arXiv:2301.01890v1 [cs.FL] 5 Jan 2023

Büchi 自动机的模块化混搭补全（技术报告）

Vojtěch Havlena[1],Ondřej Lengál[1],Yong Li[2,3],
Barbora Šmahlíková[1],和 Andrea Turrini[3,4]

[1] 布尔诺理工大学信息技术学院，捷克共和国布尔诺[2] 利物浦大学计算机科学系，英国[3] 中国科学院软件研究所计算机科学国家重点实验室，中国北京，中华人民共和国 [4] 智能软件研究所，中国广州，中华人民共和国

摘要。 非确定性 Büchi 自动机（Büchi 自动机）的补全是自动机理论中的一个重要问题，在形式化验证中有许多应用，例如程序的终止性分析、模型检测，或某些逻辑的判定过程。我们基于 Li 等人近期关于 Büchi 自动机确定化的研究思路，提出了一种用于 Büchi 自动机补全的新型模块化算法。该算法允许将多个 Büchi 自动机补全过程组合起来，每个过程分别处理 Büchi 自动机的一个强连通分量子集。通过这种方式，可以利用特定强连通分量的结构（例如其本质弱或具有确定性），并使用更高效的专用算法，而不受整个 Büchi 自动机结构的限制。我们给出了一个通用框架，可以将部分补全过程接入其中，并展示了该框架与多种算法的结合方式。一般而言，该框架可以生成具有 Emerson-Lei接受条件的补自动机，这种补自动机通常更加紧凑。借助该算法，我们为近期引入的电梯自动机类建立了一个新的上界 $O(4^n)$，比此前的上界有指数级改进。我们实现了该算法的原型，并针对大量基准测试进行了全面实验，结果表明，我们的框架与当前最先进水平相当，并可作为未来高效 Büchi 自动机补全和包含性检查算法的基础。

1 引言

非确定性 Büchi 自动机（BAs） [8]是一种优雅且概念上简单的框架，可用于对系统的无限行为及其预期满足的性质进行建模。Büchi 自动机广泛应用于许多重要的验证任务，例如程序的终止性分析 [27], 、模型检测 [51], ，或作为某些逻辑（如 S1S [8] 或 Sturmian 词上的一阶逻辑片段 [28]）判定过程的底层形式模型。许多这类应用都需要对 Büchi 自动机进行补全： 例如，在 Ultimate Automizer [27], 中进行程序终止性分析时，会使用补全来跟踪仍需证明其终止性的路径集合。另一方面，在模型检测[5] 和逻辑的判定

[5] 在这里，我们考虑针对以某种表达能力更强的逻辑给出的规范进行模型检测，例如 S1S、[8], QPTL [47], 或 HyperLTL， [12], 而不是 LTL， [41],其中否定较为简单。

过程中，补自动机通常用于实现否定和量词交替。补全往往是此处执行的最困难的自动机操作；其最坏情况下的状态复杂度为 $O((0.76n)^n)$ [2,45]（该界是紧的 [52]）。

在这些应用中，补全的效率通常决定着顶层应用的整体效率（甚至可行性）。例如，Ultimate Automizer 在国际软件验证竞赛（SV-COMP）的终止性类别中取得成功，很大程度上归功于一种高效的 BA 补全算法 [48]，该算法专门针对它经常遇到的具有特殊结构的 Büchi 自动机 [6,11]（截至本文撰写时，它已在 2017—2022 年间赢得 6 枚金牌，并在 2015 年和 2016 年赢得两枚银牌）。这里的特殊结构指所谓的 半确定性 *Büchi* 自动机 （SDBA），即由两部分组成的 Büchi 自动机：(i) 不含接受状态/转移的初始部分；(ii) 包含接受状态/转移的确定性部分，且这些状态/转移无法转移到第一部分。

使用所谓的 NCSB 算法系列中的一种算法 [5,6,11,25] 对 SDBA 进行补全，其最坏情况复杂度为 $O(4^n)$（而且通常在实践中也比通用 BA 补全过程表现好得多）。类似地，其他 Büchi 自动机子类也有高效的补全过程，例如：(i) 确定性 *Büchi* 自动机 (DBA) 可以补全为具有 $2n$ 个状态 [32] 的 Büchi 自动机（或具有 $n+1$ 个状态的 co-Büchi 自动机）；(ii) 内在弱 *Büchi* 自动机 （在这类自动机中，每个强连通分量 (SCC)里的环要么全部接受，要么全部拒绝）可以使用宫野-林算法 [39]补全为具有 $O(3^n)$ 个状态的 DBA。

长期以来，对于结构高度复杂但不属于上述任何类别的 Büchi 自动机，一直没有高效的补全算法，例如包含内在弱、确定性以及某些非确定性 SCC 的 Büchi 自动机。对于这类 Büchi 自动机，人们不得不使用复杂度为 $O((0.76n)^n)$（或更高）的通用补全算法。据我们所知，直到最近才有研究利用 Büchi 自动机的结构来获得更高效的补全算法：(i) Havlena等人 [26], 提出了电梯自动机 这一类别（由内在弱和确定性 SCC 任意组合而成的 Büchi 自动机），并为其给出了一个 $O(16^n)$ 算法。(ii) Li等人 [34], 提出了一种 Büchi 自动机确定化过程（将其确定化为确定性 Emerson-Lei 自动机）：该过程将输入的 Büchi 自动机分解为 SCC，并在同步构造中针对不同类型的 SCC（内在弱、确定性、一般）采用不同的确定化过程。

本文提出一种受 [34],启发的新型 BA 补全算法，利用了这样一个事实：从某种意义上说，补全比确定化更宽松。具体而言，我们提出一个框架，可在其中插入针对具有特定结构的强连通分量调优的不同部分补全过程。这些过程只处理给定的强连通分量，并在一定程度上独立 于 BA 的其余部分运行（从而减少潜在的状态空间爆炸）。随后，我们的顶层算法以同步 方式协调各个过程的运行（或者采用所谓的延后 策略，使其完全独立运行），从而得到一个接受条件可能更一般（通常为 Emerson-Lei 条件）的自动机，这有助于保持结果规模较小。如果这些过程满足给定的正确性要求，我们的框架就能保证其具体实例也正确。我们还提出了若干优化，例如使用轮询来减少非确定性，

针对某类部分算法，使用共享断点来减小规模和颜色数量，并将基于模拟的宏状态剪枝加以推广。

我们详细介绍了针对本质弱、确定性和初始确定性强连通分量的部分补全过程，并利用这些过程，为电梯自动机类获得了一个新的指数级更优上界 $O(4^n)$（即与其严格子类 SDBA 的上界相同）。此外，我们还针对一般强连通分量提供了两种部分过程，分别基于确定化（来自 [34]）和基于秩的构造。随后，我们通过原型实现表明，我们的算法能够与现有方法良好互补，并显著提升当前最先进水平。

2 预备知识

我们固定一个有限的非空字母表 Σ 和第一个无限序数 ω。一个（无限）词 w 是一个函数 $w\colon \omega \to \Sigma$，其中第 i 个符号记作 w_i。有时，我们将 w 表示为无限序列 $w = w_0 w_1$。.. 我们用 Σ^ω 表示 Σ 上所有无限词的集合；一个 ω-语言 是 Σ^ω 的子集。

Emerson-Lei 接受条件。给定集合 $\Gamma = \{0、\ldots\ldots、k-1\}$，其中包含 k 颜色（通常表示为 、 等）我们将*Emerson-Lei* 接受条件 $\mathbb{EL}(\Gamma)$定义为按照以下文法构造的公式集合：

$$\alpha ::= \mathsf{Inf}(c) \mid \mathsf{Fin}(c) \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha) \tag{1}$$

对于 $c \in \Gamma$。集合 $M \subseteq \Gamma$ 中的颜色与条件 α 之间的满足 关系 $\models$ 按如下方式归纳定义（对于 $c \in \Gamma$）：

$$M \models \mathsf{Fin}(c) \text{ iff } c \notin M, \qquad M \models \alpha_1 \vee \alpha_2 \text{ iff } M \models \alpha_1 \text{ or } M \models \alpha_2,$$
$$M \models \mathsf{Inf}(c) \text{ iff } c \in M, \qquad M \models \alpha_1 \wedge \alpha_2 \text{ iff } M \models \alpha_1 \text{ and } M \models \alpha_2.$$

Emerson-Lei自动机。一个定义在 Σ 上的（非确定性、基于转移的[6]）Emerson-Lei自动机（TELA）是一个元组 $\mathrm{A} = (Q, \delta, I, \Gamma, \mathrm{p}, \mathrm{Acc})$，其中 Q 是有限状态集，$\delta \subseteq Q \times \Sigma \times Q$ 是转移集[7]，$I \subseteq Q$ 是初始状态集，Γ 是颜色集，$\mathrm{p}: \delta \to 2^\Gamma$ 是转移的着色函数，而 $\mathrm{Acc} \in \mathbb{EL}(\Gamma)$。我们用 $p \xrightarrow{a} q$ 表示 $(p, a, q) \in \delta$，有时也将 δ 视为签名为 $\delta\colon Q \times \Sigma \to 2^Q$ 的函数。此外，我们将 δ 扩展到状态集$P \subseteq Q$，定义为 $\delta(P, a) = \bigcup_{p \in P} \delta(p, a)$。我们用 $\mathrm{A}\,[q]$ 表示 $q \in Q$，即自动机 $\mathrm{A}[q] = (Q, \delta, \{q\}, \Gamma, \mathrm{p}, \mathrm{Acc})$，也就是说，将 q 设为唯一初始状态后，从 A 得到的 TELA。若 $|I| \leq 1$ 且对每个 $q \in Q$ 和 $a \in \Sigma$，都有 $|\delta(q, a)| \leq 1$，则称 A 为确定性的。若 $\Gamma = \{$ $\}$且 $\mathrm{Acc} = \mathrm{Inf}($ $)$，则称 A 为 Büchi 自动机（BA），并记作 $\mathrm{A} = (Q, \delta, I, F)$，其中 F 是所有以 0 着色的转移的集合，即$F = \mathrm{p}^{-1}(\{$ $\})$。对于 BA，我们使用 $\delta_F(p, a) = \{q \in \delta(p, a) \mid \mathrm{p}(p \xrightarrow{a} q) = \{$ $\}\}$（并像对 δ 一样将此记法扩展到状态集）。若对于每个接受转移 $(p \xrightarrow{a} q) \in F$，$\mathrm{A}[q]$ 的可达片段都是确定性的，则称 BA $\mathrm{A} = (Q, \delta, I, F)$ 为半确定性 Büchi 自动机（SDBA）。

[6] 为避免在全文中反复处理接受状态和 接受转移而使行文冗长，我们只考虑基于转移的接受。将我们的方法扩展到基于状态/转移（或仅基于状态）的自动机十分简单。[7] 请注意，有些作者对 TELA 采用了更一般的定义，其中包含 $\delta \subseteq Q \times \Sigma \times 2^\Gamma \times Q$；由于我们只在输出中使用它们，因此采用较简单的定义即可。

A 中从 $q \in Q$ 出发、关于输入词 w 的一个运行是一个无限序列 $\rho\colon \omega \to Q$，它从 q 开始并遵循 δ，即 $\rho_0 = q$ 且 $\forall i \geq 0\colon \rho_i \xrightarrow{w_i} \rho_{i+1} \in \delta$。令 $\inf_\delta(\rho) \subseteq \delta$ 表示在 ρ 中无限次出现的转移集合，并令 $\inf_\Gamma(\rho) = \bigcup\{\mathrm{p}(x) \mid x \in \inf_\delta(\rho)\}$ 表示无限次出现的颜色集合。当且仅当 $\inf_\Gamma(\rho) \models \mathrm{Acc}$ 时，运行 ρ 在 A 中是接受的；A 的语言记作 L(A)，定义为存在一条从 I 中某个状态出发、在 A 中接受的运行的词 $w \in \Sigma^\omega$ 所构成的集合。

考虑一个 Büchi 自动机 $\mathrm{A} = (Q, \delta, I, F)$。对于状态集合 $S \subseteq Q$，我们用 A_S 表示 A 的副本，其中接受转移仅在 S 中的状态之间发生，即 Büchi 自动机 $\mathrm{A}_S = (Q, \delta, I, F \cap \delta|_S)$，其中 $\delta|_S = \{p \xrightarrow{a} q \in \delta \mid p, q \in S\}$。如果一个非空状态集合 $C \subseteq Q$ 中的每一对状态都能彼此到达，且 C 是这样一个集合中的极大集合，我们就称 $C \subseteq Q$ 为强连通分量（SCC）。如果 A 的某个 SCC 仅由一个不包含自环的状态构成，则该 SCC 是平凡的；否则，它是非平凡的。若 SCC C 包含至少一个接受转移，并且当且仅当满足以下条件之一时为本质弱：（i）C 中的每个环都包含来自 F 的转移；或（ii）C 中的任何环都不包含来自 F 的转移。若对于任意 $q \in C$，Büchi 自动机 $(C, \delta|_C, \{q\}, \emptyset)$ 都是确定性的，则称 SCC C 是确定性的。我们将本质弱分量称为 IWCs，将非本质弱的确定性接受分量称为 DACs（确定性接受分量），将其余接受分量称为 NACs（非确定性接受分量）。如果一个 Büchi 自动机 A 不含 NAC，则称其为电梯自动机。

我们假设 A 在其强连通分量之外不含接受转移（没有运行会沿着此类转移循环）。我们用 δ_{SCC} 表示将 δ 限制为不离开其强连通分量的转移，形式化地说，即 $\delta_{\mathrm{SCC}} = \{p \xrightarrow{a} q \in \delta \mid p$ 和 q 属于同一个强连通分量$\}$。A 的一个划分块 $P \subseteq Q$ 是其接受强连通分量的非空并集，而 A 的划分 是一个序列 P_1、……、P_n，其中各项两两不相交，且都是 A 的划分块，并且包含 A 的所有接受强连通分量。

BA A 的补自动机是一个 TELA，它接受 L(A) 的补语言 $\Sigma^\omega \setminus \mathrm{L}(\mathrm{A})$。在本文中，我们分别将补自动机的一个状态和一次运行称为宏状态和宏运行。

3 模块化补全算法

简而言之，我们的 BA 补全算法的核心思想是：首先根据 BA A 的性质将其分解为若干划分块，然后分别对每个划分块进行补全（可能使用不同的算法），并采用以下两种方式之一：使用同步式构造，在每一步中同步所有划分块的补全算法；或者使用延后式构造，先分别补全各个划分块，再通过自动机积构造合并部分结果。将 A 分解为划分块的方式可以很简单——即每个接受强连通分量对应一个划分块——也可以更复杂，例如采用这样的划分：一个划分块包含所有接受 IWCs，另一个包含所有 DAC，而每个 NAC 各自构成一个划分块。

这样一来，就可以避免对不受限制的 Büchi 自动机运行状态复杂度上界为 $\mathrm{O}((0.76n)^n)$ 的通用补全算法，而是针对每个划分块采用最合适的补全过程。这有三个主要优点：

1. 可以针对每个划分块选择不同的补全算法，以利用该块的性质。例如，对于包含

IWC 的划分块，可以使用基于断点（即所谓的 Miyano-Hayashi）构造的补全方法，宏状态数为 $O(3^n)$（参见第 4.1 节）；而对于仅包含 DAC 的划分块，可以使用基于针对 SDBA 对 NCSB 构造进行调整的算法，其状态复杂度为 $O(4^n)$（参见第 4.2 节）。对于 NAC，可以根据其性质选择秩算法 [10, 21, 24, 26, 31, 45] 或基于确定化的算法 [40, 42, 43]，等等（参见第 6 节）。

2. 不同的补全算法可以只关注各自的划分块，而无需考虑 Büchi 自动机的其他部分。这很有优势，例如，基于秩的算法可以利用这一限制，对所考虑的秩给出更紧的界（甚至比使用 [26] 中的细化方法还要紧）。

3. 由于采用了比 Büchi [44] 更通用的接受条件，得到的自动机可能更加紧凑——一般而言，它可以是任意 $\mathbb{EL}$ 个条件的合取（每个划分块对应一个条件），具体取决于补全过程的输出；通过混合使用不同条件，可以实现对生成的自动机进行更紧凑的编码。例如，使用 co-Büchi 条件补全确定性BA时，额外生成的状态数可以是常数；而使用 Büchi条件时，生成的状态数则是线性的（见第 5 节 5.1）。

随后，需要由一个顶层算法协调这些部分补全算法，以生成 A 的补自动机。

可以将我们的算法视为一种方法的优化：该方法对每个划分块 P 获得一个 Büchi 自动机 A_P，使用所选算法对 A_P 求补，并对以这种方式获得的所有 A_P 求交集（然而，这种方法无法获得我们在第 4.3 节给出的电梯自动机上界）。事实上，我们也实现了上述过程（称为延后式 方法，详见第 5.2 节），并将其与下文所述的主要过程（称为同步方法）进行了比较。

3.1 基本同步算法

本节介绍基本的同步 顶层算法。随后，我们将在第 4 节中给出该算法在电梯自动机上的具体实现，并为其补全提供一个新的上界；在第 5 节中讨论该算法的若干优化；在第 6 节中给出其针对非受限 BA 的推广。设 Büchi 自动机 $\mathrm{A} = (Q, \delta, I, F)$，不失一般性，假设 A 是完备的，即 $|I| > 0$，且所有状态 $q \in Q$ 对所有符号 $a \in \Sigma$ 都有一条出边转移。

同步算法使用针对 Büchi 自动机划分块的部分补全算法。每个此类算法 Alg 都配有一个结构条件 $\varphi_{\mathtt{Alg}}$，用于刻画该算法能够补全的划分块的性质。对于 Büchi 自动机 B，我们滥用记号，用 $\mathrm{B} \models \varphi$ 表示 B 满足条件 φ。如果 $\mathrm{A}_P \models \varphi_{\mathtt{Alg}}$，我们称 Alg 是一个针对划分块的部分补全算法 P。我们区分 Alg（能够补全给定类型划分块的一般算法）与 Alg_P（针对划分块 P 的具体实例）。我们要求每个实例 Alg_P 提供以下内容：

