鲁棒灰箱演算的类型系统和代数性质研究

Type system and algebraic theory of Robust Ambients

博士生：管旭东

导师：尤晋元

上海交通大学计算机科学与工程系

（博士论文草稿，最后修改日期： 2002 年 4 月 11 日 ）

鲁棒灰箱演算的类型系统和代数性质研究

摘 要

灰箱演算是一种用于描述移动系统及其交互的形式化方法。本文以灰箱演算的一个变体——鲁棒灰箱演算（ ROAM ）作为研究对象，从操作语义、类型系统、进程等价性和表达能力等方面对其进行了研究。

首先，本文在分析现有移动灰箱演算（ MA ）及安全灰箱演算（ SA ）不足的基础上，提出通过限制协动作使用对象来提高安全性的思想，并以此为指导给出 ROAM 的语法和归约语义。

其次，本文研究了 ROAM 的类型问题，提出一套可以解决灰箱演算中存在的类型演化问题的类型系统 ETS-MT 。在众多灰箱演算类型系统的研究中，类型的演化问题一直没有得到妥善地解决，灰箱在被打开前后的类型变化得不到体现。本文利用特殊的类型表达式区分进程类型表达式中的 **当前类型** 和 **未来类型** 部分，分别表示打开操作前后的不同类型特性，配合特殊的能力类型构造，解决了灰箱演算的类型演化问题。ETS-MT可以有效地跟踪进程的移动性和线程数，并支持子类型关系。

再次，本文参照 Gordon 和 Cardelli 研究 MA 等价性使用的方法，对 ROAM 进行了类似的研究。首先引入 ROAM 进程的硬化关系，用于将进程中可能参与归约的部分与剩余的部分分开。在硬化关系的基础上，按照进程的归约特性给出 ROAM 的标号转移语义，逐一分析这些规则与归约规则的对应关系，并证明两套语义的等价性。在标号转移语义的基础上，研究了判定进程上下文等价的一般方法，得出 ROAM 进程的上下文等价关系与特殊的简单上下文等价关系是同一关系的结论，并给出 ROAM 进程与简单上下文发生交互的各种可能，作为判定进程等价的一般性结论。

之后，本文综合类型系统和等价性判定的结论，给出用于判定进程等价的一些代数定律，包括：无上下文条件的单线程定律、有上下文条件的单线程定律、定点接收定律和一些辅助结论。同时使用这些定律证明灰箱换名例子和防火墙跨越例子在给定条件下的正确性。这些例子不仅示意了等价性定律的具体使用方法，而且通过前提条件的减弱，体现了 ROAM 较之 MA 和 SA 在安全性方面的优越性。

最后，本文利用上述等价性定律给出并证明了纯 ROAM 演算翻译 π 演算的一个方案，有力地说明 ROAM 在提高安全性的同时，没有失去灰箱演算固有的表达能力。

本文的主要贡献在于以下一些方面。

(一) 对利用协动作参数加强灰箱演算安全性这一命题进行了研究，提出了灰箱演算的变体 ROAM 。通过利用协动作参数加强交互双方的彼此控制， ROAM 在安全控制方面比起 SA 来有一定优势。同时，对协动作参数进行控制后， ROAM 并没有丧失 SA 所具有的表达能力。

(二) 针对灰箱演算中存在的类型演化问题给出了支持移动性和线程数的演化类型系统 ETS-MT 。ETS-MT通过区分进程的当前类型和未来类型, 可以精确地刻划进程的移动性和线程数, 并支持子类型关系。考虑到类型系统中所刻划属性的增减, ETS-MT在设计时还专门引入准类型集合的概念, 刻划不同的类型属性, 可以有不同的准类型集合。

(三) 将前人对灰箱演算语义、等价性、表达能力等方面的研究成果进行了贯穿和融合, 集中体现在两方面。首先, 有别于 Levi 和 Sangiorgi 使用的互模拟方法, 本文用 Gordon 和 Cardelli 基于硬化关系所得的简单上下文等价性一般判定方法 (context lemma) 证明了进程的等价性定律; 其次, 本文用进程的等价性定律完成对 π 演算翻译的代数方法证明, 较之 Zimmer 的证明方法来得更为简明。

关键词 计算理论, 进程代数, 类型系统, 灰箱演算

TYPE SYSTEM AND ALGEBRAIC THEORY OF ROBUST AMBIENTS

ABSTRACT

The ambient calculus is a formal method modelling mobile systems and their interactions. This thesis mainly focuses on the robust ambient calculus (ROAM) — a variation of the ambient calculus. Fundamental research on the operational semantics, type theory, behaviour equivalence and expressiveness of ROAM are carried out in it.

Firstly, by analysing the inadequacies found in the existing mobile ambient calculus (MA) and in the safe ambient calculus (SA), an approach that can improve those inadequacies by utilizing the paramenters of co-actions is proposed. Based on this, the syntax and reduction semantics of ROAM are given.

Secondly, the type system of ROAM is studied. The type evolution problem calls for a more precise way to trace process behavior in type system for the ambient calculi. It has not been fully addressed by many previous works. In this thesis, a type system named `ETS-MT` is proposed to fully solve the problem. By storing both the current type and the future type in process type expressions, and introducing special type syntax for capabilties, `ETS-MT` can track both the mobilty and threads of a corresponding process, with full support for type evolution.

Thirdly, following Gordon and Cardelli's way in MA equivalence research, some process equivalence properties of ROAM are studied. By introducing a hardening relation that seperated the active part of a process from the residue, a labelled transition semantics is given and proved sound with regard to the reduction semantics discussed before. Then, based on a simplified context called harness, two results for proving the equivalence relations of processes are given, a context lemma that equates the context equivalence relation with the harness equivalence relation, and an activity lemma that enumerates the ways a process may interact with any given harness.

Fourthly, based on previous results on type system and process equivalences, some algebraic laws for process equivalence are given, including a few single-threadness laws with and without contextual constraints, a few uniform receptiveness laws, and a few corollaries. To demonstrate their usages, the renaming example and the firewall-crossing example are proven through them, under certain constraints. With simpler encoding and looser constraints, these examples illustrate the advanced security features provided by ROAM.

Finally, the encoding of the π-calculus into pure ROAM is given and the operational correspondence proved using the algebraic laws. It shows that by limiting the parameters of co-actions in reduction, ROAM still holds the strong expressive power of its ancestors.

The main contributions of the thesis are summarized below.

The research on ROAM here shows that by further restricting the parameters of co-actions, ROAM has better security control over SA, as is demonstrated by the renaming example and the firewall-crossing example. Moreover, up-to-now, there is no evidence that ROAM is less expressive than SA.

The type system `ETS-MT` provides a concise framework for the type evolution problem. It has full subtyping support for both mobility and threads. Moreover, the basic structure supporting type evolution can be extended with other type information, and the idea behind `ETS-MT` can be easily adopted by other ambient calculi with co-actions.

This thesis also provides new proof method for equational laws, through Gordon and Cardelli's context lemma. Moreover, the operational correspondence proof of the π_{esc} encoding is algebraic, while the original proof by Zimmer is somewhat more complex.

KEY WORDS theory of computation, process algebra, type system, ambient calculus

目 录

第一章 概述 **1**

§1.1 灰箱演算 . . . 1

§1.2 论文的主要内容和结构安排 . . . 4

§1.3 与本论文相关的研究工作 . . . 6

§1.3.1 移动计算的形式化模型 . . . 7

§1.3.2 以 MA 为框架的相关工作 . . . 8

§1.3.3 以 SA 为框架的相关工作 . . . 11

§1.3.4 其它基于灰箱演算的相关工作 . . . 13

第二章 鲁棒灰箱演算 **15**

§2.1 语法 . . . 15

§2.2 归约语义 . . . 17

§2.3 举例 . . . 19

§2.3.1 进程同步 . . . 19

§2.3.2 进程竞争 . . . 19

§2.3.3 换名 . . . 20

第三章 演化类型系统 **23**

§3.1 类型语法 . . . 24

§3.2 类型运算符 . . . 25

§3.3 子类型关系 . . . 27

§3.4 类型判定规则 . . . 29

§3.5 类型判定规则的归约一致性证明 . . . 31

§3.6 最小类型算法 . . . 32

§3.7 举例 . . . 34

第四章 标号转移语义 **37**

§4.1 硬化关系 . . . 37

§4.2 基于硬化关系的标号转移语义 . . . 38

§4.3 归约语义和标号转移语义的等价性 . . . 39

第五章 等价性判定 **45**

§5.1 上下文等价 . . . 45

§5.2 简单上下文等价及其判定 . . . 47
§5.2.1 简单上下文等价 . . . 47
§5.2.2 简单上下文等价的判定 . . . 48
§5.3 两种等价关系间的联系 . . . 54

第六章 等价性定律 **59**
§6.1 具有类型约束的进程等价 . . . 59
§6.2 无上下文条件的单线程定律 . . . 62
§6.3 有上下文条件的单线程定律 . . . 63
§6.4 定点接收定律 . . . 67
§6.5 等价性定律应用举例 . . . 70
§6.5.1 换名 . . . 70
§6.5.2 防火墙跨越 . . . 72

第七章 π 演算的翻译 **77**
§7.1 π 演算 . . . 77
§7.1.1 语法 . . . 77
§7.1.2 归约语义 . . . 78
§7.2 π_{esc} 演算 . . . 79
§7.2.1 树状替换 . . . 79
§7.2.2 语法 . . . 79
§7.2.3 结构同余关系 . . . 80
§7.2.4 归约规则 . . . 80
§7.2.5 合法进程 . . . 81
§7.2.6 通道闭包 . . . 82
§7.2.7 π 演算和 π_{esc} 演算的关系 . . . 83
§7.3 π_{esc} 演算的翻译 . . . 83
§7.3.1 翻译过程 . . . 83
§7.3.2 类型方案 . . . 85
§7.3.3 进程在翻译环境下的等价性 . . . 86
§7.3.4 翻译方案的正确性证明 . . . 87
§7.4 π 演算的翻译 . . . 92

第八章 结语 **95**
§8.1 论文工作的创新性 . . . 95

§8.2 进一步的工作 96

参考文献 **99**

附录 1　引理 3.8 和定理 3.9 的证明 **103**

附录 2　引理 4.14 的证明 **111**

附录 3　引理 5.31 的证明 **117**

文中用到的符号及含义

符号	中文含义	英文含义	定义页码
P, Q, $\cdots$	进程	process	15
n, m, $\cdots$	灰箱名	ambient name	15
M, N, $\cdots$	能力	capability	15
A	动作	action	17
$\equiv$	结构同余关系	structural congruence relation	17
$\longrightarrow$	归约关系	reduction relation	18
T	进程类型	process type	24
W	能力类型	capability type	24
U	准类型	pre-type	24
Z	移动属性	mobility	24
Y	线程数属性	threads	24
Γ	类型环境	type environment	29
C, D, $\cdots$	硬化结果	concretion	37
$>$	硬化关系	hardening relation	38
α	标号	label	38
$\stackrel{\alpha}{\longrightarrow}$	标号转移关系	labelled transition relation	38
$\mathcal{C}()$	上下文	context	45
$\downarrow_n$, $\Downarrow_n$	观察结论	observation, or barb	45
$\approx$	上下文等价	contextual equivalence	46
H	简单上下文	harness	47
$\simeq$	简单上下文等价	harness equivalence	48
$n^{[A]}$	深层动作	deep action	61
$P \stackrel{n^{[A]}}{\Longrightarrow}$, $P \stackrel{n^{[A]}}{\not\Longrightarrow}$	深层动作观察	deep action observation	62
$\not\subset$, $\not\sqsubset$	不在上下文中出现	never occur in context	63
μ, L	可见动作及可见动作集合	visible action	63
$\downarrow_n^L$, $\Downarrow_n^L$	有上下文约束的观察结论	observation under restricted context	63
$\hat{\equiv}$	活体结构同余关系	structural congruence relation on prime	111

第一章　概述

移动计算一词虽然并不陌生，但要给出其精确定义却有一定难度。即便限定在学术研究范畴，对移动计算的研究也至少涵盖操作系统（ operating system ）、硬件支持（如：有线和无线通讯网）、程序设计语言（ programming language ）等研究领域。同时，由于研究的侧重点不同，对移动计算中移动性一词的理解也存在两种完全不同的涵义。一方面，对于熟知无线通讯网络和通讯协议的研究人员来说，移动一词通常表示物理意义上的移动，即：现实世界中计算设备的移动，其应用领域包括个人数字助理（ PDA - personal digital assitant ）和全球定位系统（ GPS - global positioning system ）、便携式计算设备及其支撑技术等等。另一方面，在程序设计语言和操作系统（尤其是分布式和并行计算）研究领域，移动一词通常意味着软件意义上的移动，其代表为研究代码在计算设备上迁移的移动代理（ mobile agent ）技术 [1, 2, 3, 4] 。由于移动性存在以上两种不同的内涵，在英文中通常将前者称为 mobile computing ，将后者称为 mobile computation[5] 。

随着移动计算的不断发展，对其形式化基础的研究 [1] 也在逐步展开。众所周知，一个精简、恰当的形式化模型对复杂的上层编程语言及应用系统的构造和正确性验证具有指导性意义。 λ 演算 [6] 与顺序执行程序、 CCS[7] 和 π 演算 [8] 等进程代数（ process algebra ）与并发执行程序的关系无不说明了这一点。作为典型的并发模型， π 演算可以很好的描述进程间基于通道的通讯，甚至进程在通道中的移动 [9] ，但作为描述移动计算方面的形式化模型，尤其是描述移动计算中普遍存在的进程和资源的位置分布、进程在不同位置间的移动以及由此带来的安全性问题，一般意义上的 π 演算并不适合 [10] 。移动计算需要新的形式化模型来支持，以解决其语言设计、程序构造、安全性方面的一系列问题。

目前，对移动计算形式化方面的研究有：以 π 演算 [8] 为基础，在 π 演算基础之上增加的表示进程和资源分布构造的一些变体 [11, 12, 13] ；在 λ 演算 [6] 基础上增加代码移动原语的 λdist 演算 [14] ；在 join 演算 [15] 基础上增加树状位置分布和位置失效（ failure ）概念的分布式 join 演算；以及专门用于描述移动计算模型及其安全性的灰箱 [2] （ ambient ）演算 [5, 16] 和 seal 演算 [17] 。本文的研究工作主要围绕灰箱演算的一个变体展开。

§1.1　灰箱演算

灰箱演算是 1998 年由 Cardelli 和 Gordon 提出的一种移动计算形式化模型。基于广域网络中资源和进程的树状嵌套结构，灰箱演算中的核心概念——灰箱（ ambient ）体现为一个有界的、树状嵌套的计算场所。例如，在灰箱演算中，可用以下进程表达式表示两个计算场所：计算机系（ cs ）和图书馆（ lib ）。同时，在计算机系中还存在一个内部计算场所（ $laptop$ ）：

$$cs[laptop[P] \mid Q] \mid lib[R] \tag{1.1}$$

[1] 在人工智能、软件工程等领域的也有对移动计算形式化方法的研究，这里及下文涉及的形式化特指程序设计语言领域的形式化。

[2] 关于 ambient 的译名，本文用“灰箱”一词：箱字取自其整体移动性，灰字取自其内部结构部分可见、部分不可见之意——发生归约的部分可见，其余不可见。采用直译译名“环境”一词不能很好的表达 ambient 作为名词代表移动的软硬件对象这层含义。

这里每对“ [ ] ”表示一个灰箱，每个灰箱前有一个名字，用于标识这个灰箱。符号“ | ”在并发模型中表示进程的并发运行，在灰箱演算中，还表示资源（灰箱）的并置，大写字母 P 、 Q 、 R 等代表任意进程表达式。灰箱演算中，计算的移动性主要体现为灰箱的移动。下面的归约步骤（符号“ $\longrightarrow$ ”表示进程的归约关系，可理解为计算的进行）示意了计算设备 $laptop$ 从 cs 迁移到 lib 的过程：

$$cs[laptop[\mathtt{out}\ cs\ .\ \mathtt{in}\ lib\ .\ P] \mid Q] \mid lib[R] \tag{1.2}$$

$$\longrightarrow \quad cs[Q] \mid laptop[\mathtt{in}\ lib\ .\ P] \mid lib[R] \tag{1.3}$$

$$\longrightarrow \quad cs[Q] \mid lib[laptop[P] \mid R] \tag{1.4}$$

这里第一个归约操作使用了灰箱演算中的移动原语 `out` ，导致灰箱 $laptop$ 从 cs 中移出；第二个归约操作使用了移动原语 `in` ，其结果为灰箱 $laptop$ 进入 lib 。灰箱演算中仅这两个移动原语即可使灰箱的嵌套结构发生任意变形。灰箱演算中还有第三个原语 `open` ，用于消去一个灰箱的边界。下面的归约示意了一个 `open` 动作的执行过程。

$$\mathtt{open}\ n\ .\ P \mid n[Q] \quad \longrightarrow \quad P \mid Q \tag{1.5}$$

灰箱的树状嵌套模型和三个动作原语 `in` 、 `out` 和 `open` 体现了灰箱演算的核心内容。无论是物理计算设备的移动还是软件代码的移动，都被抽象为灰箱的移动。由于该模型直观、精简，并且拥有惊人的表达能力（仅这三个基本动作就可以翻译同步 π 演算 [18] ），这使得灰箱演算的研究成为移动计算形式化方向一个非常活跃的领域。

最初由 Cardelli 和 Gordon 提出的灰箱演算称为移动灰箱（ mobile ambient ）演算（以下简称 MA ） [5] 。随着研究的不断深入， MA 也暴露出一些设计上的不足。 Levi 和 Sangiorgi 在文献 [16] 中首次指出 MA 在语法上存在强干扰（ grave interference ）问题——由于灰箱原语动作的单方参与性，归约顺序的不同将会导致完全不同的结果。例如以下的 MA 进程表达式：

$$h[\,] \mid n[\,\mathtt{in}\ h \mid m[\mathtt{out}\ n\ .\ P]\,] \tag{1.6}$$

可归约为（首先灰箱 n 经 `in` h 动作进入 h ，之后灰箱 m 经 `out` n 动作移出 n ，即：先 `in` 后 `out` ）：

$$\longrightarrow \quad h[n[m[\mathtt{out}\ n\ .\ P]]] \quad \longrightarrow \quad h[n[\,] \mid m[P]] \tag{1.7}$$

也可归约为（先 `out` 后 `in` ）：

$$\longrightarrow \quad h[\,] \mid n[\,\mathtt{in}\ h] \mid m[P] \quad \longrightarrow \quad h[n[\,]] \mid m[P] \tag{1.8}$$

虽然并发和分布模型中进程的归约不再具有顺序程序执行时的确定性，但是在以上两种归约结果中，不确定性却超出了可以解决的范围：进程 $m[P]$ 由于无法知道自己的确切位置，从而影响后续的归约（必然有一种情况使 $m[P]$ 无法顺利进行后续的计算）。更严重者，可能发生这种强干扰的进程很难使用类型系统（ type system ）来识别。上述强干扰问题使得 MA 的代数理论（ algebraic theory ）比较薄弱、进程的等价性不易判定，导致程序的正确性很难验证。

在指出强干扰问题的同时， Levi 和 Sangiorgi 还提出了一个保留 MA 基本思想，但可以控制强干扰问题的安全灰箱（ safe ambient ）演算（以下简称 SA ）。 SA 修改了 MA 的语法，针对 MA

动作的单方参与性，为每个动作原语增加了一个相应的协动作（ co-action ）原语。例如，原先式 (1.2) 的迁移例子在 SA 中可写作（花括弧表示参与归约的动作和协动作）：

$$cs[laptop[\underbrace{\mathtt{out}\ cs}\ .\ \mathtt{in}\ lib\ .\ P] \mid \underbrace{\overline{\mathtt{out}}\ cs}\ .\ Q] \mid lib[\overline{\mathtt{in}}\ lib\ .\ R] \tag{1.9}$$

$$\longrightarrow \quad cs[Q] \mid laptop[\underbrace{\mathtt{in}\ lib}\ .\ P] \mid lib[\underbrace{\overline{\mathtt{in}}\ lib}\ .\ R] \tag{1.10}$$

$$\longrightarrow \quad cs[Q] \mid lib[laptop[P] \mid R] \tag{1.11}$$

打开动作在 SA 中变为：

$$\mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ n\ .\ Q] \quad \longrightarrow \quad P \mid Q \tag{1.12}$$

通过以上修改，原先归约的单方参与性变成了通过正协动作互相同步的双方参与性。这样，在上述存在强干扰问题的 MA 表达式 (1.6) 中，灰箱 n 就可以通过对 m 何时离开进行控制。假设 n 希望在 m 离开后才进入 h ，则对应的 SA 表达式可写为：

$$h[\overline{\mathtt{in}}\ h] \mid n[\overline{\mathtt{out}}\ n\ .\ \mathtt{in}\ h \mid m[\mathtt{out}\ n\ .\ P]] \tag{1.13}$$

由于 $\mathtt{in}\ h$ 动作必须在 m 离开 n 后才可能得到运行，该表达式可以保证先 out 后 in 的执行次序。反之，如果 n 希望先进入 h ，则可将式中的 $\overline{\mathtt{out}}\ n$ 和 $\mathtt{in}\ h$ 改变次序，写为 $\mathtt{in}\ h\ .\ \overline{\mathtt{out}}\ n$ 。可以看出， MA 中的强干扰问题在 SA 中可以从语法角度直接进行控制。当然，如果上式中顺序执行的两个动作 [3] $\overline{\mathtt{out}}\ n$ 和 $\mathtt{in}\ h$ 写为并发执行的表示方式 $\overline{\mathtt{out}}\ n \mid \mathtt{in}\ h$ ，则 m 的最终位置仍然无法确定，但是这可以看作是一种程序编写错误，而不是 SA 中固有的问题，而且这种错误还可以从类型角度进行静态检查。文献 [16] 同时提出了具有这种功能的单线程（ single-threaded ）类型系统。一个单线程进程只允许有一个可以发生归约的能力，而且这种单线程特性在归约过程中将保持不变。

文献 [16] 中解决强干扰问题的关键在于引入了与三个动作原语对应的协动作，使归约的执行由单方参与性变为双方参与性。在语法和操作语义的取舍上， SA 将协动作的参数统一设置为包含该协动作的灰箱的名字。这样设置的一个优点在于 SA 可以描述任何 MA 可以描述的移动系统：只要在 MA 表达式的每个灰箱中增加进程 $!\overline{\mathtt{in}}\ n \mid !\overline{\mathtt{out}}\ n \mid !\overline{\mathtt{open}}\ n$ （假设该灰箱的名字为 n ）就可转化成等价的 SA 表达式。但是这种设置却使得 SA 存在以下一些问题。

- **安全性控制比较薄弱**：协动作容易被第三方挪用，例如以下的两个 SA 表达式：

$$n[\mathtt{in}\ m\ .\ \overline{\mathtt{open}}\ n\ .\ P] \mid m[\overline{\mathtt{in}}\ m\ .\ \mathtt{open}\ n\ .\ Q] \mid h[\mathtt{in}\ m] \tag{1.14}$$

$$m[\overline{\mathtt{out}}\ m\ .\ \mathtt{in}\ n.P \mid n[\mathtt{out}\ m\ .\ \overline{\mathtt{in}}\ n\ .\ Q] \mid h[\mathtt{out}\ m]] \tag{1.15}$$

  式 (1.14) 中， n 希望进入 m 后被打开，同样 m 等待 n 的进入并对其执行打开操作。但是，这个协议很容易被与 n 同层的灰箱 h 所破坏：只要 h 也发出一个进入 m 的请求， m 中的 $\overline{\mathtt{in}}\ m$ 动作就有可能先同 h 中的 $\mathtt{in}\ m$ 动作发生归约并让 h 进入，从而无法正常完成打开 n 的协议。式 (1.15) 中也存在类似的问题。

- **引入不必要的二义性**：同样考虑式 (1.14) ，在灰箱演算中打开一个进入灰箱的构造使用非常广泛，且往往具有类似 $m[\overline{\mathtt{in}}\ m\ .\ \mathtt{open}\ n\ .\ Q]$ 的结构，此时在设计进程的功能时， $\overline{\mathtt{in}}\ m$ 动作肯

[3] 本文在不强调协动作和动作的差别时，为方便行文，将其统称为动作。下同。

定是用于控制名字为 n 的灰箱进入 m 内部的，否则 $\mathtt{open}\, n\,.\, Q$ 就无法继续往下归约。但是 SA 的操作语义使得 $\overline{\mathtt{in}}\ m$ 同样具有让其它灰箱（如：式 (1.14) 中的 h ）进入的能力。由于不限制使用对象， SA 中的协动作虽然可以使得进程具有更强的并发性和归约结果的不确定性，但在很多情况下，这种功能导致进程行为具有不必要的二义性，使得程序不易编写、正确性也不容易判定。

事实上，省略 SA 中协动作的参数（例如表达式： $\overline{\mathtt{in}}\ .P$ 、 $\overline{\mathtt{out}}\ .P$ 和 $\overline{\mathtt{open}}.P$ ）可以得到与 SA 类似的效果。

基于以上分析，本文研究 SA 演算的一种变体——鲁棒灰箱（ robust ambient ）演算（以下简称 ROAM ），在利用协动作预防强干扰的同时，通过协动作的参数来进一步提高安全性和减少不必要的二义性。在 ROAM 中，协动作 $\overline{\mathtt{in}}$ 和 $\overline{\mathtt{out}}$ 的参数用来限定有权使用该协动作的灰箱名，只有在该指定灰箱中的动作才能使用该协动作。与 in 和 out 不同， open 动作无需处在某个灰箱中，因此协动作 $\overline{\mathtt{open}}$ 无需设定使用对象。基于上述原因， ROAM 中的 $\overline{\mathtt{open}}$ 保留了 SA 中操作语义，但不再附带一个不参加实际归约的参数。采用以上改进策略，式 (1.14) 、 (1.15) 在 ROAM 中分别可表示为：

$$n[\,\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.P] \mid m[\,\overline{\mathtt{in}}\, n\,.\,\mathtt{open}\, n\,.\,Q] \mid h[\,\mathtt{in}\, m] \tag{1.16}$$

$$m[\,\overline{\mathtt{out}}\, n\,.\,\mathtt{in}\, n.P \mid n[\,\mathtt{out}\, m\,.\,\overline{\mathtt{in}}\, m\,.\,Q] \mid h[\,\mathtt{out}\, m]\,] \tag{1.17}$$

式 (1.16) 中，灰箱 m 中协动作 $\overline{\mathtt{in}}\, n$ 的语义为“允许名为 n 的灰箱使用动作 $\mathtt{in}\, m$ 进入 m ”；式 (1.17) 中，灰箱 m 中协动作 $\overline{\mathtt{out}}\, n$ 的语义为“允许名为 n 的灰箱使用动作 $\mathtt{out}\ m$ 移出 m ”。显然，经过上述修改，协动作 $\overline{\mathtt{in}}\, n$ 和 $\overline{\mathtt{out}}\, n$ 被指定只能由灰箱 n 使用，灰箱 h 无法挪用。

当然，由于协动作参数的限制， ROAM 不再具有 SA 直接扩展 MA 的优点，因此 ROAM 的表达能力问题是文章在研究中要解决的一个核心问题。本文最后结论表明， ROAM 与 SA 同样具有翻译同步 π 演算的能力，这说明 ROAM 虽然对协动作参数增加了限制，但是没有失去灰箱演算固有的表达能力。

在灰箱演算中，除了 in 、 out 和 open 三个基本的原语外，还存在用于扩展灰箱演算功能的本地通讯原语 [5] 。一般将只包含三个基本原语的灰箱演算称为纯灰箱演算，将增加本地通讯原语的灰箱演算称为扩展灰箱演算。基于以下考虑，本文主要研究纯 ROAM 。

(1). ROAM 与其它灰箱演算的主要不同在于三个基本原语中的协动作及参数取舍，增加了本地通讯原语只会将问题复杂化。

(2). 本地通讯原语在引入时，主要为了能够模仿 π 演算中的通讯和替换功能，完成对 π 演算的翻译 [5] 。但最近的研究结果表明，用不具有通讯功能的纯 SA 就可完成对 π 演算的翻译 [18] 。因此对纯灰箱演算的研究也具有很重要的意义。

§1.2 论文的主要内容和结构安排

论文的研究工作是围绕 ROAM 进行的，研究的重点主要集中在 ROAM 的操作语义、类型系统、进程等价性和表达能力等方面。以下对本文的各章节内容做一简要介绍。

本文的第二章主要给出了 ROAM 的语法和归约语义(reduction semantics), 是全文对 ROAM 研究的基石。首先, 根据概述部分论述的引入理由, 给出了不带本地通讯原语的纯 ROAM 演算的语法, 包括进程(process)集合的定义和能力(capability)集合的定义, 并对进程和能力的各种构造, 特别是 ROAM 中协动作与相应的动作进行交互的方法和参数匹配进行了解释。之后, 汇总出 ROAM 的三种基本归约规则: 移入、移出、打开, 和四条推导归约规则。其中最后一条推导归约规则指出, 进程在归约前后可进行等价变换。这种定义进程等价变换的关系称为进程的结构同余(structural congruence)关系, 在进程代数的归约语义中广泛使用。第二章的最后用若干例子进一步说明了 ROAM 归约语义的使用, 同时通过这些例子也可以初步了解 ROAM 的表达能力。

本文的第三章主要研究了 ROAM 的类型(typing)问题, 提出了一套可以解决灰箱演算中存在的类型演化问题的类型系统 ETS-MT 。在众多灰箱演算类型系统的研究中, 类型的演化问题一直没有得到妥善地解决, 灰箱在被打开前后的类型变化得不到体现。 ETS-MT利用特殊的类型表达式区分了进程类型表达式中的 **当前类型** 和 **未来类型** 部分, 分别表示打开操作前后的不同类型特性, 配合特殊的能力类型构造, 有效地解决了灰箱演算中的类型演化问题。第三章对 ETS-MT的介绍从类型系统所需描述的进程属性开始, 给出了可以描述 **移动性**(mobility)和 **线程数**(threads)两种属性的类型语法, 并解释了进程类型表达式中针对类型演化问题的特定结构和作用。在此基础上, 定义了两个分别针对进程结构中的前缀构造(prefix)和并置构造(parallel composition)的类型运算符, 以方便进程类型的计算。 ETS-MT还支持进程类型的子类型(subtyping)关系, 使得类型限制较强的进程也可以出现在类型限制较弱的位置, 增加了类型系统的适应能力。在上述定义的基础上, 第三章给出了对进程和能力在给定类型环境下的类型判定规则(typing rules), 并证明通过该类型判定规则得到的进程类型的有效性(validity)、对结构同余关系的一致性(subject congruence)和对归约关系的一致性(subject reduction)。由于存在子类型关系, 除了类型判定规则, 第三章还给出了 ETS-MT中的最小类型算法(typing algorithm), 可以在给定的类型环境下, 计算进程、灰箱的最小类型, 同时证明了最小类型算法的正确性和完备性。最后, 给出了用最小类型算法计算非移动服务器类型的实例, 说明了 ETS-MT在刻划类型演化方面的能力。

本文的第四章主要给出了 ROAM 演算中与归约语义等价的一套标号转移语义(labelled transition semantics)。如果说归约语义是从宏观的角度来研究一个系统的变化, 那么标号转移语义则是从微观的角度, 研究系统中各个零部件的特性。归约语义可以直观地反映进程内各部分的交互, 但是不适合刻划进程与外部可能发生的各种交互。在进程等价性研究中, 往往通过定义与归约语义等价的标号转移语义来实现这一点。在已有灰箱演算研究中, 主要有两种途径来定义标号转移语义, 一种是基于 **提交**(commitment)和 **结果**(outcome)的方法 [19, 16], 比较直观, 但是标号众多, 且标号转移结果除了进程还可以是 **固化结果**(concretion); 另一种是基于 **硬化关系**(hardening relation)的方法 [20], 引入了一种新的关系, 但标号转移结果仍为进程, 且非内部转移标号的个数和语法中的动作个数相同。这两种方法各有特色, 我们采用后一种方法。第四章首先引入了 ROAM 进程的硬化关系, 用于将进程中可能参与归约的部分与剩余的部分分开。其次, 在硬化关系的基础上, 按照 ROAM 进程的归约特性定义了具有五条规则的标号转移语义, 并逐一分析了这些规则与归约规则的对应关系。最后详细证明了两套语义的等价性(correspondence)。

本文的第五章在标号转移语义的基础上, 参照 Gordon 和 Cardelli 研究 MA 进程等价性的方法 [20], 研究了判定两个进程等价的一般方法, 即回答这样的问题: 这里的进程等价具体应该如何定义? 如何具体进行等价性的判断? 对于第一个问题, 显然完全相同的进程、或者结构同余的

进程是等价的，但是我们希望能够范围扩大，例如应该包含 $n[\mathbf{0}]$ 和 $\mathbf{0}$ 是等价的这种结论。因此，本章进程等价性的研究目标为基于观察结论（ obervation 、或 barb ）的上下文等价性（ contextual equivalence ），即：如果两个进程等价，则把它们放在任何上下文（ context ）中，得到的结果从外部观察是无法区别的。对于第二个问题，本章研究了一类称为 **简单上下文**（ harness ）的特殊上下文，并指出只要两个进程放在任何简单上下文中看不出区别，则放在任意的上下文中也不会看出区别（ context lemma ）。这样就把研究将进程放在任何上下文中可能会发生的情况简化为只研究将进程放在这类特定的上下文中会发生的情况。第五章同时给出了 ROAM 进程与这一类简单上下文发生交互的各种可能，作为判定进程等价的一般性结论（ activity lemma ）。

本文的第六章综合前文第三、四、五章的结论，给出用于判定进程等价的一些代数定律（ algebraic laws ）。如果说上一章给出了判断两个进程等价性的一般方法，本章给出的就是判定等价的具体工具：这些定律可以看作是一些模板，满足这些模板的进程都是等价的，这些进程放在任何上下文得到的结果也是等价的。本章的等价性定律在很大程度上建立在 ETS-MT中的移动性和线程数基础上，特别是单线程属性和非移动属性。这些等价性定律分为：无上下文条件的单线程定律、有上下文条件的单线程定律、定点接收定律以及一些附带的辅助推论。本章在最后还给出了使用这些定律证明灰箱换名（ renaming ）例子和防火墙跨越（ firewall-crossing ）例子在满足一定条件下前后等价的过程，不仅示意了等价性定律的具体使用方法，而且也通过这些例子中条件的减弱，说明 ROAM 较之 MA 和 SA 在安全性方面的优点。

本文的第七章给出并证明了纯 ROAM 演算翻译 π 演算的一个方案。它既可以看作是等价性定律应用的一个实例，又可以作为 ROAM 演算表达能力的有力佐证。一个演算翻译另一个演算的证明工作一般可分为三个部分。第一部分为这两个演算各自的形式化定义，包括语法和操作语义。第二部分为具体翻译方案的制订。第三部分为翻译方案的正确性证明。这里的正确性证明又可分两层涵义。一是翻译结果对原进程行为的保持，原进程可以有哪些动作，翻译结果也可以有相应类似的动作，且得到的结果为一个与原归约结果的翻译具有等价关系的进程。二是翻译结果自身的任何行为不能偏离原进程原来的行为，翻译结果的任何归约结果都可以有原进程的一个归约结果的翻译与之等价。这两层意义综合起来，称为操作一致性（ operational correspondence ）[4]。在具体的翻译中，我们采用了文献 [18] 中与 π 演算等价的中间演算 π_{esc} 演算，通过翻译并证明对 π_{esc} 演算的翻译来得到对 π 演算正确性的证明。

第八章是对本文工作的总结和进一步工作的设想。

在阅读次序上，本文第三章类型系统的内容和第四、五章标号转移语义和等价性判定的内容没有直接联系，可自由安排先后次序。在第二章的基础上，可以依次阅读第三、四、五章，也可跳过第三章，先阅读第四、五章。但是第六章等价性定律的研究是同时建立在类型系统和等价性判定的基础上的，最好在第三、四、五章均完成后进行。

§1.3 与本论文相关的研究工作

本节给出与本论文相关的一些研究工作，分为四个部分：相关的移动计算形式化模型、以 MA 为框架的相关工作、以 SA 为框架的相关工作和其它基于灰箱演算的相关工作。由于 MA 是第一

[4] operational correspondence 在不同的文献里有不同的定义 [9, 21, 22]，有的比上述的更加严格，可细分为很多种。本文对翻译过程的证明使用上述比较粗糙的定义。

个，也是最基本的灰箱演算模型，同时SA 的提出可以看作是在灰箱演算研究过程中继首次提出灰箱演算以来的又一个里程碑。因此，本节分别将基于 MA 和 SA 的相关工作单列开来，以视区别。

§1.3.1 移动计算的形式化模型

除了灰箱演算外，对移动计算形式化模型的研究工作主要基于 λ 演算的扩展 [14]、基于 π 演算 [8] 的扩展 [11, 12, 13]、基于 join 演算 [24] 的扩展 [15] 和 Seal 演算 [17] 等。

（一）λ 演算的扩展

文献 [14] 结合移动计算中计算性与移动性并存的特点，在成熟的 λ 演算理论基础上，引入位置的概念和子表达式的移动能力，试图依托完善的 λ 演算系统来对移动计算作形式化的建模和分析。在文献 [14] 提出的 λdist 演算中，对一个特殊表达式 "go p" 的求值结果可使表达式中用 "$\langle\ \rangle$" 标注的子表达式 (代理) 移动到位置 p 并继续计算。由于 λ 演算是基于表达式求值的概念，文章还将数据的移动性做了一个分类，使得代理在移动后进行数据存取时按照数据的移动性不同而执行不同的操作。不过由于 λ 演算本身仅仅是一套表达式求值工具，λdist 演算侧重描述一些远程求值应用或简单的代码移动应用，对于以并发和通讯为基础的移动计算缺乏强有力的支持。

（二）π 演算的扩展

基于通道通讯的 π 演算以其成熟的并发理论基础成为移动计算形式化的一个很好的出发点，如何扩展 π 演算使其支持位置及资源分布等功能也成为一个研究的热点。π 演算在分布式上的扩展工作主要有 Dπ[11]、π_{1l}[12]、Distributed-π[13] 等。文献 [11] 中的 Dπ 演算在 π 演算的基础上增加了位置的概念，进程在位置中运行，进程可以通过移动原语（ `goto` ）从一个位置迁移到另一个位置，通讯只能发生在本地，远程通讯通过进程的移动和本地通讯间接完成。在其后续工作 [23] 中，原作者还对 Dπ 中类型的一致性和资源访问的权限控制做了进一步的研究。文献 [12] 提出了异步 π 演算的一个子集 π_1 演算。通过语法和一套类型系统的限制，在 π_1 演算中，每个通道有且仅有一个进程负责处理发向该通道的通讯内容。虽然作了以上限制，π_1 演算仍可翻译常规的异步 π 演算。在 π_1 演算的基础上，作者又增加了位置的概念，进程在位置（ location ）中运行，可以向位置中的进程传递信息（ message ），信息包括普通的通道信息交换或一些特殊的原语： `stop` - 使某个位置失效； `spawn` - 在某位置创建一个进程； `ping` - 测试某位置是否失效。这套增加了位置的新演算称为 π_{1l} 演算。最后作者对其语义和等价性做了研究并给出了一套将 π_{1l} 演算翻译成等价的 π_1 演算的方法。文献 [13] 中的 Distributed-π 在 π 演算基础上增加了位置的概念（ $@_u$ ）和进程迁移的操作（ `migrate_to` ），来提供对本地和远程资源的区别使用。与 Dπ 不同，其位置空间是嵌套结构的，且支持基于通道的远程通讯。另外其类型系统着重于将本地和远程通道用不同标志进行区分，以便更加有效的实现基于通道的通讯功能。

