会话类型子类型关系的逻辑阐释

RossHorne Luca Padovani卢森堡大学 卡梅
里诺大学

我们在逻辑框架下研究会话类型的子类型关系，其中会话类型是乘法/加法线性逻辑的命题，并扩展了最小不动点和最大不动点。由此得到的子类型关系具有一种简单的刻画方式，大致可以概括为这样一句显而易见的话：每个会话类型都大于最小会话类型，并小于最大会话类型。同时，我们发现，与传统的子类型关系不同，这种子类型关系除了保持会话通常具有的安全属性外，还保持终止性。我们提出了一种采用该子类型关系的会话演算，并证明，尽管子类型关系在实践中很有用，但在理论上却是多余的：所有子类型关系的用法都可以通过强制转换语义“编译掉”。

1 引言

会话类型 [12, 13, 15] 是对通信协议的描述，它们与线性逻辑之间优雅的对应关系 [23, 3, 16] 为会话类型系统提供了坚实的逻辑基础。举例来说，下面给出了一个会话类型的定义，用以描述一个数学 Server 实现的协议（在本节的示例中，$\&$和 $\oplus$ 是分别表示外部和内部带标签选择的 n 元运算符）：

$$B = \&\{\mathsf{end} : \bot, \mathsf{add} : \mathsf{Num}^{\bot} \parr \mathsf{Num}^{\bot} \parr \mathsf{Num} \otimes B\}$$

根据会话类型 B，Server 首先等待一个标签——end 或 add——以确定 Client 请求执行的操作。如果标签是 end，Client 就没有更多请求，Server 随即终止。如果标签是 add，Server 就等待两个数字，将它们的和发送回 Client，然后再次准备就绪，提供相同的协议 B。在此示例中，我们用 $\mathsf{Num}^{\bot}$ 表示所接收数字的类型，用 Num 表示所产生数字的类型。该 Server 的 Client 可以实现由以下会话类型描述的通信协议：

$$A = \oplus\{\mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^{\bot} \parr \oplus\{\mathsf{end} : \mathbf{1}\}\}$$

此 Client 发送标签 add，随后发送两个数字，接收结果，然后通过发送标签 end 终止与 Server 的交互。当我们通过一个会话连接两个进程时，我们期望它们的交互毫无差错。在许多会话类型系统中，只要确保描述一个进程行为的会话类型与描述其对等方行为的会话类型互为对偶，就能保证这一点。对偶通常记作 $\cdot^{\bot}$，它是一种作用于会话类型的运算符，可以反转消息的方向，同时不改变协议的结构。在上面的例子中，显然 A 不是 B 的对偶，B 也不是 A 的对偶。尽管如此，我们仍希望这样的 Client 和这样的 Server 被声明为兼容，因为 Client 只使用了 Server 所具备能力的一个子集。为了表达这种兼容性，我们必须采用 A 和 B 之间更复杂的关系：要么观察到 B（Server 的行为）是一个比 $A^{\bot}$ 更宽容的版本，要么观察到 A（Client 的行为）是一个比 $B^{\bot}$要求更低的版本。我们通过会话类型上的子类型关系 6 来精确定义这些关系。子类型关系增强了.....的适用性

Ilaria Castellani 和 Alceste Scalas（编）：第 14 届并发与通信中心软件编程语言方法研讨会（PLACES2023）。EPTCS 378，2023 年，第 26–37 页，doi:10.4204/EPTCS.378.3

© R. Horne 和 L. Padovani 本作品采用知识共享署名许可。

通过著名的替换原则来处理类型系统：如果 C 是 D 的子类型，那么在需要 D 类型实体的地方可以使用 C 类型实体。在 Gay 和 Hole [9] 的早期工作之后，人们研究了许多会话类型的子类型关系 [4, 20, 17, 21, 10]。这些子类型关系在定义方式和／或所保留的性质方面差异很大，但它们都具有一个共同点：子类型关系本质上是由标签所确定的会话类型分支结构定义的。为说明这一点，我们再次考虑上文定义的会话类型 A 和 B。我们有

$$B \leqslant \&\{\mathsf{add} : \mathsf{Num}^{\perp} \parr \mathsf{Num}^{\perp} \parr \mathsf{Num} \otimes \&\{\mathsf{end} : \perp\}\} = A^{\perp} \tag{1}$$

这意味着，在需要一个行为如 $A^{\perp}$ 的服务器的场合，可以安全地使用一个行为如 B 的服务器。对偶地，我们还有

$$A \leqslant \oplus\{\mathsf{end} : \mathbf{1}, \mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^{\perp} \parr B^{\perp}\} = B^{\perp} \tag{2}$$

这意味着，在需要一个行为如A 的客户端的场合，可以安全地使用一个行为如 $B^{\perp}$ 的客户端。请注意，比较两个相关类型时，子类型关系的关键决定因素是可接收／发送的标签集合。在 (1) 中，类型为 B 的服务器愿意接受集合 {end,add}中的任意标签，该集合是 的超集，其中 {add} 是我们在$A^{\perp}$中拥有的集合。在 (2) 中，客户端（最初）发送集合 {add}中的一个标签，该集合是 {end,add} 的子集，而这是我们在 $B^{\perp}$中拥有的集合。会话类型中标签的这种协变／逆变，是所有已知会话类型子类型关系概念的关键区别特征。[1]

在这项工作中，我们研究会话类型的子类型关系概念，所处的框架中，会话类型是 μ MALL$^{\infty}$ [2, 6] 的命题；μMALL$^{\infty}$ [2, 6] 是乘法加法线性逻辑的无穷证明论，并扩展了最小不动点和最大不动点。我们的研究有两个目标。首先，理解当用于描述分支会话类型的联结词（线性逻辑的 $\&$ 和 $\oplus$）具有固定元数时，是否以及如何能够刻画行为众所周知的协变／逆变。其次，理解当类型推导旨在确保逻辑可靠性时，子类型关系是否存在一些变得重要的关键方面。

