arXiv:2602.09197v1 [cs.FL] 9 Feb 2026

会合同步下的全局协议：从可实现性到类型检查

Elaine Li
efl9013@nyu.edu
纽约大学 美国

Felix Stutz
felix.stutz@uni.lu 卢森
堡大学 卢森堡

摘要

全局协议规范是自顶向下验证方法的起点，并作为综合局部规范的蓝图，以确保分布式实现的正确性。在本工作中，我们研究以通过会合同步进行通信的分布式进程为目标的全局协议规范。针对同步可实现性问题，我们得到以下正面结果：(a) 对于基于传递并发字母表的全局协议，可实现性是可判定的，其复杂度相对于协议大小为双指数时间，相对于字母表大小为三指数时间；(b) 对于能够无歧义地表示其迹语言的全局协议，可实现性可在指数时间内判定。无歧义全局协议涵盖了以往工作研究的有向选择和发送方驱动选择的若干片段。此外，我们的归约带来一个推论：同步验证问题也可在相同复杂度下求解，其中候选实现作为输入的一部分给出。随后，我们证明一个负面结果：一般情况下，同步可实现性是不可判定的。最后，我们提出一种类型系统，用于根据同步通信状态机形式的局部规范检查 π-演算进程。据我们所知，这是首个支持在存在会话交错和委托的情况下使用局部混合选择的进程的类型系统。

1 引言

全局协议是一种用于描述消息传递协议的抽象，它从全局视角共同刻画所有参与者的行为。由于简洁易用，全局协议已被业界采用，成为国际电信联盟标准的一部分 [64],；它们以高级消息序列图（HMSC）的形式进行研究[2, 4, 22, 23, 25, 26, 45, 48–50, 53], , 并在多方会话类型（MST）[10, 11, 29, 30, 33, 36, 40, 62, 69] 和编舞式编程 [17, 27, 31, 32, 57] 等编程语言框架中得到广泛应用。此类框架的实现可见于十多种编程语言。

全局协议的一个核心特征是将协议逻辑与通信分离。作为一种纯粹的规约形式，$\mathcal{G}$ 全局协议并不限定特定的网络模型。考虑图 1a 所示的示例协议p、q 和r。协议$\mathcal{G}$ 仅仅表达这样一种意图：所有参与者都交换相同的消息值，或者 m_1 或 m_2，具体选择由参与者p决定。

全局协议旨在最终由一组通过网络彼此通信的分布式进程实现。典型的工作流会根据一个全局协议，为每个参与者生成一份局部规范，用以约束局部实现的行为。图1c、图1d和图1e以以下形式展示了 G 中每个参与者的局部规范：有限自动机，其转移以发送和接收事件标记，例如p ▸ q ! m_1 表示p 向 m_1 发送给q。图1b将每个参与者的实现描绘为一个 π-演算进程。

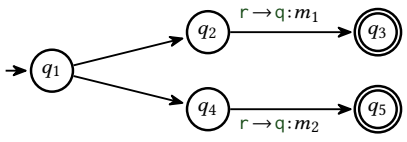
请注意，工作流中的任何组件都没有承诺采用特定的网络模型。如果允许 p，q 和 r 通过会合同步进行通信，那么参与者便会共同遵循 $\mathcal{G}$，恰好呈现所需行为并避免死锁。另一方面，如果允许 p，q 和 r 通过点对点先进先出信道进行异步通信，就会发生违反 $\mathcal{G}$ 的情况：参与者 r无法推断 p所做的选择，因此可以独立发送一条与 p 和 q之间交换的消息不匹配的消息。

网络模型的选择是可实现性问题的核心关注点。该问题旨在确定给定的全局协议是否确实可分布式实现，使其参与者尽管只能掌握整个系统的局部视图，仍能避免通信错误或系统级死锁。可实现性是相对于目标实现和网络模型定义的。文献中对可实现性的研究主要针对异步、可靠的点对点先进先出通信。针对以 HMSC [2, 4, 26, 45, 49, 50],形式表示的全局协议，以及涵盖全局类型的全局通信标记迁移系统 [41], , 已取得可判定性和复杂度结果。

近年来，人们对同步通信模型的兴趣日益浓厚，既关注其表达能力，也关注其可实现性问题 [5, 6, 15, 28, 52]。Peters 和 Yoshida [52] 研究了局部混合选择对同步通信的 π-演算进程表达能力的影响，并为这类进程提出了一种类型系统。局部混合选择允许进程在接收和发送消息之间进行选择。Castro-Perez 等人 [15] 提出了一个同步MST框架，在该框架中，进程相对于全局协议的可赋型性意味着前者正确实现了后者。借此，他们通过对全局协议和类型规则施加限制，对同步可实现问题的一种变体进行了可靠的近似，我们将其称为同步验证问题。近期受到关注的还有 [28], , 它将可实现性问题与可补全性联系起来，并提出了首批用于判定多种全局类型片段的同步可实现性的算法，其中包括发送方驱动的全局类型 [46]。发送方驱动的选择允许发送方在一次选择中向不同的接收方发送消息。

本文的贡献如下。我们的可判定性和复杂度结果总结于图 2 中。

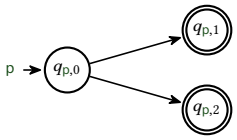
- 我们证明，对于传递并发字母表上的全局协议，同步可实现性是可判定的：按协议大小计，复杂度为双指数时间；按字母表大小计，复杂度为三指数时间。据我们所知，这是



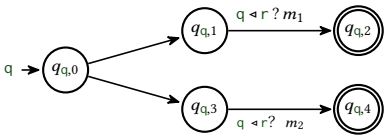
（a）全局协议。

$$P_{\mathsf{p}} = s[\mathsf{p}][\mathsf{q}]\,!\,m_1\,.\,\mathbf{0} + s[\mathsf{p}][\mathsf{q}]\,!\,m_2\,.\,\mathbf{0}$$
$$P_{\mathsf{q}} = +\begin{cases} s[\mathsf{q}][\mathsf{p}]\,?\,m_1\,.\,s[\mathsf{q}][\mathsf{r}]\,?\,m_1\,.\,\mathbf{0} \\ s[\mathsf{q}][\mathsf{p}]\,?\,m_2\,.\,s[\mathsf{q}][\mathsf{r}]\,?\,m_1\,.\,\mathbf{0} \end{cases}$$
$$P_{\mathsf{r}} = s[\mathsf{r}][\mathsf{q}]\,!\;\; m_1\,.\,\mathbf{0} + s[\mathsf{r}][\mathsf{q}]\,!\;\; m_2\,.\,\mathbf{0}$$

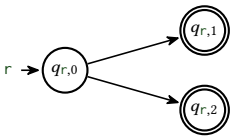
（b）会话中的进程 s。



（c）p的有限自动机。



（d）q的有限自动机。



（e）有限自动机，识别 r。

图 1：全局协议（a）、作为有限自动机的局部规范（c、d 和 e），以及作为 π-演算进程的实现（b）。

传递并发字母表下迹语言等价性和包含问题的首个上界。此外，我们还根据参与者集合、消息值集合和通信拓扑，对传递字母表进行了全面的分类讨论。

- 我们证明，对于能够无歧义地 表示其迹语言的全局协议，同步可实现性可在指数时间内判定；证明方法是将其归约为有理迹语言的重数等价问题。
- 我们的归约使我们得以断定，同步验证问题也能以相同的复杂度求解。- 我们证明，同步可实现性一般情况下不可判定。
- 我们提出了一种类型系统，用于检查 π-演算进程是否符合有限状态自动机；这是首个能够同时处理局部混合选择、会话交错和委托的类型系统。

我们的研究受到两条不同研究脉络的启发。第一条研究脉络运用 Mazurkiewicz 迹理论，研究消息序列图（MSC）和 HMSC。具体而言，归约的目标是可识别的 迹语言；这是一类性质良好的语言片段，对应于 I-闭正则语言的集合。相应地，研究者提出了各种限制，以确保 MSC 和 HMSC 属于可识别片段，例如环连通性和有界性 [45, 49, 50]。此外，已有研究给出了 Kleene 刻画和 MSO 可定义性刻画 [24]。第二条研究脉络 [39, 40, 60] 则处于运用全局协议设计编程语言和验证框架的背景之下。与标准工作流不同，这些研究将全局协议的可实现性问题与综合局部规范的问题分开处理。这种分离使目标实现模型的表达能力得以扩展。

2 预备知识

词。 设 Σ 为一个字母表。 Σ^* 表示 Σ上的有限字集合， Σ^ω 表示无限词集合，而 Σ^∞ 表示它们的并集 $\Sigma^* \cup \Sigma^\omega$。$\Sigma^+$ 表示 Σ上的非空有限字集合，即 $\Sigma^+ = \Sigma^*\Sigma$。若存在 $w \in \Sigma^\infty$ 使得 $u \cdot w = v$，则字 $u \in \Sigma^*$ 是字 $v \in \Sigma^\infty$的前缀 ， 记作 $u \le v$；我们将 u 的所有前缀记为 $\mathrm{pref}(u)$，我们按通常方式将其扩展到字集合 $\mathrm{pref}(S)$。我们用 $\mathit{alph}(u)$表示 u中的符号集合。

消息字母表。设 P 为有限参与者集合，V 为有限数据域。我们将同步字母表定义为事件集合 $\Gamma := \{\mathsf{p} \to \mathsf{q} : m \mid \mathsf{p}, \mathsf{q} \in \mathsf{P}, \mathsf{p} \neq \mathsf{q} \text{ and } m \in \mathsf{V}\}$，其中 $\mathsf{p} \to \mathsf{q} : m$表示 p 与 q 之间交换一条 m消息。对于每个参与者 $\mathsf{p} \in \mathsf{P}$，我们定义字母表 $\Gamma_{\mathsf{p}} = \{\mathsf{p} \to \mathsf{q} : m \mid \mathsf{q} \in \mathsf{P} \;\; m \in \mathsf{V}\} \cup \{\mathsf{q} \to \mathsf{p} : m \mid \mathsf{q} \in \mathsf{P} \;\; m \in \mathsf{V}\}$，它由 p 参与的事件组成。我们定义一个同态 $\Downarrow_{\Gamma_{\mathsf{p}}}$，其中 $x\Downarrow_{\Gamma_{\mathsf{p}}} = x$ if $x \in \Gamma_{\mathsf{p}}$，否则为 ε。我们用 $\mathsf{P}(w)$ 表示将 w中的事件投影到其参与者，用 $\mathsf{V}(w)$ 表示将 w 中的事件投影到其消息。

有限自动机。字母表 Σ 上的有限自动机是一个元组 $A = (Q, \Sigma, \delta, q_0, F)$，其中 Q是有限非空状态集，$\delta\colon Q \times \Sigma \to Q$ 是转移关系， $q_0 \in Q$ 是初始状态， $F \subseteq Q$ 是终态集合。A识别的语言定义为 $\mathrm{L}(A) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$，其中 δ^*是 δ的自反传递闭包。我们用 $q_1 \xrightarrow{\alpha} q_2$ 表示$q_1 \xrightarrow{w}{}^* q_2$，用 $q_1 \xrightarrow{w}{}^* q_2$ 表示 $q_2 \in \delta^*(q_1, w)$。当且仅当 δ是一个函数时，我们称 A为确定的，并将确定（非确定）有限自动机分别简称为确定有限自动机（非确定有限自动机）。当 A 对于每个 $q \in Q$和 $\alpha \in \Sigma$都有定义时，我们称其为完备。自动机 A是约化的，当且仅当对于 $q \in Q$中的每个状态，都存在 $u \in \Sigma^*$使得 $\delta(q_0, u) = q$，并且存在 $v \in \Sigma^*$使得 $\delta(q, v) \in F$。注意，有限自动机总是可以通过保语言变换转化为一个确定且约化的有限自动机。

全局协议。 我们使用同步字母表 Γ 上的约化确定有限自动机（DFA）来对全局协议建模，并将其记作 G。

将协议限制到参与者。 给定一个全局协议 $\mathcal{G} = (Q, \Gamma, \delta, q_0, F)$，我们为每个参与者定义一个非确定有限自动机 $\mathcal{G}_{\mathsf{p}}$， p通过将定义域限制到 Γ_{p}:

$$\mathcal{G}_{\mathsf{p}} := (Q, \Gamma_{\mathsf{p}} \uplus \{\varepsilon\}, \delta_{\mathsf{p}}, q_0, F) \text{ with } \delta_{\mathsf{p}} := \{q \xrightarrow{l\Downarrow_{\Gamma_{\mathsf{p}}}} q' \mid q' \in \delta(q, l)\}\ .$$

可实现性是相对于我们的目标实现模型定义的，该模型是同步通信状态机。我们的实现模型与 [14],中的松散协作系统 相同

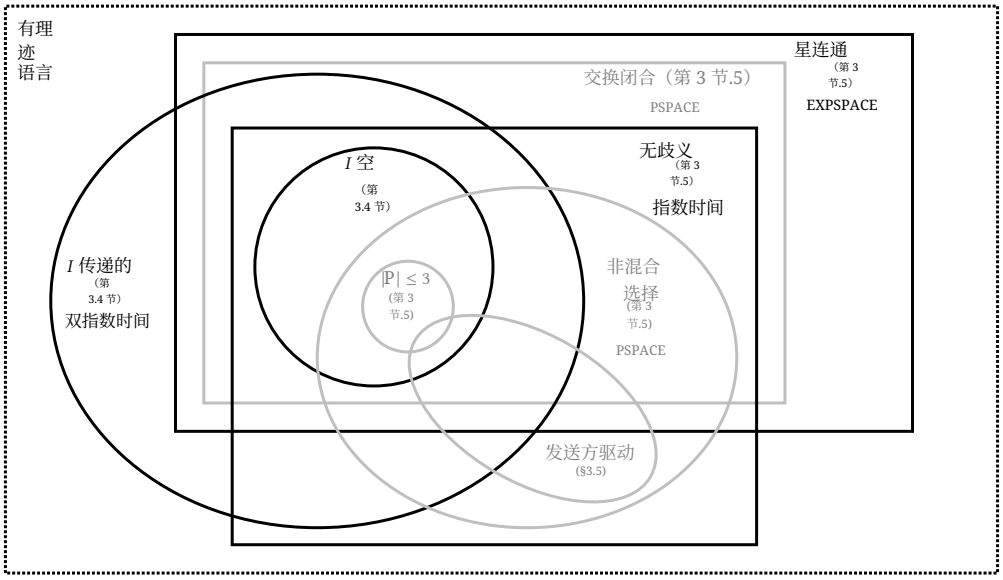


图2：正面结果的语境化。已知结果 [28] 以灰色表示。集合表示由正则字符串语言（存在性地）表示的迹语言。对于没有复杂度注释的每个集合，其复杂度是所有超集中的最小值。

其语义与 [28], 中通信有限状态机的同步运行相同，但不同于Zielonka的异步自动机 [70]，（在 [14], 中也称为同步通信系统）：各个有限自动机在通过事件同步时不会读取彼此的状态。

同步通信状态机。若 $A = \{\{A_p\}\}_{p\in\mathcal{P}}$ 是定义在 P 和 V 上的同步通信状态机（SCSM），则对于每个 $p \in P$，A_p 都是定义在 Γ_p 上的确定有限自动机，记为 $(Q_p, \Gamma_p, \delta_p, q_{0,p}, F_p)$。A 的一个配置是全局状态 $\vec{q} \in \prod_{p\in P} Q_p$。我们用 $\vec{q}_p$ 表示 $\vec{q}$ 中 p 的状态。转移关系记为 $\rightarrow$，定义如下：

$$\vec{q} \xrightarrow{p\to q:m} \vec{q}' \text{ if } (\vec{q}_p, p\to q:m, \vec{q}'_p) \in \delta_p,\ (\vec{q}_q, p\to q:m, \vec{q}'_q) \in \delta_q,$$
以及 $\vec{q}_r = \vec{q}'_r$ 对于每个 r 满足 $p \neq r \neq q$.

对于有限自动机，我们使用 $\xrightarrow{l}{}^*$ 表示自反传递闭包，并且可以省略后态。在初始配置 $\vec{q}_0$中，每个参与者的状态都是初始状态$q_{0,\,p}$属于 A_p。若配置$\vec{q}$满足最终 当且仅当 $\vec{q}_p$对每个 $p \in \mathcal{P}$都是最终状态。运行和轨迹按预期方式定义，且 可达集$(\mathcal{A})$ 表示可达配置的集合。[1] 运行若有限且以最终配置结束，则称为极大的。A 的语言 $\mathcal{A}$记作 $\mathcal{L}(\mathcal{A})$，定义为极大轨迹的集合。配置 $\vec{q}$若为死锁，则它不是最终配置且没有出边。若 SCSM 不存在死锁可达配置，则称其无死锁。

SCSM 的同步积。给定一个定义在 P 和 V 上的 SCSM A，其中 $A_p = (Q_p, \Gamma_p, \delta_p, q_{0,p}, F_p)$ 对每个 $p \in P$ 都是一个定义在 Γ_p 上的确定有限自动机，我们定义一个确定有限自动机 $Prod(A) = (\Pi_{p\in\mathcal{P}} Q_p, \Gamma, \delta, q_0, F)$，其中

- $q_0 = (q_{0,\,p})_{p\in\mathcal{P}}$,
- $F = \Pi_{p\in\mathcal{P}} F_p$，并且
- $(\vec{q}, p\to q:m, \vec{q}') \in \delta$当且仅当 $(\vec{q}p, p\to q:m, \vec{q}'p) \in \delta p, (\vec{q}q, p\to q:m, \vec{q}'q) \in \delta q$, 且对于所有满足 $p \neq r \neq q$ 的 r，均有 $\vec{q}r = \vec{q}'r$。

同步协议语义。 协议 G 的同步语义定义为所有按参与者分别与 L(G) 中某个字相等的字的集合：

$$\mathcal{L}_{sync}(\mathcal{G}) := \{w \in \Gamma^* \mid \exists w' \in \mathcal{L}(\mathcal{G}).\ \forall p \in \mathcal{P}.\ w{\Downarrow_{\Gamma_p}} = w'{\Downarrow_{\Gamma_p}}\}$$

定义 *2.1*（同步可实现性问题）。 如果以下两个性质成立，则 SCSM A 是协议 G 的同步实现：(i) 协议保真性：L(A) = L_{sync}(G)，以及 (ii) 无死锁性：A 是无死锁的。给定协议 G，同步可实现性问题 询问是否存在一个作为 G 的同步实现的 SCSM A。

对于给定的 SCSM 作为输入，也可以提出同步实现问题。我们将此问题称为同步验证问题。

定义 *2.2*（同步验证问题）。给定一个协议 $\mathcal{G}$ 和一个 SCSM $\mathcal{A}$，同步验证问题要判断 $\mathcal{A}$ 是否同步实现 $\mathcal{G}$。

在 MST 文献中，这两个问题通常分开研究，后者则被重新表述为一种子类型问题。在[39],中，研究表明，异步可实现性问题的可靠且完备的解法可用于推导异步协议验证问题的可靠且完备的解法。我们对同步设定下可实现性问题的解法与这一观察相呼应。事实上，我们将在第 3 节中证明，这两个问题的复杂度具有相同的上界。

若上下文清楚，我们可以省略“同步”一词。

[1] 我们使用“迹”一词同时指 SCSM 的执行轨迹以及自由部分交换幺半群的元素，因为其含义总是可以从上下文中明确。

3 可实现性

3.1 迹理论预备知识

我们从迹理论的预备知识开始。

轨迹与迹语言。设 Σ 为有限字母表，C 为 Σ 上的无自反、对称关系，称为并发关系，或独立关系。序对 (Σ, C) 称为并发字母表。自反且对称的关系 $D = \Sigma \times \Sigma \setminus C$ 称为依赖关系。关系 C 和 D 以预期的方式扩展到字：$(u, v) \in C$ 当且仅当 $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq C$ 且 $(u, v) \in D$ 当且仅当 $(\mathrm{alph}(u) \times \mathrm{alph}(v)) \cap D \neq \emptyset$。给定 $x \in \Sigma^*$，我们记 $C(x) = \{w \in \Sigma^* \mid (w, x) \in C\}$。

并发字母表 (Σ, C) 定义了一个**自由部分交换幺半群**，记作 $M(\Sigma, C)$，也称为**迹幺半群**。它是由 $\equiv_C$（即 C 在词上的自反传递闭包）所诱导的商结构。并发字母表 (Σ, C) 上的迹是 $t \in M(\Sigma, C)$ 中的一个元素，也就是一个词的同余类。我们用 $[w]_C$ 表示包含 $w \in \Sigma^*$ 的迹。

有理、可识别和无歧义迹语言。 令 $L \subseteq \Sigma^*$ 为一个语言，(Σ, C) 为一个并发字母表。语言 L，即我们所称的**字符串语言**，可以通过多种方式定义一个迹语言。由 L **存在地** 定义的迹语言记作 $[L]^{\exists}_C$，它是集合 $\{t \in M(\Sigma, C) \mid t \cap L \neq \emptyset\}$。由 L **全称地** 定义的迹语言记作 $[L]^{\forall}_C$，它是集合 $\{t \in M(\Sigma, C) \mid t \subseteq L\}$。如果 $[L]^{\exists}_C = [L]^{\forall}_C = T$，则称 T 由 L **一致地** 定义，即 $L = \bigcup_{t \in T} t$。此外，如果 L 是正则的，那么 $[L]^{\exists}_C$ 是存在正则迹语言，而 $[L]^{\forall}_C$ 是全称正则迹语言。

按照惯例，我们省略 $\exists$ 下标，用 $[L]_C$ 表示 $[L]^{\exists}_C$。在上下文明确时，我们进一步省略 C，用 $[L]$ 表示 $[L]^{\exists}_C$。

反过来，给定迹语言 $T \subseteq M(\Sigma, C)$，集合 $lin(T) = \bigcup_{t \in T} \varphi^{-1}(t)$ 是 T 的**线性化**，其中 φ 是从 Σ^* 到 $M(\Sigma, C)$ 的典范态射。

设 $T \subseteq M(\Sigma, C)$ 是一个迹语言。我们称 T **是有理的**，当且仅当存在一个正则语言 $L \subseteq \Sigma^*$ 使得 $[L]_C = T$。我们称 T **是可识别的**，当且仅当 $lin(T)$ 是 Σ 上的正则语言。我们称 T **是无歧义有理的**，或简称**无歧义的**，当且仅当存在一个正则语言 L 使得 $T = [L]_C$，并且 $w \in L$ 蕴含 $|lin([w]_C) \cap L| = 1$。后一条件等价于 $t \in M(\Sigma, C)$ 蕴含 $|lin(t) \cap L| \leq 1$，或者 $t \in T$ 蕴含 $|lin(t) \cap L| = 1$。注意，存在正则与有理的定义相同，而可识别与一致正则的定义相同。

可识别迹语言具有多种代数刻画。其中一种刻画是星连通有理表达式，即克莱尼星号下出现的所有语言都是连通的，也就是说，不允许并发迭代。我们尤其关注迹语言的**字典序表示**这一概念。设 $<$ 是 Σ 上的一个全序。那么，$<$ 会在 Σ^* 上诱导出一个全序的字典序，定义如下：(i) 对每个 $u \in \Sigma^*$，$\varepsilon < u$；(ii) 对每个 $u, v \in \Sigma^*$，$a, b \in \Sigma$，当且仅当 $a < b$，或 $a = b$ 且 $u < v$ 时，$au < bv$。迹语言 $T \subseteq M(\Sigma, C)$ 的**字典序表示** 是集合 $Lex(T) = \{u \in lin(T) \mid \forall v \in [u].\ u \leq v\}$。$Lex(T)$ 中的元素称为相对于 C 和 $<$ 的字典序最小 词。

注 3.1。 有理迹语言最初被定义为 $M(\Sigma, C)$ 的子集构成的最小类，该类在并集、乘积和克莱尼星号等通常的集合运算下封闭。可识别迹语言最初被定义为 $M(\Sigma, C)$ 中可由有限自动机识别的子集。无歧义迹语言可通过无歧义并集、乘积和克莱尼星号进行代数定义，也称为*1-*顺序 或确定性 迹语言。为便于阐述，我们将每个类相对于正则字符串语言的刻画直接视为定义。有关全面论述，请参阅 [19]。

等价性、包含与重数等价。设 A_1、A_2 为两个识别语言 L_1、$L_2 \subseteq \Sigma^*$ 的有限自动机。等价性(Σ, I)问题询问是否 $[L_1] = [L_2]$。包含(Σ, I)问题询问是否 $[L_1] \subseteq [L_2]$。最后，$\sharp$等价性(Σ, I)问题询问是否 $\sharp L_1 = \sharp L_2$，其中 $\sharp L$ 是一个函数 $M(\Sigma, I) \to \mathbb{N}$，使得对于每个迹 $t \in M(\Sigma, I)$，$\sharp L(t) = |\{x \in L \mid [x] = t\}|$，其中 $\mathbb{N}$ 表示自然数集。因此，当每个迹在 L_1 和 L_2 中具有相同数目的代表时，$\sharp L_1 = \sharp L_2$。注意，若 $\sharp L_1 = \sharp L_2$，则 $[L_1] = [L_2]$，而反之仅当 $[L_1]$ 和 $[L_2]$ 分别是由 L_1 和 L_2 所见证的无歧义迹语言时才成立。

3.2 全局协议及其作为迹语言的实现

注意到，全局协议自然定义了并发字母表 $(\Gamma, \equiv)$ 上的有理迹语言，其中 $\equiv$ 定义为：

定义 *3.1*。 给定定义在参与者集合 P 和消息标签集合 V 上的同步字母表 Γ，对称二元关系 $\equiv$ 定义为 $\{(x_1, x_2) \mid \mathrm{P}(x_1) \cap \mathrm{P}(x_2) = \emptyset\}$。

显然，$\equiv$ 在 Γ^* 上定义了一个等价关系，用于刻画仅在调度选择上等价的成对执行轨迹。我们证明，全局协议的同步语义恰好与其语言在 $\equiv$ 下的闭包一致，并且 SCSMs 的语义在 $\equiv$ 下是闭合的。

命题 3.2。 设 $\mathcal{G}$ 为一个全局协议。则有 $\mathcal{L}^{sync}(\mathcal{G}) = [\mathcal{L}(\mathcal{G})]_{\equiv}$，且 $\mathcal{L}^{sync}(\mathcal{G})$ 是并发字母表 $(\Gamma, \equiv)$ 上的有理迹语言，由 $\mathcal{L}(\mathcal{G})$ 存在性地定义。

命题 3.3. 设 $\mathcal{A}$ 为一个 *SCSM*，且 w 是 $\mathcal{A}$ 的一条迹。则对所有 $w' \in [w]_{\equiv}$，w' 是 $\mathcal{A}$ 的一条迹，且 $\mathcal{L}(\mathcal{A})$ 是并发字母表 $(\Gamma, \equiv)$ 上的可识别迹语言。

3.3 规范 SCSM

众所周知，若一个全局协议是可实现的，则其规范候选实现总能作为可实现性的存在性见证 [2, 28, 42]。我们定义全局协议的规范 *SCSM*，并陈述有关其规范性的事实。本节的结果并非新颖，收录于此仅是为了使本文自洽。