（三）join 演算的扩展

join 演算 [24] 是一种 π 演算的变体，起源于化学抽象机（ chemical abstract machine ）[25]，但更加强调了信息交互发生的位置。分布式 Join 演算 [15] 又在 join 演算的基础上增加了对位置的命名和进程的移动功能。位置之间呈现树状嵌套关系，节点及其子树可作整体移动。但与灰箱演算不同，分布式 join 演算中的移动可以发生在任意两个位置之间。在表达能力方面，基本的 join 演算与 π 演算的表达能力是相等的 [26]。因此，可以认为，在增加了位置和移动性后，分布式 join 演算的表达能力与上述几种分布式 π 演算比较相近。

（四）seal 演算

文献 [17] 中所述的 seal 演算也是一种分布式的进程演算，比较侧重安全控制机制。seal 演算参考了灰箱演算的概念，也具有一个树状嵌套、可以移动且具有安全防护功能的灰箱结构，称之为 seal 。但是 seal 演算中的通讯和移动机制却与灰箱演算不同。同灰箱演算的内部匿名通讯机制不同，seal 演算仍然采用 π 演算中的有名通道通讯机制，且通讯的内容只能是名字。另外，除了本地通讯以外，seal 演算还允许穿越边界的通讯方式，通过在通道上的不同标记 x 、$\uparrow$ 或 $*$ ，通讯被指定发生在下层、上层或本地。由于通讯的双方参与性，这种通讯方式不会直接引入跨越边界带来的安全性问题。在移动性方面，seal 演算中不存在灰箱演算的 `in` 、`out` 操作，取而代之为名通讯机制，一个 seal 名可以通过外部的通讯通道被送到上一层或下一层。另外，seal 演算中的设计思想，特别是灰箱内外的通讯与目前的程序语言实现比较接近，有利于其最终的实现。

§1.3.2 以 MA 为框架的相关工作

本小节主要介绍在原有 MA 的语法基础上，对 MA 的类型系统、进程等价性等方面一些现有的研究工作。

§1.3.2.1 对 MA 类型系统的研究

Cardelli 和 Gordon 在提出 MA 时，还引入了输入和输出两个本地通讯原语用于同层进程的信息交互，并利用这两个原语完成了对 π 演算的翻译 [5] 。这种具有通讯扩展的 MA 演算的具有如下的语法结构：

$$\begin{aligned} P &::= \mathbf{0} \mid (\nu n)P \mid P \mid P \mid !P \mid M.P \mid n[P] \mid (x).P \mid \langle M \rangle \\ M &::= \epsilon \mid \texttt{in}\ n \mid \texttt{out}\ n \mid \texttt{open}\ n \mid M.M \end{aligned}$$

其输入 $(x).P$ 和输出 $\langle M \rangle$ 构造在并置时，可利用如下的通讯规则进行归约，结果为将进程 P 中的所有自由变量 x 替换为 M （用 $P\{M/x\}$ 表示）。

$$(x).P \mid \langle M \rangle \quad \longrightarrow \quad P\{M/x\} \tag{1.18}$$

在扩展 MA 中，消息传递的单位为一个能力串 M 。以下示意了一个利用移动代理进行通讯的例子（花括弧表示发生归约的动作或进程）：

$$\begin{aligned} & m[p[\underbrace{\texttt{out}\ m}\ .\ \texttt{in}\ n\ .\ \langle M \rangle \mid P\ ]\ ] \mid n[\texttt{open}\ p\ .\ (x).Q] && (1.19)\\ \longrightarrow\ & m[P] \mid p[\underbrace{\texttt{in}\ n}\ .\langle M \rangle] \mid\ n[\texttt{open}\ p\ .\ (x).Q] && (1.20)\\ \longrightarrow\ & m[P] \mid n[\underbrace{\texttt{open}\ p}\ .\ (x).Q \mid\ p[\langle M \rangle]\ ] && (1.21)\\ \longrightarrow\ & m[P] \mid n[\underbrace{(x).Q}\ \mid\ \underbrace{\langle M \rangle}] && (1.22)\\ \longrightarrow\ & m[P] \mid n[Q\{M/x\}] && (1.23) \end{aligned}$$

这里，灰箱 m 利用灰箱 p 的移动将消息 M 传送给灰箱 n 中的进程 Q 。在现实中，m 和 n 可分别代表两个网络节点，p 则可代表一个用于消息传递的移动代理。

扩展 MA 的语法中会出现如由 $((x)\,.\,x\,.\,P)|\langle n \rangle$ 归约得到 $n\,.\,P$ 这种符合语法和归约规则，但没有意义的表达式，因为灰箱名不能作为进程执行的动作。而从语法上直接消除以上无意义表达式

代价比较大。因此，对扩展 MA 的类型系统研究中，最基本的一个要求便是消除上述由语法导致的无意义进程表达式 [27]。同时，一个良好的类型系统还要保证输入输出通讯时的类型匹配。在扩展 MA 类型系统上的进一步研究还包括：进程的移动性 [28]、多元（ polyadic ）通讯中的子类型关系 [29]、利用额外的组构造增加 MA 的安全性 [30] 等等。

文献 [27] 是对 MA 类型系统的最早期研究。在文献 [27] 中，类型被分为能力 $(M : W)$ 和进程 $(P : T)$ 两大类分别定义。消息 M 有灰箱 $(Amb[T])$ 和动作串 $(Cap[T])$ 两种基本类型。类型表达式 $Amb[T]$ 用来表示一类灰箱名，在这些灰箱中可以进行类型为 T 的信息交换。而 $Cap[T]$ 则用于跟踪灰箱的打开操作，执行该类型动作后可能会打开类型为 $Amb[T]$ 的子灰箱，其结果是该子灰箱中的进程被上移，使得打开者进程同样具有进行 T 类型信息交换的能力。信息交换类型 T 表示输入输出进程交换信息的类型，可以是不交换任何信息的简单握手（ Shh ），也可以是交换一个元组的信息（由于此类型系统是基于多元（ polyadic ） MA 而提出的，故此这里使用元组的概念），如果元组各元素的类型分别是 $W_1, \cdots, W_k$ ，则该进程的类型为 $W_1 \times \cdots \times W_k$ 。

在文献 [28] 中， Cardelli 等人又对 MA 中的类型概念进行了扩展，提出了灰箱可否被打开的锁定（ lock ）概念 Y ，以及进程的移动性（ mobility ）概念 Z 。灰箱可被声明为锁定和非锁定两类，锁定的灰箱永远不能被打开。增加锁定属性的灰箱类型表达式变为 $Amb^{Y}[T]$ 。进程分为可移动和非移动两类，带 `in` 和 `out` 的进程直接被赋予可移动类型，而 `open` 则可能通过引入可移动的进程而导致原进程的可移动性。这样进程的类型被细化为 ${}^{Z}T$ 。包含类型 ${}^{Z}T$ 的进程的灰箱具有类型 $Amb^{Y}[{}^{Z}T]$ ，可能会引入 ${}^{Z}T$ 类型进程的能力类型为 $Cap[{}^{Z}T]$ 。

文献 [28] 中还引入了对被动移动（ objective move ）的类型描述。从纯语法角度，被动移动存在安全性问题，且可用主动移动加上被移动灰箱的显式允许来实现 [5]。然而在类型系统中，被动移动却具有特殊的用途。如果只使用主动移动，则无法将一些只接收并打开数据包，而本身并不移动的灰箱的移动属性设为不移动 [5]，原因在于它要打开一个移动的数据包。然而如果采用被动移动，就可解决这个问题。被动移动的语法为： $\mathbf{go}\ N\,.\,M[P]$ ， N 是一串只包含 `in` 和 `out` 的动作串。对应被动移动的两条归约规则如下：

$$\begin{array}{lll} \textbf{(Red Go In)} & \mathbf{go}\ (\mathtt{in}\ m\,.\,N)\,.\,n[P] \mid m[Q] & \longrightarrow \quad m[\,\mathbf{go}\ N\,.\,n[P] \mid Q] \\ \textbf{(Red Go Out)} & m[\,\mathbf{go}\ (\mathtt{out}\ m\,.\,N)\,.\,n[P] \mid Q] & \longrightarrow \quad \mathbf{go}\ N\,.\,n[P] \mid m[Q] \end{array}$$

使用被动移动后的灰箱类型表达式扩展为 $Amb^{YZ'}[{}^{Z}T]$ 。新增的 Z' 表示灰箱的被动移动性，一个灰箱可以是被动可移动而主动不可移动的。这样，父灰箱打开这种灰箱时就无需再引入移动性了。

值得指出，上述方法通过引入新的被动移动原语来解决类型演化的一种特例，即：灰箱类型由初始的“移动”状态演化为打开前的“非移动”状态。而本文提出的演化类型系统专门针对灰箱演算中普遍存在的类型演化问题提出了一套融入类型演化思想的类型系统，无需引入这种被动移动原语，即可很好的支持更广泛意义上的类型演化。

Zimmer 对 MA 本地通讯的子类型系统研究 [29] 是在文献 [28] 之上的进一步发展。首先，在文献 [28] 中的锁定概念是针对灰箱名而言的，同名灰箱如果出现多次，其锁定属性必须完全一致。Zimmer 减轻了该限制，直接在语法层引入了锁定和非锁定两类灰箱符号。这样，灰箱的锁定与否

[5]该问题正是本文第三章的演化类型系统所要解决的类型演化问题。

无需再靠类型来判断，而且减小了锁定的颗粒度。 Zimmer 对 MA 语法的第二个改动是增加了一个用于限定非移动特性的 imm 能力，一个具有 imm 能力的进程一定是不可移动的。增加 imm 能力的好处是无需类型信息就可将进程直接定义为非移动，而不再需要借助于类型手段。在进程参与归约时， imm 能力等效于 ϵ ，不起任何作用。在文献 [28] 中所提出的类型系统之上， Zimmer 在文献 [29] 中又进行了大量的工作，不仅仅细化了进程的移动属性和灰箱的锁定属性，还引入了子类型关系，并将固定的信息交换类型扩展为一个信息交换范围。文章还进一步提出了一套确定性（ deterministric ）的类型求解算法（ typing algorithm ）。

Cardelli 等人在文献 [30] 中又对如何使用类型系统加强 MA 的安全性进行了研究。其最突出的贡献是在灰箱演算语法基础上增加了 **组**(Group) 的创建功能，每个灰箱名的类型不再是统一的 $Amb[\,]$ ，而是 $G[\,]$ ，其中 G 为该灰箱所属的组。这样，得到了一个更加细化的灰箱类型系统。通过组的概念，类型系统就可以刻划出以下一些更加详细的动作许可，如：

- 进程 P 可以且只可以穿越属于 G 组的那些灰箱。
- 进程 P 可以且只可以打开属于 G 组的那些灰箱。
- 灰箱 n 可以且只可以被动穿越属于 G 组的那些灰箱。

文献 [30] 中不再保留原有的锁定属性和移动属性，它们已经被包含在上述对组的穿越和打开概念中。通过引入组的概念， MA 的安全性得到了进一步的加强。例如通过一定手段可以保证一个新建的灰箱名字不会通过隐秘通道被传送到外部。组的概念在后来的 [31, 32, 33] 等文献中均得到了采用。这种安全特性甚至还被用在加强 π 演算的安全性上 [34] 。

最近，Cardelli 等人将其在 MA 类型系统研究中的上述一些工作 [27, 28, 30] 进行了总结，作为一个阶段性的成果，参见文献 [35] 。

§1.3.2.2 对 MA 的其它一些研究工作

除了上述对 MA 类型系统的研究工作以外，对 MA 的研究工作还有很多，如：进程等价性的研究 [19, 20] 、基于 Java 的 MA 的实现问题 [36, 37] 、基于 Jocaml [38] 的 MA 的实现问题 [39] 、与 MA 相关的模态逻辑（ model logic ） [40, 41] 和模态检查（ model checking ）研究 [42, 43] 、对进程行为的静态分析 [44, 45, 46, 47, 48, 49] ，以及利用 MA 的嵌套结构来研究万维网上半结构数据的存储和检索 [50] 等。文献 [51] 还研究了 MA 在移动代理（ mobile agent ）系统设计中的应用，并从移动代理系统设计角度给出了一些灰箱演算的扩展方向。限于篇幅关系，这里主要就与本文相关的进程等价性方面的研究工作做一简单介绍。

在最早期的 MA 文献 [5, 19] 中， Cardelli 和 Gordon 就开始着手对进程的等价性进行研究。其主要工作集中在对进程等价性的定义和对一些简单等价性问题的证明，如证明当 $n \notin fn(P)$ 时， $(\nu n)n[P]$ 和 $\mathbf{0}$ 从外部观察是等价的。在后来的文献 [20] 中， Gordon 和 Cardelli 总结了文献 [5] 和 [19] 的研究成果，并对 MA 的等价性做了一个较为完整、全面的定义，其主要结论是文中的 context lemma 和 activity lemma 。这两个结论提供了 MA 进程间等价性的证明手段。其中，context lemma 指出， MA 进程一般意义上的上下文等价（ contextual equivalence ）关系等同于基于某类称为 harness 的特殊上下文所定义的比较简单的等价关系。随后， activity lemma 给出了具体对该简单的等价关系进行判断的一般方法。最后，文章利用这两个结论，证明了 MA 中防火墙

跨越（ firewall-crossing ）例子的正确性[6]。

值得指出，在本文对 ROAM 进程等价关系的研究中，我们采用了与文献 [20] 类似的手段，证明了 ROAM 在扩展了 MA 的语法之后，其进程的等价性仍可鉴戒 MA 中已有的结论。但是，本文对进程等价性的研究不仅仅止于文献 [20] 中的两个结论，而是将 ROAM 中类似的结论与本文对 ROAM 类型系统的研究结论相结合，进一步得到了 ROAM 进程的若干等价性定律。这些定律可以方便地应用于复杂进程等价性的证明中。文中的第七章还给出了如何应用这些定律，简化对 π 演算翻译结论的证明。

§1.3.3 以 SA 为框架的相关工作

与本文研究最密切相关的工作是 Levi 和 Sangiorgi 提出的 SA 演算 [16]。由于 SA 指出并利用协动作解决了 MA 中的强干扰问题，同时又可看作是 MA 的一个超集，因此从某种意义上说，SA 正在逐渐取代 MA 作为灰箱演算研究的一个新的代表。这里将 SA 及其后续工作单列为一个部分，以示 SA 的特殊性。

Levi 和 Sangiorgi 在文献 [16] 中首次指出了 MA 中存在的强干扰问题，并通过引入协动作提出了可以消除强干扰的 SA 演算。文献 [16] 还深入地研究了 SA 的类型系统、标号转移语义和进程的等价性判定方法，并给出若干进程等价性的判定定律。文献最后还列举了大量例子对比说明克服强干扰问题后 SA 展示的优越特性，特别是对 π 演算翻译的代数方式证明。由于 SA 和本文关系密切，以下对文献 [16] 中的有关内容进行更详细的介绍。

在类型系统方面，文献 [16] 由浅入深引入了三种不同的类型概念：**基本类型**（ basic ）、**非移动类型**（ immobility ）和 **单线程类型**（ single-threadness ）。**基本类型**的表示几乎继承自 MA 的类型研究 [27, 28]：消息 M 的类型 W 被分为基本灰箱类型 $BAmb[T]$ 和基本动作串类型 $BCap[T]$，进程具有基本进程类型 $BProc[T]$，这里 T 同样是允许交换的消息类型，可以是某种消息 W，也可以是不交换任何内容的简单握手标志 Shh。在基本类型中，文献 [16] 又区分出一种特殊的 **非移动类型**，用于强调那些既不能移动也不能被打开的灰箱。具有非移动特性的灰箱、动作串和进程分别用 $IAmb[T]$、$ICap[T]$ 和 $IProc[T]$ 表示。第三种类型被称为 **单线程类型**，具有单线程特性的灰箱或进程在任何时刻只能激发一个动作，也就是说不存在同时可以并发运行的两个动作。需要指出，这些动作只是指最顶层的动作，在子灰箱中的动作不予考虑，因此，单线程进程的并发性仍然很高。具有单线程特性的灰箱、消息和进程分别用 $STAmb^I[Tt]$、$STCap^I[Tt]$ 和 $STProc^I[Tt]$ 表示，其中 I 用于标记没有线程或单线程，Tt 表示进行了有无线程的标记后消息传递类型 T。后两种类型，即非移动类型和单线程类型的提出，为文献 [16] 中后续的等价性定律的研究打下了基础。

在进程等价性研究方面，文献 [16] 采用类似文献 [19] 中的基于提交（ commitment ）和结果（ outcome ）的方法，首先给出了与操作语义等价的一套标号转移语义，然后定义了 SA 进程集上基于观察（ barb ）的互模拟（ bisimulation ）关系和类型约束条件下基于观察的同余（ congruence ）关系，并结合非移动类型和单线程类型给出了一系列的进程等价性判定定律。文献 [16] 给出的等价性定律可分为如下三个部分：

(1). 无类型限制的等价性定律，共 7 条；

[6] 本文第六章将详细比较 MA 、 SA 和 ROAM 版本防火墙跨越例子正确性所需条件的差别。

(2). 基于单线程灰箱的等价性定律，共 4 条；

(3). 基于非移动类型和单线程类型的定点接收定律，共 3 条。

文献 [16] 最后给出了大量的实例，如：灰箱的换名操作、数据信包的路由、防火墙跨越和 SA 对 π 演算的翻译等，并分别使用上述等价性定律给出了相应结论的代数证明。需要指出，文献 [16] 中翻译 π 演算使用了带本地通讯原语的扩展 SA 演算，这和后来文献 [18] 给出的使用无本地通讯原语的纯 SA 演算的翻译有本质的差别，后者的难度要大得多。

本文中对 ROAM 的类型系统和等价性定律的研究在很大程度上沿袭了文献 [16] 的思路。然而，本文的工作和后者也有很大差别。首先，在类型系统的研究中，文献 [16] 首次提出了单线程类型的概念，然而，其单线程类型系统和另外的基本类型和非移动类型没有有机地结合在一起，而本文的类型系统首次将移动和非移动，无线程、单线程和多线程这些概念统一在一个简洁的框架中，同时通过子类型关系精确地刻划了这些概念之间的联系。其次，文献 [16] 的类型系统不能很好地刻划类型的演化问题，而本文的类型系统将支持类型演化作为类型系统设计中的核心内容，使得本文中的类型系统 ETS-MT对进程和灰箱类型的描述更加合理和细腻。另外，本文等价性定律的证明与文献 [16] 中利用互模拟 (bisimulation) 的方法不同。本文采用对进程和其环境可能发生的所有交互方式进行一一分析。这种证明方法可以认为是利用文献 [20] 中的结论作为起点，进一步向前推进和发展的结果。在该方向的研究达到了与文献 [16] 异曲同工的效果。

在对 SA 的研究中， Zimmer 在文献 [18] 中使用纯 SA 翻译 π 演算的工作可以说是对 SA 优越性的又一个有力支持。在灰箱演算刚刚提出时， Cardelli 和 Gordon 便把灰箱演算和 π 演算的关系作为一个很重要的研究内容。最初灰箱演算中本地通讯原语的引入在一定程度上就是为了使灰箱演算在刻划计算的移动性和分布性的同时，也具有较强的表达能力。而翻译 π 演算就是这种表达能力的最有力证明。但在最初的文献 [5] 中，并没有给出对其翻译过程的严格证明。在 SA 的研究中， Levi 和 Sangiorgi 也利用带本地通讯原语的 SA 给出了对 π 演算的翻译，同时利用等价性定律证明了该翻译的操作一致性 [16] 。然而，上述研究中对 π 演算的翻译都无一例外地使用了经本地通讯原语扩展的灰箱演算来模拟 π 演算基于通道的通讯。文献 [18] 中首次给出并证明了不带本地通讯原语的纯安全灰箱演算对同步 π 演算的翻译方案，使人们对灰箱演算的表达能力有了更进一步的认识。它最突出的特点是用没有全局替换操作的构造来表示 π 演算和其变体中普遍存在的全局替换操作。为此，文献 [18] 中首先设计了一种没有全局替换操作的新的 π 演算变体——π_{esc} 演算并给出了 π_{esc} 演算和 π 演算的相互翻译方法。在此基础上，文献 [18] 给出了用纯 SA 对 π_{esc} 演算的翻译，最后给出并证明了对 π 演算直接进行翻译的操作一致性。

但是，这里有一点值得指出，文献 [18] 中对翻译方案的证明过程比较复杂。其直接原因在于，对 π 演算的翻译必然引入类型演化的问题，即代表 π 通道的多线程非移动灰箱必然要打开代表 π 通讯进程的移动灰箱。由于没有引入类型演化机制，对进程的类型不能精确定义，对翻译结果的证明就无法使用文献 [16] 中判断进程等价性的代数定律。而在本文最后给出的纯 ROAM 演算翻译 π 演算的证明中，则使用了类似文献 [16] 的代数方法，直接通过等价性定律完成翻译的正确性证明。这不但说明了 ROAM 具有类似 SA 的表达能力，而且说明了演化类型系统在精确刻划进程类型中的突出作用。

在文献 [52] 中， Sangiorgi 和 Valente 还初步研究了 SA 演算的分布式虚拟机实现问题，给出了一个支持单线程和非移动等进程类型的虚拟机的语法和语义，并证明了该虚拟机执行 SA 进程

的正确性。文献 [53] 对 SA 进程的静态分析进行了研究。另外文献 [32] 和文献 [31] 还以 SA 为基础，研究了基于类型系统来提高 SA 安全性的方法。

§1.3.4 其它基于灰箱演算的相关工作

灰箱演算可以说是近期一个非常活跃的研究领域，除了上述以 MA 和 SA 为基础进行的研究外，还有很多对其它的研究工作。

Amtoft 等人在灰箱演算中增加了 λ 演算的抽象和函数作用构造，得到了所谓的 AC^+ [54] 演算，并研究了 AC^+ 演算中类型的各种多态性。最近 Amtoft 又在文献 [33] 提出了利用文献 [30] 中提出的“组”的概念，在最基本的 MA 演算上实现对灰箱可以移动范围的判定，在某种意义上又从类型角度弥补了 MA 缺乏协动作带来的强干扰问题。 Bugliesi 等人在文献 [55] 中提出了一种称为 boxed ambients 的新演算，取消了灰箱演算的 `open` 原语，代之于类似 seal 演算的父子灰箱通讯构造。该演算在安全性控制方面比较突出。通过引入适当的类型系统，该演算对消息的传送方向可以进行有效地控制。

以上这些研究工作比较注重实用性，似乎并不考虑演算的代数性质。在最近发布的文献 [56] 中， Merro 和 Hennessy 在 SA 的动作和协动作上增加了称为密码（ password ）的额外参数，得到了所谓的 **密码安全灰箱演算**（ SAP — Safe Ambients with Passwords ）。 SAP 通过专用的密码参数控制动作和协动作的匹配，进一步强化了灰箱演算对安全性的控制能力。同时，密码的使用也使得 SAP 的代数性质有了很大的加强。文献 [56] 还对 SAP 的互模拟关系进行了研究，得出了 SAP 中基于观察的同余关系（ barbed congruence ）的特殊性质，即 SAP 中无论通过定义何种观察所诱导的基于观察的同余关系均相同。同时，文献 [56] 还证明了 SAP 中的基于观察的同余关系与基于标号的互模拟等价关系为同一关系这一结论。

SAP 在某种意义上可看作是灰箱演算代数性质研究的又一飞越。最初灰箱演算代数性质薄弱这一问题正在逐步被克服，这在 SAP 的研究中得到了最有力的证明。 MA 进程可以看作是 SA 进程的一个特殊子集，而 SA 进程又可以看作是 SAP 进程的一个特殊子集。国外研究人员这种在研究中非常注重继承性和兼容性的做法，值得我们去领悟和学习。

本章小结

本章是本文的引言部分，阐述了论文研究的意义和主要内容安排，并介绍了与本文相关的一些研究工作。本章首先介绍了移动计算形式化模型的研究意义。其次，以灰箱演算中的强干扰问题为线索，分别介绍了 MA 、 SA ，并给出了引入 ROAM 的理由。之后，介绍了论文的主要内容和结构安排。本章最后介绍了本文的一些相关研究工作，包括：其它的移动计算模型、基于 MA 的研究工作、基于 SA 的研究工作和其它基于灰箱演算的研究工作等。

第二章　鲁棒灰箱演算

本文的概述部分较为详细的介绍了移动计算形式化的研究意义和引入 ROAM 的理由。从本章开始，论文内容将围绕 ROAM 进行展开。本章首先给出 ROAM 的语法和归约语义，并给出一些使用 ROAM 描述的例子。

§2.1　语法

鲁棒灰箱演算（ ROAM ）是安全灰箱演算（ SA ）的一个变体， ROAM 与 SA 的最主要区别是协动作参数在归约中的作用。基于前文的论述，下面用较为形式化的语言引入 ROAM 的语法。

令 $\mathcal{N}$ 为一可数名字集合，其元素用 n 、 m 等小写字母表示。 ROAM 的进程（ process ）集合 $\mathcal{P}$ （其元素用 P 、 Q 等大写字母表示）递归定义如下：

$$P ::= \mathbf{0} \mid (\nu n : T)P \mid P \mid Q \mid !P \mid M.P \mid n[P]$$

其中 T 为类型表达式，在引入 ROAM 的类型系统之前，可不必考虑其涵义。 M 为能力（ capability ）集合 $\mathcal{M}$ 的元素。 $\mathcal{M}$ 的每个元素为一个动作（ action ）串（用 M 、 N 等元素表示），递归定义如下 [1] ：

$$M ::= \epsilon \mid \mathtt{in}\ n \mid \mathtt{out}\ n \mid \mathtt{open}\ n \mid \overline{\mathtt{in}}\ n \mid \overline{\mathtt{out}}\ n \mid \overline{\mathtt{open}} \mid M.N$$

进程的前面四种构造“ $\mathbf{0}$ ”、“ $(\nu n : T)P$ ”、“ $P \mid Q$ ”和“ $!P$ ”分别表示进程代数中通常意义上的静止进程（ inactive ）、名字约束（ restriction ）、进程并置（ parallel composition ）和进程复制（ replication ）。 $M.P$ 表示进程可以顺序执行能力 M 中的动作。 $n[P]$ 表示一个名为 n 、内部运行进程 P 的灰箱。

由于 ROAM 的语法和最初的 MA 演算 [5] 有很大的相似之处。为了使读者对灰箱演算的基本概念有一个比较直观的了解，以更好的理解本文的内容，下面参照文献 [5] 的内容，给出上述灰箱演算基本构造的描述性说明。

- **静止进程**（ $\mathbf{0}$ ）：进程递归定义的出发点，指不包含任何内容的进程。也可理解为不与外部进行任何交互的进程。一些进程代数中有时也用 **nil** 来表示的。
- **约束**（ $(\nu n : T)P$ ）：创建一个新的（不会在 P 以外使用的）灰箱名 n ，该新名字可作为 P 中的灰箱名或动作的参数，其类型用 T 表示。在不考虑类型时，可直接写作 $(\nu n)P$ 。被约束的灰箱名（这里为 n ）称为约束名（ bound name ）。与之对应，其它非约束的名字称为自由名（ free name ）。约束名可进行换名（由于历史的原因，通常称为 α 转换），换名前后的进程在语义上可认为是完全相同的。约束构造 $(\nu n : T)$ 可在必要时扩大或缩小其作用范围，但前提是不能将其它灰箱名的自由或约束性质改变，必要时可使用 α 转换换名。此外，归约操作可在名字约束构造内部进行，这可用以下规则示意 [2] ：

$$P \longrightarrow Q \quad \Longrightarrow \quad (\nu n : T)P \longrightarrow (\nu n : T)Q \tag{2.1}$$

[1] 取消 $\mathcal{M}$ 集合中的 ϵ 和 $M.N$ 构造后将得到同样的 $\mathcal{P}$ 集合，增加这两个构造用于体现下文演化类型系统中更具一般性的能力类型。

[2] 本文用 $\Longrightarrow$ 表示“如果... 那么...”的含义，用 $\Longleftrightarrow$ 表示“当且仅当”的含义。以下如不作特别说明，均同。

- **并置**（ $P \mid Q$ ）：表示两个并发运行的进程，进程间可以各自归约、也可发生交互。二元运算符“ | ”满足交换律和结合律。并置进程可各自归约这一性质可用以下规则示意：

$$P \longrightarrow Q \quad \Longrightarrow \quad P \mid R \longrightarrow Q \mid R \tag{2.2}$$

- **复制**（ $!P$ ）：复制构造是为了表达进程的一些重复或递归特性。 $!P$ 代表了无穷多个 P 的副本，可不停产生出与其并置的进程 P ，即： $!P$ 等价于 $P \mid !P$ 。值得注意， $!P$ 本身不能进行归约，即以下推导不能成立：

$$P \longrightarrow Q \quad \not\Longrightarrow \quad !P \longrightarrow !Q \tag{2.3}$$

- **动作**（ $M\,.\,P$ ）：有时也称为前缀（ prefix ）构造。进程 $M\,.\,P$ 在执行完能力 M 中的所有动作后，演变为进程 P 。对应 M 中包含的每一个动作，都有其对应的归约方式（这些归约方式由下文的归约规则具体给出）。在 M 完成之前，进程 P 不能单独进行归约。

$$P \longrightarrow Q \quad \not\Longrightarrow \quad M\,.\,P \longrightarrow M\,.\,Q \tag{2.4}$$

- **灰箱**（ $n[P]$ ）：进程 $n[P]$ 表示了一个名为 n ，内部进程为 P 的灰箱。值得强调，在灰箱 $n[P]$ 中，无论 n 是否进行移动，进程 P 始终处在运行状态，即：

$$P \longrightarrow Q \quad \Longrightarrow \quad n[P] \longrightarrow n[Q] \tag{2.5}$$

  由于 P 中可能包含多个进程的并置，其中一些进程也是灰箱，因此一般来说，灰箱具有如下的树状嵌套结构：

$$n[P_1 \mid \cdots \mid P_p \mid m_1[\cdots] \mid \cdots \mid m_q[\cdots]] \qquad (P_i \neq n_i[\cdots])$$

  其中 P_1 、 $\cdots$ 、 P_p 都是具体的动作执行进程，都具有 $A\,.\,\cdots$ 的结构[3]，其中 A 为六种基本动作之一。这些进程往往也称为灰箱 n 的 **控制进程** 或 **顶层进程**，负责控制 n 的行为。而 $m_1[\cdots]$ 、 $\cdots$ 、 $m_q[\cdots]$ 等进程都具有灰箱结构，一般称为 n 的 **子灰箱** 。

  由于灰箱演算中的归约操作都是围绕灰箱的嵌套结构进行的，因此掌握上述树状嵌套结构对于理解灰箱演算的操作语义有很大的帮助。灰箱演算中灰箱的移动和打开都是在灰箱内部的顶层进程控制下完成的。 MA 、 SA 和 ROAM 的差异也集中在对这些动作的不同定义。

  另外，灰箱演算的语法对同名灰箱没有任何限制，无论是同层的还是嵌套的灰箱都可拥有同一个名字。甚至可以使用 $!n[P]$ 来描述某种不断可以提供的一次性服务。

类似其它演算系统，在 ROAM 中，使用 $fn(P)$ 来表示进程 P 中的自由名集合。对于能力 M ，由于其中出现的所有名字均为自由名，同样使用 $fn(M)$ 表示 M 中的自由名集合。在进程表达式中，并置构造“ | ”的优先级最低。必要时，可通过括号“ () ”的使用改变操作符的结合次序。例如在进程 $M\,.\,(P \mid Q)$ 、 $(\nu n : T)(P \mid Q)$ 和 $!(P \mid Q)$ 中，如果省略括号，进程的意义将不同。为方便书写，一般将进程 $n[\mathbf{0}]$ 简写为 $n[\,]$ ，将进程 $M\,.\,\mathbf{0}$ 简写为 M 。

能力由顺序执行的动作组成。在 ROAM 中，进程可以执行的动作和协动作共有六种，分三对配合使用，完成归约。动作和协动作的参数都参与具体的归约过程（除没有参数的 $\overline{\texttt{open}}$ 以外）。

[3] 约束构造可通过下文的结构同余关系等价变换到灰箱 n 的外部，此处暂不考虑。

- **移入**（ $\texttt{in}\ n$ 和 $\overline{\texttt{in}}\ n$ ）：动作 $\texttt{in}\ n$ 试图使其所在的灰箱进入与该灰箱并置、且名为 n 的灰箱；动作 $\overline{\texttt{in}}\ n$ 允许一个名为 n 的灰箱进入该进程所在的灰箱。移入动作和移入协动作配合使用，相互配合完成一次移入归约动作，使得一对同层灰箱变成相互嵌套的父子灰箱。例如，下面的归约通过灰箱 m 中的 $\texttt{in}\ n$ 动作和灰箱 n 中的 $\overline{\texttt{in}}\ m$ 动作相互配合完成了 m 进入 n 的过程。

$$m[\texttt{in}\ n\,.\,P_1 \mid P_2] \mid n[\overline{\texttt{in}}\ m\,.\,Q_1 \mid Q_2] \longrightarrow n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] \tag{2.6}$$

- **移出**（ $\texttt{out}\ n$ 和 $\overline{\texttt{out}}\ n$ ）：动作 $\texttt{out}\ n$ 试图使其所在的灰箱移出其上层灰箱 n ；动作 $\overline{\texttt{out}}\ n$ 允许一个名为 n 的子灰箱移出。移出动作的效果与移入动作刚好相反，使得一对父子灰箱变成同层灰箱。下面的例子示意了动作 $\texttt{out}\ n$ 和 $\overline{\texttt{out}}\ m$ 的配合过程。

$$n[m[\texttt{out}\ n\,.\,P_1 \mid P_2] \mid \overline{\texttt{out}}\ m\,.\,Q_1 \mid Q_2] \longrightarrow m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] \tag{2.7}$$

- **打开**（ $\texttt{open}\ n$ 和 $\overline{\texttt{open}}$ ）：动作 $\texttt{open}\ n$ 试图将名为 n 的灰箱打开，使得 n 消失且 n 中的进程与 $\texttt{open}\ n$ 所在位置的其它进程并发运行； $\overline{\texttt{open}}$ 允许所在的灰箱被打开。需要注意，由于打开动作并不一定处在一个灰箱中，动作 $\overline{\texttt{open}}$ 是 ROAM 六个基本动作中唯一没有参数的。例如：

$$\texttt{open}\ n\,.\,P \mid n[\overline{\texttt{open}}\,.\,Q \mid R] \longrightarrow P \mid Q \mid R \tag{2.8}$$

为了突出进程执行的具体动作，用 **Action** 表示 $\mathcal{M}$ 的单个动作子集，即 $\mathbf{Action} \in \mathcal{M}$ 。**Action** 集合的元素通常用大写字母 A 、 A' 等表示。

$$\mathbf{Action} \triangleq \{\texttt{in}\ n,\ \overline{\texttt{in}}\ n,\ \texttt{out}\ n,\ \overline{\texttt{out}}\ n,\ \texttt{open}\ n,\ \overline{\texttt{open}} \mid n \in \mathcal{N}\}$$

类似地，使用 $fn(A)$ 表示 A 中的自由名集合。 $M\,.\,P$ 和 $A\,.\,P$ 各有其使用场合。 $M\,.\,P$ 表示进程具有顺序动作构造，但是不突出其执行的第一个动作的具体内容； $A\,.\,P$ 则相反，强调进程执行的第一个动作为 A 。

§2.2 归约语义

上文已经以例子的方式初步给出了 ROAM 进程归约的一些概念。本节给出 ROAM 进程间归约关系 “ $\longrightarrow$ ” 的形式化定义。这种归约语义来源于 Berry 和 Boudol 的化学抽象机（ chemical abstract machine ）[25]，在 π 演算、灰箱演算等形式化系统中被广泛使用。归约语义的推导规则在很大程度上依赖于进程间不影响进程功能的等价变换——结构同余（ structural congruence ）关系，用 “ $\equiv$ ” 表示，定义为进程集合上满足以下规则的最小二元关系：

(Struct Refl) $\quad P \equiv P$

(Struct Symm) $\quad Q \equiv P \Longrightarrow P \equiv Q$

(Struct Trans) $\quad P \equiv Q,\ Q \equiv R \Longrightarrow P \equiv R$

(Struct Res) $\quad P \equiv Q \Longrightarrow (\nu n : T)P \equiv (\nu n : T)Q$

(Struct Par) $\quad P \equiv Q \Longrightarrow P \mid R \equiv Q \mid R$

(Struct Repl) $P \equiv Q \Longrightarrow !P \equiv !Q$

(Struct Amb) $P \equiv Q \Longrightarrow n[P] \equiv n[Q]$

(Struct Act) $P \equiv Q \Longrightarrow M \,.\, P \equiv M \,.\, Q$

(Struct Par Zero) $P \mid \mathbf{0} \equiv P$

(Struct Par Comm) $P \mid Q \equiv Q \mid P$

(Struct Par Assoc) $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$

(Struct Res Zero) $(\nu n : T)\mathbf{0} \equiv \mathbf{0}$

(Struct Res Res) $(\nu n : T_n)(\nu m : T_m)P \equiv (\nu m : T_m)(\nu n : T_n)P$

(Struct Res Par) $n \notin fn(P) \Longrightarrow (\nu n : T)(P \mid Q) \equiv P \mid (\nu n : T)Q$

(Struct Res Amb) $n \neq m \Longrightarrow (\nu n : T)m[P] \equiv m[(\nu n : T)P]$

(Struct Repl Par) $!P \equiv P \mid !P$

(Struct Repl Zero) $!\mathbf{0} \equiv \mathbf{0}$

(Struct Empty) $\epsilon \,.\, P \equiv P$

(Struct Path) $(M_1 \,.\, M_2) \,.\, P \equiv M_1 \,.\, (M_2 \,.\, P)$

以下是 ROAM 进程的归约规则。前三条规则给出了移入、移出和打开三种基本可归约结构及其归约结果，紧接的三条规则指出归约动作可发生在并置、约束和灰箱构造的内部，这六条规则都在上文的介绍中做了具体的说明。最后一条规则 **(Red Struct)** 是上文没有讲解的，它表示进程可通过结构同余规则进行等价变化来满足其它规则所需的进程结构。

(Red In) $m[\,\mathtt{in}\; n \,.\, P_1 \mid P_2] \mid n[\,\overline{\mathtt{in}}\; m \,.\, Q_1 \mid Q_2] \longrightarrow n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2]$

(Red Out) $n[m[\,\mathtt{out}\; n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\; m \,.\, Q_1 \mid Q_2] \longrightarrow m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2]$

(Red Open) $\mathtt{open}\; n \,.\, P \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] \longrightarrow P \mid Q_1 \mid Q_2$

(Red Par) $\dfrac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$

(Red Res) $\dfrac{P \longrightarrow P'}{(\nu n : T)P \longrightarrow (\nu n : T)P'}$

(Red Amb) $\dfrac{P \longrightarrow P'}{n[P] \longrightarrow n[P']}$

(Red Struct) $\dfrac{P \equiv P' \quad P' \longrightarrow P'' \quad P'' \equiv P'''}{P \longrightarrow P'''}$

沿用传统的记号，下文用关系“ $\longrightarrow^*$ ”表示“ $\longrightarrow$ ”的自反、传递闭包，用关系“ $\longrightarrow^+$ ”表示“ $\longrightarrow$ ”的传递闭包。

§2.3 举例

本节给出一些 ROAM 的例子，来进一步阐明 ROAM 的归约特性和表达能力。

§2.3.1 进程同步

并发系统中必然需要使用一些手段控制共享资源的访问。早在文献 [5] 中就给出了用 MA 实现上锁和解锁的例子：

$$\begin{aligned} acquire\ n\ .\ P &\triangleq \mathtt{open}\ n\ .\ P \\ release\ n\ .\ P &\triangleq n[\,] \mid P \end{aligned}$$