我们提出的方法的核心在于一个观察：当会话类型（因而也包括进程行为）由线性逻辑命题表示时， [23, 3, 16], 不可能编写出行为如 0 的进程，而编写出行为如 $\top$的进程则非常容易。如果我们将会话类型视为遵循该类型行为的进程集合，这意味着加法常量 0 和 $\top$可以很好地充当会话子类型关系中的最小元和最大元。有些出人意料的是，由 0 和 $\top$的这些性质所定义的子类型关系，使我们能够表达与通常由标签的协变／逆变所产生的子类型关系本质上相同的关系。例如，按照我们的方法，客户端的会话类型（此前记为 A）将改写为

$$C = \oplus\{\mathsf{end} : \mathbf{0}, \mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^{\perp} \parr \oplus\{\mathsf{end} : \mathbf{1}, \mathsf{add} : \mathbf{0}\}\}$$

利用它，我们可以推导出

$$B \leqslant \&\{\mathsf{end} : \top, \mathsf{add} : \mathsf{Num}^{\perp} \parr \mathsf{Num}^{\perp} \parr \mathsf{Num} \otimes \&\{\mathsf{end} : \perp, \mathsf{add} : \top\}\} = C^{\perp} \qquad \text{as well as} \qquad C \leqslant B^{\perp}$$

不比较标签，而只利用 0 是最小会话类型、 $\top$ 是最大会话类型这一事实。基本上，我们不再省略那些对应于不可能后续行为的标签（参见缺失的

[1] Gay 和 Hole [9] 及其他作者 [4, 20, 21] 以一种使式（1）和式（2）的关系恰好相反的方式，为会话类型定义子类型关系。根据会话类型被视为信道类型还是进程类型，这两种观点都行得通。本文采用后一种立场；关于这两种方法的比较，参见 Gay [8]。

进程	P, Q ::=		| (x)(P |Q)	组合	
	$A\langle\vec{x}\rangle$	调用	| 失败x	失败	
	| x().P	信号输入	| x[]	信号输出	
	| x(y).P	信道输入	| x[y](P |Q)	通道输出	
	| 情况 $x\{P,Q\}$	选择输入	| $x[在_i].P$	选择输出	$i \in \{0,1\}$

表 1：μCP$^\infty$ 的语法。

结束 并 将 作为 A），而是使用无 inhabitant 的会话类型 **0** 或其对偶 $\top$ 作为不可能的后续行为 (参见 C）。有人可能会认为，这两种方法之间的差异主要只是表面上的。事实上，很容易设计出（反）语法糖函数，将一种语法中的会话类型改写为另一种语法中的会话类型。然而，我们提出的新方法使我们能够在逻辑框架下重新表述会话类型中广为人知的子类型关系。这一成果带来的首要结果是，带子类型关系的类型系统 的可靠性不需要特设的 证明，而是可通过适当的强制转换语义，从不带子类型关系的类型系统 的可靠性推导得出。此外，我们发现，所提出的子类型关系不仅保留了通常的安全性属性——通信安全、协议保真性和无死锁性——也保留了终止性，而这是一种活性属性。

论文结构。第 2 节定义 μCP$^\infty$，一种与μCP [16] 和 CP [23]密切相关的进程会话演算。第 3 节定义 μCP$^\infty$ 的类型语言和子类型关系。第 4 节定义 μCP$^\infty$ 的类型规则，并给出子类型关系的强制转换语义，从而证明 μCP$^\infty$ 的类型系统是 μMALL$^\infty$ [2, 6]的保守扩展。最后在第 5 节总结。

2 μCP$^\infty$ 的语法和语义

μCP$^\infty$ 的语法如表1所示，使用了一组进程名称A、B、.....以及一个由信道x、y、z 等组成的无限集合。该演算包含表示通信动作的标准形式：fail x 表示进程在x上失败；x().P和x[] 表示在 x 上输入/输出终止性信号；case x{P, Q} 和x[in$_i$].P 表示在x上输入/输出标签in$_i$；x(y).P 和x[y](P |Q) 表示在 x 上输入/输出信道 y。注意，x[y](P |Q) 输出一个新信道 y，该信道在 P 中受绑定，但在Q 中不受绑定。自由信道输出可以按照先前工作 [16] 中所示的方式进行编码。形式 (x)(P | Q) 表示连接两个并行进程 P 和 Q 的会话x，而形式A⟨x⟩ 表示调用以 x 为参数的名为 A 的进程。对于每个进程名称 A，我们假设存在一个形式为A(x) ≜ P的唯一全局定义，用于给出其含义。下文中，x 表示一个可能为空的信道序列。自由信道和受绑定信道的概念按预期方式定义。我们将进程视为在受绑定信道重命名意义下相同，并用 fn(P) 表示 P 的自由信道集合。

μCP$^\infty$ 的操作语义如表2所示，由结构预同余关系 4 和归约关系 $\rightarrow$组成，这两者都相当标准。若对于某个Q，有P $\rightarrow$ Q，我们记作 P $\rightarrow$；如果不存在 P $\rightarrow$，则称 P 已停滞，记号为 P $\not\rightarrow$。

例 2.1. 我们可以将第1 节中描述的 Client 和 Server 建模为以下进程。

$$\mathsf{Client}(x) \triangleq x[\mathsf{in}_1].x[\mathsf{in}_0].x[] \qquad \mathsf{Server}(x,z) \triangleq \mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x,z\rangle\}$$

为简便起见，我们只关注进程的整体结构，而不关注它们实际执行的数学运算，因此在此模型中省略了具体数据的交换。」

$$
\begin{array}{lrll}
\text{[S-PAR-COMM]} & (x)(P \mid Q) & \preccurlyeq & (x)(Q \mid P) \\
\text{[S-PAR-ASSOC]} & (x)(P \mid (y)(Q \mid R)) & \preccurlyeq & (y)((x)(P \mid Q) \mid R) \quad x \in \mathsf{fn}(Q) \setminus \mathsf{fn}(R), y \notin \mathsf{fn}(P) \\
\text{[S-CALL]} & \mathsf{A}\langle \overline{x} \rangle & \preccurlyeq & P \quad \mathsf{A}(\overline{x}) \triangleq P \\
 & & & \\
\text{[R-CLOSE]} & (x)(x[] \mid x().P) & \to & P \\
\text{[R-COMM]} & (x)(x[y](P \mid Q) \mid x(y).R) & \to & (y)(P \mid (x)(Q \mid R)) \\
\text{[R-CASE]} & (x)(x[\mathsf{in}_i].P \mid \mathsf{case}\ x\{Q_0, Q_1\}) & \to & (x)(P \mid Q_i) \quad i \in \{0,1\} \\
\text{[R-PAR]} & (x)(P \mid R) & \to & (x)(Q \mid R) \quad P \to Q \\
\text{[R-STRUCT]} & P & \to & Q \quad P \preccurlyeq P' \to Q' \preccurlyeq Q
\end{array}
$$

