arXiv:2301.10827v1 [cs.PL] 25 Jan 2023

MAGπ：易发生故障的通信类型

Matthew Alan Le Brun 和 Ornela Dardha

格拉斯哥大学
m.le-brun.1@research.gla.ac.uk
ornela.dardha@glasgow.ac.uk

摘要。 *Multiparty Session Types*（多方会话类型）是一种用于通信中心系统的类型纪律，可保证通信安全性、无死锁性和协议遵从性。近年来，出现了多项对故障进行建模并引入容错技术的研究。然而，这些研究通常会对底层网络作出假设， *e.g.*，例如采用 TCP 的通信可保证消息送达；或采用集中式可靠节点，并使用临时定义的可靠性概念；或 *only* 只处理 *e.g.*一种故障，即节点崩溃故障。在本研究中，我们开发了 MAGπ——一种多方、异步且广义的 π演算，它是 *first language and type system* 用于统一纳入以下内容：(*i*) 范围最广的非拜占庭故障，包括*message loss*、*delays* 和 *reordering*； *crash failures* 和 *link failures*；以及 *network partitioning*；(*ii*) 一种新颖且最一般化的 *reliability*概念，考虑协议中 *each* 参与者的视角；(*iii*) 从基于 UDP 的底层网络编程到基于 TCP 的应用层的各种网络假设。我们证明了主体归约和会话保真性；进程性质（无死锁性、终止性）；故障处理安全性和可靠性遵循性。

关键词：会话类型 · 分布式协议 · 故障 · 超时

1 引言

尽管人们在故障预防技术上投入了大量资源，复杂的分布式应用中仍时常发生故障。人们普遍认为，使用软件测试进行验证的传统方法，无法对分布式算法的正确性提供足够的信心。这主要是由于这些协议固有的 非确定性行为，使得手动测试所有边界情况并不可行。通过采用模型检测等穷举技术可以绕过这一问题 [9,30], ，这类技术能够遍历程序的整个状态空间，以验证其正确性。然而，为复杂的分布式算法构建合适的模型既艰巨又昂贵，而且往往难以实现（这是由于状态爆炸问题 [10]）。此外，即使成功将算法编码为合适的模型并完成检查，正确性保证也仅针对算法的设计，而非软件的实现；手写代码仍容易出现人为错误。相比之下，类型

和类型系统 [28] 是轻量级的验证形式。类型内置于编程语言中，能够直接为手写代码提供保证，并帮助开发者实现按构造即正确的软件。对于并发与分布式计算，会话类型[14,34,15,35,32,16] 自最初提出以来便迅速受到青睐 [14], ，其范围从二方——两个参与者，扩展到多方——多个参与者。

会话类型确保进程按照协议规范进行通信。因此，类型检查器可以静态判定通信的理想属性，例如类型安全（通信过程中不会发生错误）、协议遵从性（或会话保真性；进程的行为遵循预定义的协议）以及无死锁性（进程不会因等待而陷入停滞）。为此，会话类型已在多种编程语言中实现，包括 Java [18,11], Go [21], Haskell [17,26], Scala [31],Rust [19], Elixir [33]。

迄今为止，大多数会话类型理论都是为并发进程而非分布式进程设计的——也就是说，通常假设不会发生通信故障。少数（且数量迅速增加的）研究考虑了故障，但它们做出了诸多限制性假设，妨碍了这些方法在现实复杂分布式应用中的可行性。例如，异步理论 [23,16,32] 使用消息缓冲区，在“类似 TCP”的假设下对分布式通信建模：消息保证送达，且来自同一发送方的消息不会被重新排序。仿射会话 [24,12,6] 仅允许通过 try-catch 块处理应用层故障；对于可能源自硬件故障、网络不一致等情况的任意故障，则不提供支持。协调器模型方法 [1,8,36] 假设存在一定程度的可靠性，无论这种可靠性是由中心化弹性进程、可靠广播还是固定同步点提供的。

残酷的现实是，许多现实世界中的分布式协议（例如，共识算法）都无法假设这些条件中的任何一项。网络会在系统中引入许多故障点：节点可能崩溃，消息可能丢失、延迟或重复，节点之间的链路可能发生故障，等等。分布式协议的设计者已经认识到，故障不可避免，因此算法被设计为能够承受一定程度的故障，同时仍实现预期行为——这称为容错 [22]。如今广泛使用的容错协议示例包括 Paxos [20] 和 Raft [25] 共识算法；这些算法假设所有非拜占庭故障均有可能发生——即节点崩溃、链路故障、网络分区和消息不一致。

尽管这些算法的正确性已得到广泛研究，但由于前文所述的原因，其中许多算法在开发时，对最终可部署制品的正确性仍缺乏足够信心。为弥补这一差距，我们需要基于类型的验证，并将其提供给编程语言，从而支持设计者和开发者设计与实现正确的分布式算法。虽然（多方）会话类型在建模结构化通信和保证相关属性方面产生了重大影响，但其理论尚不足以表达这些复杂算法。

本文通过介绍 MAGπ——一种多方、异步且广义化的 π-演算——迈出了弥补这一差距的步伐。它是首个能够支持以下内容的语言和类型系统：(*i*) 最广泛的非拜占庭故障，包括消息丢失、延迟和重排序；崩溃故障和链路故障；以及网络分区——所有这些情况均通过超时处理；(*ii*) 一种新颖且最一般的可靠性概念，考虑协议中每个参与者的视角；以及 (*iii*) 一系列网络假设——从基于 UDP 的底层网络编程，到基于 TCP 的应用层编程。

示例 1（Ping Pong：类型）。我们通过互联网控制消息协议（ICMPπ ）中 ping 工具的一个简化版本来说明 MAG[1]，这是我们贯穿全文的示例。ping 工具总共包含三个角色彼此通信：两个角色**p** 和**r**彼此进行可靠的通信，并且两者都与第三个角色不可靠地通信**q**。我们的可靠性定义（§ 3.2考虑了每个角色的视角，因此允许角色拥有各自的（可能为空的）可靠性集合。根据上述假设，**p** 的可靠性集合为 $\{\mathbf{r}\}$，**r** 的可靠性集合为 $\{\mathbf{p}\}$，而**q** 的可靠性集合为 $\emptyset$。

下面给出会话类型，记作 $\mathbb{S}_{\mathbf{r}}$， $\mathbb{S}_{\mathbf{p}}$ 以及 $\mathbb{S}_{\mathbf{q}}$ ，分别对应角色**r**、**p** 和**q**。

$$\mathbb{S}_{\mathbf{r}} = \&\{\mathbf{p}?\mathsf{ok}().\mathbf{end},\ \mathbf{p}?\mathsf{ko}().\mathbf{end}\}$$

$$\mathbb{S}_{\mathbf{p}} = \mathbf{q}!\mathsf{ping}().\&\begin{cases}\mathbf{q}?\mathsf{pong}().\mathbf{r}!\mathsf{ok}().\mathbf{end},\\ \textcircled{1}.\,\mathbf{q}!\mathsf{ping}().\&\begin{cases}\mathbf{q}?\mathsf{pong}().\mathbf{r}!\mathsf{ok}().\mathbf{end},\\ \textcircled{1}.\,\mathbf{q}!\mathsf{ping}().\&\begin{cases}\mathbf{q}?\mathsf{pong}().\mathbf{r}!\mathsf{ok}().\mathbf{end},\\ \textcircled{1}.\,\mathbf{r}!\mathsf{ko}().\mathbf{end}\end{cases}\end{cases}\end{cases}$$

$$\mathbb{S}_{\mathbf{q}} = \&\begin{cases}\mathbf{p}?\mathsf{ping}().\mathbf{p}!\mathsf{pong}().\mathbf{end},\\ \textcircled{1}.\,\&\begin{cases}\mathbf{p}?\mathsf{ping}().\mathbf{p}!\mathsf{pong}().\mathbf{end},\\ \textcircled{1}.\,\&\begin{cases}\mathbf{p}?\mathsf{ping}().\mathbf{p}!\mathsf{pong}().\mathbf{end},\\ \textcircled{1}.\,\mathbf{end}\end{cases}\end{cases}\end{cases}$$

角色 r 是 *receiver* (&–称为 *branching*)，它等待两种选项：从 p 接收标签 ok 或 ko，然后终止协议（end）。角色 p 是 *sender*($\oplus$[2]–称为 *selection*)，它尝试获取有关 q 状态的信息。它首先向 q 发送一条 ping 消息（q ! ping()），然后等待从 q 接收消息。如果在上方分支中收到 pong（q ? pong()），则它得出结论：q 的状态为 *reachable*，并将此信息发送给 r（r ! ok()），之后终止。另一种情况是，p 进入一个 *timeout branch*($\textcircled{1}$)。为简单起见，我们假设 p 会尝试与 q 通信三次（如超时分支中的三级缩进所示），然后才认为 q 已 *unreachable*；之后，该会话也会通过向 r 发送 ko 并以 end 结束。在同一行中，角色 q 的协议由会话类型$\mathbb{S}_{\mathbf{q}}$给出，其中其超时分支与 $\mathbb{S}_{\mathbf{p}}$ 的超时相匹配。

[1] https://www.rfc-editor.org/rfc/rfc792[2] 为便于阅读，我们对发送至单一角色以及载荷类型为单位类型的情况采用简写，因此 $\oplus\{\mathbf{s}!\,\mathsf{m}(\text{单位类型}).\mathbb{S}\}$ 表示为$\mathbf{s}!\,\mathsf{m}().\mathbb{S}$。

1.1 贡献

现在，我们介绍我们在多方、异步和广义 π-演算（MAGπ）方面的贡献。

1. MAGπlan guage (§ 2):
   – MAGπ 是首个支持最广泛非拜占庭故障的语言，涵盖消息丢失、消息延迟和消息重排序；崩溃故障和链路故障；以及网络分区。– MAGπ 是首个在接收分支中引入超时（用于处理网络故障）的语言，同时还支持在无向分支这一广义设定下实现无向分支——即能够同时期待来自多个发送方的传入消息。

2. **MAGπ** 类型系统（§3）：– 是广义异步多方会话类型理论的保守扩展 [32], , 其优势包括：能够建模比传统语法理论（更多的协议（如全局类型）；并且能够灵活地在事后检查所需属性（例如无死锁性或终止性），而非在设计阶段进行检查。– 支持无向分支/选择，并且是首个引入超时分支的类型系统。– 支持一种新颖且最一般的可靠性定义（ § 3.2 ），该定义考虑了每个参与角色的视角，并基于可选的、依赖角色的可靠性假设。

3. 类型性质 (§ 4)：我们证明主体归约（定理 1）和会话保真性（定理 2）。我们证明故障处理安全性（推论 1）及其逆命题——可靠性遵循（推论 2），二者严格地关联了超时与可靠性。我们证明进程性质（定理 3），例如无死锁性、终止性和活性，与 [32]一致。最后，由于我们的 MAGπ 类型系统是图灵完备的，我们证明类型检查是可判定的（定理 4），并且有限消息缓冲区下的进程性质也是可判定的（定理 5）。

4. TCP 与 UDP (§ 5)：MAGπ 具有足够的表达能力，可以涵盖一系列网络假设：从基于用户数据报协议（UDP）运行的底层网络编程，到基于传输控制协议（TCP）运行的应用层软件。

5. 案例研究 (§ 6)：我们进一步展示如何使用超时和无向分支来建模一种带有缓存和内置负载均衡器的域名系统（DNS）分布式协议；我们还展示了该协议满足的性质，包括安全性和无死锁性。更多示例见附录 D，其中包括一个用于共识协议的领导者选举算法的原型规范。

2 MAGπ：语言

我们提出一种多方会话 π-微积分，基于 Scalas 和 Yoshida 的理论 [32], , 并将其扩展，以准确建模现实世界的分布式网络环境。我们假设处于最低抽象层级——进程唯一可用的故障检测机制是等待时间上限，即超时。

$c ::= \quad x \quad | \quad s[\mathrm{p}]$ (*variable, session w/ role*)$d ::= \quad v \quad | \quad c$ (*basic value, variable, session w/ role*)$w ::= \quad v \quad | \quad s[\mathrm{p}]$ (*basic value, session w/ role*)$P, Q ::= \quad \mathbf{0} \quad | (\nu s)$ P(*inaction, restriction*)$| \quad P \,|\, Q \quad | \quad P + Q$ (*composition, non-deterministic choice*) $| \quad c \oplus [\mathrm{q}]!\ \mathrm{m}\langle d\rangle.\, P$(*selection towards role* q)$| \quad c \,\&_{i \in I} \{[\mathrm{q}_i]?\ \mathrm{m}_i(x_i).P_i\}$ (*reliable branching from roles* q_i)$| \quad c \,\&_{i \in I} \{[\mathrm{q}_i]?\ \mathrm{m}_i(x_i).P_i, \rangle.\, Q\}$ (*branching from roles* q_i *w/*超时 Q)$\lrcorner|$ def D in $P \quad | \quad X\langle d\rangle$ (*process definition, process call*)$| \quad s$: σ (*session buffer*)$D ::= \quad X(\tilde{x}) = P$ (*process declaration*)$\sigma ::=$ $(\mathrm{p} \triangleright \mathrm{q} \triangleleft \mathrm{m}\langle w\rangle) \cdot \sigma \quad | \quad \epsilon$ (*session queue, empty session queue*)

图 1. MAGπ 的语法

我们的演算具有三项新颖特性：(i) 新的超时原语；(ii) 可期待来自不同发送方的消息；以及 (iii) 能够建模各种非拜占庭故障的操作语义。超时可以附加到接收动作（下文称为分支）上，用于描述在假定发生故障时要执行的替代进程（类似于错误处理程序）。

我们称故障是被假定发生的，而非被检测到的，因为我们建模了这样一种不可能性结果：无法区分消息延迟和消息丢失。因此，进程可能会过早超时，即使相应的消息并未丢失——现实世界正是如此！

我们这种方法的优势在于，故障检测机制不受故障类型影响，因此能够统一建模消息丢失、消息延迟、崩溃停止故障、链路故障和网络分区。

2.1 语法

定义 1（语言语法）。多方、异步和广义 π演算语法由图 *1*定义。

通信通过会话（s, s'）进行，参与其中的角色（p, q）取自集合 $\boldsymbol{\rho}$。该演算的基本构造包括带角色的会话 $s[\mathrm{p}]$,和基本值 v，二者均可通过变量（x, y）进行抽象。进程（P, Q）包括以下标准构造：（i）静止 0 表示进程终止；（ii）会话限制（νs） P 在 P 中绑定一个新会话 s；（iii）并行组合声明两个并发进程；（iv）选择 $c \oplus [\mathrm{q}]!\ \mathrm{m}\langle d\rangle.\, P$ 使用信道 c 向 q 发送一条标签为 m、载荷为

[R-⊕] $s[\mathrm{p}]\oplus[\mathrm{q}]\,!\,\mathrm{m}\langle w\rangle.\,P \mid s{:}\sigma \longrightarrow P \mid s{:}\sigma\cdot(\mathrm{p}\triangleright\mathrm{q}\triangleleft\mathrm{m}\langle w\rangle)$ · ¢[R- &] $s[\mathrm{q}]\,\&$ $_{i\in I}\{[\ _i]\quad{}_i\,x_i\,.P_i\,[,\quad.Q]\} \mid s\quad{}_k\triangleright\quad\triangleleft\quad{}_k\langle w\rangle\ \cdot\sigma\,\mathrm{p}\,?\,\mathrm{m}\,(\,)\,\rangle{:}(\mathrm{p}\ \mathrm{q}\ \mathrm{m}\,)$: 对 $\longrightarrow P_k[^{w}/_{x_k}] \mid s\ \sigma$ 于 $k\in I$[R- ›] $s[\mathrm{q}]\&_{i\in I}\{[\mathrm{p}_i]?\,\mathrm{m}_i(x_i).P_i, \rangle.\,Q\} \mid s{:}\sigma\longrightarrow Q \mid s{:}\sigma$[R- +] $P_1+P_2\longrightarrow P_i$ 对于 $i\in\{1,2\}$[R-$\boldsymbol{X}$] 定义 $X(x_1,\ldots,x_n)=P$在 $(X\langle w_1,\ldots,w_n\rangle \mid Q)\longrightarrow$定义 $X(x_1,\ldots,x_n)=P$在 $(P[^{w_1}/_{x_1}]\cdots[^{w_n}/_{x_n}] \mid Q)$[R- ℂ] $P\longrightarrow P'\implies \mathbb{C}[P]\longrightarrow\mathbb{C}[P'$[R-↓] $s{:}h\cdot\sigma\longrightarrow s{:}\sigma$

图 2. MAGπ 的约简规则

d——发送之后，进程按照 P 继续；(v) 定义和声明允许为进程指定名称，从而通过使用进程调用来实现递归。现在我们将进一步阐述语言中的新特性。

$\mathbf{c}\,\&_{i\in I}\{[\mathbf{q}_i]\,?\,\mathbf{m}_i(d).P\}$ 分支中的角色量化允许进程从一定范围内的其他角色接收消息。这在许多分布式协议中都有实际应用，例如负载均衡器。

$\mathrm{c}\,\&_{i\in I}\{[\mathbf{q}_i]\,?\,\mathrm{m}_i(d).P,$ 接收分支中使用超时作为故障检测机制。如果假定故障导致传入消息丢失或无法送达，则会启动进程 Q 。需要注意的是，超时具有非确定性——它们表示进程在认定故障发生之前等待的任意且未知的时长。

$P+Q$ 非确定性选择会随机地在两个可能的进程后续之间进行选择。我们使用这一结构来简化示例，以便更好地呈现。具体而言，它免去了对表达式和 if-then-else 结构的需求；这些内容属于常规机制，与我们的形式化彼此正交。

$s:\ \sigma$ 会话的消息缓冲区 s。缓冲区中的一个条目（$\mathbf{p}\triangleright\mathbf{q}\triangleleft\mathrm{m}\langle w\rangle$）表示一条从角色 **p** 发送至角色 **q** 、带有标签 m 和载荷 w的“传输中”消息。为了在我们的语言中适应异步性，这是必需的。

2.2 操作语义

我们首先给出归约上下文和缓冲区同余的定义。

定义 2（归约上下文）。归约上下文 $\mathbb{C}$ 从进程中抽象出外部环境，其形式如下：

$$\mathbb{C}\ ::=\ \mathbb{C}\,|\,P\ \mid\ (\nu s)\,\mathbb{C}\ \mid\ \mathsf{def}\ D\ \mathsf{in}\ \mathbb{C}\ \mid\ [\,]$$

因此，$\mathbb{C}[P]$ 指的是某个任意上下文 $\mathbb{C}$ 下的进程 P。

定义 3（缓冲区同余）。仅包含一个受限缓冲区的进程与静止同余。消息缓冲区可观察到完全重排。

$$(\nu s)\, s{:}\sigma \;\equiv\; \mathbf{0} \qquad\qquad s{:}\sigma_1 \cdot h_1 \cdot h_2 \cdot \sigma_2 \;\equiv\; s{:}\sigma_1 \cdot h_2 \cdot h_1 \cdot \sigma_2$$

定义 4（OS）。MAG 的操作语义π 通过一个归约关系 $\longrightarrow$ 归纳定义于图2，并结合标准的结构同余规则[32] 以及两条缓冲区同余规则（定义见定义3）。

现在我们来说明这些归约规则（图 2）。进程使用选择规则 [R-$\oplus$] 发送消息；该规则会将已发送的消息作为新条目加入会话缓冲区，并使进程推进至其续体。使用分支规则 [R- &] 从缓冲区读取已发送的消息。如果接收方有一个与发送方和消息标签匹配的有效分支，那么它就会推进至该分支对应的特定续体（此规则中的超时分支是可选的）。替换 $P_k[{}^{w}/_{x_k}]$ 表示在续体进程 P_k 中，用载荷值 w 替换变量 x_k 。超时规则 [R-$\textcircled{\scriptsize t}$] 会使进程推进至其超时分支，而不改变缓冲区。使用选择规则 [R-+] 解决非确定性选择，该规则会使进程推进至两个可能续体之一。调用规则 [R-$\boldsymbol{X}$] 会将进程调用替换为其定义的进程，并替换其中的各个参数。根据上下文规则 [R-$\mathbb{C}$]，进程可以在上下文中进行归约。最后，消息可以通过丢弃规则 [R-↓] 从缓冲区中丢失。

现在，我们来详细说明语义如何处理故障。图 2 中的约简规则允许对各种形式的故障进行建模，这得益于丢弃规则 [**R-↓**]的灵活性和简洁性。下文将进一步说明如何利用该规则对不同类型的故障进行建模：

– 消息丢失直接通过约简规则 [R-↓] 建模。– 崩溃故障通过对特定角色重复应用 [R-↓] 建模。例如，要模拟角色 p 崩溃，应将约简步骤 [R-↓] 应用于所有进入缓冲区且匹配模式 (p▷_◁_⟨_⟩) 的消息（ 表示“无关”值）。– 链路故障采用类似的方法建模；不同之处在于，丢弃的是两个特定接收方之间的消息。例如，要模拟角色 p 和 q 之间的链路故障，需要将 [R-↓] 应用于所有进入缓冲区且匹配模式 (p ▷ q ◁ _⟨_⟩) 和 (q ▷ p ◁ _⟨_⟩) 的消息。

– 消息延迟 通过将规则 [**R-$\textcircled{\scriptsize t}$**] 应用于缓冲区中有有效消息的分支来建模。 例如：

$$\begin{aligned} & s[\mathbf{q}]\,\&_{i\in I}\{[\mathbf{p}_i]\,?\,\mathsf{m}_i(x_i).P_i,\ \textcircled{\scriptsize t}.\,Q\} \;\mid\; s{:}(\mathbf{p}_k \triangleright \mathbf{q} \triangleleft \mathsf{m}_k\langle w\rangle)\cdot\sigma \\ & \quad\longrightarrow\; Q \;\mid\; s{:}(\mathbf{p}_k \triangleright \mathbf{q} \triangleleft \mathsf{m}_k\langle w\rangle)\cdot\sigma \qquad\qquad \text{for } k\in I. \end{aligned}$$