通过上述定义，下面的方法可以实现两个进程的同步（ shaking hand ）[5]：

$$acquire\ n\ .release\ m\ .\ P \mid release\ n\ .acquire\ m\ .\ Q \tag{2.9}$$

在 ROAM 中，除了可以定义类似的 $acquire$ 和 $release$ 构造外，还可以直接使用灰箱的打开动作和协动作更加方便地实现进程同步：

$$\begin{aligned} ShhL\ n\ .\ P &\triangleq \mathtt{open}\ n\ .\ P \\ ShhR\ n\ .\ P &\triangleq n[\overline{\mathtt{open}}\ .\ P] \end{aligned}$$

这样，下式即可实现进程 P 、 Q 间的同步。

$$ShhL\ n\ .\ P \mid ShhR\ n\ .Q \tag{2.10}$$

§2.3.2 进程竞争

同样，对于进程之间的竞争，仿照文献 [56] 中的进程内部选择（ internal choice ）操作，可定义如下基于 ROAM 的进程内部选择构造：

$$P \oplus Q \triangleq (\nu r)(\mathtt{open}\ r\ .\ P \mid \mathtt{open}\ r\ .\ Q \mid r[\overline{\mathtt{open}}]\,)$$

这里，灰箱名 r 的选择满足 $r \notin fn(P) \cup fn(Q)$ ，同时省略了约束构造中的类型符号。 $P \oplus Q$ 的归约存在两种可能：

$$\begin{aligned} & P \oplus Q && (2.11)\\ = \ & (\nu r)(\mathtt{open}\ r\ .\ P \mid \mathtt{open}\ r\ .\ Q \mid r[\overline{\mathtt{open}}]\,) && (2.12)\\ \longrightarrow \ & (\nu r)(P \mid \mathtt{open}\ r\ .\ Q) && (2.13)\\ \equiv \ & P \mid (\nu r)(\mathtt{open}\ r\ .\ Q) && (2.14)\\ \approx \ & P && (2.15) \end{aligned}$$

$$\begin{aligned} & P \oplus Q && (2.16)\\ = \ & (\nu r)(\mathtt{open}\ r\ .\ P \mid \mathtt{open}\ r\ .\ Q \mid r[\overline{\mathtt{open}}]\,) && (2.17)\\ \longrightarrow \ & (\nu r)(\mathtt{open}\ r\ .\ P \mid Q) && (2.18)\\ \equiv \ & (\nu r)(\mathtt{open}\ r\ .\ P) \mid Q && (2.19)\\ \approx \ & Q && (2.20) \end{aligned}$$

通过上述归约过程可以看出，进程 $P \oplus Q$ 要么归约为 P （式 (2.12 ~ 2.15) ），要么归约为 Q （式 (2.17 ~ 2.20) ），实现了 P 和 Q 的一种不确定的竞争。这里 $\approx$ 表示两个进程在外部看来完全等价，无法区分。在后面的进程等价性章节中将会详细介绍。

注意，上述的内部选择构造还不能实现 CCS 或 π 演算中的选择子 “ + ” 的功能。在文献 [5] 中给出了用 MA 实现根据不同灰箱的存在选择不同进程的构造。由于 MA 的动作不会对被测试灰箱产生影响，因此容易实现。在 ROAM 中，可在相应的被测试灰箱中添加特定的复制协动作构造以得到类似的效果。另外一种用打开灰箱作为选择依据的选择构造可定义如下：

$$\begin{aligned} n \Rightarrow P + m \Rightarrow Q \quad \triangleq \quad & (\nu r)(\mathtt{open}\, n \,.\, (n[\overline{\mathtt{open}}] \mid \mathtt{open}\, r \,.\, (\mathtt{open}\, n \mid P)) \\ & \mid \mathtt{open}\, m \,.\, (m[\overline{\mathtt{open}}] \mid \mathtt{open}\, r \,.\, (\mathtt{open}\, m \mid Q)) \\ & \mid r[\overline{\mathtt{open}}]) \end{aligned}$$

在以上定义中，动作 $\mathtt{open}\ n$（或者 $\mathtt{open}\ m$ ）后又出现的 $n[\overline{\mathtt{open}}]$（或者 $m[\overline{\mathtt{open}}]$ ）是为了防止在唯一的 $r[\overline{\mathtt{open}}]$ 已经被用掉后， $\mathtt{open}\, n$（或者 $\mathtt{open}\ m$ ）又消耗掉一个外部的 $n[\overline{\mathtt{open}}]$（或 $m[\overline{\mathtt{open}}]$ ）。容易看出该选择构造满足以下归约特性：

$$(n \Rightarrow P + m \Rightarrow Q) \mid n[\overline{\mathtt{open}}] \quad \longrightarrow^* \quad P \mid (\nu r)(\mathtt{open}\, m \,.\, (m[\overline{\mathtt{open}}] \mid \mathtt{open}\, r \,.\, (\mathtt{open}\, m \mid Q))) \tag{2.21}$$

$$(n \Rightarrow P + m \Rightarrow Q) \mid m[\overline{\mathtt{open}}] \quad \longrightarrow^* \quad Q \mid (\nu r)(\mathtt{open}\, n \,.\, (n[\overline{\mathtt{open}}] \mid \mathtt{open}\, r \,.\, (\mathtt{open}\, n \mid P))) \tag{2.22}$$

以式 (2.21) 为例，其结果的后半部分 $(\nu r)(\mathtt{open}\ m \,.\, (m[\overline{\mathtt{open}}] \mid \mathtt{open}\ r \,.\, (\mathtt{open}\ m \mid Q)))$ 由于 $r[\overline{\mathtt{open}}]$ 已经用掉，不可能再释放出 Q 。如果外部只可能存在 $m[\overline{\mathtt{open}}]$ 的形式与 $\mathtt{open}\, m$ 发生归约，则该后半部分与 0 等价。然而，该选择构造尚不能满足归约前后进程的等价性，除非限制 n 和 m 在其它地方只能以 $n[\overline{\mathtt{open}}]$ 和 $m[\overline{\mathtt{open}}]$ 的形式出现。

§2.3.3 换名

最初 Cardelli 和 Gordon 在提出 MA 时就给出了灰箱换名的例子 [5] ：

$$n\ \mathtt{be}\ m \,.\, P \quad \triangleq \quad m[\mathtt{out}\ n \,.\, \mathtt{open}\ n \,.\, P] \mid \mathtt{in}\ m$$

这样，灰箱 n 中的进程 $n\ \mathtt{be}\ m \,.\, P$ 就可使灰箱的名字 n 通过三次归约变成一个另外的名字 m ：

$$\begin{aligned} & n[n\ \mathtt{be}\ m \,.\, P \mid Q] && (2.23) \\ = \quad & n[m[\mathtt{out}\ n \,.\, \mathtt{open}\ n \,.\, P] \mid \mathtt{in}\ m \mid Q] && (2.24) \\ \longrightarrow \quad & m[\mathtt{open}\ n \,.\, P] \mid n[\mathtt{in}\ m \mid Q] && (2.25) \\ \longrightarrow \quad & m[\mathtt{open}\ n \,.\, P \mid n[Q]\,] && (2.26) \\ \longrightarrow \quad & m[P \mid Q] && (2.27) \end{aligned}$$

在 ROAM 中，很容易定义类似的换名构造：

$$n\ \ \mathtt{be}\ \ m \,.\, P \quad \triangleq \quad m[\mathtt{out}\ n \,.\, \overline{\mathtt{in}}\ n \,.\, \mathtt{open}\ n] \mid \overline{\mathtt{out}}\ m \,.\, \mathtt{in}\ m \,.\, \overline{\mathtt{open}} \,.\, P$$

同样有：

$$n[n\ \mathtt{be}\ m \,.\, P \mid Q] \longrightarrow^* m[P \mid Q] \tag{2.28}$$

需要指出，由于 MA 中存在强干扰问题，式 (2.23 ∼ 2.27) 可以顺利进行换名必须建立在大量的限制条件之上。例如：外部不能存在并发运行的同名灰箱 n 或 m，外部不能有试图进入 n 的灰箱或打开 n 、 m 的进程，进程 Q 不能使 n 在换名过程中发生意外移动等。这些限制使得 MA 的换名构造应用价值不强。即便在 SA 中，换名的正确性还需要比较复杂的协议保证。然而，用 ROAM 实现灰箱换名时，不但限制条件大大减少，而且只需相对简单的构造就可保证换名的正确性。第六章将会详细介绍换名构造前后进程等价的条件，同时还将讨论另外一个非常经典的防火墙跨越例子。

本章小结

本章基于概述的非正式分析，形式化地给出了 ROAM 的语法和基于结构同余关系的归约语义，并通过一些例子，对 ROAM 的操作语义和表达能力给予了进一步的说明。后续的章节将以此为基石，研究 ROAM 的演化类型系统、标号转移语义并研究进程的代数性质和表达能力。

第三章　演化类型系统

类型系统（ type system ）通过一套明确定义的方法限定进程表达式的构造，保证在归约过程中，进程始终保持某种期望的属性 [57]。目前对灰箱演算的类型系统已有较多的研究工作，例如：侧重于 MA 的基本类型问题（如：移动性和消息传递类型）的 [27, 28, 29]，侧重于 MA 、 SA 安全性问题的 [34, 31, 32]，研究线程数和代数理论的 [16, 58] 以及侧重于消息通讯的多态性 [54] 等。

在灰箱演算类型系统的研究中，一般有三个基本的研究对象：进程类型、灰箱名类型和能力类型 [1]。进程类型刻划了进程的一些基本特性，如移动性、线程数和消息传递类型；灰箱名的类型用于说明该灰箱的特性，如：记录该灰箱中允许运行进程的类型；由于组成能力的各动作是由动作的种类和动作的参数（即：灰箱名）所组成的，因此能力类型的确定主要通过动作种类对应的特性和灰箱名的类型来实现。

对打开动作的处理是灰箱演算类型系统中研究的一个重要问题。与移入和移出动作不同，打开动作将在现有的进程中引入新的进程，因此对执行打开动作进程的类型定义通常要包含被打开灰箱中允许运行进程的性质。举个简单的例子：假设灰箱 n 中允许运行的进程具有移动特性，那么进程 $\mathtt{open}\, n\,.\,P$ 的类型也必须至少包含移动特性，因为动作 $\mathtt{open}\, n$ 的执行将引入 n 中的进程，因此可能具有 n 中运行进程的所有特性，包括其移动特性。

上述规则简单、直观，被广泛运用于现有各灰箱演算的类型系统中 [27, 28, 29, 34, 31, 58]。事实上，这种方法在刻划进程特性方面显得比较粗糙。灰箱中运行的进程在打开前后往往具有不同的特性，例如以下的 ROAM 进程：

$$n[\,\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.\,\mathbf{0}] \mid m[\,\overline{\mathtt{in}}\, n\,.\,\mathtt{open}\, n] \tag{3.1}$$

此时，灰箱 n 中的进程 $\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.\,\mathbf{0}$ 具有移动特性，但当 n 在 m 中被打开后，却只有一个静止进程进入 m，灰箱 n 的移动特性并未引入 m。这种进程类型在打开前后发生变化的情况称为灰箱演算的 **类型演化问题**（ type evolution problem ） [59]。刻划进程的类型演化对于精确表达进程的动态特性，进而深入研究进程的代数性质至关重要。

本章提出了 ROAM 中的一套演化类型系统 ETS-MT ，利用特殊的类型表达式区分了进程类型表达式中的 **当前类型** 和 **未来类型** 部分，分别表示打开操作前后进程的不同类型特性，配合特殊的能力类型构造，有效地解决了灰箱演算中的类型演化问题。本章的主要内容围绕 ETS-MT展开，安排如下：首先，从类型系统所需描述的进程属性开始，给出了可以描述 **移动性**（ mobilty ）和 **线程数**（ threads ）两种属性的类型语法，并解释了进程类型表达式中针对类型演化问题的特定结构和作用。在此基础上，定义了两个分别针对进程结构中的顺序动作和并置构造的类型运算符，以方便进程类型的计算。之后，介绍了 ETS-MT中进程类型的子类型（ subtyping ）关系，使得类型限制较强的进程也可以出现在类型限制较弱的位置，进一步增强了 ETS-MT的适应能力。在上述定义的基础上，给出了对进程和能力在给定类型环境下的类型判定规则（ typing rules ），并证明通过该类型判定规则得到的进程类型的有效性（ validity ）、对结构同余关系的一致性（ subject congruence ）和对归约关系的一致性（ subject reduction ）。由于存在子类型关系，除了类型判定规则，本章还

[1] 这里的能力特指动作串，有别于文献 [27, 16] 中既表示动作串，又表示灰箱名的 capability 一词。

给出了 ETS-MT中的最小类型算法（ typing algorithm ），可以在给定的类型环境下，计算进程、灰箱的最小类型，同时证明了最小类型算法的正确性（ soundness ）和完备性（ completeness ）。最后，给出了用最小类型算法计算非移动服务器类型的实例，说明了 ETS-MT在刻划类型演化方面的能力。

§3.1 类型语法

ETS-MT主要刻划了进程的 **移动性**（ mobility ）和 **线程数**（ threads ）。移动性给出进程是否会执行导致灰箱移动的动作（ in 和 out ），进程的移动性为移动（ $\curvearrowright$ ）表示包含，反之，则进程的移动性为非移动（ $\underline{\vee}$ ）；线程数表示了进程中可并发执行动作个数。例如，静止进程的线程数为 0 ，进程 $\mathtt{in}\, n\,.\,\overline{\mathtt{out}}\, m\,.\,\mathtt{out}\, n$ 中的动作为顺序执行，其线程数为 1（单线程），而进程 $\overline{\mathtt{in}}\, n \mid \overline{\mathtt{out}}\, n$ 则是多线程，其线程数用符号 ω 表示。

用 Z 表示移动性、用 Y 表示线程数， ETS-MT中进程类型 T 和能力类型 W 的语法定义如下：
$(Z \in \mathcal{Z}, Y \in \mathcal{Y}, U \in \mathcal{U}, T \in \mathcal{T}, W \in \mathcal{W})$

Z	$::=$	$\underline{\vee}$	移动性：	**固定**
		$\curvearrowright$		**移动**
Y	$::=$	0	线程数：	**无线程**
		1		**单线程**
		ω		**多线程**
U	$::=$	Z^Y	准类型	
T	$::=$	$\bot$	进程类型：	**非法进程**
		U		**简单类型**
		$U[T]$		**演化类型**
W	$::=$	$-$	能力类型：	**空缺位置标记**
		$U \cdot_t W$		**顺序结构**
		$T \mid_t W$		**并发结构**
		$U[W]$		**演化结构**

上述类型语法中， U 称为 **准类型**（ pretype ），用于汇总进程所具有的各种属性。 ETS-MT主要描述进程的移动性和线程数两种属性，准类型的形式为 Z^Y ， Z 表示进程的移动性、 Y 表示线程数。 ETS-MT中，进程类型和灰箱名类型都采用 T 表示。进程 P（或灰箱 n ）具有类型 T 简写为 $P : T$（或 $n : T$ ）。为了刻划进程类型 [2] 的演化特征，进程类型 T 具有如下三种构造：非法的进程类型用记号 $\bot$ 表示；不执行 $\overline{\mathtt{open}}$ 动作（即不发生打开前后类型演化）的进程类型称为 **简单类型**，直接用 U 表示；如果进程可能执行 $\overline{\mathtt{open}}$ 动作，则其类型表达式具有 **演化类型** 构造 $U[T]$，表示进程在 $\overline{\mathtt{open}}$ 动作执行前具有简单类型 U ，在 $\overline{\mathtt{open}}$ 动作执行后进程的剩余部分类型为 T ， U 称为其 **当前类型**， T 称为其 **未来类型**。

例如，考虑进程 $\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,!\overline{\mathtt{in}}\ m$ 。动作 $\overline{\mathtt{open}}$ 执行前，进程是移动的、单线程的 [3] ，即

[2] 为方便行文，以下在不引起误会的情况下，统称 T 为 “进程类型”，而不是 “进程或灰箱类型”。
[3] 此处暂时不考虑能力的特殊上下文语法。

$\texttt{in}\ m : \curvearrowright^1$ 。之后，进程是非移动的、多线程的，即 $!\overline{\texttt{in}}\ m : \underline{\vee}^\omega$ 。因此，整个进程的类型为 $\curvearrowright^1 [\underline{\vee}^\omega]$ 。通过使用演化类型构造，在打开包含演化类型进程的灰箱时，打开动作所释放的进程类型只与灰箱的未来类型有关，而与其当前类型无关。当前类型和未来类型的划分实现了进程类型变化的刻划。

ETS-MT中的能力类型用 W 表示。一个能力 M 的类型为 W 简写为 $M : W$ 。由于 W 的定义采用了特殊的上下文方式，并用到了两个类型运算符 $\cdot_t$ 和 $|_t$ ，下一小节将专门在引入这两个运算符的基础上，对能力类型的构造进行具体的说明。

在 T 的定义中引入非法进程类型符号 $\perp$ 是为了下面运算符定义的方便。另外这里规定 $U[\perp] \equiv \perp$ ，即一个合法的类型表达式必须具有 $U_0[U_1 \cdots [U_n] \cdots]$ 的一般形式，其中 $n \geq 0$ 。

§3.2 类型运算符

为方便推导由顺序动作（ . ）和并置（ | ）所构成进程的类型特性，在上述类型语法中，使用了两个预定义的类型运算符： $\cdot_t$ 和 $|_t$ 。

首先，分析进程 $M\,.\,P$ 的类型推导。假设已知 $P : U[T]$ 且 $M : U'$ （这里先不考虑能力类型 W 所具有的上下文结构，而先把能力类型简单看作具有移动性和线程数两种属性的准类型、 M 中也不包含 open 和 $\overline{\texttt{open}}$ 这两个特殊动作）。这时 M 只对进程 $M\,.\,P$ 的当前类型产生影响，因此，$M\,.\,P$ 的类型可以推导为 $(U' \cdot_u U)[T]$ 。运算符 $\cdot_u$ 用于得到不同特性通过顺序动作“ . ”连接后呈现的新特性。例如：非移动和移动两种操作顺序执行时，表现出移动的特性，单线程和单线程循序执行，仍具有单线程的特性。运算符 $\cdot_u$ 定义在两个更加基本的运算符 $\cdot_z$ 和 $\cdot_y$ 之上。而整个由 U' 和 $U[T]$ 得到 $(U' \cdot_u U)[T]$ 的运算用 $\cdot_t$ 来表示。以下是这四个运算符的具体定义。

$\cdot_z : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$

$\cdot_z$	$\underline{\vee}$	$\curvearrowright$
$\underline{\vee}$	$\underline{\vee}$	$\curvearrowright$
$\curvearrowright$	$\curvearrowright$	$\curvearrowright$

$\cdot_y : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$

$\cdot_y$	0	1	ω
0	0	1	ω
1	1	1	ω
ω	ω	ω	ω

$\cdot_u : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$

$$Z_1^{Y_1} \cdot_u Z_2^{Y_2} \triangleq (Z_1 \cdot_z Z_2)^{(Y_1 \cdot_y Y_2)}$$

$\cdot_t : \mathcal{U} \times \mathcal{T} \to \mathcal{T}$

$$\begin{aligned} U \cdot_t \perp &\triangleq \perp \\ U \cdot_t U' &\triangleq U \cdot_u U' \\ U \cdot_t U'[T] &\triangleq (U \cdot_u U')[T] \end{aligned}$$

其次，分析进程 $P_1 \mid P_2$ 的类型推导过程。假设已知 $P_1 : T_1$ 、 $P_2 : T_2$ ，分以下四种情况讨论。

(1). T_1 、T_2 均为简单类型，即： $T_1 = U_1$ 且 $T_2 = U_2$ 。此时 $P_1 \mid P_2$ 的类型可用 $U_1 \mid_u U_2$ 表示。类似上文 $\cdot_u$ 的定义，运算符 $\mid_u$ 根据并置构造的特点，定义在运算符 $\mid_z$ 和 $\mid_y$ 之上。例如，两个单线程的进程并置结果就是多线程的。

(2). T_1 、 T_2 中有一个为简单类型，一个为演化类型，不妨设 $T_1 = U$ 、 $T_2 = U'[T]$ 。此时 P_2 在动作 $\overline{\texttt{open}}$ 执行前表现为 U' 的特性，之后表现为 T ，而并置的结果使得 $P_1 \mid P_2$ 中的 P_1 的特性在 P_2 中动作 $\overline{\texttt{open}}$ 的执行前后都可能体现，因此 P_1 的特性 U 要同时作用在 U' 和 T 上，以得到 $P_1 \mid P_2$ 的类型。

(3). T_1 、 T_2 都为演化类型。则 $P_1 \mid P_2$ 的类型很难推导，因为两个并置的 $\overline{\texttt{open}}$ 动作执行的先后不同，进程的类型也不相同，要同时考虑所有的情况过于复杂，且实际意义不大。因此在 ETS-MT中，这种情况下 $P_1 \mid P_2$ 的类型定义为非法的（ $\perp$ ） [4] 。

(4). T_1 、 T_2 中有一个非法。此时， $P_1 \mid P_2$ 的类型也是非法的。

$\mid_z : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$

$\mid_z$	$\underline{\vee}$	$\curvearrowright$
$\underline{\vee}$	$\underline{\vee}$	$\curvearrowright$
$\curvearrowright$	$\curvearrowright$	$\curvearrowright$

$\mid_y : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$

$\mid_y$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$\mid_u : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$

$$Z_1^{Y_1} \mid_u Z_2^{Y_2} \quad \triangleq \quad (Z_1 \mid_z Z_2)^{(Y_1 \mid_y Y_2)}$$

$\mid_t : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$

$$\begin{aligned}
U \mid_t U' &= U \mid_u U' \\
U \mid_t U'[T] &= (U \mid_u U')[U \mid_t T] \\
U'[T] \mid_t U &= (U' \mid_u U)[T \mid_t U] \\
U_1[T_1] \mid_t U_2[T_2] &= \perp \\
\perp \mid_t T &= \perp \\
T \mid_t \perp &= \perp
\end{aligned}$$

ETS-MT中比较特殊的是能力类型的语法构造。一个能力的类型 W 表现为进程类型的一个上下文（ context ）——只要在 W 中唯一的空缺位置（ $-$ ）填上一个进程类型 T ，其结果就是一个

[4] 理论上，这种情况也是可以给出类型的。假定 $P_1 \mid P_2 : T_3$ 且 T_1 、 T_2 和 T_3 的具体构造如下： $T_i = U_{i0}[U_{i1}[\cdots[U_{in_i}]\cdots]]$ ，这里 $i = 1,2,3$ 。则 $U_{3k} = \texttt{MAX}(U_{1i} \mid_u U_{2j})_{i+j=k}$ ， $k = 0..(n_1 + n_2)$ 。但这种推广几乎没有实用价值。

新的进程类型 T' 。这一点从能力在进程构造中的作用来看是非常自然的：任何能力 M 需要一个进程 P 来构成一个新的进程 $M.P$ 。下文用 $W(T)$ 表示将 T 填入 W 中的空缺位置并经过 $\cdot_t$ 和 $|_t$ 运算所得到的结果，显然 $W(T) \in \mathcal{T}$ 。另外，用 $W(W')$ 表示将 W' 填入 W 中的空缺位置得到的结果，$W(W') \in \mathcal{W}$ 。

§3.3 子类型关系

程序设计语言中，类型的大小关系（子类型关系）应用非常广泛。例如：一个整型（ **int** ）的表达式可以自动转换为一个浮点型（ **float** ）的表达式。 ETS-MT也定义了进程类型集合 $\mathcal{T}$ 上元素的大小关系“ $\leq$ ”。定义这种大小关系的结果使得下面的基本事实成立：

$$\frac{P : T \quad T \leq T'}{P : T'}$$

集合 $\mathcal{T}$ 上的子类型关系建立在集合 $\mathcal{Z}$ 、 $\mathcal{Y}$ 和 $\mathcal{U}$ 的子类型关系基础之上。在不引起混淆的前提下，这些集合上的子类型关系均使用符号“ $\leq$ ”表示。另外，在没有特殊说明时，这些集合上的子类型关系都是自反的和传递的。

- **集合 $\mathcal{Z}$ 上的子类型关系**： $\mathcal{Z}$ 的两个元素具有如下的大小关系：

  $$\underline{\vee} \leq \curvearrowright$$

  也就是说，一个非移动的进程可以看作是移动的（例如： $\mathbf{0} : \underline{\vee}^0$ 可以看作 $\mathbf{0} : \curvearrowright^0$ ），但是一个移动的进程不能被看作是非移动的。
- **集合 $\mathcal{Y}$ 上的子类型关系**： $\mathcal{Y}$ 的三个元素具有如下的大小关系：

  $$0 \leq 1 \leq \omega$$

  同样道理，线程数少的可以看作是线程数多的情况的特例，但是线程数多的就不能看作是线程数少的一种特例。
- **集合 $\mathcal{U}$ 上的子类型关系**： $\mathcal{U}$ 的子类型关系由以下规则给出：

  $$Z^Y \leq Z'^{Y'} \iff Z \leq Z' \wedge Y \leq Y'$$

  这是直接利用 $\mathcal{Z}$ 和 $\mathcal{Y}$ 的子类型关系得到的结果。
- **集合 $\mathcal{T}$ 上的子类型关系**： $\mathcal{T}$ 的子类型关系由以下两条规则给出（为了区别 $\mathcal{T}$ 和 $\mathcal{U}$ 的不同子类型关系，第一条规则使用了不同的符号“ $\leq_T$ ”和“ $\leq_U$ ”）：

  $$U \leq_T U' \iff U \leq_U U'$$

  $$U[T] \leq U'[T'] \iff U \leq U' \wedge T \leq T'$$

  从上述规则可以看出，如果两个进程类型具有相同的层数（即：嵌套的“ [ ] ”数目），则这两个类型的大小取决于相同层上的准类型的大小关系。否则，如果两个进程类型的层数不同，则它们的大小关系没有可比性。

以下是一些子类型关系的例子：

$$\underline{\vee}^0 \quad \leq \quad \underline{\vee}^0 \tag{3.2}$$

$$\underline{\vee}^0 \quad \leq \quad \curvearrowright^\omega \tag{3.3}$$

$$\curvearrowright^0 \quad \leq \quad \curvearrowright^1 \quad \leq \quad \curvearrowright^\omega \tag{3.4}$$

$$\underline{\vee}^0[\curvearrowright^1] \quad \leq \quad \underline{\vee}^0[\curvearrowright^\omega] \quad \leq \quad \curvearrowright^\omega [\curvearrowright^\omega] \tag{3.5}$$

$$\underline{\vee}^0[\underline{\vee}^0[\underline{\vee}^0]] \quad \leq \quad \curvearrowright^\omega [\curvearrowright^\omega [\curvearrowright^\omega]] \tag{3.6}$$

下面给出了关于类型运算符 $\cdot_t$ 和 $|_t$ 的一些性质。这些性质可直接通过 $\cdot_t$ 和 $|_t$ 的定义得到。

性质 3.1 以下关于 $\cdot_z$, $\cdot_y$, $\cdot_u$, $\cdot_t$ 的性质成立：

(Dot Zero Z) $\underline{\vee} \cdot_z Z = Z$

(Dot Zero Y) $0 \cdot_y Y = Y$

(Dot Zero U) $\underline{\vee}^0 \cdot_u U = U$

(Dot Zero T) $\underline{\vee}^0 \cdot_t T = T$

(Dot Symm Z) $Z_1 \cdot_z Z_2 = Z_2 \cdot_z Z_1$

(Dot Symm Y) $Y_1 \cdot_y Y_2 = Y_2 \cdot_y Y_1$

(Dot Symm U) $U_1 \cdot_u U_2 = U_2 \cdot_u U_1$

(Dot Assoc Z) $(Z_1 \cdot_z Z_2) \cdot_z Z_3 = Z_1 \cdot_z (Z_2 \cdot_z Z_3)$

(Dot Assoc Y) $(Y_1 \cdot_y Y_2) \cdot_y Y_3 = Y_1 \cdot_y (Y_2 \cdot_y Y_3)$

(Dot Assoc U) $(U_1 \cdot_u U_2) \cdot_u U_3 = U_1 \cdot_u (U_2 \cdot_u U_3)$

(Dot Assoc T) $(U_1 \cdot_t U_2) \cdot_t T = U_1 \cdot_t (U_2 \cdot_t T)$

(Dot Strict Z) $Z_1 \leq Z_2 \Longrightarrow Z_1 \cdot_z Z_3 \leq Z_2 \cdot_z Z_3$

(Dot Strict Y) $Y_1 \leq Y_2 \Longrightarrow Y_1 \cdot_y Y_3 \leq Y_2 \cdot_y Y_3$

(Dot Strict U) $U_1 \leq U_2 \Longrightarrow U_1 \cdot_u U_3 \leq U_2 \cdot_u U_3$

(Dot Strict T) $U_1 \leq U_2 \Longrightarrow U_1 \cdot_t T \leq U_2 \cdot_t T$

性质 3.2 以下关于 $|_z$, $|_y$, $|_u$, $|_t$ 的性质成立：

(Par Zero Z) $\underline{\vee} |_z Z = Z$

(Par Zero Y) $0 |_y Y = Y$

(Par Zero U) $\underline{\vee}^0 |_u U = U$

(Par Zero T) $\underline{\vee}^0 |_t T = T$

(Par Symm Z) $Z_1 |_z Z_2 = Z_2 |_z Z_1$

(Par Symm Y) $Y_1 |_y Y_2 = Y_2 |_y Y_1$

(Par Symm U) $U_1 |_u U_2 = U_2 |_u U_1$

(Par Symm T) $T_1 |_t T_2 = T_2 |_t T_1$

(Par Assoc Z) $(Z_1 |_z Z_2) |_z Z_3 = Z_1 |_z (Z_2 |_z Z_3)$

(Par Assoc Y) $(Y_1 |_y Y_2) |_y Y_3 = Y_1 |_y (Y_2 |_y Y_3)$

(Par Assoc U) $(U_1 \mid_u U_2) \mid_u U_3 = U_1 \mid_u (U_2 \mid_u U_3)$

(Par Assoc T) $(T_1 \mid_t T_2) \mid_t T_3 = T_1 \mid_t (T_2 \mid_t T_3)$

(Par Strict Z) $Z_1 \leq Z_2 \Longrightarrow Z_1 \mid_z Z_3 \leq Z_2 \mid_z Z_3$

(Par Strict Y) $Y_1 \leq Y_2 \Longrightarrow Y_1 \mid_y Y_3 \leq Y_2 \mid_y Y_3$

(Par Strict U) $U_1 \leq U_2 \Longrightarrow U_1 \mid_u U_3 \leq U_2 \mid_u U_3$

(Par Strict T) $T_1 \leq T_2 \Longrightarrow T_1 \mid_t T_3 \leq T_2 \mid_t T_3$

这些性质说明了类型运算符 $\cdot_t$ 和 $\mid_t$ 的一些基本特性。在下面类型判定规则的证明中，将用到这些性质。

§3.4 类型判定规则

本节介绍 ETS-MT的类型判定规则（ typing rules ），这些规则用于在一个给定的类型环境下推导出合法进程 P 和合法能力 W 的类型。

类型判定首先要给出判定所依据的条件，即类型环境（ typing environment ）。一个类型环境通常用 Γ 表示，它具有如下的基本结构： $n_1 : T_1, \cdots, n_k : T_k$ ，其中 $n_1, \cdots, n_k$ 为互不相同的灰箱名，用 $dom(\Gamma)$ 表示这些灰箱名组成的集合。一个空的环境用 ϕ 表示。

ETS-MT的类型判定（ judgement ）有以下四种形式：

(1). $\Gamma \vdash \diamondsuit$ ：表示 Γ 为一个正确的类型环境，即 Γ 中的灰箱名互不重合。

(2). $\Gamma \vdash n : T$ ：表示给定类型环境 Γ ，灰箱名 n 具有类型 T 。

(3). $\Gamma \vdash M : W$ ：表示给定类型环境 Γ ，能力 M 具有类型 W 。

(4). $\Gamma \vdash P : T$ ：表示给定类型环境 Γ ，进程 P 具有类型 T 。

ETS-MT中的类型判定规则如下，这些规则分成三个部分：

- **第一部分** 环境和灰箱名类型判定规则：

(Env Empty) $$\frac{}{\phi \vdash \diamondsuit}$$

(Env Intro) $$\frac{\Gamma \vdash \diamondsuit,\ n \notin dom(\Gamma)}{\Gamma,\ n : T \vdash \diamondsuit}$$

(ET Name) $$\frac{\Gamma,\ n : T \vdash \diamondsuit}{\Gamma,\ n : T \vdash n : T}$$

- **第二部分** 能力类型判定规则：

(ET Empty) $$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \epsilon : -}$$

(ET Cap Mbl) $$\frac{\Gamma \vdash n : T,\ M \in \{\mathtt{in}\ n, \mathtt{out}\ n\}}{\Gamma \vdash M : \curvearrowright^1 \cdot_t -}$$

(ET Cap Imm) $$\frac{\Gamma \vdash n : T,\ M \in \{\overline{\mathtt{in}}\ n, \overline{\mathtt{out}}\ n\}}{\Gamma \vdash M : \underline{\vee}^1 \cdot_t -}$$

(ET Open) $$\frac{\Gamma \vdash n : U[T]}{\Gamma \vdash \mathtt{open}\ n : \underline{\vee}^1 \cdot_t (T \mid_t -)}$$

(ET Co-open) $$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \overline{\texttt{open}} : \underline{\vee}^1[-]}$$

(ET Path) $$\frac{\Gamma \vdash M_1 : W_1,\ \Gamma \vdash M_2 : W_2}{\Gamma \vdash M_1 \,.\, M_2 : W_1(W_2)}$$

- **第三部分** 进程类型判定规则：

(ET Inact) $$\frac{\Gamma \vdash \diamondsuit}{\Gamma \vdash \mathbf{0} : \underline{\vee}^0}$$

(ET Act) $$\frac{\Gamma \vdash M : W,\ \Gamma \vdash P : T,\ W(T) \neq \bot}{\Gamma \vdash M \,.\, P : W(T)}$$

(ET Par) $$\frac{\Gamma \vdash P : T_1,\ \Gamma \vdash Q : T_2,\ T_1 \mid_t T_2 \neq \bot}{\Gamma \vdash P \mid Q : T_1 \mid_t T_2}$$

(ET Repl) $$\frac{\Gamma \vdash P : T,\ T \mid_t T \neq \bot}{\Gamma \vdash\ !P : T \mid_t T}$$

(ET Res) $$\frac{\Gamma,\ n : T_1 \vdash P : T_2}{\Gamma \vdash (\nu n : T_1) P : T_2}$$

(ET Amb) $$\frac{\Gamma \vdash n : T,\ \Gamma \vdash P : T}{\Gamma \vdash n[P] : \underline{\vee}^0}$$

(ET Sub) $$\frac{\Gamma \vdash P : T_1,\ T_1 \leq T_2}{\Gamma \vdash P : T_2}$$

通过操作符 $\cdot_t$ 、 $\mid_t$ 和上下文代入操作 $W(T)$ 、 $W(W')$ 的使用， ETS-MT中的类型判定规则显得比较简单。

在第一部分中，前面的两个规则互相配合，用来保证类型环境中灰箱名不会重复。第三条规则规定了从类型环境中可以直接取出灰箱名的类型。

第二部分根据能力的语法构造逐条给出能力类型的判定方法。每个能力的类型都具有上下文结构。ϵ 的类型直接就是一个空缺位置($-$)。四种移动动作的类型为 $Z^1 \cdot_t -$ ，这里根据动作是否会引入移动性， Z 可取移动 $\curvearrowright$ ($\texttt{in}$ 和 $\texttt{out}$)或者非移动 $\underline{\vee}$ ($\overline{\texttt{in}}$ 和 $\overline{\texttt{out}}$)。动作 $\texttt{open}$ 和 $\overline{\texttt{open}}$ 的类型由于涉及类型演化问题，需要特殊对待。首先，假定 $P : T$ ，考虑进程 $\overline{\texttt{open}} \,.\, P$ 的类型。由于在打开操作执行前，进程只能执行一个非移动操作 $\overline{\texttt{open}}$ ，在打开操作执行后，新进程 P 具有类型 T 。因此可以认为，进程 $\overline{\texttt{open}} \,.\, P$ 的当前类型为 $\underline{\vee}^1$ ，未来类型为 T ，即： $\overline{\texttt{open}} \,.\, P : \underline{\vee}^1[T]$ 。故此，动作 $\overline{\texttt{open}}$ 的类型规定为 $\underline{\vee}^1[-]$ 。其次，假定 $P : T$ ，考虑进程 $\texttt{open}\, n \,.\, P$ 的类型。为了使 $\texttt{open}\, n$ 操作可以顺利完成，灰箱 n 的类型必须具有演化结构 $U[T']$ ，且在打开操作完成后， n 中释放出的进程 $P' : T'$ 将和进程 P 并置运行，即结果进程的类型为 $T' \mid_t T$ 。而 $\texttt{open}\, n$ 又是一个非移动的动作，因此 $\texttt{open}\, n \,.\, P$ 的类型应该为 $\underline{\vee}^1 \cdot_t (T' \mid_t T)$ 。故此，在灰箱 n 的类型为 $U[T']$ 的前提下，动作 $\texttt{open}\, n$ 的类型规定为 $\underline{\vee}^1 \cdot_t (T' \mid_t -)$ 。最后，能力串 $M_1 \,.\, M_2$ ($M_1 : W_1$ 、 $M_2 : W_2$) 的类型为在上下文 W_1 中填入上下文 W_2 得到的新上下文 $W_1(W_2)$ 。

类型规则的第三部分给出了进程类型的推导方法。静止进程($\mathbf{0}$)的类型为简单类型的最小值 $\underline{\vee}^0$ 。顺序动作($M \,.\, P$)的类型通过用后续进程类型填补能力类型的空缺得到。运算符 $\mid_t$ 用于并置和复制构造的类型推导。对约束的类型推导采用了标准的方法。合法的灰箱构造必须满足灰箱名的类型和内部进程的类型一致。而对于外部来说，灰箱的存在与否对外部执行动作的移动

性和线程数均没有影响，因此，一个合法灰箱的类型为 $\underline{\vee}^0$ 。最后，进程类型可以通过子类型规则进行放大。

通过分析上述类型判定规则不难得出以下结论，即通过类型判定规则得到的进程类型都是合法的。

引理 3.3 (Validity) *如果 $\Gamma \vdash P : T$ ，那么 $T \neq \bot$ 。*

证明 通过分析 $\Gamma \vdash P : T$ 的推导过程，特别是规则 **(ET Act)** 、 **(ET Par)** 和 **(ET Repl)** 的前提条件直接可得。 ∎

在 ETS-MT中，分析规则 **(ET Par)** 和 **(ET Repl)** 可以发现，如果存在两个或以上的 $\overline{\texttt{open}}$ 动作并发运行，则进程是非法的。例如： $!\overline{\texttt{open}}$ 、 $\overline{\texttt{open}}.P \mid \overline{\texttt{open}}.Q$ 均为非法进程。