– $\mathrm{T}^{\mathrm{Alg}_P}$ — 算法生成的宏状态的类型；– 颜色$^{\mathrm{Alg}_P} = \{0, \ldots, k^{\mathrm{Alg}_P} - 1\}$ — 已使用颜色的集合；

– $\mathrm{Init}^{\mathtt{Alg}_P} \in 2^{\mathrm{T}^{\mathtt{Alg}_P}}$ — 初始宏状态的集合；– $\mathrm{Succ}^{\mathtt{Alg}_P} : (2^Q \times \mathrm{T}^{\mathtt{Alg}_P} \times \Sigma) \to 2^{\mathrm{T}^{\mathtt{Alg}_P} \times \mathtt{Colours}^{\mathtt{Alg}_P}}$ — 返回宏状态后继的函数，使得 $\mathrm{Succ}^{\mathtt{Alg}_P}(H, M, a) = \{(M_1, \alpha_1), \ldots, (M_k, \alpha_k)\}$，其中 H 是 A 中沿同一个词到达的所有状态的集合，M 是给定划分块对应的 $\mathtt{Alg}_P$ 宏状态，a 是输入符号，并且每个 (M_i, α_i) 都是一对（宏状态，颜色集合），其中 M_i 是 M 在 a 上相对于 H的后继，α_i 是从 M 到 M_i 的边上的颜色集合（H 有助于跟踪进入该划分块的新运行）；以及 – $\mathrm{Acc}^{\mathtt{Alg}_P} \in \mathbb{EL}$(颜色 $^{\mathtt{Alg}_P}$) — 接受条件。

设 P_1、……、P_n 是 A 的一个划分（不失一般性地，假设 $n > 0$），并设 $\mathrm{Alg}^1, \ldots\ldots$、$\mathrm{Alg}^n$ 是一个算法序列，满足 Alg^i 是 P_i 的部分补全算法。此外，定义如下辅助重新编号函数 λ：$\lambda(c, j) = c + \sum_{i=1}^{j-1} |$颜色$^{\mathtt{Alg}^i_{P_i}}|$，该函数用于使部分补全算法中的颜色和接受条件互不相交。我们还以自然方式将 λ 提升到颜色集合，并将其提升到 $\mathbb{EL}$ 条件，使得 $\lambda(\varphi, j)$ 具有与 φ 相同的结构，但其中每个原子 $\mathrm{Inf}(c)$ 都被替换为原子 $\mathrm{Inf}(\lambda(c, j))$（Fin 原子同理）。随后，同步补全算法生成 TELA $\mathrm{ModCompl}(\mathrm{Alg}^1_{P_1}, \ldots\ldots, \mathrm{Alg}^n_{P_n}, \mathrm{A}) = (Q\mathrm{C}, \delta^C, I\mathrm{C}, \Gamma^C, \mathrm{p}^C, \mathrm{Acc}^C)$，其各组成部分定义如下（我们使用 $[S_i]_{i=1}^n$ 来简写 $S_1 \times \cdots \times S_n$）：

– $Q^C = \prod_{i=1}^n 2^Q \times \mathrm{T}^{\mathtt{Alg}^i_{P_i}}$, – $I^C = \{I\} \times [$初始$^{\mathtt{Alg}^i_{P_i}}]_{i=1}^n$, – $\Gamma^C = \{0, \ldots, \lambda(k^{\mathtt{Alg}^n_{P_n}} - 1, n)\}$, – $\mathrm{Acc}^C = \bigwedge_{i=1}^n \lambda(\mathrm{Acc}^{\mathtt{Alg}^i_{P_i}}, i)$,[8]并– δ^C 且 p^C 定义如下：如果

$$((M'_1, \alpha_1), \ldots, (M'_n, \alpha_n)) \in [\mathrm{Succ}^{\mathtt{Alg}^i_{P_i}}(H, M_i, a)]_{i=1}^n,$$

那么 δ^C 包含转移 t：$(H, M_1, \ldots, M_n) \xrightarrow{a} (\delta(H, a), M'_1, \ldots, M'_n)$，其颜色由 $\mathrm{p}^C(t) = \bigcup\{\lambda(\alpha_i, i) \mid 1 \le i \le n\}$指定，并且 δ^C是满足条件的最小集合。

为了使 ModCompl 正确，部分补全算法需要满足某些性质，下面将对此进行讨论。

对于结构条件 φ 和 Büchi 自动机 $\mathrm{B} = (Q, \delta, I, F)$，我们定义 $\mathrm{B} \models_P \varphi$ 当且仅当 $\mathrm{B} \models \varphi$是 B 的一个划分块，且 B 在 P之外不包含任何接受转移。现在可以给出 Alg 的正确性条件。

定义 1. 如果对于每个$\mathcal{B}$，只要$\mathcal{B} \models_P \varphi_{\mathtt{Alg}}$，我们都有$\mathtt{Alg}$ 是正确的：$\mathcal{L}(\mathit{ModCompl}(\mathtt{Alg}_P, \mathcal{B})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{B})$。

于是，定理 1 证明了同步算法的正确性（前提是每个部分补全算法都正确）。

定理 1。设$\mathcal{A}$是一个 *Büchi* 自动机，P_1、……、P_n是$\mathcal{A}$的一个划分，且$\mathtt{Alg}^1$、、……、$\mathtt{Alg}^n$是一系列部分补全算法，使得$\mathtt{Alg}^i$ 对于 P_i是正确的。则有 $\mathcal{L}(\mathit{ModCompl}(\mathtt{Alg}^1_{P_1}$、……、$\mathtt{Alg}^n_{P_n}$、$\mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$。

[8]如果去掉 A 是完备的这一条件，我们还需要添加一个接受汇状态（表示 $H = \emptyset$ 的情况），并为其添加由新颜色　标记、覆盖所有符号的自环，同时将$\ldots \vee \mathrm{Inf}(\ \ )$ 加入 Acc_C。

4 电梯自动机的补全

在本节中，我们首先给出仅含接受弱确定自动机的划分块的部分算法（第 4.1 节），以及仅含 DAC 的划分块的部分算法（第 4.2 节）。随后，在第 4.3 节中，我们将证明，使用我们的算法，电梯布chi自动机的补自动机大小上界为 $O(4^n)$，这比 [26]中建立的已知上界 $O(16^n)$指数级更优。

4.1 本质弱接受组件的补全

首先，我们引入一个带有条件 $\varphi_{\mathtt{MH}}$ 的部分算法 MH，该条件规定划分块 P 中的所有强连通分量都是接受IWC。设 P 是 A 的一个划分块，且 $\mathrm{A}_P \models \varphi_{\mathtt{MH}}$。我们提出的方法采用 Miyano-Hayashi构造 [39]。由于接受 IWC 中的所有运行都是接受的，该构造的思路是接受这样的词：这些词上的所有运行最终都会离开 P。

因此，我们使用一个由状态集合组成的有序对 $(C,\ B)$ 作为宏状态，以补全 P。直观地说，我们用 C 表示 A 中处于 P 的所有运行集合（C 表示“检查”）。集合 $B \subseteq C$ 表示正在检查是否会在某个时刻离开 P 的运行（B 表示“断点”）。初始时，我们令 $C = I \cap P$，并将 P 中的所有运行也加入断点，即令 $B = C$。在读取一个 ω-词 w 的过程中，如果所有进入 P 的运行最终都会离开 P，即 B 无限多次变为空，那么 P 的补语言应包含 w（当 B 变空时，我们从当前的 C 中取出所有运行，并将其加入 B）。我们按照第 3.1 节的框架，将 MH_P 形式化为一个部分过程，如下所示：

- $\mathtt{T}^{\mathrm{MH}_P} = 2^P \times 2^P$, $\mathtt{Colours}^{\mathrm{MH}_P} = \{0\}$, $\mathtt{Init}^{\mathrm{MH}_P} = \{(I \cap P, I \cap P)\}$,
- $\mathtt{Acc}^{\mathrm{MH}_P} = \mathsf{Inf}(0)$, and $\mathrm{Succ}^{\mathrm{MH}_P}(H,(C,B),a) = \{((C',B'),\alpha)\}$ where
  - $C' = \delta(H,a) \cap P$,
  - $B' = \begin{cases} C' & \text{if } B^\star = \emptyset \text{ for } B^\star = \delta(B,a) \cap C', \\ B^\star & \text{otherwise, and} \end{cases}$
  - $\alpha = \begin{cases} \{0\} & \text{if } B^\star = \emptyset \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$

我们可以看出，检查 w 是否被 P 的补自动机接受，归约为检查 B 是否被无限多次清空。由于每次 B 变为空时，我们都会输出颜色 ，因此， w 在 P 中不被接受，当且仅当 无限多次出现。注意，转移函数 $\mathrm{Succ}^{\mathrm{MH}_P}$ 是确定性的，即恰有一个后继。

引理1。 部分算法 MH 是正确的。

4.2 确定性接受分量的补全

本节中，我们给出一个部分算法 CSB，并附加条件 $\varphi_{\mathtt{CSB}}$，规定划分块 P 由*DAC*组成。设 P 是 A 的一个划分块，且 $\mathrm{A}_P \models \varphi_{\mathtt{CSB}}$。我们的方法基于用于补全 SDBA 的 NCSB 算法族 [5, 6, 11, 25]，特别是 NCSB-MaxRank 构造 [25]。该算法利用了 DAC 中的运行具有确定性这一事实，即它们不会分支出新的运行。因此，可以检查某个运行是否为非接受运行：如果存在一个时间点，从此之后该运行不再经过任何接受转移。我们将这样的

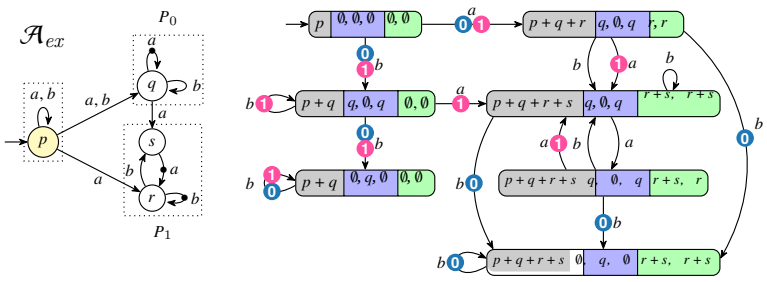


图1：左：Büchi 自动机 A_{ex} （点表示接受转移）。右：ModCompl(CSB$_{P_0}$, MH$_{P_1}$, A_{ex}) 的结果，其中 Acc：Inf( ) ∧Inf( )。状态表示为 $(H, C_0, S_0, B_0), (C_1, B_1))$；为避免使用过多花括号，集合表示为和。

不再经过任何接受转移的运行称为安全运行。那么，当且仅当 ω-词 w 在 P 中不被接受时，P 中 P 上所有关于 w 的运行要么 (i) 离开 P，要么 (ii) 最终变得安全。

为检查点 (i)，我们可以使用与 MH算法 中类似的技术，即使用一对 (C, B)。此外，为了能够检查点 (ii)，我们还使用集合 S，其中包含被认为是安全的运行，从而得到形如 (C, S, B)[9] 的宏状态。为确保所有运行都是确定性的，在计算 S 和 B 的后继时，我们将使用 δ_{SCC} 而不是 δ，因为在 P 中不同 DAC 之间可能存在非确定性跳转；我们不会遗漏 P 中的任何运行，因为如果某个运行在 P 的 DAC 之间移动，则可以将其视为离开 P 的运行，以及一个进入 P 的新运行。由于运行最终会停留在某个强连通分量中，这保证了该运行不会被遗漏。

我们在顶层框架中将 CSB$_P$ 形式化如下：

– $\text{T}^{\text{CSB}_P} = 2^P \times 2^P \times 2^P$, $\text{Init}^{\text{CSB}_P} = \{(I \cap P, \emptyset, I \cap P)\}$,– 颜色 $^{\text{CSB}_P} = \{ \}$, $\text{Acc}^{\text{CSB}_P}$ = nf( ), and– $\text{Succ}^{\text{CSB}_P}(H, (C, S, B), a)$ = U 满足以下条件

- 若 $\delta_F(S, a) \neq \emptyset$，则 $U = \emptyset$ (在 S 中运行的运行必须是安全)，
- 否则，U 包含 $((C', S', B'), c)$，其中
  - $S' = \delta_{\text{强连通分量}}(S, a) \cap P$, $C' = (\delta(H, a) \cap P) \setminus S'$,
  - $B' = \begin{cases} C' \text{ 若 } B_\star = \emptyset \text{ 对于 } B_\star = \delta_{\text{强连通分量}}(B, a), \text{ 否则，且若 } B_\star = \emptyset, \text{ 则} \\ B \end{cases}$ $c = \begin{cases} \{ \} \end{cases}$，

  否则。此外，在 $\delta_F(B, a) \cap \delta_{\text{强连通分量}}(B, a) = \emptyset$的情况下，$U$ 还包含 $((C'', S'', C''), \{ \})$，其中 $S'' = S' \cup B'$ 且 $C'' = C' \setminus S''$。

直观地，当 $\delta_F(B, a) \cap \delta_{\text{SCC}}(B, a) = \emptyset$ 时，我们会做出两种猜测：(i) B 中的运行最终都会变为安全运行（我们将它们移至 S），或者 (ii) 可能存在一些不安全运行（我们将它们保留在 B 中）。由于 B 中的运行是确定性的，因此 B 中被跟踪的运行数量不会增加。此外，如果 B 中的所有运行最终都是安全运行，那么我们可以确保在适当的时间点将它们全部移至 S，例如，所有运行都安全的最晚时间点，因为运行数量有限。

[9] 与 MH 不同，这里我们使用 $C \cup S$ 而不是 C 来跟踪 P中的所有运行。

如上所述，w 在 P 内不被接受，当且仅当 w 上的所有运行要么 (i) 离开 P，要么 (ii) 变为安全运行。在所提出的算法中，这对应于 (i) B 无限次变为空集，以及 (ii) $\delta_F(S,\ a)$ 从未遇到接受转移。于是，我们只需检查是否存在一个无限的宏状态序列 $\hat{}\rho = (C_0,\ S_0,\ B_0)$ ..... 它会无限次发出 。

引理 2. 部分算法 CSB 是正确的。

值得注意的是，当给定的划分块 P 包含 A 的所有 DAC 时，我们仍然可以使用上述构造，而 [25] 中的构造仅适用于 SDBA。例 *1.* 在图1中，我们给出了算法在 Büchi 自动机 A_{ex} 上运行的一个示例。该 Büchi 自动机包含三个强连通分量，其中一个（包含 p 的那个）不接受（因此无需出现在任何划分块中）。划分块 P_0 包含一个 DAC，因此我们可以使用算法 CSB；划分块 P_1 包含一个接受弱确定自动机，因此我们可以使用 MH。由此得到的 $\mathrm{ModCompl}(\mathrm{CSB}^{P_0}, \mathrm{MH}^{P_1}, A_{ex})$ 使用两种颜色： 来自 CSB， 来自 MH。接受条件为 Inf( ) ∧ Inf( )。□

4.3 电梯自动机补全的上界

现在我们给出使用针对电梯自动机的算法所生成补自动机的大小上界，这显著改进了此前已知的最佳上界 $O(16^n)$ [26]，将其降至 $O(4^n)$，与 SDBA 的上界相同。SDBA 是电梯自动机的真子类 [6]（我们注意到，基于确定化的算法无法达到这一上界，因为 SDBA 的确定化需要 $\Omega(n!)$ [17,37]）。

定理 2. 设 $\mathcal{A}$ 是一个有 n 个状态的电梯自动机。那么存在一个有 $O(4^n)$ 个状态的 *Büchi* 自动机，接受 $\mathcal{L}(\mathcal{A})$ 的补集。

证明（概要）。设 Q_W 为接受弱确定自动机中的所有状态，Q_D 为 DAC 中的所有状态，Q_N 为其余状态，即 $Q = Q_W \uplus Q_D \uplus Q_N$。我们构造两个划分块：$P_0 = Q_W$ 和 $P_1 = Q_D$，并分别采用 MH 和 CSB 作为部分算法，宏状态的形式为 $(H,\ (C_0,\ B_0),\ (C_1,\ S_1,\ B_1))$。对于每个状态 $q_N \in Q_N$，有两种选择：要么 $q_N \notin H$，要么 $q_N \in H$。对于每个状态 $q_W \in Q_W$，有三种选择：(i) $q_W \notin C_0$，(ii) $q_W \in C_0 \setminus B_0$，或 (iii) $q_W \in C_0 \cap B_0$。最后，对于每个 $q_D \in Q_D$，有四种选择：(i) $q_D \notin C_1 \cup S_1$，(ii) $q_D \in S_1$，(iii) $q_D \in C_1 \setminus B_1$，或 (iv) $q_D \in C_1 \cap B_1$。因此，宏状态总数为 $2 \cdot 2^{|Q_N|} \cdot 3^{|Q_W|} \cdot 4^{|Q_D|} \in O(4^n)$，其中初始因子 2 源于将两个颜色退化为一个颜色（实际上，使用我们在第 5.4 节介绍的共享断点优化即可避免这两种颜色）。□

5 模块化构造的优化

本节提出基本模块化算法的优化。在第 5.1 节中，我们给出一种部分算法，用 DAC 补全初始划分块。此外，在第 5.2 节中，我们提出延迟构造，以便对中间结果使用自动机约简。在第 5.3 节中，我们提出轮询算法，以缓解部分后继的笛卡尔积规模膨胀问题。在第 5.4 节中，我们针对基于断点构造的部分算法提出一项优化。最后，在第 5.5 节中，我们展示如何利用模拟来减小同步构造中宏状态的规模。

5.1 初始确定性划分块的补全

我们的第一个优化是面向一类包含 DAC 的划分块的优化算法 CoB。具体而言，条件 φ_{CoB} 指明划分块 P 在 A 中是确定性的，且只能通过确定性方式到达（即，移除冗余状态后，A_P 是确定性的）。在这种情况下，我们称 P 为初始确定性划分块。该算法基于将确定性 Büchi 自动机补全为 co-Büchi 自动机。