– 总的来说，消息重排序通过缓冲区同余规则（定义 3）。– 网络分区可以通过多个链路故障来表示。

我们对故障进行建模的粒度允许不同程度的定制。例如，良性的容错共识算法通常假设所有非拜占庭故障都可能发生，因此需要考虑上述所有故障。或者，假设某个应用运行在可信 TCP 网络上，则无需担心单条消息丢失，因此 [R-↓] 只应用于对节点崩溃和链路故障进行建模。

定义 5（良构性）。为确保通信能够进行，我们要求良构进程为每个会话都配有一个缓冲区，即

$$P = (\nu s)\, Q \implies Q \equiv (\nu \tilde{s'})\, (Q' \mid s{:}\sigma)$$

定义 5 引入了良构性条件，以保证会话始终保护其缓冲区，从而确保消息始终有队列可供放入。从现在起，我们只考虑良构进程。

在结束本节之前，我们回顾引言中的 Ping Pong 运行示例，并在下文给出角色 p 、 q 和 r 的进程。

示例 2（Ping Pong：进程）。

$$P_{\mathsf{p}} = s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{ping}\langle\rangle.\, s[\mathsf{p}]\,\&\begin{cases} [\mathsf{q}]\,?\,\mathsf{pong}().P_{\mathsf{p}}^{ok}, \\ \underset{1}{}.\, s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{ping}\langle\rangle.\, s[\mathsf{p}]\,\&\begin{cases} [\mathsf{q}]\,?\,\mathsf{pong}().P_{\mathsf{p}}^{ok}, \\ \underset{1}{}.\, s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{ping}\langle\rangle.\, P'_{\mathsf{p}} \end{cases} \end{cases}$$

$$P_{\mathsf{p}}^{ok} = s[\mathsf{p}] \oplus [\mathsf{r}]\,!\,\mathsf{ok}\langle\rangle.\, \mathbf{0} \qquad P'_{\mathsf{p}} = s[\mathsf{p}]\,\&\begin{cases} [\mathsf{q}]\,?\,\mathsf{pong}().P_{\mathsf{p}}^{ok}, \\ \underset{1}{}.\, s[\mathsf{p}] \oplus [\mathsf{r}]\,!\,\mathsf{ko}\langle\rangle.\, \mathbf{0} \end{cases}$$

$$P_{\mathsf{q}} = s[\mathsf{q}]\,\&\begin{cases} [\mathsf{p}]\,?\,\mathsf{ping}().P_{\mathsf{q}}^{pong}, \\ \underset{1}{}.\, s[\mathsf{q}]\,\&\begin{cases} [\mathsf{p}]\,?\,\mathsf{ping}().P_{\mathsf{q}}^{pong}, \\ \underset{1}{}.\, s[\mathsf{q}]\,\&\{[\mathsf{p}]\,?\,\mathsf{ping}().P_{\mathsf{q}}^{pong},\ \underset{1}{}.\, \mathbf{0}\} \end{cases} \end{cases}$$

$$P_{\mathsf{q}}^{pong} = s[\mathsf{q}] \oplus [\mathsf{p}]\,!\,\mathsf{pong}\langle\rangle.\, \mathbf{0}$$

$$P_{\mathsf{r}} = s[\mathsf{r}]\,\&\{[\mathsf{p}]\,?\,\mathsf{ok}().\mathbf{0}, [\mathsf{p}]\,?\,\mathsf{ko}().\mathbf{0}\}$$

3 MAGπ：类型系统

我们介绍 MAG 的类型系统π，它是对广义异步多方会话类型理论[32, 第 7]节的保守扩展。广义多方会话类型理论不再采用全局协议规范（全局类型），而是基于用户定义的、针对每个参与角色的局部规范（局部类型）进行运作。采用这类理论的好处包括：（i）与传统上用于强制实现一致通信的句法方法（例如全局类型）相比，能够涵盖更多可行协议；（ii）能够对具有不同要求的协议建模。特别是，广义理论不会通过句法强制程序员编写例如无死锁代码，而是允许程序员不受限制地设计协议，再事后针对任意数量的所需属性进行检查，例如无死锁性、终止性等。

基本类型与会话类型　缓冲区类型$\mathbb{T}$ ::
$= \mathbb{B} \mid \mathbb{S}$　　$\mathbb{M} ::= \mathrm{p}\ !\mathrm{m}(\mathbb{T})\cdot\mathbb{M} \mid \epsilon\mathbb{B} ::=$
int | bool | real | 单位类型 | $\cdots\mathbb{S} ::= \quad {}_{i\in I}\{\ {}_{i} \quad {}_{i}\ \mathbb{T}_i\ .\mathbb{S}_i[,\quad .\mathbb{S}']\}\ \&\ \mathrm{p}\ ?$
$\mid\ \oplus_{i\in I}\{\ {}_{i} \quad {}_{i}\ \mathbb{T}_i\ .\mathbb{S}_i\}$
m ( ) › p ! m ( ) 会话 - 缓冲区类型 $\tau ::= \mathbb{M} \mid \mathbb{S} \mid (\mathbb{M};\ \mathbb{S}) \mid\ \mu t.\mathbb{S} \mid t\ \mid$ 结束

图 3. 基本类型、会话类型、缓冲区类型和会话-缓冲区类型

$$\frac{}{\mathbb{T} \equiv \mathbb{T}} \qquad \frac{}{\mathbb{M}_1 \cdot \mathbb{M}_2 \equiv \mathbb{M}_2 \cdot \mathbb{M}_1} \qquad \frac{}{\epsilon \cdot \epsilon \equiv \epsilon} \qquad \frac{\mathbb{M} \equiv \mathbb{M}' \qquad \mathbb{S} \equiv \mathbb{S}'}{(\mathbb{M};\ \mathbb{S}) \equiv (\mathbb{M}';\ \mathbb{S}')}$$

图 4. 类型同余规则

我们的类型系统的新颖之处包括：(i) 无向分支/选择；(ii) 超时分支(语法见 § 3.1)；以及(iii) 可靠性集合——即从每个角色（§ 3.2）的视角来看，假定不会发生故障的角色集合。可靠性集合（可以为空）强制所有易发生故障的通信都使用超时。

与 [32], 我们的类型系统不同，我们的类型系统不使用全局类型，而仅依赖局部类型。因此，类型上下文必须满足安全性，以确保主体归约（§ 3.3）。最后，我们在§ 3.4中给出类型系统的规则，并在§ 4中讨论其关键属性。

3.1 类型

我们的多方会话类型理论针对分布式计算场景而设计。具体而言，我们的类型系统（定义 6）是异步的；它允许从多个角色进行分支（分别是选择至多个角色）；并支持超时续体类型。

定义 6（类型语法）。类型语法由图 3中的文法定义。对于无向分支和选择，$I \neq \emptyset$ 以及角色-标签元组$(\mathbf{p}_i, \mathrm{m}_i)$ 必须两两不同。递归变量不能自由出现，且必须在分支/选择类型下受保护地出现。

类型 $\mathbb{T}$ 表示基本类型 $\mathbb{B}$或会话类型 $\mathbb{S}$，用于为变量指定类型。会话类型描述信道的使用方式：(i) 无向分支（外部选择）$\&_{i\in I}\{\mathrm{p}_i\ ?\ \mathrm{m}_i(\mathbb{T}_i).\mathbb{S}_i[,\rangle.\mathbb{S}']\}$ 表示接收一条标签为 m_i、载荷类型为 $\mathbb{T}_i$ 的消息，该消息来自角色 p_i，然后按照 $\mathbb{S}_i$继续。超时续体类型 $\mathbb{S}'$ （可选）描述在该分支发生故障时的处理协议；(ii) 无向选择（内部选择）$\oplus_{i\in I}\{\mathrm{p}_i\ !\ \mathrm{m}_i(\mathbb{T}_i).\mathbb{S}_i\}$ 表示向角色 p_i发送一条标签为 m_i、载荷为 $\mathbb{T}_i$的消息，然后按照 $\mathbb{S}_i$继续；(iii) 类型 end 表示信道已关闭，并终止通信。会话缓冲区使用缓冲区

类型 $\mathbb{M}$。缓冲区中的条目必须与类型 $\mathsf{p}\,!\mathsf{m}(\mathbb{T})\cdot\mathbb{M}$对应，表示发送给 p 的消息，其标签为 m，载荷类型为 $\mathbb{T}$。具有角色的会话使用 会话缓冲区 类型指定，将会话类型与缓冲区类型结合起来。

类型同余 $\equiv$ 定义见图4。值得注意的是，缓冲区类型可以重排序；如果两个会话-缓冲区类型各自的缓冲区类型和会话类型同余，那么它们就同余。缓冲区类型重排序是必要的，以匹配语言所支持的消息完全重排序（定义3）。

3.2 可靠性

我们暂且岔开话题，谈谈可靠性。此前的相关研究 [4,1,37]已将可靠性概念纳入其类型系统。通常，研究会假设某个特定角色或预先定义的一组角色是可靠的——也就是说，涉及这组已确定角色的通信不会发生故障。

我们的可靠性定义（定义7）最为通用，也是首个将每个角色的视角纳入考量的定义。我们认为，在分布式环境中这很有必要，因为网络中的可靠性取决于进程的物理拓扑。回想一下 ping 工具（示例1），我们可以设想代表角色p和r 的进程位于同一物理硬件上，因此它们之间的通信不会受到网络故障的影响；而代表q 的进程位于地理位置分隔的硬件上，因此它与p 和r 之间的通信容易发生故障。

定义 7（可靠性）。角色 $\mathfrak{R}$ 的可靠性集合$\mathsf{p} \in \boldsymbol{\rho}$ 定义为 $\mathfrak{R} \subseteq \boldsymbol{\rho} \setminus \{\mathsf{p}\}$，用于描述$\mathsf{p}$的视角。可靠性 $\mathcal{R}$ 定义为一个将角色映射到其可靠性集合的函数，即 $\mathcal{R}\colon\ \boldsymbol{\rho} \to \mathfrak{R}$。

为了更好地对真实的分布式环境建模，我们对可靠性的定义允许每个角色拥有自己的（可能为空的）可靠性集合。

示例 3（Ping Pong：可靠性集合）。关于示例 1，由于三个角色对彼此的看法各不相同，因此每个角色的可靠性集合也各不相同。具体而言，我们有 $\mathcal{R}(\mathsf{p}) = \{\mathsf{r}\}$，$\mathcal{R}(\mathsf{r}) = \{\mathsf{p}\}$，$\mathcal{R}(\mathsf{q}) = \emptyset$。

考察两种极端情况：对于角色集合 $\boldsymbol{\rho}$，如果对所有 $\mathsf{p} \in \boldsymbol{\rho}\cdot\mathcal{R}(\mathsf{p}) = \emptyset$，那么没有任何通信是可靠的；反之，如果对所有 $\mathsf{p} \in \boldsymbol{\rho} \cdot \mathcal{R}(\mathsf{p}) = \boldsymbol{\rho} \setminus \{\mathsf{p}\}$，那么所有通信都是可靠的——这称为可靠网络。本文仅考虑 $\mathcal{R}$的静态配置，因此可靠性集合不会在运行时发生变化。我们发现，即使有这一限制，与相关研究相比，我们的定义仍然是最一般的。

3.3 上下文

定义 8（类型上下文）。上下文 Θ 是从进程变量到 n元组（由类型和上下文构成）的部分映射，Γ 是从变量到类型，以及从带角色的会话到会话-缓冲区类型的部分映射，二者定义如下：

$$\Theta ::= \emptyset \mid \Theta, X : \mathbb{T}_1, \ldots, \mathbb{T}_n \qquad \Gamma ::= \emptyset \mid \Gamma, x : \mathbb{T} \mid \Gamma, s[\mathsf{p}] : \tau$$

上下文的组合$(\Gamma_1,\ \Gamma_2)$ 当且仅当以下条件成立时定义：

$$\forall c \in \mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma_2):\quad \Gamma_i(c) = \mathbb{M}\ \wedge\ \Gamma_j(c) = \mathbb{S}$$

对于这样的 c，$(\Gamma_1,\Gamma_2)(c) = (\mathbb{M};\ \mathbb{S})$。
上下文是同余的 $\Gamma_1 \equiv \Gamma_2$ 当且仅当：

$$\mathsf{dom}(\Gamma_1) = \mathsf{dom}(\Gamma_2)\ \wedge\ \forall c \in \mathsf{dom}(\Gamma_1) : \Gamma_1(c) \equiv \Gamma_2(c)$$

上下文 Θ 是非线性的，并通过跟踪进程变量参数的类型来为其定型。上下文 Γ 是线性的，允许变量具有基本类型或会话类型，并允许带角色的会话具有会话-缓冲区类型；随着程序推进，一个角色可以同时拥有一个活动会话类型，以及排在消息缓冲区中的消息。

只要两个上下文中的公共信道在一个上下文中映射到缓冲区类型，而在另一个上下文中映射到会话类型，上下文组合就允许这两个上下文共存。

上下文同余成立的条件是，两个上下文具有相同的定义域，且其信道的类型同余。需要注意的是，根据上下文组合和同余的定义，我们有 $s[\mathbf{p}] : (\mathbb{M};\ \mathbb{S}) \equiv s[\mathbf{p}] :\ \mathbb{M},\ \ s[\mathbf{p}] :\ \mathbb{S}$。缓冲区类型（分别为会话-缓冲区类型）仅在类型系统内部使用；不要求最终用户显式定义这些类型。

定义 9（上下文归约）。动作 α 定义如下：

$$\alpha\ ::=\ s[\mathbf{p}]\,!\,\mathbf{q} : \mathsf{m}(\mathbb{T})\ \mid\ s[\mathbf{p}][\mathbf{q}] : \mathsf{m}\ \mid\ s[\mathbf{p}]_{(\mathfrak{t})}$$

从左到右，依次表示：(i) 一个已发送的消息；(ii) 一条消息的通信；以及(iii) 一个信道的超时。上下文迁移 $\xrightarrow{\alpha}_{(\Sigma;\mathcal{R})}$ 定义见图5。我们将 $\Gamma \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})}$ 记作$\exists\, \Gamma' : \Gamma \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma'$。我们定义两个上下文归约$\rightarrow_{(\Sigma;\mathcal{R})}$ 和 $\rightarrow_{\Sigma}$ 如下：

$$\Gamma \rightarrow_{(\Sigma;\mathcal{R})} \Gamma' \ \textit{holds iff}\ \Gamma \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma'$$

$$\Gamma \rightarrow_{\Sigma} \Gamma' \ \textit{holds iff}\ \Gamma \xrightarrow{\alpha}_{\Sigma} \Gamma' \ \textit{for}\ \alpha \in \{s[\mathbf{p}]\,!\,\mathbf{q} : \mathsf{m}(\mathbb{T}),\ s[\mathbf{p}][\mathbf{q}] : \mathsf{m}\}$$

我们用 $\rightarrow^{+}{}_{(\Sigma;\mathcal{R})}$（分别用 $\rightarrow^{+}_{\Sigma}$）和 $\rightarrow^{*}_{(\Sigma;\mathcal{R})}$（分别用 $\rightarrow^{*}_{\Sigma}$）表示它们各自的传递闭包和自反传递闭包。

上下文Γ使用缓冲区跟踪器 Σ来跟踪开放缓冲区。每当初始化一个新会话时，都会将其添加到 Σ中，详情见§ 3.4 条目 [T-ν]。目前只需了解，缓冲区跟踪器将通信限制为仅在受限会话上进行；因此，根据定义5 （良构性），它保证 Σ 中的所有会话都有会话缓冲区。

上下文归约（定义 9）对类型级通信进行建模。上下文Γ可以通过发送、通信或超时进行归约。根据 [Γ-$_{(\mathfrak{t})}$]，$\Gamma = s[\mathbf{p}] : \&_{i\in I}\{\mathbf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i,\ {}_{(\mathfrak{t})}.\mathbb{S}'\}$ 可以归约为超时分支延续类型 $\mathbb{S}'$，前提是 s位于缓冲区跟踪器中（即，会话 s存在一个缓冲区），并且至少有一个

$$[\Gamma\text{-}_{①}]\quad \frac{s \in \Sigma \qquad \exists k \in I:\ \mathsf{q}_k \notin \mathcal{R}\ (\mathsf{p})}{s[\mathsf{p}] : \&_{i\in I}\{\mathsf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i,\ {}_{①}.\mathbb{S}'\} \xrightarrow{s[\mathsf{p}]\ \oplus}_{(\Sigma;\mathcal{R})} s[\mathsf{p}] :\ \mathbb{S}'}$$

$$[\Gamma\text{-发送}_1]\quad \frac{s \in \Sigma \qquad k \in I}{s[\mathsf{p}] :\ \oplus_{i\in I}\{\mathsf{q}_i\,!\ \mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\} \xrightarrow{s[\mathsf{p}]!\ \mathsf{q}_k:\ \mathsf{m}_k(\mathbb{T}_k)}_{(\Sigma;\mathcal{R})} s[\mathsf{p}] :\ (\mathsf{q}_k\,!\ \mathsf{m}_k(\mathbb{T}_k)\ \cdot \epsilon;\ \ \mathbb{S}_k)}$$

$$[\Gamma\text{-发送}_2]\quad \frac{s \in \Sigma \qquad k \in I}{s[\mathsf{p}] : (\mathbb{M};\ \oplus_{i\in I}\{\mathsf{q}_i\,!\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\}) \xrightarrow{s[\mathsf{p}]!\ \mathsf{q}_k:\mathsf{m}_k(\mathbb{T}_k)}_{(\Sigma;\mathcal{R})} s[\mathsf{p}] : (\mathbb{M}\cdot\mathsf{q}_k\,!\,\mathsf{m}_k(\mathbb{T}_k)\ \cdot \epsilon;\ \mathbb{S}_k)}$$

$$[\Gamma\text{-通信}]\quad \frac{s \in \Sigma \qquad \exists k \in I : (\mathsf{p}, \mathsf{m}\ \mathbb{T}) = (\mathsf{p}_k, \mathsf{m}_k, \mathbb{T}_k)}{s[\mathsf{p}] : \mathsf{q}\,!\,\mathsf{m}(\mathbb{T})\ \cdot \mathbb{M},\ s[\mathsf{q}] : \&_{i\in I}\{\mathsf{p}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\ [,{}_{①}.\mathbb{S}']\} \xrightarrow{s[\mathsf{p}]![\mathsf{q}]:\mathsf{m}}_{(\Sigma;\mathcal{R})} s[\mathsf{p}] :\ \mathbb{M},\ s[\mathsf{q}] :\ \mathbb{S}_k}$$

$$[\Gamma\text{-}\mu]\quad \frac{s[\mathsf{p}] :\ \ \mathbb{S}[^{\mu\mathsf{t}.\mathbb{S}}/_{\mathsf{t}}] \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma'}{s[\mathsf{p}] :\ \ \mu\mathsf{t}.\mathbb{S} \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma'} \qquad [\Gamma\text{-合同行为}]\quad \frac{\Gamma_1 \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma_2}{\Gamma, \Gamma_1 \xrightarrow{\alpha}_{(\Sigma;\mathcal{R})} \Gamma, \Gamma_2}$$

图 5. 上下文约简规则

分支中的角色不可靠。后者可以避免对确定能够送达的通信进行超时处理。归约 $[\Gamma\text{-Snd}_1]$ 和 $[\Gamma\text{-Snd}_2]$通过将选择类型 $\oplus_{i\in I}\{\mathsf{q}_i\,!\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\}$归约为其某个续体 $\mathbb{S}_i$，并将发送的消息插入缓冲区类型，来模拟消息发送。二者的区别在于，$[\Gamma\text{-Snd}_1]$ 会在缓冲区类型之前未指定时创建该类型，而 $[\Gamma\text{-Snd}_2]$ 则将消息追加到已有的缓冲区类型中。两个角色之间的通信通过 $[\Gamma\text{-Com}]$，进行模拟：分支类型 $s[\mathsf{q}] : \&_{i\in I}\{\mathsf{p}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\ [,\rangle.\mathbb{S}']\}$ 从缓冲区类型 $s[\mathsf{p}] : \mathsf{q}\,!\,\mathsf{m}(\mathbb{T})\ \cdot \mathbb{M}$ 中取出消息，并归约为续体 $s[\mathsf{p}] :\ \mathbb{M},\ s[\mathsf{q}] :\ \mathbb{S}_k$。最后，$[\Gamma\text{-}\mu]$ 允许通过递归进行归约，而 $[\Gamma\text{-合同行为}]$则归约兼容组合的上下文中的子结构。

