arXiv:2301.07563v2 [cs.GT] 6 Apr 2023

计算综合的充分许可假设

Ashwani Anand[1],Kaushik Mallik[2],Satya Prakash Nayak[1],和
Anne-Kathrin Schmuck[1]

[1] 马克斯·普朗克软件系统研究所，德国凯撒斯劳滕
{ashwani,sanayak,akschmuck}@mpi-sws.org

[2] 奥地利科学与技术研究所，奥地利克洛斯特新堡 kaushik.mallik@ist.ac.at

摘要。 我们解决了在双人回合制有限图博弈中自动计算一类新环境假设的问题。这类假设刻画了环境为使系统玩家获胜所需提供的“充分合作”。给定系统玩家的一个 ω 正则获胜条件 Φ，我们计算一个针对环境玩家的 ω 正则假设 Ψ，使得：(i) 每个遵守 Ψ 的环境策略都允许系统满足 Φ （充分性）；(ii) 对系统的每个策略，环境都能满足 Ψ （可实现性）；(iii) Ψ 不会排除任何合作策略选择（宽容性）。

对于奇偶博弈（即 ω-正则博弈的规范表示），我们提出了一种用于符号计算充分宽松假设的多项式时间算法，并证明无论在理论上还是在实证上，我们的算法都比现有方法运行更快、生成的假设更优。据我们所知，对于 ω-正则博弈，我们提供了首个能够计算出充分且可实现、同时也具有宽容性的环境假设的算法。

关键词： 综合 · 双人博弈 · 奇偶博弈 · Büchi 博弈 · co-Büchi 博弈 · 符号算法。

1 引言

有限图上的双人 ω-正则博弈是计算机科学和信息物理系统设计中许多重要问题的核心算法组件。示例包括综合响应环境输入的程序、模态 μ 演算模型检测、面向信息物理系统的正确性设计控制器综合，以及自主系统的监督控制。

这些问题最终都可以归约为一个抽象的双人博弈，博弈双方分别是环境玩家 和系统玩家，分别代表外部不可预测的影响和正在设计的系统，而博弈则刻画了这两部分之间复杂的相互作用。该博弈的

博弈 是系统玩家为满足给定的ω-正则时序性质而需要做出的一系列决策，该性质定义在博弈的状态上；随后，这些决策将用于设计所需的系统或其控制器。

传统上，图上的双人博弈采用零和方式求解，即假设环境会任意行动，甚至可能采取对抗行为。尽管这种方法能够产生稳健的系统设计，但它通常会使环境过于强大，以至于系统无法实现。然而在现实中，前文概述的许多应用领域实际上都考虑到系统组件之间存在一定程度的协作，尤其是在这些组件经过协同设计的情况下。在这种情形下，了解环境（即其他进程）需要如何协作才能使系统得以实现，就很有用。这可以通过环境假设来形式化：这些假设是 ω-正则时序性质，用于限制综合博弈中环境玩家的行动。随后，这些假设可以作为额外的规格，用于其他组件的综合问题，以强制实现必要的协作（也可以与其他局部要求结合使用）；或者，也可以用于验证现有实现。

基于上述原因，假设的自动计算在反应式综合社区中受到了广泛关注。它已应用于双人博弈 [8,6], ，既用于单体系统设计 [11,20] ，也用于分布式系统设计 [19,13]。

所有这些工作都强调了假设应具备的两项理想属性。它们应当（i）充分，也就是说，当环境遵守其假设时，系统玩家能够获胜；以及（ii）可实现，也就是说，防止系统玩家通过违背假设而不遵守原始规约，从而在博弈中轻易获胜。本文提出，计算出的假设用于分布式综合时，还需要具备一项重要的第三属性——宽容性。如果一个假设保留了博弈中所有合作获胜的对局，那么该假设就是宽容的 。这一概念对分布式综合场景至关重要，因为在这种场景下，假设是在每个组件的实现方案确定之前 生成的。因此，假设需要保留所有 可行的协作方式，以便以去中心化方式找到分布式实现。

虽然本文所考虑的假设类别源于其在分布式综合中的应用，但本文仅关注这些假设的形式化与计算。也就是说，给定一个定义在有限图上的双人博弈，以及系统玩家的一个 ω-正则获胜条件 Φ，我们会自动计算出一个充分宽容 ω-正则假设 Ψ，供环境玩家使用。该假设通过满足以下条件来形式化上述直觉：(i) 充分，(ii) 可实现，且 (iii) 宽容。我们利用的主要观察是，这类充分宽松假设（简称 APA）可以由三个简单的模板构造而成，这些模板可直接从合作综合博弈中提取，从而得到一个用于计算它们的多项式时间算法。受篇幅限制，我们将一个非常有趣但与本文主题关联较弱的问题——如何使用 APA 进行基于契约的分布式综合——留待未来工作研究。

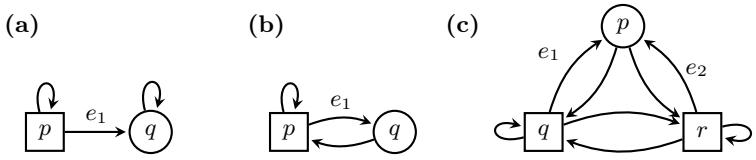

图1： 包含环境（方形）和系统（圆形）顶点的博弈图。

为了体会我们所用假设模板的简洁性，请看图1所示的博弈图，其中系统玩家和环境玩家分别控制圆形顶点和方形顶点。给定规范 $\Phi = \Diamond\Box\{p\}$ （要求对局最终只经过顶点 p），系统玩家可以在图1(a)的博弈中通过要求环境完全禁用边 e_1 来获胜。这引出了第一种模板类型——安全模板——作用于 e_1。另一方面，在图1(b)的博弈中，只要求 e_1 仅被采取有限次。我们的第二种模板类型——共活性模板——作用于 e_1，便刻画了这一要求。最后，考虑图1(c)中的博弈，其规范为 $\Phi = \Box\Diamond\{p\}$，也就是说，顶点 p 应当被无限次地经过。在这种情况下，如果边 e_1 和 e_2 的源顶点都被无限次地经过，那么只要其中一条边也被无限次地采取，系统玩家就能获胜。这由边组 $\{e_1, e_2\}$ 上的第三种模板类型——活跃群组模板——刻画。

贡献。 本文的主要贡献是证明，充分宽松假设总能由所概述的三种假设模板组合而成，并且可以在多项式时间内计算。

我们使用取自 SYNTCOMP [2] 的一组基准示例，以及在新工具 SImPA 中对算法的原型实现，通过实证表明，我们的算法速度更快，且生成的解决方案比现有方法更理想。此外，我们将 SImPA 应用于文献 [22],中广为人知的双客户端仲裁器综合基准问题。众所周知，只有在对客户端的行动加以适当限制时，才能实现该仲裁器。我们表明，将 SImPA 应用于无约束的仲裁器综合问题，会得到对客户端限制较少、但在概念上与文献中通常采用的假设相似的假设。

相关工作。 针对综合而自动计算环境假设的问题，Chatterjee 等人 [8]早已进行了研究。然而，他们所考虑的假设类别通常不允许构造宽容的 假设。此外，计算他们的假设是一个 NP-困难问题，而对于具有 n 个顶点的奇偶博弈，我们的算法可在 $\mathcal{O}(n^4)$ 时间内计算 充分宽松假设。复杂度差异源于 Chatterjee 等人要求假设具有最小性。另一方面，我们以最小性换取宽容性，从而能够利用更易求解的合作博弈。

在考虑非零和博弈的合作解时，相关工作要么为双方玩家固定策略 [7,14]，要么假设环境表现得特别理性 [4] ，要么将研究范围限制在安全假设 [19]。相比之下，我们不对环境如何选择策略作任何假设。最后，在零和博弈中的规范修复背景下，有多种

自动化方法可用于修复环境模型，例如 [23,15,16,21,8]。遗憾的是，这些方法都无法提供宽松的修复。Cavezza 等人近期的一项工作 [6] 计算出一组限制最少的假设，但仅适用于 GR(1) 规格，而这类规格只是我们所研究问题的一个严格子类。据我们所知，我们首次提出了一种完全自动化的算法，用于计算一般 ω-正则博弈的宽松假设。

2 预备知识

记号。 我们用 $\mathbb{N}$ 表示包含零的自然数集合。给定两个自然数 a，$b \in \mathbb{N}$，满足 $a < b$，我们用 $[a;b]$ 表示集合 $\{n \in \mathbb{N} \mid a \leq n \leq b\}$。对于任意给定的集合 $[a;b]$，我们将 $i \in_{\text{even}} [a;b]$ 和 $i \in_{\text{odd}} [a;b]$ 分别简写为 $i \in [a;b] \cap \{0,2,4,\ldots\}$ 和 $i \in [a;b] \cap \{1,3,5,\ldots\}$。给定两个集合 A 和 B、一个关系 $R \subseteq A \times B$ 以及一个元素 $a \in A$，我们用 $R(a)$ 表示集合 $\{b \in B \mid (a, b) \in R\}$。

语言。 令 Σ 为一个有限字母表。记号 Σ^* 和 Σ^ω 分别表示由 Σ 构成的有限词集合和无限词集合，且 Σ^∞ 等于 $\Sigma^* \cup \Sigma^\omega$。对于任意词 $w \in \Sigma^\infty$，w_i 表示 w 中第 i 个符号。给定两个词 $u \in \Sigma^*$ 和 $v \in \Sigma^\infty$，u 与 v 的连接写作词 uv。

博弈图。 一个博弈图 是一个元组 $G = (V, V^0, V^1, E)$，其中 (V, E) 是一个有限有向图，具有顶点 V和边 E，而 $V^0, V^1 \subseteq V$构成 V 的一个划分（即 $V^0 \cap V^1 = \emptyset$ 和 $V^0 \cup V^1 = V$）。不失一般性，我们假设对于每个 $v \in V$，都存在 $v' \in V$ 使得 $(v, v') \in E$。在本文中，系统 玩家和环境 玩家分别记为玩家 0 和玩家 1。始于顶点 v_0 的对局 是一个有限或无限的顶点序列 $\rho = v_0 v_1 \ldots \in V^\infty$。对局前缀 $\mathrm{p} = v_0 v_1 \cdots v_k$是一个有限对局。

获胜条件。给定一个博弈图 G，我们考虑使用公式 Φ 在线性时序逻辑（LTL）中指定的、定义在顶点集 V上的获胜条件；也就是说，我们考虑原子命题为顶点集合V的 LTL 公式。在这种情况下，所需的无限对局集合由 Φ在 G上的语义给出，它是一个 ω-正则语言 $\mathcal{L}(\Phi) \subseteq V^\omega$。任何具有任意 ω-正则所需无限对局集合的博弈图，都可以归约为一个带有 LTL 获胜条件的博弈图（可能具有扩展后的顶点集），如上所述。为简洁起见，省略 ω-正则语言和 LTL 的标准定义；相关内容可见标准教科书 [3]。

博弈与策略。 一个双人（轮流行动）博弈 是一个二元组 $\mathcal{G} = (G, \Phi)$，其中 G 是一个博弈图，Φ 是定义在 G上的获胜条件 。玩家玩家 i，$i \in \{0,1\}$的策略是一个偏函数 $\pi^i\colon V^* V^i \to V$，满足对于所有使 π 有定义的 $\mathrm{p}v \in V^* V^i$，都有 $\pi^i(\mathrm{p}v) \in E(v)$。给定策略$\pi^i$，若对所有 $k \in dom(\rho)$，$v_{k-1} \in V^i$ 都蕴含$v_k = \pi^i(v_0 \ldots v_{k-1})$，且只有当 $\pi^i(\mathrm{p})$ 未定义时，ρ才是有限对局，则称对局 $\rho = v_0 v_1 \ldots$ 与 π^i 相符合 。我们将与 π^i 相符合的对局以及与

同时与 π^0 和 π^1 相符合的对局分别称为 π^i-对局 和 $\pi^0\pi^1$-对局。我们将所有与 π^i相符合的对局，以及同时与 π^0 和 π^1 相符合的对局，分别收集在集合 $\mathcal{L}(\pi^i)$ 和$\mathcal{L}(\pi^0\pi^1)$中。

获胜。给定一个博弈 $\mathcal{G} = (G, \Phi)$，若策略 π^i 对玩家 i 而言（必然）获胜，则 $\mathcal{L}(\pi^i) \subseteq \mathcal{L}(\Phi)$；也就是说，玩家 0 的策略 π^0 若对每个玩家 1 的策略 π^1 都满足 $\mathcal{L}(\pi^0\pi^1) \subseteq \mathcal{L}(\Phi)$，则该策略获胜。同理，固定的策略组合 (π^0, π^1) 若满足 $\mathcal{L}(\pi^0\pi^1) \subseteq \mathcal{L}(\Phi)$，则为合作制胜。若存在一个获胜策略 π^i （或合作制胜策略组合 (π^0, π^1)），使得 $\pi^i(v)$ 有定义，则称顶点 $v \in V$ 对玩家 i 而言是获胜的（或合作制胜的）。我们将玩家 i 的所有获胜顶点汇集为玩家 i 的获胜区域 $\langle\langle i\rangle\rangle\Phi \subseteq V$，并将所有合作制胜顶点汇集为合作制胜区域 $\langle\langle 0,1\rangle\rangle\Phi$。注意，对于两者 $i \in \{0,1\}$，$\langle\langle i\rangle\rangle\Phi \subseteq \langle\langle 0,1\rangle\rangle\Phi$。

3 用于综合的适度宽容假设

给定一个双人博弈 $\mathcal{G}$，本文的目标是计算 Player *1* （即环境）上的假设，使双方恰好充分合作以满足 Φ，同时保留所有可能的合作策略选择。为了将这一直觉形式化，我们定义假设下获胜。

定义 1。设 $\mathcal{G} = ((V, V^0, V^1, E)\ \Phi)$ 为一个博弈，Ψ 为关于 V 的线性时序逻辑公式。那么，如果对于每个 *Player* 1 策略 π^1，满足 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$，均有 $\mathcal{L}(\pi^0\pi^1) \subseteq \mathcal{L}(\Phi)$，则 *Player* 0 策略 π^0 在 $\mathcal{G}$ 中假设 Ψ下获胜。我们将存在此类 *Player* 0 策略的顶点集合记为 $\langle\langle 0\rangle\rangle_\Psi\Phi$。

我们可以看到，定义1中引入的假设 Ψ 削弱了环境玩家（玩家1）的策略选择。如果这种削弱足够强，使得玩家0能够从合作获胜区域中的每个顶点获胜，我们就称该假设为充分。

定义2。如果 $\langle\langle 0\rangle\rangle_\Psi\Phi \supseteq \langle\langle 0,1\rangle\rangle\Phi$，则假设 Ψ对于(G, Φ) 是充分的。

遗憾的是，充分假设可能被滥用，以非预期的方式改变给定的综合问题。例如，考虑图2（左）中的博弈，其中 $\Phi = \square\Diamond\{v_0\}$ 和 $\Psi = \square\Diamond e_1$。这里不存在满足 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$ 的玩家1策略 π^1，因为系统只需在 v_1 中不再选择 e_1，就能始终使该假设失效。因此，即使 Φ 被违反，任何玩家0策略在该假设下都是获胜的。然而，假设 Ψ显然是充分的，因为 $\langle\langle 0\rangle\rangle_\Psi\Phi = V$。为防止充分假设失效，从而导致空洞的获胜，我们定义可实现性这一概念，以确保 Ψ 只限制玩家1的移动。

定义 3. 假设 Ψ 对于 (G, Φ) 是可实现 的，当且仅当 $\langle\langle 1\rangle\rangle\Psi = V$。

一个充分且可实现的假设可确保原始博弈的合作获胜区域与该假设下的获胜区域一致，即 $\langle\langle 0\rangle\rangle_\Psi\Phi = \langle\langle 0,1\rangle\rangle\Phi$。然而，它尚不能确保保留双方玩家的所有合作策略选择，而宽容性这一概念则能确保这一点。

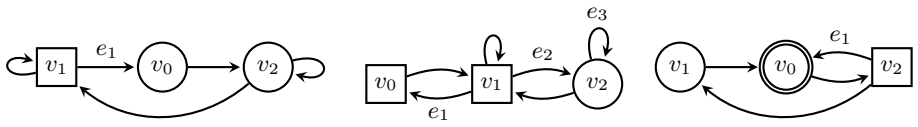

图2： 由 $Player\ 1$ （方形）和 $Player\ 0$ （圆形）顶点构成的双人博弈。

定义 4. 若假设 Ψ 满足 宽容性，则称其对于 (G, Φ) 是宽容的，当且仅当 $\mathcal{L}(\Phi) \subseteq \mathcal{L}(\Psi)$。

宽容性这一概念的提出，源于假设在组合式综合中的预期用途。在两个进程交互这一最简单的场景中，会并行考虑两个综合任务，每个进程对应一个。此时，一个综合任务中生成的假设会作为额外规约，用于另一个综合问题。因此，宽容性至关重要，可避免“跳过”可能的合作解——每个综合任务都需要完整保留双方玩家的所有允许策略选择，以便进行组合式推理。下面的示例将展示这一场景，并说明本文所考虑的假设类别。不过，对基于假设的组合式综合进行一般形式化并非本文的讨论范围。