表 2：μCP$^{\infty}$ 的结构前同余与归约语义。

本节最后给出类型系统所确保的性质的定义，即无死锁性和终止性。在我们的设定中，后者尤为重要，因为终止性保持是我们即将定义的子类型关系中的一个新颖方面。

定义 2.1（无死锁进程）。若 P $\Rightarrow$ Q $\not\to$ 意味着 Q 不与形如 (x)(R$_1$ |R$_2$) 的进程结构前同余，则称 P 是无死锁的。

无死锁　　进程要么可以归约，要么会停滞，等待在某个自由信道上进行同步。

定义 2.2（终止进程）。进程 P 的一次运行是一个（有限或无限）进程序列 (P$_0$,P1, . . .)，满足 P$_0$ = P，并且当 i + 1 是序列中的有效索引时，P$_i$ $\to$ P$_{i+1}$。若一次运行是无限的，或其最后一个进程处于停滞状态，则称其为极大的。若 P 的每次极大运行都是有限的，则称 P 是终止的。

请注意，终止进程不一定没有限制。例如，$(x)(\mathsf{fail}\, x \mid x[])$ 已经终止，但不具备无死锁性。真正能确保进程“行为良好”的，是无死锁性与终止性（如上所定义）的合取。

3 类型与子类型关系

μCP$^{\infty}$ 的类型语言由 μMALL$^{\infty}$ [2, 6, 1] 的命题构成。 μMALL$^{\infty}$ [2, 6, 1] 是扩展了最小不动点和最大不动点的乘法/加法线性逻辑的无穷证明论。我们从预类型的定义开始。预类型是线性逻辑命题，由取自一个无限集合的类型变量构成，并用 X 和 Y 表示。

$$\textbf{Pre-type} \qquad A, B ::= X \mid \bot \mid \mathbf{1} \mid \top \mid \mathbf{0} \mid A \parr B \mid A \otimes B \mid A \,\&\, B \mid A \oplus B \mid \nu X.A \mid \mu X.A$$

自由类型变量和约束类型变量的通常概念同样适用。类型是闭预类型。我们假设类型中出现的类型变量都是受保护的。也就是说，我们禁止形如 $\sigma_1 X_1 \ldots \sigma_n X_n . X_i$ where $\sigma_1, \ldots, \sigma_n \in \{\mu, \nu\}$ 的类型。我们用 A$^{\perp}$ 表示 A 的对偶，其定义符合预期，但需满足 X$^{\perp}$ = X。这种对偶化类型变量的方式没有问题，因为我们总是会应用

$\cdot^{\perp}$到类型，这些类型不含自由类型变量。按照惯例，我们将 A{B/X} 记为以下（预）类型：将预类型 A 中每个自由出现的 X 替换为类型 B 后所得的类型。下文中，我们令 κ遍历常量 0、1、 ⊥ 和 ⊤，令 ⋆ 遍历联结词 &、 ⊕、 ⅋ 和 ⊗，并令 σ遍历绑定子μ 和 ν。此外，我们称任何形如 σX .A 的类型为 σ-类型。

$$\frac{}{0 \mathbin{6} A}\ [\text{机器人}] \qquad \frac{}{\text{一个} \leqslant \top}\ [\text{置顶}] \qquad \frac{}{\kappa \leqslant \kappa}\ [\text{REFL}] \qquad \frac{A \mathbin{6} A' \qquad B \mathbin{6} B'}{A \star B \mathbin{6} A' \star B'}\ [\text{CONG}] \qquad \frac{A\{\sigma X.A/X\} \leqslant B}{\sigma X.A \mathbin{6} B}\ [\text{左-}\sigma] \qquad \frac{A \mathbin{6} B\{\sigma X.B/X\}}{A \leqslant \sigma X.B}\ [\text{右-}\sigma]$$

表 3：会话类型的子类型关系。

我们将类型上的标准子公式关系记为 $\preceq$。具体而言，关系 $\preceq$是类型上的最小预序，满足 $A \preceq \sigma X.A$ 以及 $A_i \preceq A_1 \star A_2$。例如，考虑 $A \triangleq \mu X.\nu Y.(1 \oplus X)$ 及其展开式 $A' \triangleq \nu Y.(1 \oplus A)$。我们有 $A \preceq 1 \oplus A \preceq A'$，因此 A 是 A' 的子公式。给定一组类型 $\mathscr{T}$，我们将 $\mathscr{T}$记为 $\mathscr{T}$中（若存在）按 $\preceq$关系取最小的类型。

表 3 展示了子类型判断的推理规则。这些规则采用余归纳地解释，因此，如果判断 $A \mathbin{6} B$ 是某个有限或无限推导的结论，那么它就是可推导的。规则 [BOT] 和 [TOP] 确立了 0 和 $\top$ 分别是最小和最大的会话类型；规则 [REFL] 和 [CONG] 确立了 6 关于所有常量和联结词的自反性与预同余；规则 [LEFT-σ] 和 [RIGHT-σ] 允许在 6 的任一侧展开不动点。

例 3.1. 考虑类型 $A \triangleq \mathbf{0} \oplus (\mathbf{1} \oplus \mathbf{0})$ 和 $B \triangleq \nu X.(\bot \,\&\, X)$，正如我们稍后将看到的，它们描述了例 2.1 中 Client 和 Server 的行为。我们可以推导出 $A \mathbin{6} B^{\perp}$ 以及 $B \mathbin{6} A^{\perp}$：

$$\dfrac{\dfrac{}{\mathbf{0} \leqslant \mathbf{1}}\ [\text{BOT}] \qquad \dfrac{\dfrac{\dfrac{}{\mathbf{1} \leqslant \mathbf{1}}\ [\text{REFL}] \qquad \dfrac{}{\mathbf{0} \leqslant B^{\perp}}\ [\text{BOT}]}{\mathbf{1} \oplus \mathbf{0} \leqslant \mathbf{1} \oplus B^{\perp}}\ [\text{CONG}]}{\mathbf{1} \oplus \mathbf{0} \leqslant B^{\perp}}\ [\text{RIGHT-}\mu]}{A \leqslant \mathbf{1} \oplus B^{\perp}}\ [\text{CONG}]}{A \leqslant B^{\perp}}\ [\text{RIGHT-}\mu] \qquad \dfrac{\dfrac{\dfrac{}{\bot \leqslant \bot}\ [\text{REFL}] \qquad \dfrac{}{B \leqslant \top}\ [\text{TOP}]}{\bot \parr B \leqslant \bot \parr \top}\ [\text{CONG}]}{B \leqslant \bot \,\&\, \top}\ [\text{LEFT-}\nu] \qquad \dfrac{}{\bot \leqslant \top}\ [\text{TOP}]}{\bot \parr B \leqslant A^{\perp}}\ [\text{CONG}]}{B \leqslant A^{\perp}}\ [\text{LEFT-}\nu]$$