定义 10. 性质 φ_{s} 是类型上下文上的 $(\Sigma;\mathcal{R})$-安全性，当且仅当：

$[\text{S-}\mathcal{R}_1]\ \ \varphi_{\mathsf{s}}(\Gamma,\ s[\mathsf{p}] : \&_{i\in I}\{\mathsf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\}) \implies \forall i \in I : \mathsf{q}_i \in \mathcal{R}(\mathsf{p})$ $[\text{S-}\mathcal{R}_2]\ \ \varphi_{\mathsf{s}}(\Gamma,\ s[\mathsf{p}] : \&_{i\in I}\{\mathsf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i, \rangle.\mathbb{S}'\}) \implies \exists i \in I : \mathsf{q}_i \notin \mathcal{R}(\mathsf{p})$ $[\text{S-Com}]\ \ \varphi_{\mathsf{s}}(\Gamma,\ s[\mathsf{p}] : \&_{i\in I}\{\mathsf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\ [,\rangle.\mathbb{S}']\},\ s[\mathsf{q}] : \mathbb{M})$ 和 $\mathbb{M} \equiv \mathsf{p}\,!\,\mathsf{m}(\mathbb{T})\ \cdot \mathbb{M}'$ 和 $\exists k \in I : \mathsf{q}_k = \mathsf{q} \wedge \mathsf{m}_k = \mathsf{m} \implies \mathbb{T}_k = \mathbb{T}$ $[\text{S-}\mu]\ \ \varphi_{\mathsf{s}}(\Gamma,\ s[\mathsf{p}] :\ \mu\,\mathsf{t}.\mathbb{S}) \implies \varphi_{\mathsf{s}}(\Gamma,\ s[\mathsf{p}] :\ \mathbb{S}[^{\mu\mathsf{t}.\mathbb{S}}/_{\mathsf{t}}])$ $[\text{S-}\rightarrow]\ \ \varphi_{\mathsf{s}}(\Gamma)$ 和 $\Gamma \rightarrow_{(\Sigma;\mathcal{R})} \Gamma' \implies \varphi_{\mathsf{s}}(\Gamma')$

如前所述，我们的类型系统是一个通用的系统，不使用句法方法来强制实现一致通信。因此，我们在定义安全性性质 10 中，在类型上下文上定义了该性质，10 以此保证主体归约及其他定理（见§ 4）。

我们称 φ_{s} 为保证主体归约所需的最大安全性。根据实现的要求，该性质可以用更具体的条件重新实例化（如§ 5中所示）。具体而言，[S-$\mathcal{R}_1$] 和 [S-$\mathcal{R}_2$] 分别确保仅当通信可靠时才不定义超时，以及当通信不可靠时定义超时。条件 [S-Com] 确保通信中的消息具有匹配的负载类型。最后，[S-μ] 通过递归展开保持 φ_{s}，而 [S-$\rightarrow$] 要求在上下文归约之后仍满足安全性。

3.4 类型规则

我们的类型系统由图 6中的类型规则定义。下面我们将详细解释这些规则。类型判断的形式为：$\Theta \cdot \Gamma \vdash P$ 读作“进程 P 在类型上下文 Θ 和 Γ 下具有良好类型”；以及 $\Gamma \vdash d : \mathbb{T}$ 读作“值（或变量、或信道）d 在类型上下文 $\mathbb{T}$ 下的类型为 Γ”。

[T-0] 静止进程**0** 由一个“结束类型化”的上下文进行类型判定，该上下文由谓词 结束(Γ)—定义于图 7。该谓词成立的条件为：（i）若$\Gamma = \emptyset$；（ii）若 Γ 由变量组成，则当所有变量的类型均为基本类型，或可由 结束进行类型判定时，该谓词成立；（iii）若 Γ 由带角色的会话组成，则当所有信道均可由 结束进行类型判定时，该谓词成立。

[T-Var] 变量或带角色的会话 c 在仅包含 c 到 $\mathbb{T}$的映射的上下文中具有类型 $\mathbb{T}$。

[T-Val] 若值 v 属于基本类型 $\mathbb{B}$ 的取值集合，则其类型为该基本类型。例：$42 : \mathbb{N}$ 在空上下文 $\emptyset$ 下可进行类型判定，因为 $42 \in \mathbb{N}$。[T-X] 若进程变量 X 在上下文Θ中的类型为由 $\mathbb{T}_1, \ldots, \mathbb{T}_n$ 组成的 n元组，则 Θ 将该进程变量映射到相同的 n元组类型。

[T-$\oplus$] 选择进程 $c \oplus [\mathbf{q}_k]! \, \mathsf{m}_k\langle d\rangle . P$在一个上下文中进行类型检查，该上下文将发送信道 c映射到选择会话类型 $\oplus_{i\in I}\{\mathbf{q}_i \, ! \, \mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\}$，其中一个选择选项与发送进程相匹配，即，$k \in I$。上下文应将载荷 d 映射到选择中指示的类型（$\mathbb{T}_k$），而续体进程 P 应依据续体类型 $\mathbb{S}_k$进行类型检查。
[T- &] 分支进程 $c \,\&_{i\in I}\{[\mathsf{p}_i] \, ? \, \mathsf{m}_i(x_i).P_i\}$ 在一个上下文中进行类型检查，该上下文将接收信道 c映射到分支类型 $\&_{i\in I}\{\mathsf{p}_i \, ? \, \mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\}$，其中每个分支的所有角色和消息标签都相匹配。每个续体进程 P_i 都必须依据续体类型 $\mathbb{S}_i$ 进行类型检查，且载荷的类型由 $\mathbb{T}_i$指定。如果进程是超时分支 $c \,\&_{i\in I}\{[\mathsf{p}_i] \, ? \, \mathsf{m}_i(x_i).P_i, \rightarrow . Q\}$，那么它应由一个同样包含超时续体 $\&_{i\in I}\{\mathsf{p}_i \, ? \, \mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i, \rightarrow . \mathbb{S}'\}$的会话类型进行类型检查，而超时进程 Q 必须由 $\mathbb{S}'$进行类型检查。[T-Call] 进程调用 $X\langle d_1, \ldots, d_n\rangle$ 在以下情况下类型正确：Θ 将进程变量映射到一个包含 n 个类型的元组 $\mathbb{T}_1, \ldots, \mathbb{T}_n$，并将每个参数 d_i 映射到对应的 $\mathbb{T}_i$（对于 $i \in 1..n$）。Γ 中的任何剩余信道都不能是开放的，因此必须是终止类型。

[T-0]
$$\frac{\text{结束}(\Gamma)}{\Theta \cdot \Gamma \vdash \mathbf{0}}$$

[T-变量]
$$\frac{}{c : \mathbb{T} \vdash c : \mathbb{T}}$$

[类型值]
$$\frac{v \in \mathbb{B}}{\emptyset \vdash v : \mathbb{B}}$$

[T-X]
$$\frac{\Theta(X) = \mathbb{T}_1, \ldots, \mathbb{T}_n}{\Theta \vdash X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n}$$

[T-⊕]
$$\frac{\Gamma_1 \vdash c\colon \oplus_{i \in I}\{\mathbf{q}_i\,!\ \mathsf{m}_i\ (\mathbb{T}_i)\ .\mathbb{S}_i\} \qquad k \in I \qquad \Gamma_2 \vdash d\colon \mathbb{T}_k \qquad \Theta \cdot \Gamma,\ c\colon \mathbb{S}_k \vdash P}{\Theta \cdot \Gamma,\ \Gamma_1,\ \Gamma_2 \vdash c \oplus [\mathbf{q}_k]!\ \mathsf{m}_k\langle d\rangle.\, P}$$

[T- &]
$$\frac{\Gamma' \vdash c\colon \&_{i \in I}\{\mathbf{p}_i\,?\ \mathsf{m}_i\ (\mathbb{T}_i)\ .\mathbb{S}_i\,[,\ _{\textcircled{1}}.\mathbb{S}']\} \quad [\Theta \cdot \Gamma, c\colon \mathbb{S}' \vdash Q] \qquad \forall i \in I \cdot \Theta \cdot \Gamma, x_i\colon \mathbb{T}_i, c\colon \mathbb{S}_i \vdash P_i}{\Theta \cdot \Gamma, \Gamma' \vdash c\,\&_{i \in I}\{[\mathbf{p}_i]\,?\,\mathsf{m}_i(x_i).P_i\,[,\ _{\textcircled{1}}.Q]\}}$$

[T-Call]
$$\frac{\Theta \vdash X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \qquad \text{结束}(\Gamma') \qquad \forall i \in 1..n \cdot \Gamma_i \vdash d_i : \mathbb{T}_i}{\Theta \cdot \Gamma_1, \ldots, \Gamma_n, \Gamma' \vdash X\langle d_1, \ldots, d_n\rangle}$$

[T-定义]
$$\frac{\Theta、\ X\colon\ \mathbb{T}_1,\ \ldots\ldots,\ \mathbb{T}_n \cdot x_1\colon\ \mathbb{T}_1,\ \ldots\ldots,\ x_n\colon\ \mathbb{T}_n \vdash P \qquad \Theta, X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma \vdash Q}{\Theta \cdot \Gamma \vdash \text{定义}\ X(x_1\colon \mathbb{T}_1, \ldots, x_n\colon \mathbb{T}_n) = P \text{于}\ Q}$$

[T-+]
$$\frac{\Theta \cdot \Gamma \vdash P_1 \qquad \Theta \cdot \Gamma \vdash P_2}{\Theta \cdot \Gamma \vdash P_1 + P_2}$$

[T-Lift]
$$\frac{\Theta \cdot \Gamma \vdash P}{\Theta \cdot \Gamma \vdash_{\emptyset} P}$$

[T-ϵ]
$$\frac{\text{gc}(\Gamma)}{\Theta \cdot \Gamma \vdash_{\{s\}} s : \epsilon}$$

[T-σ_1]
$$\frac{\Theta \cdot \Gamma' \vdash_{\{s\}} s : \sigma \qquad \Gamma \vdash w : \mathbb{T}}{\Theta \cdot \Gamma, \Gamma', s[\mathbf{p}] : \mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \epsilon \vdash_{\{s\}} s : (\mathbf{p} \triangleright \mathbf{q} \triangleleft \mathsf{m}\langle w\rangle) \cdot \sigma}$$

[T-σ_2]
$$\frac{\Theta \cdot \Gamma', s[\mathbf{p}] : \mathbb{M} \vdash_{\{s\}} s\colon\ \sigma \qquad \Gamma \vdash w\colon\ \mathbb{T}}{\Theta \cdot \Gamma,\ \Gamma',\ s[\mathbf{p}]\colon\ \mathbf{q}\,!\ \ \mathsf{m}\ (\mathbb{T})\ \cdot \mathbb{M} \vdash_{\{s\}} s : (\mathbf{p} \triangleright \mathbf{q} \triangleleft \mathsf{m}\langle w\rangle) \cdot \sigma}$$

[T-σ_w]
$$\frac{\Gamma = (\Gamma_0 \rightsquigarrow \Gamma_1)\ \Gamma_2 \qquad \Theta \cdot \Gamma_1 \vdash_{\Sigma} s\colon \sigma \qquad \text{gc}(\Gamma_0, \Gamma_2)}{\Theta \cdot \Gamma \vdash_{\Sigma} s\colon \sigma}$$

[T-|]
$$\frac{\Theta \cdot \Gamma_1 \vdash_{\Sigma_1} P_1 \qquad \Theta \cdot \Gamma_2 \vdash_{\Sigma_2} P_2 \qquad \Sigma_1 \cap \Sigma_2 = \emptyset}{\Theta \cdot \Gamma_1, \Gamma_2 \vdash_{\Sigma_1 \cup \Sigma_2} P_1 \mid P_2}$$

[T-ν]
$$\frac{\Gamma' = \{s[\text{段落}]\colon\ \tau_{\mathbf{p}}\}_{\mathbf{p} \in \rho} \qquad s \notin \Gamma \qquad (\{s\}\,;\,\mathcal{R})\text{-}\varphi_{\mathbf{s}}(\Gamma') \qquad \Theta \cdot \Gamma, \Gamma' \vdash_{\Sigma} P}{\Theta \cdot \Gamma \vdash_{\Sigma \setminus \{s\}} (\nu s\colon \Gamma')\ P}$$

图 6. 类型规则

$$\frac{}{\text{结束}(\emptyset)} \qquad \frac{\forall i \in 1..n \ \cdot \ \text{basic}(\mathbb{T}_i) \quad \vee \quad x_i : \mathbb{T}_i \ \vdash \ x_i : \text{结束}}{\text{结束}(x : \mathbb{T}_1, \ldots, x_n : \mathbb{T}_n)}$$

$$\frac{\text{结束}(\Gamma_1) \qquad \text{结束}(\Gamma_2)}{\text{结束}(\Gamma_1, \Gamma_2)} \qquad \frac{\forall i \in 1..n, \ \text{段落} \in \rho \ \cdot \ s_i[\text{段落}] : \tau_i \ \vdash \ s_i[\mathbf{p}] : \text{结束}}{\text{结束}(s_i[\mathbf{p}] : \tau_1, \ldots, s_i[\mathbf{p}] : \tau_n)}$$

图 7. 谓词 结束(Γ)

$$\frac{}{\mathsf{gc}(\emptyset)} \qquad \frac{\mathsf{gc}(\Gamma)}{\mathsf{gc}(\Gamma, s[\mathbf{p}] : \epsilon)} \qquad \frac{\text{basic}(\mathbb{T}) \qquad \mathsf{gc}(\Gamma, s[\mathbf{p}] : \mathbb{M})}{\mathsf{gc}(\Gamma, s[\mathbf{p}] : \mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \mathbb{M})}$$

$$\frac{\Gamma = \Gamma', s'[\mathbf{p}'] : \mathbb{T} \qquad \mathsf{gc}(\Gamma', s[\mathbf{p}] : \mathbb{M})}{\mathsf{gc}(\Gamma, s[\mathbf{p}] : \mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \mathbb{M})}$$

图 8。 垃圾收集器谓词 $\mathsf{gc}(\Gamma)$

$$s[\mathrm{p}] : \mathrm{q}\,!\,\mathrm{m}(\mathbb{T}) \ \cdot \epsilon \rightsquigarrow \Gamma, s[\mathrm{p}] : \ \mathbb{M} = \Gamma, s[\mathrm{p}] : \mathrm{q}\,!\,\mathrm{m}(\mathbb{T}) \ \cdot \mathbb{M} s[$$
$$\mathrm{p}] : \mathrm{q}\,!\,\mathrm{m}(\mathbb{T}) \ \cdot \epsilon \rightsquigarrow \Gamma \ \textit{when} \ s[\mathrm{p}] : \ \mathbb{M} \notin \Gamma = \Gamma, s[\mathrm{p}] : \mathrm{q}\,!\,\mathrm{m}(\mathbb{T}) \ \cdot \epsilon$$

图9。消息插入函数 $\Gamma' \rightsquigarrow \Gamma$

[T-Def] 如果上下文 P 是自包含的，则进程声明 $X(x_1 : \mathbb{T}_1, \ldots, x_n : \mathbb{T}_n) = P$ 是良类型的；也就是说，包含声明参数类型的上下文（以及任何先前的 Θ）应能为 P 定型。进程定义 def $X(x_1 : \mathbb{T}_1, \ldots, x_n : \mathbb{T}_n) = P$in Q 在 $\Theta \cdot \Gamma$ 下定型的条件是：其声明是良类型的，且 Q 在 Γ 下定型，并且 Θ 与新的进程变量组合。[T-+] 如果各个进程在独立情况下均由 $\Theta \cdot \Gamma$ 定型，则非确定性选择是良类型的。这与 case 或 if-then-else 进程的定型方式一致。[T-Lift] 我们为类型判断 $\Theta \cdot \Gamma \vdash P$ 添加缓冲区跟踪器，得到 $\Theta \cdot \Gamma \vdash_\Sigma P$，表示 Σ 中的会话出现在 P 中。如果无缓冲区判断（$\vdash$）能够为 P 定型（使用迄今提及的规则），则提升规则会为该类型判断添加一个空缓冲区跟踪器。[T-ϵ] 在标准异步多方会话类型理论中，空缓冲区 $s : \epsilon$ 在空上下文 $\emptyset$ 下定型，从而确保上下文中的缓冲区类型与会话缓冲区中的消息一一对应。然而，由于我们的演算对消息丢失进行建模，上下文中可能包含对应于已从进程缓冲区中丢弃的消息的缓冲区类型。因此，我们的理论会在经过垃圾回收的 Γ 下为 $s : \epsilon$ 定型。谓词 gc 在图 8 中定义，它指出有效的剩余类型 gc(Γ) 包括：(i) 空；(ii) 空缓冲区类型；(iii) 具有基本类型载荷的消息缓冲区类型；或 (iv) 具有信道载荷的消息缓冲区类型，且这些载荷在 Γ 下定型。

[T-σ_1] [T-σ_2] 会话缓冲区 $s : (\mathsf{p} \triangleright \mathsf{q} \triangleleft \mathsf{m}\langle w \rangle) \cdot \sigma$ 中的条目，其定型上下文包含从 $s[\mathsf{p}]$ 到缓冲区类型 $\mathsf{q} ! \mathsf{m}(\mathbb{T}) \cdot \mathbb{M}$ 的映射，与接收方和消息标签相匹配。消息载荷 w 必须具有缓冲区类型所指示的类型 $\mathbb{T}$，而缓冲区延续 $s : \sigma$ 应在其非空时（[T-σ_2]）在缓冲区延续类型 $\mathbb{M}$ 下定型。

[T-σ_w] 弱化允许在更大的上下文下为会话缓冲区定型，前提是新增内容可以被垃圾回收，并通过消息插入函数插入原上下文（图 9）。这是一个偏函数：它要么将消息追加到现有缓冲区类型中，要么将其插入为新缓冲区类型的头部。换句话说，弱化允许在包含可被垃圾回收的无关类型的更大上下文下为缓冲区定型。

[T-|] 若进程 P_1 的类型为 Γ_1，且进程 P_2 的类型为 Γ_2，则组合 Γ_1，Γ_2 为并行组合 定型 $P_1 \mid P_2$。此外，还要求并行进程不能 各自都包含同一会话的缓冲区 s。这保证每个受限会话只有一个会话缓冲区。

[T-ν] 会话限制（$\nu s : \Gamma'$） P要求将会话 s实例化为一个 Γ'映射，将每个带角色的会话映射到其会话缓冲区类型。 $\varphi_{\mathsf{s}}(\Gamma')$ 必须成立以确保主体归约，详见 § 3.3。会话 s 不应出现在先前的上下文 Γ中，而进程 P 应在先前上下文与新实例化上下文的组合下定型，并使用更新后的缓冲区跟踪器 $\Theta \cdot \Gamma$， $\Gamma' \vdash_\Sigma P$（因为 s的缓冲区包含在 P中）。

示例 4（Ping Pong：类型上下文）。回顾 Ping Pong 示例，整个系统可以描述为三个分别代表角色的进程 p、q、r 的并行组合，以及一个空缓冲区；该系统在类型上下文 Γ 下满足以下类型假设。

$$\Gamma = \{s[\mathsf{p}] : \mathbb{S}_{\mathsf{p}},\ s[\mathsf{q}] : \mathbb{S}_{\mathsf{q}}, s[\mathsf{r}] : \mathbb{S}_{\mathsf{r}}\}$$

$$P_{ping} = (\nu s : \Gamma)\ P_{\mathsf{p}} \mid P_{\mathsf{q}} \mid P_{\mathsf{r}} \mid s : \epsilon$$

4 类型属性

MAG 多方会话类型系统针对 π 进程的主要结果是主体归约(定理 1) 和 会话保真性 (定理 2)。需要注意的是，我们的结果以可靠性函数 $\mathcal{R}$为参数。因此，我们提出的定理适用于任意可靠性配置，即从完全没有可靠通信，到完全可靠网络。

为了协调类型与进程中的可靠性假设，我们定义可靠进程归约 $\longrightarrow_{\mathcal{R}}$，使得 $\longrightarrow_{\mathcal{R}} \subseteq \longrightarrow$。

定义 11（可靠进程归约）。可靠进程归约$\longrightarrow_{\mathcal{R}}$ 通过以下方式归纳定义：采用 $\longrightarrow$ (图 2) 中相同的约简规则，但作如下更改 [3]:

[**R-**$_{\textcircled{1}}$] $s[\mathsf{q}] \,\&_{i \in I} \{[\mathsf{p}_i] ? \mathsf{m}_i(x_i).P_i, \ldots_{\textcircled{1}}. Q\} \mid s : \sigma \longrightarrow_{\mathcal{R}} Q \mid s : \sigma$ 如果 $\exists k \in I : \mathsf{p}_k \notin \mathcal{R}(\mathsf{q})$

[R-↓] $s : (\mathsf{p} \triangleright \mathsf{q} \triangleleft \mathsf{m}\langle w \rangle) \cdot \sigma \longrightarrow_{\mathcal{R}} s : \sigma$ 适用于$\mathsf{q} \notin \mathcal{R}(\mathsf{p})$

[3] 对于完全不可靠网络，$i\,e$， $\forall \mathsf{p} .\in \boldsymbol{\rho}\ \ \mathcal{R}(\mathsf{p}) \cdot = \emptyset$， $\longrightarrow_{\mathcal{R}}$ 等价于 $\longrightarrow$。

直观来说，可靠进程归约在可靠通信中忽略网络故障。具体而言，只有当分支中 [**R-**$\oplus$] 至少有一个角色不可靠时，才能进行超时归约；而消息丢失 [**R-↓**] 只会发生在不从发送方的角度来看可靠的消息上。这确保了可靠通信中的消息不会被忽略或丢失。我们的技术结果及任何辅助结果的证明见附录A–C。

4.1 主体归约

利用 $\longrightarrow_{\mathcal{R}}$，我们现在给出主体归约的结果。直观而言，主体归约指出：如果进程 P 在安全上下文中具有类型，并且 P 可靠地归约到某个进程 P'，那么上下文也会（经过 0 步或 1 步）归约到一个安全上下文，该上下文为新进程 P'赋予类型。

定理 1（主体归约）。

$$\Theta \cdot \Gamma \vdash_{\Sigma} P \quad and \quad (\Sigma;\mathcal{R})\text{-}\varphi_{\mathsf{s}}(\Gamma) \quad and \quad P \to_{\mathcal{R}} P' \implies \exists \Gamma' : \Gamma \to_{(\Sigma;\mathcal{R})}^{\{0,1\}} \Gamma' \quad and \quad (\Sigma;\mathcal{R})\text{-}\varphi_{\mathsf{s}}(\Gamma') \quad and \quad \Theta \cdot \Gamma' \vdash_{\Sigma} P'$$