例 *1.* 考虑图2（中）的非零和双人博弈，其中两个玩家各自有两个不同的规格，分别为 $\Phi_0 = \Diamond\Box\{v_1, v_2\}$ 和 $\Phi_1 = \Diamond\Box\{v_1\}$。现在考虑针对玩家 1 的两个候选假设 $\Psi_0 = \Diamond\Box\neg e_1$ 和 $\Psi_0' =$ （$\Box\Diamond v_1 \implies \Box\Diamond e_2$）。请注意，这两个假设对于 (G, Φ_0) 都是充分且可实现的。然而， Ψ_0' 不允许对局 $\{v_1\}^\omega$，因此并非宽容的；而 Ψ_0 对于 (G, Φ_0) 是宽容的。因此，玩家 1 无法同时满足其目标 Φ_1 和假设 Ψ_0'，即使玩家 0 合作也不行，因为 $\mathcal{L}$ （ Φ_1 ） $\cap \mathcal{L}$ （ Ψ_0' ） $= \emptyset$。然而，在玩家 1 的假设 Ψ_0 和玩家 0 的假设 $\Psi_1 = \Diamond\Box\neg e_3$ 下（若交换双方玩家的顶点，则该假设对于 (G, Φ_1) 是充分且可实现的），他们能够同时满足各自的目标和对自身的假设。因此，他们能够共同满足各自的目标。

注释 *1.* 我们指出，对于例 1， [9] 中的算法会输出 Ψ_0' ，将其作为博弈 (G, Φ_0) 的所需假设，而他们使用的假设形式体系不够丰富，无法刻画假设 Ψ_0。这表明，我们感兴趣的假设类型无法由 [9]中的算法计算得出。

定义 5. 如果假设 Ψ 充分、可实现且宽容，则称其对于(G, Φ) 是充分宽容的（简称 *APA*）。

3.1 定义 1 的讨论

我们首先指出定义1的一些简单但有趣的推论。首先，我们有反单调性，即如果假设 Ψ_1 比假设 Ψ_2 更强（就对局包含而言），并且 π^0 在 Ψ_2下获胜，那么它在 Ψ_1下也获胜。作为这一观察的直接推论，我们还有合取性，即如果π^0 在 Ψ_1 下获胜，并且 π^0 在 Ψ_2下获胜，那么 π^0 在

$\Psi_1 \wedge \Psi_2$下获胜。然而，有趣的是，定义1并不允许析取性，即如果 π^0在 Ψ_1 下获胜，并且 π^0 在 Ψ_2下获胜，那么它不 一定在 $\Psi_1 \vee \Psi_2$下获胜。下面的示例说明了这一观察。

例*2.* 考虑图3中的博弈图，其规范为 $\Phi = \Diamond\Box\{a\}$（要求对局最终只访问顶点 a）。再考虑假设 $\Psi_1 = \neg e_0 \mathcal{U} X e_1$ （边 e_0 第一次被走过时，下一条边应为 e_1）和 $\Psi_2 = e_0 \mathcal{U} X e_2$ （边 e_0 第一次被走过时，下一条边应为 e_2）。注意，只有一个玩家 1策略 π^1 满足任一假设，即永不使用边 e_0 的策略。因此，任何符合 π^1 的对局最终都只会访问顶点 a，因而是获胜的。所以，任意玩家 0策略在任一假设下都能获胜。特别地，考虑只使用边 e_1 的策略 π^0 。对于每个 i，π^0 在假设 Ψ_i 下都能获胜。然而， π^0 在 $\Psi := \Psi_1 \vee \Psi_2 \equiv \mathsf{true}$ 下并不能获胜。为此，注意到任何玩家 1策略都能满足假设 Ψ ，尤其是总是从状态 a 使用 e_0 的策略 $\widetilde{\pi}^1$ 。容易看出， $\widetilde{\pi}^1$ 与 π^0 的组合产生对局 $(abc)^\omega$ ，该对局满足 Ψ ，但不满足 Φ。因此， π^0 在假设 $\Psi := \Psi_1 \vee \Psi_2$ 下不能获胜。

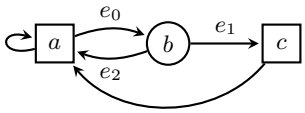

图3： 示例博弈图，说明如例2所述，在假设下获胜并不具有析取性。

此外，我们想指出，定义1与典型的线性时序综合设定略有不同。在该设定中，通常会自然地将满足假设时获胜定义为针对对局，而非策略。因此，我们想简要讨论这种设定，并说明为何对于我们计算的这类特殊假设，它与我们对获胜的定义等价。

我们首先给出定义1的一种基于对局的另一种表述。

定义6。 设 $\mathcal{G} = ((V, V^0, V^1, E)\ \Phi)$ 是一个博弈， Ψ 是关于 V 的线性时序逻辑公式。那么，如果每个对局 $\rho \in \mathcal{L}(\pi^0)$ 要么不满足假设 Ψ ，要么满足规范 Φ，则玩家 0 的策略 π^0 在 $\mathcal{G}$ 中满足假设 Ψ 时获胜。

很容易看出，根据定义6，在假设下获胜的策略 π^0 ，根据定义1也在假设下获胜。然而，反过来一般并不成立，以下示例将对此进行说明。

例*3.* 考虑例2中的同一个博弈，即图3中规范为 $\Phi = \Diamond\Box\{a\}$ 的博弈。再考虑例2中的假设 $\Psi_1 = \neg e_0 \mathcal{U} X e_1$ 。那么，根据前述相同的论证，策略 π^0 仅使用边 e_1，根据定义1，在 Ψ_1 下是获胜的。不过，请注意，对局

$(abc)^\omega$ 符合 π^0 且满足 Ψ_1，但不满足 Φ。因此，根据定义6，π^0 在 Ψ_1 下并非获胜。

有趣的是，本文计算出的假设类别无法涵盖上述类型的示例。直观而言，这是因为这些假设可由玩家1实现，并且由非常局部的模板组合而成。因此，只需限制玩家1的行动，就能强制满足这些假设。这意味着，对于任何符合该假设且 ρ 的对局 π^0，都存在一个满足该假设的策略 π^1，其结果为 ρ。因此，根据定义1，任何在假设下获胜的策略 π^0，根据定义6，也是在假设下获胜的策略。关于这类假设中两个定义等价性的完整证明见附录F，因为证明需要用到后续章节的结果。

我们在本小节最后指出，我们对“在假设下获胜”的表述方式，灵感来自分布式综合。在这种情况下，系统可能并不知道环境代理。我们的定义使我们能够自然地论证：对于玩家0（系统）的策略 π^0，无论任何由玩家1（环境）选择并满足该假设的策略 π^1 为何，它都是获胜策略。

4 计算充分宽容假设（APA）

本节介绍我们用于为 ω-正则博弈计算充分宽容的假设（简称 APA）的算法，其中奇偶博弈是这类博弈的规范表示。为循序渐进地阐述这一主题，我们首先介绍针对较简单获胜条件的算法，即安全（第 4.2 节）、布奇（第 4.3 节）和余布奇（第 4.4 节）；这些算法将作为构件，用于介绍奇偶博弈的算法（第 4.5 节）。我们先介绍一些预备知识。

4.1 预备知识

我们使用以 μ-演算 [18]表示的符号不动点算法来计算获胜区域，并通过简单的后处理步骤生成假设。**集合变换器**。设 $G = (V, V^0, V^1, E)$ 为一个博弈图，$U \subseteq V$ 为顶点的一个子集，$a \in \{0, 1\}$ 为玩家索引。接下来，我们定义两类前驱算子：

$$\mathsf{pre}_G(U) = \{v \in V \mid \exists u \in U.\ (v,u) \in E\} \tag{1}$$

$$\mathsf{cpre}^a_G(U) = \{v \in V^a \mid v \in \mathsf{pre}_G(U)\} \cup \{v \in V^{1-a} \mid \forall (v,u) \in E.\ u \in U\} \tag{2}$$

$$\mathsf{cpre}^{a,0}_G(U) = \mathsf{cpre}^a_G(U) \cup U \tag{3}$$

$$\mathsf{cpre}^{a,i}_G(U) = \mathsf{cpre}^a_G(\mathsf{cpre}^{a,i-1}_G(U)) \cup \mathsf{cpre}^{a,i-1}_G(U) \text{ with } i \geq 1 \tag{4}$$

前驱算子 $\mathsf{pre}_G(U)$ 用于计算至少有一个后继顶点属于 U 的顶点集合。可控前驱算子 $\mathsf{cpre}^a_G(U)$ 和 $\mathsf{cpre}^{a,i}_G(U)$ 用于计算从中玩家 a 能够强制在至多

一步 和 i 步内到达的顶点集合。在下文中，我们介绍吸引子算子 $\mathsf{attr}^a_G(U)$，用于计算从中 玩家 a 能够强制在有限但非零[3] 步内至少访问一次 U 的顶点集合：

$$\mathsf{attr}^a_G(U) = \left(\bigcup_{i\geq 1} \mathsf{cpre}^{a,i}(U)\right)\backslash U \tag{5}$$

根据上下文可以明确判断时，我们会省略这些算子中的下标 G 。

-演算中的不动点算法 μ**-演算**。 μ-演算 [18] 为博弈图 G上的符号算法（即操作顶点集合而非单个顶点的算法）提供了简洁的表示。μ-演算的公式在双人博弈图 G上解释，其语法由以下文法给出

$$\phi \coloneqq p \mid X \mid \phi \cup \phi \mid \phi \cap \phi \mid pre(\phi) \mid \mu X.\phi \mid \nu X.\phi$$

其中， p 遍历 V的子集， X 遍历形式变量集合， pre遍历 {pre、cpre^a、attr^a}中的单调集合变换器， μ 和 ν 分别表示定义为$X \mapsto \phi(X)$的函数的最小不动点和最大不动点。由于 Knaster-Tarski 定理 [5]，操作 $\cup$、$\cap$以及集合变换器 pre 均为单调，因此这些不动点保证存在。公式的（标准）语义从略（见 [18]）。

A μ-微积分公式在 G 上求值得到一个顶点集合，该集合可通过对公式结构进行归纳法计算，其中不动点通过迭代求值。读者可能会注意到，pre 和 cpre 可在关于顶点数量的多项式时间内计算；由于博弈图有限，attr 也可在多项式时间内计算。

4.2 安全博弈

安全博弈是一个满足 $\mathcal{G} = (G, \Phi)$ 且 $\Phi \coloneqq \Box U$ 适用于某个 $U \subseteq V$ 的博弈；若对局始终不离开 U，则该对局满足 Φ 。针对安全博弈的充分宽松假设会禁止玩家 1 在 G 中采取任何会离开合作制胜区域的行动，其中 w.r.t. $Safety(U)$。以下定理对此进行了形式化表述[4]。

定理 1。设 $\mathcal{G} = (G, \Box U)$ 为一个安全博弈， $Z^* = \nu Y.U \cap \boldsymbol{pre}(Y)$ ，且 $S = \left\{ (u, v) \in E \mid \left(u \in V^1 \cap Z^*\right) \wedge \left(v \notin Z^*\right) \right\}$。那么 $Z^* = \langle\langle 0, 1 \rangle\rangle \Box U$ 且 [5]

$$\Psi_{\text{UNSAFE}}(S) \coloneqq \Box \bigwedge\nolimits_{e \in S} \neg e, \tag{6}$$

是博弈 $\mathcal{G}$ 的一个 ***APA***。我们将上述计算 S 的算法记为 $\textsc{UnsafeA}(G, U)$，其运行时间为 $\mathcal{O}(n^2)$，其中 $n = |V|$。

我们将式（6）中的线性时序逻辑公式称为 安全模板，仅使用此模板的假设称为 安全假设。

[3] 在现有文献中，通常 $U \subseteq \mathsf{attr}^a(U)$，即 $\mathsf{attr}^a(U)$ 包含从这些顶点出发， U 经过零步即可访问的顶点。出于一个次要的技术原因，我们将 U 排除在 $\mathsf{attr}^a(U)$ 之外。[4] 所有证明均可在附录中找到。[5] 在线性时序逻辑公式中，我们使用 $e = (u, v)$ 作为 $u \wedge \bigcirc v$ 的语法糖，其中 $\bigcirc$ 是线性时序逻辑的 下一步 算子。边集 $E' = \{e_i\}_{i \in [0;k]}$ 用作原子命题时，是 $\bigvee_{i \in [0;\ k]} e_i$ 的语法糖。

4.3 布奇博弈的活组假设

布奇博弈。布奇博弈是一种博弈 $\mathcal{G} = (G, \Phi)$，其中 $\Phi = \square\lozenge U$ 对某些$U \subseteq V$成立。直观而言，如果一场对局无限次访问顶点集U，那么该对局在布奇博弈中获胜。我们首先回顾一下，合作制胜区域 $\langle\!\langle 0,1 \rangle\!\rangle \square\lozenge U$可以通过双重嵌套符号不动点算法 [10]计算得出

$$\textsc{Büchi}(G, U) := \nu Y.\mu X.\ (U \cap \mathsf{pre}(Y)) \cup (\mathsf{pre}(X)). \tag{7}$$

活跃群组模板。 根据式 (7) 中的标准算法，在最后一次迭代中， Y的最后一次迭代里，不动点变量 X 的第 i次迭代所计算出的集合 X^i，实际上包含了大量信息，可用于构造一个对布奇博弈 $\mathcal{G}$ 非常有用的假设。为此，回想一下， X^i 包含所有这样的顶点：这些顶点有一条边指向能够在至多 $i-1$ 步内到达 U 的顶点（见第 [10, 3.2] 节）。因此，对于 $X^i \setminus X^{i-1}$ 中的所有玩家 1 顶点，我们需要假设玩家 1 总是最终会通过移动到 X^i 来朝 U 取得进展。这可以用所谓的活组模板来形式化。

定义 7。 设 $G = (V, E)$ 为一个博弈图。那么，活组 $H = \{e_j\}_{j \geq 0}$ 是一组边 $e_j = (s_j, t_j)$，其源顶点为 $src(H) := \{s_j\}_{j \geq 0}$。给定一组活组 $H^\ell = \{H_i\}_{i \geq 0}$，我们将活组模板定义为

$$\Psi_{\textsc{live}}(H^\ell) \coloneqq \bigwedge_{i \geq 0} \square\lozenge src(H_i) \implies \square\lozenge H_i. \tag{8}$$

活跃群组模板表示：如果活组源中的某个顶点被无限次访问，那么就应当无限次采取该组中的某条边。我们将使用此模板为 Büchi 博弈给出假设。

注 *2*。 我们注意到，Chatterjee 等人 [8]在其环境假设中使用了活跃边 。活跃边是单元素活跃组，因此表达能力较弱。特别地，对于某些 Büchi 博弈实例，不存在宽容活跃边假设，但存在宽容活跃群组假设[6]。例如，在图1 (c) 中，活跃边假设 $\square\lozenge e_1 \wedge \square\lozenge e_2$ 是充分的，但不是宽容的；而活跃群组假设 $\square\lozenge src(H) \implies \square\lozenge H$ 带有 $H = \{e_1, e_2\}$ 是一个 APA。

在式（7）的不动点计算中，我们可以构造活组$H^\ell = \{H_i\}_{i \geq 0}$，其中每个 H_i 都包含 玩家 1 的所有边，这些边从$X^i \setminus X^{i-1}$ 出发并终止于 X^{i-1}。那么，式（8）中的活组假设准确地体现了这样一种直觉：为了无限次访问 U，如果 $src(H_i)$ 中的顶点被无限次看到，玩家 1 就应该无限次选择 H_i 中的边。不幸的是，结果表明，这个活组假设并不宽容。原因在于，它还限制了玩家 1 在那些无论如何都会朝 U 前进的顶点上的行动。例如，考虑图2（右）中的博弈。这里，通过式（10）的计算来定义活组，会将 e_1 标记为一个活组，

[6] 即使用活跃群组模板的假设

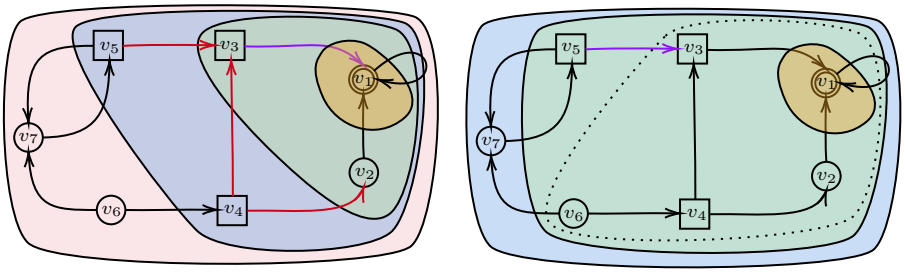

图 4： $\mu X.\, U \cup \mathsf{pre}(X)$（左）和 $\mu X.\, U \cup \mathsf{tpre}(X)$（右）的计算。每个着色区域表示对 X 的一次迭代。右侧的点状区域由 tpre 的 attr 部分添加，因此只有顶点 v_5 可以属于 $\mathit{front}(\{v_1\})$。每组同色边定义一个活跃转移组。