以下通过一个例子说明 ETS-MT类型规则的使用。

令 $\Gamma = n : U_n[T_n],\ m : U_m$ ，则有 $\Gamma \vdash \texttt{in}\, m\,.\,\overline{\texttt{in}}\, n\,.\,\texttt{open}\, n : \curvearrowright^1 \cdot_t \underline{\vee}^1 \cdot_t \underline{\vee}^1 \cdot_t (T_n \mid_t -)$ 。然而，由于 m 的类型为一简单类型，不能被打开，因此 $\texttt{open}\ m$ 在 Γ 下是非法的。同样，由于 n 具有演化类型，因此 $n[\mathbf{0}]$ 也是非法的，因为无法通过子类型关系将 **0** 的类型 $\underline{\vee}^0$ 放大为 n 的类型 $U_n[T_n]$ ，也就是说，由于 n 具有演化类型，灰箱 n 中的进程必须能够执行一个 $\overline{\texttt{open}}$ 动作，而这一点进程 **0** 无法做到。

§3.5 类型判定规则的归约一致性证明

本节主要证明上述 ETS-MT类型判定规则的归约一致性(subject reduction)，即：如果 $\Gamma \vdash P : T$ 且 $P \longrightarrow Q$, 那么 $\Gamma \vdash Q : T'$ 且 $T' \leq T$ （定理 3.9 ）。首先在上述性质 3.1 、 3.2 和一些引理的基础上，证明结构同余进程的类型相同（引理 3.8 ），之后再证明类型判定规则的归约一致性（定理 3.9 ）。

令 $\Gamma \vdash J$ 表示任意以下三种判定之一： $\Gamma \vdash \diamondsuit$ 、 $\Gamma \vdash M : W$ 、或者 $\Gamma \vdash P : T$ 。用 $fn(J)$ 表示 J 中（可能为 M ，也可能为 P ）出现的自由名。

引理 3.4 *如果 $\Gamma,\ n : T \vdash J$ ，那么 $n \notin dom(\Gamma)$ 。*

证明 由于只有规则 **(Env Intro)** 会在环境中增加元素，而由该规则的前提条件保证了所引入的名字是不重复的，因此在所有可由类型规则推导出的结论中，环境中的名字不会重复出现。 ∎

引理 3.5 (Implied judgement) *如果 $\Gamma_1,\ \Gamma_2 \vdash J$ ，那么 $\Gamma_1 \vdash \diamondsuit$ 。*

证明 对 Γ_1 中元素的个数进行归纳。

- 如果 $\Gamma_1 = \phi$ ，那么直接由 **(Env Empty)** 可得 $\phi \vdash \diamondsuit$ ；
- 如果 $\Gamma_1 = \Gamma_1',\ n : T$ ，即有 $\Gamma_1',\ n : T,\ \Gamma_2 \vdash J$ 。由归纳假设知 $\Gamma_1' \vdash \diamondsuit$ 。由引理 3.4 知 $n \notin dom(\Gamma_1')$ ，利用 **(Env Intro)** 可得 $\Gamma_1',\ n : T \vdash \diamondsuit$ ，即 $\Gamma_1 \vdash \diamondsuit$ 。

∎

引理 3.6 (Strengthening) *如果 $\Gamma,\ n : T \vdash J$ 且 $n \notin fn(J)$ ，那么 $\Gamma \vdash J$ 。*

证明 由于 $n \notin fn(J)$ ，可通过对 J 中的所有约束名换名，使得 J 中不出现 n 。这样在 $\Gamma,\ n:T \vdash J$ 的推导过程中，条件 $n:T$ 不会被使用，因此可采用同样的方法得到 $\Gamma \vdash J$ 。 ∎

引理 3.7 (Weakening) 如果 $\Gamma \vdash J$ 且 $n \notin dom(\Gamma)$ ，那么 $E,\ n:T \vdash J$ 。

证明 首先由 $\Gamma \vdash J$ 利用引理 3.5 知 $\Gamma \vdash \diamondsuit$ ，再利用 **(Env Intro)** 可得 $\Gamma,\ n:T \vdash \diamondsuit$ 。在此基础上，使用与 $\Gamma \vdash J$ 相同的规则可以得到 $\Gamma,\ n:T \vdash J$ 。 ∎

引理 3.8 (Subject congruence) 如果 $\Gamma \vdash P:T$ 且 $P \equiv Q$ ，则 $\Gamma \vdash Q:T$ 。

对引理 3.8 的证明过程参见附录 1 。

定理 3.9 (Subject reduction) 如果 $\Gamma \vdash P:T$ 且 $P \longrightarrow Q$，那么 $\Gamma \vdash Q:T'$ 且 $T' \leq T$ 。

对定理 3.9 的证明过程参见附录 1 。

§3.6 最小类型算法

类型判定规则给出了所有合法的类型判定。但是由于子类型关系的存在，有时类型判定规则不能精确给出一个进程具有的特性。本节给出一个在给定合法类型环境下获取能力、灰箱和进程最小类型的算法，并证明该算法的正确性和完备性。

对任意给定的合法类型环境 Γ （ $\Gamma \vdash \diamondsuit$ ），计算能力 M 、灰箱名 n 和进程 P 的类型算法分别用 $Type(\Gamma,\ M)$ 、 $Type(\Gamma,\ n)$ 和 $Type(\Gamma,\ P)$ 表示，递归定义如下：

(Type Name) $$\frac{}{Type((\Gamma',\ n:T,\ \Gamma''),\ n)\ =\ T}$$

(Type Empty) $$\frac{}{Type(\Gamma,\ \epsilon)\ =\ -}$$

(Type In) $$\frac{Type(\Gamma,\ n)\ =\ T}{Type(\Gamma,\ \mathtt{in}\,n)\ =\ \curvearrowright^1 \cdot_t -}$$

(Type Co-in) $$\frac{Type(\Gamma,\ n)\ =\ T}{Type(\Gamma,\ \overline{\mathtt{in}}\,n)\ =\ \underline{\vee}^1 \cdot_t -}$$

(Type Out) $$\frac{Type(\Gamma,\ n)\ =\ T}{Type(\Gamma,\ \mathtt{out}\,n)\ =\ \curvearrowright^1 \cdot_t -}$$

(Type Co-out) $$\frac{Type(\Gamma,\ n)\ =\ T}{Type(\Gamma,\ \overline{\mathtt{out}}\,n)\ =\ \underline{\vee}^1 \cdot_t -}$$

(Type Open) $$\frac{Type(\Gamma,\ n)\ =\ U[T]}{Type(\Gamma,\ \mathtt{open}\,n)\ =\ \underline{\vee}^1 \cdot_t (T\ |_t\ -)}$$

(Type Co-open) $$\frac{}{Type(\Gamma,\ \overline{\mathtt{open}})\ =\ \underline{\vee}^1[-]}$$

(Type Path) $$\frac{Type(\Gamma,\ M)\ =\ W \quad Type(\Gamma,\ M')\ =\ W'}{Type(\Gamma,\ M\,.\,M')\ =\ W(W')}$$

(Type Inact) $$\frac{}{Type(\Gamma,\ \mathbf{0})\ =\ \underline{\vee}^0}$$

(Type Act) $$\frac{Type(\Gamma,\ M)\ =\ W \quad Type(\Gamma,\ P)\ =\ T \quad W(T) \neq \bot}{Type(\Gamma,\ M\,.P)\ =\ W(T)}$$

(Type Par) $$\frac{Type(\Gamma,\ P)\ =\ T \quad Type(\Gamma,\ P')\ =\ T' \quad T\mid_t T' \neq \bot}{Type(\Gamma,\ P \mid P')\ =\ T\mid_t T'}$$

(Type Repl) $$\frac{Type(\Gamma,\ P)\ =\ T \quad T\mid_t T \neq \bot}{Type(\Gamma,\ !P)\ =\ T\mid_t T}$$

(Type Res) $$\frac{Type((\Gamma,\ n : T_n),\ P)\ =\ T}{Type(\Gamma,\ (\nu n : T_n)P)\ =\ T}$$

(Type Amb) $$\frac{Type(\Gamma,\ n)\ =\ T_n \quad Type(\Gamma,\ P)\ =\ T \quad T \leq T_n}{Type(\Gamma,\ n[P])\ =\ \underline{\vee}^0}$$

对于类型算法可以计算出结果的能力、灰箱名和进程，我们称类型算法在其（ M 、 n 或 P ）上有解；如果无法通过上述算法得出结果，则表示其类型是非法的。

上述类型算法的定义基本上类似类型判定规则的内容，不同之处在于 **(Type Amb)** 中的 $T \leq T_n$ 条件代替了 **(ET Sub)** 规则。类型算法的正确性（ soundness ）和完备性（ completeness ）由下面的两个定理保证。

定理 3.10 （正确性）

(1). 如果 $Type(\Gamma,\ n)\ =\ T$ ，那么 $\Gamma \vdash n : T$ ；

(2). 如果 $Type(\Gamma,\ M)\ =\ W$ ，那么 $\Gamma \vdash M : W$ ；

(3). 如果 $Type(\Gamma,\ P)\ =\ T$ ，那么 $\Gamma \vdash P : T$ 。

证明 对类型算法中的规则进行归纳易证（大部分情况都容易利用归纳假设直接得到结论，其中分析 **(Type Amb)** 时需要多使用一次 **(ET Sub)** ）。 ∎

定理 3.11 （完备性）

(1). 如果 $\Gamma \vdash n : T$ ，那么类型算法在 n 上有解，且 $Type(\Gamma,\ n)\ =\ T$ ；

(2). 如果 $\Gamma \vdash M : W$ ，那么类型算法在 M 上有解，且 $Type(\Gamma,\ M)\ =\ W$ ；

(3). 如果 $\Gamma \vdash P : T$ ，那么类型算法在 P 上有解，且 $Type(\Gamma,\ P)\ \leq\ T$ 。

证明 对类型判定规则进行归纳易证（其中对归纳假设的利用要用到性质 3.1 和性质 3.2 ，特别是其中的 **(Dot Strict T)** 和 **(Par Strict T)** ）。 ∎

由上述两个定理可以推论如下：在 ETS-MT中，所有具有合法类型的灰箱、能力和进程都具有一个最小类型，且该最小类型正好为类型算法的结果。

最后我们应用上述定理给出类型算法下进程最小类型的归约一致性。

定理 3.12 如果 $Type(\Gamma,\ P)\ =\ T$ 且 $P \longrightarrow Q$ ，则类型算法在 Q 上有解，且 $Type(\Gamma,\ Q)\ \leq\ T$ 。

证明 由 $Type(\Gamma,\ P)\ =\ T$ 利用定理 3.10 可知 $\Gamma \vdash P : T$ ，在由 $P \longrightarrow Q$ 利用定理 3.9 可得 $\Gamma \vdash Q : T'$ 且 $T' \leq T$ ，再利用定理 3.11 有 $Type(\Gamma,\ Q) \leq T'$ ，因此 $Type(\Gamma,\ Q) \leq T$ 。 ∎

§3.7 举例

本节给出一个非移动服务器的例子，来说明 ETS-MT在刻划类型演化方面的能力。

在该例子中，一个网络服务器（ **Server** ）不停收集代理（ **Agent** ）携带的数据。服务器和代理各自的定义如下：

$$\mathbf{Server} \triangleq s[\,!\overline{\mathtt{in}}\ a\,.\,\mathtt{open}\ a\,.\,\mathbf{Collect}]$$
$$\mathbf{Agent} \triangleq a[\,\mathtt{in}\ s\,.\,\overline{\mathtt{open}}\,.\,\mathbf{Data}]$$

这里假定进程 **Collect** 和 **Data** 都具有非移动的简单类型。

由于存在多个代理异步、自主运行，负责为代理提供服务的服务器应当全天候提供服务。为了做到这一点，服务器必备的一个特性就是非移动性。然而在不支持类型演化的类型系统中，由于服务器要打开具有移动性的代理以取得数据，服务器的类型必须也为移动的。

在 ETS-MT中，服务器的非移动性就可通过类型的演化得到解决。假定 Γ 是一个包含元素 $s : T_s$ 和 $a : T_a$ 的合法环境，且满足条件 $Type(\Gamma,\ \mathbf{Data}) = \underline{\vee}^{Y_1}$ 与 $Type(\Gamma,\ \mathbf{Collect}) = \underline{\vee}^{Y_2}$，我们通过类型算法计算 T_s 和 T_a 可能的取值如下。

(1)	$Type(\Gamma,\ s) = T_s$	已知条件
(2)	$Type(\Gamma,\ \mathtt{in}\ s) = \curvearrowright^1 \cdot_t -$	(1)+**(Type In)**
(3)	$Type(\Gamma,\ \overline{\mathtt{open}}) = \underline{\vee}^1[-]$	**(Type Co-open)**
(4)	$Type(\Gamma,\ \mathtt{in}\ s\,.\,\overline{\mathtt{open}}) = \curvearrowright^1 \cdot_t(\underline{\vee}^1[-])$	(2)(3)+**(Type Path)**
(5)	$Type(\Gamma,\ \mathbf{Data}) = \underline{\vee}^{Y_1}$	已知条件
(6)	$Type(\Gamma,\ \mathtt{in}\ s\,.\,\overline{\mathtt{open}}\,.\,\mathbf{Data}) = \curvearrowright^1 [\underline{\vee}^{Y_1}]$	(4)(5)+**(Type Act)**
(7)	$\curvearrowright^1 [\underline{\vee}^{Y_1}] \leq T_a$	(6)+**(Type Amb)**
(8)	$Type(\Gamma,\ \overline{\mathtt{in}}\ a) = \underline{\vee}^1 \cdot_t -$	(7)+**(Type Co-in)**
(9)	$Type(\Gamma,\ \mathtt{open}\ a) = \underline{\vee}^1 \cdot_t (\underline{\vee}^{Y_1} \mid_t -)$	(7)+**(Type Open)**, 取 T_a 为最小值 $\curvearrowright^1 [\underline{\vee}^{Y_1}]$
(10)	$Type(\Gamma,\ \mathbf{Collect}) = \underline{\vee}^{Y_2}$	已知条件
(11)	$Type(\Gamma,\ \mathtt{open}\ a\,.\,\mathbf{Collect}) = \underline{\vee}^{1 \cdot y(Y_1 \mid_y Y_2)}$	(9)(10)+**(Type Act)**
(12)	$Type(\Gamma,\ \overline{\mathtt{in}}\ a\,.\,\mathtt{open}\ a\,.\,\mathbf{Collect}) = \underline{\vee}^{1 \cdot y(Y_1 \mid_y Y_2)}$	(8)(11)+**(Type Act)**
(13)	$Type(\Gamma,\ !\overline{\mathtt{in}}\ a\,.\,\mathtt{open}\ a\,.\,\mathbf{Collect}) = \underline{\vee}^{\omega}$	(12)+**(Type Repl)**
(14)	$\underline{\vee}^{\omega} \leq T_s$	(13)+**(Type Amb)**

由式 (14) 可知服务器灰箱 s 的类型可以为非移动。

在下面研究进程的等价性定律时，我们还可以看到非移动性和单线程性在进程等价性研究中的重要意义。

本章小结

本章结合灰箱演算中存在的类型演化问题，提出了解决该问题的一套演化类型系统 ETS-MT 。解决了对 ROAM 进程在打开操作前后类型变化的描述问题。 ETS-MT具有简洁明了的语法和类型判定规则，可以刻划进程的移动性和线程数两个属性，并支持子类型。除了类型判定规则以外，本章还给出了一个计算进程和灰箱最小类型的算法，并用一个例子说明了 ETS-MT在描述复杂移动系统时的优越性。

本章的结论虽然是建立在 ROAM 演算的基础上的，但是可以很容易地移植到其它灰箱演算系统中，如 SA 演算 [5] 。本章的工作为下文研究 ROAM 进程的代数性质提供了重要的工具。

[5]但是由于 MA 中没有跟踪类型演化的必要动作 $\overline{\texttt{open}}$ ，移植到 MA 上还要做进一步的研究。最近 Amtoft 在文献 [33] 中通过结合类型系统中组的概念 [30] ，研究了 MA 中没有 $\overline{\texttt{open}}$ 情况下对灰箱的移动和打开的跟踪，具有很好的参考价值。

第四章　标号转移语义

本章开始研究 ROAM 中进程的等价关系。类似其它一些进程代数系统，对 ROAM 进程中等价关系的研究首先需要脱离基于结构同余关系的归约语义，去寻求一套标号转移语义（ labelled transition semantics ）来表达单个进程与外部发生交互的可能性，同时此标号转移语义必须具有与原有归约语义的等价性。

现有灰箱演算标号转移语义的研究工作主要有 Cardelli 和 Gordon 基于 MA 的工作 [19, 20] 以及 Levi 和 Sangiorgi 基于 SA 的工作 [16] 等。其中文献 [19, 16] 使用基于 **提交**（ commitment ）和 **结果**（ outcome ）的方法，语义比较直观，但是标号众多，且标号转移结果除了进程还可能是 **固化结果**（ concretion ）；文献 [20] 使用基于 **硬化关系**（ hardening relation ）的方法，引入了一个新的关系，但标号转移结果仍为进程，且非内部转移标号的个数和语法中的动作个数相同。这两种方法各有特色，本章中标号转移语义的构造采用后一种方法。本章内容和下一章内容基本上是文献 [20] 中关于 MA 的等价性结论在 ROAM 中的应用，熟悉文献 [20] 的读者可以对照阅读。当然，由于协动作的存在， ROAM 中的结论将会有所不同，主要在于标号转移语义的定义以及第五章中进程和简单上下文（ harness ）的交互种类上。

由于进程等价关系研究中的大部分内容不涉及到类型系统，为了书写的简便，这里进程中的约束构造（包括将要定义的硬化关系中的约束构造）将大量采用无类型的方式，即省略表示约束名类型的符号“ T ”。

§4.1　硬化关系

由于灰箱演算的特殊性，进程中具有归约能力的部分可能包含在某个灰箱的内部，归约的结果也可能是接收一个外来灰箱进入其内部。这使得灰箱演算的标号转移语义很难用类似 CCS 和 π 演算的方式直接描述。为了突出进程内部可参与归约的顶层子进程，这里借鉴对 MA 等价关系的研究 [20] 引入针对 ROAM 的 **硬化关系** “ $>$ ”。硬化关系具有一般形式[1]： $P > (\nu\vec{p})\langle P'\rangle P''$ ，称 $(\nu\vec{p})\langle P'\rangle P''$ 为 **硬化结果**。其中， P' 为 P 中可以与外界发生交互的某个部分，称为 **活体**（ prime ）， P'' 为 P 中除去活体后所剩余的部分，称为 **残余**（ residue ）。约束名向量 $\vec{p}$ 存放了 P' 和 P'' 共享的约束名。硬化关系选出进程 P 中多个顶层（不被灰箱边界包围的）子进程中的某一个，形成只有一个活体的硬化结果。各种类型的活体有各自特殊的交互特性，这样进程的交互特性就可由该进程能硬化出的各种不同的活体而体现出来。

通过分析 ROAM 的语法，引入硬化结果 $C \in \mathcal{C}$ 的一般形式如下：

$$\begin{array}{lll} C & ::= & (\nu\vec{p})\langle A.P\rangle Q \qquad \textbf{动作活体} \\ & & (\nu\vec{p})\langle n[P]\rangle Q \qquad \textbf{灰箱活体} \end{array}$$

这里约束名向量 $\vec{p}$ 采用无类型的方式，且其元素顺序是无关的。为书写方便，下文直接用 $\{\vec{p}\}$ 表示约束名集合 $\{p_1, p_2, \ldots, p_k\}$ 。集合 $\{\vec{p}\}$ 为空时，硬化结果写作： $(\nu)\langle P'\rangle P''$ 。定义对硬化结果的取自由名函数 $fn((\nu\vec{p})\langle P'\rangle P'') \triangleq (fn(P') \cup fn(P')) - \{\vec{p}\}$ 。

[1] 进程中的约束构造省略类型部分后，硬化关系中的 $\nu\vec{p}$ 中的类型符号也相应省略，当需要类型符号时，应当写为： $\nu\vec{p}:\vec{T}_p$ 。

围绕引入硬化关系的目的，首先对硬化关系作一直观分析。若进程 P 具有动作（ $A.Q$ ）或灰箱（ $n[Q]$ ）的形式，则可直接硬化为 $(\nu)\langle P\rangle \mathbf{0}$ ；若进程 P 、Q 中的某一个可硬化为某硬化结果，则其并置 $P \mid Q$ 也可硬化为类似的结果，唯一不同是该结果的残余为原结果的残余并置 P 、Q 中的未硬化者； $!P$ 可理解为 $!P \equiv P \mid !P$ ，故将 $!P$ 并置于 P 硬化结果的残余中即得到 $!P$ 的硬化结果；若 P 可硬化为 C ，则 $(\nu n)P$ 的硬化结果可按照 n 的约束范围不同，将 (νn) 分别放在 C 的约束名向量、活体或残余部分得到，该操作通过新定义的 $\overline{(\nu n)}C$ 记号来完成。

通过对硬化关系的直观理解，给出其形式化的定义如下：

定义 4.1 （硬化关系） 硬化关系 $> \subseteq \mathcal{P} \times \mathcal{C}$ 定义如下：

(Harden Action) $$\frac{}{A\,.\,P > (\nu)\,\langle A\,.\,P\rangle\,\mathbf{0}}$$

(Harden Amb) $$\frac{}{n[P] > (\nu)\,\langle n[P]\rangle\,\mathbf{0}}$$

(Harden Par 1) $$\frac{P > (\nu\vec{p})\,\langle P'\rangle\,P'' \quad \{\vec{p}\} \cap fn(Q) = \emptyset}{P \mid Q > (\nu\vec{p})\,\langle P'\rangle\,(P'' \mid Q)}$$

(Harden Par 2) $$\frac{Q > (\nu\vec{q})\,\langle Q'\rangle\,Q'' \quad \{\vec{q}\} \cap fn(P) = \emptyset}{P \mid Q > (\nu\vec{q})\,\langle Q'\rangle\,(P \mid Q'')}$$

(Harden Repl) $$\frac{P > (\nu\vec{p})\,\langle P'\rangle\,P''}{!P > (\nu\vec{p})\,\langle P'\rangle\,(P'' \mid !P)}$$

(Harden Res) $$\frac{P > C}{(\nu n)P > \overline{(\nu n)}C}$$

(Harden Empty) $$\frac{P > C}{\epsilon\,.\,P > C}$$

(Harden Path) $$\frac{M\,.\,(N\,.\,P) > C}{(M\,.\,N)\,.\,P > C}$$

其中，操作 $\overline{(\nu n)}C$ 定义为（设 $C = (\nu\vec{p})\,\langle P'\rangle\,P''$ ，且 $n \notin \{\vec{p}\}$ ）：

(1). 如果 $n \notin fn(P')$ ，则 $\overline{(\nu n)}C \triangleq (\nu\vec{p})\,\langle P'\rangle\,(\nu n)P''$ ；

(2). 如果 $n \in fn(P')$ ，则：

(a) 如果 $P' = m[P'_1]$ 、$m \neq n$ 、$n \notin fn(P'')$ ，则 $\overline{(\nu n)}C \triangleq (\nu\vec{p})\,\langle m[(\nu n)P'_1]\rangle\,P''$ ；

(b) 否则， $\overline{(\nu n)}C \triangleq (\nu n, \vec{p})\,\langle P'\rangle\,P''$ 。

分析上述定义不难看出，如果 $P > (\nu\vec{p})\,\langle P'\rangle\,P''$ ，则 $P \equiv (\nu\vec{p})(P' \mid P'')$ 。下文将给出这一结论的证明（引理 4.4 ）。

§4.2 基于硬化关系的标号转移语义

本节在上文硬化关系的基础上，建立一套 ROAM 的标号转移语义，并在下节中证明其与前文所述 ROAM 归约语义的等价性。

定义 4.2 （标号转移语义） *ROAM* 的标号转移语义 **LTS** $\subseteq \mathcal{P} \times \mathcal{P} \times$ **Label** 为一三元关系，其中的标号集合 **Label** （元素用 α 表示） $\triangleq \{\tau\} \cup$ **Action** 。 τ 表示一个内部转移动作。

为方便书写，将 $(P,\, Q,\, \alpha) \in$ **LTS** 简写为 $P \xrightarrow{\alpha} Q$ 。 **LTS** 的定义如下：

(Trans Cap) $$\frac{P > (\nu\vec{p})\langle A.P'\rangle P'' \quad fn(A)\cap\{\vec{p}\}=\emptyset}{P \xrightarrow{A} (\nu\vec{p})(P' \mid P'')}$$

(Trans In) $$\frac{P > (\nu\vec{p})\langle n[Q]\rangle R \quad Q \xrightarrow{\overline{\mathtt{in}}\ m} Q' \quad R > (\nu\vec{r})\langle m[S]\rangle R' \quad S \xrightarrow{in\ n} S' \quad \{\vec{r}\}\cap fn(n[Q]) = \emptyset \quad \{\vec{r}\}\cap\{\vec{p}\}=\emptyset}{P \xrightarrow{\tau} (\nu\vec{p},\vec{r})(n[m[S'] \mid Q'] \mid R')}$$

(Trans Out) $$\frac{P > (\nu\vec{p})\langle n[Q]\rangle P' \quad Q > (\nu\vec{q})\langle m[R]\rangle S \quad R \xrightarrow{\mathtt{out}\ n} R' \quad S \xrightarrow{\overline{\mathtt{out}}\ m} S' \quad n\notin\{\vec{q}\}}{P \xrightarrow{\tau} (\nu\vec{p})((\nu\vec{q})(m[R'] \mid n[S']) \mid P')}$$

(Trans Open) $$\frac{P > (\nu\vec{p})\langle n[Q]\rangle R \quad Q \xrightarrow{\overline{\mathtt{open}}} Q' \quad R \xrightarrow{open\ n} R'}{P \xrightarrow{\tau} (\nu\vec{p})(Q' \mid R')}$$

(Trans Amb) $$\frac{P > (\nu\vec{p})\langle n[Q]\rangle P' \quad Q \xrightarrow{\tau} Q'}{P \xrightarrow{\tau} (\nu\vec{p})(n[Q'] \mid P')}$$

A-转移 $P \xrightarrow{A} Q$ 表示进程 P 存在一顶层子进程可执行动作 A 。 τ-转移 $P \xrightarrow{\tau} Q$ 表示进程 P 经过一个内部动作后可转化为 Q 。下文用 $fn(\alpha)$ 表示 α 中出现的（自由）名集合。

规则 **(Trans In)** 、 **(Trans Out)** 和 **(Trans Open)** 用于和三条归约规则 **(Red In)** 、 **(Red Out)** 、 **(Red Open)** 相对应， A-转移的引入简化了这些规则中的条件部分。 **(Trans Amb)** 填补了允许 τ-转移在灰箱内部发生这一点。注意，这里无需再提供与 **(Red Par)** 、 **(Red Res)** 和 **(Red Struct)** 对应的标号转移规则，因为在硬化关系的定义中，这些规则已经体现其中（参见下文引理 4.7 、 4.9 和 4.15 的证明）。

§4.3 归约语义和标号转移语义的等价性

本节主要完成标号转移语义和最初引入的归约语义之间的等价性证明，指出它们从不同角度描述了相同的进程交互特性。同时也使得下文以标号转移语义为基础进行的进程等价性研究结论对于归约语义同样成立。

本节的一系列证明可分为两个部分。第一部分利用引理 4.4 、 4.5 证明归约语义包含标号转移语义（命题 4.6 ）。第二部分首先证明了引理 4.7 、 4.9 、 4.15 ，即标号转移语义同样提供类似 **(Red Par)** 、 **(Red Res)** 和 **(Red Struct)** 所允许的进程内部归约的发生。据此证明了标号转移语义包含归约语义（命题 4.16 ）。最后，本节综合命题 4.6 、 4.16 得出两套语义相互等价的结论（定理 4.17 ）。

引理 4.3 如果 $P \equiv Q$ ，那么 $fn(P) = fn(Q)$ 。

证明 对“ $\equiv$ ”的定义进行归纳易证，此处略。 ∎

引理 4.4 如果 $P > (\nu\vec{p})\langle P'\rangle P''$ ，则 $P \equiv (\nu\vec{p})(P' \mid P'')$ 。

证明 对 $P > (\nu\vec{p})\langle P'\rangle P''$ 的推导过程进行归纳。

(Harden Action)(Harden Amb) 直接可知假设成立。

(Harden Par 1) 此时有 $P = P_1 \mid Q$ 、 $P_1 > (\nu\vec{p})\langle P'\rangle P_1''$ 、 $P'' = P_1'' \mid Q$ 且 $\{\vec{p}\} \cap fn(Q) = \emptyset$ 。由归纳假设知 $P_1 \equiv (\nu\vec{p})(P' \mid P_1'')$ ，因此 $P = P_1 \mid Q \equiv (\nu\vec{p})(P' \mid P_1'') \mid Q \equiv (\nu\vec{p})(P' \mid P'')$ （其中，第二次结构同余由 $\{\vec{p}\} \cap fn(Q) = \emptyset$ 保证），假设成立。

(Harden Par 2) 仿 **(Harden Par 1)** 可知假设成立。

(Harden Repl) 此时有 $P = !P_1$ 、$P_1 > (\nu\vec{p})\langle P'\rangle P_1''$ 且 $P'' = P_1'' \mid !P_1$ 。由归纳假设知 $P_1 \equiv (\nu\vec{p})(P' \mid P_1'')$ ，根据引理 4.3 知 $fn(P_1) = fn((\nu\vec{p})(P' \mid P_1''))$ ，因此 $\{\vec{p}\} \cap fn(P_1) = \emptyset$ ，由此利用 **(Struct Res Par)** 可知 $P = !P_1 \equiv P_1 \mid !P_1 \equiv (\nu\vec{p})(P' \mid P_1'') \mid !P_1 \equiv (\nu\vec{p})(P' \mid (P_1'' \mid !P_1)) = (\nu\vec{p})(P' \mid P'')$ ，假设成立。

(Harden Res) 此时有 $P = (\nu n)P_1$ 、 $P_1 > (\nu\vec{p}_1)\langle P_1'\rangle P_1''$ 、 $n \notin \{\vec{p}_1\}$ 且 $(\nu\vec{p})\langle P'\rangle P'' = \overline{(\nu n)}((\nu\vec{p}_1)\langle P_1'\rangle P_1'')$ 。由归纳假设知 $P_1 \equiv (\nu\vec{p}_1)(P_1' \mid P_1'')$ ，因此 $P = (\nu n)P_1 \equiv (\nu n)(\nu\vec{p}_1)(P_1' \mid P_1'') \equiv (\nu\vec{p}_1)(\nu n)(P_1' \mid P_1'')$ 。下面对 n 和 P_1' 及 P_1'' 的关系分情况讨论：

(1). 若 $n \notin fn(P_1')$ ，则 $\overline{(\nu n)}((\nu\vec{p}_1)\langle P_1'\rangle P_1'') = (\nu\vec{p}_1)\langle P_1'\rangle (\nu n)P_1''$ （即： $\vec{p} = \vec{p}_1$ 、 $P' = P_1'$ 且 $P'' = (\nu n)P_1''$ ），而 $P \equiv (\nu\vec{p}_1)(\nu n)(P_1' \mid P_1'') \equiv (\nu\vec{p}_1)(P_1' \mid (\nu n)P_1'') = (\nu\vec{p})(P' \mid P'')$ ，假设成立。

(2). 若 $n \in fn(P')$ ，则：

(a) 若 $P_1' = m[P_2']$ 、$m \neq n$ 、$n \notin fn(P_1'')$ ，则 $\overline{(\nu n)}((\nu\vec{p}_1)\langle P_1'\rangle P_1'') = (\nu\vec{p}_1)\langle m[(\nu n)P_2']\rangle P_1''$ （即： $\vec{p} = \vec{p}_1$ 、 $P' = m[(\nu n)P_2']$ 且 $P'' = P_1''$ ），而 $P \equiv (\nu\vec{p}_1)(\nu n)(P_1' \mid P_1'') \equiv (\nu\vec{p}_1)(((\nu n)P_1') \mid P_1'') = (\nu\vec{p}_1)(((\nu n)m[P_2']) \mid P_1'') \equiv (\nu\vec{p})(m[(\nu n)P_2'] \mid P_1'') = (\nu\vec{p})(P' \mid P'')$ ，假设成立。

(b) 否则， $\overline{(\nu n)}((\nu\vec{p}_1)\langle P_1'\rangle P_1'') = (\nu n, \vec{p}_1)\langle P_1'\rangle P_1''$ （即： $\vec{p} = \{n\} \cup \vec{p}_1$ 、 $P' = P_1'$ 且 $P'' = P_1''$ ），假设直接成立。

(Harden Empty) 由 $\epsilon.P \equiv P$ 易知。

(Harden Path) 由 $M.(N.P) \equiv (M.N).P$ 易知。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 4.5 如果 $P \xrightarrow{A} Q$ ，则存在 P' 、P'' 和 $\vec{p}$ 满足 $P \equiv (\nu\vec{p})(A.P' \mid P'')$ 、$Q = (\nu\vec{p})(P' \mid P'')$ 且 $fn(A) \cap \{\vec{p}\} = \emptyset$ 。

证明 由于 $P \xrightarrow{A} Q$ 只能通过规则 **(Trans Cap)** 得到，因此有 $P > (\nu\vec{p})\langle A.P'\rangle P''$ ，$Q = (\nu\vec{p})(P' \mid P'')$ 且 $fn(A) \cap \{\vec{p}\} = \emptyset$ 。由引理 4.4 知， $P \equiv (\nu\vec{p})(A.P' \mid P'')$ 。 ∎

在引理 4.4 、 4.5 的基础上，标号转移语义的定义中各条件部分就可通过转换成相应的结构同余关系，进一步使进程变换为归约语义所要求的结构。这样，由任何标号转移语义所得的 τ-转移关系均可得到相应的归约关系。

命题 4.6 如果 $P \xrightarrow{\tau} P'$ ，则 $P \longrightarrow P'$ 。

证明 对 $P \xrightarrow{\tau} P'$ 的推导过程进行归纳。

(Trans In) 此时有 $P > (\nu\vec{p})\langle n[Q]\rangle R$ 、$Q \xrightarrow{\overline{\texttt{in}}\ m} Q'$ 、$R > (\nu\vec{r})\langle m[S]\rangle R'$ 、$S \xrightarrow{in\ n} S'$ 、$\{\vec{r}\} \cap fn(n[Q]) = \emptyset$ 、 $\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 且 $P' = (\nu\vec{p}, \vec{r})(n[m[S'] \mid Q'] \mid R')$ 。由引理 4.4 可知 $P \equiv (\nu\vec{p})(n[Q] \mid R)$ ，$R \equiv (\nu\vec{r})(m[S] \mid R')$ ；再利用引理 4.5 可得 $Q \equiv (\nu\vec{q})(\overline{\texttt{in}}\ m.Q_1 \mid Q_2)$ 、 $Q' \equiv (\nu\vec{q})(Q_1 \mid Q_2)$ 、

$m \notin \{\vec{q}\}$ ， $S \equiv (\nu\vec{s})(in\ n.S_1 \mid S_2)$ 、 $S' \equiv (\nu\vec{s})(S_1 \mid S_2)$ 、 $n \notin \{\vec{s}\}$ 。由于 $\vec{q}$ 、 $\vec{s}$ 均为约束名，不妨设 $\{\vec{q}\} \cap fn(m[S]) = \emptyset$ 、 $\{\vec{s}\} \cap fn(n[Q]) = \emptyset$ 。综合以上条件有（其中 **(Struct Res Par)** 的使用由条件 $\{\vec{r}\} \cap fn(n[Q]) = \emptyset$ 、 $\{\vec{q}\} \cap fn(m[S]) = \emptyset$ 、 $\{\vec{s}\} \cap fn(n[Q]) = \emptyset$ 保证）：

$$\begin{aligned}
P &\equiv (\nu\vec{p})(n[(\nu\vec{q})(\overline{\texttt{in}}\ m.Q_1 \mid Q_2)] \mid (\nu\vec{r})(m[(\nu\vec{s})(in\ n.S_1 \mid S_2)] \mid R')) \\
&\equiv (\nu\vec{p},\vec{r})((\nu\vec{q},\vec{s})(n[\overline{\texttt{in}}\ m.Q_1 \mid Q_2] \mid m[in\ n.S_1 \mid S_2]) \mid R') \\
&\longrightarrow (\nu\vec{p},\vec{r})((\nu\vec{q},\vec{s})(n[Q_1 \mid Q_2 \mid m[S_1 \mid S_2]]) \mid R') \\
&\equiv (\nu\vec{p},\vec{r})(n[(\nu\vec{q})(Q_1 \mid Q_2) \mid m[(\nu\vec{s})(S_1 \mid S_2)]] \mid R') \\
&\equiv (\nu\vec{p},\vec{r})(n[m[S'] \mid Q'] \mid R') \\
&= P'
\end{aligned}$$

由此可知，假设成立。

(Trans Out)(Trans Open) 采用类似 **(Trans In)** 的方法可知假设成立，此处略。

(Trans Amb) 此时有 $P > (\nu\vec{p})\langle n[Q]\rangle R$ 、 $Q \xrightarrow{\tau} Q'$ 且 $P' \equiv (\nu\vec{p})(n[Q'] \mid R)$ 。由归纳假设知 $Q \longrightarrow Q'$ ，又由引理 4.4 知 $P \equiv (\nu\vec{p})(n[Q] \mid R)$ 。因此， $P \equiv (\nu\vec{p})(n[Q] \mid R) \longrightarrow (\nu\vec{p})(n[Q'] \mid R) \equiv P'$ ，假设成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

以上证明了等价性的一半，下面在证明进程的标号转移能力被并置（引理 4.7 ）、约束（引理 4.9 ）及结构同余（引理 4.15 ）保持的基础上，证明等价性的另一半（命题 4.16 ）。

引理 4.7 如果 $P \xrightarrow{\alpha} Q$ ，则 $P \mid R \xrightarrow{\alpha} \equiv Q \mid R$ 。

证明 对 $P \xrightarrow{\alpha} Q$ 的最后一步推导使用的规则作简单分析如下。

(Trans Cap) 此时有 $P > (\nu\vec{p})\langle A.P'\rangle P''$ 、 $Q = (\nu\vec{p})(P' \mid P'')$ 、 $fn(A) \cap \{\vec{p}\} = \emptyset$ 且 $\alpha = A$ 。由于 $\vec{p}$ 为约束名向量，因此不妨设 $\{\vec{p}\} \cap fn(R) = \emptyset$ ，利用 **(Harden Par 1)** 可得 $P \mid R > (\nu\vec{p})\langle A.P'\rangle (P'' \mid R)$ ，再由 **(Trans Cap)** 得 $P \mid R \xrightarrow{A} (\nu\vec{p})(P' \mid (P'' \mid R)) \equiv Q \mid R$ ，结论成立。

(Trans In) 此时有 $P > (\nu\vec{p})\langle n[P_1]\rangle P_2$ 、 $P_1 \xrightarrow{\overline{\texttt{in}}\ m} P_1'$ 、 $P_2 > (\nu\vec{r})\langle m[P_3]\rangle P_2'$ 、 $P_3 \xrightarrow{in\ n} P_3'$ 、 $\{\vec{r}\} \cap fn(n[P_1]) = \emptyset$ 、 $\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 、 $Q = (\nu\vec{p},\vec{r})(n[m[P_3'] \mid P_1'] \mid P_2')$ 且 $\alpha = \tau$ 。同样不妨设 $(\{\vec{p}\} \cup \{\vec{r}\}) \cap fn(R) = \emptyset$ ，利用 **(Harden Par 1)** 可得 $P \mid R > (\nu\vec{p})\langle n[P_1]\rangle (P_2 \mid R)$ ，类似可得 $P_2 \mid R > (\nu\vec{r})\langle m[P_3]\rangle (P_2' \mid R)$ ，再由 **(Trans In)** 得 $P \mid R \xrightarrow{\tau} (\nu\vec{p},\vec{r})(n[m[P_3'] \mid P_1'] \mid (P_2' \mid R)) \equiv Q \mid R$ ，结论成立。