类型系统的一项关键创新成果是，运行时不会发生意外的网络故障，也就是说，对于不可靠通信，进程总是有一个已定义的故障处理子协议。这一结论源于安全性属性的定义 φ_{s}（定义 10），并且在主体归约过程中仍然成立。我们在推论 1中陈述这一结果。更准确地说，该推论指出，对于不可靠通信，超时分支必定有定义。反向结论见推论2，也就是说，对于仅包含可靠来源的分支，超时没有定义。

推论 1（故障处理安全性）。给定可靠性函数 $\mathcal{R} : \mathsf{p} \notin \mathcal{R}(\mathsf{q})$ 和 $\Theta \cdot \Gamma \vdash_{\Sigma} P$，其中 $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$ 且 $P \longrightarrow_{\mathcal{R}}^{*} P' \equiv \mathbb{C}[Q]$ 蕴含$Q \neq s[\mathsf{q}]\ \&_{i\in I}\{\ldots, [\mathsf{p}]\,?\,\mathsf{m}(x).Q'\}$。也就是说，$Q$ 不能是 q 处从 p 接收消息的分支，且不定义超时。

推论 2（可靠性遵循）。给定可靠性函数 $\mathcal{R} :\mathcal{R}(\mathsf{q}) = \mathfrak{R}_{\mathsf{q}}$ and $\Theta \cdot \Gamma \vdash_{\Sigma} P$with $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$ and $P \longrightarrow_{\mathcal{R}}^{*} P' \equiv \mathbb{C}[Q]$ implies$Q \neq s[\mathsf{q}]\ \&_{i\in I}\{[\mathsf{p}_i]\,?\,\mathsf{m}_i(x_i).Q_i, \rangle. Q'\}$ st: $\forall i \in I$: $\mathsf{p}_i \in \mathfrak{R}_{\mathsf{q}}$. 也就是说，$Q$ 不能是在 q 处仅接收来自可靠角色 p_i的分支，并定义一个超时。

4.2 会话保真性

会话保真性 陈述了主体归约的反向蕴含，即，如果 Γ为进程赋予类型 P，且 Γ 能够归约，那么 P 至少能够与其中一个上下文归约相匹配。

因此，可以从进程 P 的类型上下文行为中推导出其相关属性 Γ（正如我们将在定理3中看到的）。然而，正如 Scalas 和 Yoshida 所示 [32, 第 5 节。2], 这一结果并不适用于所有类型正确的

进程。具体而言，以下情况会违反会话保真性：(*i*) 无产出地无限递归的进程（例如 定义 $X(x) = X\langle x\rangle$ 在 $X\langle s[\mathsf{p}]\rangle$）；以及 (*ii*) 在多方会话之间交错通信而导致死锁的进程。因此，我们通过调整[32, 定义 5。3]。

定义 12（会话保真性的条件）。假设 $\emptyset \cdot \Gamma \vdash_{\{s\}} P$。我们称 P：

1. 具有受保护定义，当且仅当 P 中每个形式如下的进程定义

$$\mathsf{def}\ X(x_1 : \mathbb{T}, \ldots, x_n : \mathbb{T}) = Q\ \mathsf{in}\ P'$$

$\forall j \in 1..n$：如果 $\mathbb{T}_j$ 是会话类型，那么进程调用 $Y\langle \ldots, x_j, \ldots\rangle$ 只能作为子项出现在 Q 中

$$x_j\,\&_{i\in I}\{[\mathsf{p}_i]\,?\,\mathsf{m}_i(y_i).P_i[,\ \underset{\uparrow}{\ }.P_t]\}\quad\text{or}\quad x_j \oplus [\mathsf{p}]\,!\,\mathsf{m}\langle y\rangle.\,P'',$$

即，在使用 x_j 进行输入或输出之后。

2. 根据 Γ 仅在 s中扮演角色 p，当且仅当：(i) P 具有受保护定义（见 1）；(ii) fv(P) $= \emptyset$；(iii) $\Gamma = \Gamma_0, s[\mathsf{p}] : \tau$，其中 $\tau \neq$ 以 end 结束，且 end(Γ_0)；(iv) 对于 P的所有形如 $(\nu s' : \Gamma')\ P'$ 的子项，end(Γ')。

我们称“P仅在 p 中扮演角色 s”当且仅当 $\exists \Gamma$：$\emptyset \cdot \Gamma \vdash_{\{s\}} P$且条件 *2* 成立。

定义 12 在条件 1中形式化受保护递归，并在条件 2中提出了针对给定会话仅扮演单一角色的概念。这些条件共同确保定理 2中所述的会话保真性对所有类型正确的进程均成立。

定理 2（会话保真性）。假设 $\emptyset \cdot \Gamma \vdash_\Sigma P$with ($\Sigma$; $\mathcal{R}$)-$\varphi_s(\Gamma)$,$P \equiv (\Pi_{\mathsf{p}\in I}\, P_\mathsf{p}) \mid s : \sigma$ and $\Gamma = \bigcup_{\mathsf{p}\in I} \Gamma_\mathsf{p}$，且对于每个 P_p：(i) $\emptyset \cdot \Gamma_\mathsf{p} \vdash_\Sigma P_\mathsf{p}$，并且 (ii) P_p 为 0（在 up-to-$\equiv$ 的意义下）或仅在 s 中扮演角色 p，根据 Γ_p。那么，
$\Gamma \longrightarrow_{(\Sigma;\mathcal{R})}$ 蕴含 $\exists \Gamma', P'$：(i) $\Gamma \longrightarrow_{(\Sigma;\mathcal{R})} \Gamma'$，(ii) $P \longrightarrow^+_{\mathcal{R}} P'$，(iii) $\emptyset \cdot \Gamma' \vdash_\Sigma P'$with ($\Sigma$; $\mathcal{R}$)-$\varphi_s(\Gamma')$，(iv) $P' = (\Pi_{\mathsf{p}\in I}\, P'\mathsf{p}) \mid s : \sigma'$ and $\Gamma' = \bigcup_{\mathsf{p}\in I} \Gamma'\mathsf{p}$，并且 (v) 对每个 P'_p：$\emptyset \cdot \Gamma'\mathsf{p} \vdash_\Sigma P'\mathsf{p}$，且 $P'\mathsf{p}$ 为 0（在 up-to-$\equiv$ 的意义下）或仅在 s 中扮演角色 p，根据 $\Gamma'\mathsf{p}$。

4.3 进程性质

会话保真性方面的结果（§ 4.2）使我们能够根据 MAG 程序的类型推断其运行时属性π。我们将通过定义进程上理想的运行时属性（定义 13）；表达这些属性在类型级上的等价关系（定义 14）；并给出进程性质验证的结果进程性质验证（定理 3），从而将进程性质与其类型级等价关系联系起来。

根据定义 13，以下进程指：**(i)** $\mathcal{R}_F$-通信安全 (相对于[32])，如果它通过可靠归约到达通信结束，且缓冲区中没有剩余消息；**(ii)**无死锁，如果它要么可以归约，要么处于静止状态；**(iii)**终止，如果它无死锁，且能在有限步数内到达静止状态

；**(iv)** 永不终止，如果它始终可以无限归约；以及**(v)** 活性，如果对于它能够归约到的每个可靠分支，它最终都能够归约到某个分支延续。我们无需考虑带超时的分支，因为进程总能跨过超时进行归约，因此这些分支显然具有活性。

定义 13（进程性质，改编自 [32]）。对于某个可靠性函数 $\mathcal{R}$ 和完整可靠性函数 $\mathcal{R}_F$，进程 P 称为：

(i) $\mathcal{R}_F$-通信安全，当且仅当

$$P \longrightarrow^*_{\mathcal{R}_F} P' \not\longrightarrow_{\mathcal{R}_F} \quad and \quad P' = \mathbb{C}[s : \sigma] \quad implies \quad \sigma = \epsilon;$$

(ii) 无死锁，当且仅当 $P \longrightarrow^*_{\mathcal{R}} P' \not\longrightarrow_{\mathcal{R}}$ 蕴含 $P' \equiv \mathbf{0}$;

(iii) 终止，当且仅当它无死锁，且

$$\exists\, i \; finite \text{ st: } \forall n \geq i : P = P_0 \longrightarrow_{\mathcal{R}} P_1 \longrightarrow_{\mathcal{R}} \cdots \longrightarrow_{\mathcal{R}} P_n \quad implies \quad P_n \not\longrightarrow_{\mathcal{R}} ;$$

(iv) 永不终止，当且仅当 $P \longrightarrow^*_{\mathcal{R}} P'$ 蕴含 $P' \longrightarrow_{\mathcal{R}}$;

(v) 具有活性，当且仅当 $P \longrightarrow^*_{\mathcal{R}} P' \equiv \mathbb{C}[Q]$ 蕴含

$$if \;\; Q = c\,\&_{i\in I}\{[\mathbf{q}_i]\,?\,\mathsf{m}_i(x_i).Q'_i\}, \;\; then \quad \exists\, \mathbb{C}', k \in I, w : P' \longrightarrow^*_{\mathcal{R}} \mathbb{C}'[\,Q'_k[{}^{w}/_{x_k}]\,].$$

请注意，与其他研究不同 [4,32], ，我们的活性定义只涉及接收进程，而不涉及发送进程。通常，活性还要求已发送的消息——在 MAGπ 中，即会话缓冲区中的任何消息——最终总会被消费。然而，由于我们的演算所建模的故障，进程即使具有活性，缓冲区中仍可能有未消费的消息（例如，消息延迟导致超时的结果）。此外，对于一个 $\mathcal{R}_F$-通信安全 进程，在可靠 case 下，所有已发送的消息都会被消费。因此，传统的活性定义在可靠网络配置下仍然成立，而我们的新定义则基于故障假设，提供了尽可能强的保证。

现在，我们将介绍上述进程性质在类型层面的等价关系。对于活性，我们将其推广为最大活性属性，正如在定义10中对安全性所做的那样，从而允许用户在需要时定义更精细的活性概念。

根据定义 14 ，类型上下文具有以下性质： **(i)** $\mathcal{R}_F$-通信安全： 当它无法再可靠地归约时，不含任何已填充的缓冲区类型； **(ii)** 无死锁： 它无法再归约的原因是它已被 结束 类型化（并且可能由于网络故障而遗留了一些可进行垃圾回收的类型）； **(iii)** 终止： 它无死锁，并且能在有限步数内到达协议结束； **(iv)** 永不终止： 它始终可以无限归约；以及 **(v)** 活性： 对于它能够归约到的每个可靠分支，都存在一系列步骤能够归约到该分支的一个续体。

定义 14（类型上下文性质）。对于某个可靠性函数 $\mathcal{R}$、完整可靠性函数 $\mathcal{R}_F$以及会话集合 Σ，我们称上下文 Γ 为：

(i) $(\Sigma;\ \mathcal{R}_F)$-通信安全，当且仅当

$$\Gamma \longrightarrow^*_{(\Sigma;\mathcal{R}_F)} \Gamma' \not\longrightarrow_{(\Sigma;\mathcal{R}_F)} \textit{and}\ s[\mathtt{p}] : \mathbb{M} \in \Gamma' \ \textit{implies}\ \mathbb{M} = \epsilon;$$

(ii) 当且仅当 $(\Sigma;\mathcal{R})$-无死锁

$$\Gamma \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma' \not\longrightarrow_{(\Sigma;\mathcal{R})} \quad \textit{implies}\ \ \Gamma' = \Gamma'_0, \Gamma''\ \mathrm{st}:\ \mathtt{end}(\Gamma'_0)\ \textit{and}\ \mathtt{gc}(\Gamma'');$$

(iii) $(\Sigma;\ \mathcal{R})$-终止，当且仅当其为 $(\Sigma;\ \mathcal{R})$-无死锁，且 $\exists i$ 有限 st:

$$\forall n \geq i : \Gamma = \Gamma_0 \longrightarrow_{(\Sigma;\mathcal{R})} \Gamma_1 \longrightarrow_{(\Sigma;\mathcal{R})} \cdots \longrightarrow_{(\Sigma;\mathcal{R})} \Gamma_n \ \textit{implies}\ \Gamma_n \not\longrightarrow_{(\Sigma;\mathcal{R})};$$

(iv) $(\Sigma;\ \mathcal{R})$-永不终止，当且仅当 $\Gamma \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma'$ 蕴含 $\Gamma' \longrightarrow_{(\Sigma;\mathcal{R})}$; (v) $(\Sigma;\mathcal{R})$-具有活性，当且仅当其遵循某个活性属性 $(\Sigma;\mathcal{R})$-$\varphi_{\mathtt{L}}$ st:

$$\begin{array}{l}
(\Sigma;\mathcal{R})\text{-}\varphi_{\mathtt{L}}(\Gamma,\ s[\mathtt{p}] : \mathbb{S})\ \textit{and}\ \mathbb{S} = \&_{i\in I}\{\mathtt{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\} \\
\qquad\qquad \Longrightarrow\ \ \exists \Gamma', k \in I : \Gamma, s[\mathtt{p}] : \mathbb{S} \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma', s[\mathtt{p}] : \mathbb{S}_k \\
(\Sigma;\mathcal{R})\text{-}\varphi_{\mathtt{L}}(\Gamma,\ s[\mathtt{p}] : \mu \mathrm{t}.\mathbb{S}) \ \Longrightarrow\ (\Sigma;\mathcal{R})\text{-}\varphi_{\mathtt{L}}(\Gamma,\ s[\mathtt{p}] : \mathbb{S}[^{\mu \mathrm{t}.\mathbb{S}}/_{\mathrm{t}}]) \\
(\Sigma;\mathcal{R})\text{-}\varphi_{\mathtt{L}}(\Gamma)\ \textit{and}\ \Gamma \rightarrow_{(\Sigma;\mathcal{R})} \Gamma' \ \Longrightarrow\ (\Sigma;\mathcal{R})\text{-}\varphi_{\mathtt{L}}(\Gamma)
\end{array}$$

现在，我们可以利用这些类型层面的等价性质来推断其所类型化进程的行为。我们在定理 3 中给出我们的结果，该定理正式表明，在会话保真性（定理 2）中给出的相同假设下，如果一个进程在某个类型上下文下进行类型检查，并且该上下文满足某项性质，那么该进程本身也满足同一性质。

定理 3（进程性质验证）。假设：$\emptyset \cdot \Gamma \vdash_\Sigma P$与 $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$，$P \equiv (\Pi_{\mathtt{p}\in I} P_{\mathtt{p}}) \mid s :\ \sigma$ 以及 $\Gamma = \bigcup_{\mathtt{p}\in I} \Gamma_{\mathtt{p}}$。此外，对于每个 $P_{\mathtt{p}}$: (i) $\emptyset \cdot \Gamma_{\mathtt{p}} \vdash_\Sigma P_{\mathtt{p}}$，以及 (ii) $P_{\mathtt{p}} \equiv \mathbf{0}$ 或仅 $P_{\mathtt{p}}$ 在 s中扮演角色 p，依据 $\Gamma_{\mathtt{p}}$。那么，$\forall \phi \in \{\mathcal{R}_F$-通信安全、无死锁、终止、永不终止、具有活性$\}$；如果 $(\Sigma;\ \mathcal{R})$-$\phi(\Gamma)$，则 P 是 ϕ。

4.4 可判定性

由于 MAGπ 是图灵完备的，因此，确定定义 13 中列出的13中的进程的性质 [5]是不可判定的。我们推广后的理论有一个优点：可以从可判定的类型层面的性质推断出不可判定的进程性质。

定理 4（可判定性）。如果$(\Sigma;\ \mathcal{R})$-$\phi(\Gamma)$）是可判定的，那么“$\Theta \cdot \Gamma \vdash_\Sigma P$带有$(\Sigma;\ \mathcal{R})$-$\phi(\Gamma)$”是可判定的。

我们的可判定性结果（定理 4）指出，对于任何可判定的类型级性质，带有该性质的类型检查都是可判定的。然而，由于 MAGπ是异步的，我们尚无关于 ϕ可判定性的结果。相反，正如 [32, 第 7], 节所讨论的，异步类型理论中的类型级性质在某些情况下是不可判定的。这是将缓冲区类型与会话类型配对的结果——这使类型系统具备图灵完备的能力 [3, （定理 2）5]。Scalas 和 Yoshida 通过两种方法解决这一问题：（i）标准全局类型

生成可通过可判定的一致性属性捕获的类型上下文；以及（ii）限制消息缓冲区的大小，使相关性质可判定。前一种方法通过限制通信，使其表达能力与全局类型相匹配，从而确保可判定性。对于后一种方法，他们证明，任何受有限大小缓冲区约束的类型上下文都是可判定的（因为该类型具有有限状态转换系统表示）。与他们的结果一致，我们将其有界性的定义（即对缓冲区大小的限制）推广到 MAGπ 的类型系统中。

定义 15（有界性，摘自 [32]）。我们称 Γ 是（Σ；$\mathcal{R}$）-有界的$_k$ 当且仅当$\exists k \in \mathbb{N}$：$\Gamma \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma'$，$s[\mathrm{p}] : \mathbb{M}$ 蕴含 $|\mathbb{M}| < k$。我们称 Γ 是（Σ；$\mathcal{R}$）-有界的，当且仅当 $\exists k$ 有限：(Σ; $\mathcal{R}$)-有界$_k$(Γ)。

根据定义 15，我们在定理5中给出了可判定的有界性质的结果。

定理 5（可判定的有界性质）。（Σ; $\mathcal{R}$）-有界$_k$(Γ) 对所有 Σ、$\mathcal{R}$ 和 k 都是可判定的。此外，若 (Σ; $\mathcal{R}$)-有界(Γ)，则 $\forall \phi \in \{\mathcal{R}_F$-通信安全、无死锁、终止、永不终止、具有活性$\}$，那么 ($\Sigma$; $\mathcal{R}$)-ϕ(Γ) 是可判定的。

因此，所有可通过标准异步全局类型理论表达的协议，以及所有使用有限消息缓冲区的协议，其可判定性均得到保证——如今还可以推理并处理网络错误！

示例 5（Ping Pong：性质）。检查示例 1 和示例 4 中的类型，我们可以得出结论：$\Gamma = \{s[\mathrm{p}] : \mathbb{S}_{\mathrm{p}}, s[\mathrm{q}] : \mathbb{S}_{\mathrm{q}}, s[\mathrm{r}] : \mathbb{S}_{\mathrm{r}}\}$ 是有界的$_4$。根据定理 5，Γ 可以判定是否满足类型级性质。经检查，我们确定 Γ 具有以下性质：(i) 安全，满足主体归约所要求的安全性（定义 10）；(ii) $\mathcal{R}_F$-通信安全，因为若只考虑可靠归约，则没有缓冲区类型仍有内容；(iii) 终止，因为我们可以计算到达协议结束所需的步骤数；以及 (iv) 具有活性，因为可靠通信 $\mathbb{S}_{\mathrm{r}}$ 总是会归约——也就是说，总能得到结果。

5 推广网络假设

迄今为止介绍的工作涵盖了通信中最坏情况下的网络假设。这对底层网络编程以及在最少假设下运行的复杂分布式应用（例如共识协议）可能大有裨益，但并非所有应用都建立在这些悲观条件之上。例如，许多分布式应用通过传输控制协议（TCP）运行，因此假设：如果连续收到来自同一来源的消息，就可以保证它们按照发送顺序到达。

下面展示对 MAGπ 所做的少量修改，这些修改是改变其网络假设所必需的。需要特别注意的是，这些修改产生了 MAGπ的一个子集，因此其符合 TCP 的版本的所有相关性质仍然有效。

5.1 从完全重排序到部分重排序

在设计为通过 TCP 运行的可靠网络配置中，可以保证双方通信不会发生消息重排序。因此，我们可以调整 MAGπ 中的消息重排序，以建模这种环境，并将我们的安全性 φ_{s} 强化为 TCP安全通信。MAGπ 通过缓冲区同余规则对消息重排序进行建模。因此，只需强化同余规则，即可将通信限制在 TCP安全通信的假设范围内。

定义 16（TCP进程同余）。MAGπ 的符合TCP的子集 $\equiv_{TCP}$上的进程同余采用与定义 $\equiv$ （定义3）相同的规则归纳定义，但作如下修改：

$$\frac{\begin{array}{c} s : \sigma_1 \cdot h_1 \cdot h_2 \cdot \sigma_2 \;\equiv\; s : \sigma_1 \cdot h_2 \cdot h_1 \cdot \sigma_2 \\ \textit{replaced by} \\ \mathbf{p}_1 \neq \mathbf{p}_2 \;\textit{or}\; \mathbf{q}_1 \neq \mathbf{q}_2 \end{array}}{\begin{array}{c} s : \sigma_1 \cdot (\mathbf{p}_1 \triangleright \mathbf{q}_1 \triangleleft \mathsf{m}_1\langle w_1\rangle) \cdot (\mathbf{p}_2 \triangleright \mathbf{q}_2 \triangleleft \mathsf{m}_2\langle w_2\rangle) \cdot \sigma_2 \\ \equiv_{\mathsf{TCP}}\; s : \sigma_1 \cdot (\mathbf{p}_2 \triangleright \mathbf{q}_2 \triangleleft \mathsf{m}_2\langle w_2\rangle) \cdot (\mathbf{p}_1 \triangleright \mathbf{q}_1 \triangleleft \mathsf{m}_1\langle w_1\rangle) \cdot \sigma_2 \end{array}}$$

为了得到πMAG 的符合TCP的子集，我们假设归约发生在完全可靠网络上，并采用定义16中的 TCP进程同余；该定义不再允许各角色对之间的消息重排序。现在，我们在定义17中将此 TCP同余定义反映到类型级，并用它在定义18中定义类型上下文上的 TCP安全性属性。