但这样一来，$(v_2v_1v_0)^\omega$ 就会属于 $\mathcal{L}(\Phi)$，却不属于该假设的语言。此处的宽容假设应为 $\Psi =$真。

加速不动点计算。 为了计算宽容活跃群组假设，我们使用一种略作修改的不动点算法，它计算出相同的集合 Z^*，但使我们能够直接从不动点计算中提取宽容假设。为此，我们引入协同前驱算子。

$$\mathsf{tpre}_G(U) = \mathsf{attr}^0_G(U) \cup \mathsf{cpre}^1_G(\mathsf{attr}^0_G(U) \cup U). \tag{9}$$

直观而言，tpre 在不动点计算的每次迭代中，都会添加所有无需玩家0 进行任何协作即可到达 U 的顶点。我们发现一个有趣的现象：将式 (7) 中的内层 pre 算子替换为 tpre，不会改变计算出的集合，只会加快计算速度。下一命题将对此进行形式化表述，图 4 则对此进行了直观展示。

命题 1。 设 $\mathcal{G} = (G, \Box\Diamond U)$ 是一个 $\mathit{Büchi}$ 博弈，且

$$\mathrm{TB\ddot{U}CHI}(G, U) = \nu Y.\mu X.\ (U \cap \mathit{pre}(Y)) \cup (\mathit{tpre}(X)). \tag{10}$$

那么 $\mathsf{TBüchi}(G, U) = \mathsf{Büchi}(G, U) = \langle\!\langle 0, 1 \rangle\!\rangle \Box\Diamond U$。

命题 1 可由式 (7) 的正确性证明推出，所用依据是如下观察：对所有 $U \subseteq V$，均有 $\mu X.\, U \cup \mathsf{pre}(X) = \mu X.\, U \cup \mathsf{tpre}(X)$；这一点已在附录引理 1 中证明。

计算活组假设。 直观而言，算子 tpre_G 计算以下两类顶点集合的并集：(i) 从这些顶点出发，玩家 0 无需玩家 1 配合，经过有限步即可到达 U 的顶点；(ii) 玩家 1 的顶点中，玩家 0 至多只需一次性 得到玩家 1 配合，即可到达 U 的顶点。观察图 4，情况 (i) 用虚线表示，而情况 (ii) 则对应最后添加的玩家 1 顶点（例如 v_5）。因此，我们只需从这些顶点出发，捕捉玩家 1 所需提供的配合

最后添加的顶点，我们称之为 U 在 G 中的前沿，并将其形式化定义如下：

$$front(U) := \mathsf{tpre}_G(U) \setminus \mathsf{attr}^0_G(U). \tag{11}$$

很容易看出，事实上 $front(U) \subseteq V^1$，因为每当 $v \in front(U) \cap V^0$ 时，必然有 $v \in \mathsf{attr}^0_G(U)$ 通过 (10)。

基于前沿而非 X^i 中的所有元素来定义活组，确实能为布奇博弈得到所需的宽容假设。通过注意到，我们还需要用一个简单的安全性假设来确保玩家 1 永远不会离开合作制胜区域，我们得到以下结果。这是本节的主要贡献，其证明见附录。

定理 2。设 $\mathcal{G} = (G = (V, E)\ \Phi = \Box\Diamond U)$ 是一个 $B\ddot{u}chi$ 博弈，具有 $Z^* = \mathbf{T}$ 布奇(G, U)，且 $H^\ell = \{H_i\}_{i\geq 0}$ 满足

$$\emptyset \neq H_i := (front(X^i) \times (X^{i+1} \setminus front(X^i))) \cap E, \tag{12}$$

其中，X^i 是在对 X进行计算的第 i次迭代中，以及在 $\mathrm{TB\ddot{U}CHI}$ 中对 Y 进行计算的最后一次迭代中计算得到的集合。于是，$\Psi = \Psi_{\mathrm{UNSAFE}}(S) \wedge \Psi_{\mathrm{LIVE}}(H^\ell)$ 是关于 $\mathcal{G}$的一个 *APA*，其中 $S = \mathrm{UNSAFEA}(G, U)$。我们用$\mathrm{LIVEA}(G, U)$ 表示构造上述活组 H^ℓ 的算法，其运行时间为 $\mathcal{O}(n^3)$，其中 $n = |V|$。

事实上，对于布奇博弈中 APA 的计算，存在一种更快的算法，其运行时间与图的规模呈线性关系；我们在附录 C.1 中介绍该算法。我们选择在此介绍基于 μ 演算的算法，因为它能让我们更深入地了解活组的本质。

4.4 余布奇博弈中的共活假设

余布奇博弈是布奇博弈的对偶，其中获胜对局只能有限次访问指定的顶点集合。形式上，余布奇博弈是一个元组 $\mathcal{G} = (G, \Phi)$，其中对于某个 $U \subseteq V$，有 $\Phi = \Diamond\Box U$。计算合作制胜区域的标准符号算法如下：

$$\mathrm{CoB\ddot{U}CHI}(G, U) := \mu X.\nu Y.\ (U \cap \mathsf{pre}(Y)) \cup (\mathsf{pre}(X)). \tag{13}$$

和之前一样，在评估 (13) 时，通过第 i 次计算 X 得到的集合 X^i，包含了构造假设所需的重要信息。直观地说，X^1 恰好给出了这样一组顶点：在 {style id='6'>玩家 </style}1 的配合下，对局可以从这些顶点出发一直留在 U 中；我们希望某个假设能够体现这样一个事实：我们不希望玩家 1 无限次地远离 X^1。这一观察可以自然地用所谓的共活性模板来描述。

定义 8。设 $G = (V, E)$ 为一个博弈图，$D \subseteq V \times V$为一组边。那么，关于 G、相对于 D 的共活性模板 由以下线性时序逻辑公式定义

$$\Psi_{\mathrm{COLIVE}}(D) \coloneqq \Diamond\Box \bigwedge_{e\in D} \neg e. \tag{14}$$

采用共活性模板的假设称为共活性假设。由此，我们可以陈述本节的主要结果。

定理 3. 令 $\mathcal{G} = (G = (V, E)\ \Diamond\Box U)$， $Z^* = \mathrm{CoB\ddot{u}CHI}(G, U)$ 和

$$D = \Big(\big[(X^1 \cap V^1) \times (Z^* \setminus X^1)\big] \ \cup \big[\textstyle\bigcup_{i>1}(X^i \cap V^1) \times (Z^* \setminus X^{i-1})\big]\Big) \cap E, \quad (15)$$

其中 X^i 是在不动点变量 X 的第 i 次迭代中于$\mathrm{CoB\ddot{u}CHI}$中计算得到的集合。则 $\Psi = \Psi_{\mathrm{UNSAFE}}(S) \wedge \Psi_{\mathrm{COLIVE}}(D)$ 是 $\mathcal{G}$ 的一个 *APA*，其中 $S = \textsc{UnsafeA}(G, U)$。我们用 $\textsc{CoLiveA}(G, U)$ 表示构造上述共活边 D 的算法，其运行时间为 $\mathcal{O}(n^3)$，其中 $n = |V|$。

我们注意到， X_1 是 U 的子集，并且如果一场对局到达 X^1，玩家 0 和玩家 1 可以合作使对局一直留在 X^1 中。为此，我们通过 (15) 中 D 的定义确保玩家 1 只能有限次离开 X^1 。此外，借助 D 中的其他共活边，我们确保玩家 1 只能有限次离开 X^1 ；因此，如果玩家 0 执行其策略到达 X^1 并留在那里，那么这场对局将获胜。该假设的宽容性源于以下观察：如果共活性遭到违反，那么玩家 1 会无限次走共活边，因此会无限次离开 X^1 ，这意味着会无限次离开 U 。形式化的定理 3 证明请参见附录 D。

在式（13）的语境下，该集合

$$D = \left(\big[(X^1 \cap V^1) \times (Z^* \setminus X^1)\big] \ \cup \left[\bigcup_{i>1}(X^i \cap V^1) \times (Z^* \setminus X^{i-1})\right]\right) \cap E, \quad (16)$$

得到了所需的共活假设。我们指出，这定义了 co-Büchi 博弈的足够宽容假设。然而，利用第 4.3 节中的观察，我们也可以等价地使用加速不动点算法：将式（13）中作用于 X 的 pre 算子替换为 tpre 算子即可得到该算法。这同样只会加速计算，正如下述命题所形式化并在图 5 中所示。

命题 2。 设 $\mathcal{G} = (G, \Diamond\Box U)$ 是一个 *co-Büchi* 博弈，且

$$\mathrm{TCoB\ddot{u}CHI}(G, U) = \mu X.\nu Y.\ (U \cap \mathit{pre}(Y)) \cup (\mathit{tpre}(X)). \quad (17)$$

则 $\mathrm{TCoB\ddot{u}chi}(G, U) = \mathrm{CoB\ddot{u}chi}(G, U) = \langle\!\langle 0, 1 \rangle\!\rangle \Diamond\Box U$。

使用加速不动点算法时，在构造假设时，我们同样可以只关注 X^i 前沿中的玩家 1 顶点。在第 4.3 节中，对于布奇博弈 $B\ddot{u}chi(U)$，我们引入了活组，以便在玩家 1 有能力时，将对局引向 U。但对于 co-Büchi 博弈，我们需要限制玩家 1，使其不能离开区域 U_{good} ，因为对局会停留在 U 中。这只要求在 X^is 的前沿上施加共活性假设，因为加入 X^{i+1} 的任何其他玩家 1 顶点都位于 tpre 的 attr^0 部分，因此无论如何都无法离开 U_{good} 。由此，我们得到本节的主要结果，其证明见附录 D.1。

定理 4。 设 $\mathcal{G} = (G = (V, E)\ \Phi = \Diamond\Box U)$ 为一个 *co-Büchi* 博弈，其 $Z^* = \mathrm{TCoBüChi}(G, U)$ 且

$$D = \left(\begin{array}{r} (X^1 \cap V^1) \times (Z^* \setminus X^1) \cup \\ \bigcup_{i>1} \mathit{front}(X^i) \times (\mathit{front}(X^i) \cup Z^* \setminus X^{i+1}) \end{array} \right) \cap E, \tag{18}$$

其中，X^i 是通过不动点变量 X 的第 i 次迭代计算得到的集合。那么，$\Psi = \Psi_{\mathrm{UNSAFE}}(S) \wedge \Psi_{\mathrm{COLIVE}}(D)$ 是关于 $\mathcal{G}$ 的充分宽松假设，其中 $S = \mathrm{UnsafeA}(G、U)$。此外，$D$ 可以在时间 $\mathcal{O}(n^3)$ 内构造，其中 n 是顶点数。

事实上，对于余布奇博弈中充分宽松假设的计算，再次存在一种更快的算法，其运行时间与图的规模呈线性关系；我们将在附录 D.2 中介绍该算法。出于与布奇博弈相同的原因，我们选择介绍这一版本。

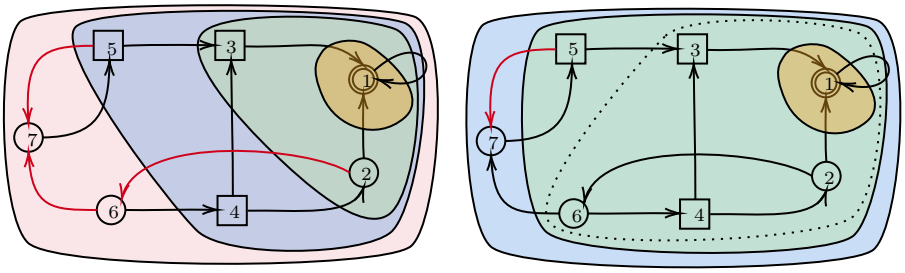

图 5： 左图描述了使用 pre 进行的余布奇计算，右图则使用 tpre。每个着色区域描述了 X 在每次迭代后的增长情况，右侧的点状区域由 attr 部分的 tpre添加。两种情况下，red 色的边表示共活边。同样，使用 pre 进行计算会给出关于 $Player\ 0$ 个顶点的假设，而使用 tpre 则只会给出关于 $Player\ 1$ 的顶点的假设。

4.5 奇偶博弈的 APA 假设

奇偶博弈。 令 $G = (V, V^0, V^1, E)$ 为一个博弈图，且 $C = \{C_0, \ldots, C_k\}$ 为顶点的一个子集集合，它们构成 V 的一个划分。那么博弈$\mathcal{G} = (G, \Phi)$ 称为一个奇偶博弈，如果

$$\Phi = \mathit{Parity}(C) := \textstyle\bigvee_{i \in_{\mathrm{odd}} [0;k]} \Box\Diamond C_i \implies \bigvee_{j \in_{\mathrm{even}} [i+1;k]} \Box\Diamond C_j. \tag{19}$$

集合 C 称为优先级集，而集合 C_i 中的顶点 v（其中 $i \in [1; k]$）称为优先级 i。如果无限对局 ρ 中无限次出现的最高优先级是偶数，则该对局对 $\Phi =$ 奇偶(C) 是获胜的。

条件活跃组模板。正如前面几节所述，对于获胜条件简单、要求无限次或仅有限次访问某个固定边集的博弈，只需一个假设（与一个简单的安全性假设合取）就足以刻画充分宽松假设（APA），因为获胜方式只有一种。然而，一般而言，奇偶博弈通常有多种获胜方式：例如，在优先级为 $\{0,1,2\}$ 的奇偶博弈中，如果对局满足以下任一条件，就能获胜：（i）仅无限次看到优先级为 0 的顶点；或（ii）无限次看到优先级 1，同时也无限次看到优先级 2。直观而言，获胜方式（i）需要使用第 4.4 节中的共活性假设。然而，获胜方式（ii）实际上需要使用第 4.3 节讨论的活跃组假设，并根据某些优先级为 1 的状态是否确实被无限次访问而进行条件化。通过将活跃组模板推广为条件活跃组模板，即可对此进行形式化。

定义 9。设 $G=(V,V^0,V^1,E)$ 是一个博弈图。那么，G 上的一个条件活跃组是一个二元组(R,H^ℓ)，其中 $R\subseteq V$，且 H^ℓ 是一个活组。给定一个条件活跃组集合 $\mathcal{H}^\ell$，我们将条件活跃组模板 定义为如下线性时序逻辑公式

$$\Psi_{\text{COND}}(\mathcal{H}^\ell) \coloneqq \bigwedge_{(R,H^\ell)\in\mathcal{H}^\ell} \left(\square\Diamond R \implies \Psi_{\text{LIVE}}(H^\ell)\right). \tag{20}$$

同样，使用条件活跃组模板的假设称为条件活跃组假设。将活跃组假设推广为条件活跃组假设后，我们实际上已具备定义奇偶博弈的 APA 所需的全部要素，其形式为以下各项的合取：

$$\Psi = \Psi_{\text{UNSAFE}}(S) \wedge \Psi_{\text{COLIVE}}(D) \wedge \Psi_{\text{COND}}(\mathcal{H}^\ell) \tag{21}$$

安全性假设、共活假设和条件活跃组假设。直观来说，我们使用：(i) 安全性假设来防止玩家 1 离开合作获胜区域；(ii) 对于每个获胜选项，使用一个共活假设，要求某个特定奇数 优先级只出现有限次；(iii) 对于每个获胜选项，使用一个条件活跃组假设：如果某些奇数 优先级已出现无限次，则要求某个偶数 优先级也出现无限次。本节余下部分将给出一个算法（算法 1），用于计算实际的安全性集合、共活集合和条件活跃组集合 S、D 和 $\mathcal{H}^\ell$，并证明所得假设 Ψ（如式 (21) 所示）确实是奇偶博弈 $\mathcal{G}$ 的 APA。

计算充分宽松假设。将 Ψ 转换为 APA 所需的不安全集合、共活集合和条件活跃组集合 S、D 和 $\mathcal{H}^\ell$ 的计算，在算法 1 中进行了形式化。算法 1 使用标准不动点算法 Parity$(G,\ C)$ [12]，以计算奇偶博弈 $\mathcal{G}$ 的协作获胜区域，定义如下：

$$\textsc{Parity}(G,C) \coloneqq \tau X_d \cdots \nu X_2\ \mu X_1\ \nu X_0.\bigcup_{i\in[0;d]}(C_i \cap \mathsf{pre}(X_i)), \tag{22}$$

其中，当 d 为偶数时，τ 为 ν，否则为 μ。此外，算法 1 以迭代方式调用 UnsafeA（定理 1）、LiveA（定理 2）和 CoLiveA（定理 3）算法，以计算 Safety、活组和共活假设。