下面将算法 CoB_P 形式化：

- $\mathtt{T}^{\mathsf{CoB}_P} = P \cup \{\emptyset\}$，$\mathtt{Init}^{\mathsf{CoB}_P} = I \cap P$，$\mathtt{Colours}^{\mathsf{CoB}_P} = \{0\}$，$\mathtt{Acc}^{\mathsf{CoB}_P} = \mathsf{Fin}(0)$，
- $\mathtt{Succ}^{\mathsf{CoB}_P}(H, q, a) = \{(q', \alpha)\}$ where
  - $q' = \begin{cases} r & \text{if } \delta(H,a) \cap P = \{r\} \text{ and} \\ \emptyset & \text{otherwise,} \end{cases}$
  - $\alpha = \begin{cases} \{0\} & \text{if } q \xrightarrow{a} q' \in F \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$

直观地说，所有运行都以确定性方式到达 P，这意味着对于词 w，至多有一个运行能够到达 P。因此，如果存在一条在 w 上到达 P 的运行，对应于构造中的 $\delta(H,a) \cap P = \{r\}$，那么对于某个 $j \geq 0$，我们有 $|\delta(H, w_j) \cap P| = 1$。要检查 w 是否未被 P 接受，只需检查从 $r \in P$ 出发、沿着 w 的运行是否仅有限次经过接受转移。我们在附录 D 的图 5 中给出了一个对包含初始确定性划分块的 Büchi 自动机进行补全的示例。请注意，使用 Fin 条件有助于构造一个更简洁的自动机，其中只有两个状态（即使在这个简单示例中，使用 CSB 而非 CoB 也会产生一个有 4 个状态的 TELA）。

引理 3. 部分算法 CoB 是正确的。

5.2 延迟构造

第 3.1 节中的模块化同步构造利用了这样一个假设：在为 a 上的每个划分块同时构造后继时，如果某个部分宏状态 M_i 在 a 上没有后继，那么 $(H, M_1, \ldots, M_n)$ 宏状态在 δ^C 上也不会有后继。例如，在包含性测试中，这一点很有用，因为此时无需生成整个补自动机。另一方面，如果我们需要生成整个自动机，那么所提出的模块化构造有一个缺点：每种部分补全算法本身都可能生成大量无用状态。本节提出延迟构造，它独立地补全各个划分块（及其周围部分），随后将中间结果合并，以获得 A 的补自动机。延迟构造的主要优点是，可以应用自动机约简（例如，基于移除无用状态或使用模拟 [1,9,13,18]），从而缩减中间自动机的规模。

在延迟构造中，我们使用实现语言交集的自动机乘积运算（即，对于两个 TELA B_1 和 B_2，，构造乘积自动机 $\mathrm{B}_1 \cap \mathrm{B}_2$，使得 $\mathrm{L}(\mathrm{B}_1 \cap \mathrm{B}_2) = \mathrm{L}(\mathrm{B}_1) \cap \mathrm{L}(\mathrm{B}_2)$[10]）。此外，我们使用函数 Red 对输入的 TELA 执行某种保持语言的约简。接下来，延迟

[10] 另一种做法是避免构造乘积，生成线性规模的交替 TELA；但通常很难处理这类自动机，因此实践中并不采用。

对于具有划分 P_1、……、 P_n 和算法序列 $\mathrm{Alg}^1, \ldots\ldots$、$\mathrm{Alg}^n$ 的电梯自动机 A，其中 Alg^i 是针对 P_i 的部分补全算法，其构造定义如下：

$$\textsc{PostpCompl}(\mathtt{Alg}^1_{P_1}, \ldots, \mathtt{Alg}^n_{P_n}, \mathcal{A}) = \bigcap_{i=1}^{n} \mathtt{Red}\left(\textsc{ModCompl}(\mathtt{Alg}^i_{P_i}, \mathcal{A}_{P_i})\right). \quad (2)$$

附录 D 展示了将延迟构造应用于图 1 中 BA 的示例。随后，以下定理概括了该构造的正确性。

定理 3. 设 $\mathcal{A}$ 是一个 *Büchi* 自动机， P_1、……、 P_n 是 $\mathcal{A}$ 的一个划分，且 $\mathtt{Alg}^1$、、……、$\mathtt{Alg}^n$ 是一系列部分补全算法，并且 $\mathtt{Alg}^i$ 对 P_i 是正确的。则，$\mathcal{L}$(后补($\mathtt{Alg}^1_{P_1}$、……、$\mathtt{Alg}^n_{P_n}$、$\mathcal{A}$)) $= \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$。

5.3 轮询算法

第 3.1 节提出的基本同步方法可能会遭遇组合爆炸，因为宏状态的后继由各个部分宏状态的所有后继的笛卡尔积给出。为缓解这种爆炸，我们提出一种轮询 顶层算法。直观地说，轮询算法一次只在一个部分补全算法中主动跟踪运行（而其他算法保持被动）。该算法会定期更换活跃算法以避免饥饿（不过，离开活动状态的决定完全由部分补全算法指导）。对于生成多个后继的算法，这可以缓解后继数量的爆炸（例如，对于基于秩的算法，为了能够接受 $[10, 21, 24, 26, 31, 45]$，需要对状态递减的秩进行非确定性选择；这种选择只需在主动阶段进行，而在被动阶段，构造只需确保运行与给定的秩一致，这可以确定性地完成）。

轮询算法作用于部分补全轮询算法这一层面。部分算法的每个实例都会提供被动类型 来表示被动的部分宏状态，并提供主动类型 来表示当前活跃的部分宏状态。与第 3.1 节中的基本部分补全算法不同，后者只提供一个后继函数，而轮询部分算法会提供多个变体。具体而言，SuccPass 返回被动部分宏状态的（被动）后继，Lift 给出被动宏状态所有可能的主动对应状态，而 SuccAct 返回主动部分宏状态的后继。如果 SuccAct 返回一个被动类型的部分宏状态，轮询算法就会将下一个部分算法提升为活跃算法。例如，在 CSB 的轮询版本中，被动类型不包含断点，只检查安全运行是否始终安全，因此它是确定性的。由于篇幅所限，我们在附录 A 中给出轮询算法的形式化定义及更多细节。

5.4 共享断点

部分补全算法 CSB 和 MH（以及后来在附录 C 中定义的 RNK）使用一个断点来检查正在考察的运行是否接受

。作为一种优化，我们考虑合并多个算法的断点，并为所有受支持的算法只保留一个断点。这样，顶层算法只需管理一个断点，并且仅当这个唯一的断点变为空时才输出一种颜色。这可能会减少生成的宏状态数量，因为我们在多个算法之间同步断点采样。第二个好处是，这使我们能够生成更少的颜色（对于使用算法 CSB 和 MH 补全的电梯自动机，我们只需一种颜色）。

5.5 模拟剪枝

我们可以利用模拟关系（或其他兼容关系）剪枝宏状态，进一步优化构造。[11] 广义而言，模拟是一种关系 $\preccurlyeq \subseteq Q \times Q$，它蕴含状态间的语言包含关系，即 $\forall p, q \in Q\colon p \preccurlyeq q \implies \mathrm{L}(\mathrm{A}[p]) \subseteq \mathrm{L}(\mathrm{A}[q])$。直观地说，如果满足以下条件，我们的优化就允许从宏状态 M中移除状态 p：存在状态 q 也属于 M，且 (i) $p \preccurlyeq q$，(ii) p 无法从 q到达，以及 (iii) 在 Q 上的任意全序中，p 小于 q（这为模拟等价且互不可达的状态提供决胜规则）。移除 p 的原因是，其行为可以由 q完全模拟。在我们的构造中，粗略地说，我们可以在每个部分补全算法和顶层算法中，分别将对函数 $\delta(U, a)$ 和 $\delta_F(U, a)$ 的每次调用（针对状态集合 U），替换为 $\mathrm{pr}(\delta(U, a))$ 和 $\mathrm{pr}(\delta_F(U, a))$，其中 $\mathrm{pr}(S)$ 是通过从 S 中剪枝所有符合条件的状态得到的。详细内容见附录 B。

6 非电梯自动机的模块化补全

非电梯自动机 A 至少包含一个 NAC，也可能包含其他 IWC 或 DAC。为了以模块化方式对 A 进行补全，我们对其 DAC 和 IWC 应用第 4 节介绍的技术，而对其 NAC 则采用通用补全算法 Alg。理论上，将基于秩、 [31], 切片、 [29], Ramsey、 [47], 子集元组- [2],以及基于确定化- [43]的补全算法调整为针对单个划分块而非整个自动机运行，都可以作为 Alg 的具体实现。下面，我们将对其中两种算法作高层次介绍：基于秩和基于确定化的算法

.

基于秩的部分补全算法。即使输入的 Büchi 自动机仅包含 NAC，逐个独立处理每个 NAC 也有利于补全算法。例如，在基于秩的算法中， $[10, 21, 24, 26, 30, 31, 45]$, 对于给定的 ω-词 w，A 的所有运行是否都不接受，是由状态的秩决定的，这些秩由所谓的秩函数给出。秩函数是一个从 Q 到 ω的（部分）函数。基于秩的算法的核心思想如下：(i) 初始时，以非确定性方式为每个运行分配一个秩；(ii) 秩沿着运行只能递减；(iii) 每当运行经过一条接受转移时，秩必须为偶数；(iv)

[11] 这一优化可视为基于模拟的剪枝技术的一般化；例如，这类技术曾在 [25, 38] 中用于具体的确定化/补全过程。这里，我们将该技术推广到所有基于运行跟踪的过程。

当且仅当所有运行最终都陷入奇数秩时，补自动机才接受[12]。在标准的基于秩的过程中，(i) 中为状态分配初始秩的方式是一个函数$Q \rightharpoonup \{0,\dots,2n-1\}$，其中 $n = |Q|$。然而，借助我们的框架，我们可以显著限制划分块 P 中考虑的秩，使其仅为 $P \rightharpoonup \{0,\dots,2m-1\}$，其中$m = |P|$（此处，将由单个强连通分量构成的划分块作为划分块是合理的）。还可以利用例如 [24,26]中介绍的技术，进一步减少所考虑的秩。

为了在我们的框架中将基于秩的构造改造为部分补全算法 RNK，我们需要通过一个新的“盒状态”扩展秩函数，用以表示划分块之外的状态。随后，秩函数用来表示新进入划分块的运行的秩。盒扩展还要求改变转移，使其始终表示从外部可达的状态。我们在附录 C 中给出该构造的详细内容，其中包括来自 [24], 的 MaxRank 优化。

基于确定化的部分补全算法。在 [26,49] 中可以看到，基于确定化的补全在实践中也是 Alg 的一种良好实例，因此，我们也将标准的 Safra-Piterman 确定化 [40,42,43] 视为用于补全 NACs 的 Alg 选项。基于确定化的算法使用分层子集构造来组织关于一个 ω-词 w的所有运行。其思路是找出一个子集 $S \subseteq H$，其中包含沿着读取 w 的过程无限多次出现的可达状态，并且在 S的任意两次出现之间满足：(i) 第二次出现 S 中的每个状态都可以从第一次出现 S 中的某个状态到达；(ii) 第二次出现中的每个状态都可以从第一次出现中的某个状态经由一次接受转移到达。根据 König引理，A 在 w上必定存在一条接受运行。

该构造最初只维护一个集合 H：可达状态的集合。由于按定义，S 不一定需要是 H，每当有运行经过接受转移时，我们就为这些运行创建一个新的子集 C，并记录它来自哪个子集 C。这样，我们实际上将所有运行的当前状态组织成一棵树结构，并对每个树节点中的集合并行执行子集构造。如果我们发现某个树节点所标记的子集（例如 S'）等于其子节点中状态的并集，就知道集合 S' 满足上述条件，于是移除其所有子节点并触发一个良事件。如果此类良事件无限次发生，就意味着 S' 也会无限次发生。因此，在补全时，我们只需确保这些良事件只发生有限次。分别处理每个 NAC 也有利于基于确定化的方法，因为可达状态数量越少，可能出现的树就越少。遵循 [34], 的思路，将其调整用于部分补全算法的构造时，我们把来自其他划分块的所有新运行放入一个新创建的、没有父节点的节点中。这样，我们实际上为部分补全构造维护了一个树森林。我们将这种基于确定化的构造记为 DET；详情请参见 [34] 。

7 实验评估

为评估所提出的方法，我们在原型工具 Kofola 中实现了该方法。Kofola 使用 C++ 编写，并基于 Spot [16] 构建。我们将其与 COLA [34], 、Ranker [25]

[12] 由于这里侧重于直观理解，我们使用运行，而不是运行的有向无环图。

表1：我们实验的统计数据。**未解决**列按以下形式对未解决实例进行分类：超时：内存不足：其他失败。对于 VBS 的情况，我们仅提供未解决案例的数量。**状态**和**运行时间**列分别提供状态数和运行时间的均值：中位数。

tool	已解决	未解决	状	行时间
KOFOLA$_S$				
KOFOLA$_P$				
VBS$_+$	39,834	3	78	).05: 0.01
VBS$_-$	39,834	3	96	).05: 0.01

tool	已解决	未解决	状态	时间
COLA	39,814	21：	0：2	.17 :0.02
Ranker	38,837	61：939：0	45	3.31 :0.01
Seminator 3	,026	238: 573：0	2	.98 :0.03
SPOT	39,827	8：	0：2	).08 :0.02

（v. 2）、Seminator [5]（v. 2.0）和 Spot [15,16]（v. 2.10.6）进行了比较；这些工具在 BA 补全方面代表了当前最先进水平 [25,26,34]。受篇幅限制，我们仅给出框架的两种实例化方案的结果：Kofola$_S$ 和 Kofola$_P$。这两种方案均对 IWCs 使用 MH、对 DACs 使用 CSB、对 NACs 使用 DET。分区选择算法将所有 IWCs 合并为一个划分块，将所有 DACs 合并为一个划分块，并让所有 NACs 保持独立。我们启用了第 5.5 节中的基于模拟的剪枝，并关闭了第 5.3 节中的轮询（因为所选算法具有较强的确定性）。Kofola$_S$采用同步策略，而 Kofola$_P$ 采用延迟策略。我们还考察了虚拟最佳求解器（VBS），即一种虚拟工具，它会针对每个基准测试，从所有工具（VBS$_+$）或除 Kofola 两个版本以外的所有工具（VBS$_-$）中选择最佳求解器。我们在一台运行 Ubuntu 20.04.4 LTS 的台式机上进行了实验，该机器配备 16 GiB 内存和 Intel 3.6 GHz i 7-4790 CPU。为限制工具的运行并收集其统计信息，我们使用了 BenchExec [3]，并设置了 12 GiB 的内存上限和 10 分钟的超时时间；我们还使用 Spot 交叉验证不同工具生成的自动机是否等价。

作为数据集，我们使用了来自 automata-benchmarks 仓库的 39,837 个 Büchi 自动机 [33]（此前也曾被例如 [25,26,34] 使用）。该仓库中的 Büchi 自动机来自以下来源：(i) [49] 中使用的随机生成 Büchi 自动机（21,876 个），(ii) 根据文献中的 LTL 公式和随机生成的 LTL 公式得到的 Büchi 自动机 [5]（3,442 个），(iii) 从 Ultimate Automizer 得到的 Büchi 自动机 [11]（915 个），(iv) 从针对 Sturmian 词上一阶逻辑的求解器 Pecan 得到的 Büchi 自动机 [28]（13,216 个），(v) 从 S1S 求解器得到的 Büchi 自动机 [23]（370 个），以及 (vi) 通过 LTL 到 SDBA 转换得到的 Büchi 自动机 [46]（18 个）。在这些 Büchi 自动机中，23,850 个是确定性的，6,147 个是 SDBA（但不确定），4,105 个是电梯自动机（但不是 SDBA），其余 5,735 个属于其他类型。

表1概述了实验结果。尽管 Kofola 仍是原型，但它已经能够为很大一部分输入自动机构造补自动机，只有极少数情况只能由 Spot 或 COLA 成功构造补自动机。就平均状态数而言，Kofola$_S$ 在所有工具中**平均值最低**（Ranker 除外，不过它有 1,000 个未解决的案例）。此外，将 Kofola 纳入 VBS 后，**平均状态数显著降低**：从 96 降至 78！我们认为，这充分验证了我们方法的实用性。就运行时间而言，Kofola 的两个版本相当相似；Kofola 仅比 Spot 和 COLA 稍慢，但远快于 Ranker 和 Seminator（运行时间

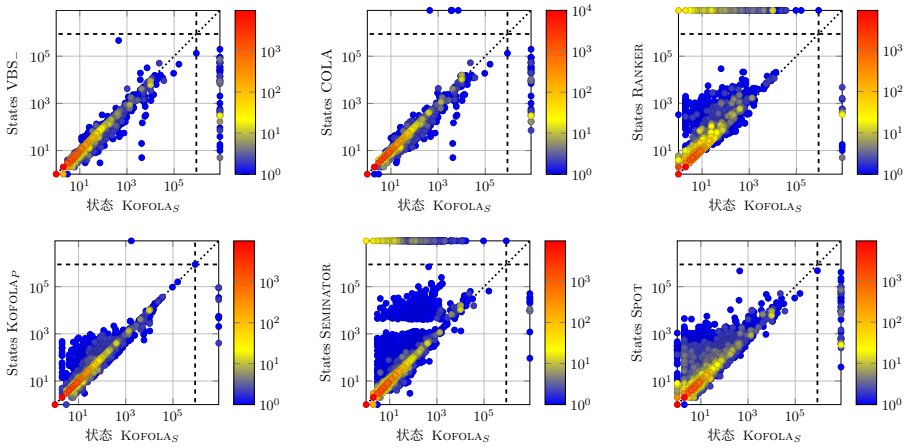
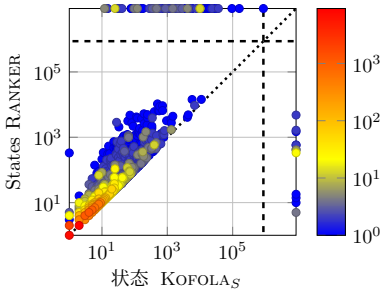


图 2：比较各工具生成的状态数的散点图。

图见附录）。作为原型，它在工程优化方面还有很大的提速空间。

在图 2 中，我们比较了 Kofola$_S$ 生成的状态数与其他工具生成的状态数；由于相应的图可以由 VBS$_-$ 的图推导得出，因此省略 VBS$_+$（由于 Ranker 和 Seminator 仅输出 Büchi 自动机，为了公平起见，我们比较所有工具转换为 Büchi 自动机后的输出大小）。在图中，每个标记所代表的基准测试数量由其颜色表示；位于对角线以上的标记表示 Kofola$_S$ 生成的自动机小于其他工具生成的自动机，而顶部边界上的标记表示其他工具运行失败而 Kofola$_S$ 成功；底部区域和右侧边界的情况则相反。虚线表示图中某个工具生成的最大状态数，坐标轴采用对数刻度。