定义 17（TCP类型同余）。MAGπ 的符合 TCP 的子集 $\equiv_{TCP}$上的类型同余采用与$\equiv$ （图4）相同的规则归纳定义，但作如下修改：

$$\frac{}{\mathbb{M}_1 \cdot \mathbb{M}_2 \equiv \mathbb{M}_2 \cdot \mathbb{M}_1} \quad \textit{replaced by} \quad \frac{\mathbf{p} \neq \mathbf{q}}{\begin{array}{c} \mathbf{p}\,!\,\mathsf{m}_1(\mathbb{T}_1) \cdot \mathbf{q}\,!\,\mathsf{m}_2(\mathbb{T}_2) \cdot \mathbb{M} \\ \equiv_{\mathsf{TCP}}\; \mathbf{q}\,!\,\mathsf{m}_2(\mathbb{T}_2) \cdot \mathbf{p}\,!\,\mathsf{m}_1(\mathbb{T}_1) \cdot \mathbb{M} \end{array}}$$

定义 18（TCP安全性）。谓词 φ_{TCP} 是类型上下文上的 Σ-TCP安全性属性，当且仅当：

$$\begin{array}{lcl} \varphi_{\mathsf{TCP}}(\Gamma,\ s[\mathbf{p}] : \&_{i\in I}\{\mathbf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\},\ s[\mathbf{q}] : \mathbb{M}) & & \\ \quad \text{and } \mathbb{M} \equiv_{\mathsf{TCP}} \mathbf{p}\,!\,\mathsf{m}(\mathbb{T}) \cdot \mathbb{M}' & & \\ \quad \text{and } \exists\, k \in I : \mathbf{q}_k = \mathbf{q} & \Longrightarrow & \mathsf{m}_k = \mathsf{m} \ \wedge\ \mathbb{T}_k = \mathbb{T} \\ \varphi_{\mathsf{TCP}}(\Gamma,\ s[\mathbf{p}] : \mu \mathrm{t}.\mathbb{S}) & \Longrightarrow & \varphi_{\mathsf{TCP}}(\Gamma,\ s[\mathbf{p}] : \mathbb{S}[{}^{\mu \mathrm{t}.\mathbb{S}}/_{\mathrm{t}}]) \\ \varphi_{\mathsf{TCP}}(\Gamma) \text{ and } \Gamma \to_{\Sigma} \Gamma' & \Longrightarrow & \varphi_{\mathsf{TCP}}(\Gamma') \end{array}$$

与我们此前在定义10中对安全性的定义类似，TCP安全性确保通信实体的负载类型相匹配。此外，它还要求消息顺序正确（允许 $\equiv_{\mathsf{TCP}}$），这通过检查消息标签来实现——这是可行的，因为双方之间的消息不会被重新排序，因此接收顺序必定与发送顺序相同。为了利用§ 4中证明的会话定理，只需证明 $\varphi_{\mathsf{TCP}} \subseteq \varphi_{\mathsf{s}}$，即任何 TCP 安全的上下文也都是安全的。这是唯一的要求，因为§ 4 (*i*)中的所有定理都以可靠性函数 $\mathcal{R}$为参数，包括完全可靠网络；且 (*ii*)这些定理均针对 $(\Sigma; \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$ 得到证明。

命题 1（ φ_{TCP} 在 φ_{s}中的包含关系）。 $\forall \Gamma \in \varphi_{\text{TCP}}$： $\Gamma \in \varphi_{\text{s}}$。

证明。 φ_{TCP} 使用 MAG 的完全可靠配置π—即不存在用于处理故障的超时，因此显然满足[S-$\mathcal{R}_1$] 和 [S-$\mathcal{R}_2$]。 [S-μ] 直接反映在 φ_{TCP}中。 [S-$\rightarrow$] 则针对 $\mathcal{R} = \mathcal{R}_F$得到体现，即针对完全可靠配置。 [S-Com] 从未被 $\Gamma \in \varphi_{\text{TCP}}$ 违反，因为 $\equiv_{\text{TCP}} \subset \equiv$。 □

6 案例研究

本文介绍 Ping（示例 1–5）和域名系统（§ 6.1）示例，因为它们广为人知，而且两者涵盖了我们贡献的全部范围。以往的相关工作无法用我们涵盖的故障假设来充分表达任一协议。因此，Ping 和 DNS 适合用于说明 MAGπ 如何拓展多方会话类型的边界。补充示例见附录 D。

6.1 DNS

现在，我们通过一个案例研究展示 MAGπ 的关键特性。我们给出一个带缓存和内置负载均衡器的域名系统（DNS）多方示例。该示例：（i）针对其不可靠连接中的故障进行推理，这些连接使用我们新提出的特定视角的可靠性集合来描述；（ii）为这些可能发生的故障定义故障处理协议；（iii）是有界的（定义 15），因此其类型级性质是可判定的；并且（iv）是安全的，$\mathcal{R}_F$ 通信安全、无死锁且可终止，并且具有活性。通常，DNS 基于 TCP 实现；不过，分布式组件仍可能遭遇硬件故障。为应对这一点，并更好地展示我们的贡献，我们在易发生故障的环境中描述该协议。

规范 我们考虑一种客户端与 DNS 的交互规范：客户端查询缓存，而 DNS 将请求委托给工作节点。

客户端以角色 **c** 表示，希望获取某个特定 URL 对应的网址，并可向 DNS 发送 请求来实现这一目的。作为一种优化，客户端还会将最近获取的网址存储在本地可靠的 缓存中——因此，在向 DNS 发送新请求之前，它会先查询此缓存。收到请求后，**DNS** 会将处理工作委托给两个工作节点之一，分别以角色 $\mathbf{w}_1$ 和 $\mathbf{w}_2$表示。获取相应网址后，工作节点会向客户端发送响应。

此应用的可靠性配置如下：客户端与缓存之间有可靠连接，形式化表示为 $\mathcal{R}$(c) = {cache} 以及 $\mathcal{R}$(cache) = {c}；DNS 与工作节点之间有可靠连接，形式化表示为 $\mathcal{R}$(DNS) = {$\mathbf{w}_1$,$\mathbf{w}_2$}以及 $\mathcal{R}(\mathbf{w}_1) = \mathcal{R}(\mathbf{w}_2)$ = {DNS}；所有其他通信均不可靠。

现在，我们给出用于指定该分布式应用通信协议的会话类型。与 Ping 示例一样，为简洁起见，我们采用单例选择的简写形式，并省略负载类型。

示例 6（DNS 协议）。

$$\mathbb{S}_{\mathsf{c}} = \mathsf{cache}\,!\,\mathsf{req}().\,\&\left\{\begin{array}{l}\mathsf{cache}\,?\,\mathsf{ans}().\mathbf{end},\\ \mathsf{cache}\,?\,404().\mathsf{DNS}\,!\,\mathsf{req}().\,\&\left\{\begin{array}{l}\mathsf{w}_1\,?\,\mathsf{ans}().\mathsf{cache}\,!\,\mathsf{new}().\mathbf{end},\\ \mathsf{w}_2\,?\,\mathsf{ans}().\mathsf{cache}\,!\,\mathsf{new}().\mathbf{end},\\ \underset{1}{}.\,\mathsf{cache}\,!\,\mathsf{ko}().\mathbf{end}\end{array}\right.\end{array}\right.$$

$$\mathbb{S}_{\mathsf{cache}} = \&\left\{\begin{array}{l}\mathsf{c}\,?\,\mathsf{req}().\,\oplus\left\{\begin{array}{l}\mathsf{c}\,!\,\mathsf{ans}().\mathbf{end},\\ \mathsf{c}\,!\,404().\,\&\left\{\begin{array}{l}\mathsf{c}\,?\,\mathsf{new}().\mathbf{end},\\ \mathsf{c}\,?\,\mathsf{ko}().\mathbf{end}\end{array}\right.\end{array}\right.\end{array}\right.$$

$$\mathbb{S}_{\mathsf{DNS}} = \&\left\{\begin{array}{l}\mathsf{c}\,?\,\mathsf{req}().\,\oplus\left\{\begin{array}{l}\mathsf{w}_1\,!\,\mathsf{req}().\mathsf{w}_2\,!\,\mathsf{ko}().\mathbf{end}\\ \mathsf{w}_2\,!\,\mathsf{req}().\mathsf{w}_1\,!\,\mathsf{ko}().\mathbf{end}\end{array}\right.\\ \underset{1}{}.\,\mathsf{w}_1\,!\,\mathsf{ko}().\mathsf{w}_2\,!\,\mathsf{ko}().\mathbf{end}\end{array}\right.$$

$$\mathbb{S}_{\mathsf{w}_i} = \&\left\{\begin{array}{l}\mathsf{DNS}\,?\,\mathsf{req}().\mathsf{c}\,!\,\mathsf{ans}().\mathbf{end},\\ \mathsf{DNS}\,?\,\mathsf{ko}().\mathbf{end}\end{array}\right.$$

我们针对特定视角定义的可靠性，是在与 DNS 和工作节点保持可靠连接的同时，指定与客户端保持不可靠连接所必需的。此外，客户端类型 $\mathbb{S}_{\mathsf{c}}$ (相应地，DNS 类型$\mathbb{S}_{\mathsf{DNS}}$) 依赖于使用无向分支（相应地，选择）。因此，先前的理论无法表达此示例 [4,32]。

7 相关工作、结论与未来工作

近年来，对故障建模已成为一个备受关注且研究广泛的主题。我们详细阐述了我们的通用类型系统和模块化语言与相关工作的差异，以及在某些情况下它们可能涵盖相关工作的部分内容。

Majumdar 等人 [23] 引入无向分支，以应对网络中可能出现的消息非确定性部分重排序；这种情况可能发生在使用传输控制协议（TCP）的网络中。如 § 5所示，我们的类型系统具有模块化特性，因此可以对 MAGπ 进行调整，使其支持这种网络配置，以及抽象层次较低的其他场景。

仿射类型系统定义的类型至多可以使用一次。仿射会话类型 [24,12,6] 利用仿射类型系统元理论，允许会话在发生故障时提前取消。这些工作只对应用层故障进行建模（使用 try/catch 块），不一定描述如何处理故障，而只允许在故障发生时放弃初始协议。

Viering 等人 [37] 提出了一种适用于事件驱动分布式系统的多方会话类型理论，其中进程崩溃时，监控器会重新启动进程。这种方法需要 一个集中式可靠节点，而这一概念已被我们视点特定的可靠性定义（定义 7）所涵盖。

Chen 等人 [8] 不再需要集中式可靠节点。他们为类型系统加入了同步点，能够检测并处理节点所遭遇的故障。类似地，Adameit 等人 [1]考虑了一种不含集中式可靠节点的环境，其中参与方之间的不稳定链路可能发生故障。他们引入了可选块的概念，

允许默认值 替代因通信故障而未能收到的数据。Viering 等人 [36], 受共识算法启发，将一组进程指定为一个永久可用的恢复系统，负责监控进程并通知它们发生的故障。因此，他们不再依赖于一个集中式强健节点，而是假设组成协调器的进程中，在任何时刻至少有一些处于活性状态。这些方法的缺点是依赖协调来处理故障。在某些网络配置和故障模型下，这可能并不适用。由于我们的类型系统通过底层技术处理故障，因此不受故障类型的限制，适用于任何非拜占庭网络配置。

Peters 等人近期的工作 [27] 通过引入故障标注扩展了全局类型理论——标记易受故障影响的通信以及故障类型（具体而言，即进程崩溃或消息丢失）。他们通过定义默认值和分支来处理故障。由于该理论是对全局类型的扩展，因此也存在广义会话类型所要解决的相同问题。此外，该工作并不独立于故障模型，因此尚不确定该理论是否足够通用，能够对他们所考虑的两种故障之外的其他故障进行建模。

与 MAGπ 最为相似的是 Barwell 等人 [4], 的工作，他们扩展了广义会话类型理论，以研究崩溃停止故障。他们保留了崩溃消息标签，可在接收分支中用于检测节点故障并指定故障处理子协议。与我们的研究一致，他们的类型系统具有通用性，因此提升了其表达能力。然而，与 MAGπ不同，他们的理论不是异步的，不支持无向分支/选择，并且假设崩溃停止故障是唯一可能的故障——而我们则处理并涵盖多种故障，例如崩溃故障、链路故障、消息丢失、延迟、重排序和网络分区。

π-演算的分布式变体 [2,29,7,13] 引入了进程位置这一概念，即对现实世界中物理硬件的表示。进程被分配到各个位置，以形成进程拓扑；这些位置可以发生崩溃，以模拟故障。然而，这些演算都无法模拟 MAGπ 所支持的各种故障，也没有能够确保以通信安全的方式处理故障的类型系统。

总之，本文介绍了 MAGπ——一种多方、异步且广义的 π-演算，它通过使用超时和最一般的可靠性定义，处理范围最广的非拜占庭故障。我们的语言建立在广义异步多方会话类型之上，后者最适用于分布式编程。我们证明了主体归约和会话保真性、一系列进程性质，以及故障处理安全性和可靠性遵循性。未来，我们计划研究用于 Curry-Howard 对应的线性逻辑，以理解故障与可靠性的基础性和规范性含义。我们还计划研究拜占庭故障与本文所处理的非拜占庭故障相结合的情形。最后，我们将探索使用模型检测来简化进程性质的验证。

参考文献

1. Adameit, M., Peters, K., Nestmann, U.: 链路故障的会话类型。载于：Bouajjani, A., Silva, A.（编）《分布式对象、组件与系统形式化技术——第37届 IFIP WG 6.1 国际会议 FORTE 2017》，作为第12届分布式计算技术国际联合会议 DisCoTec 2017 的一部分举行，瑞士纳沙泰尔，2017年6月19日至22日，论文集。计算机科学讲义，第10321卷，第1–16页。施普林格（2017）。https://doi.org/10.1007/978-3-319-60225-7_12. Amadio, R.M.: 局部性、故障与进程迁移的异步模型。载于：Garlan, D., M´etayer, D.L.（编）《协调语言与模型》，第二届国际会议 COORDINATION '97，德国柏林，1997年9月1日至3日，论文集。计算机科学讲义，第1282卷，第374–391页。施普林格（1997 10 1007 3 540 63383 9_92）。https://doi.org/ . / - - - . Bartoletti, M., Scalas, A., Tuosto, E., Zunino, R.: 通过类型实现诚实性。逻辑方法与计算机科学 12(4)（2016）。https://doi.org/10.2168/LMCS-12(4:7)2016 4. Barwell, A.D., Scalas, A., Yoshida, N., Zhou, F.: 具有崩溃停止故障的广义多方会话类型。载于：Klin, B., Lasota, S., Muscholl, A.（编）第33届国际并发理论会议 CONCUR 2022，2022年9月12日至16日，波兰华沙。LIPIcs，第243卷，第35:1–35:25页。Schloss Dagstuhl - Leibniz 信息学中心（2022）。https://doi.org/10.4230/LIPIcs.CONCUR.2022.35 5. Busi, N., Gabbrielli, M., Zavattaro, G.: 进程演算中递归、复制与迭代的表达能力。数学结构与计算机科学 19(6)，1191–1222（2009）。https://doi.org/10.1017/S096012950999017X 6. Capecchi, S., Giachino, E., Yoshida, N.: 多方会话中的全局逃逸。数学结构与计算机科学 26(2)，156–205（2016）。https://doi.org/10.1017/S0960129514000164 7. Castellani, I.: 带局部性的进程代数。载于：Bergstra, J.A., Ponse, A., Smolka, S.A.（编）《进程代数手册》，第945–1045页。北荷兰出版社 / Elsevier（2001）。https://doi.org/10.1016/b978-044482830-9/50033-3 8. Chen, T., Viering, M., Bejleri, A., Ziarek, L., Eugster, P.: 分布式系统中健壮故障处理的类型理论。载于：Albert, E., Lanese, I.（编）《分布式对象、组件与系统形式化技术——第36届 IFIP WG 6.1 国际会议 FORTE 2016》，作为第11届分布式计算技术国际联合会议 DisCoTec 2016 的一部分举行，希腊克里特岛伊拉克利翁，2016年6月6日至9日，论文集。计算机科学讲义，第9688卷，第96–113页。施普林格（2016）。https://doi.org/10.1007/978-3-319-39570-8_79. Clarke, E.M., Grumberg, O., Peled, D.A.: 模型检测。麻省理工学院出版社（2001）10. Clarke, E.M., Klieber, W., Nov´acek, M., Zuliani, P.: 模型检测与状态爆炸问题。载于：《LASER暑期学校》。计算机科学讲义，第7682卷，第1–30页。施普林格（2011）。https://doi.org/10.1007/978-3-642-35746-6_111. Dezani-Ciancaglini, M., Mostrous, D., Yoshida, N., Drossopoulou, S.: 面向对象语言的会话类型。载于：ECOOP 2006。第4067卷，第328–352页。施普林格·柏林·海德堡（2006）。https://doi.org/10.1007/11785477_20, http://link.springer.com/10.1007/11785477_20, 计算机科学讲义

12. Fowler, S., Lindley, S., Morris, J.G., Decova, S.：异常异步会话类型：无层级的会话类型。ACM《程序设计语言》会议录 3(POPL), 28:1–28 3290341 13 29 2019 10 1145 : ( )。https://doi.org/ . / . Hennessy, M.：分布式π演算。剑桥大学出版社（2007）14. Honda, K.：二元交互的类型。载于：Best, E.（编）CONCUR '93，第4届国际并发理论会议，德国希尔德斯海姆，1993年8月23日至26日，论文集。计算机科学讲义，第715卷，第509– 523 1993 10 1007 3 540 57208 2_35 页。施普林格（）。https://doi.org/ . / - - - . Honda, K., Vasconcelos, V.T., Kubo, M.：基于结构化通信的编程的语言原语与类型纪律。载于：ESOP。LNCS，第1381卷，第122 0053567 16 138页。施普林格（）。https://doi.org/ . /BFb . Honda, K., Yoshida, N., Carbone, M.：多方异步会话类型。《ACM学报》63 2827695 17 1 9 1 9 67 2016 10 1145 ( ), : – : ( )。https://doi.org/ . / . Kokke, W., Dardha, O.：线性 Haskell 中的无死锁会话类型。载于：Hage, J.（编）Haskell 2021：第14届 ACM SIGPLAN Haskell 国际研讨会论文集，线上活动，韩国，2021年8月26日至27日。第1–13页。ACM（2021）。https://doi.org/10 3472979 18 1145 3471874 . / . . Kouzapas, D., Dardha, O., Perera, R., Gay, S.J.：使用 Mungo 和 StMungo 进行协议类型检查。载于：Cheney, J., Vidal, G.（编）第18届声明式编程原理与实践国际研讨会论文集，英国爱丁堡，2016年9月5日至7日。第146–159页。ACM（2016）。https://doi.org/10 2968595 19 1145 2967973 . / . . Lagaillardie, N., Neykova, R., Yoshida, N.：恐慌时保持安全：使用多方会话类型进行仿射 Rust 编程。载于：第36届欧洲面向对象编程会议，ECOOP 2022，2022年6月6日至10日，德国柏林。LIPIcs，第222卷，第4:1–4:29页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik（2022）。https://doi.org/10 4 20 4230 2022 . / LIPIcs.ECOOP. . . Lamport, L.：兼职议会。《ACM计算机系统汇刊》16(2), 133–169（1998 279229 21 10 1145 279227）。https://doi.org/ . / . . Lange, J., Ng, N., Toninho, B., Yoshida, N.：隔离 Go：基于通道的编程的活性与安全性。载于：第44届 ACM SIGPLAN 编程语言原理研讨会论文集，POPL 2017，法国巴黎，2017年1月18日至20日。第748–761页。ACM（2017）。https://doi.org/10 3009847 22 1145 3009837 . / . . Laprie, J.C.：可信计算与容错。FTCS-15 论文摘要集 10(2), 124 （ 1985 ） 23. Majumdar, R., Mukund, M., Stutz, F., Zufferey, D.：异步多方会话类型中的投影泛化。载于：Haddad, S., Varacca, D.（编）第32届国际并发理论会议，CON-CUR 2021，2021年8月24日至27日，线上会议。LIPIcs，第203卷，第35:1–35:24页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik（2021）。https://doi.org/10 35 24 4230 2021 . /LIPIcs.CONCUR. . . Mostrous, D., Vasconcelos, V.T.：仿射会话。《逻辑方法与计算机科学》14(4)（2018 2018 25 10 23638 14 4 14）。https://doi.org/ . /L MCS- (:) . Ongaro, D., Ousterhout, J.K.：寻找一种易于理解的共识算法。载于：USENIX 年度技术会议。第305–319页。USENIX 协会（2014），https://www.usenix.org/conference/atc14/technical-sessions/presentation