(Trans Out)(Trans Open) 采用类似 **(Trans In)** 的方法可知结论成立，此处略。

(Trans Amb) 采用类似 **(Trans Cap)** 的方法可知结论成立，此处略。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 4.8 如果 $P \xrightarrow{\alpha} Q$ ，那么 $fn(\alpha) \cup fn(Q) \subseteq fn(P)$ 。

证明 对 $P \xrightarrow{\alpha} Q$ 的推导进行归纳并利用引理 4.4 易证，此处略。 ∎

引理 4.9 如果 $P \xrightarrow{\alpha} Q$ 且 $n \notin fn(\alpha)$ ，则 $(\nu n)P \xrightarrow{\alpha} \equiv (\nu n)Q$ 。

证明 对 $P \xrightarrow{\alpha} Q$ 的推导过程进行归纳。

(Trans Cap) 此时有 $P > (\nu\vec{p})\langle A.P'\rangle P''$ 、 $Q = (\nu\vec{p})(P' \mid P'')$ 、 $fn(A) \cap \{\vec{p}\} = \emptyset$ 且 $\alpha = A$ 。由于 $\vec{p}$ 为约束名向量，因此不妨设 $n \notin \{\vec{p}\}$ 。以下分两种情况讨论：

(1). 若 $n \notin fn(A.P')$ ，则 $(\nu n)P > \overline{(\nu n)}((\nu\vec{p})\langle A.P'\rangle P'') = (\nu\vec{p})\langle A.P'\rangle (\nu n)P''$ ，利用 **(Trans Cap)** 得 $(\nu n)P \xrightarrow{A} (\nu\vec{p})(P' \mid (\nu n)P'') \equiv (\nu n)Q$ ，假设成立；

(2). 若 $n \in fn(A.P')$ ，则 $(\nu n)P > \overline{(\nu n)}((\nu\vec{p})\langle A.P'\rangle P'') = (\nu n, \vec{p})\langle A.P'\rangle P''$ ，由条件 $n \notin fn(\alpha)$ 利用 **(Trans Cap)** 得 $(\nu n)P \xrightarrow{A} (\nu n, \vec{p})(P' \mid P'') \equiv (\nu n)Q$ ，假设成立。

(Trans In) 此时有 $P > (\nu\vec{p})\langle n'[P_1]\rangle P_2$ 、 $P_1 \xrightarrow{\overline{\text{in}}\ m} P_1'$ 、 $P_2 > (\nu\vec{r})\langle m[P_3]\rangle P_2'$ 、 $P_3 \xrightarrow{in\ n'} P_3'$ 、 $\{\vec{r}\} \cap fn(n'[P_1]) = \emptyset$ 、 $\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 、 $Q = (\nu\vec{p}, \vec{r})(n'[m[P_3'] \mid P_1'] \mid P_2')$ 且 $\alpha = \tau$ 。同样不妨设 $n \notin \{\vec{p}\} \cup \{\vec{r}\}$ ，以下分情况讨论：

(1). 若 $n \notin fn(n'[P_1])$ ，则 $(\nu n)P > (\nu\vec{p})\langle n'[P_1]\rangle (\nu n)P_2$ ，分情况讨论：

(a) 若 $n \notin fn(m[P_3])$ ，则 $(\nu n)P_2 > (\nu\vec{r})\langle m[P_3]\rangle (\nu n)P_2'$ ，由 **(Trans In)** 得 $(\nu n)P \xrightarrow{\tau} (\nu\vec{p}, \vec{r})(n'[m[P_3'] \mid P_1'] \mid (\nu n)P_2') \equiv (\nu n)Q$ （结构同余要用到引理 4.8 ），假设成立；

(b) 若 $n \in fn(m[P_3])$ 、 $n \neq m$ 且 $n \notin fn(P_2')$ ，则 $(\nu n)P_2 > (\nu\vec{r})\langle m[(\nu n)P_3]\rangle P_2'$ ，由归纳假设知， $(\nu n)P_3 \xrightarrow{in\ n'} P_3'' \equiv (\nu n)P_3'$ ，由 **(Trans In)** 得 $(\nu n)P \xrightarrow{\tau} (\nu\vec{p}, \vec{r})(n'[m[P_3''] \mid P_1'] \mid P_2') \equiv (\nu n)Q$ （结构同余要用到引理 4.8 ），假设成立；

(c) 否则 $(\nu n)P_2 > (\nu n, \vec{r})\langle m[P_3]\rangle P_2'$ ，直接利用 **(Trans In)** 可得 $(\nu n)P \xrightarrow{\tau} (\nu\vec{p}, n, \vec{r})(n'[m[P_3'] \mid P_1'] \mid P_2') \equiv (\nu n)Q$ ，假设成立；

(2). 若 $n \in fn(n'[P_1]$ 、 $n \neq n'$ 且 $n \notin fn(P_2)$ ，则 $(\nu n)P > (\nu\vec{p})\langle n'[(\nu n)P_1]\rangle P_2$ ，类似情况 (1).(b) 可知 $(\nu n)P \xrightarrow{\tau} \equiv (\nu n)Q$ ，假设成立。

(3). 否则 $(\nu n)P > (\nu n, \vec{p})\langle n'[P_1]\rangle P_2$ ，类似情况 (1).(c) 可知 $(\nu n)P \xrightarrow{\tau} \equiv (\nu n)Q$ ，假设成立。

(Trans Out)(Trans Open) 采用类似 **(Trans In)** 的方法易知假设成立，此处略。

(Trans Amb) 采用类似 **(Trans Cap)** 的方法易知假设成立，此处略。

通过以上对所有可能情况的分析可知，结论成立。 ∎

证明归约语义和标号转移语义等价性的另一半，即归约语义蕴涵标号转移语义时，需要对同余进程的归约进行分析（即规则 **(Red Struct)** ）。为了证明同余进程具有类似的硬化能力和标号转移能力（引理 4.14 和引理 4.15 ），首先引入下面关于能力集 $\mathcal{M}$ 上的硬化关系的定义和相应的一些引理。

定义 4.10 能力集 $\mathcal{M}$ 上的硬化关系 “ $>$ ” 定义如下：

(1). $\epsilon > \epsilon$ ；

(2). $M > M.\epsilon$ ，如果 $M \in \mathbf{Action}$ ；

(3). $M.N > A.(M'.N)$ ，如果 $M > A.M'$ ；

(4). $M.N > N'$ ，如果 $M > \epsilon$ 且 $N > N'$ 。

引理 4.11 如果 $M.P > C$ ，则必有下列之一成立：

(1). $M > A.N$ ， $C = (\nu)\langle A.P'\rangle \mathbf{0}$ 且 $P' \equiv N.P$ ；

(2). $M = \epsilon$ 且 $P > C$ 。

证明 对 $M.P > C$ 的推导过程进行归纳易证。此处略。 ∎

引理 4.12 如果 $M > A.M'$，则 $M.P > (\nu)\langle A.P'\rangle \mathbf{0}$，其中 $P' \equiv M'.P$。

证明 对 $M > A.M'$ 的推导过程进行归纳易证。此处略。 ∎

引理 4.13 如果 $M > \epsilon$ 且 $P > C$，则 $M.P > C$。

证明 由 $M > \epsilon$ 根据定义 4.10 易知，$M = \epsilon$，利用 **(Harden Empty)** 即得 $M.P > C$。 ∎

下面的引理 4.14 对证明引理 4.15 至关重要，由于证明过程较为冗长，详细的证明参见附录 2。

引理 4.14 如果 $P \equiv Q$ 且 $Q > (\nu\vec{q})\langle Q_1\rangle Q_2$，则存在 P_1、P_2 满足 $P > (\nu\vec{q})\langle P_1\rangle P_2$ 且 $P_1 \equiv Q_1$、$P_2 \equiv Q_2$。

证明 证明过程参见附录 2。 ∎

引理 4.15 如果 $P \equiv Q$ 且 $Q \xrightarrow{\alpha} Q'$，则存在 P' 满足：$P \xrightarrow{\alpha} P'$ 且 $P' \equiv Q'$。

证明 对 $Q \xrightarrow{\alpha} Q'$ 的推导过程进行归纳。

(Trans Cap) 此时有 $Q > (\nu\vec{r})\langle A.Q_1\rangle Q_2$、$\alpha = A$、$Q' = (\nu\vec{r})(Q_1 \mid Q_2)$ 且 $fn(A) \cap \{\vec{p}\} = \emptyset$。由引理 4.14 知，存在 P_1、P_2 满足 $P > (\nu\vec{r})\langle P_1\rangle P_2$、$P_1 \equiv Q_1$ 且 $P_2 \equiv Q_2$，再利用 **(Trans Cap)** 可得 $P \xrightarrow{A} (\nu\vec{r})(P_1 \mid P_2)$。令 $P' = (\nu\vec{r})(P_1 \mid P_2)$，易知 $P' \equiv Q'$，假设成立。

(Trans In) 此时有 $Q > (\nu\vec{r}_1)\langle n[Q_1]\rangle Q_2$、$Q_1 \xrightarrow{\overline{\mathtt{in}}\ m} Q_1'$、$Q_2 > (\nu\vec{r}_2)\langle m[Q_3]\rangle Q_4$、$Q_3 \xrightarrow{in\ n} Q_3'$、$\{\vec{r}_2\} \cap fn(n[Q_1]) = \emptyset$、$\{\vec{r}_2\} \cap \{\vec{r}_1\} = \emptyset$、$\alpha = \tau$ 且 $Q' = (\nu\vec{r}_1, \vec{r}_2)(n[m[Q_3'] \mid Q_1'] \mid Q_4)$。由引理 4.14 知，存在 P_1、P_2 使得 $P > (\nu\vec{r}_1)\langle n[P_1]\rangle P_2$、$P_1 \equiv Q_1$ 且 $P_2 \equiv Q_2$。根据归纳假设知，存在 P_1' 使得 $P_1 \xrightarrow{\overline{\mathtt{in}}\ m} P_1'$ 且 $P_1' \equiv Q_1'$。同理可知存在 P_3、P_4 和 P_3' 使得 $P_2 > (\nu\vec{r}_2)\langle m[P_3]\rangle P_4$、$P_3 \equiv Q_3$、$P_4 \equiv Q_4$、$P_3 \xrightarrow{in\ n} P_3'$ 且 $P_3' \equiv Q_3'$。由引理 4.3 知 $fn(n[P_1]) = fn(n[Q_1])$，因此 $\{\vec{r}_2\} \cap fn(n[P_1]) = \emptyset$。令 $P' = (\nu\vec{r}_1, \vec{r}_2)(n[m[P_3'] \mid P_1'] \mid P_4)$，易知 $P' \equiv Q'$。由 **(Trans In)** 知，$P \xrightarrow{\tau} P'$，假设成立。

(Trans Out)(Trans Open)(Trans I/O) 采用类似 **(Trans In)** 的分析易知假设成立，此处略。

(Trans Amb) 此时有 $Q > (\nu\vec{r})\langle n[Q_1]\rangle Q_2$、$Q_1 \xrightarrow{\tau} Q_1'$ 且 $Q' = (\nu\vec{r})(n[Q_1'] \mid Q_2)$。由引理 4.14 知，存在 P_1、P_2 使得 $P > (\nu\vec{r})\langle n[P_1]\rangle P_2$、$P_1 \equiv Q_1$ 且 $P_2 \equiv Q_2$。再由归纳假设知，存在 P_1' 使得 $P_1 \xrightarrow{\tau} P_1'$ 且 $P_1' \equiv Q_1'$。令 $P' = (\nu\vec{r})(n[P_1'] \mid P_2)$，易知 $P' \equiv Q'$。而由 **(Trans Amb)** 可得 $P \xrightarrow{\tau} P'$，因此假设成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

命题 4.16 如果 $P \longrightarrow P'$，则 $P \xrightarrow{\tau} \equiv P'$。

证明 对 $P \longrightarrow P'$ 的推导过程进行归纳。

(Red In) 此时有 $P = m[in\ n.P_1 \mid P_2] \mid n[\overline{\mathtt{in}}\ m.Q_1 \mid Q_2]$ 且 $P' = n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2]$。根据硬化关系和标号转移语义的定义，利用 **(Trans In)** 易知 $P \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] \mid \mathbf{0} \equiv P'$，假设成立。

(Red Out)(Red Open) 采用类似 **(Trans In)** 的方法易知假设成立，此处略。

(Red Res) 此时有 $P=(\nu n)P_1$ 、 $P'=(\nu n)P_1'$ 且 $P_1 \longrightarrow P_1'$ 。由归纳假设知 $P_1 \xrightarrow{\tau}\equiv P_1'$ ，即存在 Q 使得 $P_1 \xrightarrow{\tau} Q$ 且 $Q \equiv P_1'$ 。由 $n \notin fn(\tau)=\emptyset$ 利用引理 4.9 得 $(\nu n)P_1 \xrightarrow{\tau}\equiv (\nu n)Q$ ，而 $(\nu n)Q \equiv (\nu n)P_1'$ ，因此有 $(\nu n)P_1 \xrightarrow{\tau}\equiv (\nu n)P_1'$ ，假设成立。

(Red Amb) 此时有 $P=n[P_1]$ 、 $P'=n[P_1']$ 且 $P_1 \longrightarrow P_1'$ 。由归纳假设知 $P_1 \xrightarrow{\tau}\equiv P_1'$ ，即存在 Q 使得 $P_1 \xrightarrow{\tau} Q$ 且 $Q \equiv P_1'$ 。根据 **(Harden Amb)** 可知 $n[P_1] > (\nu)\langle n[P_1]\rangle \mathbf{0}$ ，利用 **(Trans Amb)** 可得 $n[P_1] \xrightarrow{\tau} n[Q] \mid \mathbf{0}$ ，而 $n[Q] \mid \mathbf{0} \equiv n[P_1']$ ，所以 $n[P_1] \xrightarrow{\tau}\equiv n[P_1']$ ，假设成立。

(Red Par) 此时有 $P=P_1 \mid P_2$ 、 $P'=P_1' \mid P_2$ ，且 $P_1 \longrightarrow P_1'$ 。由归纳假设知 $P_1 \xrightarrow{\tau}\equiv P_1'$ ，即存在 Q 使得 $P_1 \xrightarrow{\tau} Q$ 且 $Q \equiv P_1'$ 。利用引理 4.7 可得 $P_1 \mid P_2 \xrightarrow{\tau}\equiv Q \mid P_2$ ，而 $Q \mid P_2 \equiv P_1' \mid P_2$ ，所以 $P_1 \mid P_2 \xrightarrow{\tau}\equiv P_1' \mid P_2$ ，假设成立。

(Red Struct) 此时有 $P \equiv P_1$ 、 $P_1 \longrightarrow P_1'$ 、 $P_1' \equiv P'$ 。由归纳假设知 $P_1 \xrightarrow{\tau}\equiv P_1'$ ，即存在 Q 使得 $P_1 \xrightarrow{\tau} Q$ 且 $Q \equiv P_1'$ 。由引理 4.15 知，存在 Q' 使得 $P \xrightarrow{\tau} Q'$ 且 $Q' \equiv Q$ 。而 $Q \equiv P_1' \equiv P'$ ，所以 $P \xrightarrow{\tau}\equiv P'$ ，假设成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

由命题 4.6 和命题 4.16 ，即可得出下面关于标号转移语义和归约语义等价的结论。

定理 4.17 （等价性） $P \longrightarrow Q$ 当且仅当 $P \xrightarrow{\tau}\equiv Q$ 。

证明 根据命题 4.6（运用一次结构同余关系）和命题 4.16 即可得出结论。 ∎

本章小结

本章基于 ROAM 进程的硬化关系，给出了与归约语义等价的标号转移语义。本章的主要工作是标号转移语义的定义和两种语义等价性的证明，以保证两种语义定义的一致性，以及下文基于标号转移语义的工作将同样可用于归约语义。本章关于标号转移语义的工作为下文进程的等价性研究打下了基础。

第五章　等价性判定

本章在 ROAM 标号转移语义的基础上，研究了判定两个 ROAM 进程等价的一般方法。本章的研究对象为基于观察结论的上下文等价（ contextual equivalence ），即：如果两个进程等价，则把它们放在任何上下文中，得到的结果从外部观察是无法区别的。同时，为了得到判定进程上下文等价的方法，本章研究了一类简化的上下文，并指出只要两个进程放在任何简化的上下文中看不出区别，则放在任意的上下文中也不会看出区别，这样就把研究将进程放在任何上下文中可能会发生的情况简化为只研究将进程放在这类特定的上下文中会发生的情况。

本章对 ROAM 等价性的研究方法来源于文献 [20] 对 MA 等价性的研究。类似文献 [20]，本章的主要结论有两个。首先，基于一类特殊的上下文，分析得出了进程与之发生交互的所有可能情况及归约结果（定理 5.20 ）。其次，证明了常规的基于上下文的进程等价关系等同于基于该类特殊上下文的进程等价关系（定理 5.39 ）。

§5.1　上下文等价

上下文等价（ contextual equivalence ）是判定进程等价的通用方法：两个进程称为上下文等价当且仅当它们被放置在任何上下文中所得的结果都对外部的观察（ observation ，或 barb ）反映出相同的结果。这种等价性的一个典型的（非进程代数）例子是：

冒泡排序 $\approx$ 快速排序

作为等价判定的依据，观察方法的选择显得尤为重要。在 π 演算中，观察使用非约束通道名的存在 [8] 。在 MA 中，观察的方法是使用最外层非约束灰箱的存在 [5, 19, 20] 。然而与 π 演算和 MA 不同，在 ROAM 中，并非所有最外层非约束灰箱的存在均可被外部的观察所感知。例如，考虑进程 $n[\overline{\mathtt{out}}\ m]$ ，虽然 n 为非约束，但是无论外部使用什么观察方法， n 中都没有任何动作可以与外界进行归约，此时， n 的存在是无法为外部感知的。因此，在对 ROAM 进程选择观察方法时，还要考虑外层非约束灰箱中顶层进程可以执行的动作。在六种基本的动作中，动作 $\overline{\mathtt{out}}$ 和 open 无法与灰箱外部进程发生归约，其它动作则可与外部发生归约，当然，其参数必须为自由的。

令集合 $\mathbf{BarbedAction} \triangleq \{\mathtt{in}\ n,\ \overline{\mathtt{in}}\ n,\ \mathtt{out}\ n,\ \overline{\mathtt{open}} \mid n \in \mathcal{N}\}$ ，表示那些可用于观察的动作。定义记号 $P \downarrow_n$ 和 $P \Downarrow_n$ 表示 ROAM 中的外部观察。

定义 5.1 （ $P \downarrow_n$ ）　$P \downarrow_n \iff$ 存在 $\vec{p}$ 、A 、P_1 、P_2 和 P_3 满足 $A \in \mathbf{BarbedAction}$ 、$(fn(A) \cup \{n\}) \cap \{\vec{p}\} = \emptyset$ 且 $P \equiv (\nu\vec{p})(n[A\,.\,P_1 \mid P_2] \mid P_3)$ 。

定义 5.2 （ $P \Downarrow_n$ ）

(Conv Exb)　　$P \downarrow_n \Longrightarrow P \Downarrow_n$

(Conv Red)　　$P \longrightarrow P'$ 且 $P' \Downarrow_n \Longrightarrow P \Downarrow_n$

用 $\mathcal{C}()$ 表示一个进程上下文（ context ）。它包含零个、一个或多个空缺位置（用“ () ”表示）。在这些位置填入任意进程 P ，所得到的结果也是一个进程，用 $\mathcal{C}(P)$ 来表示。值得注意，在 P 中

自由的名字可能在 $\mathcal{C}(P)$ 中被绑定，变成约束名。因此，对进程上下文中的约束名改名后的结果不同于原来的上下文。

ROAM 中进程的上下文等价关系“$\approx$”由以下定义给出。

定义 5.3 （**上下文等价**） $P \approx Q \iff$ 对任意灰箱名 n 和进程上下文 $\mathcal{C}()$，$\mathcal{C}(P) \Downarrow_n \iff \mathcal{C}(Q) \Downarrow_n$。

上下文等价关系的一个重要性质是它的同余性（ conguence ），即上下文等价的进程放入任何上下文得到的结果进程也是上下文等价的。

定义 5.4 （**同余关系**） 关系 $\mathcal{R}$ 称为一个同余关系，如果 $\mathcal{R}$ 满足：

(1). $\mathcal{R}$ 是自反、对称、传递的；

(2). 对任意 P 、 Q 、 $\mathcal{C}()$， $P\ \mathcal{R}\ Q \implies \mathcal{C}(P)\ \mathcal{R}\ \mathcal{C}(Q)$。

显然，结构同余关系“$\equiv$”为一个同余关系。同样，上下文等价关系也为一同余关系。

命题 5.5 上下文等价关系为一同余关系。

证明 直接由上下文等价关系的定义和同余关系的定义易证。 ∎

引理 5.6 如果 $P \equiv Q$，那么 $P \downarrow_n \Longrightarrow Q \downarrow_n$，更进一步，如果 $P \Downarrow_n$ 那么 $Q \Downarrow_n$ 且这两个结论的推导步骤相同。

证明 第一部分由“$\downarrow$”的定义易证。第二部分可通过分析 $P \Downarrow_n$ 的推导过程得到。 ∎

命题 5.7 如果 $P \equiv Q$，则 $P \approx Q$。

证明 利用上下文等价的定义和引理 5.6 易证。 ∎

下面我们举一些例子说明上下文等价的判定。首先，我们给出一些进程上下文不等价的例子，判定的方法只要给出可区别它们的一个特殊上下文。

例 5.8 如果 $m \neq n$，则 $m[\overline{\texttt{open}}] \not\approx n[\overline{\texttt{open}}]$。

证明 取上下文 $\mathcal{C}() = ()$。由 $\mathcal{C}(m[\overline{\texttt{open}}]) = m[\overline{\texttt{open}}]$ 可知 $\mathcal{C}(m[\overline{\texttt{open}}]) \downarrow_m$，再由 **(Conv Exb)** 可得 $\mathcal{C}(m[\overline{\texttt{open}}]) \Downarrow_m$，而 $\mathcal{C}(n[\overline{\texttt{open}}]) = n[\overline{\texttt{open}}] \not\Downarrow_m$ 且 $\mathcal{C}(n[\overline{\texttt{open}}]) \not\longmapsto P$，因此 $\mathcal{C}(n[\overline{\texttt{open}}]) \not\Downarrow_m$。
∎

例 5.9 如果 $m \neq n$，则 $\texttt{in}\, m \not\approx \texttt{in}\, n$。

证明 取上下文 $\mathcal{C}() = n[()] \mid m[\overline{\texttt{in}}\, n\, .\, \overline{\texttt{out}}\, p \mid p[\texttt{out}\ m\, .\, \overline{\texttt{open}}]]$，则由 $\mathcal{C}(\texttt{in}\ m) \Downarrow_p$ 但 $\mathcal{C}(\texttt{in}\ n) \not\Downarrow_p$ 可知 $\texttt{in}\ m \not\approx \texttt{in}\ n$。 ∎

然而，对于进程上下文等价的例子，因为要考虑所有的上下文，证明起来就有不那么容易了。以下是一个简单的空灰箱回收定律：

定理 5.10 (Coll Garb) $n[\,] \approx \mathbf{0}$

证明 式子左边的进程不能和外界发生任何交互，不能得到对任何灰箱名的观察结果，也不能与任何上下文发生归约，因此将左右两式放在任何上下文中所得到的观察是完全相同的。根据上下文等价的定义，结论成立。 ∎

例 5.11 $(\nu m)n[\,\mathtt{in}\,m\,.\,P] \approx \mathbf{0}$ ； $(\nu n)n[\,\overline{\mathtt{open}}\,.\,P] \approx \mathbf{0}$ 。

容易证明，上例中的两个进程均和静止进程等价，这里的 $\mathtt{in}\ m$ 也可以是 $\overline{\mathtt{in}}\ m$ 、 $\mathtt{out}\ m$ ，但注意如果动作是 $\overline{\mathtt{out}}\ m$ ，就不成立了。

例 5.12 $n[\,\overline{\mathtt{out}}\,m\,.\,P \mid m[\,\mathtt{out}\,n\,.\,Q]\,] \approx n[P] \mid m[Q]$ 。

例 5.12 中，无论进程处在什么样的上下文， m 从 n 中移出这个动作必然首先发生。但是要形式化地证明该结论，却不那么容易。

本章及后续的章节将围绕如何进行类似复杂进程的等价性证明展开。在本章中将提供证明进程等价性的一般性结论。在下一章将给出一些直接判定有关进程等价的定律。

§5.2 简单上下文等价及其判定

上下文等价关系虽然是判断进程等价的一个基本准则，但由于上下文的多样性，具体操作起来却有较大的困难。常用的解决方法是将等价性判定的任意上下文条件适当放宽，只要满足对一类特殊上下文的观察等价性，就认为进程等价。同时说明进程的这种特殊等价关系与上下文等价关系的异同。本节基于同样的思想，通过一类特定的上下文——简单上下文及其相应的等价性判定方法，来解决一般上下文环境下的等价性判定问题。

本节首先定义了简单上下文和基于简单上下文的进程等价关系。其次给出了进程和任何简单上下文发生交互的所有情况及归约结果（定理 5.20 ），作为判断两个进程是否具有简单上下文等价关系的重要工具。简单上下文等价和上下文等价之间的相互联系将在下一节进行证明。

§5.2.1 简单上下文等价

文献 [20] 给出了 MA 进程等价关系中的一些结论，其中使用了基于 **简单上下文**（ harness ）的进程等价关系，并证明简单上下文等价与上下文等价是同一个关系。在 ROAM 中，我们采用了与 MA 中类似的方法，证明文献 [20] 中简单上下文理论对 ROAM 来说同样适用。

简单上下文为一类特殊的上下文，它只允许填入进程与其它进程发生三种基本的组合关系：约束、并置和处于灰箱内部，其中用符号 “ $-$ ” 表示一个简单上下文中进程的填入位置。简单上下文一般用 H 表示，定义如下：

$$H ::= \ - \ \mid (\nu n)H \mid P \mid H \mid H \mid Q \mid n[H]$$

用 $H(P)$ 表示将 P 填入 H 中位置所得的结果。显然，该结果为一个合法的进程。对于简单上下文也定义类似的取自由名函数： $fn(H)$ 。另外，与 P 类似，可定义 $H \equiv H'$ 、 $H \longrightarrow H'$ 、 $H > C$ 、 $H \xrightarrow{\alpha} H'$ 等关系，不同在于需要将原来可出现进程的地方用 H 来代替。例如：

$$P \mid H \equiv H \mid P$$

$$P \longrightarrow Q \Longrightarrow P \mid H \longrightarrow Q \mid H$$

$$n[H] > (\nu)\langle n[H]\rangle \mathbf{0}$$

$$\mathtt{in}\, n\,.\,P \mid H \xrightarrow{\mathtt{in}\, n} P \mid H$$

对于进程的上下文和简单上下文，需要做以下两点说明：

(1). $\mathcal{C}()$ 中的位置个数不限，而 H 中有且只有一个位置；

(2). 与 $\mathcal{C}()$ 不同，为了减少 H 的复杂性，规定 H 中的约束名不能在进程填入时发生名字捕获，即： H 中的约束名可进行任意换名。

基于简单上下文，类似定义进程间的简单上下文等价关系如下：

定义 5.13 （**简单上下文等价**） 称进程 P 、 Q 简单上下文等价，如果对任意 H 和 n 有 $H(P) \Downarrow_n \iff H(Q) \Downarrow_n$ ，记为： $P \simeq Q$ 。

§5.2.2 简单上下文等价的判定

简单上下文等价关系的判定关键在于对 $H(P) \Downarrow_n$ 的分析，也就是对 $H(P)$ 可能的归约结果的分析。而 $H(P)$ 的归约结果分析要比 $\mathcal{C}(P)$ 的归约结果分析简单得多。

借助 ROAM 的标号转移语义，对 $H(P) \longrightarrow R$ 的分析可以转化为对 $H(P) \xrightarrow{\tau} R$ 的分析，而标号转移语义定义在硬化关系之上，因此，首先要分析的就是 $H(P)$ 可能的硬化结果。

引理 5.14 如果 $H(P) > (\nu\vec{p})\langle P_1\rangle P_2$ ，那么以下之一成立：

(1). $H > (\nu\vec{p})\langle n[H']\rangle P_2$ 且 $P_1 = n[H'(P)]$ ；

(2). $H > (\nu\vec{p})\langle P_1\rangle H'$ 且 $P_2 = H'(P)$ ；

(3). $P > (\nu\vec{p})\langle P_1\rangle P'$ 、 $H \equiv - \mid R$ 、 $P_2 \equiv P' \mid R$ 且 $\{\vec{p}\} \cap fn(R) = \emptyset$ 。

证明 对 $H(P) > (\nu\vec{p})\langle P_1\rangle P_2$ 的推导过程进行归纳。

(Harden Action) 此时有 $H(P) = A\,.\,Q$ 、 $A \in \mathbf{Action}$ 且 $(\nu\vec{p})\langle P_1\rangle P_2 = (\nu)\langle A\,.\,Q\rangle \mathbf{0}$ 。由简单上下文的定义可知， $H(P) = A\,.\,Q$ 只可能由 $H = -$ 与 $P = A\,.\,Q$ 得到。于是 $P = A\,.\,Q > (\nu)\langle A\,.\,Q\rangle \mathbf{0}$ ，令 $R = \mathbf{0}$ ，第三种情况成立。

(Harden Amb) 此时有 $H(P) = n[Q]$ 且 $(\nu\vec{p})\langle P_1\rangle P_2 = (\nu)\langle n[Q]\rangle \mathbf{0}$ 。 $H(P) = n[Q]$ 只可能来自下列情况：

- $H = -$ 且 $P = n[Q]$ ：此时有 $P > (\nu\vec{p})\langle P_1\rangle P_2$ ，取 $R = \mathbf{0}$ 即有第三种情况成立。
- $H = n[H']$ 且 $Q = H'[P]$ ：此时有 $H > (\nu)\langle n[H']\rangle \mathbf{0}$ 且 $P_1 = n[Q] = n[H'(P)]$ ，取 $P_2 = \mathbf{0}$ 即有第一种情况成立。

(Harden Par 1) 此时有 $H(P) = Q_1 \mid Q_2$ 、 $Q_1 > (\nu\vec{p})\langle P_1\rangle P_3$ 、 $P_2 = P_3 \mid Q_2$ 且 $\{\vec{p}\} \cap fn(Q_2) = \emptyset$ 。由简单上下文的定义可知， $H(P) = Q_1 \mid Q_2$ 只可能来自下列情况：

- $H = -$ 且 $P = Q_1 \mid Q_2$ ：此时有 $P > (\nu\vec{p})\langle P_1\rangle P_2$ ，取 $R = \mathbf{0}$ 即有第三种情况成立。
- $H = H_1 \mid Q_2$ 且 $Q_1 = H_1(P)$ ：此时有 $H_1(P) > (\nu\vec{p})\langle P_1\rangle P_3$ 。由归纳假设知，必有下列情况之一成立：

- $H_1 > (\nu\vec{p})\langle n[H_1']\rangle P_3$ 且 $P_1 = n[H_1'(P)]$：此时由 $\{\vec{p}\} \cap fn(Q_2) = \emptyset$ 可知 $H > (\nu\vec{p})\langle n[H_1']\rangle (P_3 \mid Q_2)$，即 $H > (\nu\vec{p})\langle n[H_1']\rangle P_2$，取 $H' = H_1'$，即有结论的第一种情况成立。
- $H_1 > (\nu\vec{p})\langle P_1\rangle H_1'$ 且 $P_3 = H_1'(P)$：此时由 $\{\vec{p}\} \cap fn(Q_2) = \emptyset$ 可知 $H > (\nu\vec{p})\langle P_1\rangle (H_1' \mid Q_2)$ 且 $P_2 = P_3 \mid Q_2 = H_1'(P) \mid Q_2$，取 $H' = H_1' \mid Q_2$，即有结论的第二种情况成立。
- $P > (\nu\vec{p})\langle P_1\rangle P'$、$H_1 = - \mid R'$、$P_3 = P' \mid R'$ 且 $\{\vec{p}\} \cap fn(R') = \emptyset$。此时有 $H = H_1 \mid Q_2 = - \mid R' \mid Q_2$、$P_2 = P_3 \mid Q_2 = P' \mid R' \mid Q_2$ 且 $\{\vec{p}\} \cap fn(R' \mid Q_2) = \emptyset$，取 $R = R' \mid R_2$，即有结论的第三种情况成立。

- $H = Q_1 \mid H_2$ 且 $Q_2 = H_2(P)$：此时由 $\{\vec{p}\} \cap fn(Q_2) = \emptyset$ 可知 $\{\vec{p}\} \cap fn(H_2) = \emptyset$，使用 **(Harden Par 1)** 可得 $H > (\nu\vec{p})\langle P_1\rangle (P_3 \mid H_2)$ 且 $P_2 = P_3 \mid Q_2 = P_3 \mid H_2(P)$。令 $H' = P_3 \mid H_2$，即有结论的第二种情况成立。

(Harden Par 2) 采用类似 **(Harden Par 2)** 的分析同理可证，此处略。

(Harden Res) 此时有 $H(P) = (\nu m)Q$、$Q > (\nu\vec{q})\langle Q_1\rangle Q_2$ 且 $(\nu\vec{p})\langle P_1\rangle P_2 = \overline{(\nu m)}(\nu\vec{q})\langle Q_1\rangle Q_2$。

首先，由 $H(P) = (\nu m)Q$ 可知 $m \notin fn(H(P))$，而 $fn(P) \subseteq fn(H(P))$，因此 $m \notin fn(P)$。

我们考虑 $H(P) = (\nu m)Q$ 的产生，显然如果 $H = -$，则取 $R = \mathbf{0}$ 可知第三种情况成立；否则，有 $H = (\nu m)H_1$ 且 $Q = H_1(P)$，下面根据 $C = \overline{(\nu m)}(\nu\vec{q})\langle Q_1\rangle Q_2$ 在不同情况下的取值逐一分析：

- $m \notin fn(Q_1)$：此时 $C = (\nu\vec{q})\langle Q_1\rangle (\nu m)Q_2 = (\nu\vec{p})\langle P_1\rangle P_2$，因此有 $\vec{p} = \vec{q}$、$P_1 = Q_1$ 且 $P_2 = (\nu m)Q_2$。此时有 $H_1(P) = Q > (\nu\vec{p})\langle P_1\rangle Q_2$。由归纳假设可知，必有以下情况之一成立：
  - $H_1 > (\nu\vec{p})\langle n[H_1']\rangle Q_2$ 且 $n[H_1'(P)] = P_1$：此时有 $m \notin fn(Q_1) = fn(P_1) = fn(n[H_1'(P)])$，因此 $H = (\nu m)H_1 > \overline{(\nu m)}(\nu\vec{p})\langle n[H_1']\rangle Q_2 = (\nu\vec{p})\langle n[H_1']\rangle (\nu m)Q_2 = (\nu\vec{p})\langle n[H_1']\rangle P_2$，再根据 $P_1 = n[H_1'(P)]$，令 $H' = H_1'$ 可知结论的第一种情况成立。
  - $H_1 > (\nu\vec{p})\langle P_1\rangle H_1'$ 且 $Q_2 = H_1'(P)$：此时有 $H > \overline{(\nu m)}(\nu\vec{p})\langle P_1\rangle H_1' = (\nu\vec{p})\langle P_1\rangle (\nu m)H_1'$ 且 $P_2 = (\nu m)Q_2 = (\nu m)H_1'(P)$，令 $H' = (\nu m)H_1'$ 即有结论的第二种情况成立。
  - $P > (\nu\vec{p})\langle P_1\rangle P'$、$H_1 \equiv - \mid R'$、$Q_2 = P' \mid R'$ 且 $\{\vec{p}\} \cap fn(R') = \emptyset$：根据 $m \notin fn(P) \cup \{\vec{p}\}$ 以及 $P > (\nu\vec{p})\langle P_1\rangle P'$ 可知 $m \notin fn(P')$，因此 $P_2 = (\nu m)Q_2 \equiv (\nu m)(P' \mid R') \equiv P' \mid (\nu m)R'$，又有 $H = (\nu m)H_1 \equiv (\nu m)(- \mid R') \equiv - \mid (\nu m)R'$ 且 $\{\vec{p}\} \cap fn((\nu m)R') = \emptyset$，因此取 $R = (\nu m)R'$ 可知，结论的第三种情况成立。
- $Q_1 = m'[Q_1']$、$m \in fn(Q_1')$、$m \neq m'$ 且 $m \notin fn(Q_2)$：此时 $C = (\nu\vec{q})\langle m'[(\nu m)Q_1']\rangle Q_2 = (\nu\vec{p})\langle P_1\rangle P_2$，因此有 $\vec{p} = \vec{q}$、$P_1 = m'[(\nu m)Q_1']$ 且 $P_2 = Q_2$。此时 $H_1(P) = Q > (\nu\vec{q})\langle Q_1\rangle Q_2 = (\nu\vec{p})\langle m'[Q_1']\rangle P_2$。由归纳假设可知，必有以下情况之一成立：
  - $H_1 > (\nu\vec{p})\langle n[H_1']\rangle P_2$ 且 $n[H_1'(P)] = m'[Q_1']$：此时有 $n = m'$ 且 $Q_1' = H_1'(P)$。由条件 $m \in fn(Q_1')$、$m \notin fn(P)$ 可知 $m \in fn(H_1')$。同时注意到条件 $m \notin fn(Q_2) = fn(P_2)$ 和 $m \neq m' = n$，因此 $H = (\nu m)H_1 > \overline{(\nu m)}(\nu\vec{p})\langle n[H_1']\rangle P_2 = (\nu\vec{p})\langle n[(\nu m)H_1']\rangle P_2$ 且 $P_1 = m'[(\nu m)Q_1'] = n[(\nu m)H_1'(P)]$，令 $H' = (\nu m)H_1'$ 可知结论的第一种情况成立。
  - $H_1 > (\nu\vec{p})\langle m'[Q_1']\rangle H_1'$ 且 $P_2 = H_1'(P)$：由 $m \notin fn(Q_2) = fn(P_2) = fn(H_1'(P))$ 可知 $m \notin fn(H_1')$，因此 $H > \overline{(\nu m)}(\nu\vec{p})\langle m'[Q_1']\rangle H_1' = (\nu\vec{p})\langle m'[(\nu m)Q_1']\rangle H_1' = (\nu\vec{p})\langle P_1\rangle H_1'$，再根据 $P_2 = H_1'(P)$，令 $H' = H_1'$ 即有结论的第二种情况成立。

- $P > (\nu\vec{p})\langle m'[Q_1']\rangle P'$ 、 $H_1 \equiv - \mid R'$ 、 $P_2 = P' \mid R'$ 且 $\{\vec{p}\} \cap fn(R') = \emptyset$ ：根据 $m \in fn(Q_1')$ 、 $m \neq m'$ 和 $m \notin \{\vec{p}\}$ 可知 $m \in fn((\nu\vec{p})\langle m'[Q_1']\rangle P') = fn(P)$ ，这与已知 $m \notin fn(P)$ 矛盾，故此，该情况不可能发生。

• 其它情况，即： $m \in fn(Q_1)$ 且 (a) $m \in fn(Q_2)$, (b) Q_1 不是灰箱，或者 (c) Q_1 是灰箱，且 Q_1 的名字为 m ：此时 $C = (\nu\vec{q}, m)\langle Q_1\rangle Q_2 = (\nu\vec{p})\langle P_1\rangle P_2$ ，因此有 $\vec{p} = \vec{q}, m$ 、 $P_1 = Q_1$ 且 $P_2 = Q_2$ 。此时有 $H_1(P) = Q > (\nu\vec{q})\langle P_1\rangle P_2$ 。由归纳假设可知，必有以下情况之一成立：