从结果来看，Kofola$_S$ 明显优于那些不基于 SCC 分解的最先进工具（Ranker、Spot、Seminator）。其输出与 COLA 相当；COLA 同样使用 SCC 分解，也可视为我们框架的一种具体实现。这支持了我们的直觉：针对单个强连通分量进行处理有助于缩小最终自动机的规模，从而证实了我们模块化混搭式 Büchi 补



补全方法。最后，在右侧的图中，我们将针对电梯自动机的算法与 Ranker 中的算法进行比较（Ranker 是唯一一个为这一子类配备专用算法的其他工具）。我们的新算法显然优于 Ranker 中的算法。

8 相关工作

据我们所知，我们提供了首个通用框架，可在其中插入不同的 BA 补全算法，同时利用强连通分量的特定结构。我们将讨论我们的工作与现有文献之间的区别。

断点构造 [39] 原本是为仅含 IWCs 的 Büchi 自动机设计的补全方法，而我们的构造将其视为适用于 IWCs 的部分补全过程，并且不同之处在于需要处理来自其他划分块的输入状态。用于半确定性Büchi自动机的 NCSB 算法族 [5, 6, 11, 25] 在 DAC 之间存在非确定性跳转时无法奏效；不过，可以将其改造为我们框架中用于补全 DAC 的部分过程，参见第 4.2 节。在 [26], 中，一种基于消电梯化的过程被应用于电梯自动机，从而得到最大秩固定为 3 的 Büchi 自动机；对此，基于秩的构造可产生大小为 $O(16^n)$ 的结果。在我们的工作中，我们更充分地利用强连通分量的结构，从而得到指数级更优的上界 $O(4^n)$（与 SDBA 的上界相同）。补全无歧义 Büchi 自动机的上界 $O(4^n)$ 已在 [36],中得到确立；这项工作与我们的研究相互独立，但未来似乎可以纳入我们的框架。

关于一般 Büchi 自动机补全的研究成果十分丰富 [2, 5, 7, 8, 10,19–22, 24, 26, 29, 31, 40, 42, 43, 45, 47, 49, 50]；这些方法都针对输入 Büchi 自动机的整体图结构。我们的框架具有足够的通用性，可以将所有这些方法作为 NACs 的部分补全过程纳入其中。相反，我们的框架无法直接支持（至少在同步策略下）那些不 针对输入 Büchi 自动机结构运行的算法，例如 [35] 中基于学习的补全算法。作为我们灵感来源的近期确定化算法 [34]，也会分别处理强连通分量（实际上可以将其视为我们框架的一种实例化）。不过，我们当前的算法更加灵活，允许混合搭配各种构造、将强连通分量保持分离或合并为划分块，并且能够达到 $O(4^n)$ 的复杂度，而 [34] 只能达到 $O(n!)$（这是紧的，因为半确定性Büchi自动机确定化属于 $\Omega(n!)$ [17, 37]）。

关于工具 Spot [15, 16], ，不应将其视为单一的补全算法。相反，Spot 应被视为一个经过高度工程化的平台：它针对本质弱 Büchi 自动机采用断点构造，针对 SDBA 采用 NCSB [6, 11] ，针对一般 Büchi 自动机采用基于确定化的补全方法 [40, 42, 43] ，并在此过程中使用许多其他启发式方法。Seminator 使用半确定化 [4, 5, 14] 确保输入是 SDBA，然后使用 NCSB [6, 11] 计算补自动机。

9 结论与未来工作

我们提出了一个通用的 Büchi 自动机补全框架，可以利用强连通分量的特定结构，为其接入不同的部分补全过程。该框架不仅为电梯自动机带来了指数级更优的上界，也能很好地补充现有方法。实验结果（尤其是 VBS 的结果）表明，我们的框架显著改进了当前的补全算法组合。

我们认为，我们的框架是试验不同 BA 补全算法的理想测试平台，例如，原因有以下两点：(i) 可以开发一种高效的补全算法，但它只适用于相当受限的

Büchi 自动机子类（例如我们在第 5.1 节展示的用于初始确定性强连通分量的算法），而该框架可以利用它来补全所有包含此类子结构的 Büchi 自动机。(ii) 尝试改进通用补全算法时，可以专注于补全结构上较难处理的强连通分量（主要是非确定性接受强连通分量），而无需寻找一些启发式方法，以便在输入 Büchi 自动机中存在较简单的子结构时改进算法（例如 [26] 中所做的那样）。从框架的定义来看，它自然可以用于即时 Büchi 自动机语言包含测试，并利用现有的部分补全过程。最后，我们认为，该框架还为未来研究开辟了新方向：开发智能方法（可能基于机器学习），依据各强连通分量的特征，选择适用于每个强连通分量的部分补全过程。未来，我们希望纳入其他用于补全 NACs 的算法，并找出能够使用更高效算法的强连通分量性质（例如无歧义非确定性接受分量 [36]）。此外，似乎将 [24] 中的延迟优化推广到顶层算法，也有助于缩减状态空间。

致谢。感谢匿名审稿人提出的宝贵意见，帮助我们提升了论文质量；也感谢 Alexandre Duret-Lutz 分享用于绘制精美自动机的 TikZ 软件包。本研究得到中国科学院战略性先导科技专项（项目编号：XDA0320000）、国家自然科学基金委员会（项目编号：62102407 和 61836005）、中国科学院基础研究青年科学家项目（项目编号：YSBR-040）、工程与自然科学研究理事会（项目编号：EP/X021513/1）、捷克教育、青年与体育部 ERC.CZ 计划项目 LL1908、捷克科学基金会项目 GA23-07565S，以及 FIT BUT 内部项目 FIT-S-23-8151 的资助。本项目获得欧盟“地平线 2020”研究与创新计划玛丽·斯克沃多夫斯卡-居里资助协议（编号：101008233）的资助。

参考文献

1. Abdulla, P.A., Chen, Y., Holík, L., Vojnar, T.：用于约简的中介方法（关于交替布奇自动机的最小化）。理论计算机科学**552**，26–43（2014）。https://doi.org/ 10 003 2 1016 2014 08 003, 10 1016 2014 08 . /j.tcs. . . https://doi.org/ . /j.tcs. . . . Allred, J.D., Ultes-Nitsche, U.：一种简单且最优的 Büchi 自动机补全算法。载于：Dawar, A., Grädel, E.（编），第33届 ACM/IEEE 计算机科学逻辑年会（LICS 2018）论文集，英国牛津，2018年7月9日至12日。第46–55页。ACM（2018）。https://doi.org/10.1145/3209108.3209138, https://doi.org/10. 1145/3209108.3209138 3. Beyer, D., Löwe, S., Wendler, P.：可靠的基准测试：要求与解决方案。国际软件工具与技术转让杂志**21**（1），1–29（2019）。https://doi.org/10.1007/ s10009-017-0469-y, https://doi.org/10.1007/s10009-017-0469-y 4. Blahoudek, F., Duret-Lutz, A., Klokocka, M., Kretínský, M., Strejcek, J.：Seminator：一种用于 ω 自动机半确定化的工具。载于：Eiter, T., Sands, D.（编），LPAR-21，第21届逻辑编程、人工智能与推理国际会议，博茨瓦纳毛恩，2017年5月7日至12日。《计算系列 EPiC》，第46卷，第356–367页。EasyChair（2017）。https://doi.org/10.29007/k5nl, https://doi.org/10.29007/k5nl

5. Blahoudek, F., Duret-Lutz, A., Strejcek, J.：Seminator 2 可通过改进的半确定化补全广义 Büchi 自动机。载于：Lahiri, S.K., Wang, C.（编），计算机辅助验证——第32届国际会议 CAV 2020，洛杉矶，美国加利福尼亚州，2020年7月21日至24日，论文集，第二部分。《计算机科学讲义》，第12225卷，第15–27页。Springer（2020）。https://doi.org/10.1007/978-3-030-53291-8_2, https: //doi.org/10.1007/978-3-030-53291-8_2 6. Blahoudek, F., Heizmann, M., Schewe, S., Strejček, J., Tsai, M.：半确定性 Büchi 自动机的补全。载于：Chechik, M., Raskin, J.（编），系统构造与分析工具和算法——第22届国际会议 TACAS 2016，作为欧洲软件理论与实践联合会议 ETAPS 2016 的一部分举行，荷兰埃因霍温，2016年4月2日至8日，论文集。《计算机科学讲义》，第9636卷，第770–787页。Springer（2016）。https://doi.org/10.1007/978 49 7 3 662 49674 9 49, 10 1007 978 3 662 49674 9 - - - - _ https://doi.org/ . / - - - - _ . Breuers, S., Löding, C., Olschewski, J.：改进的基于 Ramsey 的 Büchi 补全。载于：Birkedal, L.（编），软件科学与计算结构基础——第15届国际会议 FOSSACS 2012，作为欧洲软件理论与实践联合会议 ETAPS 2012 的一部分举行，爱沙尼亚塔林，2012年3月24日至4月1日，论文集。《计算机科学讲义》，第7213卷，第150–164页。Springer（2012）。https://doi.org/10.1007/978-3-642-28729-9_10, https://doi.org/10.1007/978-3-642-28729-9_10 8. Büchi, J.R.：受限二阶算术中的一种判定方法。载于：Mac Lane, S., Siefkes, D.（编），《J. Richard Büchi 文集》，第425–435页。Springer（1990）。https://doi.org/10.1007/978-1-4613-8928-6_23, https://doi.org/10.1007/978-1-4613-8928-6_23 9. Bustan, D., Grumberg, O.：基于模拟的最小化。《ACM 计算逻辑汇刊》**4**（2），181–206（2003）10. Chen, Y., Havlena, V., Lengál, O.：基于秩的 Büchi 自动机补全中的模拟。载于：Lin, A.W.（编），编程语言与系统——第17届亚洲研讨会 APLAS 2019，印度尼西亚努沙杜瓦、巴厘岛，2019年12月1日至4日，论文集。《计算机科学讲义》，第11893卷，第447–467页。Springer（2019）。https://doi.org/10.1007/978-3-030-34175-6_23, https://doi.org/10.1007/978-3-030-34175-6_23 11. Chen, Y., Heizmann, M., Lengál, O., Li, Y., Tsai, M., Turrini, A., Zhang, L.：用于程序终止性检查的高级基于自动机的算法。载于：Foster, J.S., Grossman, D.（编），第39届 ACM SIGPLAN 编程语言设计与实现大会（PLDI 2018）论文集，美国宾夕法尼亚州费城，2018年6月18日至22日。第135–150页。ACM（2018）。https://doi.org/10.1145/3192366.3192405, https://doi.org/ 10.1145/3192366.3192405 12. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.：针对超属性的时序逻辑。载于：Abadi, M., Kremer, S.（编），安全与信任原理——第三届国际会议 POST 2014，作为欧洲软件理论与实践联合会议 ETAPS 2014 的一部分举行，法国格勒诺布尔，2014年4月13日至17日，论文集。《计算机科学讲义》，第8414卷，第265–284页。Springer（2014）。https://doi.org/10.1007/978-3-642-54792-8_15, https://doi.org/10.1007/978-3-642-54792-8_15 13. Clemente, L., Mayr, R.：非确定性自动机的高效约简及其在语言包含性检验中的应用。逻辑方法与计算机科学**15**（1）（2019）。https://doi.org/ 10 2019 14 23638 15 1 12 2019, 10 23638 15 1 12 . /LMCS- (:) https://doi.org/ . /LMCS- (:) . Courcoubetis, C., Yannakakis, M.：验证有限状态概率程序的时序性质。载于：第29届计算机科学基础年会，美国纽约州怀特普莱恩斯，1988年10月24日至26日。第338–345页。IEEE 计算机学会

(1988). https://doi.org/10.1109/SFCS.1988.21950, https://doi.org/10.1109/SFCS.1988.21950 15. Duret-Lutz, A., Lewkowicz, A., Fauchille, A., Michaud, T., Renault, E., Xu, L.: Spot 2.0——用于 LTL 和 ω-自动机操作的框架。载于：Artho, C., Legay, A., Peled, D.（编）《自动化验证与分析技术》第14届国际研讨会，ATVA 2016，日本千叶，2016年10月17日至20日，论文集。计算机科学讲义，第9938卷，第122–129页（2016）。https://doi.org/10.1007/978-3-319-46520-3_8, https://doi.org/10 8 16 1007 978 3 319 46520 3 . / - - - - _ . Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schlehuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: 从 Spot 2.0 到 Spot 2.10：有哪些新变化？载于：Shoham, S., Vizel, Y.（编）《计算机辅助验证》第34届国际会议，CAV 2022，以色列海法，2022年8月7日至10日，论文集，第二部分。计算机科学讲义，第13372卷，第174–187页。Springer（2022）。https://doi.org/10. 1007 9 17 978 3 031 13188 2 9, 10 1007 978 3 031 13188 2 / - - - - _ https://doi.org/ . / - - - - _ . Esparza, J., Kretínský, J., Raskin, J., Sickert, S.: 从 LTL 和极限确定性 Büchi 自动机到确定性奇偶自动机。载于：Legay, A., Margaria, T.（编）《系统构造与分析工具和算法》第23届国际会议，TACAS 2017，作为欧洲软件理论与实践联合会议 ETAPS 2017 的一部分举行，瑞典乌普萨拉，2017年4月22日至29日，论文集，第一部分。计算机科学讲义，第10205卷，第426–442页（2017）。https://doi.org/10.1007/ 978 25 18 3 662 54577 5 25, 10 1007 978 3 662 54577 5 - - - - _ https://doi.org/ . / - - - - _ . Etessami, K., Wilke, T., Schuller, R.A.: Büchi 自动机的公平模拟关系、奇偶博弈与状态空间约简。SIAM J. Comput. **34**(5), 1159–1175（2005）。https://doi.org/10.1137/S0097539703420675, https://doi.org/10.1137/S0097539703420675 19. Fogarty, S., Kupferman, O., Vardi, M.Y., Wilke, T.: Büchi 字自动机的轮廓树及其在确定化中的应用。Inf. Comput. **245**, 136–151（2015）。https://doi.org/ 10 021 20 1016 2014 12 021, 10 1016 2014 12 . /j.ic. . . https://doi.org/ . / j.ic. . . . Fogarty, S., Kupferman, O., Wilke, T., Vardi, M.Y.: 统一 Büchi 补全构造。逻辑方法与计算机科学 **9**(1)（2013）。https://doi.org/10.2168/ LMCS-9(1:13)2013, https://doi.org/10.2168/LMCS-9(1:13)2013 21. Friedgut, E., Kupferman, O., Vardi, M.Y.: 让 Büchi 补全更加紧致。Int. J. Found. Comput. Sci. **17**(4), 851–868（2006）。https://doi.org/10.1142/S0129054106004145, https://doi.org/10 0129054106004145 22 1142. /S . Gurumurthy, S., Kupferman, O., Somenzi, F., Vardi, M.Y.: 关于非确定性 Büchi 自动机的补全。载于：Geist, D., Tronci, E.（编）《正确硬件设计与验证方法》，第12届 IFIP WG 10.5 高级研究工作会议 CHARME 2003，意大利拉奎拉，2003年10月21日至24日，论文集。计算机科学讲义，第2860卷，第96–110页。Springer（2003）。https://doi.org/10.1007/ 978 10 23 3 540 39724 3 10, 10 1007 978 3 540 39724 3 1 - - - - _ https://doi.org/ . / - - - - _ . Havlena, V., Lengál, O., Smahlíková, B.: 判定 S S：深入兔子洞，穿过镜中世界。载于：Echihabi, K., Meyer, R.（编）《网络系统》第9届国际会议，NETYS 2021，线上活动，2021年5月19日至21日，论文集。计算机科学讲义，第12754卷，第215–222页。Springer（2021）。https://doi.org/10.1007/ 978-3-030-91014-3_15, https://doi.org/10.1007/978-3-030-91014-3_15 24. Havlena, V., Lengál, O.: 缩减（至）秩：高效的基于秩的 Büchi 自动机补全。载于：Haddad, S., Varacca, D.（编）第32届并发理论国际会议，CONCUR 2021，2021年8月24日至27日，线上会议。LIPIcs，第203卷，第2:1–2:19页。达格施图尔城堡—莱布尼茨信息学中心（2021）。https://doi.org/10.4230/LIPIcs.CONCUR.2021.2, https://doi.org/10.4230/LIPIcs.CONCUR.2021.2