规则 [LEFT-σ] 和 [RIGHT-σ] 可能看起来值得怀疑，因为无论 6 的直观含义如何，它们都可以应用于 6 的任一侧，而 μ和 ν 直观上分别表示最小不动点和最大不动点。事实上，如果子类型关系仅由根据表 3 中规则可推导这一条件来定义，那么这两个不动点算子将是等价的。例如，$\mu X.(\mathbf{1} \oplus X) \leqslant \nu X.(\mathbf{1} \oplus X)$ 和 $\nu X.(\mathbf{1} \oplus X) \leqslant \mu X.(\mathbf{1} \oplus X)$ 都可以推导出来，尽管只有第一个关系看起来合理。我们将在例 4.2 中看到，允许第二个关系实际上是不健全的，因为它会破坏良类型进程所具有的终止性质。按照以下（也是最终的）子类型关系定义，排除某些无限推导，从而得到健全的子类型关系。

定义 3.1（子类型关系）。如果 $A \mathbin{6} B$ 可导出，并且对于推导的每条无限分支 $(A_i \leqslant B_i)_{i \in \mathbb{N}}$，以下条件之一成立，则称 A 是 B 的子类型：(1) $\min\{C \mid \exists^{\infty} i : A_i = C\}$ 是一个 μ-类型；或 (2) $\min\{C \mid \exists^{\infty} i : B_i = C\}$ 是一个 ν-类型。下文中，$\exists^{\infty} i$ 表示存在无穷多个具有所述性质的 i。

定义 3.1 的条款 (1) 和 (2) 确保 μ 和 ν 被正确解释为最小不动点和最大不动点。特别是，我们预期最小不动点被最大不动点包含，但反之通常并不成立。例如，再次考虑前述子类型判断 $\mu X.(\mathbf{1} \oplus X) \leqslant \nu X.(\mathbf{1} \oplus X)$ 和 $\nu X.(\mathbf{1} \oplus X) \leqslant \mu X.(\mathbf{1} \oplus X)$ 对应的（简单直接的）推导。第一个推导满足这两个条款（只有一个无限分支，其中在 6 的左侧有一个 μ 类型被无限次展开，而在 6 的右侧有一个 ν 类型被无限次展开）。第二个推导两个条款都不满足。因此，$\mu X.(\mathbf{1} \oplus X)$ 是一个

$$[\text{CALL}]\ \frac{P \vdash \overline{x : A}}{\mathsf{A}\langle \overline{x} \rangle \vdash \overline{x : A}}\ \mathsf{A}(\overline{x}) \triangleq P \qquad [\text{SUB}]\ \frac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : B}{(x)(P \mid Q) \vdash \Gamma, \Delta}\ A \leqslant B^{\perp} \qquad [\top]\ \frac{}{\mathsf{fail}\ x \vdash \Gamma, x : \top} \qquad [\perp]\ \frac{P \vdash \Gamma}{x().P \vdash \Gamma, x : \perp}$$

$$[\mathbf{1}]\ \frac{}{x[] \vdash x : \mathbf{1}} \qquad [\parr]\ \frac{P \vdash \Gamma, y : A, x : B}{x(y).P \vdash \Gamma, x : A \parr B} \qquad [\otimes]\ \frac{P \vdash \Gamma, y : A \qquad Q \vdash \Delta, x : B}{x[y](P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B} \qquad [\&]\ \frac{P \vdash \Gamma, x : A \qquad Q \vdash \Gamma, x : B}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : A \,\&\, B}$$

$$[\oplus]\ \frac{P \vdash \Gamma, x : A_i}{x[\mathsf{in}_i].P \vdash \Gamma, x : A_0 \oplus A_1}\ i \in \{0, 1\} \qquad [\sigma]\ \frac{P \vdash \Gamma, x : A\{\sigma X.A/X\}}{P \vdash \Gamma, x : \sigma X.A}$$

表 4：μCP$^{\infty}$ 的类型规则。

$\nu X.(\mathbf{1} \oplus X)$ 的子类型，但 $\nu X.(\mathbf{1} \oplus X)$ 不是 $\mu X.(\mathbf{1} \oplus X)$ 的子类型。正如我们将在第 4 节中看到的，子类型关系 A 6 B 的应用可以明确地建模为一个进程：它消耗一个类型为 A 的通道，同时产生一个类型为 B 的通道。根据对子类型关系的这一解释，我们可以看到，定义 3.1 的条款 (1) 只是条款 (2) 的对偶版本。

定义 3.1 的两个子句都要求关系两侧的不动点类型由两侧无限次出现的类型中的 $\preceq$-最小值决定。这是正确处理交替不动点所必需的，因为这样可以确定哪个不动点正在对无限路径作出贡献。为了解其影响，考虑类型 $A \stackrel{\text{def}}{=} \mu X.\nu Y.(\mathbf{1} \oplus X)$、$A' \stackrel{\text{def}}{=} \nu Y.(\mathbf{1} \oplus A)$、$B \stackrel{\text{def}}{=} \mu X.\mu Y.(\mathbf{1} \oplus X)$ 和 $B' \stackrel{\text{def}}{=} \mu Y.(\mathbf{1} \oplus B)$。注意，A 展开为 A′，A′ 展开为 $\mathbf{1} \oplus A$，B 展开为 B′，而 B′ 展开为 $\mathbf{1} \oplus B$。尽管 Y 在左侧由最大不动点绑定、在右侧由最小不动点绑定，我们仍有 A 6 B。事实上，在 A 6 B 的推导的（唯一）无限分支中，A 和 A′ 都无限次出现，但 $A \preceq A'$，这符合直觉：无限次出现的 $\preceq$-最小类型对应于最外层不动点。在此例中，最外层不动点是 μX，它“覆盖”了内层不动点 νY 的贡献。感兴趣的读者可参阅有关 μMALL$^{\infty}$ [2, 6] 的文献，了解详情。