- $H_1 > (\nu\vec{q})\langle n[H_1']\rangle P_2$ 且 $n[H_1'(P)] = P_1$ ：此时有 $H = (\nu m)H_1 > \overline{(\nu m)}(\nu\vec{q})\langle n[H_1']\rangle P_2$ 。在情况 (a) 成立时，有 $m \in fn(P_2)$ ，又由 $m \notin fn(P)$ 和 $m \in fn(P_1) = fn(n[H_1'(P)])$ 可知 $m \in fn(n[H_1'])$ ；由于 $Q_1 = n[H_1'(P)]$ ，情况 (b) 不能成立；如果情况 (c) 成立，则 $m = n$ 。因此在所有可能情况下都有 $\overline{(\nu m)}(\nu\vec{q})\langle n[H_1']\rangle P_2 = (\nu\vec{q}, m)\langle n[H_1']\rangle P_2 = (\nu\vec{p})\langle n[H_1']\rangle P_2$ ，再根据 $P_1 = n[H_1'(P)]$ ，令 $H' = H_1'$ 可知结论的第一种情况成立。
- $H_1 > (\nu\vec{q})\langle P_1\rangle H_1'$ 且 $P_2 = H_1'(P)$ ：此时 (a) 由 $m \in fn(P_2)$ 、 $m \notin fn(P)$ 可知 $m \in fn(H_1')$ ；(b) P_1 不是一个灰箱；(c) P_1 为灰箱且名字为 m 。再加上 $m \in fn(P_2)$ 可知无论何种情况都有 $H > \overline{(\nu m)}(\nu\vec{q})\langle P_1\rangle H_1' = (\nu\vec{p})\langle P_1\rangle H_1'$ 且 $P_2 = H_1'(P)$ ，令 $H' = H_1'$ 即有结论的第二种情况成立。
- $P > (\nu\vec{q})\langle P_1\rangle P'$ 、$H_1 \equiv - \mid R'$ 、$P_2 = P' \mid R'$ 且 $\{\vec{p}\} \cap fn(R') = \emptyset$ ：根据 $m \in fn(P_1)$ 和 $m \notin \{\vec{q}\}$ 可知 $m \in fn((\nu\vec{q})\langle P_1\rangle P') = fn(P)$ ，这与已知 $m \notin fn(P)$ 矛盾，故此，该情况不可能发生。

(Harden Repl)(Harden Empty)(Harden Path) 类似 **(Harden Action)** 的分析可知第三种情况成立，此处略。

通过以上对所有可能情况的分析可知，结论成立。 ∎

在引理 5.14 的基础上，下面分析 $H(P) \xrightarrow{\tau} R$ 的各种可能情况。

由于进程与简单上下文的交互方式众多，首先使用简写 $H \bullet P \rightsquigarrow R$ 表示这些交互方式中的某一种。在 MA 中，共有 9 种不同的交互方式（包括涉及本地通讯原语的 2 种），在 ROAM 中，交互方式增加到以下的 11 种。

定义 5.15 $H \bullet P \rightsquigarrow R \iff$ 存在 H' 、$\vec{r}$ 满足 $\{\vec{r}\} \cap fn(P) = \emptyset$ ，且以下之一成立：

(Inter In) $H \equiv (\nu\vec{r})H'(m[- \mid R_1] \mid n[R_2])$ 、$P \xrightarrow{\mathtt{in}\ n} P'$ 、$R_2 \xrightarrow{\overline{\mathtt{in}}\ m} R_2'$ 且 $R \equiv (\nu\vec{r})H'(n[m[P' \mid R_1] \mid R_2'])$

(Inter Out) $H \equiv (\nu\vec{r})H'(n[m[- \mid R_1] \mid R_2])$ 、$P \xrightarrow{\mathtt{out}\ n} P'$ 、$R_2 \xrightarrow{\overline{\mathtt{out}}\ m} R_2'$ 且 $R \equiv (\nu\vec{r})H'(m[P' \mid R_1] \mid n[R_2'])$

(Inter Open) $H \equiv (\nu\vec{r})H'(- \mid n[R_1])$ 、$P \xrightarrow{\mathtt{open}\ n} P'$ 、$R_1 \xrightarrow{\overline{\mathtt{open}}} R_1'$ 且 $R \equiv (\nu\vec{r})H'(P' \mid R_1')$

(Inter Co-in) $H \equiv (\nu\vec{r})H'(m[R_1] \mid n[- \mid R_2])$ 、$P \xrightarrow{\overline{\mathtt{in}}\ m} P'$ 、$R_1 \xrightarrow{\mathtt{in}\ n} R_2'$ 且 $R \equiv (\nu\vec{r})H'(n[m[R_1'] \mid P' \mid R_2])$

(Inter Co-out) $H \equiv (\nu\vec{r})H'(n[m[R_1] \mid - \mid R_2])$ 、$P \xrightarrow{\overline{\mathtt{out}}\ m} P'$ 、$R_1 \xrightarrow{\mathtt{out}\ n} R_1'$ 且 $R \equiv (\nu\vec{r})H'(m[R_1'] \mid P' \mid R_2)$

(Inter Co-open) $H \equiv (\nu\vec{r})H'(n[- \mid R_1] \mid R_2)$ 、$P \xrightarrow{\overline{\mathtt{open}}} P'$ 、$R_2 \xrightarrow{\mathtt{open}\ n} R_2'$ 且 $R \equiv (\nu\vec{r})H'(P' \mid R_1 \mid$

R_2)

(Inter Amb In) $P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、 $Q \xrightarrow{\text{in } m} Q'$ 、 $H \equiv (\nu\vec{r})H'(- \mid m[R_1])$ 、 $R_1 \xrightarrow{\overline{\text{in}}\, n} R_1'$ 、 $\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H'((\nu\vec{p})(m[n[Q'] \mid R'] \mid P'))$

(Inter Amb Co-in) $P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、 $Q \xrightarrow{\overline{\text{in}}\, m} Q'$ 、 $H \equiv (\nu\vec{r})H'(- \mid m[R_1])$ 、 $R_1 \xrightarrow{\text{in } n} R_1'$ 、 $\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H'((\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P'))$

(Inter Amb Out 1) $P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、 $Q \xrightarrow{\text{out } m} Q'$ 、 $H \equiv (\nu\vec{r})H'(m[- \mid R_1])$ 、 $R_1 \xrightarrow{\overline{\text{out}}\, n} R_1'$ 、 $\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H'((\nu\vec{p})(n[Q'] \mid m[P' \mid R_1']))$

(Inter Amb Out 2) $P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、 $Q \xrightarrow{\text{out } m} Q'$ 、 $P' \xrightarrow{\overline{\text{out}}\, n} P''$ 、 $H \equiv (\nu\vec{r})H'(m[- \mid R_1])$ 、 $\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H'((\nu\vec{p})(n[Q'] \mid m[P'' \mid R_1]))$

(Inter Amb Co-open) $P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、 $Q \xrightarrow{\overline{\text{open}}} Q'$ 、 $H \equiv (\nu\vec{r})H'(- \mid R_1)$ 、 $R_1 \xrightarrow{\text{open } n} R_1'$ 、 $\{\vec{p}\} \cap fn(R_1) = \emptyset$ 且 $R \equiv (\nu\vec{r})H'((\nu\vec{p})(Q' \mid P' \mid R_1'))$

本节的主要任务是证明任意简单上下文 H 和任意进程 P 的交互方式只有上述 $H \bullet P \rightsquigarrow R$ 定义中给出的 11 种（定理 5.20 ）。下面首先给出证明过程需要用到的一些引理。

引理 5.16 如果 $H(P) > (\nu\vec{p})\langle n[P_1]\rangle P_2$ ，那么以下之一成立：

(1). $H > (\nu\vec{p})\langle n[H']\rangle P_2$ 且 $P_1 = H'(P)$ ；

(2). $H > (\nu\vec{p})\langle n[P_1]\rangle H'$ 且 $P_2 = H'(P)$ ；

(3). $P > (\nu\vec{p})\langle n[P_1]\rangle P'$ 、 $H \equiv - \mid R$ 、 $P_2 \equiv P' \mid R$ 且 $\{\vec{p}\} \cap fn(R) = \emptyset$ 。

证明 引理 5.14 的一个特例，易证。 ∎

引理 5.17 如果 $H(P) \xrightarrow{A} R$ ，则以下之一成立：

(1). $H \equiv (\nu\vec{r})(A\,.\,R' \mid H')$ 、 $R \equiv (\nu\vec{r})(R' \mid H'(P))$ 且 $\{\vec{r}\} \cap (fn(A) \cup fn(P)) = \emptyset$ ；

(2). $H \equiv - \mid R'$ 、 $P \xrightarrow{A} P'$ 且 $R \equiv P' \mid R'$ 。

证明 由 **(Trans Cap)** 利用引理 5.14 不难证明，此处略。 ∎

引理 5.18 如果 $P \mid Q \xrightarrow{A} R$ ，则以下之一成立：

(1). $P \xrightarrow{A} P'$ 且 $R \equiv P' \mid Q$ ；

(2). $Q \xrightarrow{A} Q'$ 且 $R \equiv P \mid Q'$ 。

证明 由 **(Trans Cap)** 的定义不难证明，此处略。 ∎

下面首先根据标号转移语义的定义和上述几个引理的结论证明进程和简单上下文交互的可能性（引理 5.19 ），再根据标号转移语义和归约语义的等价性，给出最终的简单上下文等价判定方法（定理 5.20 ）。

引理 5.19 如果 $H(P) \xrightarrow{\tau} R$ 那么以下之一成立：

(1). $P \longrightarrow P'$ 且 $R \equiv H(P')$ ；

(2). $H \longrightarrow H'$ 且 $R \equiv H'(P)$ ；

(3). $H \bullet P \rightsquigarrow R$ 。

证明 对 $H(P) \xrightarrow{\tau} R$ 的推导过程进行归纳。

(Trans In) 此时有 $H(P) > (\nu\vec{q})\langle n[Q_1]\rangle Q_2$ 、 $Q_1 \xrightarrow{\overline{\mathtt{in}}\, m} Q_1'$ 、 $Q_2 > (\nu\vec{r})\langle m[Q_3]\rangle Q_4$ 、 $Q_3 \xrightarrow{\mathtt{in}\, n} Q_3'$ 、 $\{\vec{r}\} \cap fn(n[Q_1]) = \emptyset$ 、 $\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 且 $R = (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid Q_1'] \mid Q_4)$ 。根据 $H(P) > (\nu\vec{q})\langle n[Q_1]\rangle Q_2$ 由引理 5.16 知，必有以下情况之一成立：

- $H \equiv (\nu\vec{q})(n[H_1] \mid Q_2)$ 且 $Q_1 = H_1(P)$ ：此时有 $H_1(P) \xrightarrow{\overline{\mathtt{in}}\, m} Q_1'$ ，根据引理 5.17 可知，必有以下情况之一成立：
  - $H_1 \equiv (\nu\vec{s})(\overline{\mathtt{in}}\ m\,.\,R' \mid H_2)$ 、 $Q_1' \equiv (\nu\vec{s})(R' \mid H_2(P))$ 且 $\{\vec{s}\} \cap (\{m\} \cup fn(P)) = \emptyset$ ：此时，由于 $\vec{s}$ 为约束名集合，不妨设 $\{\vec{s}\} \cap (\{n,\vec{r},\vec{q}\} \cup fn(Q_2)) = \emptyset$ 。令 $H' = (\nu\vec{q},\vec{r},\vec{s})(n[m[Q_3'] \mid R' \mid H_2] \mid Q_4)$ ，由 $Q_2 > (\nu\vec{r})\langle m[Q_3]\rangle Q_4$ 且 $Q_3 \xrightarrow{\mathtt{in}\, n} Q_3'$ 知，$H = (\nu\vec{q})(n[(\nu\vec{s})(\overline{\mathtt{in}}\ m\,.\,R' \mid H_2)] \mid Q_2) \longrightarrow H'$ ，且 $R \equiv H'(P)$ ，结论的第二种情况成立。
  - $H_1 \equiv - \mid R'$ 、 $P \xrightarrow{\overline{\mathtt{in}}\, m} P'$ 且 $Q_1' \equiv P' \mid R'$ ：由已知 $Q_2 > (\nu\vec{r})\langle m[Q_3]\rangle Q_4$ 利用引理 4.4 可知 $Q_2 \equiv (\nu\vec{r})(m[Q_3] \mid Q_4)$ ，再由条件 $\{\vec{r}\} \cap fn(n[Q_1]) = \emptyset$ 有 $H = (\nu\vec{q})(n[- \mid R'] \mid Q_2) \equiv (\nu\vec{q},\vec{r})(n[- \mid R'] \mid m[Q_3] \mid Q_4)$ ，另外我们有 $P \xrightarrow{\overline{\mathtt{in}}\, m} P'$ 、 $Q_3 \xrightarrow{\mathtt{in}\, n} Q_3'$ 且 $R \equiv (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid P' \mid R'] \mid Q_4)$ ，由 **(Inter Co-in)** 知 $H \bullet P \rightsquigarrow R$ ，第三种情况成立。
- $H \equiv (\nu\vec{q})(n[Q_1] \mid H_1)$ 且 $Q_2 = H_1(P)$ ：此时有 $H_1(P) > (\nu\vec{r})\langle n[Q_3]\rangle Q_4$ ，根据引理 5.16 可知，必有以下情况之一成立：
  - $H_1 \equiv (\nu\vec{r})(m[H_2] \mid Q_4)$ 且 $Q_3 = H_2(P)$ ：此时有 $H_2(P) \xrightarrow{\mathtt{in}\, n} Q_3'$ ，根据引理 5.17 可知，必有以下情况之一成立：
    (1). $H_2 \equiv (\nu\vec{s})(\mathtt{in}\ n\,.\,R' \mid H_3)$ 、 $Q_3' \equiv (\nu\vec{s})(R' \mid H_3(P))$ 且 $\{\vec{s}\} \cap (\{n\} \cup fn(P)) = \emptyset$ ：此时，由于 $\vec{s}$ 为约束名集合，不妨设 $\{\vec{s}\} \cap (\{m,\vec{q},\vec{r}\} \cup fn(Q_1) \cup fn(Q_4)) = \emptyset$ 。令 $H' = (\nu\vec{q},\vec{r},\vec{s})(n[m[R' \mid H_3] \mid Q_1'] \mid Q_4)$ ，由 $Q_1 \xrightarrow{\overline{\mathtt{in}}\, m} Q_1'$ 知，$H = (\nu\vec{q})(n[Q_1] \mid (\nu\vec{r})(m[(\nu\vec{s})(\mathtt{in}\ n\,.\,R' \mid H_3)] \mid Q_4)) \longrightarrow H'$ ，且 $R \equiv H'(P)$ ，结论的第二种情况成立。
    (2). $H_2 \equiv - \mid R'$ 、 $P \xrightarrow{\mathtt{in}\, n} P'$ 且 $Q_3' \equiv P' \mid R'$ ：此时有 $H = (\nu\vec{q})(n[Q_1] \mid (\nu\vec{r})(m[- \mid R'] \mid Q_4)) \equiv (\nu\vec{q},\vec{r})(n[Q_1] \mid m[- \mid R'] \mid Q_4)$ ，另外我们有 $P \xrightarrow{\mathtt{in}\, n} P'$ 、 $Q_1 \xrightarrow{\overline{\mathtt{in}}\, m} Q_1'$ 且 $R \equiv (\nu\vec{q},\vec{r})(n[m[P' \mid R'] \mid Q_1'] \mid Q_4)$ ，由 **(Inter In)** 知 $H \bullet P \rightsquigarrow R$ ，第三种情况成立。
  - $H_1 \equiv (\nu\vec{r})(m[Q_3] \mid H_2)$ 且 $Q_4 = H_2(P)$ ：此时令 $H' = (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid Q_1'] \mid H_2)$ ，由 $Q_1 \xrightarrow{\overline{\mathtt{in}}\, m}$ 、$Q_3 \xrightarrow{\mathtt{in}\, n} Q_3'$ 知 $H = (\nu\vec{q})(n[Q_1] \mid (\nu\vec{r})(m[Q_3] \mid H_3)) \longrightarrow H'$ ，且 $R \equiv H'(P)$ ，结论的第二种情况成立。
  - $P > (\nu\vec{r})\langle m[Q_3]\rangle P_1$ 、 $H \equiv - \mid R'$ 、 $Q_4 \equiv P_1 \mid R'$ 且 $\{\vec{r}\} \cap fn(R') = \emptyset$ ：此时归纳所有条件我们有 $P > (\nu\vec{r})\langle m[Q_3]\rangle P_1$ 、 $Q_3 \xrightarrow{\mathtt{in}\, n} Q_3'$ 、 $H \equiv (\nu\vec{q})(n[Q_1] \mid - \mid R')$ 、$Q_1 \xrightarrow{\overline{\mathtt{in}}\, m} Q_1'$ 、$\{\vec{r}\} \cap fn(n[Q_1]) = \emptyset$(不妨假设,由于 $\vec{r}$ 为约束名) 且 $R \equiv (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid Q_1'] \mid P_1 \mid R')$ ，由 **(Inter Amb In)** 知 $H \bullet P \rightsquigarrow R$ ，第三种情况成立。
- $P > (\nu\vec{q})\langle n[Q_1]\rangle P_1$ 、 $H \equiv - \mid R'$ 、 $Q_2 \equiv P_1 \mid R'$ 且 $\{\vec{q}\} \cap fn(R') = \emptyset$ ：此时首先利用引理 4.4 知 $P \equiv (\nu\vec{q})(n[Q_1] \mid P_1)$ ，其次由 $P_1 \mid R' \equiv Q_2$ 且 $Q_2 > (\nu\vec{r})\langle m[Q_3]\rangle Q_4$ 利用引理 4.14 可知，存在 Q_3'' 、 Q_4'' 满足 $Q_3 \equiv Q_3''$ 、 $Q_4 \equiv Q_4''$ 且 $P_1 \mid R' > (\nu\vec{r})\langle m[Q_3'']\rangle Q_4''$ 。由硬

化关系的定义易知，$P_1 \mid R' > (\nu\vec{r})\langle m[Q_3'']\rangle Q_4''$ 必须通过使用以下规则之一得到：

(Harden Par 1) 此时有 $P_1 > (\nu\vec{r})\langle m[Q_3'']\rangle Q_5$、$Q_4'' = Q_5 \mid R'$ 且 $\{\vec{r}\} \cap fn(R') = \emptyset$。此时综合已知条件利用 **(Trans In)** 可得 $P \xrightarrow{\tau} \equiv (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid Q_1'] \mid Q_5)$，令上式右部为 P' 可知 $H(P') = (\nu\vec{q},\vec{r})(n[m[Q_3'] \mid Q_1'] \mid Q_5) \mid R' \equiv R$，第一种情况成立。

(Harden Par 2) 此时有 $R' > (\nu\vec{r})\langle m[Q_3'']\rangle Q_5$、$Q_4'' = P_1 \mid Q_5$ 且 $\{\vec{r}\} \cap fn(P_1) = \emptyset$。此时由 $\{\vec{q}\} \cap fn(R')$ 可知 $\{\vec{r}\} \cap fn(m[Q_3'']) = \emptyset$，综合已知条件有 $P > (\nu\vec{q})\langle n[Q_1]\rangle P_1$、$Q_1 \xrightarrow{\overline{\texttt{in}}\, m} Q_1'$、$H \equiv (\vec{r})(Q_5 \mid - \mid m[Q_3''])$、$Q_3'' \xrightarrow{\texttt{in}\, n} Q_3'''$、$Q_3''' \equiv Q_3'$ 且 $R \equiv (\nu\vec{q},\vec{r})(n[m[Q_3'''] \mid Q_1'] \mid P_1 \mid Q_5)$，由 **(Inter Amb Co-in)** 知 $H \bullet P \rightsquigarrow R$，第三种情况成立。

(Trans Open) 此时有 $H(P) > (\nu\vec{q})\langle n[Q_1]\rangle Q_2$、$Q_1 \xrightarrow{\overline{\texttt{open}}} Q_1'$、$Q_2 \xrightarrow{\texttt{open}\, n} Q_2'$ 且 $R = (\nu\vec{q})(Q_1' \mid Q_2')$。

根据 $H(P) > (\nu\vec{q})\langle n[Q_1]\rangle Q_2$ 由引理 5.16 知，必有以下情况之一成立：

- $H \equiv (\nu\vec{q})(n[H_1] \mid Q_2)$ 且 $Q_1 = H_1(P)$：此时有 $H_1(P) \xrightarrow{\overline{\texttt{open}}} Q_1'$，根据引理 5.17 可知，必有以下情况之一成立：
  - $H_1 \equiv (\nu\vec{r})(\overline{\texttt{open}}.R' \mid H_2)$、$Q_1' \equiv (\nu\vec{r})(R' \mid H_2(P))$ 且 $\{\vec{r}\} \cap fn(P) = \emptyset$：此时，由于 $\vec{r}$ 为约束名集合，不妨设 $\{\vec{r}\} \cap fn(Q_2) = \emptyset$。令 $H' = (\nu\vec{q})(\nu\vec{r})(R' \mid H_2 \mid Q_2')$，由 $Q_2 \xrightarrow{\texttt{open}\, n} Q_2'$ 知，$H = (\nu\vec{q})(n[(\nu\vec{r})(\overline{\texttt{open}}.R' \mid H_2)] \mid Q_2) \longrightarrow H'$，且 $R \equiv H'(P)$，结论的第二种情况成立。
  - $H_1 \equiv - \mid R'$、$P \xrightarrow{\overline{\texttt{open}}} P'$ 且 $Q_1' \equiv P' \mid R'$：此时有 $H = (\nu\vec{q})(n[- \mid R'] \mid Q_2)$、$P \xrightarrow{\overline{\texttt{open}}} P'$、$Q_2 \xrightarrow{\texttt{open}\, n} Q_2'$ 且 $R \equiv (\nu\vec{q})(P' \mid R' \mid Q_2')$，由 **(Inter Co-open)** 知 $H \bullet P \rightsquigarrow R$，第三种情况成立。
- $H \equiv (\nu\vec{q})(n[Q_1] \mid H_1)$ 且 $Q_2 = H_1(P)$：此时有 $H_1(P) \xrightarrow{\texttt{open}\, n} Q_2'$，根据引理 5.17 可知，必有以下情况之一成立：
  - $H_1 \equiv (\nu\vec{r})(\texttt{open}\, n\, .R' \mid H_2)$、$Q_2' \equiv (\nu\vec{r})(R' \mid H_2(P))$ 且 $\{\vec{r}\} \cap (\{n\} \cup fn(P)) = \emptyset$：此时，由于 $\vec{r}$ 为约束名集合，不妨设 $\{\vec{r}\} \cap fn(Q_1) = \emptyset$。令 $H' = (\nu\vec{q})(\nu\vec{r})(Q_1' \mid R' \mid H_2)$，由 $Q_1 \xrightarrow{\overline{\texttt{open}}} Q_1'$ 知，$H = (\nu\vec{q})(n[Q_1] \mid (\nu\vec{r})(\texttt{open}\, n\, .R' \mid H_2)) \longrightarrow H'$，且 $R \equiv H'(P)$，结论的第二种情况成立。
  - $H_1 \equiv - \mid R'$、$P \xrightarrow{\overline{\texttt{open}}} P'$ 且 $Q_2' \equiv P' \mid R'$：此时有 $H = (\nu\vec{q})(n[Q_1] \mid - \mid R')$、$P \xrightarrow{\overline{\texttt{open}}} P'$、$Q_1 \xrightarrow{\overline{\texttt{open}}} Q_1'$ 且 $R \equiv (\nu\vec{q})(Q_1' \mid P' \mid R')$，由 **(Inter Open)** 知 $H \bullet P \rightsquigarrow R$，第三种情况成立。
- $P > (\nu\vec{q})\langle n[Q_1]\rangle P_1$、$H \equiv - \mid R'$、$Q_2 \equiv P_1 \mid R'$ 且 $\{\vec{q}\} \cap fn(R') = \emptyset$：此时首先利用引理 4.4 知 $P \equiv (\nu\vec{q})(n[Q_1] \mid P_1)$，其次由 $Q_2 \xrightarrow{\texttt{open}\, n} Q_2'$ 利用引理 4.15 知存在 Q_2'' 满足 $P_1 \mid R' \xrightarrow{\texttt{open}\, n} Q_2''$ 且 $Q_2'' \equiv Q_2'$。由引理 5.18 可知，必有以下情况之一成立：
  - $P_1 \xrightarrow{\texttt{open}\, n} P_1'$ 且 $Q_2'' \equiv P_1' \mid R'$：此时有 $P \longrightarrow (\nu\vec{q})(Q_1' \mid P_1')$，令 $P' = (\nu\vec{q})(Q_1' \mid P_1')$ 可知，$R \equiv (\nu\vec{q})(Q_1' \mid Q_2') \equiv (\nu\vec{q})(Q_1' \mid P_1' \mid R') \equiv (\nu\vec{q})(Q_1' \mid P_1') \mid R' \equiv H(P')$，因此结论的第一种情况成立。
  - $R' \xrightarrow{\texttt{open}\, n} R''$ 且 $Q_2'' \equiv P_1 \mid R''$：此时归纳所有条件我们有 $P > (\nu\vec{q})\langle n[Q_1]\rangle P_1$、$Q_1 \xrightarrow{\overline{\texttt{open}}} Q_1'$、$H \equiv - \mid R'$、$R' \xrightarrow{\texttt{open}\, n} R''$、$\{\vec{q}\} \cap fn(R') = \emptyset$ 且 $R \equiv (\nu\vec{q})(Q_1' \mid Q_2') \equiv (\nu\vec{q})(Q_1' \mid P_1 \mid R'')$，由 **(Inter Amb Co-open)** 知 $H \bullet P \rightsquigarrow R$，第三种情况成立。

(Trans Out)(Trans Amb) 采用类似的分析同理可证，此处略。

通过以上对所有可能情况的分析可知，结论成立。 ∎

利用以上结果，我们可以得到本章的第一个结论（ activity lemma[1] ）。

定理 5.20 (Activity) $H(P) \longrightarrow R$ 当且仅当：

(Act Proc) $P \longrightarrow P'$ 且 $R \equiv H(P')$ ，或者

(Act Har) $H \longrightarrow H'$ 且 $R \equiv H'(P)$ ，或者

(Act Inter) $H \bullet P \rightsquigarrow R$ 。

证明 从右到左的证明通过逐一检验完成。从左到右的证明通过引理 5.19 和定理 4.17 的结论可以直接得到。 ∎

§5.3 两种等价关系间的联系

从定理 5.20 可以看出，简单上下文等价的判定更加简单可行。本节将给出本章的第二个结论，即简单上下文等价关系和上下文等价关系是同一个关系（定理 5.39 ）。这样进程的上下文等价关系问题就可以转化为进程的简单上下文等价关系，进而通过简单上下文等价的判定方法加以解决。

首先，我们通过一系列的命题（命题 5.21 、 5.22 、 5.23 、 5.25 、 5.30 和 5.35 ）组成对简单上下文等价关系同余性（命题 5.36 ）的证明。之后，容易得出两种等价关系实际上为同一种关系（定理 5.39 ）。

命题 5.21 关系 $P \simeq Q$ 是一等价关系，即；它是自反、对称和传递的。另外，如果 $P \equiv Q$ ，那么 $P \simeq Q$ 。

证明 由简单上下文等价的定义易知它是一等价关系。对第二个结论，假定 $P \equiv Q$ ，则考虑任意 H 和 n ，由 $\equiv$ 的同余性质可知 $H(P) \equiv H(Q)$ ，再利用引理 5.6 可得 $H(P) \Downarrow_n \iff H(Q) \Downarrow_n$ ，因此 $P \simeq Q$ 。 ∎

命题 5.22 如果 $P \simeq P'$ ，那么 $P \mid Q \simeq P' \mid Q$ 。

证明 对任意 H 和 n ，构造 $H' = H(- \mid Q)$ 。由 $P \simeq P'$ 可知对 n 有 $H'(P) \Downarrow_n \iff H'(P') \Downarrow_n$ ，即 $H(P \mid Q) \Downarrow_n \iff H(P' \mid Q) \Downarrow_n$ ，因此 $P \mid Q \simeq P' \mid Q$ 。 ∎

命题 5.23 如果 $P \simeq P'$ ，那么 $n[P] \simeq n[P']$ 。

证明 对任意 H 和 m ，构造 $H' = H(n[-])$ 。由 $P \simeq P'$ 可知对 m 有 $H'(P) \Downarrow_m \iff H'(P') \Downarrow_m$ ，即 $H(n[P]) \Downarrow_m \iff H(n[P']) \Downarrow_n$ ，因此 $n[P] \simeq n[P']$ 。 ∎

引理 5.24 如果 $m \neq n$ ，那么 $(\nu n)P \Downarrow_m \iff P \Downarrow_m$ 。

证明

“ $\Longrightarrow$ ”：对 $(\nu n)P \Downarrow_m$ 的推导过程进行归纳易证 $P \Downarrow_m$ ，此处略。

[1] 虽然这里 activity lemma 是作为定理出现的，但由于历史原因，类似的定理被称为 activity "lemma"[20] 。因此，本文沿袭了这一用法。

“$\Longleftarrow$”：对 $P \Downarrow_m$ 的推导过程进行归纳易证 $(\nu n)P \Downarrow_m$，此处略。

∎

命题 5.25 如果 $P \simeq P'$，那么 $(\nu n)P \simeq (\nu n)P'$。

证明 对任意 H 和 m，由 $P \simeq P'$ 可知有 $H(P) \Downarrow_m \iff H(P') \Downarrow_m$，由于在 $(\nu n)P$ 及 $(\nu n)P'$ 中 n 为约束名，因此不妨设 $n \neq m$ 且 $n \notin fn(H)$，由引理 5.24 可得 $(\nu n)H(P) \Downarrow_m \iff (\nu n)H(P') \Downarrow_m$，即 $H((\nu n)P) \Downarrow_n \iff H((\nu n)P') \Downarrow_n$，因此 $(\nu n)P \simeq (\nu n)P'$。 ∎

简单上下文等价对顺序动作的同余性，即 $P \simeq P' \Longrightarrow M \,.\, P \simeq M \,.\, P'$（命题 5.30）的证明需要建立在下面的引理 5.26 到引理 5.29 的基础之上。

引理 5.26 $P \downarrow_n$ 当且仅当存在 $\vec{p}$、A、P_1、P_1' 和 P_2 满足 $P > (\nu\vec{p})\langle n[P_1]\rangle P_2$、$P_1 \xrightarrow{A} P_1'$、$A \in \mathbf{BarbedAction}$ 且 $(fn(A) \cup \{n\}) \cap \{\vec{p}\} = \emptyset$。

证明

“$\Longrightarrow$”：此时 $P \downarrow_n$，由定义可知，存在 $\vec{r}$、A、R_1、R_2 和 R_3 满足 $A \in \mathbf{BarbedAction}$、$(fn(A) \cup \{n\}) \cap \{\vec{r}\} = \emptyset$ 且 $P \equiv (\nu\vec{r})(n[A \,.\, R_1 \mid R_2] \mid R_3)$。不仿设 $\{\vec{r}\} \subseteq (fn(R_1) \cup fn(R_2) \cup fn(R_3))$，令 $\vec{p} = \vec{r}$（否则，可令 $\vec{p} = \{\vec{r}\} \cap (fn(R_1) \cup fn(R_2) \cup fn(R_3))$），利用 **(Harden Amb)**、**(Harden Par 1)** 和 **(Harden Res)** 可知 $(\nu\vec{r})(n[A \,.\, R_1 \mid R_2] \mid R_3) > (\nu\vec{p})\langle n[A \,.\, R_1 \mid R_2]\rangle R_3$。再由引理 4.14 可得，存在 Q_1、P_2 满足 $P > (\nu\vec{p})\langle Q_1\rangle P_2$ 且 $Q_1 \equiv n[A \,.\, R_1 \mid R_2]$、$P_2 \equiv R_3$。显然，$Q_1 = n[P_1]$ 且 $P_1 \equiv A \,.\, R_1 \mid R_2$。再由 $A \,.\, R_1 \mid R_2 \xrightarrow{A} R_1 \mid R_2$ 和 $P_1 \equiv A \,.\, R_1 \mid R_2$ 利用引理 4.15 可知存在 P_1' 满足 $P_1 \xrightarrow{A} P_1'$ 且 $P_1' \equiv R_1 \mid R_2$。综合上述结论可知结论成立。

“$\Longleftarrow$”：由 $P > (\nu\vec{p})\langle n[P_1]\rangle P_2$ 利用引理 4.4 可知 $P \equiv (\nu\vec{p})(n[P_1] \mid P_2)$。由 $P_1 \xrightarrow{A} P_1'$ 利用引理 4.5 可知 $P_1 \equiv (\nu\vec{q})(A \,.\, R' \mid R'')$ 且 $fn(A) \cap \{\vec{q}\} = \emptyset$。不妨设 $\{\vec{q}\} \cap (\{n\} \cup fn(P_2)) = \emptyset$，于是有 $P \equiv (\nu\vec{p},\vec{q})(n[A \,.\, R' \mid R''] \mid P_2)$。由条件 $(fn(A) \cup \{n\}) \cap \{\vec{p}\} = \emptyset$，易知 $(fn(A) \cup \{n\}) \cap \{\vec{p},\vec{q}\} = \emptyset$，再由条件 $A \in \mathbf{BarbedAction}$ 知，$P \downarrow_n$。

∎

引理 5.27 如果 $M \,.\, P \xrightarrow{A} P'$，那么以下之一成立：

(1). $M > A \,.\, N$ 且 $P' \equiv N \,.\, P$；
(2). $M = \epsilon$ 且 $P \xrightarrow{A} P'$。

证明 由引理 4.11 和 **(Trans Cap)** 易证，此处略。 ∎

引理 5.28 如果 $H(P) \downarrow_n$ 则以下之一成立：

(1). 对任意 Q，$H(Q) \downarrow_n$；
(2). $P \downarrow_n$ 且对任意 $Q \downarrow_n$ 有 $H(Q) \downarrow_n$；
(3). $P \xrightarrow{A} P'$、$A \in \mathbf{BarbedAction}$ 且对任意 $Q \xrightarrow{A'} Q'$ 且 $A' \in \mathbf{BarbedAction}$ 有 $H(Q) \downarrow_n$。

证明 由引理 5.26，$H(P)\downarrow_n$ 意味着存在 $\vec{p}$、A、P_1、P_1' 和 P_2 满足 $H(P) > (\nu\vec{p})\langle n[P_1]\rangle P_2$、$P_1 \xrightarrow{A} P_1'$、$A \in \mathbf{BarbedAction}$ 且 $(fn(A)\cup\{n\})\cap\{\vec{p}\} = \emptyset$。利用引理 5.16 可知，必有以下之一成立：

(1). $H > (\nu\vec{p})\langle n[H']\rangle P_2$ 且 $P_1 = H'(P)$。此时有 $H'(P) \xrightarrow{A} P_1'$，利用引理 5.17 可知，必有以下之一成立：

(a) $H' \equiv (\nu\vec{r})(A\,.\,R \mid H'')$、$P_1' \equiv (\nu\vec{r})(R \mid H''(P))$ 且 $\{\vec{r}\}\cap fn(A) = \emptyset$，易知第一种情况成立。

(b) $H' \equiv - \mid R$、$P \xrightarrow{A} P'$ 且 $P_1' \equiv P' \mid R$，易知第三种情况成立。

(2). $H > (\nu\vec{p})\langle n[P_1]\rangle H'$ 且 $P_2 = H'(P)$，此时易知第一种情况成立。

(3). $P > (\nu\vec{p})\langle n[P_1]\rangle P'$、$H \equiv - \mid R$、$P_2 \equiv P' \mid R$ 且 $\{\vec{p}\}\cap fn(R) = \emptyset$，此时易知第二种情况成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 5.29 *如果 $P \simeq P'$ 且 $H(M\,.\,P)\Downarrow_n$，那么 $H(M\,.\,P')\Downarrow_n$。*

证明 对 $H(M\,.\,P)\Downarrow_n$ 的推导过程进行归纳。

(Conv Exb) 此时 $H(M\,.\,P)\downarrow_n$，由引理 5.28 可知，以下之一成立：

(1). 对任意 Q 都有 $H(Q)\downarrow_n$：此时立刻有结论 $H(M\,.\,P')\Downarrow_n$。

(2). $M\,.\,P\downarrow_n$ 且对任意 Q $H(Q)\downarrow_n$ 有。由 $M\,.\,P\downarrow_n$ 利用引理 5.26 可知 $M\,.\,P > (\nu\vec{r})\langle n[R']\rangle R''$、$R' \xrightarrow{A} R'''$、$A \in \mathbf{BarbedAction}$ 且 $(fn(A)\cup\{n\})\cap\{\vec{r}\} = \emptyset$。再利用引理 4.11 可得 $M = \epsilon$。因此由 $P \simeq P'$ 和 $H(\epsilon\,.\,P)\downarrow_n$ 易知 $H(\epsilon\,.\,P')\Downarrow_n$。

(3). $M\,.\,P \xrightarrow{A} P'$、$A \in \mathbf{BarbedAction}$ 且对任意 Q $\xrightarrow{A'} Q'$ 且 $A' \in \mathbf{BarbedAction}$ 有 $H(Q)\downarrow_n$。利用引理 5.27 类似可以证明 $H(M\,.\,P')\Downarrow_n$。

(Conv Red) 此时 $H(M\,.\,P) \longrightarrow R$ 且 $R\Downarrow_n$。由定理 5.20 可知，以下之一成立：

(Act Proc) 此时有 $M\,.\,P \longrightarrow R'$ 且 $R \equiv H(R')$。易知 $M = \epsilon$ 且 $P \longrightarrow R'$。因此 $H(M\,.\,P) \equiv H(P)$，所以 $H(P)\Downarrow_n$，利用 $P \simeq P'$ 可知 $H(P')\Downarrow_n$，然而 $H(M\,.\,P') \equiv H(P')$，因此 $H(M\,.\,P')\Downarrow_n$。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(M\,.\,P)$。由 $R\Downarrow_n$ 利用引理 5.6 可知 $H'(M\,.\,P)\Downarrow_n$ 且其推导过程与 $R\Downarrow_n$ 的推导过程相同，使用归纳假设可知 $H'(M\,.\,P')\Downarrow_n$。而 $H(M\,.\,P') \longrightarrow H'(M\,.\,P')$，因此 $H(M\,.\,P')\Downarrow_n$。

(Act Inter) 此时存在 H'、$\vec{r}$ 满足 $\{\vec{r}\}\cap fn(M\,.\,P) = \emptyset$。且有多种情况成立。这里，我们以 **(Inter In)** 一种情况为例给出证明，其它情况的证明方法类似。

(Inter In) 此时有 $H \equiv (\nu\vec{r})H'(m[- \mid R_1] \mid n[R_2])$、$M\,.\,P \xrightarrow{\mathtt{in}\ n} P''$、$R_2 \xrightarrow{\overline{\mathtt{in}}\ m} R_2'$ 且 $R \equiv (\nu\vec{r})H'(n[m[P'' \mid R_1] \mid R_2'])$。由引理 5.27 可知以下之一成立：

(1). $M > \mathtt{in}\ n\,.\,N'$ 且 $P'' \equiv N'\,.\,P$。此时 $M\,.\,P' \xrightarrow{\mathtt{in}\ n} N'\,.\,P'$，因此 $H(M\,.\,P') \xrightarrow{\tau} (\nu\vec{r})H'(n[m[N'\,.\,P' \mid R_2'] \mid R_1])$。有 $R\Downarrow_n$ 和引理 5.6 可知 $(\nu\vec{r})H'(n[m[N'\,.\,P \mid R_1] \mid R_2'])\Downarrow_n$ 且其推导过程与 $R\Downarrow_n$ 的推导过程相同，使用归纳假设可知 $(\nu\vec{r})H'(n[m[N'\,.\,P' \mid R_1] \mid R_2'])\Downarrow_n$，即有 $H(M\,.\,P')\Downarrow_n$。