算法 1 ParityAssumption

输入： $G = (V, E)$，$C\colon V \to \{0, 1, \ldots\}$
输出： Ψ
1: $Z^* \leftarrow$ 奇偶(G, C)
2: $S \leftarrow$ UnsafeA(G, Z^*)
3: $G \leftarrow G|_{Z^*}$，$C \leftarrow C|_{Z^*}$
4: $(D, \mathcal{H}^\ell) \leftarrow$ComputeSets$((G, C)\ \emptyset, \emptyset)$
5: **返回** S $, D, \mathcal{H}^\ell$
6: **过程** ComputeSets$((G, C)\ D, \mathcal{H}^\ell)$
7: $d \leftarrow$ 最大$\{i \mid C_i \neq \emptyset\}$
8: **如果** d 是奇数，**则**
9: $W_{\neg d} \leftarrow$ 奇偶$(G|_{V \setminus C_d}, C)$
10: $D \leftarrow D \cup$ CoLiveA$(G, W_{\neg d})$
11: **else**
12: $W_d \leftarrow$ 布奇(G, C_d)，$W_{\neg d} \leftarrow V \setminus W_d$
13: **对于所有**奇数 $i \in [0; d]$，**执行**
14: $\mathcal{H}^\ell \leftarrow \mathcal{H}^\ell \cup (W_d \cap C_i,$ LiveA$(G|_{W_d}, C_{i+1} \cup C_{i+3} \cdots \cup C_d))$
15: **如果** $d > 0$ **那么**
16: $G \leftarrow G|_{W_{\neg d}}$ 、$C_0 \leftarrow C_0 \cup C_d$、$C_d \leftarrow \emptyset$
17: ComputeSets$((G, C)\ D, \mathcal{H}^\ell)$
18: **else**
19: **返回**$(D, \mathcal{H}^\ell)$

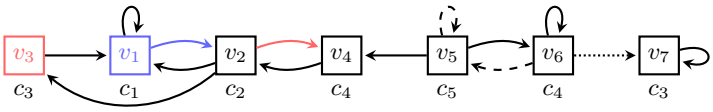

图 6： 一个奇偶博弈，其中优先级为 i 的顶点标记为 c_i。点线边是不安全边，虚线边是共活边，并且每组颜色相同的顶点-边对都构成一个条件活跃组。

此外，$G|_U := (U, U^0, U^1, E')$ 满足 $U^0 := V^0 \cap U$、$U^1 := V^1 \cap U$，并且 $E' := E \cap (U \times U)$ 表示将博弈图 $G := (V, V^0, V^1, E)$限制到其顶点 $U \subseteq V$的子集上。此外，$C|_U$ 表示将优先级集 C 从 V限制到 $U \subseteq V$。

我们通过图 6 中的示例说明算法 1 的步骤。在第 1 行，我们首先计算整个博弈的合作制胜区域 Z^*，发现从顶点 v_7 出发，即使有玩家1 的配合，也无法满足奇偶条件，即 $Z^* = \{v_1, \ldots, v_6\}$。因此，我们将从 v_6 到 v_7 的边标记为安全性假设边，将博弈限制在 $G = G|_{Z^*}$，并在新博弈上运行 ComputeSets。

在新的受限博弈 G 中，最高优先级为 $d = 5$，它是奇数，因此我们执行第 9–10 行。现在，只有当一局对局最终不再看到 v_5 时，这局对局才会获胜。因此，在步骤 9 中，我们从受限图 $G|_{V \setminus C_5}$ 中找出区域 $W_{\neg 5} = \{v_1, \ldots, v_4, v_6\}$（仅包含优先级为 $C(v_i) < 5$) 的节点 v_i），在该区域内，我们可以在不看到 v_5的情况下满足奇偶条件。接着，我们在第 10 行执行 CoLiveA$(G, W_{\neg 5})$，确保不会离开 $W_{\neg 5}$ 去访问 v_5，并在博弈G 中无限次如此。这会对边 (v_5, v_5) 和 (v_6, v_5) 加上一个共存性假设。

一旦我们限制对局不再无限次访问 v_5，就只需专注于在 $W_{\neg 5}$内不访问 v_5 的情况下满足奇偶条件。这一观察使我们能够进一步将计算限制在第 16 行的博弈 $\mathcal{G} = \mathcal{G}|_{W_{\neg 5}}$ 中，并将优先级更新为仅取值于 0 到 4。在我们的示例中，这一步不会改变任何内容。然后，我们在这个博弈上重新执行 ComputeSets。

在受限图中，最高优先级为 4，且为偶数，因此我们执行第 12–14 行。在这个博弈中，一种获胜方式是无限次访问 C_4，因此我们在第 12 行计算相应的合作制胜区域 W_4。在我们的示例中，$W_4 = W_{\neg 5} = \{v_1, \ldots, v_4, v_6\}$。现在，为了确保从那些能够通过合作看到 4 的顶点出发时确实能够获胜，我们必须确保每当一个较低的奇优先级顶点被无限次访问时，也会有一个更高的优先级被访问。这可以通过第 14 行计算出的条件活跃组公平性来保证。对于每个奇优先级 $i < 4$（即，对于$i = 1$ 和 $i = 3$），我们必须确保 2 或 4（若 $i = 1$）或 4（若$i = 3$）会被无限次访问。由此得到的活组 $\mathcal{H}_i^\ell = (R_i,\ H_i^\ell)$ 收集了 W_4 中优先级为 i 的所有顶点，以及所有允许看到偶数优先级 j 的活组，其中 $i < j \leq 4$ 位于 H_i^ℓ中；后者通过用于计算活组的定点算法 LiveA 得出。图 6 展示了针对$i = 1$（蓝色）和 $i = 3$（红色）得到的活组，它们分别为 $(\{v_1\},\ \{(v_1,\ v_2)\})$ 和 $(\{v_3\},\ \{(v_2,\ v_4)\},\ \{(v_1,\ v_2)\})$。

此时我们有 $W_{\neg 4} = \emptyset$。由此，第 16 行计算出的博弈图变为空，算法在对 C 中的所有优先级进行迭代移除后最终终止：对于优先级 3、2 和 1，ComputeSets 已运行完毕（没有进行任何计算，因为 $\mathcal{G}$ 为空）。在另一种博弈图中，如果博弈图还有其他部分不包含在 W_4 中，且从这些部分出发可以通过无限次看到优先级 2 赢得博弈，那么上述针对优先级 5 和 4 的推理也可以应用于较低优先级。所述算法正确性的核心见解是，所有计算出的假设都可以合取起来，从而为原始奇偶博弈得到一个 APA。

主要结果。有了算法 1，我们现在可以陈述本节，尤其是全文的主要结果；该结果已在附录E中证明。

定理 *5*。设 $\mathcal{G} = (G = (V, E), \mathit{Parity}(C))$ 是一个奇偶博弈，且 $(S, D, \mathcal{H}^\ell) = \mathit{ParityAssumption}(G, C)$。那么，$\Psi = \Psi_{\text{UNSAFE}}(S) \wedge \Psi_{\text{COLIVE}}(D) \wedge \Psi_{\text{COND}}(\mathcal{H}^\ell)$ 是 $\mathcal{G}$ 的一个 *APA*。此外，算法 *1* 的运行时间为 $\mathcal{O}(n^4)$，其中 $n = |V|$。

5 实验评估

我们开发了一款基于 C++ 的原型工具 SImPA[7]，用于计算适用于布奇、余布奇和奇偶博弈的**充分**、**可实现**且**宽松**的**假设**。我们首先在第 5.1 节将 SImPA 与最接近的相关工具 GIST [9]进行比较，然后在第 5.2 节展示 SImPA 如何为著名的双客户端仲裁器综合问题 [22]提供简洁且有意义的假设。

[7] 代码仓库网址：https://gitlab.mpi-sws.org/kmallik/simpa

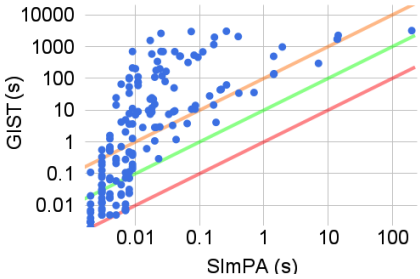

图7： SImPA 与 GIST 的运行时间（秒，采用对数刻度）

	SImPA	GIST
平均时间	64.8s	1079.0s
未超时平均时间	64.8s	209.2s
超时次数（1小时）	0(0%)	59(26%)
无假设生成	0(0%)	20(9%)
更快	230(100%)	0(0%)

表 1： 实验结果汇总

5.1 性能评估

我们将工具的效果与最接近的相关工具 GIST [9], 的重新实现版本进行比较。该工具已无法再从作者[8]处获取。GIST 最初只计算能使某个特定初始顶点对玩家 0 获胜的假设。不过，在实验中，我们运行 GIST，直到某个合作获胜顶点不再是获胜顶点为止。GIST 从最大假设开始，并不断缩小假设，直到某个固定初始顶点不再获胜。由于修改后的终止条件可以更早满足，我们对 GIST 的改动使其运行得更快。我们的工具不依赖任何固定初始顶点，而依赖初始顶点会使 GIST 运行得更慢，因此这一改动使比较更加公平。

我们在从 SYNTCOMP 基准测试套件 [2] 收集的一组奇偶博弈上，对比了 SImPA 和 GIST 计算出的假设在性能和质量方面的表现。使用 SImPA 和 GIST 计算假设时，我们将超时设为一小时。所有实验均在一台配备 Intel(R) Core(TM) i5-10600T CPU @ 2.40GHz 和 32 GiB RAM 的计算机上进行。

实验结果的所有详细信息见附录中的表 G，摘要见表 1。此外，图 7 展示了一个散点图，其中每个基准测试实例都以一个点表示，X 坐标和 Y 坐标分别表示 SImPA 和 GIST 的运行时间（秒）。我们可以看到，在每个实例中，SImPA 的计算速度都远快于 GIST（所有点都位于下方红线之上）——多数情况下快一个数量级（位于中间绿线之上），许多情况下甚至快两个数量级（位于上方橙线之上）。

此外，在某些实验中，GIST 无法计算出充分假设（按照定义 2 的含义），而我们的算法却能成功计算出 APA（见表 1 中标记为“未生成假设”的行，以及表 G 中计算时间旁标有“*”的项）。这并不令人意外，因为 GIST 使用的假设类别仅包括不安全边和活边（即单元素活跃组），表达能力不足以为所有奇偶博弈提供充分

[8] 论文中提供的链接已失效，作者告知我们该实现目前不可用。

假设（见图 1(b)，其中给出了一个简单示例：不存在能够用活边表示的充分假设）。此外，我们注意到，在 GIST 计算出的假设确实是 APA 的所有情形中，SImPA 计算出相同假设的速度都快了几个数量级。

5.2 双客户端仲裁器示例

我们考虑 [22] 中的双客户端仲裁器示例。在此示例中，客户端 $i \in \{1,2\}$（玩家 1）可以通过将输入变量 r_i 设置为 true 或 false，来请求或释放共享资源；仲裁器（玩家 0）可以将输出变量g_i 设置为 true 或 false，从而向客户端 i 授予或撤销共享资源。此示例的博弈图作为规范的一部分隐式给出（因为这是一个 GR(1) 综合问题，详情请见 [22]）。我们在图 8 中示意性地描绘了该博弈图的相关部分。在这里，矩形和圆形分别表示玩家 1 和 玩家 0 顶点，双线顶点的优先级为 2（即布奇顶点），而所有其他顶点的优先级均为 1。玩家 0 状态的标签表示请求位和授权位的当前状态，此外还记录请求当前是否处于待处理状态，即 $F_i = g_i \,\mathsf{S}\, (r_i \wedge g_i)$，其中 S 表示线性时序逻辑运算符“自从”。玩家 1 顶点的标签还会记录玩家 0 最近选择的动作。我们可以看到，所有没有待处理请求的顶点的优先级均为 2。众所周知，如果客户端（玩家 1）的动作不受约束，那么在此博弈中玩家 0 不存在获胜策略。

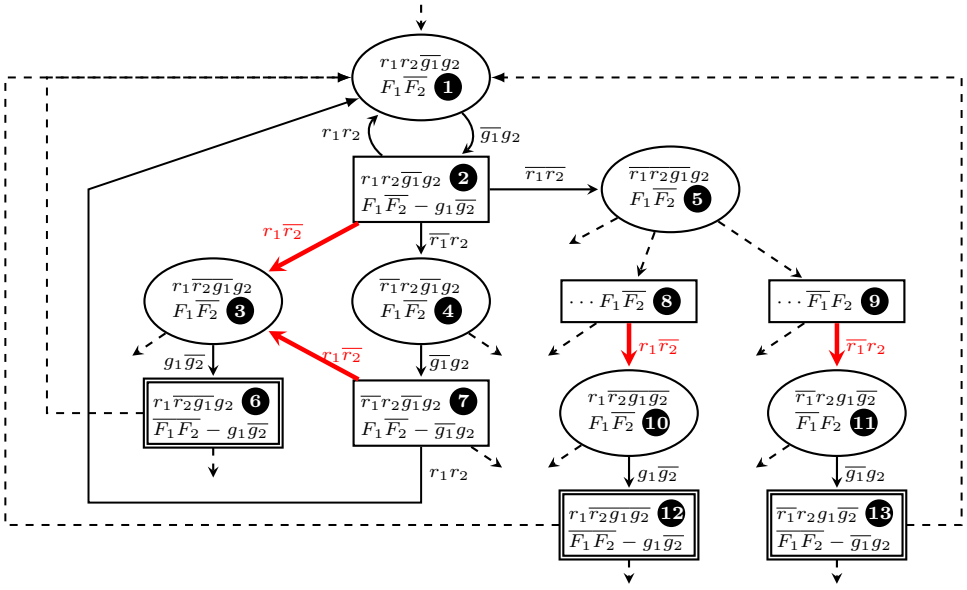

图 8： 2 客户端仲裁器博弈图相关部分的示意图。

在此示例上运行 SImPA 得到的只有活组假设（因为这是一个布奇博弈，且所有顶点在合作情况下都能获胜），这些假设的计算结果为

0.01 秒。图 8 中用粗红箭头示意了一个活组的边。可以看到，这个活组确保对局最终会转移到这样的顶点：玩家 0 能够强制访问一个布奇顶点。 [22], 为使综合问题可实现，用于限制客户端行为的假设为

$$\tilde{\Psi} = \bigwedge_i (\overline{r_i} \wedge \square\big((r_i \neq g_i) \Rightarrow (r_i = \bigcirc r_i)\big) \wedge \square\big((r_i \wedge g_i) \Rightarrow \Diamond \overline{r_i}\big)\big).$$

我们可以看到，我们的活跃组假设与之相似，但限制更少。例如，当我们持续看到标签为 $r_1 g_1$ 和 $r_2 g_2$ 的状态时（例如，在图 8 中循环经过状态 和 ），我们会强制要求最终必须采取标签为$r_1 r_2$ 的边。另一方面，$\tilde{\Psi}$ 中的第二个和第三个条件（分别由 $r_1 g_1$ 和$r_2 g_2$ 触发）强制规定，从状态 出发，除标签为$r_1 r_2$的边之外，不允许有其他出边；这就严格得多了。

我们也在这个示例上运行了 GIST。对于不受限制的初始条件，它用 6.44 秒计算出活边假设，比 SImPA 慢两个数量级。此外，为了让 无限次出现，GIST 返回了活跃边 和 。这个假设并不宽松，因为存在不无限次使用这两条边中任意一条的获胜对局。事实证明，此示例的 APA 不可避免地需要活跃组——GIST 算出的单元素活边并不足够。

致谢 S. P.Nayak 和 A.-K. Schmuck 获得德国研究联合会项目 389792660 TRR 248–CPEC 的部分资助。A. Anand 和 A.-K. Schmuck 获得德国研究联合会项目 SCHM 3541/1-1 的部分资助。K. Mallik 获得欧洲研究委员会项目 ERC-2020-AdG 101020093 的资助。

参考文献

1. 自动机、逻辑与无限博弈。施普林格，柏林，海德堡，10.1007/ 3-540-36387-4 2. 反应式综合竞赛。http://www.syntcomp.org 3. Baier, C., Katoen, J.P.：《模型检测原理》。麻省理工学院出版社 (2008) 4. Brenguier, R., Raskin, J.F., Sankur, O.：假设可容许综合。《Acta Informatica》(2017) 5. Bronisław Knaster, A.T.：《关于集合函数的一个定理》。《波兰数学会年刊》**6** (1928) 6. Cavezza, D.G., Alrajeh, D., György, A.：对可实现规格进行最小假设细化。载于：《软件工程中的形式化方法》(2020) 7. Chatterjee, K., Henzinger, T.A.：假设保证综合。载于：TACAS (2007) 8. Chatterjee, K., Henzinger, T.A., Jobstmann, B.：综合的环境假设。载于：CONCUR (2008) 9. Chatterjee, K., Henzinger, T.A., Jobstmann, B., Radhakrishna, A.：GIST：概率博弈求解器。载于：CAV (2010) 10. Chatterjee, K., Henzinger, T.A., Piterman, N.：布奇博弈算法 (2008)，https://arxiv.org/abs/0805 2620 11 . . Chatterjee, K., Horn, F., Löding, C.：义务博弈。载于：并发理论国际会议。第 284–296 页。施普林格 (2010) 12. Emerson, E., Jutla, C.：树自动机、 μ-演算与确定性。载于：FOCS (1991) 13. Finkbeiner, B., Metzger, N., Moses, Y.：信息流引导综合。载于：第 34 届计算机辅助验证国际会议论文集 (CAV 22) (2022) 14. Fisman, D., Kupferman, O., Lustig, Y.：理性综合。载于：TACAS (2010) 15. Gaaloul, K., Menghi, C., Nejati, S., Briand, L., Parache, Y.I.：结合遗传编程与模型检测来生成环境假设。《TSE》(2021) 16. Gaaloul, K., Menghi, C., Nejati, S., Briand, L.C., Wolfe, D.：使用机器学习为软件组件挖掘假设。载于：ESEC/FSE (2020) 17. King, V., Kupferman, O., Vardi, M.Y.：奇偶词自动机的复杂度。载于：《软件科学与计算结构基础》。第 276–286 页。施普林格，柏林，海德堡 (2001) 18. Kozen, D.：关于命题 μ-演算的结果。载于：ICALP。施普林格 (1982) 19. Majumdar, R., Mallik, K., Schmuck, A.K., Zufferey, D.：假设保证分布式综合。《IEEE TCAD》(2020) 20. Majumdar, R., Piterman, N., Schmuck, A.K.：环境友好的 GR(1) 综合。载于：系统构造与分析工具和算法国际会议。第 229–246 页。施普林格 (2019) 21. Maoz, S., Ringert, J.O., Shalom, R.：GR(1) 规格的符号化修复。载于：ICSE (2019) 22. Piterman, N., Pnueli, A., Sa'ar, Y.：reactive(1)设计的综合。载于：第 7 届验证、模型检测与抽象解释国际会议论文集。第 364–380 页。VMCAI'06，施普林格出版社，柏林，海德堡 (2006) 23. Schmelter, D., Greenyer, J., Holtmann, J.：迈向可学习、可实现的基于场景的形式化需求规格。载于：REW (2017)