定义 *3.4*（规范 *SCSM*）。 设 $\mathcal{G} = (Q, \Gamma, \delta, q_0, F)$ 为一个协议。对每个参与者 $\mathrm{p} \in \mathrm{P}$，我们定义一个有限自动机 $\mathrm{A}_{\mathrm{p}} = (Q_{\mathrm{p}}, \Gamma_{\mathrm{p}}, \delta_{\mathrm{p}}, q_{0,\mathrm{p}}, F_{\mathrm{p}})$，其中

- $\delta(q, a) := \{s_{\prime} \in Q \mid \exists s \in q,\ s \xrightarrow{a} \xrightarrow{\varepsilon}{}^{*} s_{\prime} \in \delta_{\mathrm{p}}\}$，对于每个 $q \subseteq Q$ 和 $a \in \Gamma_{\mathrm{p}}$

• $q_{0,\ \mathsf{p}} \coloneqq \{s \in Q \mid s_0 \xrightarrow{\varepsilon}{}^{*}\ s \in \delta_{\mathsf{p}}\}$, • $Q_{\mathsf{p}} \coloneqq$ 最小不动点 $\subseteq \{q_{0,\ \mathsf{p}}\}\lambda S.\, S \cup \{\delta(s,\ a) \mid s \in S \wedge a \in \Gamma_{\mathsf{p}}\} \setminus \{\emptyset\}$，
• $\delta_{\mathsf{p}} \coloneqq \delta|_{Q_{\mathsf{p}} \times \Gamma_{\mathsf{p}}}$，以及 • $F_{\mathsf{p}} \coloneqq \{q \in Q_{\mathsf{p}} \mid q \cap F \neq \emptyset\}$。
注意，δ_{p} 确保 A_{p} 是确定的且已约化。我们将 A 定义为 G 的规范 SCSM。

规范 SCSM 满足以下局部语言性质。

引理 3.5。设 $\mathcal{G}$ 是一个全局协议，且 $\mathcal{A}$ 是其规范 *SCSM*。那么，对于每个 $\mathsf{p} \in \mathcal{P}$，有限自动机 $\mathcal{A}_{\mathsf{p}}$ 满足：*(i)* $\mathcal{L}(\mathcal{A}_{\mathsf{p}}) = \mathcal{L}(\mathcal{G})^{\Downarrow_{\Gamma_{\mathsf{p}}}}$，以及 *(ii)* $\mathrm{pref}(\mathcal{L}(\mathcal{A}_{\mathsf{p}})) = \mathrm{pref}(\mathcal{L}(\mathcal{G})^{\Downarrow_{\Gamma_{\mathsf{p}}}})$。

显然，根据构造，全局协议的规范 SCSM 至少包含该全局协议所指定的行为。

引理 3.6。设 $\mathcal{G}$ 是一个全局协议，$\mathcal{A}$ 是其规范 *SCSM*。那么，$\mathcal{L}^{sync}(\mathcal{G}) \subseteq \mathcal{L}(\mathcal{A})$。

下面我们将全局协议与其规范 SCSM 联系起来。

引理 3.7。设 $\mathcal{G}$ 是一个全局协议。那么，$\mathcal{G}$ 可实现，当且仅当其规范 *SCSM* 实现了它。

引理 3.7 和引理 3.6 合起来意味着，判定全局协议是否可实现，等价于检查其规范 SCSM 是否无死锁，以及规范 SCSM 的语言是否包含于全局协议语义中。还需注意，由于不存在（无界）消息信道，每个 SCSM 实际上都是一个有限自动机。下面我们定义一个保持 SCSM 语言的有限自动机。

定义 *3.8*（规范同步积）。设 G 是一个全局协议，A 是其规范 SCSM。G 的规范同步积，记为 $Prod(\mathrm{G})$，是 A 的同步积。

我们用规范同步积来刻画全局协议的可实现性。

引理 3.9. 设$\mathcal{G}$是一个全局协议。那么，$\mathcal{G}$ 可实现，当且仅当 $Prod(\mathcal{G})$ 是约化的，并且满足$\mathcal{L}(Prod(\mathcal{G})) \subseteq \mathcal{L}^{sync}(\mathcal{G})$。

细心的读者可能已经注意到，根据引理 3.9，要判定可实现性，只需判定可识别性。

遗憾的是，Sakarovitch [55]给出了以下结果：

定理 3.10. 一般而言，有理迹语言的可识别性是不可判定的。

所选用的归约来自一元字母表上双带单向非确定自动机的普适性问题，Ibarra 在 [35] 中证明了该问题不可判定。事实上，所构造的有理迹语言满足一个更强的性质：它要么是空的，要么不可识别。这个更强的性质表明，该语言具有普适性当且仅当它具有可识别性。该构造可表示为一个非确定有限自动机，并可使用含有四个参与者的同步字母表进行编码。不过请注意，同一构造并不适用于证明可实现性问题的不可判定性。这是因为，对 Post 对应问题中的无效解集合进行编码需要并发迭代，而这会立即导致协议不可实现。我们将在第 3.6 节详细介绍另一种避免并发迭代的归约。

在接下来的两节中，我们将引理 3.9 与命题 3.2 和命题 3.3 中新提出的迹理论刻画相结合，将可实现性问题归约为迹语言上的不同问题。所有证明见第 A 节。

3.4 传递情形

传统上，人们从并发关系的角度研究迹语言。特别是，已有研究表明，在并发字母表上、其并发关系为传递的的有理迹语言类具有许多可判定性。

本节将这些可判定性性质推广到全局协议。在 §3.4.1 中，我们对并发关系为传递的同步字母表进行全面的分类讨论。在 §3.4.2 中，我们提出一种算法，用于判定具有传递并发字母表的全局协议是否可实现；在 §3.4.3 中，我们分析该算法的复杂度。虽然我们的算法基于已有的可判定性证明 [1]，因此并无新意，但据我们所知，此前尚未给出有理迹语言上等价性问题和包含问题的上界。在 §3.4.4 中，我们分析原始证明中的一个漏洞；该证明声称，在传递情形下，有理迹语言的等价性是可判定的。

3.4.1 *Γ 中具有传递性 $\equiv$的子集。*设 G 为一个全局协议，$\Gamma_{\mathcal{G}}$ 为其符号出现在 G 中的 Γ 的子集。设$\equiv_{\mathcal{G}}$ 为将并发关系 $\equiv$（定义 3.1）限制在 $\Gamma_{\mathcal{G}}$上得到的关系。$\equiv_{\mathcal{G}}$ 的传递性取决于 P 和 V 的大小，也取决于 G 中的通信拓扑。注意，在特殊情形下，当 $\equiv_{\mathcal{G}}$ 为空（因而平凡地具有传递性）时，可通过直接检查有限状态自动机的等价性和约简性，在 PSPACE 内判定可实现性。

当 $|\mathbb{P}| \leq 3$时，$\equiv_{\mathcal{G}}$ 为空。当 $|\mathbb{P}| \geq 4$时，我们首先根据通信拓扑是否为完备进行分类讨论，即全局协议中所有参与者对之间都进行通信。

当 $|\mathcal{P}| = 4$ 且通信拓扑为完备时，我们进一步根据 $\mathcal{V}$ 的大小进行分类讨论。设 $\mathcal{P} = \{\mathsf{p},\mathsf{q},\mathsf{r},\mathsf{s}\}$。若 $|\mathcal{V}| = 1$，则 $\equiv_{\mathcal{G}}$ 具有传递性：$\{(\mathsf{p}\to\mathsf{q}:m, \mathsf{r}\to\mathsf{s}:m), (\mathsf{p}\to\mathsf{r}:m, \mathsf{q}\to\mathsf{s}:m), (\mathsf{p}\to\mathsf{s}:m, \mathsf{q}\to\mathsf{r}:m)\}$。当一对参与者交换多个消息值时，以下模式会破坏传递性：

$$\mathsf{p}\to\mathsf{q}:m_1 \equiv_{\mathcal{G}} \mathsf{r}\to\mathsf{s}:m_1 \equiv_{\mathcal{G}} \mathsf{p}\to\mathsf{q}:m_2 \not\equiv_{\mathcal{G}} \mathsf{p}\to\mathsf{q}:m_1 .$$

当 $|\mathbb{P}| \geq 5$, $\equiv_{\mathcal{G}}$ 时，它不具传递性，因为上述模式包含在其中。

当通信拓扑不完备时，若 $|\mathbb{P}| = 4$ 且 $|\mathbb{V}| > 1$，则传递字母表可由以下性质刻画。

命题 *3.11*。设 $\Gamma_{\mathcal{G}}$ 是定义在 $|\mathcal{P}| = 4$ 和 $|V| > 1$ 上的 Γ 子集。那么，当且仅当对于所有 p、q、r、$s \in \mathcal{P}$，若 $\{p, q\} \cap \{r, s\} = \emptyset$，则 $|\{p \to q : m \mid m \in V\} \cap \Gamma_{\mathcal{G}}| \geq 1$ 蕴含 $|\{r \to s : m \mid m \in V\} \cap \Gamma_{\mathcal{G}}| = 0$ 时，$\equiv_{\mathcal{G}}$ 具有传递性。

请注意，如果两个不同的参与者对 (p,r) 和 (p,q) 交换的消息都多于一条，那么 $\Gamma_{\mathcal{G}}$ 至多还可包含 $\mathsf{p}\to\mathsf{s}: m$ 以及 $\mathsf{q}\to\mathsf{r}: m$。此外，必须存在一名“协调者”参与者，与 $\mathcal{P}$. 中的所有其他参与者通信。

最后，我们提及两个重要的非传递$\equiv_{\mathcal{G}}$实例：下面这个同步字母表，其中 $|\mathbb{P}| = 4$ 和 $|\mathbb{V}| = 2$，与一系列不可判定性结果中使用的“回旋镖”并发字母表同构，例如

用于证明有理迹语言的包含问题一般来说是不可判定的 [19]:

$$\Gamma_{abc} = \{\mathsf{p} \to \mathsf{q} : m_1, \mathsf{r} \to \mathsf{s} : m_1, \mathsf{p} \to \mathsf{q} : m_2\} \ .$$

下列带有 $|\mathbb{P}| = 4$ 和 $|\mathbb{V}| = 2$ 的同步字母表，与足以编码任何形如 $(\Sigma_1 \cup \Sigma_2, \Sigma_1 \times \Sigma_2)$ [9], 的并发字母表的“矩形”并发字母表同构；其有理迹语言可以表示为多带单向非写入自动机 [21]:

$$\Gamma_{abcd} = \{\mathsf{p} \to \mathsf{q} : m_1, \mathsf{r} \to \mathsf{s} : m_1, \mathsf{p} \to \mathsf{q} : m_2, \mathsf{r} \to \mathsf{s} : m_2\} \ .$$

3.4.2 可判定性。 我们的可判定性结果如下所述。

定理 3.12。 对于所有其并发关系 $\equiv_{\mathcal{G}}$ 为传递的、定义在 Γ 上的全局协议 $\mathcal{G}$，同步可实现性都是可判定的。

当 $|\mathbb{P}| \leq 3$ 具有空并发关系，且我们在第 3.6 节中的不可判定性归约需要 5 个参与者时，剩余的 $|\mathbb{P}| = 4$ 情形可由定理 3.12 的以下推论部分解决。

推论 *3.13*。对于所有满足 $P = \{p, q, r, s\}$ 的全局协议，当 $|V| = 1$, 或当 $|V| > 1$ 且 $|\{p \to q : m \mid m \in V\} \cap \Gamma_{\mathcal{G}}| \geq 1$ 蕴含 $|\{r \to s : m \mid m \in V\} \cap \Gamma_{\mathcal{G}}| = 0$ 对所有 p、q、r、$s \in P$ 都成立，且 $\{p, q\} \cap \{r, s\} = \emptyset$ 时，同步可实现性是可判定的。

3.4.3 在双指数时间内判定同步可实现性。我们的复杂度结果如下所述。

定理 3.14。 对于所有并发关系 $\equiv_{\mathcal{G}}$ 为传递的全局协议 $\mathcal{G}$，同步可实现性均可判定；当以 $\mathcal{G}$ 的大小为度量时，复杂度为双指数时间；当以 Γ 的大小为度量时，复杂度为三指数时间。

我们的上界通过归约到以下问题 I 得到：给定一个传递的并发字母表 (Σ, C) 和 $L_1, L_2 \subseteq \Sigma^*$， $[L_1]_C^{\forall} \subseteq [L_2]_C^{\exists}$ 是否成立?

我们提出的求解问题 I 的算法，是从引理 2.7 [1], 的证明中提炼出来的。该引理指出，当且仅当一个并发字母表是传递的时，它才属于类型 $\exists = \forall$。当并发字母表属于类型 $\exists = \forall$ 时，关于 (Σ, C) 的存在正则迹语言类与关于 (Σ, C) 的全称正则迹语言类相同。

我们在下文重述并证明引理 2.7（来自 [1] ）。

引理 3.15。 并发字母表(Σ, C) 是 $\exists = \forall$ 类型的，当且仅当 C 是传递的。

下面，我们分析用于证明 [1] 中引理 2.7 的一系列构造的复杂度。我们重点以算法方式描述这些构造并分析其复杂度。关于正确性的证明，请读者参阅原论文。

回顾一下，语言 L 在 $\Delta = \{a_1, \ldots, a_m\}$ 上的 Parikh 像，记为 $\psi(L)$，是 L 在映射 ψ 下的像： $\Delta^* \to \mathbb{N}^m$，该映射定义为 $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_m})$，其中 $|w|_a$ 表示符号 a 在 w 中出现的次数。

以下引理为每个存在正则迹语言构造一个在同一并发字母表上的等价全称正则迹语言。

引理 3.16。 设 (Σ, C) 为一个并发字母表，且 L 是由非确定有限自动机 M 给出的、定义在 Σ 上的正则字符串语言。则可以构造一个正则字符串语言 K，使得 $[L]_C^{\exists} = [K]_C^{\forall}$，其时间和规模复杂度关于 M 的规模呈双指数级，关于 Σ 的规模呈三指数级。

下一个引理将单个语言之间的对应关系推广到同一并发字母表上的语言集合。

引理 3.17。 设(Σ, C) 为一个并发字母表，且 $L_1, \ldots, L_n$ 是 Σ 上由非确定有限自动机 $M_1, \ldots, M_n$ 给出的 n 正则字符串语言。那么，对于任意指标子集 $s \subseteq \{1, \ldots, n\}$，都可以构造 $K_1, \ldots, K_n$ 这些正则字符串语言，使其时间和规模复杂度关于 M_i 的规模为双指数级，关于 Σ 的规模为三指数级。

以下引理描述了 [1] 中引理 2.5 所述自动机构造的复杂度。直观而言，给定一个由非确定有限自动机 A 表示的、定义在字母表 Σ_1, Σ_2 的不交并上的正则语言 L，我们构造一个定义在新字母表上的非确定有限自动机；该新字母表由三元组构成，其中包含 A 的状态对，以及 1,2。新的非确定有限自动机以交替方式接受 L 中的词，依次读取由 Σ_1 和 Σ_2 中符号组成的极大子串。新字母表中的每个三元组 (s, s', i) 都表示如下语言的 Parikh 像：在 A 中，将初始状态设为 s、唯一终态设为 s' 后得到的语言与 Σ_i^* 的交集。换言之，(s, s', i) 表示仅使用 Σ_i 中的符号时，能够在 A 中从 s 可达集 s' 的词。由于 Σ_i 中的所有符号都可交换，半线性集与 Σ_i^* 上的语言之间存在双射。下一条引理将详细描述这一构造，并分析其复杂度。

引理 *3.18*。设 $A = (Q, \Sigma, \delta, q_0, F)$ 是一个识别 L Σ_1 Σ_2 Σ $\Delta_1 = Q \times Q \times 1$ $\Delta_2 =$ 的完备确定有限自动机。设 $(,)$ 是 的一个划分。设 $\{\}$、$\{\}$ 和 。设 $\{(_{\mathcal{G}},) (,)$ $\Delta \mid n \geq 1$ $s_1 = q_0$ $s_{n+1} \in F$ $i_j \neq i_{j+1}$ $1 \leq j \leq n-1$ $\varepsilon \in L$, , 且对于 $\}$。 $Q \times Q \times 2$ $\Delta = \Delta_1 \cup \Delta_2$ $K' =$ $s_1\ s_2\ i_1\ \ldots\ s_n\ s_{n+1}\ i_n$

若 ，则令 $K = K' \cup \{\varepsilon\}$，否则令 $K = K'$。令 $\eta \colon \Delta^* \to 2^{\Sigma^*}$ 定义为 $\eta((s, s', i)) = \{w \in \Sigma_i^+ \mid \delta(s, w) = s'\}$。可以用常数个状态构造一个识别 K 的确定有限自动机，且 Δ 关于 Q 的规模是多项式的。对于每个 $(s, s', i) \in \Delta$，可以使用 $|Q|^2$ 个状态构造 $\eta((s, s', i))$ 的有限自动机表示。此外， K 和 η 满足以下性质：*(1)* K 是正则字符串语言；*(2)* η 是正则替换；*(3)* $\eta(K) = L$；*(4)* $\eta(\Delta_1) \subseteq \Sigma_1^+$ 且 $\eta(\Delta_2) \subseteq \Sigma_2^+$；*(5)* $K \cap \Delta^* \Delta_1 \Delta_1 \Delta^* = \emptyset$ 和 $K \cap \Delta^* \Delta_2 \Delta_2 \Delta^* = \emptyset$。

下面的引理描述了如何组合定义在互不相交的并发字母表上的两个迹语言，并直观地给出了定理 3.20 证明中的归纳步骤。

引理 3.19. 设 (Σ_1, C_1) 和 (Σ_2, C_2) 是类型为 $\exists = \forall$ 的两个并发字母表，其中 Σ_1 和 Σ_2 互不相交。设 L 是定义在 $\Sigma_1 \cup \Sigma_2$ 上、由确定有限自动机 $A = (Q, \Sigma_1 \cup \Sigma_2, \delta, q_0, F)$ 给出的正则语言。那么，可以构造一个确定有限自动机 M，使得 $[L]_{C_1 \cup C_2}^{\exists} = [\mathcal{L}(M)]_{C_1 \cup C_2}^{\forall}$；其时间和规模对于 Q 为双指数，对于 Σ 为三指数。

定理 3.20. 以下问题对于 M_1 和 M_2 的规模而言可在双指数时间内判定，对于 Σ 的规模而言可在三指数时间内判定：输入：并发字母表 (Σ, C)，其中 C 是传递的，以及确定有限自动机 M_1, M_2。问题：$[\mathcal{L}(M_1)]_C^{\forall} \subseteq [\mathcal{L}(M_2)]_C^{\exists}$?

作为推论，我们得到以下结论。

推论 3.21. 针对全局协议的同步验证问题是可判定的：对于 $\mathcal{G}$ 和输入 *SCSM*，其复杂度为双指数时间；对于 Σ，其复杂度为三指数时间。

3.4.4 中的引理 *5.2* 存在错误 *[9].* 引理 5.2 提出了以下构造，用于判定传递并发字母表上有理迹语言的等价性。定理 5.2（见 [9]）陈述了这一可判定性结果。设 (Σ, C) 是一个并发字母表，其中 C 是传递的，并设 Σ 由 $C \quad M = Q\ \Sigma\ \delta\ q_0\ F \qquad N\ M = q'\ j\ q'' \mid$ 划分为 $\Sigma_1, \exists\ \alpha\ \sigma'\ z\ \sigma''\ k \in \Sigma^*, \sigma', \ldots, \Sigma_n$。设 $(,,,,)$ 是一个确定有限自动机。定义 $(\ )\ \{(,,)\ ,,,,,$ $\sigma'' \notin \Sigma_j \wedge z \in \Sigma_j^* \qquad 1 \le j \le$，其中存在 $\eta\ \delta \qquad \alpha\sigma' \qquad \sigma''k \in F \qquad N\ M$ $,(q_0,) \quad = q'' \wedge \delta(q'',)\}$。直观而言，$(\ )$ 捕获状态对 (q', q'') 以及一个字母表 Σ_j，使得 M 存在一条运行：它读取一个来自 Σ_j^* 的“纯”字，该字前后各有一个不属于 Σ_j 的符号，并最终到达一个终态。

令 $L_{q'q''}$ 表示由 M 所得到的确定有限自动机接受的语言，其中将 q' 设为初始状态，并将 q'' 设为唯一的终态。令 $P_{q'jq''}$ 表示由 $[L_{q'q''} \cap \Sigma_j^*]_C$ 定义的 Parikh 语言。由于 $L_{q'q''} \cap \Sigma_j^*$ 是置换封闭的，因此它由其 Parikh 像唯一表示。

引理 5.2 声称，对于识别语言 L_1 和 L_2 的两个有限自动机 M_1 和 M_2，以下陈述等价：(a) $[L_1]_C = [L_2]_C$

(b) $\forall j, q_1', q_2', q_1'', q_2''.\ (q_1', j, q_1'') \in N(M_1) \wedge (q_2', j, q_2'') \in N(M_2)$ $\implies P_{q_1' j q_1''} = P_{q_2' j q_2''}$

令 $\Sigma = a\,b\ a_2\ \mathrm{d}\ \mathrm{l} \qquad C \qquad a_1\ a_2\ ,\ a_2\ a_1 \qquad L =$„ an et 为 $\{(,)\ (,)\}$。考虑 $\cdot \{a_1, a_2\} \cdots \{a_1, a_3\}$。那么，$[L]_C = [L]_C$ 显然成立，但识别 L 的自动机不满足(b)。进一步考虑 $L_1 = b \cdot \{a_1, a_2\} \cdot b$，以及 $L_2 = b \cdot \{a_1, a_2\} \cdot b \cdot \{a_1, a_2\}$。$[L_1]_C \neq [L_2]_C$，但分别识别 L_1 和 L_2 的自动机满足(b)。因此，(a)不蕴含(b)，(b)也不蕴含(a)。

3.5 无歧义情形

无歧义迹语言最初是通过对迹语言 [13] 上定义的 $\sqcup$ $\cdot$ 和 * 运算的无歧义限制，以代数方式定义的。

定义 *3.22* $\Sigma\ C \qquad T_1\ T_2 \subseteq ..$ $M\ \Sigma\ C \qquad T_1 \sqcup T_2 \qquad T_1 \cup T_2 \qquad T_1 \cap T_2 = \emptyset\ T_1 \quad T_2$ 设 $(,)$ 是一个并发字母表，且 $,(,)$。那么，当 iff 时，定义为；$\circ$ 定义为 $T_1 \cdot T_2$ iff 对所有 $x, z \in T_1$ 和 $y, w \in T_2, xy = zw \implies = z \wedge y = w$ 均成立；最后，$T_@$ 定义为 T^* iff T 是 $M(\Sigma, C)$ 的一个自由子幺半群 T^* 的基。(Σ, C) 上的无歧义迹语言类，是包含有限集合且对 $\sqcup$、$\circ$ 和 $(-)^@$ 封闭的 $M(\Sigma, I)$ 子集的最小类。

该定义自然也定义了字母表（Γ，$\equiv$）上的无歧义全局协议。然而，当我们转而采用由正则语言定义的迹语言视角时，无歧义迹语言被定义为这样一类语言：存在 一个能够无歧义地表示它的正则语言。给定一个由正则语言 L 定义的迹语言 $[L]_C$，即使 L 不能无歧义地表示它，该迹语言也可能是无歧义的。Sakarovitch [54]证明，在一个并发字母表上，有理迹语言类与无歧义迹语言类相同，当且仅当并发关系是传递的。这意味着，对于表示为 $[L]_C$ 的每个有理迹语言，都可以有效地构造 L'，使其 L' 无歧义地表示 $[L]_C$。然而，在一般情形下，这样的构造并非有效。

尽管如此，如果某个迹语言由一个能够无歧义地表示它的正则语言定义，就可以保证它属于无歧义有理迹语言这一类。下文中，我们称 G 为无歧义全局协议，或称 G 是无歧义的，当且仅当 G 无歧义地表示 $[\mathrm{L(G)}]_{\equiv}$，也就是说，对于每个迹 $t \in M(\Gamma, \equiv)$，t 在 L(G) 中恰有 0 个或 1 个表示。

在第 3.5.1 节中，我们证明了此前针对全局协议中分支选择提出的限制是无歧义性的充分条件。在第 3.5.2 节中，我们证明了无歧义全局协议的可实现性是可判定的。我们证明的关键事实是：对于给定为一个有限状态自动机的可识别迹语言，若该自动机识别其所有线性化形式，则可以在线性时间内计算出一个无歧义表示。在第 3.5.3 节中，我们通过引用无歧义迹语言的重数等价问题的一个上界，陈述了可实现性的复杂度。

3.5.1 选择限制。 在异步场景中，混合选择是说明可实现性通常不可判定的关键 [45, 59]。对于采用更严格选择形式（称为发送方驱动选择 ）的协议 [46], , 可实现性是 co-NP 完全的 [41]。自然，人们也研究了同步设定下的发送方驱动选择，Di Giusto 等人 [18] 证明，发送方驱动选择全局类型的同步可实现性可在 PSPACE 内判定。他们进一步将发送方驱动选择放宽为交换确定性，并证明可实现性仍属于 PSPACE。发送方驱动选择要求协议中同一状态的所有出转换均有一个专属发送方，而交换确定性只要求每一对状态都共享一个参与方。下面给出这三种定义。

定义 3.23。设 $\mathrm{G} = (Q, \Gamma, \delta, q_0, F)$ 是一个全局协议。我们称 G 为混合选择，当且仅当存在 $q \in Q$ 满足 $q \xrightarrow{a} q'$ 且 $q \xrightarrow{b} q''$，使得 $a \equiv b$。我们称 G 为非混合选择，当且仅当对于所有 $q \in Q$，若满足 $q \xrightarrow{a} q'$ 且 $q \xrightarrow{b} q''$，则有 $a \not\equiv b$。我们称 G 为发送方驱动，当且仅当对于所有 $q \in Q$，若满足 $q \xrightarrow{a} q'$ 且 $q \xrightarrow{b} q''$，则 $a = \mathrm{p} \to \mathrm{q}: m$ 且 $b = \mathrm{p} \to \mathrm{r}: m'$，其中存在某个 p、q、r m、m'。

请注意，交换确定性 [28] 与非混合选择的定义一致，而发送方驱动选择是非混合选择的子类。我们将证明，非混合选择使协议属于无歧义全局协议这一类。

引理 3.24. 每个非混合选择全局协议都定义了并发字母表 $(\Gamma, \equiv)$ 上的无歧义迹语言。

然而，无歧义全局协议不一定是非混合选择。因此，非混合选择只是一个充分条件，如图 2 中的严格包含关系所示。

备注 3.2. 存在无歧义的混合选择全局协议，例如以下协议：

$$\{\mathtt{p} \to \mathtt{q}:m \cdot \mathtt{r} \to \mathtt{s}:m \cdot \mathtt{p} \to \mathtt{q}:m_1\} \cup \{\mathtt{r} \to \mathtt{s}:m \cdot \mathtt{p} \to \mathtt{q}:m \cdot \mathtt{p} \to \mathtt{q}:m_2\}\ .$$