25. Havlena, V., Lengál, O., Šmahlíková, B.: 使用 Ranker 补全 Büchi 自动机。载于：Shoham, S., Vizel, Y.（编）《计算机辅助验证》第34届国际会议，CAV 2022，以色列海法，2022年8月7日至10日，论文集，第二部分。计算机科学讲义，第13372卷，第188–201页。Springer（2022）。https://doi.org/10.1007/ 978-3-031-13188-2_10, https://doi.org/10.1007/978-3-031-13188-2_10 26. Havlena, V., Lengál, O., Šmahlíková, B.: 天空不是极限——Büchi 自动机补全中电梯自动机更紧的秩界。载于：Fisman, D., Rosu, G.（编）《系统构造与分析工具和算法》第28届国际会议，TACAS 2022，作为欧洲软件理论与实践联合会议 ETAPS 2022 的一部分举行，德国慕尼黑，2022年4月2日至7日，论文集，第二部分。计算机科学讲义，第13244卷，第118–136页。Springer（2022）。https://doi.org/10. 1007/978-3-030-99527-0_7, https://doi.org/10.1007/978-3-030-99527-0_7 27. Heizmann, M., Hoenicke, J., Podelski, A.: 通过学习终止程序进行终止性分析。载于：Biere, A., Bloem, R.（编）《计算机辅助验证》第26届国际会议，CAV 2014，作为维也纳逻辑夏季会议 VSL 2014 的一部分举行，奥地利维也纳，2014年7月18日至22日，论文集。计算机科学讲义，第8559卷，第797–813页。Springer（2014）。https://doi.org/10.1007/978-3-319-08867-9_53, https://doi.org/10.1007/978-3-319-08867-9_53 28. Hieronymi, P., Ma, D., Oei, R., Schaeffer, L., Schulz, C., Shallit, J.O.: Sturmian 词的可判定性。载于：Manea, F., Simpson, A.（编）第30届 EACSL计算机科学逻辑年会，CSL 2022，2022年2月14日至19日，德国哥廷根（线上会议）。LIPIcs，第216卷，第24:1–24:23页。达格施图尔城堡—莱布尼茨信息学中心（2022）。https://doi.org/10.4230/LIPIcs.CSL.2022.24, https://doi.org/10.4230/LIPIcs.CSL.2022.24 29. Kähler, D., Wilke, T.: 统一 Büchi 自动机的补全、消歧与确定化。载于：Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I.（编）《自动机、语言与编程》第35届国际研讨会，ICALP 2008，冰岛雷克雅雅克，2008年7月7日至11日，论文集，第一部分：专题 A：算法、自动机、复杂性与博弈。计算机科学讲义，第5125卷，第724–735页。Springer（2008）。https://doi.org/10.1007/978-3-540-70575-8_59, https://doi.org/10.1007/978-3-540-70575-8_59 30. Karmarkar, H., Chakraborty, S.: Büchi 补全中的最小奇数秩赋值。载于：Liu, Z., Ravn, A.P.（编）《自动化验证与分析技术》第7届国际研讨会，ATVA 2009，中国澳门，2009年10月14日至16日，论文集。计算机科学讲义，第5799卷，第228–243页。Springer（2009）。https://doi.org/10.1007/978-3-642-04761-9_18, https://doi.org/10.1007/978-3-642-04761-9_18 31. Kupferman, O., Vardi, M.Y.: 弱交替自动机并没有那么弱。ACM Trans. Com- put. Log. **2**(3), 408–429（2001）。https://doi.org/10.1145/377978.377993, https: //doi.org/10.1145/377978.377993 32. Kurshan, R.P.: 多项式时间内补全确定性 Büchi 自动机。J. Comput. Syst. Sci. **35**(1), 59–71（1987）。https://doi.org/10.1016/0022-0000(87)90036-5, https://doi.org/10.1016/0022-0000(87)90036-5 33. Lengál, O.: 自动机基准测试（2022），https://github.com/ondrik/ automata-benchmarks 34. Li, Y., Turrini, A., Feng, W., Vardi, M.Y., Zhang, L.: 基于 SCC 分解的 Büchi 自动机分治确定化。载于：Shoham, S., Vizel, Y.（编）《计算机辅助验证》第34届国际会议，CAV 2022，以色列海法，2022年8月7日至10日，论文集，第二部分。计算机科学讲义，第13372卷，第152–173页。

Springer（2022）。https://doi.org/10.1007/978-3-031-13188-2_8，https://doi.org/10 8 35 1007 978 3 031 13188 2 . / - - - - _ . Li, Y., Turrini, A., Zhang, L., Schewe, S.：学习补全 Büchi 自动机。载于：Dillig, I., Palsberg, J.（编）《验证、模型检测与抽象解释》—第19届国际会议 VMCAI 2018，2018年1月7日至9日，美国加利福尼亚州洛杉矶，论文集。计算机科学讲义，第10747卷，第313–335页。Springer（2018）。https://doi.org/10.1007/978-3-319-73721-8_15，https://doi.org/10.1007/978-3-319-73721-8_15 36. Li, Y., Vardi, M.Y., Zhang, L.：论无歧义性在布奇自动机补全中的作用。载于：Raskin, J., Bresolin, D.（编）《第11届博弈、自动机、逻辑与形式化验证国际研讨会论文集》，GandALF 2020，比利时布鲁塞尔，2020年9月21日至22日。EPTCS，第326卷，第182–198页（2020）。https://doi.org/10.4204/EPTCS. 326.12，https://doi.org/10.4204/EPTCS.326.12 37. Löding, C.：ω自动机变换的最优界。载于：Rangan, C.P., Raman, V., Ramanujam, R.（编）《软件技术与理论计算机科学基础》，第19届会议，印度金奈，1999年12月13日至15日，论文集。计算机科学讲义，第1738卷，第97–109页。Springer（1999）。https://doi.org/ 10.1007/3-540-46691-6_8，https://doi.org/10.1007/3-540-46691-6_8 38. Löding, C., Pirogov, A.：Büchi 自动机确定化的新型优化与启发式方法。载于：Chen, Y., Cheng, C., Esparza, J.（编）《自动化验证与分析技术》—第17届国际研讨会 ATVA 2019，台湾台北，2019年10月28日至31日，论文集。计算机科学讲义，第11781卷，第317–333页。Springer（2019）。https://doi.org/10.1007/978-3-030-31784-3_18，https://doi.org/10.1007/978-3-030-31784-3_18 39. Miyano, S., Hayashi, T.：ω字上的交替有限自动机。《理论计算机科学》**32**，321–330（1984）。https://doi.org/10.1016/0304-3975(84)90049-5，https:// doi.org/10 5 40 1016 0304 3975 84 90049 . / - ( ) - . Piterman, N.：从非确定性 Büchi 自动机和斯特里特自动机到确定性奇偶自动机。《逻辑方法与计算机科学》**3**(3)（2007）。https://doi.org/10.2168/LMCS-3(3: 5)2007，https://doi.org/10.2168/LMCS-3(3:5)2007 41. Pnueli, A.：程序的时序逻辑。载于：第18届计算机科学基础年会，美国罗得岛州普罗维登斯，1977年10月31日至11月1日。第46–57页。IEEE 计算机学会（1977）。https://doi.org/10.1109/SFCS.1977.32，https://doi.org/10 32 42 1109 1977 . /SFCS. . . Redziejowski, R.R.：利用导数改进确定性ω自动机的构造。《Fundam. Informaticae》**119**(3-4)，393–406（2012）。https://doi.org/10.3233/FI-2012-744，https://doi.org/10.3233/FI-2012-744 43. Safra, S.：论ω自动机的复杂度。载于：第29届计算机科学基础年会，美国纽约州怀特普莱恩斯，1988年10月24日至26日。第319–327页。IEEE 计算机学会（1988）。https://doi.org/10.1109/SFCS.1988.21948，https: //doi.org/10 21948 44 1109 1988 . /SFCS. . . Safra, S., Vardi, M.Y.：论ω自动机与时序逻辑（初步报告）。载于：Johnson, D.S.（编）《第21届 ACM 理论计算年会论文集》，1989年5月14日至17日，美国华盛顿州西雅图。第127–137页。ACM（1989）。https: //doi.org/10.1145/73007.73019，https://doi.org/10.1145/73007.73019 45. Schewe, S.：紧界 Büchi 补全。载于：Albers, S., Marion, J.（编）《第26届理论计算机科学专题国际研讨会》，STACS 2009，2009年2月26日至28日，德国弗赖堡，论文集。LIPIcs，第3卷，第661–672页。达格施图尔城堡—莱布尼茨信息学中心，德国（2009）。https://doi.org/10.4230/LIPIcs.STACS.2009.1854，https://doi.org/10.4230/LIPIcs.STACS.2009.1854

46. Sickert, S., Esparza, J., Jaax, S., Kretínský, J.：线性时序逻辑的极限确定性 Büchi 自动机。载于：Chaudhuri, S., Farzan, A.（编）《计算机辅助验证》—第28届国际会议 CAV 2016，加拿大安大略省多伦多，2016年7月17日至23日，论文集，第II部分。计算机科学讲义，第9780卷，第312–332页。Springer（2016）。https://doi.org/10.1007/978-3-319-41540-6_17，https://doi.org/10.1007/978-3-319-41540-6_17 47. Sistla, A.P., Vardi, M.Y., Wolper, P.：Büchi 自动机的补全问题及其在时序逻辑中的应用。《理论计算机科学》**49**，217–237（1987）。https://doi.org/10.1016/0304-3975(87)90008-9，https://doi.org/10.1016/0304-3975(87)90008-9 48. SV-COMP 社区：国际软件验证竞赛（2022），https://sv-comp.sosy-lab.org/ 49. Tsai, M., Fogarty, S., Vardi, M.Y., Tsay, Y.：Büchi 补全的现状。《逻辑方法与计算机科学》**10**(4)（2014）。https://doi.org/10.2168/LMCS-10(4:13)2014，https: //doi.org/10.2168/LMCS-10(4:13)2014 50. Vardi, M.Y., Wilke, T.：自动机：从逻辑到算法。载于：Flum, J., Grädel, E., Wilke, T.（编）《逻辑与自动机：历史与展望》。逻辑与博弈丛书，第2卷，第629–736页。阿姆斯特丹大学出版社（2008）51. Vardi, M.Y., Wolper, P.：自动程序验证的自动机理论方法（初步报告）。载于：《计算机科学逻辑研讨会论文集》（LICS '86），美国马萨诸塞州剑桥，1986年6月16日至18日。第332–344页。IEEE 计算机学会（1986）52. Yan, Q.：通过全自动机技术求ω自动机补全的下界。《逻辑方法与计算机科学》**4**(1)（2008）。https://doi.org/10.2168/LMCS-4(1: 5)2008，https://doi.org/10.2168/LMCS-4(1:5)2008

函数 $\Delta_C((H, M_1, \ldots, M_n, l), a)$**：**
  $\mathrm{M} := \emptyset$;
  设 AlgRR_i 为 $\mathrm{AlgRR}^i_{P_i}$;
  对每个 $1 \le i \le n$，$\mathrm{fSucc}_i := \mathrm{SuccPass}^{\mathrm{AlgRR}_i}$;
  $\mathrm{fSucc}_l := \mathrm{SuccAct}^{\mathrm{AlgRR}_\ell}$;
  对于 $((M'_1, c_1), \ldots, (M'_n, c_n)) \in [\text{后继函数}_i(H, M_i, a)]^n{}_{i=1}$ **do**
    $l' := l$;
    $L_i := \{M'_i\}$ 对每个 $1 \le i \le n$;
    如果 $M' \in \mathrm{PT}^{\mathrm{AlgRR}_\ell}_\ell$ **则**
      $l' := (l \mod n) + 1$;
      $L_{l'} := \mathrm{Lift}^{\mathrm{AlgRR}_{\ell'}}(M', l')$;
    $\mathcal{M} := \mathcal{M} \cup \{(\delta(H, a), (M''_1, c_1), \ldots\ldots, (M''_n, c_n), l') \mid M''_i \in L_i, \forall 1 \le i \le n\}$;
  返回 $\mathcal{M}$;

一种轮询算法

本节将详细介绍轮询算法。轮询算法在部分补全轮询算法AlgRR 的层面上运行。我们要求部分轮询算法 AlgRR_P 提供以下内容:

– $\mathrm{T}^{\mathrm{AlgRR}_P} = \mathrm{PT}^{\mathrm{AlgRR}_P} \cup \mathrm{AT}^{\mathrm{AlgRR}_P}$ — 算法产生的宏状态类型，由被动类型和主动类型组成；– 颜色$^{\mathrm{AlgRR}_P}$ — 算法产生的颜色集合；– $\mathrm{Init}^{\mathrm{AlgRR}_P}$: $2^{\mathrm{PT}^{\mathrm{AlgRR}_P}}$ — 返回初始宏状态集合的函数；– $\mathrm{SuccAct}^{\mathrm{AlgRR}_P} : (2^Q \times \mathrm{AT}^{\mathrm{AlgRR}_P} \times \Sigma) \to 2^{\mathrm{T}^{\mathrm{AlgRR}_P} \times \mathtt{Colours}^{\mathrm{AlgRR}_P}}$ — 在主动阶段返回后继的转移函数；– $\mathrm{SuccPass}^{\mathrm{AlgRR}_P} : (2^Q \times \mathrm{PT}^{\mathrm{AlgRR}_P} \times \Sigma) \to 2^{\mathrm{PT}^{\mathrm{AlgRR}_P} \times \mathtt{Colours}^{\mathrm{AlgRR}_P}}$ — 将被动状态提升为一组主动状态的函数；– $\mathrm{Lift}^{\mathrm{AlgRR}_P} : \mathrm{PT}^{\mathrm{AlgRR}_P} \to 2^{\mathrm{AT}^{\mathrm{AlgRR}_P}}$ — 用于从被动阶段切换到主动阶段的转移函数；– $\mathrm{Acc}^{\mathrm{AlgRR}_P} : \mathbb{EL}(\text{颜色}^{\mathrm{AlgRR}_P})$ — 返回接受条件公式的函数。

设 $\mathrm{A} = (Q, \delta, I, F)$ 是一个 Büchi 自动机，P_1、……、P_n 是一种划分，且 AlgRR^1、、……、AlgRR^n 是一系列算法，其中 AlgRR^i 是针对 P_i 的部分轮询补全算法。随后，补全算法会生成
$\mathrm{TELA\ ModComplRR}(\mathrm{AlgRR}^1_{P_1}$、……、$\mathrm{AlgRR}^n_{P_n}$、$\mathrm{A}) = (Q_C$、$\delta^C$、$I_C$、$\Gamma^C$、$\mathrm{p}^C$、$\mathrm{Acc}^C)$, 其各个组成部分定义如下:

– $Q^C = 2^Q \times [\mathrm{T}^{\mathrm{AlgRR}^i_{P_i}}]^n_{i=1} \times \{1, \ldots, n\}$,
– $I^C = \{I\} \times \mathtt{Lift}\big(\mathtt{Init}^{\mathrm{AlgRR}^1_{P_1}}\big) \times [\mathtt{Init}^{\mathrm{AlgRR}^i_{P_i}}]^n_{i=2} \times \{1\}$,
– $\Gamma^C = \{0, \ldots, \lambda(k^{\mathrm{AlgRR}^n_{P_n}} - 1, n)\}$,
– $\mathrm{Acc}^C = \bigwedge^n_{i=1} \lambda(\mathtt{Acc}^{\mathrm{AlgRR}^i_{P_i}}, i)$, and

– δ^C 和 p^C 的定义满足以下条件：

$$(H', (M'_1, c_1), \ldots, (M'_n, c_n), \ell') \in \Delta^C((H, M_1, \ldots, M_n, \ell), a),$$

然后 δ^C 包含转移 $t\colon (H,\ M_1,\ \ldots,\ M_n,\ l) \xrightarrow{a} (H',\ M'_1,\ \ldots,\ M'_n,\ l')$，其着色设为 $\mathrm{p}^C(t) = \bigcup\{c_i \mid 1 \le i \le n\}$，并且 δ^C 是最小的此类集合。我们还为每个 $q \in Q^C$ 设置 $\mathrm{p}^C(q) = \emptyset$。

接下来，我们关注轮询算法的正确性条件。对于一次运行 ρ，我们使用 $\rho_{i:j}$，其中 $i \le j$ 表示序列 $\rho_i,\ \rho_{i+1},\ \ldots,\ \rho_j$。令 AlgRR$_P$ 为部分轮询补全算法的一个实例。我们用 Union(AlgRR$_P$) 表示一个部分补全算法，它与 AlgRR$_P$ 具有相同的类型、颜色集合、初始宏状态集合和接受条件。后继函数定义如下：

$$\mathtt{Succ}^{\mathrm{Union}(\mathtt{AlgRR}_P)}(H, M, a) = \begin{cases} \mathtt{SuccAct}^{\mathtt{AlgRR}_P}(H, M, a) & \text{if } M \in \mathtt{AT}^{\mathtt{AlgRR}_P}, \\ S \cup \bigcup_{(M', c) \in S} \mathtt{Lift}^{\mathtt{AlgRR}_P}(H, M') & \text{if } M \in \mathtt{PT}^{\mathtt{AlgRR}_P} \end{cases}$$

其中 $S = \mathrm{SuccPass}^{\mathtt{AlgRR}_P}(H,\ M,\ a)$。

此外，对于满足 A $\models_P \varphi_{\mathtt{AlgRR}}$ 的 A，若在自动机 ModCompl(Union(AlgRR$_P$), A) 中满足以下条件，则称 AlgRR$_P$ 是一致的：

(C1) 对于每个 $w \in$ L(ModCompl(Union(AlgRR$_P$), A))，都存在一个接受运行 ρ，它在 w 上满足 $\rho_j \in \mathrm{AT}^{\mathtt{AlgRR}_P}$，并且对于无限多个 j，满足 $\rho_{j+1} \notin \mathrm{AT}^{\mathtt{AlgRR}_P}$；
(C2) 对于每个接受运行 ρ 和每个 $i \in \omega$，都存在接受运行 ρ' 和 ρ''，使得 $i > 1 \Rightarrow \rho_{i-1} \in \mathrm{PT}^{\mathtt{AlgRR}_P}$ 且 $\rho'_{1:i-1} = \rho''_{1:i-1} = \rho_{1:i-1}$，并且 $\rho'_i \in \mathrm{PT}^{\mathtt{AlgRR}_P}$ 且 $\rho''_i \in \mathrm{AT}^{\mathtt{AlgRR}_P}$；(C3) 设 ρ 是一个运行。如果对于无限多个 j，$\rho_j \in \mathrm{AT}^{\mathtt{AlgRR}_P}$ 且 $\rho_{j+1} \notin \mathrm{AT}^{\mathtt{AlgRR}_P}$，那么 ρ 是接受运行。

直观地说，第一个条件确保对于一个被接受的词，存在一条运行，其中被动类型与主动类型之间的切换发生无限多次。第二个条件则表示，切换到主动阶段可以推迟有限步，同时仍保留接受性。最后一个条件表示，如果一条运行在主动类型与被动类型之间遇到无限多次切换，那么这条运行就是接受的。部分轮询算法的正确性条件如下：

定义 2. 如果对于每个$\mathcal{A}$，只要$\mathcal{A} \models_P \varphi_{\mathtt{AlgRR}}$，就有AlgRR$_P$是一致的，且 $\mathcal{L}(\mathit{ModCompl}(\text{并集}(\mathtt{AlgRR}_P), \mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$，那么我们称AlgRR是正确的。