/Ongaro

26. Orchard, D., Yoshida, N.：Haskell中具有线性性的会话类型。行为类型：从理论到工具，第219页（2017）

27. Peters, K., Nestmann, U., Wagner, C.：容错多方会话类型。载于：Mousavi, M.R., Philippou, A.（编）分布式对象、组件与系统形式化技术——第42届 IFIP WG 6.1 国际会议 FORTE 2022，作为第17届分布式计算技术国际联合会议 DisCoTec 2022 的一部分举办，意大利卢卡，2022年6月13日至17日，会议录。计算机科学讲义，第13273卷，第93–113页。施普林格（2022）。https://doi.org/ 10 1007 978 3 031 08679 3_7 。 / - - - - . Pierce, B.C.：类型与
28
程序设计语言。麻省理工学院出版社（2002）29. Riely, J., Hennessy, M.：分布式进程与位置故障。《理论计算机科学》266(1-2), 693–735（2001）。https://doi.org/10 1 30 1016 0304 3975 00 00326 . /S - ( ) - . Rossi, M.：通信系统的建模与分析。《计算机形式化方面》33 8 31 2 297 298 2021 10 1007 00165 021 00533 ( ), – ( )。https://doi.org/ . /s - - - . Scalas, A., Dardha, O., Hu, R., Yoshida, N.：面向安全分布式编程的多方会话线性分解。载于：M¨uller, P.（编）第31届欧洲面向对象编程会议 ECOOP 2017，2017年6月19日至23日，西班牙巴塞罗那。LIPIcs，第74卷，第24:1–24:31页。Schloss Dagstuhl - Leibniz-Zentrum f¨ur Informatik（2017）。https://doi.org/10 24 32 4230 2017 . /LIPIcs.ECOOP. . . Scalas, A., Yoshida, N.：少即是多：多方会话类型再探。ACM《程序设计语言》会议录 3 3290343 33 30 1 30 29 2019 10 1145 (POPL), : – : ( )。https://doi.org/ . / . Tabone, G., Francalanza, A.：Elixir中的会话类型。载于：Castegren, E., Koster, J.D., Fowler, S.（编）AGERE 2021：第11届 ACM SIGPLAN 基于参与者、代理和分散式控制的编程国际研讨会会议录，线上活动 / 美国伊利诺伊州芝加哥，2021年10月17日。第12–23页。ACM（2021 3486708 34 10 1145 3486601）。https://doi.org/ . / . . Takeuchi, K., Honda, K., Kubo, M.：一种基于交互的语言及其类型系统。载于：PARLE '94。LNCS，第817卷，第398–413页。施普林格（1994）35. Vasconcelos, V.T.：会话类型基础。《信息与计算》217, 52–70（2012）。https://doi.org/10 002 36 1016 2012 05 . /j.ic. . . . Viering, M., Chen, T., Eugster, P., Hu, R., Ziarek, L.：一种用于分布式系统中静态验证崩溃故障处理的类型纪律。载于：Ahmed, A.（编）程序设计语言与系统——第27届欧洲程序设计研讨会 ESOP 2018，作为欧洲软件理论与实践联合会议 ETAPS 2018 的一部分举办，希腊塞萨洛尼基，2018年4月14日至20日，会议录。计算机科学讲义，第10801卷，第799–826页。施普林格（ 2018 10 1007 978 3 319 89884 1_28 ）。
37
https://doi.org/ . / - - - - . Viering, M., Hu, R., Eugster, P., Ziarek, L.：面向容错事件驱动分布式编程的多方会话类型纪律。ACM《程序设计语言》会议录 5(OOPSLA), 1–30（2021）。https://doi.org/10.1145/3485501

主体归约

引理 1. $(\Sigma;\ \mathcal{R})$-$\varphi_s(\Gamma,\ \Gamma') \implies (\Sigma;\ \mathcal{R})$-$\varphi_s(\Gamma)$

证明。反证法。
假设 Γ 并非 $(\Sigma;\ \mathcal{R})$-安全。
那么，$\Gamma \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma''$ st: Γ'' 违反$(\Sigma;\ \mathcal{R})$-φ_s (通过违反以下任一项: [S-$\mathcal{R}_1$]、 [S-$\mathcal{R}_2$] 或 [S-Com])。
但是，根据 [Γ-合同行为], $\Gamma,\ \Gamma' \longrightarrow^*_{(\Sigma;\mathcal{R})} \Gamma'',\ \Gamma'$; 其中 Γ'', Γ'不是（$\Sigma;\ \mathcal{R}$）-安全的（因为 Γ''违反了这一条件）。
∴ $\Gamma,\ \Gamma'$违反了 [S-$\to$], 因此 $\Gamma,\ \Gamma'$是 不$(\Sigma;\ \mathcal{R})$-安全；矛盾。
因此，$(\Sigma;\ \mathcal{R})$-$\varphi_s(\Gamma)$。 □

引理 2. $(\Sigma_1;\mathcal{R})$-$\varphi_s(\Gamma_1)$ 和 $(\Sigma_2;\mathcal{R})$-$\varphi_s(\Gamma_2)$, 以及 $\Sigma_1 \cap \Sigma_2 = \emptyset \implies (\Sigma_1 \cup \Sigma_2;\mathcal{R})$-$\varphi_s(\Gamma_1, \Gamma_2)$

证明。在 Γ_1 中的通信包含在 Σ_1 中的会话之下。
同样，在 Γ_2 中的通信包含在 Σ_2 中的会话之下。
∴ Γ_1, Γ_2 不会引入任何可能违反 [S-Com]，的新通信，因为它们的会话互不相交（$\Sigma_1 \cap \Sigma_2 = \emptyset$）。

此外，上下文组合不会改变分支的结构，因此 Γ_1, Γ_2 不会违反[S-$\mathcal{R}_1$] 或 [S-$\mathcal{R}_2$]。

[S-μ] 和 [S-$\to$] 均可通过 [Γ-μ] 和 [Γ-合同行为] 得以保持。
因此，$(\Sigma_1 \cup \Sigma_2;\mathcal{R})$-$\varphi_s(\Gamma_1, \Gamma_2)$。 □

引理 3（消息垃圾回收）。若 $\Theta \cdot \Gamma \vdash_\Sigma s :\ h \cdot \sigma$，则 $\Theta \cdot \Gamma \vdash_\Sigma s :\ \sigma$，其中 h是某个（$\mathsf{p} \triangleright \mathsf{q} \triangleleft \mathsf{m}\langle w \rangle$）。

证明。对 σ 的结构进行归纳证明。

（BC）基本情况：设 $\sigma = \epsilon$

(1) $\Theta \cdot \Gamma \vdash_\Sigma s :\ h \cdot \epsilon$ （根据蕴含假设）
(2) $\Gamma = \Gamma_0,\ \Gamma',\ s[\mathsf{p}] :\ \mathsf{q}!\ \mathsf{m}(\mathbb{T}) \cdot \mathbb{M}$ （由1）以及对 [T-σ_1]/[T-σ_2]取逆）

现在，我们对（1）的推导进行归纳证明。
(SC1) 子情况 [T-σ_1]

(3) $\Theta \cdot \Gamma' \vdash_\Sigma s :\ \epsilon$ 且 $\Gamma_0 \vdash w :\ \mathbb{T}$ 且 $\mathbb{M} = \epsilon$ （由逆 of [T-σ_1])
(4) $\Gamma_0 = w :\ \mathbb{T}$ （由（3), [T-变量])
(5) $\mathsf{gc}(\Gamma')$ （依据 (3), [T-ϵ])
(6) 根据（2）和上下文组合，我们知道 $s[\mathsf{p}] \notin \Gamma'$。因此：$\mathsf{gc}(\Gamma', s[\mathsf{p}] : \epsilon)$（根据（5）
(7) $\mathsf{gc}(\Gamma)$（根据（2）、（4）、（6）、 以及图8）
(8) $\Theta \cdot \Gamma \vdash_\Sigma s :\ \epsilon$ （根据（7） 以及图8)

∴ 由（8），引理 3 适用于 **(BC)**：**(SC1)**。

(SC2) 子情况 $[\text{T-}\sigma_2]$

$\Theta \cdot \Gamma', s[\mathsf{p}] : \mathbb{M} \vdash_\Sigma s : \epsilon$ 和 $\Gamma_0 \vdash w : \mathbb{T}$ (根据 $[\text{T-}\sigma_1]$ 的逆) (9)$\Gamma_0 = w : \mathbb{T}$ (根据 (9) 和 [T-Var) (10) gc($\Gamma', s[\mathsf{p}] : \mathbb{M}$) (根据 (9) 和 $[\text{T-}\epsilon]$) (11) gc(Γ) (根据 (2)、(10)、(11)、图 8) (12) $\Theta \cdot \Gamma \vdash_\Sigma s : \epsilon$ (根据 (12) 和 $[\text{T-}\epsilon]$) (13)

∴ 根据 (13)，引理 3 适用于 **(BC)**：**(SC2)**。

因此，由于引理 3 适用于基本情况的所有子情况，因此它适用于 **(BC)**。

(IC) 归纳情形：设 $\sigma = h' \cdot \sigma'$

(14) $\Theta \cdot \Gamma \vdash_\Sigma s : (\mathsf{p} \triangleright \mathsf{q} \triangleleft \mathsf{m}\langle w \rangle) \cdot h' \cdot \sigma'$ (蕴含假设)

现在，我们对 (14)**(SC1)** 子情况 $[\text{T-}\sigma_1]$ 的推导进行归纳证明

(15) $\Gamma = \Gamma_0, \Gamma_1, \Gamma_2,$ st: (由（14）及 $[\text{T-}\sigma_1]$ 的逆得出)

(15a) $\Theta \cdot \Gamma_1 \vdash_\Sigma s : h' \cdot \sigma'$

$\Gamma_2 \vdash w : \mathbb{T}$ (15b)

$\Gamma_0 = s[\mathsf{p}] : \mathsf{q}\,!\mathsf{m}(\mathbb{T}) \cdot \epsilon$ (15c)

(16) $\Gamma = (\Gamma_0 \rightsquigarrow \Gamma_1)\ \Gamma_2$ (由 (15)、(15a)、(15c)，以及图 $\rightsquigarrow$ 9 中

(17) $\mathsf{gc}(\Gamma_0, \Gamma_2)$ (由 (15b)、(15c)，以及图 gc 8 中

(18) $\Theta \cdot \Gamma \vdash_\Sigma s : h' \cdot \sigma'$ (由 (16)、(15a)、(17)，以及 $[\text{T-}\sigma_w]$)

∴ 根据（17），引理 3 对 （IC）： （SC1）成立。

（SC2）子情况 $[\text{T-}\sigma_2]$

(19) $\Gamma = \Gamma_0、\ \Gamma_1、\ \Gamma_2、$ st: （根据（14）以及 $[\text{T-}\sigma_2]$ 的逆）

$\Theta \cdot \Gamma_1,\ s[\mathsf{p}] : \mathbb{M} \vdash_\Sigma s : h' \cdot \sigma'$ （19a）

$\Gamma_2 \vdash w : \mathbb{T}$ （19b）

$\Gamma_0 = s[\mathsf{p}] : \mathsf{q}\,!\ \mathsf{m}(\mathbb{T}) \cdot \mathbb{M}$ (19c)

$\Gamma = (s[\mathsf{p}] : \mathsf{q}\,!\ \mathsf{m}(\mathbb{T}) \cdot \epsilon \rightsquigarrow \Gamma_1, s[\mathsf{p}] : \mathbb{M})\ \Gamma_2$ (根据 (19)、(19a)、(19c) 以及图9中的定义) (20) gc($s[\mathsf{p}] : \mathsf{q}\,!\ \mathsf{m}(\mathbb{T}) \cdot \epsilon, \Gamma_2$) (根据 (19b) 以及图 8 中 gc 的定义) (21)$\Theta \cdot \Gamma \vdash_\Sigma s : h' \cdot \sigma'$ (根据 (20)、(19a)、(21) 和 $[\text{T-}\sigma_w]$) (22)

∴ 根据 (22)，引理 3 适用于 **(IC)**：**(SC2)**。

He因此，由于引理 3 对归纳情形的所有子情况均成立，因此它对 **(IC)** . □

定理 1（主体归约）如果一个进程在安全上下文下具有类型，并且该进程发生归约，那么上下文会（经过 0 步或 1 步）归约为一个安全上下文，该上下文为新进程赋予类型。形式化地：

$$\Theta \cdot \Gamma \vdash_\Sigma P \ \wedge\ (\Sigma;\mathcal{R})\text{-}\varphi_{\mathsf{s}}(\Gamma) \ \wedge\ P \to_{\mathcal{R}} P' \implies \exists \Gamma' : \Gamma \to^{\{0,1\}}_{(\Sigma;\mathcal{R})} \Gamma' \ \wedge\ (\Sigma;\mathcal{R})\text{-}\varphi_{\mathsf{s}}(\Gamma') \ \wedge\ \Theta \cdot \Gamma' \vdash_\Sigma P'$$

其中 $\to^{\{0,1\}}$ 指 0 步或 1 步归约。

证明。我们对 $P \to_{\mathcal{R}} P'$ 的推导进行归纳法，以证明主体归约。我们可以假设如下：

(A1) $\Theta \cdot \Gamma \vdash_\Sigma P($
A2) $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$
(A3) $P \to_{\mathcal{R}} P'$

在这些假设下，我们证明定理1 对 $P \to_{\mathcal{R}} P'$的所有情形：

(C1) 情形 [R-⊕]

(23) $P = s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{m}\langle w\rangle.\,Q \ \mid\ s : \sigma$ (由 **(A3)** 以及 [**R-⊕**] 的逆)

(24) $P' = Q \ \mid\ s : (\mathsf{p} \triangleright \mathsf{q} \triangleleft \mathsf{m}\langle w\rangle) \cdot \sigma \cdot \epsilon$ (由 **(A3)** 以及 [**R-⊕**] 的逆)

(25) $\Gamma = \Gamma_\oplus, \Gamma_s$ 和 $\Sigma = \Sigma_1 \cup \Sigma_2$ st: (由 **(A1)** 以及 inv. of [T-|])

(25a) $\Theta \cdot \Gamma_\oplus \vdash_{\Sigma_1} s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{m}\langle w\rangle.\,Q$

$\Theta \cdot \Gamma_s \vdash_{\Sigma_2} s : \sigma$ (25b)

$\Sigma_1 \cap \Sigma_2 = \emptyset$ (25c)

唯一能够为（25a）中的进程定型的规则是 [T-⊕]。因此，我们可以得出结论： :

(26) $\Sigma_1 = \emptyset$ and $\Theta \cdot \Gamma_\oplus \vdash s[\mathsf{p}] \oplus [\mathsf{q}]\,!\,\mathsf{m}\langle w\rangle.\,Q$ （根据（25a）、 [T-⊕] 和 [T-Lift]）

(27) $\Gamma_\oplus = \Gamma_0, \Gamma_1, \Gamma_2$ st: （根据（26）及 [T-⊕] 的逆）

$\Gamma_1 \vdash s[\mathsf{p}] : \ \oplus_{i\in I}\{\mathsf{q}_i\,!\ \mathsf{m}_i\ (\mathbb{T}_i)\ .\mathbb{S}_i\}$ （27a）

$\Gamma_2 \vdash w :$ （27b）

$\Theta \cdot \Gamma_0,\ s[\mathsf{p}] : \ \mathbb{S} \vdash Q$ (27c)

$\exists k \in I : \ \mathsf{q}_k = \mathsf{q},\ \ \mathsf{m}_k = \mathsf{m}\ \ \mathbb{T}_k = \mathbb{T},\ \ \mathbb{S}_k = \mathbb{S}$ （27d）

(28) $\Gamma_1 = s[\mathsf{p}] : \ \oplus_{i\in I}\{\mathsf{q}_i\,!\ \mathsf{m}_i\ (\mathbb{T}_i)\ .\mathbb{S}_i\}$ (by (27a) and [类型值])

(29) $\Sigma_2 = \{s\}$ （对（25b）的推导作归纳法）

(30) $\Sigma\ = \{s\},\ \ \therefore\ s \in \Sigma$ （根据（25）、（26）和（29））

我们考虑两种子情况：当 $s[\mathsf{p}]$ 包含于 Γ_s其中，以及当它不包含于其中时。

(SC1) 子情况：令 $s[\mathbf{p}] \in \Gamma_s$

(31) $\Gamma \xrightarrow{s[\mathbf{p}]\,!\,\mathbf{q}:\,\mathsf{m}\,(\mathbb{T})}_{\Sigma} \Gamma' = \Gamma_0,\ \Gamma_2,\ \Gamma'_1,\ \Gamma'_s$ st:　　(依据（27d）、（30），以及 [Γ-Snd$_2$])

(31a) $\Gamma'_1 = s[\mathbf{p}] : (\mathbb{M} \cdot \mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \epsilon\,;\,\mathbb{S})$

$\Gamma'_s = \Gamma_s \setminus \{s[\mathbf{p}] :\ \mathbb{M}\}$ (31b)

(32) $\Gamma'_1 = \Gamma'_{1M}, \Gamma'_{1s}$ st:　　（依据上下文组合和同余（定义 8））

$\Gamma'_{1M} = s[\mathbf{p}] :\ \mathbf{q}\,!\ \ \mathsf{m}(\mathbb{T}) \cdot \mathbb{M} \cdot \epsilon$ (32a)

(32b) $\Gamma'_{1s} = s[\text{段落}] :\ \mathbb{S}$

(33) $\Theta \cdot \Gamma_0, \Gamma'_{1s} \vdash Q$　　(依据 27c) 和 (32b)

(34) $\Theta \cdot \Gamma_2, \Gamma'_s, \Gamma'_{1M} \vdash_{\Sigma} s : (\mathbf{p} \triangleright \mathbf{q} \triangleleft \mathsf{m}\langle w \rangle) \cdot \sigma$　　（根据（25b）、（27b）、[T-σ_2]）

(35) $\Theta \cdot \Gamma_0, \Gamma'_{1s} \vdash_{\emptyset} Q$　　（根据（33）以及 [T-Lift]）

(36) $\Theta \cdot \Gamma' \vdash_{\Sigma} P'$　　（根据（34）、（35）、因为 $\Sigma \cap \emptyset = \emptyset$，并根据 [T-|]）

(37) $\Gamma \rightarrow_{(\Sigma;\mathcal{R})} \Gamma'$　　（由（31）可直接得出，因为 $\mathcal{R}$ 未被使用）

(38) $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma')$　　(因为 $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$ 且由 (37) 和 [S-$\rightarrow$])

∴ 由 (36)、(37) 和 (38)，定理1 对 **(C1)**成立：**(SC1)**。

(SC2) 子情况：令 $s[\mathbf{p}] \notin \Gamma_s$

(39) $\Gamma \xrightarrow{s[\mathbf{p}]\,!\,\mathbf{q}:\,\mathsf{m}\,(\mathbb{T})}_{\Sigma} \Gamma' = \Gamma_0, \Gamma'_1, \Gamma_2, \Gamma_s$ st:　　（根据（27d）、（30）和 [Γ-Snd$_1$]）

(39a) $\Gamma'_1 = s[\mathbf{p}] :\ (\mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \epsilon\,;\ \mathbb{S})$

(40) $\Gamma'_1 = \Gamma'_{1M}, \Gamma'_{1s}$ st:　　（根据上下文组合和同余（定义 8））

(40a) $\Gamma'_{1M} = s[\mathbf{p}] : \mathbf{q}\,!\,\mathsf{m}(\mathbb{T}) \cdot \epsilon$

$\Gamma'_{1s} = s[\mathbf{p}] :\ \mathbb{S}$ (40b)

(41) $\Theta \cdot \Gamma_0, \Gamma'_{1s} \vdash Q$　　(由 (27c) 和 (40b))

(42) $\Theta \cdot \Gamma_2,\ \Gamma_s,\ \Gamma'_{1M} \vdash_{\Sigma} s :\ \mathbf{p} \triangleright \mathbf{q} \triangleleft \mathsf{m}\langle w \rangle) \cdot \sigma$　　（由（25b）、（27b）、[T-σ_1]）

(43) $\Theta \cdot \Gamma_0, \Gamma'_{1s} \vdash_{\emptyset} Q$　　(根据 (41) 和 [T-Lift])

(44) $\Theta \cdot \Gamma' \vdash_{\Sigma} P'$　　(根据 (42), (43), 因为 $\Sigma \cap \emptyset = \emptyset$, 并根据 [T-|])

(45) $\Gamma \rightarrow_{(\Sigma;\mathcal{R})} \Gamma'$　　(根据 (39)，因为 $\mathcal{R}$ 未被使用，所以显然成立)

(46) $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma')$　　(因为 **(A2)**，并根据 (45) 和 [S-$\rightarrow$])

∴ 根据（44）、（45）和（46），定理1 对**(C1)**成立：**(SC2)**。

因此，由于定理1 对所有子情况均成立，因此它对 case **(C1)**成立。

(C2) case [R- &]

(1) $P = s[\mathbf{q}]\,\&_{i\in I}\{[\mathbf{p}_i]?\ \mathsf{m}_i\ (x_i)\ .Q_i\,[,_{①}.\,Q']\} \mid s:\quad (\mathbf{p} \triangleright \mathbf{q} \triangleleft \mathsf{m}\langle w\rangle)\quad \cdot\sigma$ (由 **(A3)**，逆于 **[R-&]**)

(2) $P' = Q_k[^{w}/_{x_k}] \mid s:\quad \sigma$ 和 $\exists k \in I:\ \mathbf{p}_k = \mathbf{p},\ \mathsf{m}_k = \mathsf{m}$ (由 **(A3)**，逆于 **[R-&]**)

(3) $\Gamma = \Gamma_{\&}, \Gamma_s$ 和 $\Sigma = \Sigma_1 \cup \Sigma_2$ st: (由 **(A1)** 和 [T-|] 的逆)

$\Theta \cdot \Gamma_{\&} \vdash_{\Sigma_1} s[\mathbf{q}]\,\&_{i\in I}\{[\mathbf{p}_i]?\,\mathsf{m}_i(x_i).Q_i\,[,_{①}.\,Q']\}$ (3a)

(3b) $\Theta \cdot \Gamma_s \vdash_{\Sigma_2} s : (\mathbf{p}_k \triangleright \mathbf{q} \triangleleft \mathsf{m}_k\langle w\rangle) \cdot \sigma$

$\Sigma_1 \cap \Sigma_2 = \emptyset$ (3c)