3.5.2 可判定性。 我们的可判定性结果陈述如下。

定理 3.25. 无歧义全局协议的同步可实现性是可判定的。

通往定理 3.25 的一个关键观察是，所有可识别迹语言都是无歧义迹语言。根据可识别迹语言的正则字典序表示特征，这一结论可立即得证。我们还给出了无歧义代表的规模上界。

引理 3.26。 设 T 是字母表 (Σ, C) 上的一个可识别迹语言。那么， T 是一个无歧义迹语言。此外，给定一个识别 $lin(T)$ 的自动机，可以在线性时间内计算出一个生成 T 的无歧义正则语言。

根据引理 3.26，当全局协议无歧义地表示其迹语言时，协议保真性相当于检查两个无歧义有理迹语言是否等价。与第 3.4 节不同，我们在那里利用规范性的定义将协议保真性简化为语言包含问题；在这里，我们直接将可实现性简化为等价性。事实上，不能使用包含关系，因为有如下否定结果。

定理 3.27 (定理 5.4.16，出自 [19])。一般而言，无歧义有理迹语言的包含问题不可判定。

相反，由于两个无歧义有理迹语言等价当且仅当它们重数等价，因此等价性是可判定的。重数等价性又可以通过归约到两个单变量有理形式幂级数相等来判定；更多详情请见 [19]。

3.5.3 复杂度。下文给出了无歧义全局协议的同步可实现性和验证问题的复杂度。

定理 3.28。 无歧义全局协议的同步可实现性在 *EXPTIME* 中是可判定的。

推论 3.29。 无歧义全局协议的同步验证问题在 *EXPTIME* 中是可判定的。

备注 3.3。 我们通过提及可识别迹语言的情形来结束本节。当 $L(G) = lin([L(G)]_{=})$ 时，可实现性在 PSPACE 中是可判定的，并且该情形与来自 [28], 或 *I*-闭自动机 [70] 的交换闭合全局类型类相一致。当 G 是星连通正规表达式时，得益于来自 [50] 的指数构造，可实现性在 EXPSPACE 中是可判定的。

3.6 不可判定性

本节将展示，同步可实现性一般而言是不可判定的。我们改编了全局协议异步可实现性的不可判定性证明，该证明见 [59], 中的定理 9.1，而后者又受 [45] 中定理 3.4 的启发。在该证明中，将图灵机的接受问题归约为可实现性。该证明构造了一个协议 G_{TM}：当且仅当图灵机 *TM* 不存在接受计算时，它才是可异步实现的。我们事实上证明，G_{TM} 同步可实现，当且仅当 G_{TM} 可异步实现。此外，编码 G_{TM} 是汇终态的，这意味着终态没有出转换。

定义 *3.30*（汇点终结性）。设 $A = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$ 是一个有限自动机。若对每个终态 $q \in F$，都不存在 $l \in \Sigma$ 和 $q' \in Q$ 使得 $q \xrightarrow{l} q'$，则称 A 为汇终态。

我们介绍异步场景所需的预备知识，完整的定义和证明留待 §B 给出。

异步场景中的字母表和通信状态机。一个同步事件可以拆分为发送事件和接收事件，从而得到异步事件：$\Gamma_{\mathsf{p}} = \{\mathsf{p} \triangleright \mathsf{q}\,!\,m \mid \mathsf{q} \in \mathrm{P}\ \ m \in \mathrm{V}\} \cup \{\mathsf{p} \triangleleft \mathsf{q}\,?\,m \mid \mathsf{q} \in \mathrm{P}\ \ m \in \mathrm{V}\}$ 和 $\hat{\Gamma} = \bigcup_{\mathsf{p} \in \mathrm{P}} \Gamma_{\mathsf{p}}$。同态 $\mathrm{split}(\mathsf{p} \to \mathsf{q} : m) \coloneqq \mathsf{p} \triangleright \mathsf{q}\,!\,m.\, \mathsf{q} \triangleleft \mathsf{p}\,?\,m$ 将同步字母表映射到其对应的异步字母表。事件 $\mathsf{p} \triangleright \mathsf{q}\,!\,m$ 表示参与者 p 向 q 发送消息 m，而 $\mathsf{p} \triangleleft \mathsf{q}\,?\,m$ 表示参与者 p 从 q 接收消息 m。一个（异步）通信状态机（CSM）$\mathrm{A} = \{\{A_{\mathsf{p}}\}\}_{\mathsf{p} \in \mathcal{P}}$ 由每个参与者各自的一个确定有限自动机 A_{p} 组成，这些自动机的字母表为 $\hat{\Gamma}_{\mathsf{p}}$，并由点对点先进先出信道连接。显然，可以将一个有限自动机 A 上的每条转移的标签应用 split(-)，再将结果投影到 p 的异步字母表上，从而从一个有限自动机 A 上得到一个有限自动机 Γ_{p}。按预期，这一过程也适用于 SCSM：对于一个 SCSM B，其异步通信状态机记为 $\hat{\mathrm{B}}$。若一个通信状态机或 SCSM A 满足对每个 $\mathsf{p} \in \mathrm{P}$，$A_{\mathsf{p}}$ 都是汇终态，则称 A 为汇终态。通信状态机的异步语义记为 L(A)，其中 A 为通信状态机。全局协议的异步语义根据点对点先进先出通信所诱导的重排序来定义，对于协议 G，记为 $L^{async}(G)$。

定义 *3.31*（异步可实现性问题）。若 CSM A 满足以下两个性质，则称其为协议 G 的异步实现：(i) 协议保真性：$L(A) = L^{async}(G)$；(ii) 无死锁性：A 无死锁。给定协议 G，异步可实现性问题询问是否存在一个 CSM A，使其成为 G 的异步实现。

我们要证明，G_{TM} 可同步实现，当且仅当 G_{TM} 可异步实现。“若”方向可直接由 [18, 定理 5.2] 得出。“仅若”方向则复杂得多。

编码 G_{TM} 的（正则）语言满足一个有趣的性质：每条已发送消息都会得到即时确认。我们将这一概念形式化如下。

定义 3.32。若一个字 $w \in \Gamma^*$ 满足：对于某个 $1 \le i < n$，只要 $w_i = \mathsf{p} \to \mathsf{q} : m$，就有 $w_{i+1} = \mathsf{q} \to \mathsf{p} : m$，则称该字是即时确认的。若有限自动机 A 在 Γ 上的语言 $L(A)$ 中每个字都是即时确认的，则称该有限自动机是即时确认的。若对于每个 $\mathsf{p} \in \mathrm{P}$，$A_{\mathsf{p}}$ 都是即时确认的，则称 SCSM A 是即时确认的。

即时确认意味着，对于任何已发送的消息，在异步场景和同步设定下，其接收方都有权阻止发送方继续采取行动。尽管这是一项有用的性质，但仅有即时确认并不足以保证一个可同步实现的协议也可异步实现。图 1a 所示的协议就是一个反例：关键区别在于，在异步场景中，发送操作始终可执行；而在同步设定下，消息交换中的发送方和接收方都有权否决此次交换。若协议未赋予接收方这项额外的否决权，那么只要该协议可同步实现，它就可异步实现。我们在下文用一个充分条件来表达这一点：对于每条可从一个可达 CSM 配置发送的消息，要么接收方可以立即接收该消息，要么在执行一段立即确认事件序列后，可以接收发送方发出的任意消息值，

我们给出一个中间定义，为充分条件的定义做准备。

定义 3.33。设 A 是一个定义在 Γ_{p} 上的立即确认有限自动机，且 q 是 A 中的一个状态。若存在 $w \in (\bigcup_{\mathsf{p}\neq\mathsf{q}\in\mathcal{P}} m\in\mathcal{V}\ \{\mathsf{p}\to\mathsf{q}:m\cdot \mathsf{q}\to\mathsf{p}:m\})^*$ 使得 $q \xrightarrow{w}{}^* q'$，则称状态 q' 为发送确认可达状态。若不存在任何状态 q'，使得 q' 能从 q 通过 $q' \neq q$ 发送确认可达，则称 q 关于发送确认可达性是极大的。

我们针对同步通信状态机而非协议定义这一充分条件，因为证明是基于某个协议的规范 SCSM 进行推理的。在以下定义中，$\mathcal{V}(\mathcal{A},\ \mathsf{p},\ \mathsf{q})$ 表示在 CSM A中从 p向 q发送的所有消息值。

定义 *3.34*（条件（*X*））。若对每个 $\vec{q} \in$ 可达集$(\mathcal{A})$ 以及 $\mathsf{p},\ \mathsf{r} \in \mathcal{P}$，以下情况之一成立，则同步通信状态机 $\vec{q}_{\mathsf{p}}\ \mathsf{p}\xrightarrow{\to\ m}\mathsf{r}:$ ：

(a) $\vec{q}_{\mathsf{r}}\ \mathsf{p}\xrightarrow{\to\ m}\mathsf{r}:$，或 (b) 对于每个极大发送确认可达$q'_{\mathsf{r}}$，都有 $s'_{\mathsf{r}}\ \mathsf{p}\xrightarrow{\to\ m}\mathsf{r}:$ 对于每个 $m \in \mathcal{V}(\mathcal{A}、\mathsf{p}、\mathsf{r})$。

我们证明，条件（X）是协议实现中的充分条件，可保持从同步设定到异步场景的可实现性。有趣的是，证明无死锁性时才需要条件（X）。

引理 3.35。设 $\mathcal{G}$ 为一个立即确认协议，设 $\mathcal{B}$ 为一个同步实现 $\mathcal{G}$ 的汇点终止的立即确认 *SCSM*。则有 $\mathcal{L}^{async}(\mathcal{G}) = \mathcal{L}(\hat{}\mathcal{B})$。如果 $\mathcal{B}$ 满足条件 (X)，则 $\hat{}\mathcal{B}$ 是异步无死锁的。

最后，我们证明，当 G_{TM} 具有可实现性时，G_{TM} 的规范 SCSM 满足条件（X）。

引理 3.36。如果其规范 *SCSM* $\mathcal{A}$ 同步实现 $\mathcal{G}_{TM}$，则 $\mathcal{A}$ 满足条件（*X*）。

引理 3.35 和 3.36 用于证明该等价关系的“仅当”方向，即 G_{TM} 可同步实现意味着 G_{TM} 可异步实现。借助这一等价关系，我们得到了主要结果。

定理 3.37。汇终态全局协议的同步可实现性是不可判定的。

由于该编码可以表示为高级消息序列图，我们得到以下结论：

推论 3.38。高级消息序列图的同步可实现性是不可判定的。

4 类型系统

本节介绍一种类型系统，用于检查混合选择进程是否符合 SCSM；因此，SCSM 充当全局协议与进程之间的接口。事实上，这使我们的类型系统适用于任意 SCSM，而该 SCSM 不必由全局协议导出，甚至不必与全局协议相关。我们的类型系统将 [60]（其灵感来自 [56]）中的类型系统从异步设定调整为同步设定，并扩展了对混合选择的支持。这是首个支持混合选择进程、并处理会话交错与委托的类型系统。

定义 *4.1.* 进程和进程定义由以下文法定义：

$$
\begin{aligned}
c &::= x \mid s[\mathsf{p}] \\
\pi &::= c[\mathsf{q}]\,!\,l\langle c\rangle \mid c[\mathsf{q}]\,?\,l(y) \\
P &::= \mathbf{0} \mid P_1 \parallel P_2 \mid (\nu s : \mathcal{A})\,P \mid \sum_{i\in I} \pi_i\,.\,P_i \mid \mathsf{Q}[\vec{c}] \\
\mathcal{D} &::= \big(\mathsf{Q}[\vec{x}] = \sum_{i\in I} c[\mathsf{q}_i]\,\pi_i\,.\,P_i\big);\ \mathcal{D} \mid \varepsilon
\end{aligned}
$$

项 c 要么是会话端点 $s[\mathsf{p}]$，要么是变量 x（运行时会解析为会话端点）。按照惯例，$\mathbf{0}$ 表示显式终止，$\parallel$ 表示并行组合。项 $(\nu s : \mathrm{A})$ 限定一个会话 s，其中 SCSM A 提供预期的会话行为；该注释仅用于类型检查，不参与约简。我们采用混合选择（Σ），因此进程可以选择发送消息（$!\ l c$）或接收消息（$?\ l(y)$）。我们假设 $|I| > 0$，并在 $|I| = 1$ 时省略 Σ。借助 $\mathsf{Q}\ [\vec{c}]$，我们指定一个进程调用：Q 是进程标识符，$\vec{c}$ 是其参数。进程定义由 D 给出：例如 $\mathsf{Q}\ [\vec{x}] = \Sigma_{i\in I}\,\pi_i\,.\,P_i$，它提供带守卫的进程定义。用 $\mathsf{D}(\mathsf{Q}\ \vec{c})$ 表示进程定义的展开，即 $\big(\Sigma_{i\in I}\,\pi_i\,.\,P_i\big)\,[\vec{c}/\vec{x}]$。

结构同余是定义进程是否应被视为等价的一种标准方式，例如，带有$\mathbf{0}$ 的并行组合可以被移除（或添加）。

定义 *4.2*。结构同余记作 $\equiv$，由以下规则定义：[2]

- $P_1 \parallel P_2 \equiv P_2 \parallel P_1$
- $(P_1 \parallel P_2) \parallel P_3 \equiv P_1 \parallel (P_2 \parallel P_3)$
- $P \parallel \mathbf{0} \equiv P$
- $(\nu s : \mathrm{A})\ (\nu s' : \mathrm{B})\ P \equiv (\nu s' : \mathrm{B})\ (\nu s : \mathrm{A})\ P$
- $(\nu s : \mathrm{A})\ (P_1 \parallel P_2) \equiv P_1 \parallel (\nu s : \mathrm{A})\ P_2$，若 s 不在 P_1中自由出现

我们将进程的结构预同余 $\sqsubseteq$ 定义为包含 $\equiv$ 和 $(\nu s : \mathrm{A})\ \mathbf{0} \sqsubseteq \mathbf{0}$的最小预同余关系。

当我们让进程进行归约时，允许借助归约上下文深入进程的结构。

定义 *4.3*。我们将归约上下文定义如下：

$$\mathbb{C} ::= \mathbb{C} \parallel P \mid P \parallel \mathbb{C} \mid (\nu s : \mathcal{A})\,\mathbb{C} \mid [\,]\ .$$

我们在图 3 中定义归约规则。规则 PR-exc 允许两个进程的组合交换一条消息：标签相同，并且发送和接收一个值。规则 PR-Q 展开进程定义。规则 PR-ctx 允许我们借助上下文深入进行归约。最后，PR-$\sqsubseteq$ 允许我们将结构预同余进程纳入归约考虑。

定义 *4.4*。进程定义类型上下文 Θ 是一个将进程标识符映射到其参数类型的函数，即 $\Theta\colon \mathsf{Q} \to \vec{L}$。句法类型上下文 由以下文法给出：

$$\Lambda ::= \Lambda, s[\mathsf{p}] : L \mid \Lambda, x : L \mid \emptyset\ .$$

如果每个元素至多具有一种类型，则语法类型上下文就是一个类型上下文，本文仅考虑后者。类型上下文在重新排序意义下被视为等价，因此可以

[2]注意，$\equiv$ 也用于表示第 3 节中的并发关系。$\equiv$的预期含义始终可由上下文明确。

$$\frac{k' \in J \qquad \pi_k = s[\mathsf{p}][\mathsf{q}]\,!\,l_k\langle v_k\rangle \qquad \pi_{k'} = s[\mathsf{q}][\mathsf{p}]\,?\,l_{k'}(y_{k'}) \qquad l_k = l_{k'} \qquad I \cap J = \emptyset \qquad k \in I}{\sum_{i\in I}\pi_i . P_i \parallel \sum_{j\in J}\pi_j . P_j \longrightarrow P_k \parallel P_{k'}[v_k/y_{k'}]}\ \text{PR-EXC}$$

$$\frac{\mathcal{D}(\mathsf{Q},\vec{c}) \parallel P \longrightarrow P'}{\mathsf{Q}[\vec{c}] \parallel P \longrightarrow P'}\ \text{PR-Q} \qquad \frac{P \longrightarrow P'}{\mathbb{C}[P] \longrightarrow \mathbb{C}[P']}\ \text{PR-CTX}$$

$$\frac{P_1 \sqsubseteq P_1' \qquad P_1' \longrightarrow P_2' \qquad P_2' \sqsubseteq P_2}{P_1 \longrightarrow P_2}\ \text{PR-}\sqsubseteq$$

图3：进程的归约规则。

$$\frac{}{\vdash \varepsilon : \Theta}\ \text{PT-DEF-}\varepsilon \qquad \frac{\Theta \shortmid \vec{x} : \vec{L} \vdash P \qquad \Theta(\mathsf{Q}) = \vec{L} \qquad \vdash \mathcal{D} : \Theta}{\vdash (\mathsf{Q}[\vec{x}] = P); \mathcal{D} : \Theta}\ \text{PT-DEF}$$

$$\frac{\Theta(\mathsf{Q}) = \vec{L}}{\Theta \shortmid \vec{c} : \vec{L} \vdash \mathsf{Q}[\vec{c}]}\ \text{PT-Q} \qquad \frac{}{\Theta \shortmid \emptyset \vdash \mathbf{0}}\ \text{PT-0}$$

$$\frac{\Theta \shortmid \Lambda \vdash P \qquad \mathrm{end}(q)}{\Theta \shortmid c : q, \Lambda \vdash P}\ \text{PT-END} \qquad \frac{\Theta \shortmid \Lambda_1 \vdash P_1 \qquad \Theta \shortmid \Lambda_2 \vdash P_2}{\Theta \shortmid \Lambda_1, \Lambda_2 \vdash P_1 \parallel P_2}\ \text{PT-}\parallel$$

$$\frac{\begin{array}{c}\delta(q) = \{(\mathsf{p}\to\mathsf{q}_i : l_i(L_i), q_i) \mid i \in I\} \uplus \{(\mathsf{q}_j \to \mathsf{p} : l_j(L_j), q_j) \mid j \in J\} \\ \forall i \in I.\, \Theta \shortmid \Lambda, c : q_i, \{c_j : L_j\}_{j\in I\setminus\{i\}} \vdash P_i \\ \forall j \in J.\, \Theta \shortmid \Lambda, c : q_j, \{c_i : L_i\}_{i\in I}, y_j : L_j \vdash P_j\end{array}}{\begin{array}{c}\Theta \shortmid \Lambda, c : q, \{c_i : L_i\}_{i\in I} \\ \vdash \sum_{i\in I} c[\mathsf{q}_i]\,!\,l_i\langle c_i\rangle . P_i + \sum_{j\in J} c[\mathsf{q}_j]?l_j(y_j) . P_j\end{array}}\ \text{PT-}\Sigma$$

$$\frac{\vec{q} \in \mathrm{reach}(\mathcal{A}) \qquad \Lambda_s = \{s[\mathsf{p}] : \vec{q}_\mathsf{p}\}_{\mathsf{p}\in\mathcal{P}_\mathcal{A}} \qquad \Theta \shortmid \Lambda, \Lambda_s \vdash P}{\Theta \shortmid \Lambda \vdash (\nu s : \mathcal{A})\, P}\ \text{PT-}\nu$$

图4：进程的类型规则。

将其视为映射。记号 $\{\Lambda_i\}_{i\in I}$表示将 Λ拆分为$|I|$类型上下文。

在任何系统中，对于两个标签和类型 $l_1(L_1)$ 以及 $l_2(L_2)$，我们假设 $l_1 = l_2$ 蕴含 $L_1 = L_2$。其他同步多方会话类型框架在投影全局类型时会通过合并算子确保这一点。有些框架还会将分支决策与值的发送和接收分离，以规避这一根本问题。实际上，这类类型可以是常见的基本类型（如布尔值、整数），也可以是用于委托的信道端点 $s[\mathsf{p}]$。众所周知，可以将基本类型添加到这类类型系统中（且彼此正交），因此，按照 [56],，我们将重点放在信道端点上。

定义*4.5*。我们在图4中给出了进程的类型规则。对于任意状态 q，若 q是终态且没有出边，则 $\mathrm{end}(q)$ 成立。规则 PT-def-ε和 PT-def 用于进程定义，使我们能够推理其参数的类型。要为进程定义指定类型（如 PT-Q 中所示），就需要这些信息。借助 PT-**0**，我们可以为进程 **0**指定类型，但其第二个类型上下文必须为空；这可以通过 PT-end 实现：若 $\mathrm{end}(q)$ 成立，就可以移除 $c : q$。规则 PT-Σ 允许我们为这样的进程指定类型：先将选择划分为发送和接收，再进行处理。使用 PT-‖时，会拆分进程的并行组合及其类型上下文，然后分别为它们指定类型。使用 PT-ν时，会添加会话的类型绑定，其余进程则应使用这些绑定来指定类型。

定义*4.6*。我们定义类型上下文的规约，使其与 SCSM 的语义精确对应：

$$\frac{q_1 \xrightarrow{\mathsf{p}\to\mathsf{q}:I(L)} q_3 \qquad q_2 \xrightarrow{\mathsf{p}\to\mathsf{q}:I(L)} q_4}{s[\mathsf{p}] : q_1, s[\mathsf{q}] : q_2, \Lambda \longrightarrow s[\mathsf{p}] : q_2, s[\mathsf{q}] : q_4, \Lambda}\ \text{TR-}\Sigma$$

命题 4.7。令 Λ_1、Λ_1'和 Λ_2为类型上下文。只要 $\Lambda_1 \longrightarrow \Lambda_1'$成立，就有 Λ_1、$\Lambda_2 \longrightarrow \Lambda_1'$、$\Lambda_2$。

现在，我们可以陈述（并证明）我们的主要性质。主体约化 表明，类型上下文可以匹配进程（使用该上下文进行类型标注）能够执行的任何步骤，并且新的类型上下文可以为新的进程进行类型标注。

定理 4.8（主体约化）。设 P是一个具有会话集合$\mathcal{S}$的进程。如果
(1) $\vdash D : \Theta$, *(2)* $\Theta \shortmid \Lambda \vdash P$ *with* $\Lambda = \Lambda_s\{\Lambda_s\}_{s\in\mathcal{S}}$, *(3)* 对于每个 $s \in S$，都存在 $\vec{q} \in$ 可达集(A_s)，使得$\Lambda_s = \{s[p] : \vec{q}_\mathsf{p}\}_{\mathsf{p}\in\mathcal{P}_\mathcal{A}}$, *and(4)* $P \longrightarrow P'$,

那么存在 Λ' 以及 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \shortmid \Lambda' \vdash P'$.

第二个性质，即进展，考虑相反的方向：如果一个类型上下文为某个进程进行类型标注，并且可以执行一步，那么该进程也可以匹配这一步。按照惯例，由于潜在的依赖关系，我们将进展限制在单会话范围内。已有研究探讨了克服这一限制的正交技术，但这超出了本文的范围 [16, 37]。

为实现这一限制，我们将 $\vdash_{SF}$ 定义为 $\vdash$，但不包含规则 PT-ν。我们将进程的 $\vdash_{SF}$ 定义如下：

$$\frac{\vec{q} \in \mathrm{reach}(\mathcal{A}) \qquad \forall \mathsf{p} \in \mathcal{P}_\mathcal{A}.\, \forall c : q' \in \Lambda_\mathsf{p}.\ \mathrm{end}(q') \qquad \forall \mathsf{p} \in \mathcal{P}_\mathcal{A}.\, \Theta \shortmid \Lambda_\mathsf{p}, s[\mathsf{p}] : \vec{q}_\mathsf{p} \vdash_{SF} Q_\mathsf{p}}{\Theta \shortmid \{\Lambda_\mathsf{p}\}_{\mathsf{p}\in\mathcal{P}_\mathcal{A}} \vdash_{SF}^{\vec{q}} (\nu s : \mathcal{A})\, (\prod_{\mathsf{p}\in\mathcal{P}_\mathcal{A}} Q_\mathsf{p})}\ \text{PT-}\nu'$$

注意，结论中使用的是 $\vdash_{SF}^{\vec{q}}$而不是 $\vdash$：我们通过一种简单的方式扩充类型规则，以获得 SCSM 配置。

定理 4.9（进展）。设 $\mathcal{A}$ 是一个汇点终结且无死锁的 *SCSM*。设 P 是一个进程。我们假设
(1) $\vdash_{SF} \mathcal{D} : \Theta$,
(2) $\Theta \shortmid \Lambda \vdash_{SF}^{\vec{q}} (\nu s : \mathcal{A})\, P$，并且
(3) $\vec{q} \to \vec{q}'$对于某个 $\vec{q}'$。
那么，存在 P'，使得 $P \longrightarrow P'$ 满足 $\Theta \shortmid \Lambda \vdash_{SF}^{\vec{q}'} (\nu s : \mathcal{A})\, P'$。

这两个结果的证明见§C。此外，我们还概述了如何由带类型进程的进展和主体约化推出其无死锁性。

5 相关工作

我们围绕两种最密切相关的形式体系——HMSC 和 MST——来组织对相关工作的讨论。

5.1 高层消息序列图

高级消息序列图（HMSC）已在学术界得到广泛研究 [22, 23, 25, 48, 53]，并已在行业标准中得到定义 [64]。有趣的是，大多数关于 HMSC 的研究都考察了异步通信模型。就可实现性而言，已研究了两

个问题：弱可实现性和安全可实现性 [3, 45]。二者都要求协议保真性；前者允许死锁，而后者不允许。Lohrey 证明，这两种可实现性问题通常都是不可判定的 [45]。为使问题变得可处理，研究者提出了对 HMSC 的各种限制。我们讨论能够恢复（安全）可实现性可判定性的限制，且不改变协议行为。有界／正则 HMSC [4, 50]总是存在具有有界信道的实现，从而得到有限状态系统。实现这一点的条件是：每个循环的通信拓扑都强连通。若将此条件放宽为弱连通，则得到 c-HMSC [49] 或全局协作 HMSC [26]。对于正则 HMSC 和全局协作 HMSC，Lohrey [45] 证明了可实现性问题是 EXPSPACE 完全的。Lohrey [45] 还引入了所谓的 I-闭 HMSC，其中 I 是提升至原子 MSC 上的独立关系 ≡（定义 3.1），并证明了可实现性问题是 PSPACE 完全的。

如 §1 所述，Mazurkiewicz 迹理论已被证明有助于解决 HMSC 上的其他判定问题。大多数研究将关于可识别迹语言的结果加以迁移。Muscholl 和 Peled [50] 证明，对于 MSC 图，竞争问题和合流问题一般是不可判定的；但对于对应于可识别迹语言的局部同步图，这些问题是 EXPSPACE 完全的。一般而言，异步 FIFO 协议语义无法定义为某个非自反对称二元字母关系的闭包。作者通过禁止通信模式 $(p \to q:\ m)^*$来规避这一问题；直观而言，该模式要求对 FIFO 信道中尚未接收的消息进行计数。接着，通过要求 MSC 具有环连通性来确保可识别性；这类似于正则表达式的星连接性，后者已被证明能够刻画可识别性 [19]。Muscholl 和 Peled 还证明，NFA 的环连通性判定是 co-NP 完全的。有趣的是，我们也呼应了 [50]中的观察：迹理论中的复杂度结果寥寥无几。