定理 4. 设$\mathcal{A}$是一个 *Büchi* 自动机，$P_1, \ldots, P_n$是$\mathcal{A}$的一个划分，且$\mathtt{AlgRR}^1, \ldots, \mathtt{AlgRR}^n$是一列部分轮询补全算法，并且$\mathtt{AlgRR}^i$对 P_i是正确的。那么，$\mathcal{L}(\mathit{ModComplRR}(\mathtt{AlgRR}^1_{P_1}, \ldots, \mathtt{AlgRR}^n_{P_n}, \mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$。

证明。 首先，我们提出以下辅助断言：

断言 1: 设 ρ 是 *ModCompl*(并集(AlgRR$_P$), $\mathcal{A}_P$) 中的一个接受运行，且存在 i_1、i_2、$i_1 \le i_2$ 和 $\forall i_1 \le l \le i_2$，使得 $\rho_\ell \in \mathrm{PT}^{\mathtt{AlgRR}_P}$ 且 $\rho_{i_2+1} \in \mathrm{AT}^{\mathtt{AlgRR}_P}$。那么，

对于每个 $k \geq i_1$，都存在一个接受运行 ρ'，使得 $\rho'_{k+1} \in \mathrm{AT}^{\mathrm{AlgRR}_P}$，$\forall i_1 \leq l \leq k$，有 $\rho'_\ell \in \mathrm{PT}^{\mathrm{AlgRR}_P}$，且 $\rho'_{1:\min(k,i_2)} = \rho_{1:\min(k,i_2)}$。

证明：由多次应用 (C2) 可直接得出。 ■

由于 P_1、……、P_n 是 A 的一个划分，因此我们有

$$\bigcup_{i=1}^{n} \mathcal{L}(\mathcal{A}_{P_i}) = \mathcal{L}(\mathcal{A}). \tag{3}$$

现在，我们继续证明该定理。

(⊆) 设 $\varrho = (M_1^1、\ldots\ldots、M_n^1,\ i_1)\ldots\ldots(M_1^k、\ldots\ldots、M_n^k,\ i_k)\ldots\ldots$ 是 $\mathrm{ModComplRR}(\mathrm{AlgRR}^1_{P_1}、\ldots\ldots、\mathrm{AlgRR}^n_{P_n},\ \mathrm{A})$ 在 w 上的一个接受运行。根据构造，我们有 $M_\ell^1\ldots\ldots$ 是 $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^\ell_{P_\ell}),\ \mathrm{A}_{P_\ell})$ 的一个接受运行。因此，$w \in \bigcap_{\ell=1}^n \mathrm{L}(\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^\ell_{P_\ell}),\ \mathrm{A}_{P_\ell})) = \bigcap_{\ell=1}^n \Sigma^\omega \setminus \mathrm{L}(\mathrm{A}_{P_\ell}) = \Sigma^\omega \setminus \mathrm{L}(\mathrm{A})$。后者由 AlgRR_P 的正确性条件以及 (3) 得出。

(⊇) 考虑一个词 $w \in \Sigma^\omega \setminus \mathrm{L}(\mathrm{A})$。我们通过归纳法证明，在自动机 $\mathrm{ModComplRR}(\mathrm{AlgRR}^1_{P_1},\ldots,\mathrm{AlgRR}^n_{P_n},\mathrm{A})$ 中也存在一个接受运行 $\varrho = (M_1^1,\ldots,M_n^1,i_1)\ldots(M_1^k,\ldots,M_n^k,i_k)\ldots$，而且对于每个 j，$M_\ell^1\ldots M^j l$，它都是 w $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^\ell_{P_\ell}),\mathrm{A}_{P_\ell})$ 中某个接受运行的前缀。对于每个 l，在下文中，当我们使用“（接受）运行”时，默认指在 w 上的运行。

– 基本情形：由于 ρ_1 满足条件 (C2)，根据 (C2) 存在一个接受运行 ρ'，使得 $\rho'(1) \in \mathrm{AT}^{\mathrm{AlgRR}^1_{P_{i_1}}}$。此外，还存在运行 ρ'_2、……、ρ'_n，使得 $\rho'_i(1) \in \mathrm{PT}^{\mathrm{AlgRR}^1_{P_{i_1}}}$。因此，我们令 $\varrho_1 = ((\rho'_1)_1,\ldots\ldots,\ (\rho'_n)_1,\ 1)$。– 归纳情形：设 $(M_1^1,\ldots,\ M_n^1,i_1)\ldots(M_1^k,\ldots,\ M_n^k,i_k)$ 是 ϱ 的前 k 个宏状态序列。我们证明 ϱ 还存在第 $(k+1)$ 个宏状态。由于 $M_{i_k}^k \in \mathrm{AT}^{\mathrm{AlgRR}^k_{P_{i_k}}}$，并且 $M_{i_k}^1\ldots\ldots M_{i_k}^k$ 是自动机 $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^{i_k}_{P_{i_k}}),\ \mathrm{A}^{P_{i_k}})$ 中某个接受运行 ρ' 的前缀。现在假设 $(\rho'_{i_k})_{k+1} \in \mathrm{AT}^{\mathrm{AlgRR}^{i_k}_{P_{i_k}}}$。由于对于每个 $l \neq i_k$，$p_\ell = M_\ell^1\ldots\ldots M_\ell^k$ 都是自动机 $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^\ell_{P_\ell}),\ \mathrm{A}_{P_\ell})$ 中某个接受运行的前缀，此外 $M_\ell^k \in \mathrm{PT}^{\mathrm{AlgRR}^\ell_{P_\ell}}$。因此，根据 (C2)，存在来自 (C2) 的接受运行 ρ'_ℓ，满足 $(\rho'_\ell)_{k+1}$ 属于相应的被动类型。因此，我们将 $((\rho'_1)_{k+1},\ \ldots\ldots,\ (\rho'_n)_{k+1},i_k)$ 设为 ϱ 的第 $(k+1)$ 个宏状态。现在假设 $(\rho'_{i_k})_{k+1} \notin \mathrm{AT}^{\mathrm{AlgRR}^k_{P_{i_n}}}$。令 $j = (i_k \bmod n) + 1$。$p_j = M_j^1\ldots\ldots M_j^k$ 是自动机 $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^j_{P_j}),\ \mathrm{A}^{P_j})$ 中某个接受运行的前缀；由断言 1 可知，还存在一个接受运行 ρ'_j，它延伸自 p_j，且满足 $(\rho'_j)_{k+1} \in \mathrm{AT}^{\mathrm{AlgRR}^j_{C_j}}$。此外，由于对于每个 $l \notin \{j, i_k\}$，$p_\ell = M_\ell^1\ldots\ldots M_\ell^k$ 都是自动机 $\mathrm{ModCompl}(\mathrm{Union}(\mathrm{AlgRR}^\ell_{P_\ell}),\ \mathrm{A}_{P_\ell})$ 中某个接受运行的前缀，此外 $M_\ell^k \in \mathrm{PT}^{\mathrm{AlgRR}^\ell_{P_\ell}}$。因此，根据 (C2)，存在来自 (C2) 的接受运行 ρ'_ℓ，满足 $(\rho'_\ell)_{k+1}$ 属于相应的被动类型。我们将 $((\rho'_1)_{k+1},\ \ldots\ldots,\ (\rho'_n)_{k+1},\ j)$ 设为 ϱ 的第 $(k+1)$ 个宏状态。

很容易证明，ϱ 是 ModComplRR(AlgRR$^1_{P_1}$，……，AlgRR$^n_{P_n}$，A) 中的一个运行。由于每个分量在被动阶段和主动阶段之间无限次切换，根据 (C3)，每个部分运行都是相应 ModCompl(Union(Alg$^i_{P_i}$)) 中的接受运行，因此 ϱ 也是接受运行。□

A.1 本质弱组件的补全

在本节中，我们定义算法 AlgRR $=$MHRR，并给出一个条件 φ_{MHRR}，用于规定一个划分 P 是本质弱且可接受的。

我们将实例 MHRR$_P$ 形式化如下：

- $\mathtt{T}^{\mathtt{MHRR}_P} = \mathtt{PT}^{\mathtt{MHRR}_P} \cup \mathtt{AT}^{\mathtt{MHRR}_P}$ where $\mathtt{PT}^{\mathtt{MHRR}_P} = 2^P$ and $\mathtt{AT}^{\mathtt{MHRR}_P} = 2^P \times 2^P$,
- $\mathtt{Colours}^{\mathtt{MHRR}_P} = \{0\}$,
- $\mathtt{Init}^{\mathtt{MHRR}_P} = \{I \cap P\}$,
- $\mathtt{SuccAct}^{\mathtt{MHRR}_P}(H,(C,B),a) = \begin{cases} \{(\delta(H,a)\cap P, \{0\})\} & \text{if } B' = \emptyset \text{ for} \\ & B' = \delta(B,a)\cap P \\ \{((\delta(H,a)\cap P, B'),\emptyset)\} & \text{otherwise,} \end{cases}$
- $\mathtt{SuccPass}^{\mathtt{MHRR}_P}(H,C,a) = \{(\delta(H,a)\cap P,\emptyset)\}$,
- $\mathtt{Lift}^{\mathtt{MHRR}_P}(C) = \{(C,C)\}$,
- $\mathtt{Acc}^{\mathtt{MH}_P} = \mathsf{Inf}(0)$.

引理 4. 部分轮询算法 MHRR是正确的。

证明。（概要）考虑一个 Büchi 自动机 A，使得 A $\models_P \varphi_{\text{MHRR}}$，以及一个词$w \in$ L(ModCompl(Union(MHRR$_P$), A)。一个接受运行 φ 必定包含无穷多个标记为的接受转移。这些转移从活跃状态指向被动状态。因此，我们可以在清空 B' 后切换到被动状态，然后至少经过一步后再切换回活跃状态。这满足条件 (C1)。切换回活跃状态前经过多少步并不重要，但步数必须有限。这满足条件 (C2)。条件 (C3)也得到满足，因为我们只会在 B' 为空时才从活跃阶段切换到被动阶段，也就是说，只有在执行接受转移时才会切换。□

A.2 确定性接受分量的补全

下面，我们定义算法 Alg $=$CSBRR，并加入条件 φ_{CSBRR}，用于规定划分 P 在强连通分量内确定。

我们将实例 CSBRR$_P$ 形式化如下：

– $\mathtt{T}^{\mathtt{CSBRR}_P} = \mathtt{PT}^{\mathtt{CSBRR}_P} \cup \mathtt{AT}^{\mathtt{CSBRR}_P}$，其中 $\mathtt{PT}^{\mathtt{CSBRR}_P} = 2^P \times 2^P$ 和 $\mathtt{AT}^{\mathtt{CSBRR}_P} = 2^P \times 2^P \times 2^P$，– 颜色$^{\mathtt{CSB}_P} = \{\ \}$，– 初始状态 $\mathtt{Init}^{\mathtt{CSBRR}_P} = \{(I \cap P, \emptyset)\}$– $\mathtt{SuccAct}^{\mathtt{CSBRR}_P}(H,(C,S,B),a) = U$ 满足

- 若 $\delta_F(S,a) \cap P \neq \emptyset$，则 $U = \emptyset$,
- 否则 U 包含有序对 (V,c)，其中

∗ $V = \begin{cases} (C', S'), & \text{如果 } B_\star = \emptyset \text{ 对于 } B_\star = \delta_{\text{强连通分量}}(B, a) \\ (C', S', B_\star) & \text{否则} \end{cases}$

∗ $S' = \delta_{\text{强连通分量}}(S, a) \cap P$,

∗ $C' = (\delta(H, a) \cap P) \setminus S'$,

∗ $c = \begin{cases} \quad & \text{如果 } B_\star = \emptyset \text{ 且 } \emptyset \text{ 否则。并且如果 } \delta_F(B, a) \cap \delta_{\text{强连通分量}}(B, a) \\ = \emptyset, & \text{那么 } U \text{ 还包含这一对 } ((C'', S''), \{\quad\}), \text{ 其中} \end{cases}$

∗ $S'' = S' \cup B'$ 并且

∗ $C'' = C' \setminus S''$.– SuccPass$^{\text{CSBRR}_P}(H, (C, S), a) = U$ 使得

- 如果 $\delta_F(S, a) \cap P \neq \emptyset$, 那么 $U = \emptyset$,
- 否则 $U = \{((C', S'), \emptyset)\}$, 其中
  - ∗ $S' = \delta(S, a) \cap P$, 并且
  - ∗ $C' = (\delta(C, a) \cap P) \setminus S'$, –

Lift$^{\text{CSBRR}_P}(C, S) = \{(C, S, C)\}$– Acc$^{\text{CSBRR}_P} = \text{Inf}(\quad)$。

引理 5。 部分轮询算法 CSBRR是正确的。

证明。（概要）考虑一个 BA A，使得 $A \models_P \varphi_{\text{CSBRR}}$ 以及一个词$w \in L(\text{ModCompl}(\text{Union}(\text{CSBRR}_P, A))$。一个接受运行 φ 必须包含无穷多个标记为 的接受转移。这些转移从活动状态转到被动状态。因此，我们可以在清空 $B^\star$ 后切换到被动状态，然后至少执行一步后再切换回活动状态。这满足条件 (C1)。在切换回活动状态之前执行多少步并不重要，但步数必须是有限的。这满足条件 (C2)。条件 (C3) 也得到满足，因为我们仅在 $B^\star$ 为空时从活跃阶段切换到被动阶段，即仅在执行接受转移时才进行切换。□

B 基于模拟的优化

以下与正文一样，固定一个 Büchi 自动机 $A = (Q, \delta, I, F)$，其极大强连通分量集合为 $\{C_1, \ldots, C_n\}$。本节将推广 [25] 中介绍的结果。我们用 $p \rightsquigarrow q$ 表示 q 可从 p 到达。设 $w \in \Sigma^\omega$ 是一个词。设 Π, Π' 是 w 上的轨迹集合。若 $\exists \pi \in \Pi: \pi$ 在 A 中被接受，当且仅当 $\exists \pi' \in \Pi': \pi'$ 在 A 中被接受，则称 Π 与 Π' 接受等价，记作 $\Pi \sim \Pi'$。

这里回顾第 5.5 节中平局决胜函数和剪枝关系的定义。令 $\#: Q \to \omega$ 为满足以下条件的函数，对每个$p, q \in Q$均成立：(i) $\#(p) = \#(q)$ 当且仅当 $p, q \in C_i$，且 (ii) 若 $p \rightsquigarrow q$，则 $\#(p) \leq \#(q)$。令$\sqsubseteq \subseteq Q \times Q$ 为 A 的状态上的关系，定义如下：$p \sqsubseteq q$ 当且仅当 (i) $p \preccurlyeq q$且 (ii) $\#(p) < \#(q)$。于是，对每个$S \subseteq Q$，剪枝函数 pr：$2^Q \to 2^Q$ 定义为$pr(S) = S'$，其中 $S' \subseteq S$是满足以下条件的最小集合：$\forall q \in S \exists q' \in S'$: $q \sqsubseteq q'$。直观地说，pr 会移除模拟关系下较小的状态。

令 $\rho = S_1 S_2 \ldots\ldots$为状态集序列， w 为一个词。我们将 Π_ρ 定义为与这些状态集相匹配的、在 w 上的轨迹集合。形式化地说， $\Pi_\rho = \{\pi \mid \pi$ 在 w上， $\pi_i \in S_i$ 对每个 i均成立$\}$。此外，对于状态集合 B，我们用 ρ_w^B 表示序列 $S_1 S_2 \ldots\ldots$，使得对每个 $i \in \omega$，均有 $S_1 = B, S_{i+1} = \delta(S_i, w_i)$。我们用 ρ_w 表示 ρ_w^I。此外，给定映射 $\theta\colon 2^Q \to 2^Q$ 和状态集序列 ρ_w^B，我们定义$\theta(\rho_w^B) = \theta(B)B_2\ldots\ldots$，其中对每个 $i \in \omega$，均有 $B_{i+1} = \theta(\delta(B_i, w_i))$。若存在某个 $i \in \omega$ 使得 $\pi_{i:\omega} \preccurlyeq \pi'_{i:\omega}$，则称轨迹 π最终被 π' 模拟。

引理 6. 设 w 是一个词。那么，$\Pi_{\rho_w} \sim \Pi_{pr(\rho_w)}$。

证明。首先注意到 $\Pi_{pr(\rho_w)} \subseteq \Pi_{\rho_w}$。因此，只需证明：若存在接受轨迹 $\pi \in \Pi_{\rho_w}$，则也存在接受轨迹 $\pi' \in \Pi_{pr(\rho_w)}$。我们假设前者成立，现在证明存在 $\pi' \in \Pi_{pr(\rho_w)}$，使得 π 最终被 π'模拟。若 $\pi \in \Pi_{pr(\rho_w)}$，则证明完成。现在，假设情况并非如此，并且存在一个最大轨迹集 $P = \{\pi^1, \pi^2, \ldots\} \subseteq \Pi_{\rho_w}$，其索引为 $\mathrm{l}_1 < \mathrm{l}_2 < \ldots$，使得对于每个 i，都有 $p_i = \pi^i_{\ell_i} \sqsubseteq \pi^{i+1}_{\ell_i} = p'_i$，且 $\pi_1 = \pi$。根据定义，我们还可以得到 $\#(p_i) < \#(p'_i)$。由于这些数值有限，且无法到达编号更小的状态，该序列最终会稳定下来，因此 P 是有限的。由于集合 $P = \{\pi_1, \ldots, \pi_n\}$ 是最大且有限的，因此有 $\pi_n \in \Pi_{pr(\rho_w)}$。此外，$\pi' = \pi_n$ 最终会 $\preccurlyeq$-模拟 π（由模拟的逐步性质可知），至此证明完成。□

考虑函数 $\theta\colon 2^Q \to 2^Q$，并令 A 在词 w 关于 θ 的运行有向无环图为一个有向无环图 $\mathsf{G}_w^\theta = (V, E)$，其中包含顶点 V 和边 E，使得

- $V \subseteq Q \times \omega$ such that $(q, i) \in V$ iff $q \in \tau_i$ where $\tau = \theta(\rho_w^I)$,
- $E \subseteq V \times V$ such that $((q, i), (q', i')) \in E$ iff $i' = i + 1$ and $q' \in \delta(q, w_i)$.