(2). $M = \epsilon$ 且 $P \xrightarrow{\mathtt{in}\ n} P''$，此时有 $H(M\,.\,P) \equiv H(P)$、$H(M\,.\,P') \equiv H(P')$，再由 $P \simeq P'$ 可得 $H(M\,.\,P')\Downarrow_n$。

通过以上对所有可能情况的分析可知，结论成立。 ∎

命题 5.30 如果 $P \simeq P'$，那么 $M.P \simeq M.P'$。

证明 对任意 H 和 m，由 $P \simeq P'$ 利用引理 5.29 可知 $H(M.P)\Downarrow_n \iff H(M.P')\Downarrow_n$。因此 $M.P \simeq M.P'$。 ∎

下面是对简单上下文等价对复制构造同余性的证明（命题 5.35），首先需要证明一些技术引理。以下使用 P^k 代表 k 个拷贝的 P 并发运行。递归定义如下：$P^0 \triangleq \mathbf{0}$, $P^{k+1} \triangleq P \mid P^k$。

引理 5.31 如果 $H(!P)\Downarrow_n$，那么存在整数 k 使得 $H(P^k)\Downarrow_n$。

证明 对本引理的证明参见附录 3。 ∎

引理 5.32 如果 $H(\mathbf{0})\Downarrow_n$，那么 $H(P)\Downarrow_n$。

证明 对 $H(\mathbf{0})\Downarrow_n$ 的推导过程进行归纳。

(Conv Exb) 此时有 $H(\mathbf{0})\downarrow_n$，由引理 5.28 可知三种情况只有第一种成立，即：对任意 Q 均有 $H(Q)\downarrow_n$，此时直接有 $H(P)\Downarrow_n$。

(Conv Red) 此时有 $H(\mathbf{0}) \longrightarrow Q$ 且 $Q\Downarrow_n$。由定理 5.20 和 **0** 无法归约的事实可知，必有 $H \longrightarrow H'$ 且 $Q \equiv H'(\mathbf{0})$。利用引理 5.6 可知 $H'(\mathbf{0})\Downarrow_n$ 且其推导过程与 $Q\Downarrow_n$ 的推导过程相同，使用归纳假设可知 $H'(P)\Downarrow_n$，再由 $H \longrightarrow H'$ 可知 $H(P) \longrightarrow H'(P)$，所以 $H(P)\Downarrow_n$。

∎

引理 5.33 如果 $P \simeq P'$ 且 $Q \simeq Q'$，那么 $P \mid P' \simeq Q \mid Q'$。

证明 我们证明对任意 H 和 n，如果 $H(P \mid Q)\Downarrow_n$，那么 $H(P' \mid Q')\Downarrow_n$。通过完全对称的方法可证明结论的另一半。

取 $H' = H(- \mid Q)$，$H(P \mid Q)\Downarrow_n$ 即 $H'(P)\Downarrow_n$。由 $P \simeq P'$ 知有 $H'(P')\Downarrow_n$，即 $H(P' \mid Q)\Downarrow_n$，再取 $H'' = H(P' \mid -)$ 同理可得 $H(P' \mid Q')\Downarrow_n$。 ∎

引理 5.34 如果 $H(P)\Downarrow_n$，那么 $H(P \mid Q)\Downarrow_n$。

证明 令 $H' = H(P \mid -)$，则 $H'(\mathbf{0}) = H(P)\Downarrow_n$，由引理 5.32 可知 $H'(Q)\Downarrow_n$，即 $H(P \mid Q)\Downarrow_n$。 ∎

命题 5.35 如果 $P \simeq P'$，那么 $!P \simeq !P'$。

证明 我们证明对任意 H 和 n，如果 $H(!P)\Downarrow_n$，那么 $H(!P')\Downarrow_n$。通过完全对称的方法可证明结论的另一半。

由 $H(!P)\Downarrow_n$ 利用引理 5.31 可知存在 k 满足 $H(P^k)\Downarrow_n$，而由 $P \simeq P'$ 利用引理 5.33 可得 $P^k \simeq P'^k$，因此有 $H(P'^k)\Downarrow_n$，再由 $!P' \equiv P'^k \mid !P'$ 利用引理 5.34 可得 $H(!P')\Downarrow_n$。 ∎

综合上述命题的结论，可知简单上下文等价是一同余关系。

命题 5.36 如果 $P \simeq P'$，那么 $\mathcal{C}(P) \simeq \mathcal{C}(P')$。

证明 综合命题 5.22 、 5.23 、 5.25 、 5.30 和 5.35 的结论。 ∎

以此为基础，容易证明下面的两个命题。

命题 5.37 *如果 $P \simeq P'$ ，那么 $P \approx P'$ 。*

证明 对任意上下文 $\mathcal{C}$ 和 n ，由 $P \simeq P'$ 利用命题 5.36 可知 $\mathcal{C}(P) \simeq \mathcal{C}(P')$ 因此对任意 n 有 $\mathcal{C}(P) \Downarrow_n \iff \mathcal{C}(P') \Downarrow_n$ ，结论成立。 ∎

命题 5.38 *如果 $P \approx P'$ ，那么 $P \simeq P'$ 。*

证明 由 $\approx$ 和 $\simeq$ 的定义直接可得。 ∎

以下是本章的第二个结论，即上下文等价关系和简单上下文等价关系是同一关系（ context lemma ）。

定理 5.39 (Context) *$P \approx P'$ 当且仅当 $P \simeq P'$ 。*

证明 综合命题 5.38 和 5.37 的结论。 ∎

本章小结

本章和上一章的内容基本上是对文献 [20] 中关于 MA 进程等价性研究的结论在 ROAM 中的验证。首先通过定义进程的硬化关系建立起一套基于硬化关系的标号转移语义，并证明该标号转移语义与归约语义完全等价。之后，通过简单上下文的定义，建立起一套进程等价性判定方法，最后证明简单上下文等价关系与更一般的上下文等价关系是同一关系。

但是，本文对 ROAM 进程等价性的研究与文献 [20] 中的工作又有不同。首先， ROAM 中的协动作和参数限制给标号转移语义的定义带来了更多的复杂性。文献 [16] 使用了基于提交和结果的方法研究了具有协动作的 SA 演算的标号转移语义。本文基于硬化关系对具有协动作的灰箱演算标号转移语义研究还是首次。其次，由于协动作的存在， ROAM 中进程和简单上下文的交互种类更加繁多，在不引入本地通讯原语的纯 ROAM 演算中，就有 11 种之多。最后，文献 [20] 中关于进程等价性的研究工作止于 activitiy lemma 和 context lemma 的证明，而本文后面将利用这些结论并结合对演化类型系统的研究，给出更加实用的等价性判定定律。较之文献 [20] ，本文的工作更加深入。

第六章　等价性定律

上一章给出了判断进程等价的一般性方法。然而在实际使用过程中，我们希望能够提供类似归约规则那样容易使用的一些具体方法。本章的工作结合了第三章的类型系统研究结论和第四、五章进程等价性判定的一般结论，给出了一些 ROAM 中具有一定通用性的等价性定律。它们有一个共同的特点，即等价的进程 P 和 Q 都具有 $P \longrightarrow Q$ 的关系。但是这些定律中的条件限制保证了在任何满足条件的简单上下文 H 中，P 和 Q 都是可以互换的，即：$H(P)$ 中的其它可归约动作和 $P \longrightarrow Q$ 中的可归约动作之间是协调的（confluent）——无论先归约哪一个，都能得到相同的结果。本章最后给出了利用这些等价性定律的一些例子，包括对上文换名构造正确性的证明。

本章的等价性定律主要分为三个部分：无上下文条件的单线程定律、有上下文条件的单线程定律和定点接收定律，同时还附带了一些简化的或无类型的等价性定律作为推论。在具体给出这些定律之前，首先进行一些基本概念的定义和引理的证明。

§6.1　具有类型约束的进程等价

本章的等价性定律很大程度建立在类型系统 ETS-MT之上，尤其是其中进程的单线程性和非移动性。此时，等价的进程首先必须具有相同的类型，且任何用于测试的简单上下文必须在填入该类型的进程后是合法的。

定义 6.1　简单上下文 H 称为一个 (Γ/T) 型简单上下文，如果在 H 的空缺位置放入类型为 T 的进程后，其结果的类型是合法的。即：给定一个在 Γ 中类型为 T 的进程 P，存在 Γ_H 满足 $\Gamma_H,\ \Gamma \vdash H(P) : T'$。

定义 6.2　进程 P、Q 称为 (Γ/T) 型简单上下文等价，如果 $\Gamma \vdash P : T$、$\Gamma \vdash Q : T$ 且对任意 (Γ/T) 型简单上下文H 有 $H(P) \Downarrow_n \iff H(Q) \Downarrow_n$。记作：$\Gamma \triangleright P \simeq Q : T$。当类型 T 不重要时，简写为：$\Gamma \triangleright P \simeq Q$。

以下给出一些演化类型系统下特殊类型进程的一些性质（引理 6.3 ~ 引理 6.6）。为方便下文的描述，定义关于进程类型的一些简写如下：

$Current(T)$	属于 $\mathcal{U}$，用于取 T 的当前 (准) 类型。
$Future(T)$	属于 $\mathcal{T}$，用于取 T 经过一次演化后的类型。
$U.Z, U.Y$	取一个准类型的移动性或线程数分量。
$Threads(\Gamma,\ n)$	$Current(Type(\Gamma,\ n)).Y$，如果 $Type(\Gamma,\ n)$ 有解，否则没有意义。
$Threads(\Gamma,\ P)$	$Current(Type(\Gamma,\ P)).Y$，如果 $Type(\Gamma,\ P)$ 有解，否则没有意义。
$Mobility(\Gamma,\ n)$	$Current(Type(\Gamma,\ n)).Z$，如果 $Type(\Gamma,\ n)$ 有解，否则没有意义。
$Mobility(\Gamma,\ P)$	$Current(Type(\Gamma,\ P)).Z$，如果 $Type(\Gamma,\ P)$ 有解，否则没有意义。
$Stable(\Gamma,\ n)$	为 True, 如果 $Type(\Gamma,\ n) = U$ 且 $U.Z = \underline{\vee}$；否则为 False。

其中 $Stable(\Gamma,\ n)$ 用于检查灰箱 n 在类型环境 Γ 下是否为稳定的，即它为非移动，且不能被打开（为简单类型）。稳定的灰箱将在归约过程中始终存在且位置固定（相对于其所在的父灰箱而言）。其涵义相当于文献 [16] 中的表示不能移动也不能被打开的 **非移动类型**。

引理 6.3 $Threads(\Gamma,\ P) = 0$ 当且仅当 $Type(\Gamma,\ P) = \underline{\vee}^0$ 。

证明 从右至左直接由定义可得；从左至右通过对类型算法的简单分析易证。 ∎

分析类型算法可知，如果某进程的类型为 $\underline{\vee}^0$ ，则其只可能是静止进程、灰箱或是灰箱的并置，不能激发任何顶层动作，因此下面的推论成立。

推论 6.4 如果 $Threads(\Gamma,\ P) = 0$ ，则不可能有 $P \xrightarrow{A} P'$ 。

引理 6.5 如果 $\Gamma \vdash n[P] : T$ ，则 $Threads(\Gamma,\ P) \leq Threads(\Gamma,\ n)$ 。

证明 由类型算法的完备性定理易证。 ∎

引理 6.6 如果 $\Gamma \vdash A\,.\,P \mid Q : T$ 且 $Current(T).Y = 1$ ，则 $Threads(\Gamma,\ Q) = 0$ 。

证明 由 $\Gamma \vdash A\,.\,P \mid Q : T$ 利用定理 3.11 可知 $Type(\Gamma,\ A\,.\,P \mid Q) = T' \leq T$ 。再由 **(Type Par)** 可知 $Type(\Gamma,\ A\,.\,P) \mid_t Type(\Gamma,\ Q) = T'$ ，因此 $Threads(\Gamma,\ A\,.\,P) \mid_y Threads(\Gamma,\ Q) \leq 1$ 。而分析类型算法可知 $1 \leq Threads(\Gamma,\ A\,.\,P)$ ，因此必有 $T'.Y = 1$ 、 $Threads(\Gamma,\ A\,.\,P) = 1$ 且 $Threads(\Gamma,\ Q) = 0$ 。 ∎

在证明具体的等价性定律之前，首先证明以下三个技术引理。在此后的章节中，由于类型在大部分情况下都具有重要的作用，因此，四、五两章中被大量省略的约束构造和硬化结果中的类型符号将重新补上。例如： $(\nu\vec{p})(P' \mid P'')$ 将恢复为 $(\nu\vec{p} : \vec{T_p})(P' \mid P'')$ 、 $(\nu\vec{p})\langle P'\rangle P''$ 将恢复为 $(\nu\vec{p} : \vec{T_p})\langle P'\rangle P''$ 。用 $\Gamma,\ \vec{p} : \vec{T_p}$ 表示在类型环境 Γ 中增加 $p_1 : T_{p_1},\ \cdots,\ p_k : T_{p_k}$ 这些元素所得的新的类型环境，这里 $\{\vec{p}\} \cap dom(\Gamma) = \emptyset$ 。

引理 6.7 如果 $\Gamma \vdash n[A\,.\,P_1 \mid P_2] : T$ 、 $Threads(\Gamma,\ n) = 1$ 且 $n[A\,.\,P_1 \mid P_2] \longrightarrow R$ ，则必有以下之一成立：

(1). $P_2 \longrightarrow P_2'$ 且 $R \equiv n[A\,.\,P_1 \mid P_2']$ ；

(2). $A = \overline{\texttt{out}}\ m$ 、 $P_2 > (\nu\vec{p} : \vec{T_p})\langle m[R_1]\rangle R_2$ 、 $R_1 \xrightarrow{\texttt{out}\ n} R_1'$ 、 $n,\ m \notin \{\vec{p}\}$ 且 $R \equiv (\nu\vec{p} : \vec{T_p})(n[P_1 \mid R_2] \mid m[R_1'])$ ；

(3). $A = \texttt{open}\ m$ 、 $P_2 > (\nu\vec{p} : \vec{T_p})\langle m[R_1]\rangle R_2$ 、 $R_1 \xrightarrow{\overline{\texttt{open}}} R_1'$ 、 $m \notin \{\vec{p}\}$ 且 $R \equiv (\nu\vec{p} : \vec{T_p})n[P_1 \mid R_1' \mid R_2]$ 。

证明 令 $H = n[A\,.\,P_1 \mid -]$ ，由 $H(P_2) \longrightarrow R$ 利用定理 5.20 可知必有以下情况之一成立：

(Act Proc) 此时有 $P_2 \longrightarrow P_2'$ 且 $R \equiv H(P_2')$ ，即有 $R \equiv n[A\,.\,P_1 \mid P_2']$ ，第一种情况成立。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P_2)$ ，然而 $H = n[A\,.\,P_1 \mid -]$ 不可能进一步归约，因此，该情况不可能出现。

(Act Inter) 此时，分析十一种情况中对 H 的要求可以发现，只可能有以下三种情况满足这里 H 的结构：

(Inter Amb Out 1) 此时易知有结论的第二种情况成立。

(Inter Amb Out 2) 此时要有 $P_2 > (\nu\vec{p} : \vec{T_p})\langle m[R_1]\rangle R_2$ 、 $R_2 \xrightarrow{\overline{\texttt{out}}\ m} R_2'$ 且 $m \notin \{\vec{p}\}$ ，显然有 $Threads((\Gamma,\ \vec{p} : \vec{T_p}),\ R_2) \geq 1$ ，因此 $Threads(\Gamma,\ P_2) \geq 1$ 。而由 $Threads(\Gamma,\ n) = 1$ 利用引理 6.5 、引理 6.6 和推论 6.4 可知 $Threads(\Gamma,\ P_2) = 0$ ，因此该情况不可能成立。

(Inter Amb Co-open) 此时易知有结论的第三种情况成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 6.8 如果 $\Gamma \vdash n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] : T$ 、$Mobility(\Gamma,\ n) = \underline{\vee}$ 、并且 $n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \longrightarrow R$，则必有以下之一成立：

(1). $P_2 \longrightarrow P_2'$ 且 $R \equiv n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2']$；

(2). $P_2 > (\nu\vec{p} : \vec{T}_p)\langle r[R_1]\rangle R_2$ 、 $R_1 \xrightarrow{\mathtt{out}\, n} R_1'$ 、 $R_2 \xrightarrow{\overline{\mathtt{out}}\, r} R_2'$ 、 $n, r \notin \{\vec{p}\}$ 且 $R \equiv (\nu\vec{p} : \vec{T}_p)(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid R_2'] \mid r[R_1'])$ 。

证明 类似引理 6.7 的证明过程易证。 ∎

引理 6.9 如果 $\Gamma \vdash P \mid n[A\,.\,Q_1 \mid Q_2] : T$ 、 $Threads(\Gamma,\ P) = 0$ 、 $Threads(\Gamma,\ n) = 1$ 且 $P \mid n[A\,.\,Q_1 \mid Q_2] \longrightarrow R$，则必有以下之一成立：

(1). $P \longrightarrow P'$ 且 $R \equiv P' \mid n[A\,.\,Q_1 \mid Q_2]$；

(2). $n[A\,.\,Q_1 \mid Q_2] \longrightarrow R'$ 且 $R = P \mid R'$；

(3). $A = \mathtt{in}\, m$ 、 $P > (\nu\vec{p} : \vec{T}_p)\langle m[R_1]\rangle R_2$ 、 $R_1 \xrightarrow{\overline{\mathtt{in}}\, n} R_1'$ 、 $m,\ n \notin \{\vec{p}\}$ 且 $R \equiv (\nu\vec{p} : \vec{T}_p)(m[R_1' \mid n[Q_1 \mid Q_2]] \mid R_2)$；

(4). $A = \overline{\mathtt{in}}\, m$ 、 $P > (\nu\vec{p} : \vec{T}_p)\langle m[R_1]\rangle R_2$ 、 $R_1 \xrightarrow{\mathtt{in}\, n} R_1'$ 、 $m,\ n \notin \{\vec{p}\}$ 且 $R \equiv (\nu\vec{p} : \vec{T}_p)(n[Q_1 \mid Q_2 \mid m[R_1']] \mid R_2)$ 。

证明 令 $H = - \mid n[A\,.\,Q_1 \mid Q_2]$，由 $H(P) \longrightarrow R$ 利用定理 5.20 可知必有以下情况之一成立：

(Act Proc) 此时有 $P \longrightarrow P'$ 且 $R \equiv H(P')$，即有 $R \equiv P' \mid n[A\,.\,Q_1 \mid Q_2]$，第一种情况成立。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P)$，由 $H = - \mid n[A\,.\,Q_1 \mid Q_2]$ 可知，必有 $H' = - \mid R'$ 且 $n[A\,.\,Q_1 \mid Q_2] \longrightarrow R'$，因此 $R = P \mid R'$，第二种情况成立。

(Act Inter) 此时，分析十一种情况中对 H 的要求可以发现，只可能有以下四种情况满足这里 H 的结构：

(Inter Open) 此时要有 $P \xrightarrow{\mathtt{open}\, n} P'$，这与已知 $Threads(\Gamma,\ P) = 0$ 矛盾，因此该情况不可能成立。

(Inter Amb In) 此时易知有结论的第三种情况成立。

(Inter Amb Co-in) 此时易知有结论的第四种情况成立。

(Inter Amb Co-open) 此时要有 $n[A\,.\,Q_1 \mid Q_2] \xrightarrow{\mathtt{open}\, n} R_1'$，而显然该结果不成立，因此该情况不可能成立。

通过对以上所有可能情况的分析可知，结论成立。 ∎

在等价性定律中，有时对进程和上下文中可以出现的能力需要有特殊的要求。我们用一个特殊的 **深层动作** 集合表示灰箱内可以执行的动作： $\mathbf{DeepAction} \triangleq \{n^{[A]} \mid n \in \mathcal{N} \wedge A \in \mathbf{Action}\}$，其元素用 δ 表示。

在此基础上，下面定义给出了表示进程中顶层灰箱可能执行某动作的 **深层动作观察** 符号 $P \stackrel{\delta}{\Longrightarrow}$ 。

定义 6.10 $P \overset{n^{[A]}}{\Longrightarrow}$ 当且仅当存在 P' 满足 $P \longrightarrow^* P'$ 、 $P' \equiv (\nu\vec{p} : \vec{T_p})(n[A \,.\, P_1 \mid P_2] \mid P_3)$ 且 $(\{n\} \cup fn(A)) \cap \{\vec{p}\} = \emptyset$ 。用 $P \overset{n^{[A]}}{\not\Longrightarrow}$ 表示 $P \overset{n^{[A]}}{\Longrightarrow}$ 不成立。

在下面的两条无上下文条件的单线程定律中，使用深层动作观察条件来表示对子灰箱中可能出现动作的限制。

§6.2 无上下文条件的单线程定律

本节介绍两条对上下文没有任何要求的等价性定律。当然，这些定律是以灰箱的单线程性作为前提条件的。

定理 6.11 (ST Out) $\Gamma \triangleright n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid P_2 \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]] \simeq n[P_1 \mid P_2] \mid m[Q_1 \mid Q_2]$，如果 $Threads(\Gamma,\ n) = 1$ 、$Threads(\Gamma,\ m) = 1$ 、$\Gamma \vdash n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid P_2 \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]] : T$ 且 $P_2 \overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ 。

证明 令 $P = n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid P_2 \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]]$ 、 $Q = n[P_1 \mid P_2] \mid m[Q_1 \mid Q_2]$ ，则原命题等价于：对任意 (Γ/T) 型简单上下文H 和灰箱名 h 有 $H(P) \Downarrow_h \iff H(Q) \Downarrow_h$ 。由 $P \longrightarrow Q$ ，有 $H(Q) \Downarrow_h \Longrightarrow H(P) \Downarrow_h$ ，只需再证 $H(P) \Downarrow_h \Longrightarrow H(Q) \Downarrow_h$ 即可。

对 $H(P) \Downarrow_h$ 的推导过程归纳如下：

(Conv Exb) 此时有 $H(P) \downarrow_h$ ，根据引理 5.28 给出的三种情况可知，在第一种情况下直接有 $H(Q) \downarrow_h$ ，其它两种情况对 P 均不能成立。

(Conv Red) 此时有 $H(P) \longrightarrow R$ 且 $R \Downarrow_h$ 。由定理 5.20 可知，必有以下情况之一成立：

(Act Proc) 此时有 $P \longrightarrow P'$ 且 $R \equiv H(P') \Downarrow_h$ 。由 $Threads(\Gamma,\ n) = 1$ 利用引理 6.7 可知必有以下情况之一成立：

(1). $P_2 \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2] \longrightarrow R'$ 且 $P' \equiv n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid R']$ ：此时再利用引理 6.9 可知必有以下情况之一成立；

(a) $P_2 \longrightarrow P_2'$ 且 $R' \equiv P_2' \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]$ ：此时 $P' \equiv n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid P_2' \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]]$ ，因此 $R \equiv H(n[\overline{\mathtt{out}}\ m \,.\, P_1 \mid P_2' \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2]]) \Downarrow_h$ 。利用归纳假设可得 $H(n[P_1 \mid P_2'] \mid m[Q_1 \mid Q_2]) \Downarrow_h$ ，而由 $P_2 \longrightarrow P_2'$ 可知 $H(Q) \longrightarrow H(n[P_1 \mid P_2'] \mid m[Q_1 \mid Q_2])$ ，因此 $H(Q) \Downarrow_h$ 。

(b) $m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2] \longrightarrow R''$ 且 $R' \equiv P_2 \mid R''$ ：此时由引理 6.7 可知只可能有 $Q_2 \longrightarrow Q_2'$ 且 $R' \equiv P_2 \mid m[\mathtt{out}\ n \,.\, Q_1 \mid Q_2']$ ，再利用 $R \Downarrow_h$ 类似第一种情况可知有 $H(Q) \Downarrow_h$ 。

另外两种情况不可能成立。

(2). 由条件 $P_2 \overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ 可知，只可能有 $P \longrightarrow Q$ 且 $R \equiv H(Q) \Downarrow_h$ 。

(3). 此种情况要求 $A = \mathtt{open}\ m$ ，不可能成立。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P) \Downarrow_h$ ，由归纳假设知 $H'(Q) \Downarrow_h$ ，由 $H(Q) \longrightarrow H'(Q)$ 可得 $H(Q) \Downarrow_h$ 。

(Act Inter) 此时分析各种情况对 P 的要求可知，最多只能有后 5 种情况成立。而已知 $Threads(\Gamma,\ n) = 1$ 保证了这 5 种情况同样不可能成立。

通过以上对所有可能的分析可知，结论成立。 ∎

定理 6.12 (ST Open 1) $\Gamma \triangleright n[\mathtt{open}\, m\,.\,P_1 \mid P_2 \mid m[\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2]] \simeq n[P_1 \mid P_2 \mid Q_1 \mid Q_2]$，如果 $Threads(\Gamma, n) = 1$、$Threads(\Gamma, m) = 1$、$\Gamma \vdash n[\mathtt{open}\, m\,.\,P_1 \mid P_2 \mid m[\overline{\mathtt{open}}\,.\,Q_1 \mid Q_2]] : T$ 且 $P_2 \overset{m^{[\overline{\mathtt{open}}]}}{\not\Longrightarrow}$ 。

证明 证明方法类似定理 6.11，此处略。 ∎

定律 **(ST Out)** 和 **(ST Open 1)** 指出，在一定的条件下，跳出动作和打开动作前后两个进程可以认为没有任何区别，进程内部的其它归约不会对这些动作的归约产生干扰。定律中灰箱的单线程性保证了并置的进程 P_2 和 Q_2 不会干扰归约动作的正常进行。如果除去进程表达式中的 P_2 和 Q_2，则相应的单线程条件，甚至类型条件都不再需要了。至此，第五章例 5.12 的证明就变得非常简单，直接套用上述等价定律即可。

§6.3 有上下文条件的单线程定律

很多情况下，进程的等价性必须建立在一定的上下文条件之下。本节讨论两条有上下文条件的单线程定律。首先定义一些对上下文的约束条件。

定义 6.13

(1). $A \not\sqsubseteq H$(或 P) $\Longleftrightarrow$ $H \not\xrightarrow{A} H'$(或 $P \not\xrightarrow{A} P'$)，即：A 不出现在 H(或 P) 的前缀部分。

(2). $n^{[A]} \not\sqsubseteq H$(或 P) $\Longleftrightarrow$ 对于 H(或 P) 中的任意名字为自由名 n 的灰箱 $n[H']$(或 $n[P']$) 都有 $A \not\sqsubseteq H'$(或 $A \not\sqsubseteq P'$)。

(3). $A, n^{[A]} \not\sqsubseteq H$(或 P) $\Longleftrightarrow$ $H \longrightarrow^* H'$(或 $P \longrightarrow^* P'$) 且 $A, n^{[A]} \not\sqsubseteq H'$(或 P')。

虽然判断任意一个进程或简单上下文是否包含某可见动作是没有可使用的算法的，但是可通过一些规则来得出对某些特殊进程的结论。

用 **VisibleAction** 表示由 $\mathtt{open}\, n$、$n^{[\overline{\mathtt{open}}]}$、$n^{[\mathtt{in}\, m]}$、$n^{[\overline{\mathtt{in}}\, m]}$、$n^{[\mathtt{out}\, m]}$ 和 $n^{[\overline{\mathtt{out}}\, m]}$（其中的 n、m 为任意灰箱名）组成的 **可见动作** 集合。用 μ 代表其的元素，用 L 表示其子集。与 **DeepAction** 不同，**VisibleAction** 用来表示上下文中可能出现的动作，而 **DeepAction** 用来表示子灰箱中可能激发的顶层动作。下面的一些等价性定律需要限制上下文中允许出现的动作，即不允许某个可见动作集合 L 中的任何元素出现在上下文中。

记 $L \not\sqsubseteq H$(或 P)，如果对任意 $\mu \in L$ 都有 $\mu \not\sqsubseteq H$(或 P)。

限定上下文中不能出现 L 中的可见动作后，外部观察也应作相应的约束。

定义 6.14 $P \downarrow_n^L \Longleftrightarrow P \equiv (\nu \vec{p} : \vec{T_p})(n[A\,.\,Q_1 \mid Q_2] \mid Q_3)$、$A \in \mathbf{BarbedAction}$、$(fn(A) \cup \{n\}) \cap \{\vec{p}\} = \emptyset$ 且 $\overline{n^{[A]}} \notin L$。

(Conv Ctx Exb) $\dfrac{}{P \downarrow_n^L \Longrightarrow P \Downarrow_n^L}$ **(Conv Ctx Red)** $\dfrac{P \longrightarrow P' \quad P' \Downarrow_n^L}{P \Downarrow_n^L}$

其中对任意 $\mu \in \mathbf{VisibleAction}$，$\overline{\mu}$ 的定义如下：

$$\begin{array}{lcllcl}
\overline{\texttt{open}\,n} & = & n^{[\overline{\texttt{open}}]} & \overline{n^{[\overline{\texttt{open}}]}} & = & \texttt{open}\,n \\
\overline{n^{[\texttt{in}\,m]}} & = & m^{[\overline{\texttt{in}}\,n]} & \overline{m^{[\overline{\texttt{in}}\,n]}} & = & n^{[\texttt{in}\,m]} \\
\overline{n^{[\texttt{out}\,m]}} & = & m^{[\overline{\texttt{out}}\,n]} & \overline{m^{[\overline{\texttt{out}}\,n]}} & = & n^{[\texttt{out}\,m]}
\end{array}$$

例如，令 $P = m[\,\texttt{in}\,n\,.\,Q]$ 、 $L = \{n^{[\overline{\texttt{in}}\,m]}\}$ ，则 $P \downarrow_m$ 但 $P \not\downarrow_m^L$ 。

在有约束的外部观察基础上，有上下文限制的 (Γ/T) 型简单上下文等价定义如下：

定义 6.15 $\Gamma\{L\} \triangleright P \simeq Q : T \iff$ 对任意 (Γ/T) 型简单上下文H 和 n ，如果 $L \not\sqsubseteq H$ ，则 $H(P) \Downarrow_n^L \iff H(Q) \Downarrow_n^L$ 。

对于有约束的外部观察，同样有以下类似引理 5.28 的结论成立。

引理 6.16 如果 $H(P) \downarrow_n^L$ 则以下之一成立：

(1). 对任意 Q ， $H(Q) \downarrow_n^L$ ；

(2). $P \downarrow_n^L$ 且对任意 $Q \downarrow_n^L$ 有 $H(Q) \downarrow_n^L$ ；

(3). $P \xrightarrow{A} P'$ 、 $A \in \mathbf{BarbedAction}$ 、 $\overline{n^{[A]}} \notin L$ 且对任意 $Q \xrightarrow{A'} Q'$ 、 $A' \in \mathbf{BarbedAction}$ 且 $\overline{n^{[A']}} \notin L$ 有 $H(Q) \downarrow_n^L$ 。

证明 仿照引理 5.28 的证明过程，此处略。 ∎

下面给出两条限定上下文中允许出现动作的单线程等价性定律。定律 **(ST In)** 指出，两个单线程灰箱的移入归约动作在测试上下文中不出现干扰该移入动作的构造时，归约前后进程是等价的，即进程中其它部分发生的计算不会影响该移入动作的顺利归约。定律 **(ST Open 2)** 给出了对打开操作类似的结果。

定理 6.17 (ST In) $\Gamma\{n^{[\overline{\texttt{in}}\,m]},\, m^{[\texttt{in}\,n]}\} \triangleright n[\,\overline{\texttt{in}}\,m\,.\,P_1 \mid P_2] \mid m[\,\texttt{in}\,n\,.\,Q_1 \mid Q_2] \simeq n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]]$ ，如果 $Threads(\Gamma,\, n) = 1$ 、 $Threads(\Gamma,\, m) = 1$ 且 $\Gamma \vdash n[\,\overline{\texttt{in}}\,m\,.\,P_1 \mid P_2] \mid m[\,\texttt{in}\,n\,.\,Q_1 \mid Q_2] : T$ 。

证明 令 $L = \{n^{[\overline{\texttt{in}}\,m]},\, m^{[\texttt{in}\,n]}\}$ 、 $P = n[\,\overline{\texttt{in}}\,m\,.\,P_1 \mid P_2] \mid m[\,\texttt{in}\,n\,.\,Q_1 \mid Q_2]$ 、 $Q = n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]]$ ，则原命题等价于：对任意 (Γ/T) 型简单上下文H 和灰箱名 h ，如果 $L \not\sqsubseteq H$ ，那么 $H(P) \Downarrow_h^L \iff H(Q) \Downarrow_h^L$ 。由 $P \longrightarrow Q$ ，有 $H(Q) \Downarrow_h^L \Longrightarrow H(P) \Downarrow_h^L$ ，只需再证 $H(P) \Downarrow_h^L \Longrightarrow H(Q) \Downarrow_h^L$ 即可。

对 $H(P) \Downarrow_h^L$ 的推导过程归纳如下：

(Conv Ctx Exb) 此时有 $H(P) \downarrow_h^L$ ，根据引理 6.16 的三种情况可知，只有第一种情况可能成立，即有 $H(Q) \downarrow_h^L$ 。对第二种情况，分析 P 的构造可知 h 只可能是 n 或 m ，但此时能观察到 n 或 m 的简单上下文都必须知晓 L 集合中的某元素。因此该情况不可能出现。第三种情况要求 $P \xrightarrow{A} P'$ 更不可能出现。

(Conv Ctx Red) 此时有 $H(P) \longrightarrow R$ 且 $R \Downarrow_h^L$ 。由定理 5.20 可知，必有以下情况之一成立：

(Act Proc) 此时有 $P \longrightarrow P'$ 且 $R \equiv H(P') \Downarrow_h$ 。由 $Threads(\Gamma,\, n[\,\overline{\texttt{in}}\,m\,.\,P_1 \mid P_2]) = 0$ 和 $Threads(\Gamma,\, m) = 1$ 利用引理 6.9 可知必有以下情况之一成立：

(1). $n[\,\overline{\texttt{in}}\,m\,.\,P_1 \mid P_2] \longrightarrow R'$ ：此时再利用引理 6.7 可知只可能有以下一种情况成立；

(a) $P_2 \longrightarrow P_2'$ 且 $R' \equiv n[\overline{\mathtt{in}}\, m \,.\, P_1 \mid P_2']$ ：此时 $P' \equiv n[\overline{\mathtt{in}}\, m \,.\, P_1 \mid P_2'] \mid m[\mathtt{in}\, n \,.\, Q_1 \mid Q_2]$ ，所以 $H(P') \equiv R \Downarrow_h^L$ 。利用归纳假设可得 $H(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]]) \Downarrow_h^L$ ，而由 $P_2 \longrightarrow P_2'$ 可知 $H(Q) \longrightarrow H(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]])$ ，因此 $H(Q) \Downarrow_h^L$ 。

另外两种情况不可能成立。

(2). $m[\mathtt{in}\, n \,.\, Q_1 \mid Q_2] \longrightarrow R'$ ：此时类似第一种情况利用引理 6.7 可知有 $H(Q) \Downarrow_h^L$ 。

(3). 即有 $P \longrightarrow Q$ 且 $R \equiv H(Q) \Downarrow_h$ 。

(4). 此种情况不符合 $A = \mathtt{in}\, n$ 的条件，不可能成立。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P) \Downarrow_h^L$ ，由归纳假设知 $H'(Q) \Downarrow_h^L$ ，由 $H(Q) \longrightarrow H'(Q)$ 可得 $H(Q) \Downarrow_h^L$ 。

(Act Inter) 此时分析各种情况对 P 的要求可知，最多只能有后 5 种情况成立。而已知 $Threads(\Gamma,\, n) = 1$ 保证了这 5 种情况中只有 **(Inter Amb In)** 和 **(Inter Amb Co-in)** 有可能成立，但进一步分析可以发现，这两种情况对 H 的要求均不满足给定 $L \not\sqsupseteq H$ 的条件，同样不可能成立。

通过以上对所有可能的分析可知，结论成立。 ∎

定律 **(ST In)** 中的上下文限制条件用于避免上下文中存在具有相同能力的同名灰箱干扰进入动作的正常归约。上述限制在参与移入动作的两个灰箱都没有任何保护的情况下才是必须的，如果这两个灰箱中有一个的名字不为外界环境所知，或者两个灰箱都处在另外一个灰箱内，而在这个另外的灰箱中能够保证在移动操作发生前，没有其它的灰箱会干扰移入操作的顺利进行，则上述上下文条件限制就可以取消。下面的推论总结了这几种情况。

推论 6.18 (ST Aux In) 给定条件 $Threads(\Gamma,\, n) = Threads(\Gamma,\, m) = 1$ ，下面的等价性关系成立：

(1). $\Gamma \triangleright (\nu n : T_n)(n[\overline{\mathtt{in}}\, m \,.\, P_1 \mid P_2] \mid m[\mathtt{in}\, n \,.\, Q_1 \mid Q_2]) \simeq (\nu n : T_n)(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]])$

(2). $\Gamma \triangleright (\nu m : T_m)(n[\overline{\mathtt{in}}\, m \,.\, P_1 \mid P_2] \mid m[\mathtt{in}\, n \,.\, Q_1 \mid Q_2]) \simeq (\nu m : T_m)(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]])$

(3). $\Gamma \triangleright h[n[\overline{\mathtt{in}}\, m \,.\, P_1 \mid P_2] \mid m[\mathtt{in}\, n \,.\, Q_1 \mid Q_2] \mid A \,.\, R] \simeq h[n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid A \,.\, R]$ ，这里 A 可以为某个 $\overline{\mathtt{out}}$ 或 $\mathtt{open}$ 动作。

定理 6.19 (ST Open 2) $\Gamma\{\mathtt{open}\, n,\, n^{[\overline{\mathtt{open}}]}\} \triangleright \mathtt{open}\, n \,.\, P_1 \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] \simeq P_1 \mid Q_1 \mid Q_2$ ，如果 $Threads(\Gamma,\, n) = 1$ 且 $\Gamma \vdash \mathtt{open}\, n \,.\, P_1 \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2] : T$ 。

证明 令 $L = \{\mathtt{open}\, n,\, n^{[\overline{\mathtt{open}}]}\}$ 、$P = \mathtt{open}\, n \,.\, P_1 \mid n[\overline{\mathtt{open}} \,.\, Q_1 \mid Q_2]$ 、$Q = P_1 \mid Q_1 \mid Q_2$ ，则原命题等价于：对任意 (Γ/T) 型简单上下文 H 和灰箱名 h ，如果 $L \not\sqsupseteq H$ ，那么 $H(P) \Downarrow_h^L \iff H(Q) \Downarrow_h^L$ 。由 $P \longrightarrow Q$ ，有 $H(Q) \Downarrow_h^L \Longrightarrow H(P) \Downarrow_h^L$ ，只需再证 $H(P) \Downarrow_h^L \Longrightarrow H(Q) \Downarrow_h^L$ 即可。

对 $H(P) \Downarrow_h^L$ 的推导过程归纳如下：

(Conv Ctx Exb) 此时有 $H(P) \downarrow_h^L$ ，根据引理 6.16 的三种情况可知，只有第一种情况可能成立，即有 $H(Q) \downarrow_h^L$ 。对第二种情况，分析 P 的构造可知 h 只可能是 n ，但此时能观察到 n 的简单上下文必须知晓 L 集合中的 $\mathtt{open}\, n$ 。因此该情况不可能出现。第三种情况要求 $P \xrightarrow{A} P'$ 且 $\overline{h^{[A]}} \notin L$ ，由于 P 中唯一的顶层动作为 $\mathtt{open}\, n$ ，也不能满足要求。