(4) $\Sigma_1 = \emptyset$ 且 $\Theta \cdot \Gamma_{\&} \vdash s[\mathbf{q}]\,\&_{i\in I}\{[\mathbf{p}_i]?\,\mathsf{m}_i(x_i).Q_i\,[,_{①}.\,Q']\}$ (由 (3a)、[T-&], 以及 [T-Lift])

(5) $\Gamma_{\&} = \Gamma_0, \Gamma_1$ st: (由 (4) 和 [T-&] 的逆)

(5a) $\Gamma_1 \vdash s[\mathbf{q}] : \&_{i\in I}\{\mathbf{p}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i\,[,\,\mathbb{S}']\}$

$[\Theta \cdot \Gamma_0, s[\mathbf{q}] :\ \mathbb{S}' \vdash Q']$ (5b)

$\exists i \in I:\ \Theta \cdot \Gamma_0,\ x_i :\ \mathbb{T}_i,\ s[\mathbf{q}] :\ \mathbb{S}_i \vdash Q_i$ (5c)

(6) $\Gamma_1 = s[\mathbf{q}] : \&_{i\in I}\{\mathbf{p}_i?\ \mathsf{m}_i\ (\mathbb{T}_i)\ .\mathbb{S}_i\,[,\,\mathbb{S}']\}$ (依据 (5a) 且 [类型值])

(7) $\Sigma_2 = \{s\}$ (对 (3b) 的推导进行归纳法)

(8) $\Sigma\ = \{s\},\ \therefore\ s \in \Sigma$ (依据 (3), (4) 和 (7))

现在，我们对 (3b) 的推导进行归纳证明，开展一个子证明**(SC1)** 子情况 [T-σ_1]

(9) $\Gamma_s = \Gamma_{s0},\ \Gamma_{s1},\ s[\mathbf{p}_k] :\ \mathbf{q}!\ \ \mathsf{m}_k\ (\mathbb{T})\ \cdot\epsilon$ st: (由 (3b) 和 [T-σ_1] 的逆)

$\Theta \cdot \Gamma_{s1} \vdash_{\Sigma} s : \sigma$ (9a)

$\Gamma_{s0} \vdash w : \mathbb{T}$ (9b)

$\exists k \in I:\ \mathbb{T}_k = \mathbb{T}$ (由于 **(A2)**，并由 (6)、(9)、(2) 和 [S-Com]) (10)

对于 Γ，有两种可能的上下文归约，但我们只需找到一种满足必要条件的归约。考虑:

(11) $\Gamma \xrightarrow{s[\mathbf{p}_k][\mathbf{q}]:\ \mathsf{m}\ k}_{\Sigma} \Gamma' = \Gamma_0, \Gamma_{s0}, \Gamma_{s1}, s[\mathbf{p}_k] :\ \ \epsilon,\ \ s[\mathbf{q}] :\ \ \mathbb{S}$ 且 $\exists k \in I:\ \ \mathbb{S}_k = \mathbb{S}$(根据 (8) 、 (2) 、 (10) 和 [$\Gamma$-Com])

$\Theta \cdot \Gamma_0, \Gamma_{s0}, s[\mathsf{q}] : \mathbb{S}_k \vdash_\emptyset Q_k[^w/_{x_k}]$ (根据（5c）、（9b）、（11）和 [T-Lift]) (12)$\Theta \cdot \Gamma_{s1}, s[\mathsf{p}_k] : \epsilon \vdash_\Sigma s : \sigma$ (根据（9a）、（11）、gc，且由于 $s[\mathsf{p}_k] \notin \mathrm{dom}(\Gamma_{s1})$) (13)$\Theta \cdot \Gamma' \vdash_\Sigma P'$ (由于 $\Sigma \cap \emptyset = \emptyset$，且根据（12）、（13）和 [T-|]) (14)$\Gamma \to_{(\Sigma;\mathcal{R})} \Gamma'$ (根据（11），显然成立，因为未使用 $\mathcal{R}$) (15) (Σ; $\mathcal{R}$)-$\varphi_s(\Gamma')$ (由于 (Σ; $\mathcal{R}$)-$\varphi_s(\Gamma)$，且根据（15）和 [S-$\to$]) (16).∴ 根据（14）、（15）和（16），定理 1 对 (C2):(SC1) 成立。

(SC2) 子情况 [T-σ_2] 与子情况 **(SC1)**类似。因此，由于定理1 对所有子情况均成立，故其对 case **(C2)**也成立。

(C3) Case [R-$\textcircled{\uparrow}$]

$P = s[\mathsf{q}] \,\&_{i\in I}\{[\mathsf{p}_i]?\,\mathsf{m}_i(x_i).Q_i, \textcircled{\uparrow}.Q'\} \mid s : \sigma$ (由 **(A3)**， [**R-**$\textcircled{\uparrow}$]的逆) (1) $P' = Q' \mid s : \sigma$ (由 **(A3)**， [**R-**$\textcircled{\uparrow}$]的逆) (2)

(3) $\Gamma = \Gamma_\&$、 Γ_s 以及 $\Sigma = \Sigma_1 \cup \Sigma_2$ 满足： (由 **(A1)**， [T-|] 的逆)
$\Theta \cdot \Gamma_\& \vdash_{\Sigma_1} s[\mathsf{q}] \,\&_{i\in I}\{[\mathsf{p}_i]?\,\mathsf{m}_i(x_i).Q_i, \textcircled{\uparrow}.Q'\}$ (3a)
$\Theta \cdot \Gamma_s \vdash_{\Sigma_2} s : \sigma$ (3b)
$\Sigma_1 \cap \Sigma_2 = \emptyset$ (3c)

能够为（3a）中的进程确定类型的唯一规则是 [T-&]。因此我们可以得出：

(4) $\Sigma_1 = \emptyset$ 和 $\Theta \cdot \Gamma_\& \vdash s[\mathsf{q}] \,\&_{i\in I}\{[\mathsf{p}_i]?\,\mathsf{m}_i(x_i).Q_i, \textcircled{\uparrow}.Q'\}$ （依据（25a），[T-&], 和 [T-Lift])

(5) $\Gamma_\& = \Gamma_0, \Gamma_1$ st: (由 (4) 和 [T-&] 的逆)
$\Gamma_1 \vdash s[\mathsf{q}] : \,\&_{i\in I}\{\mathsf{p}_i?\ \mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i, \textcircled{\uparrow}.\mathbb{S}'\}$ (5a)
$\Theta \cdot \Gamma_0, \ s[\mathsf{q}] : \ \mathbb{S}' \vdash Q'$ (5b)
$\forall i \in I : \ \Theta \cdot \Gamma_0, \ x_i : \ \mathbb{T}_i, \ s[\mathsf{q}] : \ \mathbb{S}_i \vdash Q_i$ (5c)

(6) $\Gamma_1 = s[\mathsf{q}] : \,\&_{i\in I}\{\mathsf{p}_i?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i, \textcircled{\uparrow}.\mathbb{S}'\}$ （根据（5a）以及 [T-Var]）
(7) $\Sigma = \Sigma_2 = \{s\}$, $\therefore s \in \Sigma$ （根据（3c）并对（3b）作归纳）
(8) $\exists k \in I : \ \mathsf{p}_k \notin \mathcal{R}\ (\mathsf{q})$ （根据 **(A2)**、（6）以及 [S-$\mathcal{R}_2$]）

(9) $\Gamma = \Gamma_0, \Gamma_1, \Gamma_s \xrightarrow{s[\mathsf{p}\ \ ]\textcircled{\uparrow}}_\Sigma \Gamma' = \Gamma_0, \Gamma_1', \Gamma_s$ st: (由（7）、（8）和 [Γ-$\textcircled{\uparrow}$])
(9a) $\Gamma_1' = s[\mathsf{q}] : \mathbb{S}'$

$\Theta \cdot \Gamma_0$, $\Gamma_1' \vdash Q'$（根据 (5b) 和 (9a)）（10）$\Theta \cdot \Gamma_0$, $\Gamma_1' \vdash_\emptyset Q'$（根据（10）和 [T-Lift]）（11）$\Theta \cdot \Gamma' \vdash_\Sigma P'$（根据 [T-|]，使用（11）、（（7）和（3b））、$\Sigma \cap \emptyset = \emptyset$）（12）$\Gamma \to_{(\Sigma;\mathcal{R})} \Gamma'$（根据（8）和（9））（13）$(\Sigma;\ \mathcal{R})$-$\varphi_s(\Gamma')$（因为（A2），并根据（13）和 [S-→]）（14）∴ 根据（12）、（13）和（14），定理 1 对情形 (C3) 成立。

(C4) case [R-+]

(1) $P = P_1 + P_2$ (由 **(A3)** 以及 R- 的逆 [**R-+**])

唯一能使 **(A1)** 成立的情况是 $\Sigma = \emptyset$（否则没有适用的类型规则），因此我们得出：

$\Sigma = \emptyset$（由 **(A1)** 得出）(2)$\Theta \cdot \Gamma \vdash P$（由 (2) 以及 [T-Lift] 的逆得出）(3)$\Theta \cdot \Gamma \vdash P_1$ 以及 $\Theta \cdot \Gamma \vdash P_2$（由 (3) 以及 [T-+] 的逆得出）(4)

现在我们考虑两个子情况，分别对应非确定性选择的每一种可能归约：(SC1) 子情况：令 $P' = P_1$

(5)∴ 依据（5），$\Theta \cdot \Gamma \vdash_\Sigma P'$（由 (4)、(2) 以及 [T-Lift] 得出），$\Gamma \to^0_{(\Sigma;\mathcal{R})} \Gamma$，且 **(A2)**，定理 1 对 **(C4)** 成立：**(SC1)**。

(SC2) 子情况：设 $P' = P_2$

(6) $\Theta \cdot \Gamma \vdash_\Sigma P'$ （依据（4）、（2），以及 [T-Lift]）

∴ 依据（6），$\Gamma \to^0_{(\Sigma;\mathcal{R})} \Gamma$，且 **(A2)**，定理 1 对 **(C4)** 成立：**(SC2)**。

因此，由于定理1 对所有子情况均成立，所以它对 case **(C4)** 成立。

（C5）案例 [R-***X***]

$P =_{\text{def}} X(x_1, \ldots, x_n) = Q$ 属于 $(X\langle w_1, \ldots, w_n\rangle \mid Q')$（根据 [R-***X***] 的逆）(1)$P' =_{\text{def}} X(x_1, \ldots, x_n) = Q$ 属于 $(Q[{}^{w_1}/_{x_1}] \cdots [{}^{w_n}/_{x_n}] \mid Q')$（根据 [R-***X***] 的逆）(2)

(3) $\Theta, X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot x_1\colon \mathbb{T}_1, \ldots, x_n\colon \mathbb{T}_n \vdash Q$ (根据 **(A1)**、(1) 和 [T-Def] 的逆)

(4) $\Theta, X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma \vdash X\langle w_1, \ldots, w_n\rangle \mid Q'$ （根据 **(A1)**，(1) 以及 [T-Def] 的逆）

(5) $\Gamma = \Gamma_1, \Gamma_2$ st: （根据（4）以及 [T-|] 的逆）

$\Theta, X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma_1 \vdash X\langle w_1, \ldots, w_n\rangle$ (5a)

$\Theta, X\colon \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma_2 \vdash Q'$ (5b)

(6) $\Gamma_1 = \Gamma_{11}, \ldots, \Gamma_{1n}, \Gamma_1'$ 秒: (由 (5a) 和 [T-Call] 的逆)

$\forall i \in 1..n \cdot \Gamma_{1i} \vdash w_i : \mathbb{T}_i$ (6a)

结束(Γ_1') (6b)

(7) $\Theta, X : \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma_1 \vdash Q[w_1/x_1] \cdots [w_n/x_n]$ (依据 (3)、(6a)、(6b))

(8) $\Theta, X : \mathbb{T}_1, \ldots, \mathbb{T}_n \cdot \Gamma \vdash Q[w_1/x_1] \cdots [w_n/x_n] \mid Q'$ (由 7、5b，以及 [T-|])

(9) $\Theta \cdot \Gamma \vdash P'$ (由 3和 8，以及 [T-Def])

∴ 由 9， $\Gamma \to^0_{(\Sigma;\mathcal{R})} \Gamma$，以及**(A2)**，定理 1 对 case **(C5)**成立。

(C6) 情形 [R-$\mathbb{C}$]

现在，我们进一步对 $\mathbb{C}$ 的结构进行归纳证明。由定义 9可知，我们必须证明定理 1 适用于以下三种情形:

(SC1) $P = Q \mid P_{\mathbb{C}}$和 $P' = Q' \mid P_{\mathbb{C}}$和 $Q \longrightarrow Q'$

(1) $\Gamma = \Gamma_Q, \Gamma_{\mathbb{C}}$ st: (根据 [T-|] 的逆)

$\Theta \cdot \Gamma_Q \vdash_{\Sigma_Q} Q$(1a)

$\Theta \cdot \Gamma_{\mathbb{C}} \vdash_{\Sigma_{\mathbb{C}}} P_{\mathbb{C}}$ (1b)

$\Sigma_Q \cap \Sigma_{\mathbb{C}} = \emptyset$ 和 $\Sigma = \Sigma_Q \cup \Sigma_{\mathbb{C}}$ (1c)

(2) $(\Sigma; \mathcal{R})$-$\varphi_s(\Gamma_Q)$ (由**(A2)** 和引理 1)

(3) $\exists \Gamma_Q'$ st: (由 (1a), (2)，因为 $Q \longrightarrow Q\prime$，并由归纳假设)

$\Theta \cdot \Gamma_Q' \vdash_{\Sigma_Q} Q'$ (3a)

$\Gamma_Q \longrightarrow_{(\Sigma_Q;\mathcal{R})} \Gamma_Q'$ (3b)

$(\Sigma_Q; \mathcal{R})$-$\varphi_s(\Gamma_Q')$ (3c)

(4) $\Theta \cdot \Gamma_Q', \Gamma_{\mathbb{C}} \vdash_{\Sigma} P'$ (由 (3a)、(1b)、(1c)以及 [T-|])

(5) $\Gamma \longrightarrow_{(\Sigma;\mathcal{R})} \Gamma_Q', \Gamma_{\mathbb{C}}$ (由 (3b)、[Γ-合同行为], 且自 $\Sigma_Q \subseteq \Sigma$以来)

由于 $P_{\mathbb{C}}$ 在 Σ_Q 中不通过会话进行通信，我们得到:

(6) $(\Sigma_{\mathbb{C}}; \mathcal{R})$-$\varphi_s(\Gamma_{\mathbb{C}})$ (因为 **(A2)**、引理 1和（1c）)

(7) $(\Sigma; \mathcal{R})$-$\varphi_s(\Gamma_Q', \Gamma_{\mathbb{C}})$ (根据（3c）、（6）、（1c）和引理2)

∴ 由（4）、（5）和（7），定理1 适用于**(SC1)**。

(SC2) $P = (\nu s)\ Q$ 和 $P' = (\nu s)\ Q'$ 和 $Q \longrightarrow Q'$

$s:\ \Gamma'$ 圣
(8) $\Gamma' = \{s[\mathsf{p}]:\ \tau_{\mathsf{p}}\}_{\mathsf{p}\in\rho}$ （由 [T-ν] 的逆）
(8a) $s \notin \Gamma$ (8b) $(\{s\};\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma')$ (8c) $\Theta \cdot \Gamma$,
$\Gamma' \vdash_{\Sigma\cup\{s\}} Q$ (8d) $s \notin \Sigma$
(8e)

(9) $(\Sigma \cup \{s\};\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma, \Gamma')$ （由 **(A2)**、（8c）、（8e）以及引理2）

由于 Γ 和 Γ' 的会话互不相交，我们可以考虑进行一次 $Q \longrightarrow Q'$ 会话s上的归约，产生 Γ'；或者在另一个会话上进行归约，产生 Γ。
对于后一种情况，定理 1 可直接由归纳假设得出。
对于前一种情况，下面考虑涉及会话 s 的归约：

$\exists \Gamma''$ st:
(10) $\Gamma,\ \Gamma' \longrightarrow_{(\{s\};\mathcal{R})} \Gamma,\ \Gamma''$ (因为 $Q \longrightarrow Q'$，且由（8d）、（9）及归纳假设)
(10a) $\Theta \cdot \Gamma,\ \Gamma'' \vdash_{\Sigma\cup\{s\}} Q'$
(10b) $(\Sigma \cup \{s\};\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma,\ \Gamma'')$
(10c)

(11) $\Gamma' \longrightarrow_{(\{s\};\mathcal{R})} \Gamma''$ （由（10a）及 [Γ-合同行为]）
(12) $(\{s\};\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma'')$ (作者 (8c),(11), [S-$\rightarrow$])

由于不存在能够向类型上下文中添加一个带有角色的新会话的 (11) 归约，因此我们得到：

(13) $\Gamma'' = \{s[\text{段落}]:\ \tau'_{\mathsf{p}}\}_{\mathsf{p}\in\rho}$ (根据 (8a) 和 (11))
(14) $\Theta \cdot \Gamma \vdash_{\Sigma} (\nu s{:}\Gamma'')\ Q'$ （由（13）、（8b）、（12）、（10b）以及 [T-ν]）

∴ 因为 $\Gamma \longrightarrow^{0}_{\Sigma;\mathcal{R}} \Gamma$，并且根据**(A2)** 和（14），定理1 适用于**(SC2)**。

(SC3) $P =$ 定义 D 于 Q 和 $P' =$ 定义 D 于 Q' 和 $Q \longrightarrow Q'$此子情况可直接由归纳假设推出。∴ 由于定理1 适用于全部三个子情况，因此它也适用于条目**(C6)**。

(C7) 情况 **[R-↓]** 此情况可直接由**(A1)**、**(A2)**、 $\Gamma \rightarrow^{0}_{(\Sigma;\mathcal{R})} \Gamma$以及引理3 .

∴ 定理1 成立，因为它适用于所有可能的情况。 □

推论 1（故障处理安全性）给定可靠性函数 $\mathcal{R}: \mathsf{p} \notin \mathcal{R}(\mathsf{q})$，且 $\Theta \cdot \Gamma \vdash_{\Sigma} P$with $(\Sigma;\ \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$，并且 $P \longrightarrow^{*}_{\mathcal{R}} P' \equiv \mathbb{C}[Q]$ implies $Q \neq s[\mathsf{q}]\,\&_{i\in I}\{\ldots, [\mathsf{p}]?\,\mathsf{m}(x).Q'\}$。即， Q 不能是一个在 q 接收来自 p 的消息、却未定义超时的分支。

证明（反证法）。假设 $Q = s[\mathtt{q}]\,\&_{i\in I}\{\ldots, [\mathtt{p}]\,?\,\mathtt{m}(x).Q'\}$，根据定理 1，有 $\exists \Gamma' : \varphi_{\mathtt{s}}(\Gamma')$ and $\Theta \cdot \Gamma' \vdash_{\Sigma} Q$。根据 [T- &] 的逆，$\Gamma'$ 的形式为 $s[\mathtt{q}] : \&_{i\in I}\{\mathtt{p}_i\,?\,\mathtt{m}_i(x_i).\mathbb{S}_i\}$ and $\exists k \in I : \mathtt{p}_k = \mathtt{p}$。但 $\mathtt{p} \notin \mathcal{R}(\mathtt{q})$，因此违反了 [S-$\mathcal{R}_1$]。矛盾，$\therefore Q \neq s[\mathtt{q}]\,\&_{i\in I}\{\ldots, [\mathtt{p}]\,?\,\mathtt{m}(x).Q'\}$。□

推论 2（可靠性遵循）给定可靠性函数 $\mathcal{R}$：$\mathcal{R}(\mathtt{q}) = \mathfrak{R}_{\mathtt{q}}$ 和 $\Theta \cdot \Gamma \vdash_{\Sigma} P$，满足 $(\Sigma; \mathcal{R})$-$\varphi_{\mathtt{s}}(\Gamma)$ 且 $P \longrightarrow^*_{\mathcal{R}} P' \equiv \mathbb{C}[Q]$ 蕴含 $Q \neq s[\mathtt{q}]\,\&_{i\in I}\{[\mathtt{p}_i]\,?\,\mathtt{m}_i(x_i).Q_i,\ \text{›}.Q'\}$ st: $\forall i \in I : \mathtt{p}_i \in \mathfrak{R}_{\mathtt{q}}$。也就是说，$Q$ 不能是在 q 处仅接收来自可靠角色 $\mathtt{p}_i$ 的分支，并定义超时。

证明。证明与推论相似。1，但会因违反 [S-$\mathcal{R}_2$] 而导致矛盾。□

B 会话保真性

命题 2（主体同余）。假设 $\Theta \cdot \Gamma \vdash_{\Sigma} P$ 和 $P \equiv P'$，则 $\exists \Gamma'$ 使得 $\Gamma \equiv \Gamma'$ 且 $\Theta \cdot \Gamma' \vdash_{\Sigma} P'$。

证明。通过对 $P \equiv P'$ 的推导进行归纳法来证明。大多数 case 都很直接，对 $\Gamma = \Gamma'$ 成立。下面我们详细说明两个缓冲区同余的 case：

1. $P = (\nu s : \Gamma'')\ s : \sigma \equiv \mathbf{0} = P'$。我们必须有 $\Theta \cdot \Gamma'' \vdash_{\{s\}} s : \sigma$ 和 end(Γ)。这是因为 Γ'' 包含检查 $s : \sigma$ 所需的所有类型（由缓冲区跟踪器保证）。因此 gc(Γ, Γ'') 成立，并且由于 $s \notin \Gamma$，所以 end(Γ)。$\therefore$ 我们根据类型规则 [T-0]，由 $\Gamma' = \Gamma$ 得出结论。
2. $P \equiv P'$ 通过交换缓冲区中的消息位置。对于此 case，我们通过应用相应的缓冲区类型消息交换操作，在类型中镜像消息重排序操作（图 4）。□