安全博弈的 APA 假设

为方便读者，我们在此重述定理 1。

定理 1。 设 $\mathcal{G} = (G, \Box U)$ 为一个安全博弈， $Z^* = \nu Y.U \cap \mathit{pre}(Y)$，且$S = \{(u, v) \in E \mid (u \in V^1 \cap Z^*) \wedge (v \notin Z^*)\}$。则 $Z^* = \langle\!\langle 0, 1 \rangle\!\rangle \Box U$ 且 [9]

$$\Psi_{\text{UNSAFE}}(S) := \Box \bigwedge_{e \in S} \neg e, \tag{6}$$

是该博弈 $\mathcal{G}$ 的一个 *APA*。我们将$\textsc{UnsafeA}(G, U)$ 记为上述计算 S 的算法，其运行时间为 $\mathcal{O}(n^2)$，其中 $n = |V|$。

证明。 关于 $Z^* = \langle\!\langle 0, 1 \rangle\!\rangle \Box U$ 的证明，请参阅 [1, 第 2] 章。下面我们将通过证明充分性、可实现性和宽容性，说明 $\Psi_{\text{UNSAFE}}(S)$ 是安全博弈的 APA 假设。

▶ **(ii)** 可实现性：很容易看出，如果玩家 1 永远不选择 S 中的边，那么 $\Psi_{\text{UNSAFE}}(S)$ 是可由他实现的。
▶ **(ii)** 充分性：为证明该假设的充分性，考虑玩家 0 的策略 π^0：在顶点 $v \in Z^*$，选择使对局留在 Z^* 中的转移；在其他顶点，则任意选择。该策略定义良好，因为如果在 $v \in Z^*$ 处不存在这样的转移，根据定义， v 就不会在 Z^* 中。

令 $v_0 \in Z^* = \langle\!\langle 0, 1 \rangle\!\rangle \Box U$。令 π^1 为玩家 1 的任意策略，使得 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$，且令 $\rho = v_0 v_1 \ldots$ 为任意 $\pi^0\pi^1$-对局。则 $\rho \in \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$。

现在，假设 $\rho \notin \mathcal{L}(\Box U)$，即对于某个 i， $v_i \notin U$。不失一般性，假设 i 是满足条件的最小索引，也就是说，对于所有 $j < i$， $v_j \in U$。那么根据 S 的定义，$(v_{i-1}, v_i) \in S$，这就产生了矛盾。因此， $\rho \in \mathcal{L}(\Box U)$。
▶ **(ii)** 宽容性：现在来看宽容性，令 $\rho \in \mathcal{L}(\Box U)$。假设 $\rho \notin \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$。那么，在 ρ 中会经过某条边 $(v, v') \in S$。到达 v' 后， ρ 仍满足安全条件。因此，根据命题 1， $v' \in Z^*$，但这样一来 $(v, v') \notin S$，这就产生了矛盾。因此， $\rho \in \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$。

▶ 复杂度分析。不动点计算需要耗时 $\mathcal{O}(n^2)$，随后计算 S 还需要耗时 $\mathcal{O}(n^2)$。

B tpre 加速 μ-不动点计算

在第 4.3 节中，我们提到，命题 1 可由式 (7) 的正确性证明（见 [10]）以及以下引理推出，我们在此给出证明。

引理 1。对于 $U \subseteq V$， $\mu X.\, U \cup \mathit{pre}(X) = \mu X.\, U \cup \mathit{tpre}(X)$。

证明。 令 $\mu X.\, U \cup \text{pre}(X)$ 给出 X 的如下计算，我们将其称为*pre* 计算，

[9] 我们在 LTL 公式中将 $e = (u, v)$ 用作 $u \wedge \bigcirc v$ 的语法糖，其中 $\bigcirc$ 是线性时序逻辑的下一步算子。边集 $E' = \{e_i\}_{i \in [0;k]}$用作原子命题时，是 $\bigvee_{i \in [0;\, k]} e_i$ 的语法糖。

$$\begin{array}{ccccccc} X_0 & \subsetneq & X_1 & \subsetneq & \cdots & = & X_{k+1} \\ \| & & \| & & & & \| \\ \emptyset & & U \cup \mathsf{pre}(X_0) & & \cdots & & X_k \cup \mathsf{pre}(X_k) \end{array}$$

类似地，令 $\mu X.\ U \cup \mathsf{tpre}(X)$ 给出 X 的如下计算，我们将其称为 *tpre*计算，

$$\begin{array}{ccccccc} X_0^{\mathsf{t}} & \subsetneq & X_1^{\mathsf{t}} & \subsetneq & \cdots & = & X_{l+1}^{\mathsf{t}} \\ \| & & \| & & & & \| \\ \emptyset & & U \cup \mathsf{tpre}(X_0^{\mathsf{t}}) & & \cdots & & X_l^{\mathsf{t}} \cup \mathsf{tpre}(X_l^{\mathsf{t}}) \end{array}$$

我们首先注意到，集合的 tpre 包含该集合的 pre。

命题。对于 $U \subset V$，$\mathsf{pre}(U) \subseteq \mathsf{tpre}(U)$。

证明。 根据 attr 的定义，我们知道 $\mathsf{cpre}^0(U) \subseteq \mathsf{attr}^0(U)$；根据 cpre^0 的单调性，$\mathsf{cpre}^1(U) \subseteq \mathsf{cpre}^1(\mathsf{attr}^0(U) \cup U)$。于是，$\mathsf{pre}(U) = \mathsf{cpre}^0(U) \cup \mathsf{cpre}^1(U) \subseteq \mathsf{attr}^0(U) \cup \mathsf{cpre}^1(\mathsf{attr}^0(U) \cup U) = \mathsf{tpre}(U)$。◁

现在我们证明，在 pre 计算的第 i 次迭代中出现的每个顶点，也会出现在 tpre 计算的第 i 次迭代中。

Claim. $\forall i \in \mathbb{N}, X_i \subseteq X_i^{\mathsf{t}}$.

证明。 我们对 i 使用归纳法证明该命题。对于基例，当 $i = 0$ 时，该陈述显然成立。对于归纳假设（IH），假设该陈述对某个 $i \in \mathbb{N}$ 成立。

$$\begin{aligned} X_{i+1}^{\mathsf{t}} &= \mathsf{tpre}(X_i^{\mathsf{t}}) \cup X_i^{\mathsf{t}} && (23) \\ &\supseteq \mathsf{tpre}(X_i) \cup X_i, && \text{by IH and monotonicity of } \mathsf{tpre} \quad (24) \\ &\supseteq \mathsf{pre}(X_i) \cup X_i, && \text{by the claim above} \quad (25) \\ &= X_{i+1}, && (26) \end{aligned}$$

因此，根据归纳法，该命题对任意 $i \in \mathbb{N}$ 都成立。 ◁

该命题证明了引理的一个方向，即 $\mu X.\ U \cup \mathsf{pre}(X) \subseteq \mu X.\ U \cup \mathsf{tpre}(X)$。上述两个命题还表明 $l \leq k$，其中 l 和k 分别是 tpre 和 pre 计算的终止步。对于另一个方向，我们证明，在 tpre 计算的第 i 次迭代中出现的每个顶点最终也会出现在 pre 计算中。

命题。 $\forall i \in \mathbb{N}$，$\exists j \geq i$ 使得 $X_i^{\mathsf{t}} \subseteq X_j$。

证明。我们再次通过对 i 进行归纳法来证明该命题。对于基例，带有 $j = 0$时，该陈述显然成立。然后，对于归纳假设（IH），假设该陈述对某个 $i \in \mathbb{N}$ 成立，且有某个 $j \geq i$，即 $X_i^{\mathsf{t}} \subseteq X_j$。

令 $v \in X_{i+1}^{\mathsf{t}} \setminus X_i^{\mathsf{t}}$为任意顶点。则 $v \in \mathsf{attr}^0(X_i^{\mathsf{t}})$ 或 $v \in \mathsf{cpre}^1(\mathsf{attr}^0(X_i^{\mathsf{t}}) \cup X_i^{\mathsf{t}})$。

在前一种情况下，当 $v \in \mathsf{attr}^0(X_i^{\mathsf{t}})$ 时，我们有

$$\begin{aligned} v &\in \mathsf{cpre}^{0,t}(X_i^{\mathsf{t}}), && \text{for some } p \in \mathbb{N} && (27)\\ &\subseteq \mathsf{cpre}^{0,p}(X_i), && \text{by IH} && (28)\\ &\subseteq \mathsf{pre}^{0,p}(X_i), && && (29)\\ &\subseteq X_{p+i}. && && (30) \end{aligned}$$

然后，自从 $\mathsf{attr}^0(X_i^{\mathsf{t}})$ 至多经过 $|V| = n$ 次迭代便会终止，即 $\cup_{i\geq 1}^{n} \mathsf{cpre}_G^{a\,i}(X_i^{\mathsf{t}}) = \mathsf{attr}^0(X_i^{\mathsf{t}})$，我们有 $\mathsf{attr}^0(X_i^{\mathsf{t}}) \subseteq X_{n+i}$。而在后一种情况下，当 $v \in \mathsf{cpre}^1(\mathsf{attr}^0(X_i^{\mathsf{t}}) \cup X_i^{\mathsf{t}})$ 时，我们有

$$\begin{aligned} v &\in \mathsf{cpre}^1(\mathsf{attr}^0(X_i^{\mathsf{t}}) \cup X_i^{\mathsf{t}}), && && (31)\\ &\subseteq \mathsf{cpre}^1(X_{n+i} \cup X_j), && \text{by IH and discussion above} && (32)\\ &\subseteq \mathsf{pre}(X_{max\{n+i,j\}}), && && (33)\\ &\subseteq X_{max\{n+i,j\}+1}. && && (34) \end{aligned}$$

由于 $i \leq p + i \leq max\{n + i, j\} + 1$, $v \in X_{max\{n+i,j\}+1}$。然后根据归纳法，命题成立。◁

该命题表明 $X_k = X_k^{\mathsf{t}}$，因为 $l \leq k$ 和 $X_k^{\mathsf{t}} = X_l^{\mathsf{t}}$。因此，引理得证。

用于布奇博弈的 C APA 假设

为方便读者，我们在此重述定理 2。

定理 2. 设 $\mathcal{G} = (G = (V, E)\ \Phi = \Box\Diamond U)$ 是一个带有 $Z^* = \mathrm{TB\ddot{U}CHI}(G, U)$ 的 *Büchi* 博弈，且 $H^\ell = \{H_i\}_{i\geq 0}$ 满足

$$\emptyset \neq H_i := (\mathit{front}(X^i) \times (X^{i+1} \setminus \mathit{front}(X^i))) \cap E, \tag{12}$$

其中，X^i 是对 X 的计算在第 i 次迭代中，以及对 Y 的计算在$\mathrm{TB\ddot{U}CHI}$中的最后一次迭代中所得的集合。则 $\Psi = \Psi_{\mathrm{UNSAFE}}(S) \wedge \Psi_{\mathrm{LIVE}}(H^\ell)$ 是针对 $\mathcal{G}$ 的 *APA*，其中 $S = \mathrm{UNSAFEA}(G, U)$。我们用$\mathrm{LIVEA}(G, U)$ 表示构造上述活组的算法，其运行时间为 $\mathcal{O}(n^3)$，其中 $n = |V|$。

证明。由于 $\mathit{front}(X^i) \subseteq V^1$，我们观察到，$Z^* \cap V^0$ 中的每个顶点都会在 tpre 的 attr^0 部分中被加入到 X 的最小不动点计算中。于是，$V^0 \cap Z^*$可以划分为集合 V_1、$V_2 \ldots$、V_p，其中 $V_i = (X^i \setminus X^{i-1}) \cap V^0$。基于

这一观察，我们将在下文证明充分性、可实现性和宽容性，最后讨论 UnsafeA 的复杂度。

▶ 可实现性：由于活组的来源是玩家 1 的顶点子集，因此，若玩家 1无限次选择这些活组边之一，且这些边的来源被无限次访问，并且玩家 0无法对此予以否定，那么该假设很容易实现。

▶ 充分性：考虑玩家 0的策略 π^0：在顶点 $v \in V_i$处，她采用 attr^0 策略到达 X^{i-1}；对于其他顶点，她任意选择。我们证明，在所有合作制胜区域中的顶点上，π^0 在假设 Ψ 下均获胜。

令 $v_0 \in Z^* = \langle\!\langle 0,1 \rangle\!\rangle \Box\Diamond U$（来自命题 1）。令 π^1 为玩家 1的任意策略，使得 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$，并令 $\rho = v_0 v_1 \ldots$ 为任意 $\pi^0\pi^1$-对局。于是 $\rho \in \mathcal{L}(\Psi)$。剩下要证明 $\rho \in \mathcal{L}(\Phi)$。

假设 $\rho \notin \mathcal{L}(\Phi)$，即 $inf(\rho) \cap U = \emptyset$。令 Y在 k次迭代后达到饱和，也就是说 $Y^{k-1} \supsetneq Y^k = Y^{k+1}$。注意，根据标准不动点计算（更多详情请参见 [10]），我们有

$$X^1 = \big(U \cap \mathsf{tpre}(Y^k)\big) = (U \cap \mathsf{tpre}(Z^*)) \,. \tag{35}$$

于是，$\exists\, i \in [2,p]$，使得 $inf(\rho) \cap X^i \neq \emptyset$，但 $inf(\rho) \cap X^j = \emptyset$，$\forall j < i$，因为如果不存在这样的 i，则 $inf(\rho) \cap X^1 \neq \emptyset$，且 $inf(\rho) \cap U \neq \emptyset$，根据公式（35），这与我们的假设矛盾。

根据 π^0 和 attr^0, 的定义，每当 ρ 访问 前沿(X^{i-1}) 时，它最终也会访问 X^{i-1}。如果 ρ 无限次访问 前沿(X^{i-1})，根据H^ℓ的定义，边 $(v,\ v') \in$ (前沿(X^{i-1}) $\times$ ($X^i \setminus$ 前沿(X^{i-1}))) $\cap E$ 会被无限次经过，其中 $v' \in (X^i \setminus$ 前沿(X^{i-1})) $= \mathsf{attr}^0($ $X^{i-1}) \cup X^{i-1}$，因此 X^{i-1} 同样会被无限次访问，这与我们的假设矛盾。因此，$\rho \in \mathcal{L}(\Phi)$，且 $v_0 \in \langle\!\langle 0 \rangle\!\rangle_\Psi \Phi$。

▶ 宽容性：现在考虑该假设的宽容性，令 $\rho \in \mathcal{L}(\Phi)$。假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$，那么对局 ρ 中会经过某条边 $(v,\ v') \in S$。到达 v' 后，ρ 仍满足布奇条件。因此，根据命题 1，$v' \in Z^*$，但这样一来 $(v,\ v') \notin S$，这就产生了矛盾。

情况 2：如果 $\rho \notin \mathcal{L}(\Psi_{\text{LIVE}}(H^\ell))$，那么存在 $\exists H_i \in H^\ell$，使得 ρ 无限次访问 $src($ $H_i) =$前沿(X^i)，但 H_i 中没有任何边被无限次经过。由于 H_i 中的边会通向 ($X^{i+1} \setminus$前沿(X^i))，对局必须停留在前沿(X^i) 中，或者在某个 $j > i+1$ 时刻转至 $X^j \setminus X^i$。第一种情况下，由于 $U \cap Z^* \subseteq X^1$，$\rho \notin \Phi$，这会产生矛盾。另一方面，在第二种情况下，转至 X^j 后，ρ 有一条边从某个 $v \in Z^* \setminus X^{j-1}$ 指向某个 $v' \in X^i$（否则无法到达 $U \cap Z^* \subseteq X^1 \subseteq X^i$）。但这样一来，$v$ 就会被添加到 X^{i+1}中，这与 $j > i+1$ 这一事实相矛盾。无论哪种情况，都会产生矛盾，因此 $\rho \in \mathcal{L}(\Psi)$。