此后，除非另有说明，我们用 A 6 B 表示 A 是 B 的子类型，而不只是判断 A 6 B 可被推导。可以证明，6 是一个预序，且 A 6 B 蕴含 $B^{\perp} \leqslant A^{\perp}$。事实上，如例 3.1 所示，我们可以通过对偶化每个判断，并将每个 [LEFT-σ] 的应用（分别将 [RIGHT-σ]、[BOT]、[TOP] 的应用）转换为 [RIGHT-$\sigma^{\perp}$] 的应用（分别转换为 [LEFT-$\sigma^{\perp}$]、[TOP], [BOT]），从 A 6 B 的推导得到 $B^{\perp} \leqslant A^{\perp}$ 的推导。

4 类型规则

本节介绍 μCP$^{\infty}$ 的类型规则。类型判断的形式为 $P \vdash \Gamma$，其中 P 是一个进程，Γ 是一个类型上下文，即从信道到类型的有限映射。我们可以将此判断理解为：对于 Γ 的定义域中的信道，P 的行为符合 Γ 值域中的类型所描述的内容。我们用 dom(Γ) 表示 Γ 的定义域；用 $x : A$ 表示定义域为 {x}、将 x 映射到 A 的类型上下文；当 $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$ 时，用 Γ, Δ 表示 Γ 和 Δ 的并集。μCP$^{\infty}$ 的类型规则见表 4，除 [CALL] 和 [SUB], 外，这些规则都对应于证明

μMALL$^{\infty}$ [2, 6] 的规则，其中上下文就是待证明的序列式，而进程则是该证明（几乎）在语法上的表示。乘法/加法常项和联结词的规则都是标准的。规则 [σ] 无论不动点的性质如何，都会将其展开。规则 [CALL] 将进程调用展开为其定义，并检查调用和定义在同一上下文中是否类型正确。最后，[SUB] 检查组合 (x)(P |Q) 是否类型正确，条件是 A（P 关于 x 的行为）是 $B^{\perp}$（Q 关于 x 的行为）的子类型。从这个意义上说，[SUB] 嵌入了由 6 所引出的替换原则，因为它允许在需要行为符合 $B^{\perp}$ 的地方使用行为符合 A 的进程。请注意，由于 6 的自反性，μMALL$^{\infty}$ 的标准切割规则是 [SUB] 的一个特例。

如同 μMALL$^{\infty}$ 中，这些规则旨在以余归纳地方式解释，因此，如果存在一个任意的（有限或无限）推导，其结论为 $P \vdash \Gamma$，则判断 $P \vdash \Gamma$ 被视为可推导。

例 4.1。让我们展示例 2.1 中讨论的进程的类型推导。为此，令 $A_{\text{def}} = \mathbf{0} \oplus (\mathbf{1} \oplus \mathbf{0})$且$B_{\text{def}} = \nu X.(\perp \& X)$，并回顾例 3.1，A 6 $B^{\perp}$。我们推导：

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x[] \vdash \mathbf{1}}\,[\mathbf{1}]}{x[\mathsf{in}_0].x[] \vdash \mathbf{1} \oplus \mathbf{0}}\,[\oplus]}{x[\mathsf{in}_1].x[\mathsf{in}_0].x[] \vdash x : A}\,[\oplus]}{\mathsf{Client}\langle x \rangle \vdash x : A}\,[\text{CALL}] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{z[] \vdash z : \mathbf{1}}\,[\mathbf{1}]}{x().z[] \vdash x : \perp, z : \mathbf{1}}\,[\perp] \qquad \dfrac{\vdots}{\mathsf{Server}\langle x, z \rangle \vdash x : B, z : \mathbf{1}}}{\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : \perp \& B, z : \mathbf{1}}\,[\&]}{\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : B, z : \mathbf{1}}\,[\nu]}{\mathsf{Server}\langle x, z \rangle \vdash x : B, z : \mathbf{1}}\,[\text{CALL}]$$

$$\dfrac{\mathsf{Client}\langle x \rangle \vdash x : A \qquad \mathsf{Server}\langle x, z \rangle \vdash x : B, z : \mathbf{1}}{(x)(\mathsf{Client}\langle x \rangle \mid \mathsf{Server}\langle x, z \rangle) \vdash z : \mathbf{1}}\,[\text{SUB}]$$

通过交换 Client 和 Server，并使用关系 B 6 $A^{\perp}$，我们可以获得类似的类型推导。请注意，由于 $A \neq B^{\perp}$，因此无法使用标准 cut 直接组合 Client 和 Server。因此，在上述类型推导中使用子类型关系对于获得类型正确的组合非常重要。」

众所周知，并非每个 μMALL$^{\infty}$ 推导都是有效推导 [2, 6, 1]。为了刻画有效推导，我们需要一些辅助概念，下面将对这些概念进行回顾。

定义 4.1（线程）。设 $\gamma = (P_i \vdash \Gamma_i)_{i \in \mathbb{N}}$ 是类型推导中的一条无限分支，并回忆 $P_{i+1} \vdash \Gamma_{i+1}$ 是 $P_i \vdash \Gamma_i$的一个前提。γ的线程是一个信道序列 $(x_i)_{i \geq k}$，满足 $x_i \in \mathrm{dom}(\Gamma_i)$，并且对每个 $i \geq k$，以下条件之一成立：$x_i = x_{i+1}$，或 $P_i = x_i[x_{i+1}](P_{i+1} \mid Q)$，或 $P_i = x_i(x_{i+1}).P_{i+1}$。

直观地说，线程是一个无限信道名称序列 $(x_i)_{i \geq k}$，它从无限分支 $(P_i \vdash \Gamma_i)_{i \in \mathbb{N}}$ 中某个位置 k 开始取出，且属于同一个会话。例如，考虑例 4.1 中的推导，并注意到其中只有一条无限分支，即最右侧的分支。序列 (x,x,x . . .) 是一条线程，它从推导结论的正上方开始。

定义 4.2（ν-线程）。给定分支 $\gamma = (P_i \vdash \Gamma_i)_{i \in \mathbb{N}}$ 以及 γ的线程 $t = (x_i)_{i \geq k}$，我们记 $\inf(\gamma,t)_{\text{def}} = \{A \mid \exists^{\infty} i \geq k : \Gamma_i(x_i) = A\}$。如果 min inf(γ,t) 是一个 ν-类型，则称 t 是 γ 的 ν-线程。