我们用 G_w 表示 G_w^{id}。如果图中存在一条路径，无限多次经过对应于接受状态/转移的顶点，则称 G_w^θ 为接受的。

引理 7。 设 w 为一个词。那么，$\mathcal{G}_w^{pr}$为接受的，当且仅当$\mathcal{G}_w$ 为接受的。

证明。 直接由引理 6 得出。 □

所有 MH、CSB、CoB 和 RNK 补全算法都可以视为以运行有向无环图作为输入，并检查该图是否为接受图的过程。因此，只要确保修改后的有向无环图当且仅当原图为接受图，我们就可以以“任意”方式修改该有向无环图。因此，我们可以修改每种算法，使其将剪枝运行有向无环图考虑在内。这意味着，我们可以分别将函数 $\delta(W, a)$ 和 $\delta_F(W, a)$ 的每次调用改为 剪枝($\delta(W, a)$) 和 剪枝($\delta_F(W, a)$)。这是因为这些算法都在运行有向无环图这一层面上运作，而运行有向无环图正是由这些转移函数创建的。

C 基于秩的 NAC 补全

本节中，我们定义 RNK 算法，并加入条件 $\varphi_{\mathtt{RNK}}$，规定划分块 P 是一般非确定性接受分量。设 P 为 A 的一个划分，使得 $\mathrm{A}_P \models \varphi_{\mathtt{RNK}}$。这里，我们考虑针对 P 的基于秩的补全过程的一种框架实例化（我们采用 Schewe 的最优算法的一种修改版本，该算法来自 [45]），以及其在 [24,26] 中的优化。让我们从一些定义开始。

对于 $i \in \omega$，我们用 $\lfloor\!\lfloor i \rfloor\!\rfloor$ 表示小于或等于 i 的最大偶数，例如，$\lfloor\!\lfloor 42 \rfloor\!\rfloor = \lfloor\!\lfloor 43 \rfloor\!\rfloor = 42$。设 $f\colon X \to Y$ 是一个函数。我们用 $dom(f)$ 表示 f 的定义域。对于函数 g，我们用 $f \lhd g$ 表示这样一个函数：对于每个 $x \in X$，若 $x \in dom(g)$，则返回 $g(x)$；否则返回 $f(x)$。令 $Q_P^{\blacksquare} = P \cup \{\blacksquare\}$，其中 $\blacksquare$ 是一个新符号（用于表示 P 之外运行的秩；我们将其读作“*box*”），并用 $reach(q)$ 表示在 A 中从 q 可达的状态集合。给定状态集合 $T \subseteq Q$，我们定义 δ^T 如下：

- $\delta_P^T(\blacksquare, a) = (\delta(T \setminus P, a) \cap P) \cup A \quad \text{where } A = \begin{cases} \{\blacksquare\} & \text{if } reach(\delta(T,a) \setminus P) \cap P \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$
- $\delta_P^T(q, a) = \delta(q, a) \cap P \quad \text{for } q \neq \blacksquare.$

直观地说，δ_P^T 用于考虑 P 之外的运行（由 $\blacksquare$ 统一表示）。我们也照常将 δ_P^T 扩展到状态集合。

现在，我们来定义修改后的基于秩的过程所用的秩赋值。$Q_P^{\blacksquare}$-秩赋值是一个（部分）函数 $f\colon Q_P^{\blacksquare} \rightharpoonup \{0, \ldots\ldots, 2|Q_P^{\blacksquare}|\}$。$f$ 的秩是其值秩$(f) = \max\{f(q) \mid q \in Q_P^{\blacksquare}\}$。对于集合 $S \subseteq Q_P^{\blacksquare}$，若秩赋值 f 满足以下条件，则称其为 S-紧的：(i) $r =$秩(f)是奇数；(ii) f 是满射到$\{1, 3, \ldots\ldots, r\}$；(iii) $dom(f) = S$。注意，由于我们的定义中秩赋值是一个部分函数，其定义域会告诉我们哪些状态是活跃的；因此，我们无需为此另行维护一个独立集合（即 [45]中使用的集合 S）。此外，若一个$Q_P^{\blacksquare}$-秩赋值 f 满足以下条件，则称其为-紧的：

(i) f is $dom(f)$-tight and
(ii) if $\blacksquare \in dom(f)$ then $f(\blacksquare) = rank(f)$, and $rank(f \setminus \{\blacksquare \mapsto f(\blacksquare)\}) < rank(f)$.

直观地说，-紧秩赋值在其定义域上是紧的；如果定义域包含 $\blacksquare$，则的秩严格大于 P 中任意状态的秩。

对于一对 $Q_P^{\blacksquare}$-秩赋值 f 和 g，当且仅当满足以下条件时，我们定义 $f \circledcirc\!\!\rightarrow_T^a g$：

(i) 定义域$(g) = \delta_P^T($定义域$(f), a)$；(ii) 对于每个 $q \in$定义域(f) 以及 $q' \in \delta_P^T(q, a)$，都有 $g(q') \leq f(q)$；(iii) 对于每个 $q \in$定义域$(f) \setminus \{\blacksquare\}$ 和 $q' \in \delta_F(q, a)$，都有 $q' \in P \Rightarrow g(q') \leq \lfloor\!\lfloor f(q) \rfloor\!\rfloor$。

我们进一步定义 $f_1 \leq^{\bullet} f_2$ 当且仅当秩$(f_1) =$秩(f_2)，并且对于每个 $q \in dom(f_1)$，都有 $f_1(q) \leq f_2(q)$。我们还定义

$$maxrank_T(f, a) = \begin{cases} g_{\max} & \text{if } g_{\max} = \max_{\leq^{\bullet}} \{g \mid f \circledcirc\!\!\rightarrow_T^a g\} \text{ is tight} \\ \bot & \text{otherwise} \end{cases} \tag{4}$$

最后，我们准备给出一种适用于基于分解的构造的基于秩的补全过程实例。我们从类型的定义开始：

1. $\mathrm{PT}^{\mathrm{RNK}_P} = \mathrm{PT}^{wait}_P \cup \mathrm{PT}^{tight}_P$ 其中– $\mathrm{PT}^{wait}_P = 2^{Q^{\blacksquare}_P}$。这部分表示补全的BA的等待部分。请注意，我们使用 $2^{Q^{\blacksquare}_P}$ 而不是 2^P；这是因为我们需要跟踪是否存在某个能够到达 P 的运行（使用 表示）。– $\mathrm{PT}^{tight}_P = \{f \mid f$ 是一个 -紧的 $Q^{\blacksquare}_P$-秩$\}$。

2. $\mathrm{AT}^{\mathrm{RNK}_P} = \mathrm{AT}^{wait}_P \cup \mathrm{AT}^{tight}_P$ 其中– $\mathrm{AT}^{wait}_P = 2^{Q^{\blacksquare}_P}$ 且– $\mathrm{AT}^{tight}_P = \{(f, O, i) \mid f$ 是一个 -紧的 $Q^{\blacksquare}_P$-秩, $O \subseteq dom(f) \cap f^{-1}(i), i \in \{0, 2, \ldots, rank(f) - 1\}\}$。

实例 RNK_P 实现了针对 $\mathrm{SuccPass}^{\mathrm{RNK}_P}$的 MaxRank构造（论文 [24]中的版本），但不包含 η_4 （η_4 负责非确定性地降低秩）。随后，对于 $\mathrm{SuccAct}^{\mathrm{RNK}_P}$ ，我们使用标准MaxRank，其中一个宏状态可能有两个非确定性后继。形式上，实例 RNK_P提供以下函数：

– $\mathtt{Init}^{\mathrm{RNK}_P} = \{(I \cap P) \cup A \mid \text{if } reach(I) \cap P \neq \emptyset \text{ then } A = \{\blacksquare\} \text{ else } A = \emptyset\}$.
– $\mathtt{SuccPass}^{\mathrm{RNK}_P}$ is defined as follows based on the type of the partial macrostate:
  - $\mathtt{SuccPass}^{\mathrm{RNK}_P}(T, S, a) = \{(\delta^T_P(S, a), \emptyset)\}$ if $S \in \mathrm{PT}^{wait}_P$ else
  - $\mathtt{SuccPass}^{\mathrm{RNK}_P}(T, f, a) = \begin{cases} \{(f', \emptyset)\} & \text{if } f' = maxrank_T(f, a) \neq \bot, \\ \emptyset & \text{otherwise.} \end{cases}$

– $\mathrm{Lift}^{\mathrm{RNK}_P}$ 也会考虑部分宏状态的类型：yp p
  - $\mathtt{Lift}^{\mathrm{RNK}_P}(S) = \{S\} \cup \{(g, g^{-1}(0), 0) \mid g \in \max_{\leq^{\bullet}}\{f \mid f \text{ is } \blacksquare\text{-tight}, dom(f) = S\}\}$
  - $\mathtt{Lift}^{\mathrm{RNK}_P}(f) = \{(f, O', 0)\}$ where
    * $O' = f^{-1}(0)$.
    * Note that it can happen that $\mathtt{Lift}^{\mathrm{RNK}_P}(f) = \emptyset$, e.g., when $\mathtt{SuccPass}^{\mathrm{RNK}_P}(T, f, a) = \emptyset$.

– $\mathrm{SuccAct}^{\mathrm{RNK}_P}$ 根据部分宏状态的类型定义如下：
  - $\mathtt{SuccAct}^{\mathrm{RNK}_P}(T, S, a) = \begin{cases} \{(U, \{0\})\} \text{ with } U = \delta^T_P(S, a) \in \mathrm{PT}^{\mathrm{RNK}_P} & \text{if } S \subseteq \{\blacksquare\} \\ \mathtt{Lift}^{\mathrm{RNK}_P}(U) \in \mathrm{AT}^{\mathrm{RNK}_P} & \text{otherwise} \end{cases}$
  - $\mathrm{SuccAct}^{\mathrm{RNK}_P}(T, (f, O, i), a)$：首先定义函数 η_3和 η_4如下（对于一个 $Q^{\blacksquare}_P$-秩 g 和 $i \in \omega$，我们使用 $i{+}{+}^g = (i + 2) \bmod (\text{秩}(g) + 1)$）：
    * 如果 $\mathrm{SuccPass}^{\mathrm{RNK}_P}(T, f, a) = \{g\}$，则
      · $\eta_3 = \begin{cases} (g, \delta^T_P(O, a) \cap g^{-1}(i), i) & O \neq \emptyset, (\text{如果 } (dom(\cdot), \cdot)(\cdot, \cdot) \text{ 如果} \\ (g, \delta^T_P(f, a) \cap g^{-1}(i{+}{+}^g), i{+}{+}^g) & O = \emptyset \end{cases}$
      · $\eta_4 = \{(g', M, i) \mid g' = g \lhd \{q \mapsto g(q) - 1 \mid q \in M, i \neq 0\}\}$ 其中 $M = \delta^T_j(O, a) \cap g^{-1}(i)$ * 否则 $\eta_3(T, (f, O, i), a) = \eta_4(T, (f, O, i), a) = \emptyset$。

令 $U = \{(f', O', i') \in \eta_3 \cup \eta_4 \mid O' = \emptyset \wedge i' = \text{秩}(f') - 1\}$。接着定义 $\text{SuccAct}^{\text{RNK}_P}(T, (f, O, i), a) = \{((f', O', i'), \emptyset) \mid (f', O', i') \in \eta_3 \cup \eta_4 \setminus U\} \cup \{(f', \{\quad\}) \mid (f', O', i') \in U\}$ 注意，在前文中，第一个集合来自 AT^{RNK_P}，另一个来自 PT^{RNK_P}。

正确性可由以下引理概括。

引理8. 部分算法并集(RNK) 是正确的。

证明。（概要）在此证明中，我们用 MaxRank 表示 [24] 中的 MaxRank 算法，并用 InitDet 表示对自动机初始部分执行子集构造（该部分不含接受转移）的过程。考虑一个满足 $B \models_P \varphi_{\text{RNK}}$ 的 Büchi 自动机 B。此外，假设 B 没有冗余状态。考虑词 w 上、位于 ModCompl(Union(RNK$_P$),B) 中的一次运行$\rho = (H_1, M_1)(H_2, M_2)\ldots$。我们可以构造一次运行 ρ'，使其位于 w 上的 InitDet(MaxRank(B)) 中，并且 ρ'_i 可由 ρ_i 替换为 $H_i \setminus dom(f_i)$ 而得到，其中 f_i 是宏状态 M_i 的秩函数。容易证明，ρ 是接受运行当且仅当 ρ' 是接受运行。□

引理9. 部分轮询算法RNK 是正确的。

证明。（概要）我们从条件 (C1) 入手。对于任意词$w \in \text{L(ModCompl(Union(RNK}_P), \text{B}))$，我们可以构造一个接受运行 ρ，使得在清空 O-集合 ($O = \emptyset$) 后，我们可以切换到被动状态一个步骤，然后在下一步切换回活跃状态。由于我们需要无限多次清空 O-集合，因此何时进行新的采样并不重要（只需确保在被动阶段执行有限步即可）——这也满足条件 (C2)。条件 (C3) 可由以下事实推出：只有在 O-集合变为空时，才会从活跃阶段切换到被动阶段（因此，无限次切换意味着该运行是接受运行）。其余正确性由引理8推出。□

D 附加示例

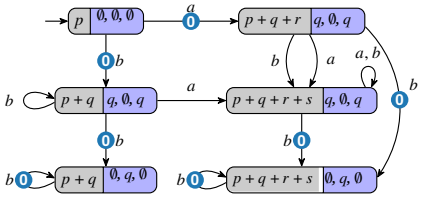
本节提供了该优化的更多示例。图3展示了延迟构造的示例，其中也描绘了构造的中间步骤。图4展示了轮询算法的示例。

初始确定性划分块的补全示例如图5所示：

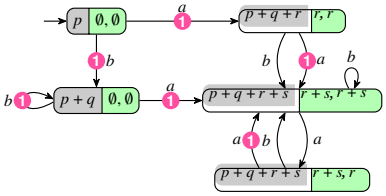
E 正文中遗漏的证明

E.1 第3节的证明

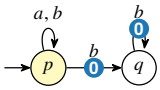
定理 1. 设$\mathcal{A}$ 是一个 *Büchi* 自动机，P_1、...、P_n 是$\mathcal{A}$的一个划分，且$\texttt{Alg}^1$,、...、$\texttt{Alg}^n$是一个部分补全算法序列，并且$\texttt{Alg}^i$ 对 P_i正确。那么，我们有 $\mathcal{L}(\mathit{ModCompl}(\texttt{Alg}^1_{P_1}、\ldots、\texttt{Alg}^n_{P_n}、\mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$。



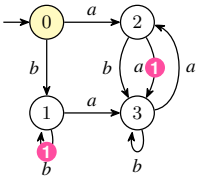
(a) $A_1 = \text{ModCompl}(\text{CSB}^{P_0}, A_{ex})$



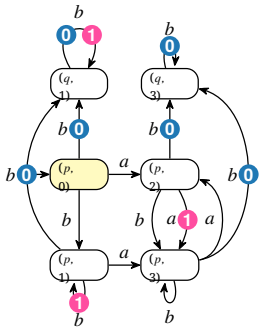
(b) $A_2 = \text{ModCompl}(\text{MH}^{P_1}, A_{ex})$



(c) 归约(A_1)



(d) 归约(A_2)



(e) 后补(CSB^{P_0}, MH^{P_1}, A_{ex})

图 3：将延迟构造应用于 A_{ex} 的示例，所得结果的接受条件 Acc：Inf( ) ∧ Inf( )。

证明. 设 $A = (Q, \delta, I, F)$ 是一个 Büchi 自动机。此外，由于 P_1、…、P_n 是 A 的一个划分，我们有

$$\bigcup_{i=1}^{n} \mathcal{L}(\mathcal{A}_{P_i}) = \mathcal{L}(\mathcal{A}) \tag{5}$$

在证明的第一部分，我们证明以下断言

断言 2：

$$\bigcap_{i=1}^{n} \mathcal{L}(\mathit{ModCompl}(\mathtt{Alg}^i_{P_i}, \mathcal{A}_{P_i})) = \mathcal{L}(\mathit{ModCompl}(\mathtt{Alg}^1_{P_1}, \ldots, \mathtt{Alg}^n_{P_n}, \mathcal{A})) \tag{6}$$

证明：($\subseteq$) 考虑一个词 $\alpha \in \bigcap_{i=1}^{n} \mathrm{L}(\text{ModCompl}(\text{Alg}^i_{P_i}, \text{A}^{P_i}))$。那么，存在形如 $\rho_i = (H^1_i, M^1_i)(H^2_i, M^2_i)\ldots$ 的接受运行 ρ_i，使得 $H^{\ell+1}_i = \delta(H^\ell_i, \alpha_\ell)$，并且对每个 $1 \le i \le n$，都有 $M^{\ell+1}_i \in \text{Succ}^{\mathtt{Alg}^i_{P_i}}(H^\ell_i, M^\ell_i, \alpha_\ell)$。根据这些运行的定义，我们可知，对每个 $1 \le i$，$H^\ell_i = H^\ell_j, j \le n$。因此，在自动机 $\text{ModCompl}(\text{Alg}^1_{P_1}, \ldots, \text{Alg}^n_{P_n}, \text{A})$ 中，$\varrho = (H^1_1, M^1_1, M^1_2, \ldots, M^1_n)(H^2_1, M^2_1, M^2_2, \ldots, M^2_n)\ldots$ 也是 α 上的一个运行。由于对每个 i，都有 $\rho_i \models \text{Acc}^{\mathtt{Alg}^i_{P_i}}$，我们也有 $\varrho \models \bigwedge_{i=1}^{n} \text{Acc}^{\mathtt{Alg}^i_{P_i}}$，这意味着 ϱ 是 $\text{ModCompl}(\text{Alg}^1_{P_1}, \ldots, \text{Alg}^n_{P_n}, \text{A})$ 中的一个接受运行。

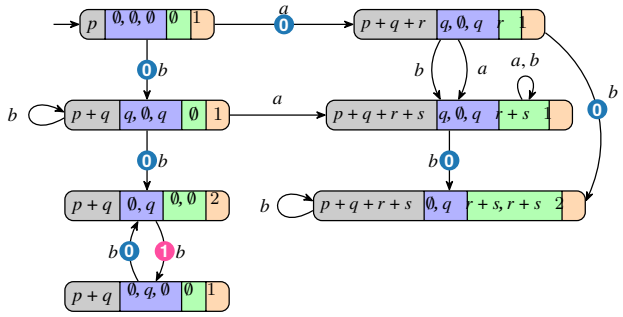


图4：在图1所示的 Büchi 自动机上应用带有 Acc: Inf( ) ∧Inf( ) 的 ModComplRR(CSBRR$_{P_0}$, MHRR$_{P_1}$, A$_{ex}$) 的结果。