相关问题是判断有限自动机识别的语言是否对 I 闭合；[51] 证明了该问题是 PSPACE 完全的。I 闭包问题要求：给定一个定义在某个字母表和并发关系上的有限自动机，判断该自动机的语言是否在并发关系所诱导的等价关系下闭合。Morin [49] 研究了星连接 HMSC，这一限制同样确保了可识别性，并由此得出可实现性可立即判定。Genest 等人 [24] 证明了一个克莱尼定理，将单子二阶逻辑、（异步）通信状态机和全局协作的组合式 MSC 图联系起来。鉴于上述结果，我们将可识别迹语言的可判定性结果迁移到我们设定中的可实现性问题（第 3.5 节），但不声称这是我们的原创贡献。在我们的工作中，我们研究了某种意义上与可识别性正交的方面：无歧义性，以及独立字母表的传递性。具体而言，我们与无歧义迹语言之间的联系，灵感来自多方会话类型从二元会话类型继承而来的句法选择限制，即定向选择；后来，Majumdar 等人 [46] 将其推广为发送方驱动选择，Giusto 等人 [28] 则将其推广为交换确定性 。

5.2 多方会话类型

多方会话类型框架通过类型保证不会发生通信错误。第一个多方会话类型框架针对异步通信进程 [33]，，不久之后又出现了一个同步变体 [8]。多方会话类型框架通常结合全局类型和局部类型上的句法限制、用于从全局类型计算局部类型的投影算子，以及能够进一步排除局部实现中不良行为的类型系统，来处理可实现性问题。这种混合方法以完备性换取效率，通过可投影性这一概念对可实现性进行可靠的近似。比较解决可投影性和可实现性等不同问题的算法既困难，也没有意义。我们重点讨论：（1）将我们的结果与其他针对多方会话类型可实现性的判定过程进行比较；（2）自顶向下的多方会话类型框架；（3）具有混合选择的自底向上多方会话类型框架；以及（4）合成多方会话类型和编舞自动机。请注意，（2）是经典的多方会话类型方法，而（3）和（4）则是较新的发展。

判定可实现性。 据我们所知，[28] 是唯一针对全局类型的同步可实现性问题提出解决方案的工作。我们在第 1 节讨论了他们的贡献，在第 3.5 节说明了他们的片段与我们的片段之间的关系，并在图 2 中展示了他们的工作与我们贡献之间的联系。对于异步场景，[40] 采用了与我们类似的方法：将可实现性与类型检查分开处理。他们提供了首个完备的投影算子，用于判定发送方驱动的汇点终结全局协议的可实现性。后来，同一批作者证明了该问题是 co-NP 完全的 [41]。他们的 CSM 实现可以使用 [60]提出的类型系统进行类型检查。

自顶向下的多方会话类型框架。从图结构来看，句法上的多方会话类型对应于树形、汇终有限自动机，其环仅存在于叶节点与其祖先之间。Stutz 和 D’Osualdo [60] 证明，只有汇点终结性会限制表达能力。全局类型、局部类型和进程还受到额外的分支选择限制。我们的工作表明，就可实现性的可判定性而言，选择的作用是确保全局协议无歧义。我们的局部进程不再受选择限制。我们将讨论范围限定于同步多方会话类型框架，并请读者参阅 [34] 和 [66] 中的综述。大多数同步多方会话类型框架 [8, 20, 61, 63, 67, 68] 都对全局类型施加定向选择：每次选择中都有唯一的一对发送方和接收方，且不同选择中的消息各不相同。在局部层面，还施加了匹配限制：进程在每种情形下只能发送或接收，并且只能有一个通信伙伴。在这类经典框架中，一个称为投影算子的偏函数会隐式地尝试解决可实现性问题；也就是说，如果每个参与者的投影都有定义，那么该全局类型就是可实现的。在这一过程中，会使用合并算子来“合并”参与者未参与分支选择的分支。大多数此类框架采用所谓的普通合并算子；如果参与者未参与决策，该算子便不允许其在不同分支中表现出不同的行为，从而禁止参与者之后获知分支情况。 [63]采用所谓的完全合并算子，改进了这种情况，但例如在合并时仍无法展开递归。还有可能出现以下情况：可实现的全局类型没有定义投影，且已有研究表明，基于合并的方法本质上是不完备的 。

sender-driven 或混合选择仍不受支持；全局类型、局部类型和进程均是如此。

自底向上的多方会话类型框架与混合选择。 尽管前述工作采用经典的“自顶向下”方法， [56] 则舍弃全局类型，通过模型检测来确立局部类型的性质，这种方法被称为“自底向上”方法。他们的类型系统仅在不存在全局类型时，保证局部类型与进程之间的操作对应性。我们的类型系统间接受到 [56]的启发，但在此基础上扩展了混合选择，并针对从全局协议获得的无死锁 SCSM 对进程进行类型检查。Peters 和 Yoshida [52] 也舍弃了全局类型，但他们的类型系统支持局部类型和进程中的混合选择。然而，他们的类型系统只能为单个会话赋予类型，因此不考虑会话交错或委托。在会话类型中，子类型为进程中行为的实现方式提供了灵活性。通常的看法是，发送方可以发送比规定更少的消息，而接收方可以接收比规定更多的消息。在存在广义选择的情况下，情况未必如此——至少在异步场景中是这样 [39]。Peters 和 Yoshida [52] 提出了一种子类型机制，将混合选择拆分为发送部分和接收部分，并对每个部分分别应用上述通常的看法。遵循我们的解耦规范，我们的类型系统不提供子类型规则；相反，通过同步验证问题来处理一种保留完整协议语义的子类型变体。

合成多方会话类型与编排自动机。Castro-Perez 等人近期的一项工作 [15] 舍弃了局部类型（而非全局类型），并研究了同步验证问题的一种变体：直接针对全局类型对进程进行类型检查。他们的类型系统仅支持单会话，因此其进程与我们的同步通信状态机（SCSM）实现模型非常接近，但不允许接收方从不同的发送方接收消息。[15]中的协议语义被简单定义为带标记迁移系统（LTS）的字语言，而不是正则语言在等价关系下的闭包。请注意，当 LTS 是有限状态的（且只有在这种情况下，已知该类型系统是可判定的）时，这意味着其语义只是一个正则语言。因此，可实现性问题和验证问题都可以在 PSPACE 中得到可靠且完备的判定（见注 3.3 ） —— 对于一种表达能力更强的实现模型而言——事实上，这些问题也是 PSPACE 困难的。随后，可以利用该实现对局部进程进行类型检查。然而，作者采用了另一种混合方法，通过结合全局协议的良态条件与类型规则限制，对验证问题进行可靠的近似。其中一项条件要求 LTS 在 $\equiv$ 下闭合，但仅在特定条件下成立。遗憾的是，实现模型的语义在 $\equiv$ 下闭合，因此，任何未在 $\equiv$ 下闭合的 LTS 都显然不可实现。Barbanera 等人 [5] 将编排自动机引入为一种全局协议规范，其语义与 [18], 一样，不在任何关系下闭合。因此，编排自动机的语义也是正则语言，适用于判定可实现性和验证的相同特征刻画也同样适用。Di Giusto 等人 [18] 通过以下方式近似可实现性问题对编排自动机施加良好序列性和良好分支性条件，并指出良好序列性蕴含在 $\equiv$ 下闭合。值得注意的是，研究表明，所提出的可实现性条件在异步 [47] 和同步 [6] 场景下均存在缺陷。

6 讨论

最后，我们将讨论对两个彼此独立且正交的无限设定进行扩展的可能性。

请注意，本文引用的所有迹理论结果都将语言定义为有限字的集合。第一个扩展涉及全局协议及其实现的无限语义，即当 $L_{sync}(G) \subseteq \Gamma^* \cup \Gamma^\infty$ 对于有限 Γ时。本文中，全局协议和 SCSM 语义都是 Γ^* 的子集。 [46], 的作者为全局协议定义了无限语义，其本质是一个所有状态均为接受状态的 Büchi 接受条件。另一方面，点对点通信状态机也导出了一种无限字的定义，其中无限字就是所有无限执行迹。Li 和 Wies [44] 指出，[46] 中引入并在后续工作 [39, 40, 58]中采用的全局协议无限字语义，实际上与预期实现模型的无限字语义并不匹配，因此提出了修订后的无限字语义。Stutz [58] 指出，约化全局协议“无需额外付出”便可保证其无限行为具备某些性质；这一事实可直接推广到我们的设定。

第二个扩展涉及具有无限多个状态和转移的全局协议，即 Q 和 Γ 为无限集的情形。这类全局协议通常使用逻辑谓词以符号方式表示，相关研究见 [41, 43, 65, 69]。Li 等人 [41, 43] 利用一种本质上属于（反）可达性归约的语义刻画，判定符号协议的异步可实现性。该刻画具有足够的普适性，能够统一导出有限状态和无限状态全局协议的判定过程。至于我们工作所采用的“直接”方法如何扩展到以符号方式表示的无限状态协议，目前尚不明确。另一方面，我们的直接方法能够立即为 验证问题给出判定过程和上界。我们将这一探索留待未来工作。

参考文献

[1] IJsbrand Jan Aalbersberg 和 Emo Welzl。1986 年。由正则字符串语言定义的迹语言。*RAIRO Theor. Informatics Appl.* 20，2（1986），103–119。https://doi.org/10 1051 1986200201031 。/ITA/ Rajeev Alur、Kousha Etessami 和 Mihalis Yannakakis。2003 年。消息序列图的推断。*IEEE Trans. Software Eng.* 29，7（2003），623–633。https://doi.org/10 1109 2003 1214326 。/TSE。. Rajeev Alur、Kousha Etessami 和 Mihalis Yannakakis。2005 年。MSC 图的可实现性与验证。*Theor. Comput. Sci.* 331，1（2005），97–114。https: //doi.org/10 1016 2004 09 034 。/J.TCS。. . Rajeev Alur 和 Mihalis Yannakakis。1999 年。消息序列图的模型检测。载于 *CONCUR '99*：并发理论，第 *10* 届国际会议，荷兰埃因霍温，*1999* 年 *8* 月 *24* 日至 *27* 日，会议录（计算机科学讲义，第 *1664* 卷），Jos C. M. Baeten 和 Sjouke Mauw（编）。Springer，114–129。https://doi.org/10 1007 3 540 48320 9_10 。/ - - - Franco Barbanera、Ivan Lanese 和 Emilio Tuosto。2020 年。编舞自动机。载于 协调模型与语言——*IFIP WG 6.1* 第 *22* 届国际会议 *COORDINATION 2020*，作为分布式计算技术国际联合会议第 *15* 届会议 *DisCoTec 2020* 的一部分举行，马耳他瓦莱塔，*2020* 年 *6* 月 *15* 日至 *19* 日，会议录（计算机科学讲义，第 *12134* 卷），Simon Bliudze 和 Laura Bocchi（编）。Springer，86–106。https://doi.org/10.1007/978- 3-030-50029-0 6[6] Franco Barbanera、Ivan Lanese 和 Emilio Tuosto。2023 年。形式化编排语言理论。*Log. Methods Comput. Sci.*19，3（2023）。https://doi. org/10 46298 19 3 9 2023 。/LMCS- (:) Simon Beier、Markus Holzer 和 Martin Kutrib。2017 年。半线性集上运算的描述复杂度。载于 第 *15* 届自动机与形式语言国际会议论文集，*AFL 2017*，匈牙利德布勒森，*2017* 年 *9* 月 *4* 日至 *6* 日（*EPTCS*，第 *252* 卷），Erzsébet Csuhaj-Varjú、Pál Dömösi 和 György Vaszil（编）。41 55 10 4204 252 8 –。https://doi.org/。/EPTCS。. Andi Bejleri 和 Nobuko Yoshida。. 同步多方会话类型。载于 第一届并发与通信中心软件编程语言方法研讨会论文集，*PLACES@DisCoTec 2008*，挪威奥斯陆，*2008* 年 *6* 月 *7* 日（理论计算机科学电子札记，第 *241* 卷），Vasco T. Vasconcelos 和 Nobuko Yoshida（编）。Elsevier，3–33。https://doi.org/10 1016 2009 06 002 。/J.ENTCS。. Alberto Bertoni、Giancarlo Mauri 和 Nicoletta Sabadini。1982 年。正则迹语言的等价性与成员资格问题。载于 自动机、语言与编程，第 *9* 届研讨会，丹麦奥胡斯，*1982* 年 *7* 月 *12* 日至 *16* 日，会议录（计算机科学讲义，第 *140* 卷），Mogens Nielsen 和 Erik Meineche Schmidt（编）。Springer，61 71 10 1007 0012757 –。https://doi.org/。/ BFB Laura Bocchi、Romain Demangeon 和 Nobuko Yoshida。. 多方多会话逻辑。载于 可信全局计算——第 *7* 届国际研讨会，*TGC 2012*，英国泰恩河畔纽卡斯尔，*2012* 年 *9* 月 *7* 日至 *8* 日，修订精选论文（计算机科学讲义，第 *8191* 卷），Catuscia Palamidessi 和 Mark Dermot Ryan（编）。Springer，97–111。https://doi.org/10.1007/978-3-642-41157-1 7[11] Laura Bocchi、Kohei Honda、Emilio Tuosto 和 Nobuko Yoshida。2010 年。分布式多方交互的契约式设计理论。载于 *CONCUR 2010*——并发理论，第 *21* 届国际会议，*CONCUR 2010*，法国巴黎，*2010* 年 *8* 月 *31* 日至 *9* 月 *3* 日，会议录（计算机科学讲义，第 *6269* 卷），Paul Gastin 和 François Laroussinie（编）。Springer，162–176。https: //doi.org/10 1007 978 3 642 15375 4_12 。/ - - - - Daniel Brand 和 Pitro Zafiropulo。. 关于通信有限状态机。*J. ACM* 30, 2 1983 323 342 10 1145 322374 322380 （），–。https://doi.org/。/。Danilo Bruschi、Giovanni Pighizzini 和 Nicoletta Sabadini。. 关于最小异步自动机的存在性以及无歧义正则迹语言的等价性问题。*Inf. Comput.* 108，2（1994），262–285。https://doi.org/10 1006 1994 1010 。/INCO。. Ilaria Castellani、Madhavan Mukund 和 P. S. Thiagarajan。1999 年。从全局规范综合分布式迁移系统。载于 软件技术与理论计算机科学基础，第 *19* 届会议，印度钦奈，*1999* 年 *12* 月 *13* 日至 *15* 日，会议录（计算机科学讲义，第 *1738* 卷），C. Pandu Rangan、Venkatesh Raman 和 Ramaswamy Ramanujam（编）。Springer，219 231 10 1007 3 540 46691 6_17 –。https://doi.org/。/ - - - David Castro-Perez 、 Francisco Ferreira 和 Sung-Shik Jongmans。2026 年。一种多方会话类型的综合重构。*Proc.ACMProgram. Lang.* 10，POPL（2026 50 1 50 29 10 1145 3776692 ），：–：。https://doi.org/。/ Mario Coppo、Mariangiola Dezani-Ciancaglini、Nobuko Yoshida 和 Luca Padovani。2016 年。动态交错多方会话的全局进展。*Math. Struct. Comput. Sci.* 26，2（2016），238–302。https://doi.org/10.1017/ S0960129514000188[17] Luís Cruz-Filipe 和 Fabrizio Montesi。2020 年。编舞式编程核心模型。*Theor. Comput. Sci.* 802（2020），38–66。https://doi.org/10.1016/j. tcs.2019.07.005[18] Cinzia Di Giusto、Etienne Lozes 和 Pascal Urso。2025 年。多方会话类型的可实现性与可补性。载于 第 27 届国际 *25* 声明式编程原理与实践研讨会（*PPDP*），’11，共 12 页。https://doi.org/10 1145 3756907 3756918 . / . Volker Diekert 和 Grzegorz Rozenberg（编）。1995。《迹之书》。世界科学出版社。https://doi.org/10 1142 2563 . / Burak Ekici、Tadayoshi Kamegai 和 Nobuko Yoshida。2025。为多方会话进程形式化主体约化与进展。载于第 *16* 届交互式定理证明国际会议，*ITP 2025*，*1*，*2025* 年 *9* 月 *28* 日至 *10* 月 *352* 日，冰岛雷克雅未克（*LIPIcs*，第 卷），Yannick Forster 和 Chantal Keller（编）。Schloss Dagstuhl - Leibniz 信息学中心，19:1–19:23。https: //doi.org/10 4230 2025 19 . / LIPICS.ITP. . Patrick C. Fischer 和 Arnold L. Rosenberg。1968。多带单向非写入自动机。*J. Comput. Syst. Sci.* 2, 1 (1968), 88–101。https://doi.org/10.1016/S0022- 0000 68 80006 6 ( ) - Thomas Gazagnaire、Blaise Genest、Loïc Hélouët、P. S. Thiagarajan 和 Shaofa Yang。2007。因果消息序列图。载于*CONCUR 2007*——并发理论，第 *18* 届国际会议，*CONCUR 2007*，葡萄牙里斯本，*2007* 年 *9* 月 *3* 日至 *8* 日，论文集（计算机科学讲义，第 *4703* 卷），Luís Caires 和 Vasco Thudichum Vasconcelos（编）。Springer，166–180。https://doi.org/10. 1007 978 3 540 74407 8_12 / - - - - Blaise Genest 和 Anca Muscholl。2005。消息序列图：综述。载于第五届并发在系统设计中的应用国际会议（*ACSD 2005*），*2005* 年 *6* 月 *6* 日至 *9* 日，法国圣马洛。IEEE计算机学会，2–4。https: // doi.org/10 1109 2005 25 . /ACSD. . Blaise Genest、Anca Muscholl 和 Dietrich Kuske。2004。一类通信自动机的克莱尼定理及其有效算法。载于语言理论进展，第八届国际会议，*DLT 2004*，新西兰奥克兰，*2004* 年 *12* 月 *13* 日至 *17* 日，论文集（计算机科学讲义，第 *3340* 卷），Cristian Calude、Elena Calude 和 Michael J. Dinneen（编）。Springer，30–48。https://doi.org/10 1007 978 3 540 30550 7_4 . / - - - - Blaise Genest、Anca Muscholl 和 Doron A. Peled。2003。消息序列图。载于《并发与佩特里网讲义》，《佩特里网进展》*[*这部教程文集源自 *2003* 年 *9* 月在德国艾希施泰特举行的第 *4* 届佩特里网高级课程 *ACPN 2003*。除 *ACPN 2003* 上的讲座外，文集还约请撰写了其他章节*]*（计算机科学讲义，第 *3098* 卷），Jörg Desel、Wolfgang Reisig 和 Grzegorz Rozenberg（编）。Springer，537 558 10 1007 978 3 540 27755 2_15 – 。https://doi.org/ 。 / - - - - Blaise Genest 、 Anca Muscholl 、 Helmut Seidl 和 Marc Zeitoun。2006。无限状态高级 MSC：模型检验与可实现性。*J. Comput. Syst. Sci.* 72, 4 2006 617 647 10 1016 2005 09 007 ( ), – 。https://doi.org/ 。 /J.JCSS. . .[27] Saverio Giallorenzo、Fabrizio Montesi、Marco Peressotti、David Richter、Guido Salvaneschi 和 Pascal Weisenburger。2021。多方语言：编舞与多层情形（珍珠）。载于第 *35* 届欧洲面向对象编程会议，*ECOOP 2021*，*2021* 年 *7* 月 *11* 日至 *17* 日，丹麦奥胡斯（线上会议）（*LIPIcs*，第 *194* 卷），Anders Møller 和 Manu Sridharan（编）。Schloss Dagstuhl - Leibniz 信息学中心，22:1–22:27。https://doi.org/10.4230/ LIPIcs.ECOOP.2021.22[28] Cinzia Di Giusto、Étienne Lozes 和 Pascal Urso。2025。多方会话类型的可实现性与可补全性。载于第 *27* 届声明式编程原理与实践国际研讨会论文集，*PPDP 2025*，意大利伦德，*2025* 年 *9* 月 *10* 日至 *11* 日，Malgorzata Biernacka、Carlos Olarte、Francesco Ricca 和 James Cheney（编）。ACM，11:1–11:12。https://doi.org/10 1145 3756907 3756918 . / . Jonas Kastberg Hinrichsen、Jesper Bengtson 和 Robbert Krebbers。2020。Actris：基于会话类型的分离逻辑推理。*Proc. ACM Program. Lang.* 4, POPL (2020 6 1 6 30 10 1145 3371074), : – : . https://doi.org/ . / Jonas Kastberg Hinrichsen、Jesper Bengtson 和 Robbert Krebbers。2022。Actris 2.0：基于异步会话类型的分离逻辑推理。*Log. Methods Comput. Sci.* 18, 2 2022 10 46298 18 2 16 2022 ( )。https://doi.org/ . /LMCS- (:) Andrew K. Hirsch 和 Deepak Garg。 。Pirouette：高阶类型化函数式编舞。*CoRR* abs/2111.03484 (2021)。arXiv:2111.03484 https://arxiv.org/abs/2111 03484 。Andrew K. Hirsch 和 Deepak Garg。2022。Pirouette：高阶类型化函数式编舞。*Proc. ACM Program. Lang.* 6, POPL (2022), 1–27。https://doi.org/10 1145 3498684 . / Kohei Honda、Nobuko Yoshida 和 Marco Carbone。2008。多方异步会话类型。载于第 *35* 届 *ACM SIGPLAN-SIGACT* 编程语言原理研讨会论文集，*POPL 2008*，美国加利福尼亚州旧金山，*2008* 年 *1* 月 *7* 日至 *12* 日，George C. Necula 和 Philip Wadler（编）。ACM，273–284 10 1145 1328438 1328472 。https://doi.org/ 。 / 。Hans Hüttel、Ivan Lanese、Vasco T. Vasconcelos、Luís Caires、Marco Carbone、Pierre-Malo Deniélou、Dimitris Mostrous、Luca Padovani、António Ravara、Emilio Tuosto、Hugo Torres Vieira 和 Gianluigi Zavattaro。2016。《会话类型与行为契约基础》。*ACM Comput. Surv.* 49, 1 (2016), 3:1–3:36。https://doi.org/10 1145 2873052 . / Oscar H. Ibarra。 。反转次数受限多计数器机及其判定问题。*J. ACM* 25, 1 (1978), 116–133。https://doi.org/10.1145/322047. 322058