▶ 复杂度分析。活组的计算需要 $\mathcal{O}(n^2)$ 的时间来计算内部最小不动点变量 X，并且此类计算最多进行 n 次。在计算内部不动点期间，可在 Y 的最后一次迭代中，以 $\mathcal{O}(n^2)$ 的额外开销计算活组。因此，总计算时间为 $\mathcal{O}(n^3)$。

C.1 Büchi 博弈的更快算法

算法 2 LIVEA

输入： $G = (V = V^0 \cup V^1,\ E)$，Büchi 目标 $\Phi = \square\lozenge I$，针对 $I \subseteq V$

输出： 假设 Ψ 关于 $Player\ 1$

1: $Z^* \leftarrow$ 布奇$_{0,1}(G,\ I)$
2: $(S^i, S^j) \leftarrow$ UnsafeA(G, Z^*)
3: $G \leftarrow G|_{Z^*}, I \leftarrow I \cap Z^*$ ▷ 所有顶点均为协作式布奇获胜
4: $H^\ell \leftarrow$ComputeLiveGroups$((G,\ I)\ \emptyset)$
5: **返回**(S, H^ℓ)

6: **过程** ComputeLiveGroups$((G, I)\ H^\ell)$
7: $U \leftarrow I$
8: **当** $U \neq V$ **do 时**
9: $W_{attr} \leftarrow \text{attr}^0_G(G,\ U)$
10: $U \leftarrow U \cup W_{attr}$
11: $C \leftarrow \text{cpre}^1_G(U) \setminus U$
12: $H^\ell \leftarrow H^\ell \cup \{\{(u,\ v) \in E \cap (C \times U)\}\}$
13: $U \leftarrow U \cup C$
14: **返回** (H^ℓ)

定理 6。给定博弈图 $G = (V = V^0 \cup V^1, E)$，以及玩家 0 的布奇胜利条件 $\Phi = \square\lozenge I$。那么，算法 *2* 在 $\mathcal{O}(m+n)$ 时间内终止，并且$\Psi = \Psi_{\text{UNSAFE}}(S) \wedge \Psi_{\text{LIVE}}(H^\ell)$ 是玩家 1 关于 Z^* 的充分宽容假设。在此，$n = |V|$ 和 $m = |E|$。

证明。 我们首先证明该算法会终止。我们证明过程 ComputeLiveGroups 会终止。由于在步骤 3 中，博弈图被限制在合作式 Büchi 获胜区域 Z^*，因此我们需要证明在该过程中，$U = V = Z^*$ 最终会成立。令 U_l 为 ComputeLiveGroups$((G,\ I)\ \emptyset,\ \emptyset)$ 在第 l 次迭代后的 U 值，其中 $U^0 = I$。由于顶点只会被添加到 U（绝不会被移除），且顶点数量有限，因此对于某个 $m \in \mathbb{N}$，$U_0 \subset U_1 \subset \ldots \subset U_m = U_{m+1}$。

自从另一个方向显然成立，我们证明 $Z^* \subseteq U_m$。假设情况并非如此，即 $v \in Z^* \backslash U_m$。自从 $v \in Z^*$，两个玩家可以协作从 v 访问 I。那么存在一条有限路径 $\mathsf{p} = v_0 v_1 \cdots v_k$，满足 $v_k \in I$ 和 $v_0 = v$。但自从 $I = U_0 \subseteq U_m$，而 $v \notin U_m$，p 最终进入 U_m。令 l 为满足 $v_l \notin U_m$ 但 $v_{l+1} \in U_m$ 的最高索引。

那么，如果 $v_l \in V^0$，它会在第 $(m+1)$ 次迭代的步骤 10 中被加入 U，即 $U_m \neq U_{m+1}$。否则，如果 $v_l \in V^1$，由于 $v_{l+1} \in U_m$，它会在第 $(m+1)$ 次迭代的步骤 13 中被加入 U，即 $U_m \neq U_{m+1}$。无论哪种情况，都会得到一个矛盾。因此，$v \in U_m$，这意味着 $Z^* = U_m$。因此，ComputeLiveGroups 过程以及算法 2 都会终止。

现在我们证明，得到的假设是充分宽容的。

▶ 可实现性：注意，在步骤 11 中， $C \subseteq V^1$：因为如果 $v \in V^0 \cap C$，那么从 v 到 U 存在一条边，因此根据步骤 9 和 10， $v \in U$ 已经成立。接着，由于活组的源（它们仅在步骤 12 中添加）是玩家*1* 的顶点的子集，因此很容易实现该假设：如果玩家 1 在这些活跃组边中有一条被无限次对局，那么当这些源被无限次访问时，且玩家 0 无法证伪该假设。
▶ 充分性：再次设 U_l 和 m 如前所定义。定义 $X_l := U_l \backslash U_{l-1}$（针对 $1 \leq l \leq m$），以及 $X_0 = U_0 = I$。那么，对于某个$l \in [0$，每个顶点 $v \in Z^*$ 都属于 X_l ； $m]$。

考虑玩家 *0* 的策略 π^0 ： 在顶点 $v \in V_0 \cap X_l$处，她采用 attr^0 策略到达 U_{l-1}；对于其他顶点，她则任意行动。我们证明，对于玩家 *0*，在合作制胜区域 Z^*内的所有顶点上，策略 π^0 在假设 Ψ 下均为获胜策略。

设 $v_0 \in Z^* = \langle\!\langle 0,1 \rangle\!\rangle \Box \Diamond I$ （来自步骤 1）。设 π^1 是玩家*1*的任意策略，满足 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$；再设 $\rho = v_0 v_1 \ldots$ 是任意的 $\pi^0\pi^1$- 对局。则 $\rho \in \mathcal{L}(\Psi)$。还需证明 $\rho \in \mathcal{L}(\Phi)$。

假设 $\rho \notin \mathcal{L}(\Phi)$，即 $inf(\rho) \cap I = \emptyset$。注意，由于安全假设模板， ρ 永远不会离开 Z^*。接着考虑在 ρ中无限次出现的顶点集合 R。令 $0 \leq k \leq m$ 为满足 $R \cap X_k \neq \emptyset$的最小指标。根据该假设， $k > 0$。令 $v \in R \cap X_k$。

如果 $v \in V^0$，根据 π^0的定义，每当 ρ 到达 v时，它就必须到达U_{k-1}，这与 k的最小性矛盾。否则，如果 $v \in V^1$，那么根据H^ℓ的定义，无限次到达 v 意味着无限次到达 $\text{attr}_0(U_{k-1})$。但如上所述，对局仍会访问 U_{k-1} ，由此产生矛盾。

无论哪种情况，我们都会得到矛盾，这意味着该假设是错误的。因此，$\rho \in \mathcal{L}(\Phi)$，且 $v_0 \in \langle\!\langle 0 \rangle\!\rangle_\Psi \Phi$。
▶ 宽容性：现在来讨论假设的宽容性，令 $\rho \in \mathcal{L}(\Phi)$。假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$，那么对局中会经过某条边 $(v, v') \in S$ 。到达 v'后，ρ 仍满足 Büchi 条件。因此， $v' \in Z^* = \langle\!\langle 0,1 \rangle\!\rangle \Box \Diamond I$，但此时 $(v, v') \notin S$，这就产生了矛盾。

情况 2：如果 $\rho \notin \mathcal{L}(\Psi_{\text{LIVE}}(H^\ell))$，那么存在 $\exists H_i \in H^\ell$，使得 ρ 无限多次访问 $src(H_i) = C_l$（取 l 次迭代后的 C 值），但 H_i 中的任何边都不会被无限次经过。由于 H_i 中的边通向 $\text{attr}^0(U_{l-1})$，对局必定停留在 C_l中，或在某个 $k > l+1$时刻前往 $U^k \setminus U^l$。第一种情况下，由于$I \cap Z^* \subseteq U^0$， $\rho \notin \Phi$，这将产生矛盾。另一方面，在第二种情况下，前往 $U^k \setminus U^l$后， ρ有一条边从某个 $v \in Z^* \backslash U^{k-1}$通向某个 $v' \in U^l$ （否则无法到达 $I \cap Z^* \subseteq U^0 \subseteq U^l$）。但这样一来， v 就会被加入 U^{k+1}，这与 $k > l+1$的事实相矛盾。无论哪种情况，都会产生矛盾，因此 $\rho \in \mathcal{L}(\Psi)$。

▶ 复杂度分析。 计算合作制胜区域所需的时间与博弈图的大小呈线性关系，即 $\mathcal{O}(m+n)$。ComputeLiveGroups 过程需要 $\mathcal{O}(m+n)$ 时间。因此，最终所需时间与博弈图中的边数呈线性关系。

D APA余布奇博弈的假设

为方便读者，我们在此重述定理 3。

定理 3。令 $\mathcal{G} =$（$G =$（V、E）$\Diamond\Box U$），$Z^* =$CoBüchi（G，U）且

$$D = \Big(\big[(X^1 \cap V^1) \times (Z^* \setminus X^1)\big] \cup \big[\bigcup_{i>1}(X^i \cap V^1) \times (Z^* \setminus X^{i-1})\big]\Big) \cap E, \quad (15)$$

其中，X^i 是通过对不动点变量 X 在CoBüchi中进行第 i 次迭代所计算出的集合。则 $\Psi = \Psi_{\text{UNSAFE}}$（$S$）$\wedge \Psi_{\text{COLIVE}}$（$D$）是关于 $\mathcal{G}$ 的 *APA*，其中 $S =$UnsafeA（G、U）。我们用 CoLiveA（G、U）表示构造上述共活边 D 的算法，其运行时间为 $\mathcal{O}$（n^3），其中 $n = |V|$。

证明。我们注意到，$V^0 \cap Z^*$ 可以划分为集合 V_1、$V_2 \ldots$、V_p，其中$V_i =$（$X^i \setminus X^{i-1}$）$\cap V^0$。如果 $v \in Z^*$ 具有秩 i，则称其为 $v \in X^i \setminus X^{i-1}$。

▶ 可实现性：我们再次观察到，共活边的源点是玩家 1的顶点；根据构造，每个源点至少有一条既非共活也非不安全的备选边。因此，玩家 1只需有限次选择共活边，就能轻松实现它们。接下来，我们再次证明充分性、可实现性和宽容性，最后讨论 CoLiveA 的复杂度。

▶ 充分性：考虑玩家 0的以下策略 π^0：在顶点 $v \in V_1$处，她选择边（v，v'）$\in E$，使得 $v' \in X^1$；在顶点 $v \in V_i$处，对于 $i \in [2;\ p]$，她选择现有的边以到达 X^{i-1}；对于所有其他顶点，她任意选择。

由不动点的正确性（见 [1, 章2]），令 $v_0 \in Z^* = \langle\!\langle 0, 1\rangle\!\rangle\Diamond\Box U$。设 π^1 是玩家 1的任意策略，满足 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$，并且$\rho = v_0 v_1 \ldots$是任意的 $\pi^0\pi^1$-对局。那么 $\rho \in \mathcal{L}(\Psi)$。

自从 $\rho \in \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$，$v_i \in Z^*$ 对所有 i 都成立。现在假设 $\rho \notin \mathcal{L}(\Phi)$，即 $\inf(\rho) \cap (Z^* \setminus U) \neq \emptyset$。令 $u \in Z^* \setminus U$。那么，为了无限多次到达 u，D 中必有一条边在 ρ 中被无限多次选取，这与 $\rho \in \mathcal{L}(\Psi)$ 的事实矛盾。因此，$\rho \in \mathcal{L}(\Phi)$。

▶ 宽容性：令 $\rho = v_0 v_1 \ldots$，使得 $v_0 \in Z^*$ 且 $\rho \in \mathcal{L}(\Phi)$。假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \Psi_{\text{UNSAFE}}(S)$，那么与布奇情形相同的论证会导出矛盾。

情况 2：如果 $\rho \notin \Psi_{\text{COLIVE}}(D)$，也就是说 $\exists(u,\ v) \in D$，，使得 ρ 无限多次取值为 $(u,\ v)$。根据 D 的定义，$v \in Z^* \setminus X^1$，由此可知 $v \notin U$；因为如果 $v \in U$，那么它就会属于 X^1（见 [1]）。因此，$\rho \notin \mathcal{L}(\Phi)$，从而导出矛盾。所以 $\rho \in \mathcal{L}(\Psi)$。

▶ 复杂度分析。与活跃组假设的情况非常相似，因此省略。

D.1 余布奇快速不动点算法

现在我们证明加速不动点算法的正确性。

定理 4. 设 $\mathcal{G} = (G = (V, E)\ \Phi = \Diamond\Box U)$ 是一个余布奇博弈，其 $Z^* = \mathrm{TCoBüChi}(G, U)$ 并且

$$D = \left(\begin{array}{r} (X^1 \cap V^1) \times (Z^* \setminus X^1) \cup \\ \bigcup_{i>1} \mathit{front}(X^i) \times (\mathit{front}(X^i) \cup Z^* \setminus X^{i+1}) \end{array} \right) \cap E, \tag{18}$$

其中 X^i 是在不动点变量 X 的第 i 次迭代中计算得到的集合。则 $\Psi = \Psi_{\mathrm{UNSAFE}}(S) \wedge \Psi_{\mathrm{COLIVE}}(D)$ 是针对 $\mathcal{G}$ 的 *APA*，其中 $S = \mathrm{UnsafeA}(G, U)$。此外，可以在 $\mathcal{O}(n^3)$ 时间内构造 D，其中 n 是顶点数。

证明。 与布奇情形类似， $Z^* \cap V^0$ 中的每个顶点都会在 tpre 的 attr^0 部分被加入到 X 的最小不动点计算中，并且 $V^0 \cap Z^*$ 可划分为集合 $V_1, V_2 \ldots, V_p$，其中 $V_i = (X^i \setminus X^{i-1}) \cap V^0$。如果 $v \in X^i \setminus X^{i-1}$，我们称 $v \in Z^*$ 的秩为 i。

考虑玩家 0 的策略 π^0：在顶点 $v \in V_1$ 处，选择满足 $v' \in X^1$ 的边 $(v, v') \in E$；在顶点 $v \in V_i$ 处，对于 $i \in [2;\ p]$，采用 attr^0 策略到达 X^{i-1}；对于其他顶点，则任意行动。

令 $v_0 \in Z^* = \langle\!\langle 0, 1 \rangle\!\rangle U$，由命题 2 可知。令 π^1 为玩家 1 的任意策略，使得 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$，并令 $\rho = v_0 v_1 \ldots$ 为任意 $\pi^0\pi^1$-对局。则 $\rho \in \mathcal{L}(\Psi)$。

自从 $\rho \in \mathcal{L}(\Psi_{\mathrm{UNSAFE}}(S))$，对所有 i 都有 $v_i \in Z^*$。现在假设 $\rho \notin \mathcal{L}(\Phi)$，即 $\mathrm{inf}(\rho) \cap (Z^* \setminus U) \neq \emptyset$。令 $u \in Z^* \setminus U$。那么，为了使 u 无限次出现，必须在 ρ 中无限次经过来自 D 的某条边，这与 $\rho \in \mathcal{L}(\Psi)$ 的事实矛盾。因此， $\rho \in \mathcal{L}(\Phi)$。

现在我们证明该假设的宽容性。令 $\rho = v_0 v_1 \ldots$，使得 $v_0 \in Z^*$ 且 $\rho \in \mathcal{L}(\Phi)$。假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \Psi_{\mathrm{UNSAFE}}(S)$，那么布奇情形中的同一论证会导出矛盾。

情况 2：如果 $\rho \notin \Psi_{\mathrm{COLIVE}}(D)$，也就是说 $\exists(\{\mathrm{v4}, v) \in D$, 使得 ρ 无限次取值为 (u, v)。根据 D 的定义， $v \in Z^* \setminus X^1$，这意味着 $v \notin U$，因为如果 $v \in U$，那么它就会属于 X^1 （见 [1]）。因此， $\rho \notin \mathcal{L}(\Phi)$，从而导出矛盾。所以 $\rho \in \mathcal{L}(\Psi)$。

我们再次观察到，共活边的起点是玩家 1 的顶点，并且根据构造，每个起点至少有一条既非共活也非不安全的替代边。因此，玩家 1 只需有限次选择这些边，就能轻松实现这一点。

复杂度分析与活跃组假设的分析类似。

D.2 更快的余布奇博弈算法

算法 3 CoLiveA

输入： $G = (V = V^0 \cup V^1,\ E)$，$I \subseteq V$

输出： 假设 Ψ 在 $Player\ 1$

```
1: Z* ← CoBüchi^{0,1}(G, I)
2: S ← UnsafeA(G, Z*)
3: G ← G|_{Z*}, I ← I ∩ Z*                ▷ 所有顶点均为合作式 co-Büchi 获胜
4: D ←ComputeCoLive((G, I) ∅)
5: 返回 (S, D)

6: 过程 ComputeCoLive((G, I) D)
7:     U ← Safety^{0,1}(G, I)                                   ▷ U ⊆ I
8:     D ← (U × V\U) ∩ E
9:     当 U ≠ V 时执行
10:        W_attr ← attr^0_G(G, U)
11:        U ← U ∪ W_attr
12:        C ← cpre^1_G(U) \ U
13:        D ← D ∪ {(u, v) ∈ E ∩ (C × V \ U)}
14:        U ← U ∪ C
15:    返回 D
```