给定一个分支 $\gamma = (P_i \vdash \Gamma_i)_{i \in \mathbb{N}}$ 以及 $= (x_i)_{i \geq k}$的一个线程 t，γ该线程确定了一个类型的无限序列 $(\Gamma_i(x_i))_{i \geq k}$。集合 inf(γ,t) 是该序列中无限次出现的类型所组成的集合，而 min inf(γ,t) 是这些类型中按 $\preceq$排序最小的类型（可以证明，任何集合 inf(γ,t) 中的最小值总是有定义的 [6]）。如果这个最小类型是 ν-类型，我们就称 t 为 ν-线程。在例 4.1 中，线程 $t = (x, x, x \ldots)$确定了类型序列 $(B, B \perp \& B, B \ldots)$，其中 B 和 $\perp \& B$ 都无限次出现。由于 $B \preceq \perp \& B$ 且 B 是 ν-类型，我们得出结论：t 是 ν-线程。

$$\left[\!\!\left[\frac{}{\mathbf{0} \leqslant A}\right]\!\!\right]_{x,y} \triangleq \mathsf{fail}\ x \qquad \left[\!\!\left[\frac{}{\mathbf{1} \leqslant \mathbf{1}}\right]\!\!\right]_{x,y} \triangleq x().y[]$$

$$\left[\!\!\left[\frac{\pi_1 :: A \leqslant A' \qquad \pi_2 :: B \leqslant B'}{A \oplus B \leqslant A' \oplus B'}\right]\!\!\right]_{x,y} \triangleq \mathsf{case}\ x\{y[\mathsf{in}_0].[\![\pi_1]\!]_{x,y}, y[\mathsf{in}_1].[\![\pi_2]\!]_{x,y}\}$$

$$\left[\!\!\left[\frac{\pi_1 :: A \leqslant A' \qquad \pi_2 :: B \leqslant B'}{A \otimes B \leqslant A' \otimes B'}\right]\!\!\right]_{x,y} \triangleq x(u).y[v]([\![\pi_1]\!]_{u,v} \mid [\![\pi_2]\!]_{x,y}) \quad (u \text{ and } v \text{ fresh})$$

$$\left[\!\!\left[\frac{\pi :: A\{\sigma X.A/X\} \leqslant B}{\sigma X.A \leqslant B}\right]\!\!\right]_{x,y} \triangleq [\![\pi]\!]_{x,y} \qquad \left[\!\!\left[\frac{\pi :: A \leqslant B\{\sigma X.B/X\}}{A \leqslant \sigma X.B}\right]\!\!\right]_{x,y} \triangleq [\![\pi]\!]_{x,y}$$

表5：子类型关系的强制转换语义（部分方程）。

定义4.3（有效分支）。设 $\gamma = (P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ 是类型推导的一条无限分支。如果存在 ν-线程 $(x_i)_{i\geq k}$ 属于 γ，使得 [ν] 被应用于无限多个 x_i，我们就称 γ有效。

定义4.3指出，如果一个分支包含一条 ν-线程，且其中无限次出现的 ν-类型也被无限次展开，那么该分支就是有效的。例 4.1便是如此：[ν] 规则被无限次应用，以展开 x 的类型。熟悉 μMALL$^\infty$ 文献的读者或许已经注意到，我们对有效分支的定义与标准定义 [2, 6] 有一处细微差异。在 μMALL$^\infty$ 中，只有当其中的 ν-线程并非“最终恒定”时，分支才有效；也就是说，定义 ν-线程的最大不动点必须被无限次展开。我们的有效分支概念满足这一条件，因为它要求无限次应用 [ν]，且这些应用涉及 ν-线程中的名称。现在我们可以定义有效类型推导的概念。

定义4.4（有效推导）。如果一个类型推导中的每个无限分支都是有效的，那么该类型推导就是有效的。

遵循 Pierce [22]，我们通过两个翻译函数为子类型关系提供强制转换语义：一个作用于子类型关系 A 6 B 的推导，另一个作用于使用子类型关系的类型推导 P ⊢ Γ。第一个翻译在表5中给出了（部分）定义。该翻译接受子类型关系 A 6 B 的一个推导 π ——记作 π:: A 6 B——并生成一个进程$[\![\pi]\!]_{x,y}$，将 A（所描述的协议）转换为 B（所描述的协议）。该翻译以信道 x 和 y 为参数，转换通过这两个信道进行：从 x“消费”协议 A，并在 y 上重新发出协议 B。表5列出了相当完整的一组情形，其余情形都是显而易见的变体。很容易证明，若 A 6 B，则 $[\![\pi]\!]_{x,y} \vdash x : A^{\perp}, y : B$。具体而言，考虑强制转换的类型推导中的一条无限分支 $\gamma \text{ def}= ([\![\pi_i]\!]_{x_i,y_i} \vdash x_i : A_i^{\perp}, y : B_i)_{i\in\mathbb{N}}$，其中 $A_0 = A$ 且 $B_0 = B$。这条分支对应于 π:: A 6 B 中的一条无限分支 $(A_i \leqslant B_i)_{i\in\mathbb{N}}$。根据定义 3.1，该分支满足条件 (1) 或条件 (2)。不失一般性，假设条件 (1) 成立。那么 $\min\{C \mid \exists^\infty i \in \mathbb{N} : A_i = C\}$ 是一个 μ-类型。根据表5，$(x_i)_{i\in\mathbb{N}}$ 是 γ 的一个 ν-线程，因此 γ 是一条有效分支。请注意，一般而言，$[\![\pi]\!]_{x,y}$ 是一个递归进程的（调用）。

关于类型推导的翻译，其定义如下列等式所示：

$$\left[\!\!\left[\frac{\pi_1 :: P \vdash \Gamma, x : A \qquad \pi_2 :: Q \vdash \Gamma, x : B}{(x)(P \mid Q) \vdash \Gamma}\right]\!\!\right] = \frac{\dfrac{[\![\pi_1\{y/x\}]\!] \qquad [\![\pi]\!]_{y,x} \vdash y : A^{\perp}, x : B}{(y)(P\{y/x\} \mid [\![\pi]\!]_{y,x}) \vdash \Gamma, x : B} \qquad [\![\pi_2]\!]}{(x)((y)(P\{y/x\} \mid [\![\pi]\!]_{y,x}) \mid Q) \vdash \Gamma} \tag{3}$$