[36] Jules Jacobs、Jonas Kastberg Hinrichsen 和 Robbert Krebbers。2023。基于第一性原理的分离逻辑依赖会话协议（函数式珍珠）。*Proc. ACM Program. Lang.* 7, ICFP (2023), 768–795。https://doi.org/10.1145/ 3607856[37] Naoki Kobayashi。2006。无死锁进程的新型类型系统。载于*CONCUR*（计算机科学讲义，第 *4137* 卷）。Springer，233–247。https: //doi.org/ 10 1007 11817949_16 . / Eryk Kopczynski 和 Anthony Widjaja To。2010。文法的 Parikh 像：复杂度与应用。载于第 *25* 届 *IEEE* 逻辑与计算机科学年度研讨会论文集，*LICS 2010*，*2010* 年 *7* 月 *11* 日至 *14* 日，英国爱丁堡。IEEE计算机学会， 80 89 10 1109 2010 21 – 。[38] https://doi.org/ 。 /LICS. . Elaine Li、Felix Stutz 和 Thomas Wies。。判定异步多方会话的子类型关系。载于编程语言与系统——第 *33* 届欧洲编程研讨会，*ESOP 2024*，作为欧洲软件理论与实践联合会议 *ETAPS 2024* 的一部分举行，卢森堡卢森堡市，*2024* 年 *4* 月 *6* 日至 *11* 日，论文集，第一部分（计算机科学讲义，第 *14576* 卷），Stephanie Weirich（编）。Springer，176–205。https://doi.org/ 10 1007 978 3 031 57262 3_8 . / - - - - 2024[39] Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。2023。基于自动机的完备多方会话类型投影。载于计算机辅助验证——第 *35* 届国际会议，*CAV 2023*，法国巴黎，*2023* 年 *7* 月 *17* 日至 *22* 日，论文集，第三部分（计算机科学讲义，第 *13966* 卷），Constantin Enea 和 Akash Lal（编）。Springer， 350 373 10 1007 978 3 031 37709 9_17 – 。[40] https://doi.org/ 。 / - - - - Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。。刻画无限状态与数据全局协议的可实施性。*Proc. ACM Program. Lang.* 9, 1 2025 1434 1463 10 1145 3720493 OOPSLA ( ), – 。[41] https://doi.org/ 。 / Elaine Li 、 Felix Stutz 、 Thomas Wies 和 Damien Zufferey。。刻画无限状态与数据全局协议的可实施性。arXiv: 2411 05722 [ ] 2411 05722 。cs.PL https://arxiv.org/abs/ 。[42] Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。2025。Sprout：符号多方协议验证器。载于计算机辅助验证——第 *37* 届国际会议，*CAV 2025*，克罗地亚萨格勒布，*2025* 年 *7* 月 *23* 日至 *25* 日，论文集，第三部分（计算机科学讲义，第 *15933* 卷），Ruzica Piskac 和 Zvonimir Rakamaric（编）。Springer，304–317。https://doi.org/10.1007/978-3-031-98682- 6 16[44] ¯ Elaine Li 和 Thomas Wies。2025。全局多方协议的经认证可实施性。载于第 *16* 届交互式定理证明国际会议，*ITP 2025*，冰岛雷克雅未克，*2025* 年 *9* 月 *28* 日至 *10* 月 *1* 日（*LIPIcs*，第 *352* 卷），Yannick Forster 和 Chantal Keller（编）。Schloss Dagstuhl - Leibniz 信息学中心， 15 1 15 20 10 4230 2025 15 : – : 。[45] https://doi.org/ 。 /LIPICS.ITP. . Markus Lohrey。。高级消息序列图的可实现性：填补空白。*Theor. Comput. Sci.* 309, 1-3 (2003), 529–554。https://doi.org/10.1016/J. TCS.2003.08.002[46] Rupak Majumdar、Madhavan Mukund、Felix Stutz 和 Damien Zufferey。2021。推广异步多方会话类型中的投影。载于第 *32* 届国际并发理论会议，*CONCUR 2021*，*2021* 年 *8* 月 *24* 日至 *27* 日——线上会议（*LIPIcs*，第 *203* 卷），Serge Haddad 和 Daniele Varacca（编）。Schloss Dagstuhl - Leibniz 信息学中心，35:1–35:24。https: //doi.org/ 10 4230 2021 35 . /LIPICS.CONCUR. . [47] Rupak Majumdar、Madhavan Mukund、Felix Stutz 和 Damien Zufferey。2021。推广异步多方会话类型中的投影。*CoRR* abs/ 2107 03984 2021 2107 03984 2107 03984 。（）。arXiv: 。[48] https://arxiv.org/abs/ 。Sjouke Mauw 和 Michel A. Reniers。。高级消息序列图。载于 *SDL '97* 测试时代：*SDL*、*MSC* 与趋势——第 *8* 届国际 *SDL* 论坛，法国埃夫里，*1997* 年 *9* 月 *23* 日至 *29* 日，论文集，Ana R. Cavalli 和 Amardeo Sarma（编）。Elsevier，291–306。[49] Rémi Morin。2002。消息序列图的可识别集合。载于*STACS 2002*，第 *19* 届理论计算机科学年度研讨会，法国昂蒂布—胡安莱潘，*2002* 年 *3* 月 *14* 日至 *16* 日，论文集（计算机科学讲义，第 *2285* 卷），Helmut Alt 和 Afonso Ferreira（编）。Springer，523–534。https://doi.org/ 10 1007 3 540 45841 7_43 . / - - - Anca Muscholl 和 Doron A. Peled。[50] 。消息序列图与 Mazurkiewicz 迹上的判定问题。载于数学计算机科学基础 *1999*，第 *24* 届国际研讨会，*MFCS'99*，波兰什克拉尔斯卡-波伦巴，*1999* 年 *9* 月 *6* 日至 *10* 日，论文集（计算机科学讲义，第 *1672* 卷），Miroslaw Kutylowski、Leszek Pacholski 和 Tomasz Wierzbicki（编）。Springer， 81 91 10 1007 3 540 48340 3_8 – 。https://doi.org/ 。 / - - - Doron A. Peled、Thomas Wilke 和 Pierre Wolper。1998。检验时序逻辑规范和 ω 正则语言闭包性质的算法方法。*Theor. Comput. Sci.* 195, 2 (1998), 183–203。https://doi.org/ 10 1016 0304 3975 97 00219 3 . /S - ( ) - Kirstin Peters 和 Nobuko Yoshida。[51] 。混合选择多方会话中的分离性与可编码性。载于第 *39* 届 *ACM/IEEE* 逻辑与计算机科学年度研讨会论文集，*LICS 2024*，爱沙尼亚塔林，*2024* 年 *7* 月 *8* 日至 *11* 日，Pawel Sobocinski、Ugo Dal Lago 和 Javier Esparza（编）。ACM，62:1–62:15。https://doi.org/ 10 1145 3661814 3662085 . / 。Abhik Roychoudhury、Ankit Goel 和 Bikram Sengupta。2012。符号消息序列图。*ACM Trans. Softw. Eng. Methodol.* 21, 2 (2012), 12:1–12:44。[52] https://doi.org/ 10 1145 2089116 2089122 . / . Jacques Sakarovitch。。关于正则迹语言。理论计算机科学 52 (1987 59 75 10 1016 0304 3975 87 90080 6), – 。 https://doi.org/ 。 / - ( ) - [53] Jacques Sakarovitch。。有理迹语言的 " 最后 " 判定问题。载于 *LATIN '92*：第一届拉丁美洲理论信息学研讨会，巴西圣保罗，*1992*年*4*月*6*日至*10*日，论文集（计算机科学讲义，第 *583*卷），Imre Simon（编）。Springer，460–473。https://doi.org/10.1007/ BFB0023848[54] Alceste Scalas 和 Nobuko Yoshida。2019。少即是多：重新审视多方会话类型。*ACM* 编程语言会议论文集 3，POPL（2019），30:1–30:29。https: //doi.org/ 10 1145 3290343 . / Gan Shen、Shun Kashiwa 和 Lindsey Kuper。2023。[55] HasChor：面向所有人的函数式编舞编程（函数式珍珠）。*CoRR* abs/2303.00924（2023）。https://doi.org/ 10 48550 2303 00924 2303 00924 . /ARXIV. 。[57] arXiv：。Felix Stutz。。异步多方会话类型可实现性是可判定的——从消息序列图中汲取的经验。载于 第*37*届欧洲面向对象编程会议，*ECOOP 2023*，*2023*年*7*月*17*日至*21*日，美国华盛顿州西雅图（*LIPIcs*，第*263*卷），Karim Ali 和 Guido Salvaneschi（编）。Schloss Dagstuhl - Leibniz 信息学中心，32:1–32:31。https: //doi.org/ 10 4230 2023 32 . /LIPICS.ECOOP. 。Felix Stutz。。异步通信协议的可实现性——选择的力量。博士论文。德国凯撒斯劳滕工业大学。[58] https://kluedo.ub.rptu.de/frontdoor/index/index/docId/ 8077 2025 Felix Stutz 和 Emanuele D'Osualdo。。用于多方协议规范与类型检查的自动机理论基础。载于 编程语言与系统——第*34*届欧洲编程研讨会，*ESOP 2025*，作为软件理论与实践国际联合会议 *ETAPS 2025* 的一部分举行，加拿大安大略省汉密尔顿，*2025*年*5*月*3*日至*8*日，论文集，第二部分（计算机科学讲义，第*15695*卷），Viktor Vafeiadis（编）。Springer，314–346。https://doi.org/ 10 1007 978 3 031 91121 7_13 . / - - - - Dawit Legesse Tirore、Jesper Bengtson 和 Marco Carbone。2023。全局类型的可靠且完备的投影。载于第 *14* 届交互式定理证明国际会议，*ITP 2023*，*2023* 年 *7* 月 *31* 日至 *8* 月 *4* 日，波兰比亚韦斯托克（*LIPIcs*，第*268*卷），Adam Naumowicz 和 René Thiemann（编）。Schloss Dagstuhl - Leibniz 信息学中心，28:1–28:19。https://doi.org/10.4230/LIPICS. ITP.2023.28[62] Bernardo Toninho 和 Nobuko Yoshida。2017。多方会话类型中的数据认证。逻辑与代数方法编程杂志 90（2017），61–83。https: //doi.org/ 10 1016 2016 11 005 . /J.JLAMP. 。。Thien Udomsrirungruang 和 Nobuko Yoshida。2025。自顶向下还是自底向上？同步多方会话类型的复杂度分析。*ACM* 编程语言会议论文集 9, 2025 1040 1071 10 1145 3704872 POPL（），–。[63] https://doi.org/。/国际电信联盟。。*Z.*：消息序列图。技术报告。国际电信联盟。https://www.itu.int/ rec/T-REC-Z.120[65] Martin Vassor 和 Nobuko Yoshida。2024。多方消息传递协议的细化：规范无关理论与实现。载于 第*38*届欧洲面向对象编程会议，*ECOOP 2024*，*2024*年*9*月*16*日至*20*日，奥地利维也纳（*LIPIcs*，第*313*卷），Jonathan Aldrich 和 Guido Salvaneschi（编）。Schloss Dagstuhl - Leibniz 信息学中心，41:1–41:29。https: //doi.org/ 10 4230 2024 41 . /LIPICS.ECOOP. 。[66] Nobuko Yoshida。。基于多方会话类型的编程语言实现。载于 主动对象语言：当前研究趋势，Frank S. de Boer、Ferruccio Damiani、Reiner Hähnle、Einar Broch Johnsen 和 Eduard Kamburjan（编）。计算机科学讲义，第14360卷。Springer， 147 165 10 1007 978 3 031 51060 1_6 –。https://doi.org/。/ - - - - [67] Nobuko Yoshida 和 Lorenzo Gheri。。多方会话类型的温和入门。载于 分布式计算与互联网技术——第*16*届国际会议，*ICDCIT 2020*，印度布巴内斯瓦尔，*2020*年*1*月*9*日至*12*日，论文集（计算机科学讲义，第*11969*卷），Dang Van Hung 和 Meenakshi D'Souza（编）。Springer，73–93。https://doi.org/10.1007/978-3-030- 36987-3 5[68] ¯ Nobuko Yoshida 和 Ping Hou。2024。重新审视"少即是多"：与全局多方会话类型的关联。载于 形式化方法实践：纪念 *Cliff Jones* 文集，第二部分，Ana Cavalcanti 和 James Baxter（编）。计算机科学讲义，第14781卷。Springer，268–291。https://doi.org/10. 1007/978-3-031-66673-5 14_[69] Fangyi Zhou、Francisco Ferreira、Raymond Hu、Rumyana Neykova 和 Nobuko Yoshida。2020。多方协议的静态验证细化。*ACM* 编程语言会议论文集 4, 2020 148 1 148 30 10 1145 3428216 OOPSLA（），：–：。https://doi.org/。/ Wieslaw Zielonka。。有限异步自动机札记。*RAIRO* 理论信息学应用 21，2（1987），99–135。https://doi.org/10.1051/ITA/1987210200991 [70] 1987

A 第 3 节的补充材料——可判定性

引理 3.5。 设 $\mathcal{G}$ 是一个全局协议，$\mathcal{A}$ 是其规范 *SCSM*。那么，对于每个 $\mathsf{p} \in \mathcal{P}$，有限自动机 $\mathcal{A}_{\mathsf{p}}$ 满足：*(i)* $\mathcal{L}(\mathcal{A}_{\mathsf{p}}) = \mathcal{L}(\mathcal{G})^{\Downarrow_{\Gamma_{\mathsf{p}}}}$，以及 *(ii)* $\mathrm{pref}(\mathcal{L}(\mathcal{A}_{\mathsf{p}})) = \mathrm{pref}(\mathcal{L}(\mathcal{G})^{\Downarrow_{\Gamma_{\mathsf{p}}}})$。

证明。命题 (ii) 的两个包含关系可通过对前缀长度进行归纳，并利用 A_{p} 的确定性得出。命题 (i) 可由命题 (ii) 和终态 F_{p} 的定义得出。□

命题 3.2. 设 $\mathcal{G}$ 是一个全局协议。那么，有 $\mathcal{L}^{sync}(\mathcal{G}) = [\mathcal{L}(\mathcal{G})]_{\equiv}$，并且 $\mathcal{L}^{sync}(\mathcal{G})$ 是并发字母表 $(\Gamma, \equiv)$上的有理迹语言，其存在性由 $\mathcal{L}(\mathcal{G})$定义。

证明。 回顾一下， $\sim$定义为：对所有参与者 $\mathsf{p} \in \mathcal{P}$，$w_1 \Downarrow_{\Gamma_{\mathsf{p}}} = w_2 \Downarrow_{\Gamma_{\mathsf{p}}}$。必要性方向由以下事实得出： $\equiv$保持每个参与者对事件的全序。充分性方向则由以下事实得出：每个参与者对事件的全序都唯一确定一个消息序列图（MSC），因此我们有 $msc(w_1) = msc(w_2)$，其中 w_1 和 w_2 都是线性化。证明基于以下事实：同一个 MSC 的所有线性化都可以通过 $\equiv$相互关联。定义请参阅 [45]。 □

引理 3.24。 每个非混合选择全局协议都在并发字母表 $(\Gamma, \equiv)$ 上定义了一个无歧义迹语言。

证明。令 $\mathrm{G} = (Q, \Gamma, \delta, q_0, F)$ 为一个全局协议。反设某个迹 $t \in M(\Gamma, \equiv)$ 有两个表示 $w_1 \neq w_2$ in L(G)。注意，L(G) 指的是由 G 直接定义的正则语言，而不是 G 的同步语义。于是，存在 G 的两个运行，分别接受 w_1和 w_2。由于 $w_1 \neq w_2$，存在一个状态 $s \in Q$，在该状态处，两条接受运行分别发生分歧，并且每个字采用不同的转移。不妨设这些转移分别标记为 $\mathsf{p} \to \mathsf{q} : m$和 $\mathsf{p} \to m'$ $w_1 \Downarrow_{\Gamma_{\mathsf{p}}} \neq w_2 \Downarrow_{\Gamma_{\mathsf{p}}}$ $t = [w_1] = \mathrm{r}$:。 $[w_2]$ 3 2 □ 由此可得 。然而， 。根据命题 。 产生矛盾。

定理 3.12。 对于并发关系 $\equiv_{\mathcal{G}}$ 具有传递性的所有全局协议 $\mathcal{G}$，其同步可实现性都是可判定的。

证明。设 G 是定义在 Γ 上的全局协议， $Prod$(G) 是其规范同步积。$Prod$(G) 的无死锁性（即检查 $Prod$(G) 是否为约化的）可在 PSPACE 内判定。由于 $\Gamma_{\mathcal{G}}$ 是传递的，因此可以判定 $[\mathrm{L}(Prod(\mathrm{G}))]_{\equiv} = [\mathrm{L}(\mathrm{G})]_{\equiv}$。

□

引理 3.15。 并发字母表 (Σ, C) 是 $\exists = \forall$ 型，当且仅当 C 是传递的。

证明。由于 C是传递的，因此它是有限个完备并发关系的不交并。将 C划分为团 C_1、C_2..... C_n，其字母表 Σ_1、Σ_2..... Σ_n构成 Σ的一个划分。证明对 n进行归纳，其中引理 2.3 [1]涵盖基础情形，而引理 2.6 [1]涵盖归纳步骤。引理 2.3 表明，所有并发字母表（Σ、 C）在 C完备时均属于类型 $\exists = \forall$。引理 2.6 表明，如果（Σ_1、 C_1）属于类型 $\exists = \forall$，且（Σ_2、 C_2）属于类型 $\exists = \forall$，并且 Σ_1与 Σ_2不相交，那么（$\Sigma_1 \cup \Sigma_2$、 $C_1 \cup C_2$）属于类型 $\exists = \forall$。□

引理 3.16。 设（Σ、 C） 为一个并发字母表，设 L 为 Σ 上的正则字符串语言，并由非确定有限自动机 M给出。那么，可以构造一个正则字符串语言 K，使得 $[L]_C^{\exists} = [K]_C^{\forall}$，其时间和规模复杂度关于 M 的规模为双指数级，关于 Σ的规模为三指数级。

证明。我们构造 $K = \overline{\psi^{-1}(\psi(\Sigma^*) - \psi(L))}$。由非确定有限自动机表示的正则语言的 Parikh 像，可在关于该自动机大小的多项式时间和关于其字母表大小的指数时间内计算，其半线性集描述的大小与运行时间相当 [38]。半线性集的补运算可在关于其表示大小的双指数时间内完成 [7]。给定一个半线性集，可在关于该半线性集大小的多项式时间内构造一个以它为 Parikh 像的正则语言。最后，以非确定有限自动机表示的正则语言的补运算可在线性时间内完成，时间与该自动机的大小成正比。关于 $[L]_C^{\exists} = [L]_C^{\forall}$ 的证明，见 [1]的引理 2.3。□

引理 3.17。 设(Σ, C)是一个并发字母表，且 $L_1, \ldots, L_n$ 是 n 上的正则字符串语言，由 n 个非确定有限自动机 $M_1, \ldots, M_n$ 给出。那么，可以构造正则字符串语言 $K_1, \ldots, K_n$，使得对于每个指标子集 $s \subseteq \{1, \ldots, n\}$， $[\bigcup_{i \in s} L_i]_C^{\exists} = [\bigcup_{i \in s} K_i]_C^{\forall}$ 的时间和规模复杂度在 M_i 的规模上为双指数级，在 Σ 的规模上为三指数级。

证明。首先，对于每个 $s \subseteq \{1, \ldots, n\}$，我们构造一个 K'_s，使得 $[\bigcup_{i \in s} L_i]_C^{\exists} = [K'_s]_C^{\forall}$。由于 $\{1, \ldots, n\}$ 有 2^n 个子集，且两个非确定有限自动机的并集规模仍为其规模的多项式，因此每个 K'_s 都可以在关于输入非确定有限自动机规模的双指数时间内构造出来；总之，所有 K'_s 均可在双指数时间内构造。我们定义 $K_i = \bigcap_{i \in s, s \subseteq \{1, \ldots, n\}} K_s$。关于如此构造的语言 K_i 的正确性证明，请参阅 [1]中引理 2.4 的证明。□

引理 *3.18.* 设 $A = (Q, \Sigma, \delta, q_0, F)$ 是一个完备的确定有限自动机，识别 L Σ_1 Σ_2 Σ $\Delta_1 = Q \times Q \times \{1\}$ $\Delta_2 =$。设 $(,)$ $Q \times Q \times \{2\}$ $\Delta = \Delta_1 \cup \Delta_2$ $K' =$ $s_1\ s_2\ i_1\ \ldots\ s_n\ s_{n+1}\ i_n$ 是 的一个划分。设 $\{\}$、$\{\}$ 和 。设 $\{(,,)(,,)$ $\in$ $\Delta \mid n \geq 1\ s_1 = q_0\ s_{n+1} \in F$ $i_j \neq i_{j+1}$ $1 \leq j \leq n-1$ $\varepsilon \in L,,$ *and* f *or* $\}$。若 ，则令 $K = K' \cup \{\varepsilon\}$，否则令 $K = K'$。令 η*:* $\Delta^* \to 2^{\Sigma^*}$ 定义为 $\eta((s, s', i)) = \{w \in \Sigma_i^+ \mid \delta(s, w) = s'\}$。可以用常数个状态构造一个识别 K 的确定有限自动机，并且 Δ 关于 Q 的大小是多项式的。对于每个 $(s, s', i) \in \Delta$，可以使用 $|Q|^2$ 个状态构造 $\eta((s, s', i))$ 的有限自动机表示。此外，K 和 η 满足以下性质：*(1)* K 是正则字符串语言；*(2)* η 是正则替换；*(3)* $\eta(K) = L$；*(4)* $\eta(\Delta_1) \subseteq \Sigma_1^+$ 且 $\eta(\Delta_2) \subseteq \Sigma_2^+$；*(5)* $K \cap \Delta^* \Delta_1 \Delta_1 \Delta^* = \emptyset$ 和 $K \cap \Delta^* \Delta_2 \Delta_2 \Delta^* = \emptyset$。

证明。根据构造， K的大小显而易见。要计算由有限自动机表示的Δ的元素，需要进行$2*|Q|^2$次自动机交运算。每次交运算都会得到一个具有$|Q|^2$个状态的有限自动机。关于如何验证性质 (1) 至 (5)，请读者参阅 [1]中的引理 2.5 的证明。□

引理 3.19。 设 (Σ_1, C_1)、(Σ_2, C_2)是类型为 $\exists = \forall$ 的两个并发字母表，其中 Σ_1和 Σ_2不相交。设 L是由确定有限自动机 $A = (Q, \Sigma_1 \cup \Sigma_2, \delta, q_0, F)$ 给出的、定义在 $\Sigma_1 \cup \Sigma_2$ 上的正则语言。则可以构造一个确定有限自动机 M，使得 $[L]_{C_1 \cup C_2}^{\exists} = [\mathcal{L}(M)]_{C_1 \cup C_2}^{\forall}$，且其构造所需的时间和规模关于 Q 为双指数级，关于 Σ 为三指数级。

证明。为构造 M，我们首先使用引理 3.18，从 L 构造一个定义在字母表 Δ 上的中间自动机 K，该字母表的大小为 $2 * |Q|$。令 $\Delta = \Delta_1 \cup \Delta_2$，并令 $\Delta_1 = \{a_1, \ldots, a_n\}$ 和 $\Delta_2 = \{b_1, \ldots, b_n\}$。注意，根据构造，$\Delta_1$ 和 Δ_2 的大小相同。对于每个 $a \in \Delta_1$（分别为 $b \in \Delta_2$），我们计算 $\eta(a)$（分别为 $\eta(b)$）的有限自动机表示；根据引理 3.18，该表示的大小为 $|Q|^2$。对于由有限自动机表示的 n 个语言 $\eta(a_1) \ldots, \eta(a_n)$，我们使用引理 3.17 中的构造，找出一列能够普遍表示它们的 n 个语言；对于 n 个语言 $\eta(b_1) \ldots, \eta(b_n)$，也进行相同的操作。整个构造需要关于 $|Q|$ 的双指数级时间。关于该构造的正确性，请参阅 [1] 中引理 2.6 的证明。□

定理 3.20。以下问题可在关于 M_1 和 M_2 的大小的双指数时间内，以及关于 Σ 的大小的三指数时间内判定：输入：并发字母表 (Σ, C)，其中 C 是传递的；以及确定有限自动机 M_1, M_2。问题：$[\mathcal{L}(M_1)]_C^\forall \subseteq [\mathcal{L}(M_2)]_C^\exists$?

证明。首先，注意：

$$
\begin{aligned}
[\mathcal{L}(M_1)]_C^\forall \subseteq [\mathcal{L}(M_2)]_C^\exists &\iff \\
[\mathcal{L}(M_1)]_C^\forall \setminus [\mathcal{L}(M_2)]_C^\exists = \emptyset &\iff \\
[\mathcal{L}(M_1) \setminus \mathcal{L}(M_2)]_C^\forall = \emptyset &\iff \\
[N]_C^\exists = \emptyset, \text{ where } [\mathcal{L}(M_1) \setminus \mathcal{L}(M_2)]_C^\forall = [N]_C^\exists &\iff \\
N = \emptyset &
\end{aligned}
$$

设 n 为识别 $\mathrm{L}(M_1) - \mathrm{L}(M_2)$ 的确定有限自动机的状态数，并设 $m = |\Sigma|$。由于 C 是传递的，它是有限多个完全并发关系的不相交并集。将 C 划分为团 $C_1, C_2, \ldots, C_n$，其字母表 $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ 构成 Σ 的一个划分。设 M 为识别 K 的确定有限自动机，该自动机在引理 3.18 中定义，并推广到多于 2 个字母表的情形。于是，M 有 nm 个状态和 n^2m 个转移。我们将 M 的每个转移 a 替换为 $\eta(a)$；根据引理 3.19，其大小关于 n 为双指数，关于 m 为三指数，从而得到自动机 A_N，其状态规模关于 n 仍为双指数，关于 m 仍为三指数。接着，根据引理 3.19，$\mathrm{L}(A_N) = N$。空性判定的时间复杂度与输入自动机的大小呈线性关系，因此整个算法的运行时间关于 n 为双指数，关于 m 为三指数。□

定理 3.14. 对于其并发关系 $\equiv_{\mathcal{G}}$ 具有传递性的所有定义在 Γ 上的全局协议 $\mathcal{G}$，同步可实现性均可判定：相对于 $\mathcal{G}$ 的大小，复杂度为双指数时间；相对于 Γ 的大小，复杂度为三指数时间。

证明。设 G 是定义在 $(\Gamma, \equiv)$ 上的全局协议，且 $Prod(\mathrm{G})$ 是其规范同步积，其语言 $\mathrm{L}(Prod(\mathrm{G}))$ 是一个正则语言，并以存在和全称方式定义相同的迹语言。根据引理 3.9，G 可实现当且仅当 $Prod(\mathrm{G})$ 可达且 $[\mathrm{L}(Prod(\mathrm{G}))]_\equiv^\forall \subseteq [\mathrm{L}(\mathrm{G})]_\equiv^\exists$。根据定理 3.20，此包含问题可判定：相对于 G 的大小，复杂度为双指数时间；相对于 Σ 的大小，复杂度为三指数时间。□

定理 3.25. 无歧义全局协议的同步可实现性是可判定的。

证明。设 G 为一个全局协议，$Prod(\mathrm{G})$ 为其规范同步积。令 A 表示有限状态自动机在 Γ 上识别 $Prod(\mathrm{G})$ 的字典序表示。根据引理 3.26 A 是 $\mathrm{L}(Prod(\mathrm{G}))$ 无歧义性的一个见证。因此，可实现性归约为等价性；对于无歧义的表示，这又归约为重数等价，而根据 [19] 的定理 5.4.10，这是可判定的。□

引理 3.26。设 T 是定义在并发字母表 (Σ, C) 上的可识别迹语言。那么，T 是无歧义迹语言。此外，给定一个识别 $lin(T)$ 的自动机，可以在线性时间内计算出生成 T 的无歧义正则语言。

证明。根据命题 6.3，4, [19], 每个可识别迹语言都有一个正则的字典序表示。由于 T 是可识别的，设 M 是满足 $\mathrm{L}(M) = lin(T)$ 的自动机。设 N 为识别正则语言 $\bigcup\{\Sigma^* bC(a)a\Sigma^* \mid (a, b) \notin C \text{ and } a < b\}$ 的自动机的补自动机。因此，$\mathrm{L}(N)$ 是根据 $<$ 在 Σ 上的所有字典序最小词的集合。将 M 与 N 求交，可得到一个大小与 M 成线性关系、识别 T 的字典序表示的自动机。由于 $<$ 在词上的关系是完备的，T 的字典序表示对于 T 中的每个迹至多包含一个代表元，因此无歧义地表示了 T。□

定理 3.28. 无歧义全局协议的同步可实现性可在指数时间内判定。

证明。回顾一下，规范 SCSM 的无死锁性可以在 PSPACE 内检查。检查协议保真性的算法直接来自我们对重数等价的归约，而重数等价又可以通过小模型性质进行检查。根据定理 5.4 的证明，模型大小受限于表示两个输入迹语言的有限自动机的状态空间之和。令 $\mathrm{G} = (Q, \Gamma, \delta, q_0, F)$。我们首先构造规范同步积 M，其状态空间为 $\mathrm{P}^{|Q|}$，即相对于 G 的大小呈指数级。令 $m = |Q| + \mathrm{P}^{|Q|}$。接下来，我们枚举长度不超过 m 的、基于 $(\Gamma, \equiv)$ 的所有迹，并针对每个迹 $t \in T_m$，检查 t 是否包含于 $\mathrm{L}(\mathrm{G})$ 当且仅当 t 包含于 $\mathrm{L}(M)$。注意，由于 M 是一致正则迹语言，我们可以在线性于 $|t|$ 的时间内检查成员关系。另一方面，G 表示一种有理迹语言，根据命题 5.5，成员关系可以在 $|t|^\alpha$ 时间内检查，其中 α 是 $\equiv$ 中的最大团大小。□

B 第 3 节补充材料——不可判定性

请注意，可以定义以通信模型的选择为参数的通信状态机（参见 [18]）。下文中，我们定义基于点对点先进先出信道的通信，并沿用其在 [12] 中的原始定义，将其简称为 CSM。

通信状态机。若 $= \{\{\mathrm{A}_\mathrm{p}\}\}_{\mathrm{p} \in \mathcal{P}}$ 是定义在 $\mathrm{P}\hat{}$ 和 V 上的 CSM，且对于每个 $\mathrm{p} \in \mathrm{P}$，$A_\mathrm{p}$ 都是定义在 Γ_p 上的有限自动机，记作 $(Q_\mathrm{p}, \hat{\Gamma}_\mathrm{p}, \delta_\mathrm{p}, q_{0,\mathrm{p}}, F_\mathrm{p})$。令 $\prod_{\mathrm{p} \in \mathrm{P}} q_\mathrm{p}$ 表示全局状态集合，$\mathrm{Chan} = \{(\mathrm{p}, \mathrm{q}) \mid \mathrm{p}, \mathrm{q} \in \mathrm{P}, \mathrm{p} \neq \mathrm{q}\}$ 表示通道集合。A 的配置是一个二元组 $(\vec{q}, \xi)$，其中 $\vec{q}$ 是一个全局状态，$\xi : \mathrm{Chan} \to \mathrm{V}^*$ 是一个从每个通道到消息序列的映射。我们用 $\vec{q}_\mathrm{p}$ 表示 $\vec{q}$ 中 p 的状态。CSM 的转移关系记作 $\to$，定义如下。