定理 7。 给定一个博弈图 $G = \big(V = V^0 \cup V^1, E\big)$，其余布奇目标为 $\Phi = \Diamond\Box I$，玩家 i 的目标。则算法 *3* 的运行时间为 $\mathcal{O}(m+n)$，并且$\Psi = \Psi_{\text{UNSAFE}}(S) \wedge \Psi_{\text{COLIVE}}(D)$是玩家 1从 Z^*出发的充分宽容假设。这里， $n = |V|$ 和 $m = |E|$。

证明。 我们首先证明该算法会终止。由于我们在步骤 3 中将图限制在合作制胜区域，因此当游戏图的所有顶点对于 co-Büchi 目标 $\Phi = \Diamond\Box I$ 都是合作制胜时，我们证明过程 ComputeCoLive 会终止。我们最终会证明 $U = V = Z^*$。

令 U_l 为 while 循环第 l 次迭代后变量 U 的值，其中 $U_0 = \text{Safety}^{0,1}(G, I)$。由于顶点只会在 U 中添加，因此对于某个 $m \in \mathbb{N}$， $U_0 \subset U_1 \subset \ldots \subset U_m = U_{m+1}$。假设 $V \not\subseteq U_m$，那么存在 $v \in V \backslash U_m$。由于 $v \in Z^*$，存在一条从 v 到 U_0 的 $\rho = vv_1v_2\ldots$，并且永远停留在那里。接着考虑满足 $v_l \notin U_m$ 的最大索引 l，但 $v_{l+1} \in U_m$。注意该索引存在，因为 $U_0 \subseteq U_m$。

那么，如果 $v_l \in V^0$，它会在第 $(m+1)$ 次迭代的步骤 11 中被添加到 U，即 $U_m \neq U_{m+1}$。否则，如果 $v_l \in V^1$，它会在第 $(m+1)$ 次迭代的步骤 14 中被添加到 U，因为 $v_{l+1} \in U_m$，即 $U_m \neq U_{m+1}$。无论哪种情况，都会得到矛盾。因此，$v \in U_m$，这意味着 $Z^* = U_m$。因此，ComputeCoLive 过程以及算法 3 都会终止。

现在我们证明 Ψ 是一个充分宽容的假设。同样，设U_l 和 m 如前所定义。定义 $X_l := U_l \backslash U_{l-1}$，其中 $1 \leq l \leq m$，并且$X_0 = U_0 = I$。那么，对于某个 $l \in [0$，每个顶点 $v \in Z^*$ 都属于 X_l； $m]$。

我们再次分别证明充分性、可实现性和宽容性，最后讨论 CoLiveA 的复杂度。

▶可实现性：我们再次观察到，D 中的共活边的源点是玩家1 的顶点，并且根据构造，每个源点至少有一条既非共活也非不安全的替代边。因此，玩家1 只需有限次选择这些边，就能轻松实现它们。

▶ 充分性：考虑玩家0 的以下策略 π^0 ：在顶点 $v \in X_0 \cap V^0$处，她选择边（v, v'） $\in E$，使得 $v' \in X_0$；在顶点 $v \in X_l \cap V^0$处，对于$l \in [2$； $m]$，她采用 attr^0 策略到达 U_{l-1}；对于所有其他顶点，她任意行动。

令 $v_0 \in Z^* = \langle\langle 0, 1 \rangle\rangle \Diamond \Box I$ （来自步骤 1）。令 π^1 为玩家 1 的任意策略，使得 $\mathcal{L}(\pi^1)$ $\subseteq \mathcal{L}(\Psi)$，并令 $\rho = v_0 v_1 \ldots$为任意 $\pi^i \pi^1$-对局。于是 $\rho \in \mathcal{L}(\Psi)$。接下来只需证明 $\rho \in \mathcal{L}(\Phi)$。

自从 $\rho \in \mathcal{L}(\Psi_{\text{UNSAFE}}(S))$，且根据 π^i的定义，对于所有 i都有 $v_i \in Z^*$。现在假设$\rho \notin \mathcal{L}(\Phi)$，即 inf$(\rho) \cap (Z^* \setminus I) \neq \emptyset$。令 $u \in Z^* \setminus I$。要使 u 无限次出现， D 中的某条边必须在 ρ中被无限次经过，因为 π^i 会使对局朝着 I发展。但这与 $\rho \in \mathcal{L}$ (Ψ) 相矛盾。因此， $\rho \in \mathcal{L}(\Phi)$。▶ 宽容性：令 $\rho = v_0 v_1 \ldots$满足 $v_0 \in Z^*$ 且 $\rho \in \mathcal{L}$ (Φ)。假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \Psi_{\text{UNSAFE}}(S)$。那么，沿用布奇情形中的相同论证即可得到矛盾。

情况 2：如果 $\rho \notin \Psi_{\text{COLIVE}}(D)$，即存在 $\exists(u, v) \in D$，使得 ρ 无限多次采取 (u, v)。根据 D 的定义， $v \in Z^* \setminus U_0$，这意味着 $v \notin I$，因为如果 $v \in I$，那么它就会属于 U_0。因此， $\rho \notin \mathcal{L}(\Phi)$，从而得到矛盾。所以 $\rho \in \mathcal{L}(\Psi)$。

▶ 复杂度分析。与 Büchi 目标的情况非常相似，因此省略。

奇偶博弈的 APA 假设

为方便读者，我们在此重述定理5。

定理 *5.* 设 $\mathcal{G} = (G = (V, E), \textit{Parity}(C))$ 是一个奇偶博弈，且 $(S, D, \mathcal{H}^\ell) = \textit{ParityAssumption}(G, C)$。那么 $\Psi = \Psi_{\text{UNSAFE}}(S) \wedge \Psi_{\text{COLIVE}}(D) \wedge \Psi_{\text{COND}}(\mathcal{H}^\ell)$ 是 $\mathcal{G}$ 的一个 *APA*。此外，算法 *1* 在 $\mathcal{O}(n^4)$ 时间内终止，其中 $n = |V|$。

证明。 下面我们证明充分性、可实现性和宽容性，然后分析算法 1 的复杂度。

▶ 可实现性：我们注意到，根据 Safety、活性假设和共活性假设的可实现性，该假设是可实现的：如果某个条件活跃组所对应的顶点集被无限次到达，并且活组的源也被无限次访问，玩家 1 就可以选择活跃

组边，因为它们由玩家 1控制。此外，不会因条件活跃组而产生任何冲突，因为按照构造，条件活跃组中不可能包含不安全边或共活边。

▶ 充分性：我们给出一个玩家 0的策略，该策略取决于博弈中出现的最高优先级的奇偶性，并证明在假设 Ψ 适用于合作制胜区域 $Z^* = \text{Parity}(G, C)$ 中所有顶点的情况下，该策略能够获胜。该策略使用有限记忆，以及玩家 1在具有布奇（定理 2）和余布奇（定理 3）目标的子博弈中的获胜策略。

我们用$B\ddot{u}chi(G, U)$表示博弈$(G, \square\Diamond U)$，用$co\text{-}B\ddot{u}chi(G, U)$表示博弈$(G, \Diamond\square U)$。我们还使用算法1中的 d、 W_d和 $W_{\neg d}$的定义。考虑玩家 0的以下策略 π^0：

▷ d为**奇数**：如果对局处于 $V \setminus W_{\neg d}$，那么玩家 0 会采用$co\text{-}B\ddot{u}chi(G, W_{\neg d})$获胜策略，最终到达 $W_{\neg d}$。如果对局处于 $W_{\neg d} \cap Z^*$，玩家 0 会采用针对 ($G|_{W_{\neg d}}$,奇偶(C)) 的递归胜利策略。否则，她会任意行动。

▷ d 为**偶数**：如果对局处于 W_d，玩家 0 会在$B\ddot{u}chi(G, W_d)$、$B\ddot{u}chi(G, W_d \cup W_{d-2})$、…、$B\ddot{u}chi(G, W_d \cup W_{d-2} \cup \cdots W_2)$ 这些获胜策略之间切换；也就是说，对于每个顶点，她先采用上述序列中的第一个策略，然后当该顶点再次出现时，她在下一步采用第二个策略，并在每次从同一顶点行动时继续切换到下一个策略。如果对局处于 $V \setminus W_d \cap Z^*$，那么她会采用针对 ($G|_{W_{\neg d}}$,奇偶(C)) 的递归胜利策略，其中 C 会像第 16 行那样再次修改。否则，她会任意行动。

我们对出现的最高优先级 d 进行归纳，证明上述构造的策略 π^0 能够确保玩家 0 在原博弈图上满足奇偶目标，前提是满足假设 Ψ。在基例中，当 $d = 0$时，构造出的策略显然是获胜策略，因为唯一存在的颜色是偶数。现在设该策略对于 $d - 1 \geq 0$ 是获胜策略。

设 $v_0 \in Z^* = \text{Parity}(G, C)$。设 π^1 是玩家 1 的任意策略，满足 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$，并设 $\rho = v_0 v_1 \ldots$ 是任意 $\pi^0\pi^1$-对局。那么$\rho \in \mathcal{L}(\Psi)$。我们需要证明 ρ 是获胜策略，即 $\rho \in \mathcal{L}(\Phi)$。注意，根据安全性假设以及 π^0 的构造， ρ 始终处于顶点集 Z^*中。

情况 1：如果 d 是奇数，那么由于在 $V \setminus W_{\neg d}$ 中的顶点处，玩家 0 会选择最终留在 $W_{\neg d}$ 中的走法，因此对局不可能在不违反 $\Psi_{\text{COLIVE}}(D)$ 的情况下始终留在 $W_{\neg d}$ 中。如果 ρ 最终留在 $W_{\neg d}$ 中，那么根据归纳假设，它是获胜的，因为 $W_{\neg d} \cap C_d = \emptyset$。

情况 2：如果 d 是偶数，那么如果对局最终一直停留在 W_d 中，且对局无限次访问奇优先级为 i 的顶点，那么根据条件活跃组假设， π^1 满足 LiveA(G, $C_{i+1} \cup C_{i+2} \cup \cdots \cup C_d$)。注意，玩家 0 从 ρ 中出现的每个顶点出发，都会进行无限多步，并采用 $B\ddot{u}chi(G, C_{i+1} \cup C_{i+2} \cup \cdots \cup C_d)$ 获胜策略。由于 π^1 满足 LiveA(G, $C_{i+1} \cup C_{i+2} \cup \cdots \cup C_d$)，这些移动之后，对局也会经过 $(C_{i+1} \cup C_{i+2} \cup \cdots \cup C_d)$。因此，对局将无限次经过偶数颜色 $> i$ 的顶点，这意味着ρ 获胜。否则，如果 ρ 最终留在 $V \setminus W_d$ 中，那么根据归纳假设，它也会获胜——

归纳假设。因此，这证明了算法所计算出的假设的充分性。

▶ 宽容性：现在来看假设的宽容性，令 $\rho \in \mathcal{L}(\Phi)$。我们通过反证法证明该命题，并假设 $\rho \notin \mathcal{L}(\Psi)$。

情况 1：如果 $\rho \notin \Psi_{\text{UNSAFE}}(S)$。那么在 ρ 中会经过某条边 $(v,\ v') \in S$ 。到达 v' 后，ρ 仍满足奇偶目标。因此，$v' \in Z^*$，但此时 $(v,\ v') \notin S$，这就产生了矛盾。

情况 2：如果 $\rho \notin \Psi_{\text{COND}}(\mathcal{H}^\ell)$。那么对于某个偶数 j 和奇数 $i < j$，ρ 会无限次访问$W_j \cap C_i$，但不满足活跃转移组假设 LiveA$(G',\ C_{i+1} \cup C_{i+2} \cdots \cup C_j)$，其中 $G' = G|_{W_j}$。根据集合 W_j 的构造，很容易看出，一旦 ρ 访问了 W_j，就再也无法访问 $V \setminus W_j$。因此，最终 ρ 会留在博弈 $\mathcal{G}'$ 中，并无限次访问 C_i。由于 $\rho \in \mathcal{L}(\Phi)$，它也会无限次访问某个偶优先级为 $> i$ 的顶点，因此，它在 G' 中满足 $\square\Diamond($ $C_{i+1} \cup C_{i+3} \cdots \cup C_j))$。由于 LiveA$(G',\ C_{i+1} \cup C_{i+3} \cdots \cup C_j))$ 是 $(G',\ \square\Diamond($ $C_{i+1} \cup C_{i+3} \cdots \cup C_j)))$ 的宽容假设，对局 ρ必须满足 LiveA$(G',\ C_{i+1} \cup C_{i+3} \cdots \cup C_j))$，这与假设矛盾。

情况 3：如果 $\rho \notin \Psi_{\text{COLIVE}}(D)$。那么对于某个奇数 i，边 $(u, v) \in$ CoLiveA$(G, W_{\neg i})$ 会被无限次经过。于是顶点 $v \in V \setminus W_{\neg i}$ 会被无限次访问。注意，ρ 不可能通过访问某个偶数 $j > i$而获胜，因为否则 v 就会属于 Büchi(G, C_j)：从 v 出发，我们可以无限次看到 j，因此在下一次递归步骤中，v 就会从 $\mathcal{G}$ 中移除。因此，ρ会无限次访问某个偶数 $j < i$，也就是说，i 不会被无限次访问。于是v 就会属于 $W_{\neg d}$，这就产生了矛盾。

▶ 复杂度分析。 我们注意到，合作式奇偶游戏可以在 $\mathcal{O}((n + m) \log d)$ 时间内求解，其中 n, m 和 d 分别是顶点数、边数和优先级数：考虑一个由 pz 拥有所有顶点的图，在 $\mathcal{O}(n + m)$ 时间内求出其强连通分量，并通过归约到偶环问题 [17]，检查这些分量中哪些包含最高优先级为偶数的环。对于偶数情况，ComputeSets 的运行时间为 $\mathcal{O}(n^2)$，但其复杂度被奇数情况下的 $\mathcal{O}(n^3)$ 时间所主导。对于每个优先级，ComputeSets 都会被调用一次，总调用次数至多为 $2n$ 次。因此，该算法的总运行时间为 $\mathcal{O}((n + m) \log d + 2n.(n^3)) = \mathcal{O}(n^4)$。

F 定义1与定义6的等价性

我们证明以下结果，说明对于我们所考虑的假设类，定义1与定义6是等价的。

命题3. *给定一个奇偶博弈 $\mathcal{G}$，设 Ψ 是由定理5计算得到的 APA。那么，当且仅当玩家 0 的策略在定义1下关于 Ψ 获胜时，该策略在定义6下关于 Ψ 也获胜。*

证明。 假设玩家 0 的策略 π^0 根据定义 6 在假设 Ψ 下获胜。则每个对局 $\rho \in \mathcal{L}(\pi^0)$ 要么不满足假设 Ψ，要么满足规范 Φ。因此，$\mathcal{L}(\pi^0) \subseteq \mathcal{L}(\Phi) \cup \mathcal{L}(\neg\Psi)$。现在，令 π^1 为玩家 1

策略，使得 $\mathcal{L}(\pi^1) \subseteq \mathcal{L}(\Psi)$。于是，

$$\begin{aligned}\mathcal{L}(\pi^0\pi^1) &= \mathcal{L}(\pi^0) \cap \mathcal{L}(\pi^1) \\ &\subseteq \big(\mathcal{L}(\Phi) \cup \mathcal{L}(\neg\Psi)\big) \cap \mathcal{L}(\Psi) \\ &= \big(\mathcal{L}(\Phi) \cap \mathcal{L}(\Psi)\big) \cup \big(\mathcal{L}(\neg\Psi) \cap \mathcal{L}(\Psi)\big) \\ &= \mathcal{L}(\Phi) \cap \mathcal{L}(\Psi) \\ &\subseteq \mathcal{L}(\Phi).\end{aligned}$$

因此，根据定义1， π^0 在假设 Ψ 下也获胜。

现在来看另一个方向，假设根据定义1， π^0 在假设 Ψ下获胜。令 $\rho \in \mathcal{L}$（π^0）。如果 $\rho \notin \mathcal{L}$（Ψ），那么结论成立。假设 $\rho \in \mathcal{L}$（Ψ），则我们必须证明 $\rho \in \mathcal{L}$（Φ）。我们声称存在一个玩家 1策略 π^1，使得 $\pi^1 \subseteq \mathcal{L}$（$\Psi$），且 ρ 遵循该策略。于是，根据定义1，$\mathcal{L}$（$\pi^0\pi^1$） $\subseteq \mathcal{L}$（Φ）。由于 ρ 同时遵循 π^0 和 π^1，因此 $\rho \in \mathcal{L}$（Φ），结论成立。

现在我们只需证明该命题。由于 Ψ 是一个可实现假设，存在一个玩家 1策略 π^1_*，使得 $\mathcal{L}$（π^1_*） $\subseteq \mathcal{L}$（Ψ）；根据定义，该策略在所有顶点上都有定义。令 $\rho = v_0 v_1 \cdots$。现在，令 π^1 为另一个玩家 1策略，对于每个对局前缀p $\in V^* V^1$，其定义如下：

$$\pi^1(\mathrm{p}) = \begin{cases} v_k & \text{if } \mathrm{p} = v_0 v_1 \cdots v_{k-1} \\ \pi^1_*(\mathrm{p}) & \text{otherwise.} \end{cases}$$