˜命题 3（范式）。对于所有 P，$P \equiv \text{def}\ D \text{ in } (\nu \tilde{s})\ P_1 \mid \cdots \mid P_n \text{ where } \forall i \in 1..n, P_i$，其要么是分支（可能带有超时）、选择、进程调用或会话缓冲区。

证明。与 [32, 命题 L.1] 类似。□

引理 4（会话逆转）。假设 $\emptyset \cdot \Gamma \vdash_{\Sigma} (\Pi_{\mathtt{p}\in I} P_{\mathtt{p}})$ 其中每个 $P_{\mathtt{p}}$ 均为 $\mathbf{0}$（至多为-$\equiv$）或仅扮演角色 $\mathtt{p}$ 于 s。则 $\Gamma = \{s[\mathtt{p}] : \mathbb{S}_{\mathtt{p}}\}_{\mathtt{p}\in I'}$，$\{s[\mathtt{p}] : \mathbb{M}_{\mathtt{p}}\}_{\mathtt{p}\in I''}$ 对于某个 I'，I''。此外，$\forall \mathtt{p} \in I'$：

1. 若 $\mathbb{S}_{\mathtt{p}} = \oplus_{i\in I}\{\mathtt{q}^j\,!\,m^j(\mathbb{T}_j).\mathbb{S}_j\}$ 则 $\mathtt{p} \in I$ 且对于某个 $\mathbb{C}$，$\mathbb{C}'$ 并且 $k \in J$，满足以下任一条件：

$$P_{\mathtt{p}} \equiv \mathbb{C}[s[\mathtt{p}] \oplus [\mathtt{q}_k]\,!\,m_k\langle w_k\rangle.P'_{\mathtt{p}}] \quad or \qquad \text{其中 } w \text{ 是某个值 } v \text{ 或某个 } s'[\mathtt{r}]$$

$$P_{\mathtt{p}} \equiv \mathbb{C}\left[\begin{array}{l} def\ X(x_1 : \mathbb{T}_1, \ldots, x_n : \mathbb{T}_n) = \\ \qquad \mathbb{C}'[x_l \oplus [\mathtt{q}_k]\,!\,m_k\langle d_k\rangle.P'_{\mathtt{p}}] \quad in \\ X\langle w_1, \ldots, w_{l-1}, s[\mathtt{p}], w_{l+1}, \ldots, w_n\rangle \end{array}\right] \qquad \begin{array}{l}\text{具有 } 1 \le l \le n \\ \text{且每个 } w_m \text{ 要么是} \\ \text{某个值 } v_m \text{ 或某个 } s'_m[\mathtt{r}_m]\end{array}$$

2. 若 $\mathbb{S}_{\mathtt{p}} = \&_{j\in J}\{\mathtt{q}^j?\ m^j(\mathbb{T}_j).\mathbb{S}_j\}$，则 $\mathtt{p} \in I$ 且对于某个 $\mathbb{C}$，$\mathbb{C}'$，满足以下任一条件：

$$P_{\mathtt{p}} \equiv \mathbb{C}[s[\mathtt{p}]\,\&_{j\in J}\{[\mathtt{q}_j]\,?\,m_j(x_j).P'_{\mathtt{p}_j}\}] \quad or \qquad \text{其中 } w \text{ 是某个值 } v \text{ 或某个 } s'[\mathtt{r}]$$

$$P_{\mathtt{p}} \equiv \mathbb{C}\left[\begin{array}{l} def\ X(x_1 : \mathbb{T}_1, \ldots, x_n : \mathbb{T}_n) = \\ \qquad \mathbb{C}'[x_l\,\&_{j\in J}\{[\mathtt{q}_j]\,?\,m_j(x_j).P'_{\mathtt{p}_j}\}] \quad in \\ X\langle w_1, \ldots, w_{l-1}, s[\mathtt{p}], w_{l+1}, \ldots, w_n\rangle \end{array}\right] \qquad \begin{array}{l}\text{带有 } 1 \le l \le n \\ \text{且每个 } w_m \text{ 都是} \\ \text{某个值 } v_m \text{ 或某个} s'_m[\mathtt{r}_m]\end{array}$$

3. 若 $\mathbb{S}_{\mathsf{p}} = \&_{j\in J}\{\mathbf{q}_j?\ m_j(\mathbb{T}_j).\mathbb{S}_j,\ \textcircled{\scriptsize\uparrow}.\mathbb{S}'\}$ 则 $\mathsf{p} \in I$ 且对于某个 $\mathbb{C}$，$\mathbb{C}'$，满足以下任一条件：

$$P_{\mathsf{p}} \equiv \mathbb{C}[s[\mathsf{p}]\,\&_{j\in J}\{[\mathbf{q}_j]?\,m_j(x_j).P'_{\mathsf{p}_j},\ \textcircled{\scriptsize\uparrow}.P'_{\mathsf{p}_t}\}] \quad or$$

其中 w 是某个值 v 或某个 $s'[\mathsf{r}]$

$$P_{\mathsf{p}} \equiv \mathbb{C}\left[\begin{array}{l}\mathit{def}\ X(x_1:\mathbb{T}_1,\ldots,x_n:\mathbb{T}_n) = \\ \qquad \mathbb{C}'[x_l\,\&_{j\in J}\{[\mathbf{q}_j]?\,m_j(x_j).P'_{\mathsf{p}_j},\ \textcircled{\scriptsize\uparrow}.P'_{\mathsf{p}_t}\}]\ \ \mathit{in} \\ X\langle w_1,\ldots,w_{l-1},s[\mathsf{p}],w_{l+1},\ldots,w_n\rangle\end{array}\right]$$

其中 $1 \le l \le n$ 且每个 w_m 要么是某个值 v_m 或某个 $s'_m[\mathsf{r}_m]$

此外，4. $\forall \mathsf{p} \in I \setminus I' :\ P_{\mathsf{p}} \equiv \mathbf{0}$。

证明。Pr of 与[32, 命题 L.5] 类似，但使用了我们对范式的定义（命题 3 ）. □

定理 2（会话保真性）假设 $\emptyset \cdot \Gamma \vdash_\Sigma P$ with $(\Sigma;\mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$，$P \equiv (\Pi_{\mathsf{p}\in I}\,P_{\mathsf{p}}) \mid s:\sigma$ and $\Gamma = \bigcup_{\mathsf{p}\in I}\Gamma_{\mathsf{p}}$，并且对于每个 P_{p}：(i) $\emptyset \cdot \Gamma_{\mathsf{p}} \vdash_\Sigma P_{\mathsf{p}}$，以及 (ii) P_{p} 为 0（至 $\equiv$）或在 s 中仅扮演角色 p，根据 Γ_{p}。那么，

$$\begin{array}{lll}\Gamma \longrightarrow_{(\Sigma;\mathcal{R})} \ \mathit{implies} & \exists \Gamma', P' : & \Gamma \longrightarrow_{(\Sigma;\mathcal{R})} \Gamma' \\ & & \mathit{and} \quad P \longrightarrow_{\mathcal{R}}^{+} P' \\ & & \mathit{and} \quad \emptyset \cdot \Gamma' \vdash_\Sigma P' \ \mathit{with}\ (\Sigma;\mathcal{R})\text{-}\varphi_{\mathsf{s}}(\Gamma') \\ & & \mathit{and} \quad P' = (\Pi_{\mathsf{p}\in I}\,P'_{\mathsf{p}}) \mid s:\sigma' \ \mathit{and}\ \Gamma' = \bigcup_{\mathsf{p}\in I}\Gamma'_{\mathsf{p}} \\ & & \mathit{and} \quad \mathit{for\ each}\ P'_{\mathsf{p}}: \\ & & \qquad \emptyset \cdot \Gamma'_{\mathsf{p}} \vdash_\Sigma P'_{\mathsf{p}},\ \mathit{and} \\ & & \qquad P'_{\mathsf{p}}\ \mathit{is}\ \mathbf{0}\ \mathit{(up\text{-}to\text{-}}\equiv\mathit{)}\ \text{or}\ \mathit{only\ plays\ role}\ \mathsf{p}\ \mathit{in}\ s,\ \mathit{by}\ \Gamma'_{\mathsf{p}}\end{array}$$

证明。 通过对 Γ 的归约进行归纳法得证。

Case $\Gamma \xrightarrow{s[\mathsf{p}]\,\textcircled{\scriptsize\uparrow}}_{(\Sigma;\mathcal{R})}$.

$$(1)\ \frac{s \in \Sigma \qquad \exists\, k \in I : \mathbf{q}_k \notin \mathcal{R}(\mathsf{p})}{\Gamma = \Gamma_{I\setminus\mathsf{p}}, \Gamma_{\mathsf{p}\mathbb{M}}, s[\mathsf{p}] : \&_{i\in I}\{\mathbf{q}_i\,?\,\mathsf{m}_i(\mathbb{T}_i).\mathbb{S}_i,\ \textcircled{\scriptsize\uparrow}.\mathbb{S}'\} \xrightarrow{s[\mathsf{p}]\,\textcircled{\scriptsize\uparrow}}_{(\Sigma;\mathcal{R})} s[\mathsf{p}] : \mathbb{S}', \Gamma_{\mathsf{p}\mathbb{M}}, \Gamma_{I\setminus\mathsf{p}} = \Gamma'}\ (\text{by } [\Gamma\text{-}_{\textcircled{\scriptsize\uparrow}}])$$

根据假设（1）和引理 4，进程 P_{p} 是一个具有已定义超时的分支（可能出现在某个进程定义之下）。

$\therefore P \longrightarrow_{\mathcal{R}}^{+} P'$ 通过一次 [R-$_{\textcircled{\scriptsize\uparrow}}$]，归约，或在有限次 [R-$\boldsymbol{X}$] 后应用 [R-$_{\textcircled{\scriptsize\uparrow}}$]。

然后根据定理 1，可得 $\emptyset \cdot \Gamma' \vdash_\Sigma P'$ with $(\Sigma;\mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma')$。

通过分析约简规则 [**R-**$_{\textcircled{\scriptsize\uparrow}}$] 和 [**R-**$\boldsymbol{X}$]，我们观察到 P' 保持了 $P' = (\Pi_{\mathsf{q}\in I\setminus\mathsf{p}}\,P_{\mathsf{q}}) \mid P'_{\mathsf{p}} \mid s:\sigma'$ (对于 $\sigma = \sigma'$) 所要求的结构。同样，$\Gamma' = \bigcup_{\mathsf{q}\in I\setminus\mathsf{p}}\Gamma_{\mathsf{q}} \cup \Gamma'_{\mathsf{p}} = \Gamma_{\mathsf{p}\mathbb{M}}, s[\mathsf{p}]:\mathbb{S}'$ 对于某个可能为空的队列类型 $\Gamma_{\mathsf{p}\mathbb{M}}$。

最后，我们已经知道，对于每个 P_{q}：$\emptyset \cdot \Gamma_{\mathsf{q}} \vdash_\Sigma P_{\mathsf{q}}$，$P_{\mathsf{q}}$ 均为 0（在-$\equiv$范围内）或仅在 s 中扮演角色 $\mathbf{q}$，根据 Γ_{q}。因此，剩下要证明的只有 $\emptyset \cdot \Gamma'_{\mathsf{p}} \vdash_\Sigma P'_{\mathsf{p}}$ 和 P'_{p} 为 0（在-$\equiv$范围内）或仅在s中扮演角色 p，根据 Γ'_{p}——这可由规则 [T-&] 得出。

情形 $\Gamma \xrightarrow{s[\mathsf{p}]\,!\mathbf{q}:\mathsf{m}(\mathbb{T})}_{(\Sigma;\mathcal{R})}$ 以及 $\Gamma \xrightarrow{s[\mathsf{p}]\,[\mathbf{q}]:\mathsf{m}}_{(\Sigma;\mathcal{R})}$ 遵循与第一个情形相似的步骤。我们使用上下文约简规则推导 Γ 的结构，并利用引理 4 得到 P 的结构。我们按照上述步骤，运用类似的推理继续证明。□

C 进程性质与可判定性

定理 3（进程性质验证）假设：$\emptyset \cdot \Gamma \vdash_{\{s\}} P$with $(\{s\}; \mathcal{R})$-$\varphi_{\mathsf{s}}(\Gamma)$，$P \equiv (\Pi_{\mathsf{p}\in I}\, P_{\mathsf{p}}) \mid s : \sigma$ 以及 $\Gamma = \bigcup_{\mathsf{p}\in I} \Gamma_{\mathsf{p}}$。此外，对于每个 P_{p}：(i) $\emptyset \cdot \Gamma_{\mathsf{p}} \vdash_{\{s\}} P_{\mathsf{p}}$，以及 (ii) $P_{\mathsf{p}} \equiv \mathbf{0}$ 或 P_{p}仅在 s中扮演角色 p，根据 Γ_{p}。

那么，$\forall \phi \in$ {无可靠通信、无死锁、终止、永不终止、活性}，如果$(\Sigma; \mathcal{R})$-$\phi(\Gamma)$，则 P 是 ϕ。

证明。这些结果可由定理2推出。

定理 4（可判定性）如果$(\Sigma; \mathcal{R})$-$\phi(\Gamma)$ 是可判定的，那么“$\Theta \cdot \Gamma \vdash_{\Sigma} P$带有$(\Sigma; \mathcal{R})$-$\phi(\Gamma)$”是可判定的。

证明。通过对图6中的类型规则求逆，可以得到一个用于类型检查的可判定算法。需要注意的是：

(*1*对于 P 的每种形式，至多只有一条规则可以确定 $\Theta \cdot \Gamma \vdash_{\Sigma} P$，但 $\Theta \cdot \Gamma \vdash_{\Sigma} s : \sigma$ 这一情况除外，此时应忽略 $[\text{T-}\sigma_w]$，因为可以通过$[\text{T-}\epsilon]$对剩余缓冲区类型进行集体垃圾回收；（2）每条规则都会确定性地填充 Θ 和 Γ；（3）每条规则都会产生 P 的更小子项，从而确保递归的类型检查能够终止。

最后，规则 $[\text{T-}\nu]$ 的可判定性取决于 φ_{s}的可判定性，而根据假设，后者是可判定的。□

定理 5（可判定的有界性质）对于所有 Σ、$\mathcal{R}$、k，$(\Sigma; \mathcal{R})$-有界$_k(\Gamma)$ 是可判定的。此外，如果 $(\Sigma; \mathcal{R})$-有界(Γ)，那么对于 $\forall \phi \in$ {安全、可靠通信安全、无死锁、终止、永不终止、活性}，$(\Sigma; \mathcal{R})$-$\phi(\Gamma)$ 是可判定的。

证明。$(\Sigma; \mathcal{R})$-有界$_k(\Gamma)$ 是可判定的，因为它在有限次归约中即可确定——也就是说，即使 Γ 可以进行无限归约，对有界条件 k 的违反也会在无限归约之前发生。此外，如果 $(\Sigma; \mathcal{R})$-有界$_k(\Gamma)$，那么 Γ 可以表示为有限状态转换系统，因此对于 $\forall \phi \in$ {安全、可靠通信安全、无死锁、终止、永不终止、活性}，$(\Sigma; \mathcal{R})$-$\phi(\Gamma)$ 是可判定的。□

D 补充示例

D.1 领导者选举

现在我们给出一个领导者选举协议的示例类型（基于 Raft 共识算法 [25]），该协议具有以下属性：

1. 选举集群由 3 个静态参与者组成；2. 参与者知道集群规模；3. 该协议可容忍所有非拜占庭故障；4. 参与者采用故障停止模型，即节点发生故障意味着它立即崩溃；5. 该协议假设集群规模为 $2f+1$ 时，故障阈值上限为 f，即同一时间最多有 1 个崩溃节点。

规范。 在任何时刻，参与者都处于以下状态之一： 跟随者、 候选者或 领导者。所有参与者一开始都是 跟随者。处于 跟随者 状态时，参与者不能主动发起通信，只能在收到消息后回复。 跟随者 可以在候选者请求时，投票赞成或反对。如果 跟随者 等待过久仍未收到任何消息，就会以非确定性方式超时，并成为 候选者。 候选者 会向整个集群发送投票请求，并需要获得简单多数票才能当选。集群规模为 3 名参与者；由于 候选者 总是投票给自己，因此只需再获得 1 张外部选票即可赢得选举。 候选者 赢得选举后，就会晋升为 领导者。 领导者 会定期发送心跳消息，以阻止 跟随者 超时并发起不必要的新选举。感兴趣的读者可参阅 [25] 了解更多详情。

示例 7（领导者选举）。所讨论的算法是（准）对称的，这意味着可以针对一个（泛化的）角色描述该类型，并为每个参与者复制该描述。我们使用角色 p、q 和 r，以区分该类型中的三个不同参与者。请注意，与本文介绍的其余内容不同，p、q 和 r 并不代表特定角色，而是等待实例化的占位符。

$$
\begin{aligned}
\mathbb{S}_{\mathsf{p}} = \mu \mathrm{F}.\, \&\Big\{ & \mathsf{q}?\ \mathsf{hb}().\oplus\{\mathsf{q}!\ \mathsf{ok}()。\mathrm{F},\ \mathsf{q}!\ \mathsf{ko}()。\mathrm{F}\},\\
& \mathsf{r}?\ \mathsf{hb}().\oplus\{\mathsf{r}!\ \mathsf{ok}()。\mathrm{F},\ \ \mathsf{r}!\mathsf{ko}().\mathrm{F}\},\\
& \mathsf{q}?\ \mathsf{rv}().\oplus\{\mathsf{q}!\ \mathsf{yes}()。\mathrm{F},\ \mathsf{q}!\ \mathsf{no}()。\mathrm{F}\},\\
& \mathsf{r}?\ \mathsf{rv}().\oplus\{\mathsf{r}!\ 是()。\mathrm{F},\ \ \ \mathsf{r}!\ 否()。\mathrm{F}\},\\
& \mathsf{q}?\ 是().F,\ \ \mathsf{r}?\ 是().F,\ \ \mathsf{q}?否().F,\ \ \ \mathsf{r}?否().F,\\
& \mathsf{q}?\,\mathsf{ok}().F,\ \ \mathsf{r}?好().F,\ \ \ \mathsf{q}?\ \ \mathsf{ko}().F,\ \ \mathsf{r}?\,\mathsf{ko}().F,\\
& {}_{①}.\ \mu \mathrm{C}。\ 问!\ \mathsf{rv}()。\mathsf{r}!\ \ \mathsf{rv}()。\ \ \mu C'.\ \&\Big\{
\begin{aligned}
& \mathsf{q}?心跳().\oplus\{\mathsf{q}!正常().\mathrm{F},\mathsf{q}!失败().C'\},\\
& \mathsf{r}?\,\mathsf{hb}().\oplus\{\mathsf{r}!\,\mathsf{ok}().\mathrm{F},\ \mathsf{r}!\ \ \mathsf{ko}().C'\},\\
& \mathsf{q}?\,\mathsf{rv}().\oplus\{\mathsf{q}!是()。\mathrm{F},\ \ \ \mathsf{q}!\ 不()。C'\},\\
& \mathsf{r}?\,\mathsf{rv}().\oplus\{\mathsf{r}!是()。\mathrm{F},\ \ \ \mathsf{r}!\ 不()。C'\},\\
& \mathsf{q}?\ 否()。C',\ \ \ \mathsf{r}?不()。C',\\
& \mathsf{q}?好的()。C',\ \ \ \mathsf{r}?\mathsf{ok}()。C',\\
& \mathsf{q}?\,\mathsf{ko}().C',\ \ \mathsf{r}?\,\mathsf{ko}().C',\\
& \mathsf{q}?是().\mathbb{S}_{\mathsf{p}}^{L}\ ,\ \mathsf{r}?\ 是().\mathbb{S}_{\mathsf{p}}^{L}\\
& {}_{①}.\ \mathrm{C}
\end{aligned}\Big.\\
\Big.\Big\}
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{S}_{\mathsf{p}}^{L} = \mu \mathrm{L}.\mathsf{q}!\,\mathsf{hb}().\mathsf{r}!\,\mathsf{hb}().\mu \mathrm{L}'.\ \&\Big\{ & \mathsf{q}?\ \ \mathsf{hb}().\oplus\{\mathsf{q}!\ \ 好()。\mathrm{F},\ \ \mathsf{q}!\ \ 不()。\mathrm{L}'\},\\
& \mathsf{r}?\ \mathsf{hb}().\oplus\{\mathsf{r}!\ \ 好()。\mathrm{F},\ \ \ \mathsf{r}!\ \ 不()。\mathrm{L}'\},\\
& \mathsf{q}?\ \ \mathsf{rv}().\oplus\{\mathsf{q}!\ \ 是()。\mathrm{F},\ \ \ \mathsf{q}!\ \ 否()。\mathrm{L}'\},\\
& \mathsf{r}?\ \ \mathsf{rv}().\oplus\{\mathsf{r}!\ \ 是()。\mathrm{F},\ \ \mathsf{r}!\ \ 否()。\mathrm{L}'\},\\
& \mathsf{q}?\ \ 是()。\mathrm{L}',\ \ \ \mathsf{r}?\ \ 是()。\mathrm{L}',\ \ \ \mathsf{q}?\ \ 否()。\mathrm{L}',\ \ \ \mathsf{r}?\ \ 否()。\mathrm{L}',\\
& \mathsf{q}?\ \ 确定()。\mathrm{L}',\ \ \ \mathsf{r}?\ \ 确定()。\mathrm{L}',\ \ \mathsf{q}?\,\mathsf{ko}().\mathrm{L}',\ \ \mathsf{r}?\,\mathsf{ko}().\mathrm{L}',\\
& {}_{①}.\ \mathrm{L}
\end{aligned}\Big.\Big\}
$$