• $(\vec{q},\xi)\ \mathtt{p} \xrightarrow{\ \mathtt{p}\ \triangleright\ \mathtt{q}\,!\,m\ }_{m} (\vec{q}'\,\xi')$，当且仅当 $(\vec{q}\mathtt{p},\mathtt{p}\ \triangleright\ \mathtt{q}\,!\,m,\vec{q}'\mathtt{p})\ \in\delta\mathtt{p}$，且对于每个参与者 $\mathtt{r}\ \neq\mathtt{p}$，$\vec{q}\mathtt{r}\ =\vec{q}'\mathtt{r}$；$\xi'(\mathtt{p},\mathtt{q})\ =\xi(\mathtt{p},\mathtt{q})\cdot m$，并且对于 Chan 中的每个其他通道 $c\in$，$\xi'(c)\ =\xi(c)$。
• $(\vec{q},\xi)\ \mathtt{q} \xrightarrow{\ \mathtt{q}\ \triangleleft\ \mathtt{p}\,?\,m\ }_{m} (\vec{q}'\,\xi')$ 若 $(\vec{q}\mathtt{q},\mathtt{q}\ \triangleleft\mathtt{p}\,?\,m,\vec{q}'\mathtt{q})\ \in\delta\mathtt{q},\vec{q}\mathtt{r}\ =\vec{q}'\mathtt{r}$ 对每个参与者 $\mathtt{r}\ \neq\mathtt{q}$，$\xi(\mathtt{p},\mathtt{q})\ =m\cdot\xi'(\mathtt{p},\mathtt{q})$，且对于每个其他通道 $c\in\mathrm{Chan}$，$\xi'(c)\ =\xi(c)$。
在初始配置 $(\vec{q}_0,\xi_0)$ 中，每个参与者在 $\vec{q}_0$ 中的状态都是 $q_{0,\mathtt{p}}$ 的初始状态 $A_\mathtt{p}$，以及将每个通道映射到 ε 的 $\xi_0=\xi_\varepsilon$。如果配置 $(\vec{q},\xi)$ 满足：对于每个 $\mathtt{p}\in\mathrm{P}$ 和 $\xi=\xi_\varepsilon$，$\vec{q}_\mathtt{p}$ 均为最终状态，则称其为最终配置。当以预期方式定义运行和轨迹。如果运行是有限的并以最终配置结束，则称其为极大运行。CSM A 的（异步）语言 L(A) 是定义在 $\hat{}\Gamma^*$ 上的词集合，其定义为极大轨迹的集合。如果配置 $(\vec{q},\xi)$ 不是最终配置且没有出转换，则称其为死锁。若没有可达配置是死锁，则称 CSM 为无死锁。

异步协议语义。要定义协议 G 的异步语义，我们首先使用 split(-) 将 G 的有限词和无限词映射到其异步对应形式，从而得到一组异步词，其中匹配的发送和接收事件彼此相邻。

我们将协议语义定义为一组满足通道合规 的词 [46]，它们在这种不可区分性概念下是封闭的。通道合规性刻画了遵循先进先出顺序的词，即接收事件出现在与其匹配的发送事件之后，并且每个通道中的接收事件顺序与发送事件顺序一致。通道合规性是基于词的点对点线性化对应概念，用于描述消息序列图。

定义 *B.1*（通道合规性）。设 $w\in\Sigma^\infty_{async}$。若对于所有前缀 $w'\le w$，以及所有$\mathtt{p}\ \neq\mathtt{q}\in\mathrm{P}$，$\mathrm{V}(w'\Downarrow_{\mathtt{q}\triangleleft\mathtt{p}\,?}_)\ \le\mathrm{V}\,(w'\Downarrow_{\mathtt{p}\triangleright\mathtt{q}\,!}_)$，则称$w$满足通道合规。

协议的异步语义定义如下。

定义 *B.2*（异步协议语义）。设 G 为一个全局协议。其异步语义定义为：

$$\mathcal{L}_{async}(\mathcal{G})=\{w'\in\hat{\Gamma}^*\mid\exists w\in\hat{\Gamma}^*.\,w\in\mathtt{split}(\mathcal{L}(\mathcal{G}))\ \wedge\ w'\text{ is channel-compliant}\ \wedge\ \forall\mathtt{p}\in\mathcal{P}.\,w'\Downarrow_{\hat{\Gamma}_\mathtt{p}}=w\Downarrow_{\hat{\Gamma}_\mathtt{p}}\}\ .$$

在证明中，我们将该条件的一部分用二元关系表示 ~，它定义在 $\hat{}\Gamma$上的词之间。设 $w\in\hat{}\Gamma^*$ 是一个信道合规词，且 $w'\in\hat{}\Gamma^*$。我们定义 $w\sim w'$ 当且仅当 w' 是信道合规的，且$\forall\mathtt{p}\in\mathcal{P}.\,w'\Downarrow_{\hat{\Gamma}_\mathtt{p}}=w\Downarrow_{\hat{\Gamma}_\mathtt{p}}$。

此外，在下面的证明中，我们使用异步发送事件，即$\hat{}\Gamma_!=\{\mathtt{p}\ \triangleright\mathtt{q}!\,m\mid\mathtt{p}\ \neq\mathtt{q}\ \in\mathcal{P}\ \wedge\ m\in\mathcal{V}\}$，它是$\hat{}\Gamma$的子集。

引理 3.35。设$\mathcal{G}$ 为一个立即确认协议，且$\mathcal{B}$为一个汇点终止的立即确认 *SCSM*，它同步实现$\mathcal{G}$。则$\mathcal{L}^{async}(\mathcal{G})\ =\mathcal{L}(\hat{}\mathcal{B})$。若$\mathcal{B}$满足条件$(X)$，则$\hat{}\mathcal{B}$是异步无死锁的。

证明。我们列举这些断言：$\hat{}$(a) L(B) $=\mathrm{L}^{async}$(G)，以及 (b) 若 B 满足条件（X），则 $\hat{}$B 无死锁。

尽管 CSM（分别为 SCSM）的语义始终是异步的（分别为同步的），我们仍可能明确地说异步（分别为同步）运行，或异步（分别为同步）无死锁。对于两者，我们都会用到以下性质，并先对其进行证明。

性质（i）：对于每个异步运行

$$(\vec{q}_0,\xi_0)\xrightarrow{w_1}(\vec{q}_1,\xi_1)\xrightarrow{w_2}\ldots(\vec{q}_{n-1},\xi_{n-1})\xrightarrow{w_{n-1}}(\vec{q}_n,\xi_n)$$

满足 $w=w_1\ldots w_{n-1}$，存在 $u\sim w$和 $u=u_1\ldots u_{n-1}$，以及 $0<j<n$，使得 $u_1\ldots u_j\in(\mathrm{split}(\Gamma))^*$、$u_{j+1}\ldots u_{n-1}\in\hat{}\Gamma_!^*$，且后者中的发送方彼此不同：对于每个 i、k，若满足$j<i<n$且 $j<k<n$ 满足 $u_i=\mathtt{p}\ \triangleright\mathtt{q}!\,m_1$和 $u_k=\mathtt{r}\ \triangleright\mathtt{s}\,!\,m_2$，那么如果 $i\neq k$，则 $\mathtt{p}\ \neq\mathtt{r}$。

性质（i）的证明。我们通过对 n 归纳来证明这一断言。对于归纳基础，令 $n=0$。显然，该断言对空运行和空迹 ε成立。对于归纳步骤，我们假设该断言对 n（如上所述）成立，并证明它对 $n+1$成立。令

$$(\vec{q}_0,\xi_0)\xrightarrow{w_1}(\vec{q}_1,\xi_1)\ldots(\vec{q}_{n-1},\xi_{n-1})\xrightarrow{w_{n-1}}(\vec{q}_n,\xi_n)\xrightarrow{w_n}(\vec{q}_{n+1},\xi_{n+1})$$

即为要考虑的异步运行。我们根据 w_n的形式进行分类讨论，即判断它是发送事件还是接收事件。

假设 $w_n=\mathtt{p}\ \triangleright\mathtt{q}\,!\,m$。那么，我们无需重新排序 w，只需证明 p 不是 $u_j\ldots u_{n-1}$中的发送方。为导出矛盾，假设情况并非如此，即存在 $j<i<n$使得 $u_i=\mathtt{p}\ \triangleright\mathtt{r}\,!\,m'$。然而，此时 $\vec{q}_{n,\mathtt{p}}$ 将等待即时确认，这与 p 是 w_n中的发送方这一事实相矛盾。

假设 $w_n=\mathtt{p}\ \triangleleft\mathtt{q}?\,m$。我们利用归纳假设得到如下运行：

$$(\vec{q}_0,\xi_0)\xrightarrow{u_1}(\vec{q}'_1,\xi'_1)\ldots(\vec{q}_{n-1},\xi_{n-1})\xrightarrow{u_{n-1}}(\vec{q}_n,\xi_n)\xrightarrow{w_n}(\vec{q}_{n+1},\xi_{n+1})$$

根据 CSM语义的定义，存在 $j<k<n$使得 $u_k=\mathtt{q}\ \triangleright\mathtt{p}!\,m$。我们声称不存在 $j<i<n$使得 $u_i=\mathtt{p}\ \triangleright\mathtt{r}\,!\,m'$。

为得出矛盾，假设存在这样的 i。我们对 $i<k$或 $k<i$进行分类讨论。（它们不可能相同，这一点是平凡的。）首先，设 $k<i$，即 p 在 q 之后发送：那么 p 当前会等待 r 对 m' 的即时确认。即使 $\mathtt{r}\ =\mathtt{q}$，上述发送事件也不可能属于所请求的确认，因为它是在实际消息之前发送的，从而得出矛盾。其次，设 $i<k$，即 q 在 p 之后发送：尽管如此，p 仍在等待该确认（若非如此，则 $i\le j$，因为它已被接收）；若 w_n不是该确认，则我们得出矛盾，因为 w_n 不可能；若 w_n 是该确认，则 q 必须先前已接收 m'，然后 $j<i$，从而得出矛盾。

因此，我们知道 w_n是p在$u_{j+1}\ldots u_{n-1}\cdot w_n$中的唯一事件。根据归纳假设，我们还知道 u_k是q在$u_{j+1}\ldots u_{n-1}\cdot w_n$中的唯一（发送）事件。因此，不存在依赖关系，我们可以将 u_k和 w_n移到发送事件的前面，使用 ~：

$$u_{j+1}\ldots u_{n-1}\cdot w_n\ \sim\ u_k\cdot w_n\cdot u_{j+1}\ldots u_{k-1}\cdot u_{k+1}\ldots u_{n-1}$$

因此，

$$u_1\ldots u_{n-1}\cdot w_n\ \sim\ u_1\ldots u_j\cdot u_k\cdot w_n\cdot u_{j+1}\ldots u_{k-1}\cdot u_{k+1}\ldots u_{n-1}$$

而 $u_k\cdot w_n$ 基本上属于已得到确认的部分。根据归纳假设，其余发送事件满足关于不同发送方的要求。

性质（i）的证明结束。

我们通过两个包含来证明（a）。首先，我们证明 $L(\hat{}B) \subseteq L^{async}(G)$。设 $w \in L(\hat{}B)$。我们证明 $w \in L^{async}(G)$。根据定义，存在一个针对 w的异步运行：

$$(\vec{q}_0, \xi_0) \xrightarrow{w}{}^{*} (\vec{q}, \xi_\varepsilon) \ .$$

根据性质（i），存在 $u \in (\text{split}(\Gamma))^*$，使得 $u \sim w$满足$j = |w|$，因为没有剩余的空通道发送事件。根据 SCSM 的语义，可知存在 $u' \in \Gamma^*$，满足 $\text{split}(u') = u$ 且 $u \in L(B)$。根据假设，我们有 $u' \in L^{sync}(G)$。因此，根据定义，有 $u = \text{split}(u') \in L^{async}(G)$。结合 $w \sim u$，我们有 $w \in L^{async}(G)$，由此得证本情形。

其次，我们证明 $L^{async}(G) \subseteq L(\hat{}B)$。因此，设 $w \in L^{async}(G)$。我们证明 $w \in L(\hat{}B)$。根据定义，存在 $w' \in L(G)$，满足 $\text{split}(w') \sim w$。根据假设，我们有 $w' \in L(B)$，从而得到一个同步运行

$$\vec{q}_0 \xrightarrow{w'}{}^{*} \vec{q} \ .$$

不难看出，该运行可以被异步地模拟：在每两个配置之间，都需要添加一个除一个通道外其余通道均为空的配置。因此，$w' \in L(\hat{}B)$。根据 $\hat{}$ CSM 语义在 $\sim$下的闭包性，有 $w \in L(B)$，这就完成了该情形的证明。

我们通过逆否命题证明（b）。假设 $\hat{}B$ 存在异步死锁，并证明 B 也存在异步死锁。令

$$(\vec{q}_0, \xi_0) \xrightarrow{w_1} (\vec{q}_1, \xi_1') \dots (\vec{q}_{n-1}, \xi_{n-1}) \xrightarrow{w_{n-1}} (\vec{q}_n, \xi_n) \xrightarrow{w_n} (\vec{q}_{n+1}, \xi_{n+1})$$

为一个以死锁结束的异步运行。

根据性质（i），存在 $u \sim w$ 和 $u = u_1 \dots u_{n-1}$，以及 $0 < j < n$，使得 $u_1 \dots u_j \in (\text{split}(\Gamma))^*$、$u_{j+1} \dots u_{n-1} \in \hat{\Gamma}_!^*$，且后者中的发送方彼此不同：对于每个 i、k，若 $j < i < n$，以及对于 $j < k < n$、$u_i =$，若 $\mathsf{p} \triangleright \mathsf{q}!m_1$ 且 $u_k = \mathsf{r} \triangleright \mathsf{s}!m_2$，则 $i \neq k$ 时 $\mathsf{p} \neq \mathsf{r}$。

我们证明 $\vec{q}_j$ 是$\mathcal{B}$中的一个（可达）同步死锁。经过上述重新排序后，显然 $\vec{q}_j$ 是同步可达的。由于汇点终结性，$\vec{q}_j$ 不可能是最终配置，因为某些参与者可以发送消息，因此我们需要延长运行以避免死锁。一般而言，从 $\vec{q}_j$开始的同步运行可能永远不会遇到$u_{j+1} \dots u_{n-1}$中的发送事件，因为相应的接收方会阻塞交互。不过，条件（X）在此排除了这种情况。令 S为 $u_{j+1} \dots u_{n-1}$中的所有发送方。根据性质（i），所有发送方互不相同。我们将条件（X）应用于 $\vec{q}_j$。对于每个发送方$\mathsf{p} \in S$，若其发送事件为$\mathsf{p} \triangleright \mathsf{q}!m$，条件（X）的第一种情形绝不可能成立。若成立，$\mathsf{q}$便可以接收消息，这与存在异步死锁相矛盾。因此，对每个发送方，第二种情形必定成立；这意味着每个可能的接收方实际上都有义务在接收另一条消息之前先发送消息（包括确认）。因此，每个发送方都在等待其接收方，而接收方自身也必须先发送消息，然后等待对方的确认。这在同步设定下同样构成死锁，而这正是我们要证明的。□

引理 3.36。 如果其规范 *SCSM* $\mathcal{A}$ 同步实现了 $\mathcal{G}_{TM}$，那么 $\mathcal{A}$ 满足条件（X）。

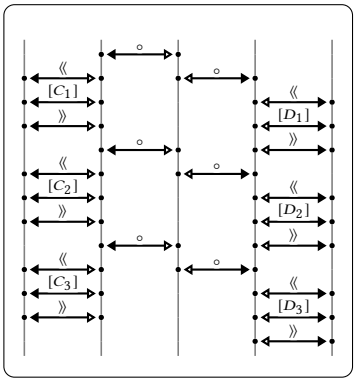


图5：MSC表示 $\text{msc}(w(C_1, D_1, C_2, D_2, C_3, D_3))$其中，双向箭头表示两个消息交互的简写，带实心箭头的一侧表示先发生的交互，而 $[C_1]$ 表示此类消息交互组成的序列。

证明。令 Δ 为带字母表，令 Q 为具有 $\Delta \cap Q = \emptyset$的图灵机的状态。为证明该断言，我们回顾从 [59, 构造 G_{TM}所需的记号和直观理解，详见1] 定理 9.，其中给出了完整构造。

该编码包含五个参与者 $\mathsf{p}_1, \dots, \mathsf{p}_5$，它们彼此发送配置。因此，消息来自集合 $\Delta \uplus \{\circ, \langle\langle, \rangle\rangle \ \bot\} \uplus Q$。

为简洁起见，使用以下记法：$\mathsf{p} \leftrightarrow \mathsf{q} : m$ 表示 $\mathsf{p} \to \mathsf{q} : m \cdot \mathsf{q} \to \mathsf{p} : m$。交互实际上只用 $_ \leftrightarrow _$ 指定：$_$，因此该编码具有即时确认特性。它们也用于定义正则表达式及其补集，其中 $\mathsf{p} \leftrightarrow \mathsf{q} : m$ 被视为独立的符号。对于一个字 $w = w_1 \dots w_i$，序列 $\mathsf{p} \leftrightarrow \mathsf{q} : w_1 \cdots \mathsf{p} \leftrightarrow \mathsf{q} : w_i$简写为 $\mathsf{p} \leftrightarrow \mathsf{q} : w$。

对于词 $C_1, \ D_1, \ C_2, \ D_2 \dots, \ C_m, \ D_m \in (\Delta \uplus Q)^*$，let

$$\begin{aligned} w(C_1, D_1, C_2, D_2, \dots, C_m, D_m) \coloneqq \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_2 : \circ \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \langle\langle \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : C_1 \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \rangle\rangle \cdot \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_4 : \circ \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \langle\langle \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : D_1 \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \rangle\rangle \cdot \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_2 : \circ \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \langle\langle \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : C_2 \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \rangle\rangle \cdot \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_4 : \circ \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \langle\langle \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : D_2 \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \rangle\rangle \cdot \\ & \dots \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_2 : \circ \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \langle\langle \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : C_m \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \rangle\rangle \cdot \\ & \mathsf{p}_3 \leftrightarrow \mathsf{p}_4 : \circ \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \langle\langle \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : D_m \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \rangle\rangle \ . \end{aligned}$$

图 5 将词 $w(C_1, D_1, C_2, D_2, \dots, C_m, D_m)$ 展示为消息序列图（MSC）。直观地说，p_2向 p_1发送序列 C_i，而 p_4向 p_5发送序列 D_i。每个序列都由相应参与者对之间的一条 -消息启动，并以一条 -消息结束。参与者 p_3 通过发送 $\circ$ 来启动每一轮次。

定义了两种语言 L_l和 L_r，协议的每个分支各对应一种：

$$\begin{aligned} L_l \coloneqq \{ & \mathcal{L}_{async}(w(C_1, D_1, \dots, C_m, D_m)) \mid \\ & m \geq 1, C_1, D_1, \dots, C_m, D_m \in (\Delta \uplus Q)^* \} \\ L_r \coloneqq L_l \setminus \{ & \mathcal{L}_{async}(w(u_1, u_1, \dots, u_m, u_m)) \mid \\ & (u_1, \dots, u_m) \text{ is an accepting computation } . \} \end{aligned}$$

接受计算的定义与预期一致，但对本证明并不重要，因此省略。以下消息序列确保该协议处于汇终态：

$$w_{end} \coloneqq \mathsf{p}_3 \leftrightarrow \mathsf{p}_2 : \bot \cdot \mathsf{p}_2 \leftrightarrow \mathsf{p}_1 : \bot \cdot \mathsf{p}_3 \leftrightarrow \mathsf{p}_4 : \bot \cdot \mathsf{p}_4 \leftrightarrow \mathsf{p}_5 : \bot$$

将其附加到 L_l和 L_r上，以得到 $L_l' \coloneqq \{w \cdot\ w_{end} \mid w \in L_l\}$ 和 $L_r' \coloneqq \{w \cdot\ w_{end} \mid w \in L_r\}$。

最终，构造全局协议 G_{TM}，使得

$$\mathcal{L}_{async}(\mathcal{G}_{TM}) = \{\mathsf{p}_2 \to \mathsf{p}_3 : \mathsf{l} \cdot w \mid w \in L_l'\} \uplus \{\mathsf{p}_2 \to \mathsf{p}_3 : \mathsf{r} \cdot w \mid w \in L_r'\}\ .$$

根据 L_l'和 L_r'的定义，每个字都以 w_{end}结尾，因此 G_{TM}是汇终态。显然，只有 p_2和 p_3会知道最初的分支选择。在 [59, 定理 9.1] 中，已证明 G_{TM}可异步实现，当且仅当 $L_l = L_r$。我们声称，同步可实现性蕴含 $L_l = L_r$。（尽管反方向也很可能成立，但本证明不需要用到。）为导出矛盾，假设 $L_l \neq L_r$。参与者 p_4始终不知道该选择，并向 p_5发送配置序列（期间会与 p_3通信）。但根据构造，$L_l{\Downarrow_{\Gamma_{\mathsf{p}_4}}} \neq L_r{\Downarrow_{\Gamma_{\mathsf{p}_4}}}$成立，因此 p_4将在其中一个分支中发送未指定的消息，从而导致死锁。事实上，这会发生在 L_l对应的分支中，因为根据构造，$L_l \supseteq L_r$。

对于我们的证明，两种语言相等这一点非常有帮助：根据假设，G_{TM}可同步实现，因此我们知道 $L_l = L_r$，而我们只需证明 G_{TM} 的规范 SCSM满足条件（X），其中

$$\mathcal{L}_{async}(\mathcal{G}_{TM}) = \{\mathsf{p}_2 \to \mathsf{p}_3 : \mathsf{l} \cdot w \mid w \in L_l'\} \uplus \{\mathsf{p}_2 \to \mathsf{p}_3 : \mathsf{r} \cdot w \mid w \in L_l'\}\ .$$

对于 L_l'，我们构造规范 SCSM 中参与者 p_1, p_2, 和 p_3对应的有限自动机。参与者 p_5 和 p_4 对应的自动机分别与参与者 p_1 和 p_2 对应的自动机类似。一旦证明这些有限自动机满足条件（X），就很容易看出，为 p_2 和 p_3 添加初始选择仍能保持条件（X）。图 6 展示了这些有限自动机。注意，我们将消息及其确认一并指定。因此，严格来说，每次消息交换都需要额外引入一个中间状态。

注意，这些有限自动机都会立即确认。因此，消息发送后，发送方会等待即时确认，从而满足条件（X）的第一项要求；所以我们将重点放在实际传输的消息上。

在规范 SCSM 的同步语义下，我们可以穷举检查条件（X）是否成立。不过，我们在此给出更抽象的论证。没有任何参与者会与 p_3，发起交互；也就是说，它只接收即时确认，因此显然满足条件（X）。可以看出，p_1（分别是 p_5）只会从 p_2（分别是 p_4）接收消息。因此，它们的行为“同步”：当且仅当 p_2 处于$q_{2,1}$ 或 $q_{2,2}$ 时，p_1 处于$q_{1,1}$；当且仅当 p_2 处于$q_{2,3}$ 时，p_1 处于 $q_{1,2}$。容易看出，p_1（分别是 p）总能立即接收 p_2（分别是 p_4）发送的消息，因此满足条件（X）的第一种情形。除即时确认外，只有 p_3 会向 p_2, 发送消息，即 $\circ$ 和 $\bot$。有可能 p_2 无法立即接收这条消息，因为它可能处于$q_{2,2}$ 或 $q_{2,3}$，但它满足条件（X）的第二种情形：它只向另一位会即时确认的参与者发送消息，并且可以从接下来的 p_3 接收所有可能的值，即处于$q_{2,1}$ 时。它无法在$q_{2,4}$ 接收这些值中的任何一个，但很容易看出，p_3不可能发送这些值：$\circ$ 不会在 $\bot$之后发送，而 $\bot$本身也只会发送一次。论证

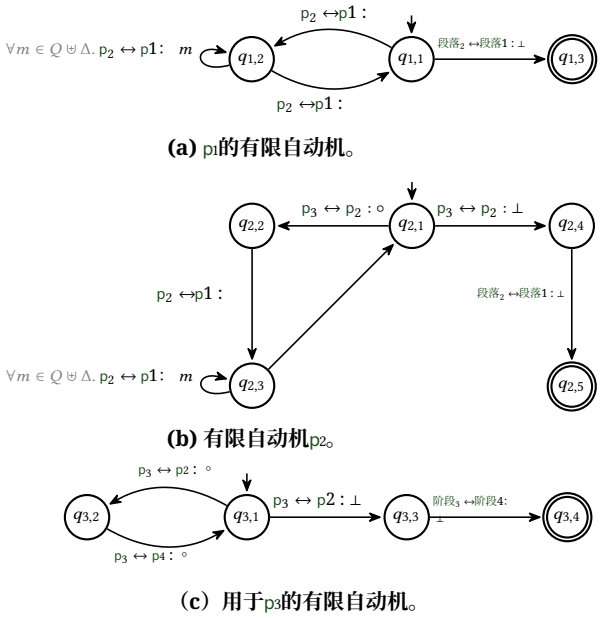


图6：$\mathcal{G}$的规范 SCSM（如果可实现）：用于p_1, p_2, 和p_3 的有限自动机，其中$\mathsf{p}_1 \to \mathsf{p}_2$：$m$ 是$\mathsf{p}_1 \to \mathsf{p}_2$：$m \cdot \mathsf{p}_2 \to \mathsf{p}_1$：$m$的缩写。

对于 p_4 的情况，论证类似。这表明规范 SCSM 满足条件（X），从而完成证明。□

定理 3.37。 汇终态全局协议的同步可实现性是不可判定的。

证明。如前所述，我们使用汇点终结（混合选择）协议 G_{TM}：当且仅当某个图灵机 [59, Thm.9.1], 不存在接受计算时，该协议才可异步实现，而这反过来是一个不可判定问题。因此，只需证明 G_{TM}可同步实现，当且仅当 G_{TM} 可异步实现即可。“如果”方向直接由 [18, Thm. 5.2]得出：尽管该工作使用“全局类型”一词，但它们实际上表示与我们的协议同样一般的协议。对于“仅当”方向，我们利用这样一个事实：由于 G_{TM} 可同步实现，规范 SCSM 确实是一个同步实现。引理 3.36 表明，规范 SCSM 确实满足条件（X）。这使我们能够应用引理 3.35，从而得到规范 SCSM 可以转换为异步实现。□

C §4 的补充材料

引理 C.1（类型判断中结构前同余的容许性）。设 P_1 和 P_2 均为进程。若 $\Theta \shortmid \Lambda \vdash P_1$ 且 $P_1 \sqsubseteq P_2$，则 $\Theta \shortmid \Lambda \shortmid \Omega \vdash P_2$。

证明。我们对 $\equiv$ 进行分类讨论，并分别证明两个方向。随后，我们考虑关于 $\sqsubseteq$ 的附加规则。

- $P_1 \parallel P_2 \equiv P_2 \parallel P_1$：通过反演，我们知道 PT-$\parallel$ 是类型推导中应用的第一条规则。该规则具有对称性，因此同一类型推导仍然适用。

- $(P_1 \parallel P_2) \parallel P_3 \equiv P_1 \parallel (P_2 \parallel P_3)$:
  通过反演，我们知道 PT-∥ 是类型推导中应用的第一条和第二条规则。很容易看出，可以重新排列类型推导，使其与该结构相匹配。