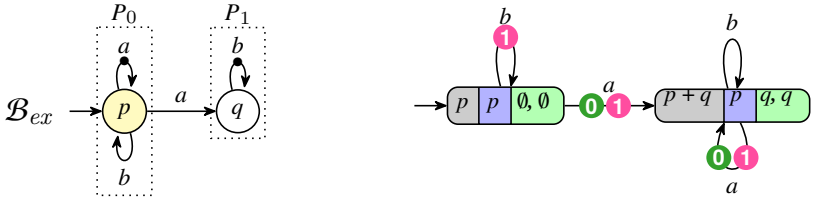


图5：左：B$_{ex}$。右：带有 Acc: Fin( ) ∧Inf( ) 的 ModCompl(CoB$_{P_0}$, IH$_{P_1}$, B)。

(⊇) 考虑一个词 $\alpha \in \mathrm{L}(\mathrm{ModCompl}(\mathrm{Alg}^1_{P_1}, \ldots, \mathrm{Alg}^n_{P_n}, \mathrm{A}))$。那么，存在一个接受运行 $\varrho = (H^1, M^1_1, M^1_2, \ldots, M^1_n)(H^2, M^2_1, M^2_2, \ldots, M^2_n)\ldots$，其运行于 α 上。根据 ModCompl 的定义，对每个 $1 \le i \le n$，在 $\mathrm{ModCompl}(\mathrm{Alg}^i_{P_i}, \mathrm{A}_{P_i})$ 中，α 上都存在运行 $\rho_i = (H^1, M^1_i)(H^2, M^2_i)\ldots$。由于 $\varrho \vDash \bigwedge_{i=1}^n \mathrm{Acc}^{\mathrm{Alg}^i_{P_i}}$，我们可知每个 ρ_i 也都是接受运行。■

接下来我们按如下步骤进行。由定义 1 可得

$$\mathcal{L}(\textsc{ModCompl}(\mathtt{Alg}^i_{P_i}, \mathcal{A}_{P_i})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A}_{P_i})$$

因此，利用（5）

$$\bigcap_{i=1}^n \mathcal{L}(\textsc{ModCompl}(\mathtt{Alg}^i_{P_i}, \mathcal{A}_{P_i})) = \Sigma^\omega \setminus \left(\bigcup_{i=1}^n \mathcal{L}(\mathcal{A}_{P_i})\right) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A}),$$

结合（6），即可完成证明。 □

E.2 第 4 节的证明

引理1. 部分算法 MH是正确的。

证明。 令 w 为一个 ω-词。我们的目标是证明： w 在 P 中不被接受，当且仅当无限次出现。

假设存在一个宏状态序列 $\hat{\rho} = (C_0, B_0) \cdots (C_i, B_i) \cdots$，其在 w 上无限次发出颜色 。我们的目标是证明 w 在 P中不被接受。注意，A 是完备的，因此 A 在 w 上的每个运行 ρ 都是无限运行。由于 P中的所有强连通分量都是接受的且本质弱，我们只需证明每个进入 P 的运行ρ 最终都会离开 P。首先，设 ρ 在某个时刻进入 P，例如在 $k > 0$。也就是说，我们有 $\rho_k \in C_k$。由于 $\hat{\rho}$ 无限次发出颜色 0，因此必定存在一个整数 $l \geq k$，使得 $\delta(B_\ell, w_\ell) \cap C'_{\ell+1} = \emptyset$。由此可得，对于从 ρ 分支出的所有运行，其第 $(l+1)$ 个状态为 $\rho'_{\ell+1}$时，都满足 $\rho'_{\ell+1} \in B_{\ell+1} = C'_{\ell+1}$。因此，从$\rho$ 分支出的所有运行都会出现在 $B_{\ell+1}$-集合中。同样，根据假设，必定存在一个整数 $l' \geq l$，使得 $\delta(B_{\ell'}, w_{\ell'}) \cap C'_{\ell'+1} = \emptyset$。由此可得，对于所有 $j > l'$，从 ρ 分支出的所有运行都必须已离开 B_j-集合。由于 无限次出现，也就是说，沿着 $\hat{\rho}$存在无限多个空的 B-集合，因此所有进入 P 的运行最终都必须离开 P。因此， w 在划分块 P中不被接受。

现在我们假设 w 未被 P 接受，并证明 会无限次出现。我们用反证法证明。假设沿着 w上的宏状态序列 $\hat{\rho} = (C_0, B_0) \cdots (C_i, B_i) \cdots$， 只出现有限次。那么存在一个整数 $k > 0$，使得对所有 $j \geq k$，都有 $B_j \neq \emptyset$。由此可得，对所有 $j \geq k$，都有 $B_{j+1} = \delta(B_j, w_j) \cap C'_{j+1} = \delta(B_j, w_j) \cap \delta(H_j, w_j) \cap P$，也就是说，对所有 $j \geq k$，都有 $B_{j+1} \subseteq \delta(B_j, w_j) \subseteq P$。根据 König引理， $B \subseteq P$中必定存在一个无限运行 ρ。由于 P 中的所有强连通分量都是接受的且本质弱，我们知道 ρ 必定是接受的，这与假设 w 未被 P接受相矛盾。因此，我们证明了 必定会无限次出现。

因此，我们已经证明， w 在 P 中不被接受，当且仅当 无限次出现。

引理 2. 部分算法CSB 是正确的 *ct.*

证明。 设 w 是一个 ω-词。我们的目标是证明， w 在 P 中不被接受，当且仅当存在一个无限的宏状态序列 $\hat{\rho} = (C_0, S_0, B_0) \cdots$，它无限次发出颜色 。

首先，假设存在一个宏状态的无限序列 $\hat{\rho} = (C_0, S_0, B_0) \cdots$，其上的 w 会无限多次发出颜色 。接下来需要证明，A 在 w 上的一个进入 P 的运行 ρ 要么会在某个时刻离开，要么不是接受运行。为简单起见，令 $h > 0$ 为一个整数，使得对于所有 $j \geq h$，ρ_j 都属于 P 中的同一个强连通分量（SCC）。当运行 s 上的一个状态从某个强连通分量转移到另一个强连通分量中的状态 t 时，我们称该运行已经消亡，并且有一个新运行从状态 t进入 P。假设运行 ρ 在时刻 $k \geq h$ 出现在 P 中。一旦 ρ处于 P中，我们就知道 ρ是确定性的，即不会从 ρ导出分支运行。因此，我们只需关注这个确定性运行。令 $\rho_k \in C_k$（$\rho_k \in B_k$和 $\rho_k \in S_k$的情形更简单，稍后讨论）。由于 无限多次出现，必然存在一个整数 $l \geq k$，使得 $B^{\star}_{\ell+1} = \delta_{\mathrm{SCC}}(B_\ell, w_\ell) = \emptyset$。因此，$(C_{\ell+1}, S_{\ell+1}, B_{\ell+1})$ 有两种可能：(1) $(C'_{\ell+1}, S'_{\ell+1}, B'_{\ell+1} = C'_{\ell+1})$，即将 ρ 移动到 B-集合；(2) $(C''_{\ell+1}, S''_{\ell+1}, B''_{\ell+1} = C''_{\ell+1})$，即将 ρ移动到 S-集合。由于 $\rho_{\ell+1} \in C'_{\ell+1}$，我们必有 $\rho_{\ell+1} \in B_{\ell+1}$或 $\rho_{\ell+1} \in S_{\ell+1}$。假设 $\rho_{\ell+1} \in S_{\ell+1}$。由于 $\hat{\rho}$无限多次发出 0，$\hat{\rho}$必须是无限的。也就是说，对于所有 $j \geq l+1$，都有 $\delta_F(B_j, w_j) \cap P = \emptyset$。由于对于所有 $j \geq l+1$都有 $\rho_j \in \delta_{\mathrm{SCC}}(B_j, w_j) \cap P$，根据定义，对于所有 $j \geq l+1$都有 $\delta_F(B_j, w_j) = \emptyset$。也就是说，在 $j \geq l+1$之后， ρ不再访问接受转移；

否则，如果存在某个 j使得 $\delta_F(B_j, w_j) \neq \emptyset$，那么 $\hat{\rho}$将是有限的。如果 $\rho_{\ell+1} \in B_{\ell+1}$，我们知道必然存在一个整数 $l' \geq l$，使得 $\delta_{\mathrm{SCC}}(B_{\ell'}, w_{\ell'}) = \emptyset$。也就是说，$\rho_{\ell'+1} \notin P$ 或 $\rho_{\ell'+1} \in P$ 成立，但 $\rho_{\ell'+1}$ 和 $\rho_{\ell'}$ 不在同一个强连通分量（SCC）中。在后一种情况下，我们将 ρ 视为已经消亡，并且会有一个新运行进入 C。由于 ρ 最终会永远停留在某个强连通分量中，因此很容易看出，ρ 要么处于 B 中，要么离开 P。因此，我们证明了：A 在 w 上进入 P 的所有运行，要么会在某个时刻离开，要么不是接受运行。

现在，假设 w 未被 P接受。我们的目标是证明，存在一个无限的宏状态序列 $\hat{\rho} = (C_0, S_0, B_0) \cdots$，其取值于 w，并且会无限次发出颜色 。由于 A 在 w 上的每个运行 ρ要么离开 P，要么变得安全，因此我们可以构造这样一个无限序列 $\hat{\rho}$。首先，我们需要确保 $\hat{\rho}$ 是无限的，并且只需谨慎处理条件 $\delta_F(S, a) \neq \emptyset$。$S$中的所有运行都可以看作来自 B（包括 (C'', S'', C'')，因为它仍需先计算 B'）。构造 $\hat{\rho}$ 时，我们只需处理非确定性选择。如果 B 始终为空，那么证明完毕。否则，令 k 为满足 $B \neq \emptyset$的最小整数。也就是说，当前宏状态为 (C_k, S_k, B_k)。由于 B 中的所有运行都是确定性的，我们可以采用标准构造（沿着后继 (C', S', B')）进行），直到到达这样的一个位置：B-集合中的所有运行要么消亡，要么变得安全。如果 B 变为空，我们仍沿着后继 (C', S', B')继续，且构造会发出 。由于 B 中的运行数量有限，并且根据假设它们最终都会变得安全，因此 B 中的所有运行可能都会变得安全。在这种情况下，容易看出 $\delta_F(B, a) \cap \delta_{\mathrm{SCC}}(B, a) = \emptyset$。这次我们沿着后继 (C'', S'', C'')继续，并发出 。由于我们移动到 S的所有运行都是安全的，因此今后不会满足 $\delta_F(S, a) \neq \emptyset$。如此一来，对于某个 $l \geq 1$，我们得到一个宏状态 $(C_{k+\ell}, S_{k+\ell}, B_{k+\ell})$。我们可以重复上述过程，构造一个取值于 w 的无限宏状态序列 $\hat{\rho}$，并使其无限次发出颜色 。□

引理 3. 部分算法 CoB 是正确的。

证明。设 w 是一个 ω-词。我们需要证明，当且仅当我们仅有限次收到颜色 时，w 不会被 P 接受。

首先，我们用逆否命题证明从右到左的方向。根据假设，沿着词 w，颜色 只出现有限次。假设 w 被 P接受。那么必定存在一个接受运行 ρ，它最终始终停留在 P中。容易看出，ρ 被约简确定性 Büchi 自动机 A_P接受。设 $k \geq 0$ 是满足 $\rho_k \in P$的最小整数。因此，对于所有 $j \geq k$，都有 $\mathrm{Succ}^{\mathrm{CoB}_P}(H_j, \rho_j, w_j) = \{(\rho_{j+1}, \alpha_{j+1})\}$。由于 ρ 将访问无限多个接受转移，我们也会无限多次看到颜色 。这与我们的假设矛盾。因此，w不能被 P接受。

其次，我们也通过逆否命题证明另一个方向。根据假设，w 未被 P接受。假设我们看到无穷多个 ，并且 w 上的宏状态序列为 。那么必定存在无穷多个整数 $k > 0$，使得$\hat{\rho}_k$、$\hat{\rho}_{k+1} \in P$且$\hat{\rho}_k \xrightarrow{w_k} \hat{\rho}_{k+1} \in F$。若对所有 $j \geq k$都有$\hat{\rho}_j \in P$，那么我们必定在 P中有一个接受运行，这与假设 w 未被 P接受相矛盾。因此，必定存在某个整数$l > k$，使得 $\delta(H_\ell, w_\ell) \cap P = \emptyset$。这表明所有从$\hat{\rho}_k$ 开始的运行都是有限的。由于 P 是确定性的，因此 w 上所有进入 P 的运行都是有限的，

因此 w 未被 P接受。矛盾。由此，我们证明了如果 w 未被 P接受，那么颜色 只会出现有限次。□

定理 2. 设 $\mathcal{A}$ 是一个具有 n 个状态的电梯自动机。那么存在一个有 $O(4^n)$ 个状态的 *Büchi* 自动机，其接受语言为 $\mathcal{L}(\mathcal{A})$ 的补集。

证明。设 Q_D 是 A 中所有满足 φ_{CSB}的强连通分量的并集，Q_W 是所有满足 φ_{MH} 的强连通分量的并集，Q_N 是所有非接受强连通分量的并集；此外，$Q_D \cap Q_W = \emptyset$、$Q_N \cap Q_W = \emptyset$和 $Q_N \cap Q_D = \emptyset$。由于 A 是电梯自动机，$Q_D \cup Q_W \cup Q_N$是 A 中所有状态的集合，$Q_D \cup Q_W$ 是 A 的所有划分块的并集。由定理 1、引理1和引理 2可知，L (ModCompl(CSB$_{Q_D}$, MH$_{Q_W}$, A)) =$\Sigma^\omega \setminus$ L (A)。现在我们计算 ModCompl(CSB$_{Q_D}$, MH$_{Q_W}$, A) 的状态数。对于状态 $q \in Q_D$，在 $\text{T}^{\text{CSB}_{Q_D}}$ 中分配 q 有 4 种可能：(i) $q \notin C \cup S$，(ii) $q \in C$，(iii) $q \in C \cap B$，(iv) $q \in S$。对于状态 $q \in Q_W$，在 $\text{T}^{\text{MH}_{Q_W}}$中分配 q 有 3 种可能：(i) $q \notin C$，(ii) $q \in C$，(iii) $q \in C \cap B$。最后，对于状态$q \in Q_N$，在可达状态 H 中分配 q 有 2 种可能：$q \in H$ 或$q \notin H$。因此，宏状态的数量为 $4^{|Q_D|} \cdot 3^{|Q_W|} \cdot 2^{|Q_N|} \in$ O(4^n)。

□

E.3 第 5 节的证明

定理 3. 设 $\mathcal{A}$ 是一个 *Büchi* 自动机，$P_1, \ldots, P_n$ 是对$\mathcal{A}$的划分，且$\texttt{Alg}^1, \ldots, \texttt{Alg}^n$是一个部分补全算法序列，并满足$\texttt{Alg}^i$ 对 P_i正确。则$\mathcal{L}$(后补($\texttt{Alg}^1_{P_1}, \ldots, \texttt{Alg}^n_{P_n}, \mathcal{A}$)) $= \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$。

证明。 由断言 2 和定理 1 可知

$$\bigcap_{i=1}^{n} \mathcal{L}(\textsc{ModCompl}(\texttt{Alg}^i_{P_i}, \mathcal{A}_{P_i})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A}).$$

由于归约 Red 保持语言不变，对于每个 i，都有 L(ModCompl(Alg$^i_{P_i}$, A$_{P_i}$)) =L $\Big($Red $\big($ModCompl(Alg$^i_{P_i}$, A$_{P_i}$)$\big)\Big)$，由此证明得证。□

F 实验的更多图表

本节将展示更多关于实验结果的图表。

图 6 给出了仙人掌图，展示了每种工具（包括虚拟最佳求解器）在 y 轴所示时间限制内成功完成补自动机的基准测试数量（x 轴）；曲线越靠近右侧边界，工具的表现就越好。图 7 更清晰地展示了图 6 中状态数超过 39,000 的部分。从图中可以看出，在考虑补全输入 TBA 所需的时间时，Spot 显然胜出，因为它的曲线几乎与两个 VBS 的曲线重合；这证实了 Spot 的高品质和成熟度，以及它所实现的多种技术能够有效处理 Büchi 自动机操作。在需要较短时间完成补自动机的自动机上，Kofola$_P$ 略优于 Kofola$_S$，且与 COLA 表现非常接近；之后，Kofola 的两个版本表现相似，而 COLA 生成较大的自动机时速度稍快。

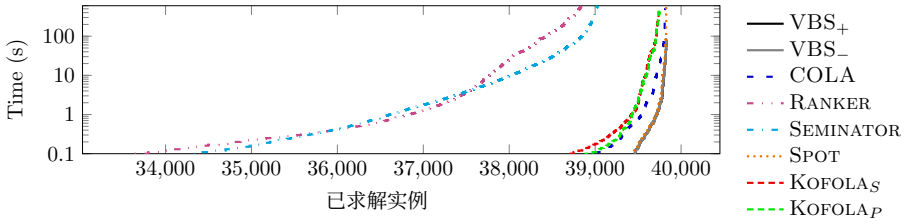


图 6：仙人掌图，显示各工具在纵轴所示时间内解决的实例数量。

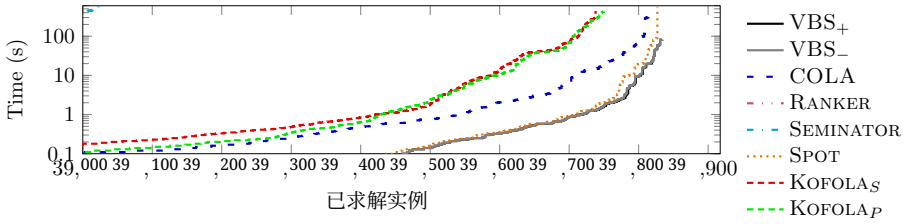


图 7：图 6 中仙人掌图的局部细节。