那么，显然， ρ 符合 π^1。现在，令 $\rho' \in \mathcal{L}(\pi^1)$，那么只需证明 $\rho' \in \mathcal{L}(\Psi)$。如果 $\rho' = \rho \in \mathcal{L}(\Psi)$，则证明完毕。否则，设 p' 是 ρ' 的一个最长前缀，同时也是 ρ 的前缀（也可以为空）。根据构造，在 ρ' 中，前缀 p' 之后所采取的动作符合 π^1_*。由于条件活跃组模板和共活性模板是尾部性质，且与前缀无关，因此 ρ' 满足假设 Ψ中的这些模板。此外，由于 p'是 $\rho \in \mathcal{L}(\Psi)$的前缀，且在 ρ' 中，p' 之后所采取的动作符合 π^1_*，因此对局 ρ' 不可能包含由假设 Ψ标记的任何不安全边。因此， $\rho' \in \mathcal{L}(\Psi)$。

G 实验结果完整表

Name	数量 顶点	数量 边	数量 优先级	计算 SImPA 的计算时间（秒）	计算 GIST 的计算时间（秒）
abcg仲裁器	417	638	3	0.016	9.79
ActionConverter	134	200	3	0.003	0.005
amba分解仲裁器2	141	212	4	0.005	0.21 *
amba分解仲裁器3	741	1176	4	0.064	3。798 *
AMBA 分解仲裁器 4	2468	3950	4	0.703	35。911 *
AMBA 分解仲裁器 5	7273	11，576	4	7.095	297。975 *
AMBA 分解仲裁器	36，824	67，018	4	203.104	超时
AMBA 分解解码	19	26	3	0.002	0.004
amba 分解编码 10	78，313	117，676	3	11.882	超时
amba 分解编码 11	156，184	234，506	3	45.455	超时

amba 分解编码 12	311，879	468，072	3	181.623	超时
amba 分解编码 13	623，222	935,110	3	732.713	超时
amba 分解编码 14	1,245,861	1，869，092	3	2960.11	超时
amba 分解编码 2	95	140	3	0.003	0.562
amba 分解编码 3	264	402	3	0.005	14.5
amba 分解编码 4	499	760	3	0.008	97.901
amba 分解编码 5	1534	2342	3	0.025	2616.04
amba 分解编码 6	2965	4500	3	0.06	超时
amba 分解编码 7	5804	8770	3	0.168	超时
amba 分解编码 8	11，459	17，264	3	0.534	超时
amba 分解编码 9	39，354	59，214	3	3.296	超时
amba 分解编码	292	476	3	0.006	4.23
AMBA 分解锁 2	398	592	3	0.04	50.282
AMBA 分解锁 3	1558	2336	3	0.074	2999.65
AMBA 分解锁 4	6182	9280	3	0.608	超时
AMBA 分解锁 5	24，646	36，992	3	9.563	超时
AMBA 分解锁 6	98，438	147,712	3	158.854	超时
AMBA 分解锁 7	393，478	590，336	3	2801.93	超时
AMBA 分解锁	180	288	3	0.009	1.288
AMBA 分解移位	44	64	3	0.006	0.021
AMBA 分解 tburst4	1061	1618	4	0.112	685.687
AMBA 分解 tincr	2082	3280	3	0.227	超时
AMBA 分解 tsingle	950	1450	4	0.094	464.176
仲裁器	39	54	3	0.008	0.024
带缓冲区的仲裁器	84	132	3	0.003	0.032
带取消功能的仲裁器	139	212	3	0.004	1.592
按钮	12	16	3	0.007	0.007
检测器	84	126	4	0.008	0.457
虚幻检测器	141	210	4	0.005	0。301*
敌人模块	20	28	3	0.005	0.005
双向自动扶梯	1622	2464	3	0.18	4.123
自动扶梯计数初始化	99	148	3	0.006	0.045
自动扶梯计数	79	118	3	0.003	0.012
自动扶梯非计数	21	30	3	0.002	0.003
自动扶梯非响应	7	8	3	0.002	0.002
EscalatorSmart	1783	3008	5	0.146	超时
完整仲裁器 2	204	324	3	0.007	4.939
完整仲裁器 3	1403	2396	3	0.082	538.772
完整仲裁器 4	7444	12,764	3	3.171	超时
完整仲裁器 5	44,019	76,920	3	109.252	超时
完整仲裁器	774	1218	3	0.025	704.267
完整仲裁器 unreal1	774	1218	3	0.041	697.743
完整仲裁器 unreal2	774	1218	3	0.027	698.161
游戏模块	143	214	3	0.008	0.032
递增	12	16	3	0.002	0.002
厨房计时器 V0	20	28	3	0.002	0.002
厨房计时器 V1	80	124	3	0.003	0.03
厨房计时器 V2	731	1138	3	0.038	3.302
厨房计时器 V3	1583	2482	3	0.165	14.573
厨房计时器 V4	2272	3554	3	0.408	34.324
厨房计时器 V5	4111	6584	3	1.422	134.261
厨房计时器 V6	4099	6560	3	1.438	140.525
lilydemo01	104	148	3	0.003	0.045
lilydemo02	104	148	3	0.003	0.044
lilydemo03	211	312	3	0.005	24.546
lilydemo04	304	448	3	0.011	77.08
lilydemo05	293	436	3	0.011	59.884
lilydemo06	369	548	3	0.026	121.068
lilydemo07	78	108	3	0.004	1.26
lilydemo08	26	36	4	0.005	0.01
lilydemo09	47	66	4	0.006	0.023*
lilydemo10	46	68	4	0.002	0.11
lilydemo11	65	100	3	0.007	0.108
lilydemo12	110	172	3	0.003	0.23
lilydemo13	11	14	3	0.007	0.011
lilydemo14	399	660	6	0.03	166.612
lilydemo15	133	206	3	0.008	0.48
lilydemo17	3102	5334	7	0.22	超时

lilydemo18	449	728	9	0.017	1566.9
lilydemo19	108	162	4	0.009	0.167 *
lilydemo20	1989	3252	4	0.4	超时
lilydemo21	4087	6824	3	0.428	超时
lilydemo22	373	556	3	0.02	109.443
lilydemo23	38	48	3	0.008	0.072
lilydemo24	263	406	4	0.008	54.354
负载均衡器	255	390	4	0.008	105.297
负载均衡器 unreal1	328	506	4	0.01	202.229
负载均衡器 unreal2	326	502	4	0.024	200.707
loadcomp2	387	614	4	0.024	2.974
loadcomp3	1556	2734	4	0.262	62.287
loadcomp4	3943	6968	4	1.94	979.018
loadcomp5	10，657	19，136	4	15.525	超时
loadfull2	331	552	4	0.043	1.187
loadfull3	1159	2030	4	0.143	28.051
loadfull4	3399	6084	4	1.421	496.464
loadfull5	6483	11,184	4	5.595	超时
线性时序逻辑到确定性广义 Büchi 自动机01	101	152	4	0.011	9.012
线性时序逻辑到确定性广义 Büchi 自动机02	489	758	4	0.011	74.949
线性时序逻辑到确定性广义 Büchi 自动机03	135	200	4	0.006	65.785
线性时序逻辑到确定性广义 Büchi 自动机04	165	260	4	0.007	0.28
线性时序逻辑到确定性广义 Büchi 自动机05	361	560	4	0.01	7。105 *
线性时序逻辑到确定性广义 Büchi 自动机06	329	496	4	0.009	703.712
线性时序逻辑到确定性广义 Büchi 自动机07	4368	6656	4	0.399	2085.74 *
线性时序逻辑到确定性广义 Büchi 自动机08	41,071	67,522	4	5.515	超时
线性时序逻辑到确定性广义 Büchi 自动机09	75	120	4	0.004	0.023
线性时序逻辑到确定性广义 Büchi 自动机10	77	118	4	0.003	0.026
线性时序逻辑到确定性广义 Büchi 自动机11	21	30	4	0.002	0.003
线性时序逻辑到确定性广义 Büchi 自动机12	81	120	4	0.003	0.335
线性时序逻辑到确定性广义 Büchi 自动机13	151	222	4	0.009	0。198 *
线性时序逻辑到确定性广义 Büchi 自动机14	97	144	4	0.004	0.065 *
线性时序逻辑到确定性广义 Büchi 自动机15	39	60	4	0.002	0.007
线性时序逻辑到确定性广义 Büchi 自动机16	75	114	4	0.003	0.025
线性时序逻辑到确定性广义 Büchi 自动机17	663	1020	4	0.009	213.754
线性时序逻辑到确定性广义 Büchi 自动机18	265	420	4	0.009	11.288
线性时序逻辑到确定性广义 Büchi 自动机19	106	152	4	0.004	0。652 *
线性时序逻辑到确定性广义 Büchi 自动机20	1078	1760	4	0.018	587.085
线性时序逻辑到确定性广义 Büchi 自动机21	2197	3380	4	0.043	超时
线性时序逻辑到确定性广义 Büchi 自动机22	21	30	4	0.008	0.025
线性时序逻辑到确定性广义 Büchi 自动机23	51	78	4	0.008	0.073
线性时序逻辑到确定性广义 Büchi 自动机24	19	26	4	0.002	0.011
线性时序逻辑到确定性广义 Büchi 自动机25	104	158	4	0.004	0.074
线性时序逻辑到确定性广义 Büchi 自动机26	148	236	4	0.003	0.19
线性时序逻辑到确定性广义 Büchi 自动机27	21	30	4	0.005	0.059
线性时序逻辑到确定性广义 Büchi 自动机 alpha	63	96	4	0.003	0.128
线性时序逻辑到确定性广义 Büchi 自动机 beta	907	1454	4	0.022	341.418
线性时序逻辑到确定性广义 Büchi 自动机 C2	19	26	4	0.003	0.005
线性时序逻辑到确定性广义 Büchi 自动机 E	19	26	4	0.003	0.013
线性时序逻辑到确定性广义 Büchi 自动机 Q	46	68	4	0.003	0.077
线性时序逻辑到确定性广义 Büchi 自动机 R	19	26	4	0.002	0.003
线性时序逻辑到确定性广义 Büchi 自动机 theta	275	440	5	0.011	112.133
线性时序逻辑到确定性广义 Büchi 自动机 U1	61	92	4	0.009	0.142
线性时序逻辑到确定性奇偶自动机01	102	164	5	0.004	0.52
线性时序逻辑到确定性奇偶自动机02	87	138	4	0.007	0.073
线性时序逻辑到确定性奇偶自动机03	8135	14，280	6	2.481	超时
线性时序逻辑到确定性奇偶自动机04	40	56	4	0.004	0.017
线性时序逻辑到确定性奇偶自动机05	40	56	4	0.003	0.008
线性时序逻辑到确定性奇偶自动机06	70	100	4	0.003	0。044 *
线性时序逻辑到确定性奇偶自动机07	137	204	4	0.004	0.555
线性时序逻辑到确定性奇偶自动机08	71	112	4	0.003	0.089
线性时序逻辑到确定性奇偶自动机09	137	204	4	0.004	0.556
线性时序逻辑到确定性奇偶自动机10	996	1704	6	0.049	500.069
线性时序逻辑到确定性奇偶自动机11	174	268	4	0.015	0.612
线性时序逻辑到确定性奇偶自动机12	2544	4422	7	0.714	超时
线性时序逻辑到确定性奇偶自动机13	661	1118	5	0.066	317.768
线性时序逻辑到确定性奇偶自动机14	114	184	5	0.006	0.711
线性时序逻辑到确定性奇偶自动机15	121	194	4	0.004	0.696

线性时序逻辑到确定性奇偶自动机16	118	188	4	0.009	0.438
线性时序逻辑到确定性奇偶自动机17	82	128	4	0.008	0。115 *
线性时序逻辑到确定性奇偶自动机18	82	128	4	0.008	0.111 *
线性时序逻辑到确定性奇偶自动机19	841	1392	5	0.041	9。497 *
线性时序逻辑到确定性奇偶自动机20	340	560	4	0.008	34.289
线性时序逻辑到确定性奇偶自动机21	996	1704	6	0.05	500.785
线性时序逻辑到确定性奇偶自动机22	9351	16,128	6	4.603	超时
ltl2dpa23	443	676	4	0.019	22.212
ltl2dpa24	443	676	4	0.017	22.209
音乐应用反馈	1584	2590	3	0.166	1159.62
音乐应用激励	2501	4118	3	0.261	3064.71
简易音乐应用	344	562	3	0.018	11.77
单计数器图形用户界面A0	12,353	19,052	3	13.995	1720.65
单计数器图形用户界面A1	21,272	32,778	3	38.838	超时
单计数器图形用户界面A2	35,020	54,106	3	104.356	超时
OneCounterGuiA3	62,852	97,434	3	411.197	超时
OneCounterGuiA4	63,275	98,280	3	412.184	超时
OneCounterGuiA5	63,698	99,126	3	405.678	超时
OneCounterGuiA6	69,756	109,186	3	500.2	超时
OneCounterGuiA7	71,106	111,886	3	508.946	超时
OneCounterGuiA8	84,017	133,596	3	747.632	超时
OneCounterGuiA9	87,077	139,716	3	786.04	超时
OneCounterGui	12,353	19,052	3	13.862	1725.76
OneCounterInRangeA0	429	688	3	0.017	7.494
OneCounterInRangeA1	518	832	3	0.02	15.474
OneCounterInRangeA2	534	864	3	0.018	24.461
OneCounterInRangeA3	447	724	3	0.018	8.735
OneCounterInRange	429	688	3	0.033	7.41
单计数器	87,077	139,716	3	805.637	超时
优先级仲裁器	125	196	4	0.009	0.38 *
优先级仲裁器 unreal1	851	1412	4	0.075	10。748 *
优先级仲裁器 unreal2	851	1412	4	0.085	10。739 *
优先级仲裁器 unreal3	10，588	18,196	4	14.389	2273.01 *
寄存器管理器	20	28	3	0.004	0.005
机器人网格	3136	5408	3	0.091	超时
旋转计算器	560	856	3	0.023	0.295
轮询仲裁器	125	190	3	0.007	2.067
轮询仲裁器 unreal1	157	242	3	0.008	3.642
轮询仲裁器 unreal2	157	242	3	0.009	3.689
轮询仲裁器 unreal3	2087	3388	3	0.048	超时
传感器寄存器	12	16	3	0.007	0.007
传感器	7905	13,596	4	8.325	超时
简单仲裁器	662	1184	3	0.039	8.903
简单仲裁器 unreal1	2178	3676	3	0.17	1985.03
简单仲裁器 unreal2	6703	11，846	3	1.408	超时
简单仲裁器 unreal3	45，260	80，600	3	201.107	3202.98
默认滑块	751	1250	3	0.029	80.1
SPI读取时钟	37	54	3	0.002	0.003
SPI读取串行数据输入	178	284	3	0.008	0.477
SPI写入时钟	144	220	3	0.004	0.018
SPIWriteSdi	547	830	3	0.02	1.077
starve.ehoa	13	18	3	0.002	0.003
starve-smart.ehoa	17	22	3	0.002	0.02
test2.ehoa	176	280	4	0.008	7.772
TorcsAccelerating	21	30	3	0.007	0.008
TorcsGearing	27	38	3	0.002	0.006
TorcsSimple	85	130	3	0.003	0.017
TorcsSteeringSimple	189	306	3	0.005	0.263
TorcsSteeringSmart	404	628	3	0.013	2.859
TwoCounters2	39,747	61,044	3	140.334	超时
TwoCounters3	30,511	46,672	3	78.866	超时
TwoCounters4	46,702	72,904	3	195.105	超时
TwoCountersInRangeA0	22,231	34,212	3	42.399	超时
TwoCountersInRangeA1	31，271	48，192	3	84.137	超时
TwoCountersInRangeA2	44，750	69,000	3	179.063	超时
TwoCountersInRangeA3	45,122	69,744	3	185.008	超时
TwoCountersInRangeA4	45，494	70，488	3	183.387	超时
TwoCountersInRangeA5	46，098	71,696	3	189.761	超时

TwoCountersInRangeA6	46，702	72，904	3	192.715	超时
TwoCountersInRangeM0	46，702	72，904	3	196.336	超时
TwoCountersInRangeM1	46，702	72，904	3	194.149	超时
范围内双计数器M2	46,098	71，696	3	189.046	超时
范围内双计数器M3	46,098	71，696	3	186.365	超时
范围内双计数器M4	46,098	71，696	3	187.458	超时
范围内双计数器M5	46,098	71，696	3	189.278	超时
范围内的双计数器	22，231	34，212	3	42.468	超时
精化后的双计数器（再次精化）	1933	3140	3	0.209	43.752
精化后的双计数器	1961	3196	3	0.217	45.308
双计数器	1216	1970	3	0.086	14.067
下近似示例2	42	60	3	0.003	0.012
下近似示例	20	24	3	0.003	0.007
Zoo10	479	768	3	0.02	2.868
Zoo5	479	768	3	0.021	2.872