- $P \parallel \mathbf{0} \equiv P$:
  首先，假设存在一个类型推导

$$\frac{\Theta \mid \Lambda_1 \vdash P \qquad \Theta \mid \Lambda_2 \vdash \mathbf{0}}{\Theta \mid \Lambda_1, \Lambda_2 \vdash P \parallel \mathbf{0}}\ \text{PT-}\parallel$$

  我们证明存在一个类型推导 $\Theta \mid \Lambda_1,\ \Lambda_2 \vdash P$。反演表明，对于给定的类型推导 $\Theta \mid \Lambda_2 \vdash \mathbf{0}$，可以应用两条规则：PT-end 和 PT-0。因此可得$\Lambda_2 = \{c\colon \vec{q}_\mathsf{p}\}_{c \in S}$，对于一组会话端点 S，$\mathrm{end}(\vec{q}_\mathsf{p})$ 对每个 $c \in S$都成立。根据反演，存在一个关于 $\Theta \mid \Lambda_1 \vdash P$的类型推导。与我们的目标唯一不同之处在于类型上下文 Λ_2。按照另一个类型推导中的做法，可以使用 PT-end 处理这一点，从而证明此情形。
  其次，假设存在一个针对 $\Theta \mid \Lambda \vdash P$ 的类型推导。我们证明存在一个针对 $\Theta \mid \Lambda \vdash P \parallel \mathbf{0}$ 的类型推导。首先，我们应用 PT-∥，得到

$$\frac{\Theta \mid \Lambda \vdash P \qquad \dfrac{}{\Theta \mid \emptyset \vdash \mathbf{0}}\ \text{PT-}\mathbf{0}}{\Theta \mid \Lambda \vdash P \parallel \mathbf{0}}\ \text{PT-}\parallel$$

  其中右侧前提由 PT-0 满足，左侧前提则由假设给出。

- $(\nu s : \mathcal{A})\ (\nu s' : \mathcal{B})\ P \equiv (\nu s' : \mathcal{B})\ (\nu s : \mathcal{A})\ P$:
  根据反演，两个类型推导在开始时都需要两次应用 PT-ν。显然，这两次规则应用互不干扰，从而得到相同的待证前提：

$$\Theta \mid \Lambda, \Lambda_s, \Lambda_{s'} \vdash P$$

  因此，这由假设给出。
- $(\nu s\colon \mathrm{A})\ (P_1 \parallel P_2) \equiv P_1 \parallel (\nu s\colon \mathrm{A})\ P_2$，且 s 在 P_1 中不是自由的：首先，我们假设存在一个针对 $(\nu s\colon \mathrm{A})\ (P_1 \parallel P_2)$ 的类型推导，并证明存在一个针对 $P_1 \parallel (\nu s\colon \mathrm{A})\ P_2$ 的类型推导。连续两次应用反演可得

$$\text{PT-}\parallel\ \frac{\Theta \mid \Lambda_1 \vdash P_1 \qquad \Theta \mid \Lambda_2, \Lambda_s \vdash P_2}{\dfrac{\Theta \mid \Lambda_1, \Lambda_2, \Lambda_s \vdash P_1 \parallel P_2 \qquad (\vec{q}, \xi) \in \mathrm{reach}(\mathcal{A}) \qquad \Lambda_s = \{s[\mathsf{p}] : \vec{q}_\mathsf{p}\}_{\mathsf{p} \in \mathcal{P}_\mathcal{A}}}{\Theta \mid \Lambda_1, \Lambda_2 \vdash (\nu s : \mathcal{A})\ (P_1 \parallel P_2)}\ \text{PT-}\nu}$$

  在上面的推导中，我们在 Λ_s 的类型推导中使用了 $\Theta \mid \Lambda_2$、$\Lambda_s \vdash P_2$。下面解释为什么可以这样假设。根据定义，这些只包含与 s相关的类型绑定，而根据假设，P_1 中并不包含它。可能存在某个类型推导，其中 Λ_s 的部分内容出现在 P_1 的类型推导中，但这些部分只能通过规则 PT-end移除。因此，可以在 P_2的类型推导中模拟这类推导，从而证明我们对 Λ_s 和 Ω_s的处理是合理的。我们构造如下类型推导：

$$\frac{\dfrac{(\vec{q}, \xi) \in \mathrm{reach}(\mathcal{A}) \qquad \Lambda_s = \{s[\mathsf{p}] : \vec{q}_\mathsf{p}\}_{\mathsf{p} \in \mathcal{P}_\mathcal{A}} \qquad \Theta \mid \Lambda_2, \Lambda_s \vdash P_2}{\Theta \mid \Lambda_2 \vdash (\nu s : \mathcal{A})\ P_2}\ \text{PT-}\nu \qquad \Theta \mid \Lambda_1 \vdash P_1}{\Theta \mid \Lambda_1, \Lambda_2 \vdash P_1 \parallel (\nu s : \mathcal{A})\ P_2}\ \text{PT-}\parallel$$

  所有前提都与原类型推导中的前提一致，从而得出此情形。

  其次，我们假设存在一个针对 $P_1 \parallel (\nu s : \mathrm{A})\ P_2$ 的类型推导，并证明存在一个针对 $(\nu s : \mathrm{A})\ (P_1 \parallel P_2)$ 的类型推导。证明与前一种情形类似，但我们不必推理 Λ_s 和 Ω_s 的处理方式，只需证明存在一个类型推导，并且可以选择相应的处理方式。

- $(\nu s\colon \mathrm{A})\ \mathbf{0} \sqsubseteq \mathbf{0}$:
  假设存在 $\Theta \mid \Lambda \vdash (\nu s : \mathrm{A})\ \mathbf{0}$ 的类型推导。我们证明存在 $\Theta \mid \Lambda \vdash \mathbf{0}$ 的类型推导。通过反演可知，最后应用的规则是 PT-ν，并得到以下某个前提：$\Theta \mid \Lambda, \Lambda_s \vdash \mathbf{0}$，其中 $\Lambda_s = \{s[\mathsf{p}] : \mathrm{reach}(A_\mathsf{p})\}_{\mathsf{p} \in \mathcal{P}_\mathcal{A}}$。通过反演可知，在 $\Theta \mid \Lambda, \Lambda_s \vdash \mathbf{0}$ 的类型推导中，只有 PT-0 和 PT-end 可以应用。由于 PT-0 要求第二个类型上下文为空，因此它最后应用，其他所有推导都是应用 PT-end 的结果。因此，对于某个不包含 s 的会话集合 S，且对每个 $\mathsf{p} \in \mathsf{P}_{s'}$ 和 $s' \in \mathsf{S}$，$\mathrm{end}(\vec{q}_\mathsf{p})$，有 $\Lambda = \bigcup_{s'} \mathsf{S}\{s'[\mathsf{p}] : \vec{q}_\mathsf{p}\}_{\mathsf{p} \in \mathcal{P}_{s'}}$。因此，我们也可以先应用 PT-end $|\Lambda|$ 次，最后应用 PT-0，从而得到 $\Theta \mid \Lambda \vdash \mathbf{0}$ 的类型推导。

至此，$\sqsubseteq$的可容许性证明完成。 □

引理 C.2. 设 P 是一个进程。若 $\Theta \mid \Lambda \vdash P$和 x 不在 Λ中，则x 不能出现在 P中。

证明。反设 x 出现在 P中。那么，在类型推导的某个步骤中

$$\Theta \mid \Lambda \vdash P$$

以下两条规则之一适用于处理 x：PT-Q或PT-Σ。这两条规则都要求所有变量出现在各自的类型上下文中，故产生矛盾。□

引理 C.3（替换引理）。设 P是一个进程。对于所有 L，若有 $\Theta \mid \Lambda,\ x\colon\ L \vdash P$，则 $\Theta \mid \Lambda,\ v\colon\ L \vdash P[v/x]$.

证明。我们对类型推导的深度进行归纳，并根据推导中最后应用的规则进行分类讨论。
对于归纳基础，我们考虑深度为 0 的（唯一）规则：PT-0。在这种情形下，PT-0 的第二个类型上下文为空，这与我们的假设矛盾：$x\colon L$ 或 $v\colon L$ 会出现。
对于归纳步骤，归纳假设表明，该结论对深度较小的类型推导成立。

- PT-Q: We have that

$$\frac{\Theta(\mathrm{Q}) = L_1, \ldots, L_n}{\Theta \mid c_1 : L_1, \ldots, c_{i-1} : L_{i-1}, x : L_i, c_{i+1} : L_{i+1}, \ldots c_n : L_n \vdash \mathrm{Q}[\vec{c}]}\ \text{PT-Q}$$

  显然，我们只需为所需的类型推导证明完全相同的前提：

$$\Theta \mid c_1 : L_1, \ldots, c_{i-1} : L_{i-1},\ v : L_i, c_{i+1} : L_{i+1}, \ldots, c_n : L_n \vdash (\mathrm{Q}[\vec{c}])[v/x]\ .$$

- PT-end：我们分两种情形讨论。
  首先，我们有

$$\frac{\Theta \mid \Lambda \vdash P \qquad \mathrm{end}(q)}{\Theta \mid x : q, \Lambda \vdash P}\ \text{PT-end}$$

  我们证明

$$\frac{\Theta \vdash \Lambda \vdash P[v/x] \qquad \mathrm{end}(q)}{\Theta \vdash v : q, \Lambda \vdash P[v/x]}\ \text{PT-END}$$

对于第一个类型推导，我们有 $x : q,\ \Lambda$ 作为类型上下文。由于 Λ 是一个类型上下文（而不是句法类型上下文），x 不出现在 Λ 中。通过反演，我们有 $\Theta \vdash \Lambda \vdash P$。因此，$x$ 不可能出现在 P 中。如果确实出现，P 只能在包含 x 的类型上下文中进行类型判定；根据引理 C.2的逆否命题，而 x不出现在 Λ中。因此，$P = P[v/x]$，从而两个前提一致。

其次，我们有

$$\frac{\Theta \vdash \Lambda, x : L \vdash P \qquad \mathrm{end}(q)}{\Theta \vdash c : q, \Lambda, x : L \vdash P}\ \text{PT-END}$$

我们证明

$$\frac{\Theta \vdash \Lambda, v : L \vdash P[v/x] \qquad \mathrm{end}(q)}{\Theta \vdash c : q, \Lambda, v : L \vdash P[v/x]}\ \text{PT-END}$$

通过对第一个类型推导进行反演，我们知道两个前提都成立。两个推导的第二个前提相同。对于第一个前提，归纳假设适用。

- PT-Σ：我们按情形讨论 $x = c$、$x = c_k$ 是否满足某个 $k \in I$，或两者都不满足。
  - $x \neq c$和 $x \neq c_k$ 对任意 $k \in I$：
    在这种情形下，我们可以应用反演和归纳假设适用于第二个和第三个前提的所有情形。
  - $x = c$：
    我们有

$$\frac{\begin{array}{c}\delta(q) = \{(\mathsf{p} \to \mathsf{q}_i : l_i(L_i), q_i) \mid i \in I\} \uplus \{(\mathsf{q}_j \to \mathsf{p} : l_j(L_j), q_j) \mid j \in J\} \\ \forall i \in I.\, \Theta \vdash \Lambda, x : q_i, \{c_j : L_j\}_{j \in I \setminus \{i\}} \vdash P_i \\ \forall j \in J.\, \Theta \vdash \Lambda, x : q_j, \{c_i : L_i\}_{i \in I}, y_j : L_j \vdash P_j\end{array}}{\begin{array}{c}\Theta \vdash \Lambda, x : q, \{c_i : L_i\}_{i \in I} \\ \vdash \sum_{i \in I} x[\mathsf{q}_i] ! l_i\langle c_i\rangle . P_i + \sum_{j \in J} x[\mathsf{q}_j] ? l_j(y_j) . P_j\end{array}}\ \text{PT-}\Sigma$$

    我们需要证明

$$\frac{\begin{array}{c}\delta(q) = \{(\mathsf{p} \to \mathsf{q}_i : l_i(L_i), q_i) \mid i \in I\} \uplus \{(\mathsf{q}_j \to \mathsf{p} : l_j(L_j), q_j) \mid j \in J\} \\ \forall i \in I.\, \Theta \vdash \Lambda, v : q_i, \{c_j : L_j\}_{j \in I \setminus \{i\}} \vdash P_i[v/x] \\ \forall j \in J.\, \Theta \vdash \Lambda, v : q_j, \{c_i : L_i\}_{i \in I}, y_j : L_j \vdash P_j[v/x]\end{array}}{\begin{array}{c}\Theta \vdash \Lambda, v : q, \{c_i : L_i\}_{i \in I} \\ \vdash (\sum_{i \in I} x[\mathsf{q}_i] ! l_i\langle c_i\rangle . P_i + \sum_{j \in J} x[\mathsf{q}_j] ? l_j(y_j) . P_j)[v/x]\end{array}}\ \text{PT-}\Sigma$$

    第一个前提相同，而第二个和第三个前提由归纳假设得到。

  - $x = c_k$for $k \in I$:
    我们有

$$\frac{\begin{array}{c}\delta(q) = \{(\mathsf{p} \to \mathsf{q}_i : l_i(L_i), q_i) \mid i \in I\} \uplus \{(\mathsf{q}_j \to \mathsf{p} : l_j(L_j), q_j) \mid j \in J\} \\ \forall i \in I.\, \Theta \vdash \Lambda, c : q_i, x : L, \{c_{i'} : L_{i'}\}_{i' \in I \setminus \{i,k\}} \vdash P_i \\ \Theta \vdash \Lambda, c : q_k, \{c_i : L_i\}_{i \in I \setminus \{k\}} \vdash P_k \\ \forall j \in J.\, \Theta \vdash \Lambda, c : q_j, \{c_i : L_i\}_{i \in I}, y_j : L_j \vdash P_j\end{array}}{\begin{array}{c}\Theta \vdash \Lambda, c : q, x : L, \{c_i : L_i\}_{i \in I \setminus \{k\}} \\ \vdash \sum_{i \in I} c[\mathsf{q}_i] ! l_i\langle c_i\rangle . P_i + \sum_{j \in J} c[\mathsf{q}_j] ? l_j(y_j) . P_j\end{array}}\ \text{PT-}\Sigma$$

    We show that

$$\frac{\begin{array}{c}\delta(q) = \{(\mathsf{p} \to \mathsf{q}_i : l_i(L_i), q_i) \mid i \in I\} \uplus \{(\mathsf{q}_j \to \mathsf{p} : l_j(L_j), q_j) \mid j \in J\} \\ \forall i \in I \setminus \{k\}.\, \Theta \vdash \Lambda, c : q_i, v : L, \{c_{i'} : L_{i'}\}_{i' \in I \setminus \{i,k\}} \vdash P_i[v/k] \\ \Theta \vdash \Lambda, c : q_k, \{c_i : L_i\}_{i \in I \setminus \{k\}} \vdash P_k[v/x] \\ \forall j \in J.\, \Theta \vdash \Lambda, c : q_j, \{c_i : L_i\}_{i \in I}, y_j : L_j \vdash P_j[v/x]\end{array}}{\begin{array}{c}\Theta \vdash \Lambda, c : q, x : L, \{c_i : L_i\}_{i \in I \setminus \{k\}} \\ \vdash (\sum_{i \in I} c[\mathsf{q}_i] ! l_i\langle c_i\rangle . P_i + \sum_{j \in J} c[\mathsf{q}_j] ? l_j(y_j) . P_j)[v/x]\end{array}}\ \text{PT-}\Sigma$$

    第一个前提相同。第二个和第四个前提均可通过反演并应用归纳假设得到。对于第三个前提，我们断言 x 不会出现在 P_k 中。在第一个推导的结论中，我们有以下类型上下文：$\Lambda,\ c : q,\ x : L,\ \{c_i : L_i\}_{i \in I \setminus \{k\}}$。因此，根据类型上下文中的每个元素至多具有一个类型这一假设，我们知道 x 不会出现在 $\Lambda,\ c : q,\ \{c_i : L_i\}_{i \in I \setminus \{k\}}$ 中。由引理 C.2，x 因而不能出现在 P_k 中。因此，$P_k[v/x] = P_k$ 以及两个类型推导中的第三个前提相同，从而完成此情形。

- PT-‖:
  存在两个对称的情形。我们只考虑其中一个。因此，我们有

$$\frac{\Theta \vdash \Lambda_1, x : L \vdash P_1 \qquad \Theta \vdash \Lambda_2 \vdash P_2}{\Theta \vdash \Lambda_1, x : L, \Lambda_2 \vdash P_1 \parallel P_2}\ \text{PT-}\parallel$$

  我们证明存在一个关于以下内容的类型推导

$$\Theta \vdash \Lambda_1, v : L, \Lambda_2 \vdash (P_1 \parallel P_2)[v/x]$$

  根据我们的假设，每个元素在类型上下文中至多对应一种类型，Λ_1 和 Λ_2 不可能共享任何名称。通过反演，我们有 $\Theta \vdash \Lambda_2 \vdash P_2$。因此，根据引理 C.2，$x$ 不可能出现在 P_2 中。因此，$(P_1 \parallel P_2)[v/x] = P_1[v/x] \parallel P_2$。只需证明存在以下类型推导：

$$\frac{\Theta \vdash \Lambda_1, v : L, \vdash P_1 \qquad \Theta \vdash \Lambda_2 \vdash P_2}{\Theta \vdash \Lambda_1, v : L, \Lambda_2 \vdash P_1[v/x] \parallel P_2}\ \text{PT-}\parallel$$

  第二个前提与原始类型推导中的相同。第一个前提可以通过对原始类型推导进行反演并应用归纳假设得到。

- PT-ν:
  我们有一个关于以下对象的类型推导：

$$\Theta \vdash \Lambda, x : L, \Lambda_s \vdash (\nu s : \mathcal{A})\, P\ .$$

  此情形可通过反演轻松得到，并应用归纳假设，获得关于以下对象的类型推导：

$$\Theta \vdash \Lambda, v : L, \Lambda_s \vdash (\nu s : \mathcal{A})\, P[v/x]\ .$$

这就完成了替换引理的证明。 □

定理 4.8（主体约化）。设 P 是一个具有会话集合 $\mathcal{S}$ 的进程。如果
(1) $\vdash D : \Theta$, *(2)* $\Theta \vdash \Lambda \vdash P$ 与 $\Lambda = \Lambda, \{\Lambda_s\}_{s \in \mathcal{S}}$, *(3)* 对每个 $s \in S$，都有 $\vec{q} \in \mathit{reach}(A_s)$，使得$\Lambda_s = \{s[p] : \vec{q}_p\}_{p \in \mathcal{P}_\mathcal{A}}$, *and(4)* $P \longrightarrow P'$,

那么存在 Λ'，使得 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，满足 $\Theta \vdash \Lambda' \vdash P'$.

证明。我们对 $P \longrightarrow P'$ (4)的推导树深度进行归纳。

对于 $d = 1$，PR-exc 是唯一适用的规则。对 (4) 进行反演，我们得到：

$$\frac{I \cap J = \emptyset \qquad k \in I \qquad \pi_k = s[\mathsf{p}][\mathsf{r}]!l_k\langle c_k\rangle \qquad k' \in J \qquad \pi_{k'} = s[\mathsf{r}][\mathsf{p}]?l_{k'}(y_{k'}) \qquad l_k = l_{k'}}{\sum_{i\in I}\pi_i.P_i \parallel \sum_{j\in J}\pi_j.P_j \longrightarrow P_k \parallel P_{k'}[c_k/y_{k'}]}\ \text{PR-EXC}$$

因此，$P = Q_1 \parallel Q_2$ 满足 $Q_1 = \Sigma_{i\in I}\pi_i.P_i$ 和 $Q_2 = \Sigma_{j\in J}\pi_j.P_j$ 以及 $P' = P_{k'}[c_k/y_{k'}]$。对 (2) 进行反演：

$$\frac{(i)\ \Theta \vdash \Lambda_1 \vdash Q_1 \qquad (ii)\ \Theta \vdash \Lambda_2 \vdash Q_2}{\Theta \vdash \Lambda_1, \Lambda_2 \vdash Q_1 \parallel Q_2}\ \text{PT-}\parallel$$

对 (i) 和 (ii) 进行反演：

$$\frac{\begin{array}{c}\delta(\hat{q}_1) = \{(\mathsf{p}\to\mathsf{q}_i{:}l_i(L_i), q_i) \mid i \in I_1\} \uplus \{(\mathsf{q}_j\to\mathsf{p}{:}l_j(L_j), q_j) \mid j \in J_1\} \\ \forall i \in I_1.\ \Theta \vdash \hat{\Lambda}_1, s[\mathsf{p}] : q_i, \{c_j : L_j\}_{j\in I_1\setminus\{i\}} \vdash P_i \\ \forall j \in J_1.\ \Theta \vdash \hat{\Lambda}_1, s[\mathsf{p}] : q_j, \{c_i : L_i\}_{i\in I_1}, y_j : L_j \vdash P_j\end{array}}{\begin{array}{c}(i)\ \Theta \vdash \hat{\Lambda}_1, s[\mathsf{p}] : \hat{q}_1, \{c_i : L_i\}_{i\in I_1} \\ \vdash \sum_{i\in I_1} s[\mathsf{p}][\mathsf{q}_i]!l_i\langle c_i\rangle.P_i + \sum_{j\in J_1} s[\mathsf{p}][\mathsf{q}_j]?l_j(y_j).P_j\end{array}}\ \text{PT-}\Sigma$$

$$\frac{\begin{array}{c}\delta(\hat{q}_2) = \{(\mathsf{r}\to\mathsf{s}_i{:}l_i(L_i), q'_i) \mid i \in I_2\} \uplus \{(\mathsf{s}_j\to\mathsf{r}{:}l_j(L_j), q'_j) \mid j \in J_2\} \\ \forall i \in I_2.\ \Theta \vdash \hat{\Lambda}_2, s[\mathsf{r}] : q'_i, \{c_j : L_j\}_{j\in I_2\setminus\{i\}} \vdash P'_i \\ \forall j \in J_2.\ \Theta \vdash \hat{\Lambda}_2, s[\mathsf{r}] : q'_j, \{c_i : L_i\}_{i\in I_2}, y_j : L_j \vdash P'_j\end{array}}{\begin{array}{c}(ii)\ \Theta \vdash \hat{\Lambda}_2, s[\mathsf{r}] : \hat{q}_2, \{c_i : L_i\}_{i\in I_2} \\ \vdash \sum_{i\in I_2} s[\mathsf{r}][\mathsf{s}_i]!l_i\langle c_i\rangle.P'_i + \sum_{j\in J_2} s[\mathsf{r}][\mathsf{s}_j]?l_j(y_j).P'_j\end{array}}\ \text{PT-}\Sigma$$

其中 $I = I_1 \uplus J_1$ 和 $J = I_2 \uplus J_2$ 以及

$\Lambda_1 = \hat{}\Lambda_1, s[\mathsf{p}] : \hat{}q_1, \{c_i : L_i\}^{i\in I_1}$，和 $\Lambda_2 = \hat{}\Lambda_2, s[\mathsf{r}] : \hat{}q_2, \{c_i : L_i\}_{i\in I_2}$.