其中 π:: A 6 $B^{\perp}$，并以同态方式扩展到表 4 中的所有其他类型规则。注意，(3) 将每次应用 [SUB] 转换为两次标准 μMALL$^{\infty}$ 切割规则的应用。由此得到的类型推导的有效性，立即可由原类型推导的有效性以及强制转换的有效性推出，正如前文所论证的那样。

得益于 μCP$^{\infty}$ 的类型规则与 μMALL$^{\infty}$ 之间的对应关系，类型良好的 μCP$^{\infty}$ 进程具有良好的行为。特别地，在单例上下文中类型良好的进程是无死锁的。

定理 4.1（无死锁性）。如果 P ⊢ x: A，则 P 是无死锁的。

此外，可以利用 μMALL$^{\infty}$ [2, 6] 的切割消去性质证明类型良好的 μCP$^{\infty}$ 进程会终止，这与相关系统 [16, 5] 中的情形类似。

定理 4.2（终止性）。如果 P ⊢ Γ那么 P 是终止的。

证明概要。对 P ⊢ Γ的类型推导（其中子类型强制转换已显式表示）可直接映射为一个有效的 μMALL$^{\infty}$ 证明。P 的每个归约步骤都可直接映射为 μMALL$^{\infty}$ 证明中的一个或多个主归约。每个进程归约可能对应多个主归约，其原因在于我们选择不使用显式的进程形式来触发不动点的展开（见 [σ]）。现在，假设 P 存在无限运行。那么，从 P 开始将存在一个无限归约步骤序列，因此相应的μMALL$^{\infty}$ 证明中将存在一个无限割归约序列，这与 [6,命题 3.5] 矛盾。因此，P 的每次运行都必须是有限的。 □

请注意，定理 4.2 仅保证类型良好的进程不会无限约简，并不一定保证进程的最终配置中不含受限会话。它们可能受进程中某个自由信道相关前缀的保护。结合定理 4.1 和定理 4.2，我们可以表述“成功终止”这一性质。

推论 4.1。如果 P ⊢ x :1，那么 P 最终会规约为 x[]。

我们以一个例子结束本节，该例说明定义 3.1 中的附加条款对于确保 6 是保持终止性的子类型关系至关重要。

例 4.2。 考虑一个退化的客户端 Chatter(x) ≙x[属于$_1$].Chatter⟨x⟩，它与例 2.1 中的 Server 进行无限交互，并令 C 定义为 = νX.(**1** ⊕ X)。该推导

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{Chatter}\langle x\rangle \vdash x : C}\,[\nu]}{x[\mathsf{in}_1].\mathsf{Chatter}\langle x\rangle \vdash x : \mathbf{1} \oplus C}\,[\oplus]}{\mathsf{Chatter}\langle x\rangle \vdash x : \mathbf{1} \oplus C}\,[\text{CALL}]}{\mathsf{Chatter}\langle x\rangle \vdash x : C}\,[\nu]$$

这是有效的，因为唯一的无限分支包含一个 ν-线程（x, x . . .），沿着该分支我们会发现[ν] 的无限多次应用。如果我们允许关系C 6 $B^{\perp}$ （参见引出定义3.1 的讨论），则组合 (x)(Chatter⟨x⟩ | Server⟨x,z⟩) 将是良类型的，并且良类型进程终止这一点将不再成立，因为Chatter 与 Server 之间的交互会永远持续下去。」

5 结语

我们将会话类型的子类型关系定义为一种预序关系，它不受递归类型展开与折叠的影响，并且使 0 和 ⊤ 分别成为最小元素和最大元素。尽管该关系看起来极为简约，而且类型的语法似乎十分僵硬——内部选择与外部选择的元数都是固定的——但得益于赋予 0 和 ⊤ 的解释，6 捕捉到了标签通常所具有的协变／逆变性质。过去也有人研究过具有最小元素和最大元素的会话类型的其他精化关系 [19, 21], ，但没有明确建立它们与逻辑之间的对应关系。

不同于仅保留会话安全属性（通信安全、协议保真性和无死锁性）的会话类型子类型关系 [9, 4, 17, 10] ，6 还保留了终止性这一活性属性。因此，6 与公平子类型关系 [20, 21], 存在一定关联，后者保留公平终止 [11, 7]。6 似乎比公平子类型关系更宽松，不过，由于类型语法中表示递归行为的方式存在根本差异，很难准确刻画这两种关系之间的确切联系。本文定义的子类型关系继承了 μMALL$^{\infty}$ [2, 6]中的最小不动点和最大不动点，而公平子类型关系则是在会话类型语言上进行研究的：这些语言要么使用一般递归 [20]，要么直接使用正规树 [21]。更具结论性的比较留待未来工作完成。

本工作对不动点的处理与一种相关的会话子类型关系逻辑方法之间的一个关键区别在于，尽管两者都能保证无死锁性，当前方法还保证终止性。关于不动点设计的见解应当能够独立于任何逻辑解释，应用于其他会话演算。特别是，依据定义 3.1 研究异步会话类型的子类型关系将会很有意思。通过采用合适的强制转换语义，使消息能够像在简单编排器中那样缓冲，即可开展这项研究。

致谢。我们感谢匿名审稿人提出的宝贵意见。

参考文献

[1] David Baelde、Amina Doumane、Denis Kuperberg 和 Alexis Saurin（2022）：循环与非良基证明中的线程反弹：迈向循环证明的组合性。载于 Christel Baier 和 Dana Fisman（编）：LICS ’22：第 37 届 ACM/IEEE 计算机科学逻辑年度研讨会，海法，以色列，2022 年 8 月 2—5 日，计算机协会，第 63:1–63:13 页，doi:10.1145/3531130.3533375。[2] David Baelde、Amina Doumane 和 Alexis Saurin（2016）：无穷证明论：乘法加法情形。载于 Jean-Marc Talbot 和 Laurent Regnier（编）：第 25 届 EACSL 计算机科学逻辑年度会议，CSL 2016，2016 年 8 月 29 日至 9 月 1 日，法国马赛，LIPIcs 62，Schloss Dagstuhl - Leibniz 信息学中心，第 42:1–42:17 页，doi:10.4230/LIPIcs.CSL.2016.42。[3] Lu´ıs Caires、Frank Pfenning 和 Bernardo Toninho（2016）：作为会话类型的线性逻辑命题。《数学结构与计算机科学》26(3)，第 367–423 页，doi:10.1017/S0960129514000218。[4] Giuseppe Castagna、Mariangiola Dezani-Ciancaglini、Elena Giachino 和 Luca Padovani（2009）：会话类型基础。载于 Ant´onio Porto 和 Francisco Javier L´opez-Fraguas（编）：第 11 届 ACM SIGPLAN 声明式编程原理与实践国际会议论文集，2009 年 9 月 7—9 日，葡萄牙科英布拉，计算机协会，第 219–230 页，doi:10.1145/1599410.1599437。[5] Farzaneh Derakhshan 和 Frank Pfenning（2022）：作为会话类型进程的循环证明：局部有效性条件。《计算机科学逻辑方法》18 卷第 2 期，doi:10.46298/lmcs-18(2:8)2022。

