fredkin, E.：字典树存储器。Commun. ACM 3(9)，490–499（1960年9月）。https://doi. org/10 367400 23 1145 367390 . / . . Friedenberg, M.D., Lita, A., Gilbert, M.R., Larion, M., Celiku, O.：癌症代谢的概率模型检验。《科学报告》12(1)，18870（2022）。https://doi.org/10.1038/s41598-022-21846-5, https://doi.org/10. 1038 5 24 41598 022 21846 /s - - - . Fujita, M., Fujisawa, H., Kawato, N.：基于二元决策图的布尔比较方法的评估与改进。载于： [1988] IEEE计算机辅助设计国际会议（ICCAD-89）技术论文摘要。第2 122450 25 5 1988 10 1109 1988 –（1988年11月）。https://doi.org/ . /ICCAD. . . Hartmanns, A., Hermanns, H.：《Modest 工具集：用于定量建模与验证的集成环境》。载于：Ábrahám, E., Havelund, K.（编）TACAS。《计算机科学讲义》，第8413卷，第593–598页。施普林格（2014）26. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.：概率模型检验器Storm。《软件工具与技术转移国际期刊》（2021年7月）。https://doi.org/10.1007/s10009-021-00633-z 27. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.：概率模型检验器Storm。《软件工具与技术转移国际期刊》24(4)，589–610（2022年8月）。https://doi.org/10.1007/s10009-021-00633-z

28. Hermanns, H.、Kwiatkowska, M.、Norman, G.、Parker, D.、Siegle, M.：关于使用多终端二叉决策图进行可性能分析和随机系统验证。《逻辑与代数编程杂志》56(1)，23–67（2003年5月）。https: //doi.org/10 8 29 1016 1567 8326 02 00066 . /S - ( ) - . Israelsen, B.、Taylor, L.、Zhang, Z.：化学反应网络中的高效轨迹生成与稀有事件分析。载于：C altais, G.、Schilling, C.（编）：软件模型检测。第13872卷，第83–102页。Springer Nature Switzerland，Cham（5月 2023 10 1007 978 3 031 32157 3 _5 ）。https://doi.org/ . / - - - - . 30 Jakob, W.、Rhinelander, J.、Moldovan, D.：pybind11——c ++11 2017 11 与Python之间的无缝互操 31 作（），https://github.com/pybind/pybind 。 Jensen, P.G.、Larsen, K.G.、Srba, J.：PTrie：通过前缀共享压缩和存储集合的数据结构。载于：Hung, D.V.、Kapur, D.（编）：计算理论方面——ICTAC 2017。第248–265页。施普林格国际出版，Cham（ 2017 10 1007 978 3 319 67729 3 _15 ）。https://doi.org/ . / - - - - . 32 Jeppson, J.、Volk, M.、Israelsen, B.、Roberts, R.、Williams, A.、Buecherl, L.、Myers, C.J.、Zheng, H.、Winstead, C.、Zhang, Z.：C++中的STAMINA：现代化的无限状态概率模型检测器。载于：Jansen, N.、Tribastone, M.（编）：系统定量评估。第101–109页。Springer Nature Switzerland，Cham（ 2023 10 1007 978 3 031 43835 6 _7 ）。https://doi.org/ . / - - - - . 33 Knuth, D.E.：《计算机程序设计艺术：排序与查找》第3卷。Addison-Wesley Professional（1998年4月）34. Kwiatkowska, M.、Norman, G.、Parker, D.：PRISM 4.0：概率实时系统验证。载于：Gopalakrishnan, G.、Qadeer, S.（编）：计算机辅助验证。第585–591页。《计算机科学讲义》，施普林格，柏林，海德堡（ 2011 10 1007 978 3 642 22110 1 _47 ）。https://doi.org/ . / - - - - . 35 Madsen, C.、 Myers, C.、 Roehner, N.、 Winstead, C.、 Zhang, Z.：合成生物学中的高效分析方法，第217–257页。施普林格纽约，纽约，NY（2015）。https://doi.org/10.1007/978-1-4939-1878-2 _11，https://doi.org/10.1007/ 978-1-4939-1878-2 1136 。Marques-Silva, J.、Lynce, I.、Malik, S.：冲突驱动子句学习SAT求解器。载于：《可满足性手册》，《人工智能与应用前沿》，第336卷，第133–182页。IOS出版社（2021）37. Miner, A.、Parker, D.：大型概率系统的符号表示与分析。载于：Baier, C.、Haverkort, B.R.、Hermanns, H.、Katoen, J.P.、Siegle, M.（编）：《随机系统验证：当前研究指南》，第296–338页。施普林格，柏林，海德堡（ 2004 10 1007 978 3 540 24611 4 _9 ）。38 https://doi.org/ . / - - - - . Myers, C.J.：《遗传电路工程》。《Chapman & Hall / CRC数学与计算生物学丛书》，CRC Press，第1版（2010）39. Nielsen, A.A.K.、Der, B.S.、Shin, J.、Vaidyanathan, P.、Paralanov, V.、Strychalski, E.A.、Ross, D.、Densmore, D.、Voigt, C.A.：遗传电路设计自动化。《科学》352(6281)，aac7341–aac7341（2016年4月）。https://doi.org/10.1126/science. aac7341 40. Raseen, M.、Thanushkodi, K.：使用子图复杂度进行ROBDD优化（2008年1月）41. Roh, M.K.、Gillespie, D.T.、Petzold, L.R.：加权随机模拟算法中用于重要性采样的状态依赖偏置方法。《化学物理学杂志》133 3493460 42 17 174106 2010 10 1063 1 ( )，（11月）。https://doi.org/ . / . . Sesic, A.、Dautovic, S.、Malbasa, V.：使用概率模型检测对具有双优先级请求队列的系统进行动态功率管理。《IEEE集成电路与系统计算机辅助设计汇刊》

27(2)，403–407（2008年2月）。https://doi.org/10.1109/TCAD.2007.911342, https: //ieeexplore.ieee.org/abstract/document/4378215 43. Taylor, L.、Israelsen, B.、Zhang, Z.：Cycle and Commute：化学反应网络的稀有事件概率验证。载于：形式化方法与计算机辅助设计。第284–293页。TU Wien Academic Press（2023年10月）。https://doi.org/ 10 34727 2023 978 3 85448 060 0_37 .//isbn.----。Volk, M.、Junges, S.、Katoen, J.P.：通过模型检测技术进行快速动态故障树分析。《IEEE工业信息学汇刊》**14**(1)，370–379（2018年1月）。https://doi.org/10.1109/TII.2017.2710316