因此，我们有

$$\Lambda = \hat{\Lambda}_1, s[\mathsf{p}] : \hat{q}_1, \{c_i : L_i\}_{i\in I_1}, \hat{\Lambda}_2, s[\mathsf{r}] : \hat{q}_2, \{c_i : L_i\}_{i\in I_2}$$

并定义

$$\Lambda' = \hat{\Lambda}_1, s[\mathsf{p}] : q_k, \{c_i : L_i\}_{i\in I_1}, \hat{\Lambda}_2, s[\mathsf{r}] : q'_{k'}, \{c_i : L_i\}_{i\in I_2}\ .$$

我们需要证明以下两个断言。根据假设，我们知道标签 $l_k = l_{k'}$ 决定了类型：这里，两种情形下我们都使用了 $L_k = L_{k'}$。

- $\Theta, \Lambda' \vdash P_k \parallel P'_{k'}[c_k/y_{k'}]$:

$$\frac{(a)\ \Theta \vdash \hat{\Lambda}_1, s[\mathsf{p}] : q_k, \{c_i : L_i\}_{i\in I_1\setminus\{k\}}, \vdash P_k \qquad (b)\ \Theta \vdash \hat{\Lambda}_2, s[\mathsf{r}] : q_{k'}, \{c_i : L_i\}_{i\in I_2}, c_k : L_k \vdash P'_{k'}[c_k/y_{k'}]}{\Theta \vdash \Lambda' \vdash P_k \parallel P'_{k'}[c_k/y_{k'}]}\ \text{PT-}\parallel$$

  推导 (a) 直接由对 (i) 进行反演所得的第二个前提推出。推导 (b) 则由对 (ii) 进行反演所得的第三个前提、应用替换引理（引理 C.3），以及 $L_k = L_{k'}$ 这一事实推出。

- $\Lambda \longrightarrow \Lambda'$（或 $\Lambda = \Lambda'$）：

$$\frac{q'_1 \xrightarrow{\mathsf{p}\to\mathsf{q}_i:l_k(L_k)} q_k \qquad q'_2 \xrightarrow{\mathsf{s}_j\to\mathsf{r}:l_{k'}(L_{k'})} q_{k'}}{\Lambda \longrightarrow \Lambda'}\ \text{TR-}\Sigma$$

  对 (4) 进行反演可得 $\mathsf{q}_i = \mathsf{r}$，$\mathsf{p} = \mathsf{s}_j$ 并且 $k = k'$；结合 $L_k = L_{k'}$ 即可。

对于归纳步骤，我们假设对于所有推导树深度至多为 d 的 $P \longrightarrow P'$，P 和 P' 都满足该命题。我们根据深度为 $d+1$ 的推导树中最后应用的规则进行分类讨论。

- PR-Q:

我们有

$$\frac{\mathcal{D}(\mathrm{Q}, \vec{c}) \parallel P_2 \longrightarrow P'}{\mathrm{Q}[\vec{c}] \parallel P_2 \longrightarrow P'_2}\ \text{PR-Q}$$

此外，我们知道 (1) 至 (4) 对 $P = \mathrm{Q}[\vec{c}] \parallel P_2$ 以及 $P' = P'_2$ 都成立。我们需要证明存在 Λ'，满足 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \vdash \Lambda' \vdash P'_2$。我们希望对 $P = \mathcal{D}(\mathrm{Q}\ \vec{c}) \parallel P_2$ 以及 $P' = P'_2$ 应用归纳假设。我们需要证明所有条件 (1) 至 (4) 均成立。显然，(1) 和 (3) 不变，因此成立。根据假设，$\mathcal{D}(\mathrm{Q}\ \vec{c}) \parallel P_2 \longrightarrow P'_2$ 有一棵深度为 d 的推导树，因此也满足 (4)。对于 (2)，我们需要证明

$$\frac{(a)\ \Theta \vdash \vec{c} : \vec{L} \vdash \mathcal{D}(\mathrm{Q}, \vec{c}) \qquad (b)\ \Theta \vdash \Lambda_2 \vdash P_2}{\Theta \vdash \vec{c} : \vec{L}, \Lambda_2 \vdash \mathcal{D}(\mathrm{Q}, \vec{c}) \parallel P_2}\ \text{PT-}\parallel$$

我们对原始的 (2) 进行反演：

$$\text{PT-Q}\ \frac{\Theta(\mathrm{Q}) = \vec{L}}{\Theta \vdash \vec{c} : \vec{L} \vdash \mathrm{Q}[\vec{c}]} \qquad \Theta \vdash \Lambda_2 \vdash P_2}{\Theta \vdash \vec{c} : \vec{L}, \Lambda_2 \vdash \mathrm{Q}[\vec{c}] \parallel P_2}\ \text{PT-}\parallel$$

由此可立即得出 (b)。对于 (a)，我们依赖于进程定义的良类型性 (1)。根据假设，D 具有以下形式：D $P[\vec{c}\ \vec{x}]$ = D ; (Q $[\vec{x} = P]$); D，使得 D(Q $\vec{c}$) / 。因此，我们的目标 ( ) 是为$\Theta \vdash \vec{c}$ 提供类型推导：$\vec{L} \vdash P[\vec{c}/\vec{x}]$。对 (1) 应用反演 $|\mathrm{D}_1|$ 次后，我们得到$\Theta \vdash \vec{x} : \vec{L} \vdash P$。应用替换引理（引理 C.3）$|\vec{x}|$次后，我们得到 (a)。由此，我们证明了归纳假设适用，并得出其结论：存在 Λ'，满足 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \vdash \Lambda' \vdash P'_2$。这正是我们的目标，由此完成该情形。

- PR-ctx:

我们对不同类型的上下文进行分类讨论。

  - $\mathbb{C} \parallel Q$:

我们有

$$\frac{P_1 \longrightarrow P'_1}{P_1 \parallel Q \longrightarrow P'_1 \parallel Q}\ \text{PR-CTX}$$

此外，我们知道 (1) 到 (4) 对 $P = P_1 \parallel Q$ 和 $P' = P'_1 \parallel Q$ 均成立。我们需要证明存在 Λ' $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \vdash \Lambda' \vdash P'_1 \parallel Q$。我们想对 $P = P_1$ 和 $P' = P'_1$ 应用归纳假设。我们需要证明所有条件 (1) 到 (4) 均成立。显然，(1) 和 (3) 没有变化，因此仍然成立。根据假设，$P_1 \parallel Q \longrightarrow P'_1 \parallel Q$ 有一棵深度为 d 的推导树，因此也满足 (4)。对于 (2)，我们需要证明 $\Theta_1 \vdash \Lambda \vdash P_1$；这可直接由对原来的 (2) 进行反演得出：

$$\frac{\Theta \vdash \Lambda_1 \vdash P_1 \qquad \Theta \vdash \Lambda_2 \vdash Q}{\Theta \vdash \Lambda_1, \Lambda_2 \vdash P_1 \parallel Q}\ \text{PT-}\parallel$$

根据归纳假设的结论，存在 Λ'_1，满足 $\Lambda_1 \longrightarrow \Lambda'_1$ 或 $\Lambda_1 = \Lambda'_1$，使得 $\Theta \vdash \Lambda'_1 \vdash P'_1$。假设 $\Lambda_1 \neq \Lambda'_1$。由命题 4.7 可得 Λ_1, $\Lambda_2 \longrightarrow \Lambda'_1$, Λ_2。对于两种情形，即 $\Lambda_1 = \Lambda'_1$ 或非 $\Lambda_1 = \Lambda'_1$，同样有：

因此我们需要证明 $\Theta \vdash \Lambda_1', \ \Lambda_2 \vdash P_1' \parallel Q$。这很容易看出：

$$\frac{\Theta \vdash \Lambda_1' \vdash P_1' \qquad \Theta \vdash \Lambda_2 \vdash Q}{\Theta \vdash \Lambda_1', \Lambda_2 \vdash P_1' \parallel Q} \text{ PT-}\parallel$$

其中第一个前提由归纳假设的结论给出，第二个前提与上述反演相同。

- $Q \parallel \mathbb{C}$:
  该证明与前一情形类似。
- $(\nu s : \mathrm{A})\, \mathbb{C}$:
  我们有

$$\frac{Q \longrightarrow Q'}{(\nu s : \mathcal{A})\, Q \longrightarrow (\nu s : \mathcal{A})\, Q'} \text{ PR-CTX}$$

此外，我们知道，对于 $P = (\nu s : \mathrm{A})\, Q$ 和 $P' = (\nu s : \mathrm{A})\, Q'$，条件 (1) 至 (4) 均成立。我们需要证明，存在 Λ'，使得 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，并且满足 $\Theta \vdash \Lambda' \vdash (\nu s : \mathrm{A})\ Q'$。我们希望对 $P = Q$ 和 $P' = Q'$ 应用归纳假设。我们需要证明所有条件 (1) 至 (4) 均成立。显然，(1) 与之前相同，而根据假设，(4) 成立。接下来考虑 (2)。我们对原来的 (2) 进行反演，得到：

$$\frac{\begin{array}{c}\vec{q} \in \mathrm{reach}(\mathcal{A}) \\ \Lambda_s = \{s[\mathsf{p}] : \vec{q}_{\mathsf{p}}\}_{\mathsf{p} \in \mathcal{P}_{\mathcal{A}}} \qquad \Theta \vdash \Lambda, \Lambda_s \vdash Q\end{array}}{\Theta \vdash \Lambda \vdash (\nu s : \mathcal{A})\, Q} \text{ PT-}\nu$$

严格来说，我们需要先在命题中对类型上下文进行推广 Λ，才能进行以下推导。这一点很直接，且仅适用于这一种情形，因此在一般情形下省略。我们选取 Λ，Λ_s 作为类型上下文，并立即得到 (2)，其中副条件分别由 Λ 和 Λ_s 满足。对于 (3)，我们注意到，通过对原来的 (2) 进行反演，所有会话 $s' \in \mathcal{S} \setminus \{s\}$ 均满足该条件，而通过上述反演，s 也满足该条件。归纳假设的结论表明，存在 Λ' 和 Λ_s'，使得Λ，$\Lambda_s \longrightarrow \Lambda'$，$\Lambda_s'$或者 Λ，$\Lambda_s = \Lambda'$，Λ_s'，使得 $\Theta \vdash \Lambda'$，$\Lambda_s' \vdash Q'$。我们需要证明两个命题。
首先，我们要证明

$$\frac{\begin{array}{c}\vec{q} \in \mathrm{reach}(\mathcal{A}) \\ \Lambda_s = \{s[\mathsf{p}] : \vec{q}_{\mathsf{p}}\}_{\mathsf{p} \in \mathcal{P}_{\mathcal{A}}} \qquad \Theta \vdash \Lambda', \Lambda_s' \vdash Q'\end{array}}{\Theta \vdash \Lambda' \vdash (\nu s : \mathcal{A})\, Q'} \text{ PT-}\nu$$

虽然前两个前提与上文相同，但第三个前提可由归纳假设的结论推出。

其次，我们需要证明 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$。我们对 Λ、$\Lambda_s \longrightarrow \Lambda'$、$\Lambda_s'$或 Λ、$\Lambda_s = \Lambda'$、Λ_s'进行分类讨论。如果是后一种情形，那么 $\Lambda = \Lambda'$（因为在 Λ中不会出现会话 s 的类型绑定）。如果是前一种情形，我们再对 $\Lambda_s' = \Lambda_s$是否成立进行分类讨论。这基本上是在判断会话 s 是否与这一步有关：这一步不可能同时涉及两者，因为一步总是发生在单个会话之内。如果 $\Lambda_s' = \Lambda_s$，那么 $\Lambda \longrightarrow \Lambda'$。如果 $\Lambda_s' \neq \Lambda_s$，那么 $\Lambda = \Lambda'$。这就结束了这一情形。

- $[\,]$: 证明很简单，因为归纳假设立即适用。
- PR-$\sqsubseteq$: 我
  们有

$$\frac{P_1 \sqsubseteq P_1' \qquad P_1' \longrightarrow P_2' \qquad P_2' \sqsubseteq P_2}{P_1 \longrightarrow P_2} \text{ PR-}\sqsubseteq$$

此外，我们知道 (1) 至 (4) 对 $P = P_1$ 和 $P' = P_2$均成立。我们需要证明存在 Λ'，满足 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \vdash \Lambda \vdash P_2$。我们想对 $P = P_1'$ 和 $P' = P_2'$应用归纳假设。我们需要证明所有条件 (1) 至 (4) 均成立。显然，(1) 和 (3) 没有变化，因此仍然成立。(4) 根据假设成立。对于 (2)，我们注意到 $\Theta \vdash \Lambda \vdash P_1$ 根据假设成立。根据可容许性引理（引理 C.1），我们得到 $\Theta \vdash \Lambda \vdash P_1'$。由归纳假设的结论，我们得到 Λ'，满足 $\Lambda \longrightarrow \Lambda'$ 或 $\Lambda = \Lambda'$，使得 $\Theta \vdash \Lambda' \vdash P_2'$。应用可容许性引理（引理 C.1），我们得到 $\Theta \vdash \Lambda \vdash P_2$，由此结束这一情形。

至此，主体约化的证明结束。 □

关于类型安全。 通常，人们将类型安全作为主体约化的推论加以证明。也就是说，任何可类型化的进程都不会归约到错误。然而，与许多同步多方会话类型框架一样，我们并未在归约规则中定义任何错误 [8, 15, 20, 67]。Peters 和 Yoshida [52], 通过支持混合选择而与我们的工作最为接近，定义了两类错误：标签不匹配和值错误。后者检查条件表达式是否确实为布尔值。我们的进程不支持条件表达式。纳入这一特性并不困难，因为它与消息传递并发是正交的。标签不匹配此前是针对具有定向选择的进程加以考虑的，即一个进程在任意给定的分支处只能向一个指定的接收方发送，或从一个指定的发送方接收。在此设定下，标签不匹配发生于如下情形：p 想要向 q 发送，但不存在共同标签。在混合选择下，这种情况未必成问题：接收方可能正在等待某个发送方，而该发送方在此期间可能会向其他人发送。Peters 和 Yoshida [52] 将这一条件推广到混合选择（针对我们称为 s的单个会话），如下所示：P 具有标签错误，如果

$$P \equiv \sum_{i \in I} s[\mathsf{p}][\mathsf{q}_i]\, \hat{\pi}_i \,.\, P_i \parallel \sum_{j \in J} s[\mathsf{r}][\mathsf{s}_j]\, \hat{\pi}_j \,.\, P_j \parallel P'$$

使得存在 $i \in I$满足$\mathsf{q}_i = \mathsf{r}$ 且$\hat{}\pi_i = !\ \ l_i\langle m_i\rangle$，且对于所有$j \in J$满足$\hat{}\pi_j = ?\ \ l_j(x_j)$，都有 $l_j \neq l_i$。

直观而言，当p 想要向q发送消息，但q 无法接收该消息时，就会出现标签错误。我们并不认为所有此类情形都有问题。考虑下面这个简单协议，其中q 先与r 交互，然后再与p交互。考虑这个简单协议：$\mathsf{q} \to \mathsf{r} : l_1\langle m_1\rangle \cdot \mathsf{p} \to \mathsf{q} : l_2\langle m_2\rangle$。尽管根据 [52]: 出现标签错误，以下仍是该协议的一个无死锁实现

$$\begin{aligned} P_{\mathsf{p}} &= s[\mathsf{p}][\mathsf{q}]\,!\,l_2\langle m_2\rangle \,.\, \mathbf{0} \\ P_{\mathsf{q}} &= s[\mathsf{q}][\mathsf{r}]\,!\,l_1\langle m_1\rangle \,.\, s[\mathsf{q}][\mathsf{p}]\,?\,l_2(x_2) \,.\, \mathbf{0} \\ P_{\mathsf{r}} &= s[\mathsf{r}][\mathsf{q}]\,?\,l_1(x_1) \,.\, \mathbf{0} \end{aligned}$$

与此同时，根据他们和我们对约简的定义，以下约简序列也是可能的：

$$P_{\mathsf{p}} \parallel P_{\mathsf{q}} \parallel P_{\mathsf{r}} \longrightarrow P_{\mathsf{p}} \parallel s[\mathsf{q}][\mathsf{p}]\,?\,l_2(x_2) \,.\, \mathbf{0} \parallel \mathbf{0} \longrightarrow \mathbf{0} \parallel \mathbf{0} \parallel \mathbf{0} \equiv \mathbf{0}$$

因此，仅考虑两个参与者来预判错误是不可能的。真正可能出问题的情形是，一组依赖进程陷入循环，无法继续执行。如果这涉及所有参与者，我们会将其检测为死锁；但如果仍有部分参与者可以行动，我们就检测不到。换言之，进展保证任何时刻至少有一个参与者可以迈出一步，但某些参与者可能会在某个时刻之后无法再迈出一步。对于任何 SCSM，都可以检查后一种情形是否成立，而带类型进程的行为将与 SCSM 相符。

定理 4.9（进展）。设$\mathcal{A}$是一个汇终态且无死锁的 *SCSM*。设 P 为一个进程。我们假设
(1) $\vdash_{SF} \mathcal{D}\colon \Theta,_{-}(2)$
$\Theta \mid \Lambda \overset{q}{\vdash}_{SF} (\nu s\colon \mathcal{A})\, P$，以及
(3) $\vec{q} \to \vec{q}'$对某个 $\vec{q}'$成立。
那么，存在 P'，其满足 $P \longrightarrow P'$，使得 $\Theta \mid \Lambda \overset{\vec{q}'}{\vdash}_{SF} (\nu s\colon \mathcal{A})\, P'$。

证明。不失一般性，我们假设 (3) 的转移标签为 $\mathtt{t} \to \mathtt{r}\colon l(L)$。对 (2) 进行反演，得到

$$\frac{\begin{array}{c}\forall \mathtt{p} \in \mathcal{P}_{\mathcal{A}}.\, \forall c : q' \in \Lambda_{\mathtt{p}}.\ \mathrm{end}(q') \\ \vec{q} \in \mathrm{reach}(\mathcal{A}) \qquad \forall \mathtt{p} \in \mathcal{P}_{\mathcal{A}}.\, \Theta \mid \Lambda_{\mathtt{p}}, s[\mathtt{p}] : \vec{q}_{\mathtt{p}} \Vdash_{SF} Q_{\mathtt{p}}\end{array}}{\Theta \mid \{\Lambda_{\mathtt{p}}\}_{\mathtt{p} \in \mathcal{P}_{\mathcal{A}}} \vdash_{SF} (\nu s : \mathcal{A})\, (\prod_{\mathtt{p} \in \mathcal{P}_{\mathcal{A}}} Q_{\mathtt{p}})}\ \text{PT-v'}$$

我们实例化前提

$$\forall \mathtt{p} \in \mathcal{P}_{\mathcal{A}}.\, \Theta \mid \Lambda_{\mathtt{p}}, s[\mathtt{p}] : \vec{q}_{\mathtt{p}} \Vdash_{SF} Q_{\mathtt{p}}$$

与 $\mathtt{t}$ 和 $\mathtt{r}$，并观察如何获得它们的类型推导。我们知道 $\vec{q}_{\mathtt{t}}$ 和 $\vec{q}_{\mathtt{r}}$ 都有出边，因此它们都不是终态，所以 $\mathrm{end}(\vec{q}_{\mathtt{t}})$ 和 $\mathrm{end}(\vec{q}_{\mathtt{r}})$ 均不成立。因此，只有 PT-Σ 和 PT-Q 可以应用。我们进行分类讨论，并证明应用 PT-Q 最终也会导向应用 PT-Σ。首先，我们进行一次反演：

$$\frac{\Theta(\mathtt{Q}) = \vec{L}}{\Theta \mid \vec{c} : \vec{L} \Vdash_{SF} \mathtt{Q}[\vec{c}]}\ \text{PT-Q}$$

假设进程定义 $\mathtt{Q}[\vec{x}] = P'$ 中有 Θ。由 (1) 可知，$\Theta \mid \vec{x} : \vec{L} \Vdash_{SF} P'$。根据假设，进程定义是受保护的。因此，$P'$ 应当使用 PT-Σ 本身进行类型推导。从这里开始，两种情形的推理又非常相似。实际上有两点不同：第一，我们会一直携带 $\vec{x} : \vec{L}$；第二，我们会应用 PR-Q 来证明 $P \longrightarrow P'$。为简洁起见，我们省略通过进程定义进行的间接推导。

因此，我们将 PT-Σ 视为两者的类型规则

$$Q_{\mathtt{t}} = \sum_{i \in I_1} s[\mathtt{t}][\mathtt{q}_i]\,!\,l_i\langle c_i\rangle.\, P_i + \sum_{j \in J_1} s[\mathtt{t}][\mathtt{q}_j]\,?\,l_j(y_j).\, P_j \text{ and}$$
$$Q_{\mathtt{r}} = \sum_{i \in I_2} s[\mathtt{r}][\mathtt{s}_i]\,!\,l_i\langle c_i\rangle.\, P'_i + \sum_{j \in J_2} s[\mathtt{r}][\mathtt{s}_j]\,?\,l_j(y_j).\, P'_j\ .$$

我们对其类型推导进行反演，得到

$$\frac{\begin{array}{c}\delta(\vec{q}_{\mathtt{t}}) = \{(\mathtt{t} \to \mathtt{q}_i : l_i(L_i), q_i) \mid i \in I_1\} \uplus \{(\mathtt{q}_j \to \mathtt{t} : l_j(L_j), q_j) \mid j \in J_1\} \\ \forall i \in I_1.\, \Theta \mid \hat{\Lambda}_{\mathtt{t}}, s[\mathtt{t}] : q_i, \{c_j : L_j\}_{j \in I_1 \setminus \{i\}} \vdash P_i \\ \forall j \in J_1.\, \Theta \mid \hat{\Lambda}_{\mathtt{t}}, s[\mathtt{t}] : q_j, \{c_i : L_i\}_{i \in I_1}, y_j : L_j \vdash P_j \\ (i)\ \Theta \mid \hat{\Lambda}_{\mathtt{t}}, \{c_i : L_i\}_{i \in I_1}, s[\mathtt{t}] : \vec{q}_{\mathtt{t}} \\ \vdash \sum_{i \in I_1} s[\mathtt{t}][\mathtt{q}_i]\,!\,l_i\langle c_i\rangle.\, P_i + \sum_{j \in J_1} s[\mathtt{t}][\mathtt{q}_j]\,?\,l_j(y_j).\, P_j\end{array}}{}\ \text{PT-Σ}$$

$$\frac{\begin{array}{c}\delta(\vec{q}_{\mathtt{r}}) = \{(\mathtt{r} \to \mathtt{s}_i : l_i(L_i), q'_i) \mid i \in I_2\} \uplus \{(\mathtt{s}_j \to \mathtt{r} : l_j(L_j), q'_j) \mid j \in J_2\} \\ \forall i \in I_2.\, \Theta \mid \hat{\Lambda}_{\mathtt{r}}, s[\mathtt{r}] : q'_i, \{c_j : L_j\}_{j \in I_2 \setminus \{i\}} \vdash P'_i \\ \forall j \in J_2.\, \Theta \mid \hat{\Lambda}_{\mathtt{r}}, s[\mathtt{r}] : q'_j, \{c_i : L_i\}_{i \in I_2}, y_j : L_j \vdash P'_j \\ (ii)\ \Theta \mid \hat{\Lambda}_{\mathtt{r}}, \{c_i : L_i\}_{i \in I_2}, s[\mathtt{r}] : \vec{q}_{\mathtt{r}} \\ \vdash \sum_{i \in I_2} s[\mathtt{r}][\mathtt{s}_i]\,!\,l_i\langle c_i\rangle.\, P'_i + \sum_{j \in J_2} s[\mathtt{r}][\mathtt{s}_j]\,?\,l_j(y_j).\, P'_j\end{array}}{}\ \text{PT-Σ}$$

其中 $\Lambda_{\mathtt{t}} = \hat{}\Lambda_{\mathtt{t}}, \{c_i\colon L_i\}_{i \in I_1}$ 和 $\Lambda_{\mathtt{r}} = \hat{}\Lambda_{\mathtt{r}}, \{c_i\colon L_i\}_{i \in I_2}$。根据 (3) 以及我们的假设（转移标签为 $\mathtt{t} \to \mathtt{r} : l(L)$），存在 $k \in (I_1 \cap J_2) \cup (I_2 \cap J_1)$，使得 $l_k = l$。不失一般性，假设 $k \in I_1 \cap J_2$，即 $\mathtt{t}$ 发送而 $\mathtt{r}$ 接收。于是，

$$P' = \prod_{\mathtt{t} \in \mathcal{P}_{\mathcal{A}} \setminus \{\mathtt{t},\mathtt{r}\}} Q_{\mathtt{p}} \parallel P_k \parallel P'_k[c_k/y_k]\ .$$

我们证明存在一个类型推导

$$\frac{\begin{array}{c}(a)\ \forall \mathtt{p} \in \mathcal{P}_{\mathcal{A}}.\, \forall c : q' \in \Lambda_{\mathtt{p}}.\ \mathrm{end}(q') \qquad (b)\ \vec{q}' \in \mathrm{reach}(\mathcal{A}) \\ (c)\ \forall \mathtt{p} \in \mathcal{P}_{\mathcal{A} \setminus \{\mathtt{t},\mathtt{r}\}}.\, \Theta \mid \Lambda_{\mathtt{p}}, s[\mathtt{p}] : \vec{q}'_{\mathtt{p}} \Vdash_{SF} Q_{\mathtt{p}} \\ (d)\ \Theta \mid \hat{\Lambda}_{\mathtt{t}}, s[\mathtt{t}] : \vec{q}'_{\mathtt{t}}, \{c_j : L_j\}_{j \in I_1 \setminus \{k\}} \Vdash_{SF} P_k \\ (e)\ \Theta \mid \Lambda_{\mathtt{r}}, s[\mathtt{r}] : \vec{q}'_{\mathtt{r}}, c_k : L_k \Vdash_{SF} P'_k[c_k/y_k]\end{array}}{\Theta \mid \{\Lambda_{\mathtt{p}}\}_{\mathtt{p} \in \mathcal{P}_{\mathcal{A}}} \vdash_{SF} (\nu s : \mathcal{A}) \left(\prod_{\mathtt{p} \in \mathcal{P}_{\mathcal{A}} \setminus \{\mathtt{t},\mathtt{r}\}} Q_{\mathtt{p}} \parallel P_k \parallel P'_k[c_k/y_k] \right)}\ \text{PT-v'}$$

前提 (*a*) 和 (*b*) 与 (2) 上第一次反演的前提相同，而 (*c*) 可直接由同一反演的第三个前提推出。前提 (*d*) 可由对 (*i*) 进行反演的第二个前提得到，将其实例化为 k，因为 $\vec{q}'\mathtt{t} = q_k$。前提 (*e*) 也可类似得到：将对 (*ii*) 进行反演的第三个前提实例化为 k，因为$\vec{q}'\mathtt{r} = q'_k$，由此得到 $\Theta \mid \Lambda_{\mathtt{r}}, s[\mathtt{r}] : \vec{q}'\mathtt{r}\ y_k\colon L_k \Vdash_{SF} P'_k$，再对其应用替换引理（引理 C.3），即可得到目标。 □

引理 C.4（汇聚终态有限状态机的无死锁性）。设 $\mathcal{A}$ 是一个汇点终结且无死锁的 *SCSM*，P 是一个进程。我们假设

- $\vdash_{SF} \mathtt{D} : \Theta$,
- $\Theta \mid \Lambda \vdash_{SF} (\nu s\colon \mathcal{A})\, P$，并且
- $(\nu s\colon \mathcal{A})\, P \longrightarrow^* P_1$。那么，成立的是 $P_1 \sqsubseteq \mathbf{0}$ 或者存在 P_2 使得 $P_1 \longrightarrow P_2$。

证明概要。首先，我们声称，对于所有满足 $k \geq 0$ 且使得$_{-}(\nu s : \mathrm{A})\, P \longrightarrow^k P'$的 k，都存在某个 $\vec{q}$使得 $\Theta \mid \Lambda \overset{q}{\vdash}_{SF} P'$。这可以用与主体约化（定理 4.8）类似的方法证明。

若存在满足 $\vec{q} \longrightarrow \vec{q}'$的 $\vec{q}'$，我们就进行分类讨论。

若是如此，根据定理 4.9，我们知道存在满足 $P' \longrightarrow P''$的 P''，由此即可得出本情形。

假设不存在 $\vec{q}'$满足 $\vec{q} \longrightarrow \vec{q}'$。根据假设，$\mathcal{A}$是无死锁的，因此 $\vec{q}$中的所有状态都是终态。因此，对于每个 $\mathrm{end}(\vec{q}_{\mathtt{p}})$ 均成立，其中 $\mathtt{p} \in \mathcal{P}$。同时，根据汇点终结性，

无法从这些状态中的任何一个进行转移。我们对类型推导上进行反演 $\Theta \mid \Lambda \overset{q}{\vdash_{SF}} P'$，并在改写 Λ 和 P' 后得到如下结果：

$$\frac{\vec{q} \in \mathrm{reach}(\mathcal{A}) \qquad \begin{array}{c}\forall \mathsf{p} \in \mathcal{P}_{\mathcal{A}}.\, \forall c : q' \in \Lambda_{\mathsf{p}}.\ \mathrm{end}(q') \\ \forall \mathsf{p} \in \mathcal{P}_{\mathcal{A}}.\, \Theta \mid \Lambda_{\mathsf{p}}, s[\mathsf{p}] : \vec{q}_{\mathsf{p}} \Vdash_{SF} Q_{\mathsf{p}}\end{array}}{\Theta \mid \{\Lambda_{\mathsf{p}}\}_{\mathsf{p} \in \mathcal{P}_{\mathcal{A}}} \vdash_{SF} (\nu s : \mathcal{A})\,(\prod_{\mathsf{p} \in \mathcal{P}_{\mathcal{A}}} Q_{\mathsf{p}})}\ \text{PT-}\nu'$$

我们声称，对于每个 $\mathsf{p} \in \mathsf{P}$，都有 $Q_{\mathsf{p}} = \mathbf{0}$。这足以证明我们的目标，因为它蕴含 $P' \sqsubseteq \mathbf{0}$。设 $\mathsf{p} \in \mathsf{P}$。我们对 $\Theta \mid \Lambda_{\mathsf{p}}, s[\mathsf{p}]$ 做（多次）反演：$\vec{q}_{\mathsf{p}} \Vdash_{SF} Q_{\mathsf{p}}$，以了解可能出现哪些规则。进程 Q_{p} 是无限制的，因此不可能出现 PT-ν'。我们知道，对于类型上下文 Λ_{p} 中的每个状态 q，$s[\mathsf{p}]$：$\vec{q}_{\mathsf{p}}$ 都成立，并且 end(q) 成立。因此不可能出现 PT-Σ。PT-Q 也不可能出现，因为进程定义是受保护的，而这样的保护无法在这个类型上下文中进行类型检查，因为其中所有状态都满足 end(-)。因此，只可能出现 PT-$\|$、PT-end 和 PT-0，最终得到 0 的一个并行组合，其中可能包含嵌套。由此可得 $P' \sqsubseteq \mathbf{0}$。