[6] Amina Doumane（2017）：不动点逻辑的无穷证明论。（不动点逻辑的无穷证明论）。博士论文，法国巴黎狄德罗大学。可访问：https://tel.archives-ouvertes.fr/tel-01676953。[7]Nissim Francez（1986）：公平性。《计算机科学专著》，施普林格，doi:10.1007/ 978-1-4612-4886-6。[8] Simon J. Gay（2016）：子类型关系支持安全的会话替换。载于 Sam Lindley、Conor McBride、Philip W. Trinder 和 Donald Sannella（编）：《改变世界的成功之道——献给 Philip Wadler 六十华诞的论文集》，《计算机科学讲义》9600，施普林格，第 95–108 页，doi:10.1007/978-3-319-30936-1_5。[9] Simon J. Gay 和 Malcolm Hole（2005）：π 演算中会话类型的子类型关系。《Acta Informatica》42(2-3)，第 191–225 页，doi:10.1007/s00236-005-0177-z。[10] Silvia Ghilezan、Jovanka Pantovi´c、Ivan Proki´c、Alceste Scalas 和 Nobuko Yoshida（2022）：异步多方会话的精确子类型关系。《ACM计算逻辑汇刊》，doi:10.1145/3568422。已正式接收。[11] Orna Grumberg、Nissim Francez 和 Shmuel Katz（1984）：通信进程的公平终止。载于：《第三届 ACM 分布式计算原理年度研讨会论文集》，PODC ’84，计算机协会，美国纽约州纽约市，第 254–265 页，doi:10.1145/800222.806752。[12] Kohei Honda（1993）：二元交互的类型。载于 Eike Best（编）：CONCUR ’93：第 4 届并发理论国际会议，德国希尔德斯海姆，1993 年 8 月 23—26 日，论文集，《计算机科学讲义》715，施普林格，第 509–523 页，doi:10.1007/3-540-57208-2_35。[13] Kohei Honda、Vasco Thudichum Vasconcelos 和 Makoto Kubo（1998）：基于结构化通信的编程的语言原语与类型纪律。载于 Chris Hankin（编）：《编程语言与系统——ESOP’98：第 7 届欧洲编程研讨会，葡萄牙里斯本，1998 年 3 月 28 日至 4 月 4 日》，《计算机科学讲义》1381，施普林格，第 122–138 页，doi:10.1007/BFb0053567。[14] Ross Horne（2020）：使用循环序贯式实现会话子类型关系与多方兼容性。载于 Igor Konnov 和 Laura Kov´acs（编）：第 31 届并发理论国际会议，CONCUR 2020，2020 年 9 月 1—4 日，奥地利维也纳（线上会议），LIPIcs 171，Schloss Dagstuhl - Leibniz 信息学中心，第 12:1–12:22 页，doi:10.4230/LIPIcs.CONCUR.2020.12。[15] Hans H¨uttel、Ivan Lanese、Vasco T. Vasconcelos、Lu´ís Caires、Marco Carbone、Pierre-Malo Deni´elou、Dimitris Mostrous、Luca Padovani、Ant´onio Ravara、Emilio Tuosto、Hugo Torres Vieira 和 Gianluigi Zavattaro（2016）：会话类型与行为契约基础。《ACM计算调查》49(1)，第 3:1–3:36 页，doi:10.1145/2873052。[16] Sam Lindley 和 J. Garrett Morris（2016）：谈论香蕉：会话类型的结构递归。载于 Jacques Garrigue、Gabriele Keller 和 Eijiro Sumii（编）：第 21 届 ACM SIGPLAN 函数式编程国际会议论文集，ICFP 2016，日本奈良，2016 年 9 月 18—22 日，计算机协会，第 434–447 页，doi:10.1145/2951913.2951921。[17] Dimitris Mostrous 和 Nobuko Yoshida（2015）：高阶 π-演算的会话类型与异步子类型关系。《信息与计算》241，第 227–263 页，doi:10.1016/j.ic.2015.02.002。[18] Luca Padovani（2010）：基于契约的 Web 服务发现及其简单编排器。《理论计算机科学》411(37)，第 3328–3347 页，doi:10.1016/j.tcs.2010.05.002。[19] Luca Padovani（2010）：会话类型 = 交类型 + 并类型。载于 Elaine Pimentel、Betti Venneri 和 Joe B. Wells（编）：第五届交类型及相关系统研讨会论文集，ITRS 2010，英国爱丁堡，2010 年 7 月 9 日，EPTCS 45，第 71–89 页，doi:10.4204/EPTCS.45.6。[20] Luca Padovani（2013）：开放会话类型的公平子类型关系。载于 Fedor V. Fomin、Rusins Freivalds、Marta Z. Kwiatkowska 和 David Peleg（编）：《自动机、语言与编程——第 40 届国际研讨会，ICALP 2013，拉脱维亚里加，2013 年 7 月 8—12 日，论文集，第二部分》，《计算机科学讲义》7966，施普林格，第 373–384 页，doi:10.1007/978-3-642-39212-2_34。[21] Luca Padovani（2016）：多方会话类型的公平子类型关系。《数学结构与计算机科学》26(3)，第 424–464 页，doi:10.1017/S096012951400022X。

[22] Benjamin C. Pierce（2002）：类型与程序设计语言。麻省理工学院出版社。[23] Philip Wadler（2014）：作为会话的命题。《函数式编程杂志》24(2-3)，第384–418页，doi:10.1017/ S095679681400001X。