(Conv Ctx Red) 此时有 $H(P) \longrightarrow R$ 且 $R \Downarrow_h^L$ 。由定理 5.20 可知，必有以下情况之一成立：

(Act Proc) 此时有 $P \longrightarrow P'$ 且 $R \equiv H(P') \Downarrow_h$ 。分析 P 的构造，由 $Threads(\Gamma,\ n) = 1$ 可知必有以下两种情况之一成立：

(1). $Q_2 \longrightarrow Q_2'$ 且 $P' \equiv \mathtt{open}\ n\ .\ P_1 \mid n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2']$ ：此时由 $H(P') \equiv R \Downarrow_h^L$ 利用归纳假设可得 $H(P_1 \mid Q_1 \mid Q_2') \Downarrow_h^L$ ，而由 $Q_2 \longrightarrow Q_2'$ 可知 $H(Q) \longrightarrow H(P_1 \mid Q_1 \mid Q_2')$ ，因此 $H(Q) \Downarrow_h^L$ 。

(2). $P' \equiv Q$ ：此时 $R \equiv H(Q) \Downarrow_h$ 。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P) \Downarrow_h^L$ ，由归纳假设知 $H'(Q) \Downarrow_h^L$ ，由 $H(Q) \longrightarrow H'(Q)$ 可得 $H(Q) \Downarrow_h^L$ 。

(Act Inter) 此时由于 $Threads(\Gamma,\ n) = 1$ ，分析各种情况对 P 的要求可知，最多只能有 **(Inter Open)** 和 **(Inter Amb Co-open)** 有可能成立，但进一步分析可以发现，这两种情况对 H 的要求均不满足给定 $L \not\sqsubseteq H$ 的条件，同样不可能成立。

通过以上对所有可能的分析可知，结论成立。 ∎

同样，对于 **(ST Open 2)** ，如果取消上下文条件，有类似推论 6.18 的结论。

推论 6.20 (ST Aux Open) 给定条件 $Threads(\Gamma,\ n) = 1$ ，下面的等价性关系成立：

(1). $\Gamma \triangleright (\nu n : T_n)(\mathtt{open}\ n\ .\ P_1 \mid n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2]) \simeq (\nu n : T_n)(P_1 \mid Q_1 \mid Q_2)$

(2). $\Gamma \triangleright h[\mathtt{open}\ n\ .\ P_1 \mid n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2] \mid A\ .\ R] \simeq h[P_1 \mid Q_1 \mid Q_2 \mid A\ .\ R]$ ，这里 A 可以是某个 $\overline{\mathtt{out}}$ 动作、或是一个 $\mathtt{open}\ n'$ 动作，这里 $n' \neq n$ 。

以上讨论了有类型条件的单线程定律，实际上，这些定律中的类型条件无非是为了限制其它并发进程，如这些定律中的 P_2 、 Q_2 等干扰动作的归约正常进行。如果不允许这些并发进程的存在，则显然有下面的无类型等价性定律成立。注意这里的约束构造采用了无类型的方式。

定理 6.21 以下进程等价性定律成立：

(UT Res In 1)	$(\nu n)(n[\overline{\mathtt{in}}\ m\ .\ P] \mid m[\mathtt{in}\ n\ .\ Q]) \simeq (\nu n)n[P \mid m[Q]]$
(UT Res In 2)	$(\nu m)(n[\overline{\mathtt{in}}\ m\ .\ P] \mid m[\mathtt{in}\ n\ .\ Q]) \simeq (\nu m)n[P \mid m[Q]]$
(UT Amb In 1)	$(h[n[\overline{\mathtt{in}}\ m\ .\ P] \mid m[\mathtt{in}\ n\ .\ Q] \mid \overline{\mathtt{out}}\ k\ .\ R]) \simeq h[n[P \mid m[Q] \mid \overline{\mathtt{out}}\ k\ .\ R]$
(UT Amb In 2)	$(h[n[\overline{\mathtt{in}}\ m\ .\ P] \mid m[\mathtt{in}\ n\ .\ Q] \mid \mathtt{open}\ k\ .\ R]) \simeq h[n[P \mid m[Q] \mid \mathtt{open}\ k\ .\ R]$
(UT Out)	$(n[\overline{\mathtt{out}}\ m\ .\ P \mid m[\mathtt{out}\ n\ .\ Q]] \simeq n[P] \mid m[Q]$
(UT Res Open)	$(\nu n)(\mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q]) \simeq (\nu n)(P \mid Q)$
(UT Amb Open 1)	$(h[\mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q]]) \simeq h[P \mid Q]$
(UT Amb Open 2)	$(h[\mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q] \mid \overline{\mathtt{out}}\ k\ .\ R]) \simeq h[P \mid Q \mid \overline{\mathtt{out}}\ k\ .\ R]$
(UT Amb Open 3)	$(h[\mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q] \mid \mathtt{open}\ k\ .\ R]) \simeq h[P \mid Q \mid \mathtt{open}\ k\ .\ R]$ ， $k \neq n$

证明 分别仿照 **(ST In)** 、 **(ST Out)** 、 **(ST Open 1)** 和 **(ST Open 2)** 可证，只需将原来利用进程单线程性所作的分析简化即可。 ∎

在具体的进程等价性分析过程中，有类型限制和无类型限制的等价性定律各有应用。有类型限制的定律可以处理存在与对应动作并发的进程的复杂情况，但对进程的类型有一定的约束，无类型限制的等价性定律则刚好相反。可以按照不同的场合分别使用不同的等价性定律进行证明。

§6.4 定点接收定律

另一类常用的进程结构是具有定点接收特性的复制结构，即进程中包含类似 $!A.P$ （或 $!n[A.P]$ ）的构造，并且在进程中其它位置不出现任何与 $!A.P$ （或 $!n[A.P]$ ）并置的动作 $A.Q$ （或 $n[A.Q]$ ）。这样任何与该定点接收构造发生的归约动作往往与其它归约动作执行的先后次序是可以互换的。本节将研究这一类等价性定律。

定义 6.22 （**定点接收特性**） 给定类型环境 Γ 和进程 P 满足 $\Gamma \vdash P : T$ ，称下列 P 中的复制构造具有定点接收特性，如果满足下表中各自对应的类型条件和唯一性条件。

复制构造及对应的可见动作	类型条件	唯一性条件： P 的任何归约结果中不会出现以下情况：
$!n[\mathtt{in}\, m.P \mid Q],\ n^{[\mathtt{in}\ m]}$	$Threads(\Gamma,\ n) = 1$	复制构造与 $n[\mathtt{in}\, m.P' \mid Q']$ 并置，并且 $n[\mathtt{in}\, m.P \mid Q] \not\longrightarrow^* n[\mathtt{in}\, m.P' \mid Q']$
$n[!\overline{\mathtt{in}}\, m.P \mid Q],\ n^{[\overline{\mathtt{in}}\ m]}$	$Stable(\Gamma,\ n) = True$	n 中并置 $\overline{\mathtt{in}}\, m.P'$
$!n[\mathtt{out}\, m.P \mid Q],\ n^{[\mathtt{out}\ m]}$	$Threads(\Gamma,\ n) = 1$	复制构造与 $n[\mathtt{out}\, m.P' \mid Q']$ 并置，并且 $n[\mathtt{out}\, m.P \mid Q] \not\longrightarrow^* n[\mathtt{out}\, m.P' \mid Q']$
$n[!\overline{\mathtt{out}}\, m.P \mid Q],\ n^{[\overline{\mathtt{out}}\ m]}$	$Stable(\Gamma,\ n) = True$	n 中并置 $\overline{\mathtt{out}}\, m.P'$
$!\mathtt{open}\, n.P,\ \mathtt{open}\, n$	（无）	复制构造与 $\mathtt{open}\, n.P'$ 并置
$!n[\overline{\mathtt{open}}.P \mid Q],\ n^{[\overline{\mathtt{open}}]}$	$Threads(\Gamma,\ n) = 1$	复制构造与 $n[\overline{\mathtt{open}}.P' \mid Q']$ 并置，并且 $n[\overline{\mathtt{open}}.P \mid Q] \not\longrightarrow^* n[\overline{\mathtt{open}}.P' \mid Q']$

如果复制构造对应的可见动作为 μ ，则也称 μ 在 P 中具有定点接收特性。

显然根据定点接收特性的定义有下面的引理成立。

引理 6.23 如果 μ 在 P 中具有定点接收特性，则对任意 $P \longrightarrow P'$ 有 μ 在 P' 中具有定点接收特性。

证明 考虑 P' 中对应复制构造的类型条件和唯一性条件。唯一性条件直接由定点接收特性的定义可以保证，而类型条件则可由类型的归约不变型定理可得。 ∎

在定点接收特性的定义中，最难确定唯一性条件，两个基于定点接收特性而等价的进程 P 和 Q ，其中的定点接收构造必须对于任意上下文 H 满足 $H(P)$ （假设 $P \longrightarrow Q$ ）中的该复制构造同样具有定点接收特性。但是可通过以下一些启发式规则来判定进程中复制构造的定点接收特性。

(1). 复制构造中灰箱名不在 H 或 P 的其它位置出现；

(2). 复制构造中的唯一性动作不在 H 或 P 的其它位置出现。

如果不能使用上述两种条件，则必须通过其它方法来检验唯一性条件，从而判断是否可以使用定点接收定律。

下面给出与上述六种情况相对应的六条定点接收定律。定点接收定律使用如下的形式表示：

$$\Gamma\{!\mu\} \triangleright P \simeq Q$$

其含义为：$\Gamma \vdash P : T$ 且对任意简单上下文 H 和灰箱名 h，如果 μ 在 $H(P)$（以及 $H(Q)$）中保持定点接收特性，那么 $H(P) \Downarrow_h^{\{\mu\}} \iff H(Q) \Downarrow_h^{\{\mu\}}$，限定观察上下文中不能出现 μ 也是为了保持 μ 的定点接收性。

定理 6.24 以下有关定点接收的等价性定律成立：

(Rec In) 如果 $Thread(\Gamma,\ m) = 1$，并且 $Thread(\Gamma,\ n) = 1$ 或 $Stable(\Gamma,\ n) = True$，那么
$\Gamma\{\,!m^{[\mathtt{in}\ n]}\} \triangleright n[\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2] \mid !\,m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2] \simeq n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !\,m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2]$

(Rec Co-in) 如果 $Stable(\Gamma,\ n) = True$ 且 $Threads(\Gamma,\ m) = 1$，那么
$\Gamma\{\,!n^{[\overline{\mathtt{in}}\ m]}\} \triangleright n[\,!\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2] \mid m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2] \simeq n[\,!\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2 \mid P_1 \mid m[Q_1 \mid Q_2]]$

(Rec Out) 如果 $Thread(\Gamma,\ m) = 1$，并且 $Threads(\Gamma,\ n) = 1$ 或 $Stable(\Gamma,\ n) = True$，那么
$\Gamma\{\,!m^{[\mathtt{out}\ n]}\} \triangleright n[\,\overline{\mathtt{out}}\ m\,.\,P_1 \mid P_2 \mid !\,m[\,\mathtt{out}\ n\,.\,Q_1 \mid Q_2] \simeq n[P_1 \mid P_2 \mid !\,m[\,\overline{\mathtt{out}}\ .\,Q_1 \mid Q_2]] \mid m[Q_1 \mid Q_2]$

(Rec Co-out) 如果 $Stable(\Gamma,\ n) = True$ 且 $Threads(\Gamma,\ m) = 1$，那么
$\Gamma\{\,!n^{[\overline{\mathtt{out}}\ m]}\} \triangleright n[\,!\,\overline{\mathtt{out}}\ m\,.\,P_1 \mid P_2 \mid m[\,\mathtt{out}\ n\,.\,Q_1 \mid Q_2] \simeq n[\,!\,\overline{\mathtt{out}}\ m\,.\,P_1 \mid P_2 \mid P_1] \mid m[Q_1 \mid Q_2]$

(Rec Open) 如果 $Threads(\Gamma,\ n) = 1$，那么
$\Gamma\{\,!\mathtt{open}\ n\} \triangleright\ !\mathtt{open}\ n\,.\,Q \mid n[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2] \simeq\ !\mathtt{open}\ n\,.\,Q \mid Q \mid P_1 \mid P_2$

(Rec Co-open) 如果 $Thread(\Gamma,\ n) = 1$，那么
$\Gamma\{\,!n^{[\overline{\mathtt{open}}]}\} \triangleright \mathtt{open}\ n\,.\,Q \mid !\,n[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2] \simeq Q \mid P_1 \mid P_2 \mid !\,n[\,\overline{\mathtt{open}}\,.\,P_1 \mid P_2]$

证明 我们只给出 **(Rec In)** 的完整证明，其它定律的证明方法类似。

令 $L = \{m^{[\mathtt{in}\ n]}\}$、$P = n[\,\overline{\mathtt{in}}\ m\,.\,P_1 \mid P_2] \mid !\,m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2]$、$Q = n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !\,m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2]$。对任意简单上下文 H 和灰箱名 h 满足 $H(P)$ 中的复制构造 $!\,m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_2]$ 具有定点接收特性，要证明 $H(P) \Downarrow_h^L \iff H(Q) \Downarrow_h^L$。由 $P \longrightarrow Q$，有 $H(Q) \Downarrow_h^L \Longrightarrow H(P) \Downarrow_h^L$，只需再证 $H(P) \Downarrow_h^L \Longrightarrow H(Q) \Downarrow_h^L$ 即可。

为了证明 $H(P) \Downarrow_h^L \Longrightarrow H(Q) \Downarrow_h^L$，我们证明一个更加一般的结论，即：对任意 P_m 有 $H(P \mid P_m) \Downarrow_h^L \Longrightarrow H(Q) \Downarrow_h^L$，其中：$P_m \equiv m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_{21}] \mid \cdots \mid m[\,\mathtt{in}\ n\,.\,Q_1 \mid Q_{2k}]$，$Q_2 \longrightarrow^* Q_{2i}$ $(i = 1..k)$，当 $k = 0$ 时，$P_m \equiv \mathbf{0}$。显然 $H(P) \longrightarrow H(P \mid P_m)$，由引理 6.23 可知 $m^{[\mathtt{in}\ n]}$ 在 $H(P \mid P_m)$ 中也具有定点接收特性，且在下面证明过程中的任意 $H(P \mid P_m) \longrightarrow^* R$ 中，$m^{[\mathtt{in}\ n]}$ 同样保持定点接收特性，在下面证明中使用归纳假设时也将默认使用此条件。

对 $H(P \mid P_m) \Downarrow_h^L$ 的推导过程归纳如下：

(Conv Ctx Exb) 此时有 $H(P \mid P_m) \downarrow_h^L$，根据引理 6.16 的三种情况可知：(1). 对于第一种情况，显然有 $H(Q) \downarrow_h^L$；(2). 对于第二种情况，分析 $P \mid P_m$ 的构造可知 h 只可能是 n 或 m，但此时能由 n 的单线程性可知观察到 n 的简单上下文都必须包含可见动作 $m^{[\mathtt{in}\ n]}$，而这违反了定点接受特性，因此 h 只可能为 m，而同时也有 $Q \downarrow_m^L$，因此 $H(Q) \downarrow_h^L$；(3). 第三种情况要求 $P \mid P_m \xrightarrow{A} P'$ 不可能出现。

(Conv Ctx Red) 此时有 $H(P \mid P_m) \longrightarrow R$ 且 $R \Downarrow_h^L$。由 $H(P \mid P_m) \longrightarrow R$ 利用定理 5.20 可知，必有以下情况之一成立：

(Act Proc) 此时有 $P \mid P_m \longrightarrow P'$ 且 $R \equiv H(P')$。令 $H' = - \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m$，则有 $H'(n[\mathtt{in}\, m\,.\,P_1 \mid P_2]) \longrightarrow P'$，再利用定理 5.20 可知，必有以下情况之一成立：

(Act Proc) 此时有 $n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2] \longrightarrow P''$ 且 $R \equiv H(H'(P'')) \Downarrow_h^L$。由已知 $Thread(\Gamma,\, n) = 1$ 或 $Stable(\Gamma,\, n) = True$ 利用引理 6.7 及引理 6.8 可知必有以下情况之一成立：

(1). $P_2 \longrightarrow P_2'$ 且 $P'' \equiv n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2']$：此时有 $R \equiv H(H'(P'')) \equiv H(H'(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2'])) \equiv H(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid P_2'] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m) \Downarrow_h^L$。利用归纳假设可得 $H(n[P_1 \mid P_2' \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2]) \Downarrow_h^L$，而由 $P_2 \longrightarrow P_2'$ 可知 $H(Q) \longrightarrow H(n[P_1 \mid P_2' \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2])$，因此 $H(Q) \Downarrow_h^L$。

(2). $P_2 > (\nu\vec{p} : \vec{T_p})\langle r[R_1]\rangle R_2$、$R_1 \xrightarrow{\mathtt{out}\, n} R_1'$、$R_2 \xrightarrow{\overline{\mathtt{out}}\, r} R_2'$、$n,\, r \notin \{\vec{p}\}$ 且 $P'' \equiv (\nu\vec{p} : \vec{T_p})(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid R_2'] \mid r[R_1'])$：此时有 $R \equiv H(H'(P'')) \equiv H(H'((\nu\vec{p} : \vec{T_p})(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid R_2'] \mid r[R_1']))) \equiv H((\nu\vec{p} : \vec{T_p})(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid R_2'] \mid r[R_1']) \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m)$。由于 $\vec{p}$ 为约束名，不妨设 $fn(\,!m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2]\,) \cap \{\vec{p}\} = \emptyset$。令 $H'' = H((\nu\vec{p} : \vec{T_p})(- \mid r[R_1']))$，则有 $R \equiv H''(n[\overline{\mathtt{in}}\, m\,.\,P_1 \mid R_2'] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m) \Downarrow_h^L$。利用归纳假设可得 $H''(n[P_1 \mid R_2' \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2]) \Downarrow_h^L$。而由已知条件可得 $Q \longrightarrow (\nu\vec{p} : \vec{T_p})(n[P_1 \mid R_2' \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid r[R_1'])$，因此 $H(Q) \longrightarrow H''(n[P_1 \mid R_2' \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2])$，因此有 $H(Q) \Downarrow_h^L$。

(3). 引理 6.7 的其它两种情况不可能发生。

(Act Har) 此时有 $H' \longrightarrow H''$ 且 $R \equiv H(H''(n[\mathtt{in}\, m\,.\,P_1 \mid P_2])) \Downarrow_h^L$。此时由于 $H' = - \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m$，由 $Threads(\Gamma,\, m) = 1$、$Threads(\Gamma,\, m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{2i}]) = 0$ 利用引理 6.7、6.9 可知，无论是哪种情况发生，只可能有 (1). 某个 $!m[\overline{\mathtt{in}}\, n\,.\,Q_1 \mid Q_2]$ 中的 $Q_2 \longrightarrow Q_2'$，或者 (2). 某个 $Q_{2i} \longrightarrow Q_{2i}'$ 对于情况 (1)，令 $P_m' = m[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_2'] \mid P_m$，对于情况 (2)，令 $P_m' = m[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_{21}] \mid \cdots \mid m[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_{2i}'] \mid \cdots \mid m[\overline{\mathtt{in}}\, m\,.\,Q_1 \mid Q_{2k}]$。以上两种情况下，$P_m'$ 均同样符合对 P_m 的要求。因此由 $R \equiv H(H''(n[\mathtt{in}\, m\,.\,P_1 \mid P_2])) \equiv H(n[\mathtt{in}\, m\,.\,P_1 \mid P_2] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m') \Downarrow_h^L$ 利用归纳假设可知 $H(Q) \Downarrow_h^L$。

(Act Inter) 此时分析 H' 和 $n[\mathtt{in}\, m\,.\,P_1 \mid P_2]$ 的构成可知，只可能有 **(Inter Amb Co-in)** 成立，且根据进入 n 的 m 不同，可有以下两种情况产生：

(1). $!m[\cdots]$ 中的某个 m 进入 n：即 $H(P') \equiv H(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m) \Downarrow_h^L$。此时由于 $Q_2 \longrightarrow^* Q_{2i}$，因此有 $Q \equiv n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \longrightarrow^* n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m$，因此 $H(Q) \Downarrow_h^L$。

(2). P_m 中的某个 $m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{2i}]$ 进入 n：即 $H(P') \equiv H(n[P_1 \mid P_2 \mid m[Q_1 \mid Q_{2i}]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m') \Downarrow_h^L$，其中 $P_m' \equiv m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{21}] \mid \cdots \mid m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{2i-1}] \mid m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{2i+1}] \mid \cdots \mid m[\mathtt{in}\, n\,.\,Q_1 \mid Q_{2k}]$。此时由于 $Q_2 \longrightarrow^* Q_{2i}$，因此有 $Q \equiv n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \longrightarrow^* n[P_1 \mid P_2 \mid m[Q_1 \mid Q_{2i}]] \mid !m[\mathtt{in}\, n\,.\,Q_1 \mid Q_2] \mid P_m'$，因此 $H(Q) \Downarrow_h^L$。

(Act Har) 此时有 $H \longrightarrow H'$ 且 $R \equiv H'(P \mid P_m) \Downarrow_h^L$。由归纳假设知 $H'(Q) \Downarrow_h^L$，由 $H(Q) \longrightarrow$

$H'(Q)$ 可得 $H(Q)\Downarrow_h^L$ 。

(Act Inter) 此时分析各种情况对 P 的要求可知，最多只能有后 5 种情况成立。

(Inter Amb In) 此时由于 $Thread(\Gamma,\ n)=1$ 或 $Stable(\Gamma,\ n)=True$ ，因此只可能 P 中的某个灰箱 m 进入 H ， P_m 这时保持不变或减少一项，但仍保持假设的条件。无论何种情况，用 P'_m 表示其该结果。因此有 $R\equiv H'(n[\overline{\mathtt{in}}\ m\,.\,P_1\mid P_2]\mid !\,m[\,\mathtt{in}\ n\,.\,Q_1\mid Q_2]\mid P'_m)\Downarrow_h^L$ 。由 $H(P\mid P_m)$ 中 $m^{[\mathtt{in}\ n]}$ 的定点接收特性可知，在 R 中 $m^{[\mathtt{in}\ n]}$ 也具有定点接收特性，因此利用归纳假设可知 $H'(Q)\Downarrow_h^L$ ，再由 $H(Q)\longrightarrow^* H'(Q)$ 可得 $H(Q)\Downarrow_h^L$ 。

(Inter Amb Co-in) 此时只可能有 $P_2\xrightarrow{\overline{\mathtt{in}}\ m'}P'_2$ 且 $R\equiv H'(n[\overline{\mathtt{in}}\ m\,.\,P_1\mid P'_2\mid m'[P_3]]\mid !\,m[\,\mathtt{in}\ n\,.\,Q_1\mid Q_2]\mid P_m)\Downarrow_h^L$ （否则可归入 **(Act Proc)** 情况，不然将违反定点接收特性）。同样由 R 中 $m^{[\mathtt{in}\ n]}$ 的定点接收特性利用归纳假设可知 $H'(n[\overline{\mathtt{in}}\ m\,.\,P_1\mid P'_2\mid m'[P_3]\mid m[Q_1\mid Q_2]]\mid !\,m[\,\mathtt{in}\ n\,.\,Q_1\mid Q_2])\Downarrow_h^L$ ，而同样有 $H(Q)\longrightarrow H'(n[\overline{\mathtt{in}}\ m\,.\,P_1\mid P'_2\mid m'[P_3]\mid m[Q_1\mid Q_2]]\mid !\,m[\,\mathtt{in}\ n\,.\,Q_1\mid Q_2])$ ，因此 $H(Q)\Downarrow_h^L$ 。

(Inter Amb Out 1)(Inter Amb Out 2)(Inter Amb Co-open) 这三种情况均不符合 $Thread(\Gamma,\ n)=1$ 或 $Stable(\Gamma,\ n)=True$ 以及 $Threads(\Gamma,\ m)=1$ 的要求，不可能成立。

通过以上对所有可能的分析可知，结论成立。 ∎

§6.5 等价性定律应用举例

本节将给出上述等价性定律的一些应用实例。第一部分证明第二章中灰箱换名例子的正确性。第二部分证明灰箱演算中经典的防火墙跨越例子在 ROAM 中的几种实现方案及其正确性证明。在下一章中还将用这些等价性定律证明 ROAM 演算翻译 π 演算的正确性。

§6.5.1 换名

第二章最后的换名例子中，我们指出，由于 ROAM 克服了 MA 的强干扰，换名操作所需要的条件将大大减少。这里我们利用本章的等价性定律给出换名操作前后进程等价的条件。

第二章中的换名构造 be 定义如下：

$$n\ \mathtt{be}\ m\,.\,P\ \triangleq\ m[\,\mathtt{out}\ n\,.\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n]\mid\overline{\mathtt{out}}\ m\,.\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,P$$

考虑进程 $n[n\ \mathtt{be}\ m\,.\,P\mid Q]$ ，假定类型环境 Γ 满足：

(1). $Type(\Gamma,\ P)=T_p$ 、 $Threads(\Gamma,\ P)=1$ ；

(2). $Threads(\Gamma,\ Q)=0$ ；

(3). $Type(\Gamma,\ n)=\curvearrowright^1[T_p]$ ；

(4). $Type(\Gamma,\ m)=\curvearrowright^1\cdot_t T_p$ 。

在上述类型环境 Γ 下，容易验证 $\Gamma\vdash n[n\ \mathtt{be}\ m\,.\,P\mid Q]:\underline{\vee}^0$ 。如果加上条件 $Q\overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ ，则下面的等价关系成立：

$$\Gamma\{n^{[\mathtt{in}\ m]},\ m^{[\overline{\mathtt{in}}\ n]}\}\ \triangleright\ n[n\ \mathtt{be}\ m\,.\,P\mid Q]\ \simeq\ m[P\mid Q]:\underline{\vee}^0 \tag{6.1}$$

式 (6.1) 的证明过程可完全通过代数方式得到。

首先，由 n 、 m 的单线程性和条件 $Q \overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ 利用等价性定律 **(ST Out)** 可知：

$$\begin{aligned}\Gamma \ \triangleright\ & n[m[\mathtt{out}\, n\,.\,\overline{\mathtt{in}}\, n\,.\,\mathtt{open}\, n] \mid \overline{\mathtt{out}}\, m\,.\,\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.\,P \mid Q] \\ \simeq\ & m[\overline{\mathtt{in}}\, n\,.\,\mathtt{open}\, n] \mid n[\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.\,P \mid Q] : \underline{\vee}^0 \end{aligned} \tag{6.2}$$

再由 n 、 m 的单线程性利用定律 **(ST In)** 可得：

$$\begin{aligned}\Gamma\{n^{[\mathtt{in}\ m]},\ m^{[\overline{\mathtt{in}}\ n]}\} \ \triangleright\ & m[\overline{\mathtt{in}}\, n\,.\,\mathtt{open}\, n] \mid n[\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\,.\,P \mid Q] \\ \simeq\ & m[\mathtt{open}\, n \mid n[\overline{\mathtt{open}}\,.\,P \mid Q]] : \underline{\vee}^0 \end{aligned} \tag{6.3}$$

类似利用 **(ST Open 1)** 可得：

$$\Gamma \ \triangleright\ m[\mathtt{open}\, n \mid n[\overline{\mathtt{open}}\,.\,P \mid Q]] \ \simeq\ m[P \mid Q] : \underline{\vee}^0 \tag{6.4}$$

综合上述等价性结论可知式 (6.1) 成立。

式 (6.1) 说明，如果不存在外界的干扰，即不存在并置的可能进入 m 的灰箱 n 和允许 n 进入的灰箱 m ，则换名操作前后的进程是等价的，即换名过程在所有满足上述条件的上下文中不会出现其它不希望的归约结果。

文献 [16] 中还研究了其它更加安全的换名构造，如下面的换名结构：

$$n[(\nu m)(n\ \mathtt{be}\ m\,.\,P) \mid Q] \tag{6.5}$$

此时只有进程 P 知道换名操作的结果灰箱名 m ，在这种情况下，上述对上下文中不可出现可见动作 $m^{[\overline{\mathtt{in}}\ n]}$ 、 $n^{[\mathtt{in}\ m]}$ 的限制就可以取消。

但是上述SA 版本的换名操作正确的条件还必须有 $Q \overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ ，因为在SA 版本中，$m[\mathtt{out}\, n\,.\,\cdots]$ 需要灰箱 n 中的协动作 $\overline{\mathtt{out}}\ n$ 即可发生归约，而不是 ROAM 版本中的 $\overline{\mathtt{out}}\ m$ 。在 ROAM 版本的换名构造中，无需条件 $Q \overset{m^{[\mathtt{out}\ n]}}{\not\Longrightarrow}$ ，即有下面的等价性规则成立。当然，这里假定 $m \notin fn(Q)$ ，并且 Γ 满足上述对 P 、 Q 和 n 的类型条件：

$$\Gamma \ \triangleright\ n[(\nu m : \curvearrowright^1 \cdot_t T_p)(n\ \mathtt{be}\ m\,.\,P) \mid Q] \ \simeq\ (\nu m : \curvearrowright^1 \cdot_t T_p) m[P \mid Q] : \underline{\vee}^0 \tag{6.6}$$

文献 [16] 中给出的 SA 版本的另外一种换名操作为（这里 h 为任意某个新灰箱名）：

$$\begin{aligned} n\ \mathtt{be}^+\ m\,.\,P \quad \triangleq\quad & (\nu h)(h[\mathtt{out}\, n\,.\,\mathtt{in}\, h\,.\,\overline{\mathtt{out}}\, h \mid m[\overline{\mathtt{in}}\, m\,.\,\mathtt{open}\, n]] \\ & \mid \overline{\mathtt{out}}\, n\,.\,\mathtt{in}\, h\,.\,\mathtt{in}\, m\,.\,\overline{\mathtt{open}}\, n\,.\,\mathtt{out}\, h\,.\,P) \end{aligned}$$

文献 [16] 中指出，如果满足条件 n 、 m 为单线程且 $n \notin fn(Q)$ ，则：

$$\Gamma \ \triangleright\ (\nu n)n[n\ \mathtt{be}^+\ m\,.\,P \mid Q] \ \simeq\ (\nu n)m[P \mid Q] \tag{6.7}$$

类似地，上述结论完全可以在 ROAM 中得到同样的实现，而且 SA 中一个很强的条件，即 n 为约束名且只在 P 中出现，在 ROAM 可以完全取消。ROAM 对协动作参数的限制就可以使得 Q 中的灰箱不会取代 h 作为协动作 $\overline{\mathtt{out}}\ h$ 的使用者。ROAM 中的 $\mathtt{be}^+$ 操作可定义如下：

$$\begin{aligned} n\ \mathtt{be}^+\ m\,.\,P \quad \triangleq \quad & (\nu h:\curvearrowright^1)(h[\,\mathtt{out}\ n\,.\,\overline{\mathtt{in}}\ n\,.\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,]] \\ & \mid \overline{\mathtt{out}}\ h\,.\,\mathtt{in}\ h\,.\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P) \end{aligned}$$

对进程 $n[n\ \mathtt{be}^+\ m\,.\,P \mid Q]$，只要选择合适的 P 和 Q 使得其线程数分别为 1 和 0，则容易得到类型环境 Γ 满足 $\Gamma \vdash n[n\ \mathtt{be}^+\ m\,.\,P \mid Q] : \underline{\vee}^0$ 且 $Threads(\Gamma,\ n) = Threads(\Gamma,\ m) = 1$。采用等价性定律可以相继得到以下结论：

$$\begin{aligned}
&\textbf{(ST Out)} \quad \Gamma \triangleright\ n[(\nu h:\curvearrowright^1)(h[\,\mathtt{out}\ n\,.\,\overline{\mathtt{in}}\ n\,.\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,]] \\
&\qquad\qquad \mid \overline{\mathtt{out}}\ h\,.\,\mathtt{in}\ h\,.\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P) \mid Q] \\
&\qquad \simeq (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{in}}\ n\,.\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,]] \mid n[\,\mathtt{in}\ h\,.\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,) \\
&\textbf{(ST Aux In)} \quad \Gamma \triangleright\ (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{in}}\ n\,.\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,]] \mid n[\,\mathtt{in}\ h\,.\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,) \\
&\qquad \simeq (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,] \mid n[\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,) \\
&\textbf{(ST Aux In)} \quad \Gamma \triangleright\ (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\overline{\mathtt{in}}\ n\,.\,\mathtt{open}\ n\,] \mid n[\,\mathtt{in}\ m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,) \\
&\qquad \simeq (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\mathtt{open}\ n \mid n[\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,]\,) \\
&\textbf{(ST Open 1)} \quad \Gamma \triangleright\ (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\mathtt{open}\ n \mid n[\,\overline{\mathtt{open}}\,.\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,]\,) \\
&\qquad \simeq (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,) \\
&\textbf{(ST Out)} \quad \Gamma \triangleright\ (\nu h:\curvearrowright^1)(h[\,\overline{\mathtt{out}}\ m \mid m[\,\mathtt{out}\ h\,.\,P \mid Q]\,]\,) \\
&\qquad \simeq (\nu h:\curvearrowright^1)(h[\,]\,) \mid m[P \mid Q] \\
&\textbf{(Coll Garb)} \quad \Gamma \triangleright\ (\nu h:\curvearrowright^1)(h[\,]\,) \mid m[P \mid Q] \\
&\qquad \simeq m[P \mid Q]
\end{aligned}$$

类似地，对于无类型限制的情况，有下面的等价关系成立：

$$n[n\ \mathtt{be}^+\ m\,.\,P] \quad \simeq \quad m[P] \tag{6.8}$$

其证明可以使用定理 6.21 中相应的无类型限制的等价性定律得到。可以看出这是一个很强的换名操作，对 n 、 m 和 P 均无任何限制。

上述等价性关系可以说明，ROAM 中的 $\mathtt{be}^+$ 构造具有非常健壮的换名功能，体现了 ROAM 进程在经过协动作参数限制后安全性的提高。

§6.5.2 防火墙跨越

防火墙跨越的例子 [5] 很大程度上说明了灰箱演算在描述移动计算安全性方面的优越特性。用灰箱 w 代表一个防火墙，其名字不为外部所知，用灰箱 m 代表一个在防火墙外部的授权代理（ agent ）， m 中的进程 Q 希望可以进入 w 内部运行。在文献 [5] 中，防火墙跨越协议是这样的。首先，防火墙灰箱 w 派出一个密钥灰箱 k ，其中包含有进入 w 所必须的能力 $\mathtt{in}\ w$ 。然后，代理 m 通过打开密钥灰箱 k 得到能力 $\mathtt{in}\ w$ ，并进入 w 。最后， m 在 w 中被打开，完成进程 Q 的释

放。在该协议中，要求代理灰箱名 m 和密钥灰箱 k 都是保密的。以下是 MA 版本防火墙跨越协议的典型实现 [5, 20] （ k' 用于防止 Q 的归约干预协议的进行 ）：

$$AG_{MA} \triangleq m[\,\mathtt{open}\,k\,.\,k'[Q]\,]$$

$$FW_{MA} \triangleq (\nu w)(w[\,k[\,\mathtt{out}\,w\,.\,\mathtt{in}\,m\,.\,\mathtt{in}\,w\,] \mid \mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P\,]\,)$$

文献 [20] 中给出了下面等价性关系的证明：

$$(\nu m,k,k')(FW_{MA} \mid AG_{MA}) \simeq (\nu w)(w[Q \mid P]\,) \tag{6.9}$$

这里要求 $w \notin fn(Q)$ 且 $(fn(Q) \cup fn(P) \cap \{m,\ k,\ k'\} = \emptyset$ 。

在文献 [16] 中详细分析了上述协议，并指出了其不足。例如，在上述等价性关系中对 m 和 k 的名字约束是必须的，否则 m 或 k 可能在协议进行过程中被第三方打开，或者被别的灰箱进入。没有条件 $\{m\} \cap fn(Q) = \emptyset$ ， $k'[Q]$ 可能会在 m 进入防火墙之前跳出 m 。文献 [16] 针对上述存在的问题，还给出了 SA 版本的防火墙跨越协议的如下实现方案：

$$AG_{SA} \triangleq m[\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,k\,.\,(x)x\,.\,\overline{\mathtt{open}}\,m\,.\,Q]$$

$$FW_{SA} \triangleq (\nu w)(w[\,\overline{\mathtt{out}}\,w\,.\,\overline{\mathtt{in}}\,w\,.\,\mathtt{open}\,m\,.\,P \mid k[\,\mathtt{out}\,w\,.\,\mathtt{in}\,m\,.\,\overline{\mathtt{open}}\,k\,.\langle \mathtt{in}\,w\rangle]\,]\,)$$

该版本与 MA 版本的主要差别在于： (i) Q 的行为不再需要一个额外的 k' 灰箱进行控制； (ii) 能力 $\mathtt{in}\,w$ 是通过消息通信完成传递的。在给定条件 $w \notin fn(Q)$ 和 $m \notin fn(P)$ 的条件下，文献 [16] 给出了如下两种进程等价关系：

$$(\nu m)(AG_{SA} \mid FW_{SA}) \simeq (\nu m,\ w)(w[P \mid Q]\,) \tag{6.10}$$

$$\{\mathtt{in}\,m,\ \overline{\mathtt{in}}\,m\} \triangleright AG_{SA} \mid FW_{SA} \simeq (\nu w)(w[P \mid Q]\,) \tag{6.11}$$

即如果 m 不为外界所知，或者外界在任何情况下不包含能力 $\mathtt{in}\,m$ 和 $\overline{\mathtt{in}}\,m$ ，则上述 SA 版本防火墙跨越协议的正确性就可以保证。 SA 版本对 MA 版本的重要改进在于取消了对密钥灰箱名 k 的约束限制，同时对灰箱名 m 的约束也有了一定程度的减弱。

下面给出用无类型 ROAM 实现防火墙协议的一些方法。通过这些例子可以从一定程度上看出 ROAM 在安全性方面的增强。

首先模仿类似 MA 和 SA 中采用的密钥灰箱进入代理灰箱的途径，第一个 ROAM 版本的防火墙跨越实现如下：

$$AG_1 \triangleq m[\,\overline{\mathtt{in}}\,k\,.\,\mathtt{open}\,k\,.\,k'[\,\overline{\mathtt{open}}\,.\,Q]\,]$$

$$FW_1 \triangleq (\nu w)(w[\,\overline{\mathtt{out}}\,k\,.\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P \mid k[\,\mathtt{out}\,w\,.\,\mathtt{in}\,m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,]\,]\,)$$

此时给定条件 $w \notin fn(Q)$ ，利用定理 6.21 中无类型的等价性定律可知，下面的等价性关系成立：

$$(\nu k)(AG_1 \mid FW_1) \simeq (\nu w,k)w[P \mid Q] \tag{6.12}$$

证明过程示意如下：

$$
\begin{array}{lcl}
 & & (\nu k)(AG_1 \mid FW_1) \\
 & = & (\nu k)(m[\,\overline{\mathtt{in}}\,k\,.\,\mathtt{open}\,k\,.\,k'[\,\overline{\mathtt{open}}\,.\,Q]\,] \\
 & & \mid (\nu w)(w[\,\overline{\mathtt{out}}\,k\,.\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P \\
 & & \mid k[\,\mathtt{out}\,w\,.\,\mathtt{in}\,m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,]\,]\,)) \\
\textbf{(UT Out)} & \simeq & (\nu w,k)(m[\,\overline{\mathtt{in}}\,k\,.\,\mathtt{open}\,k\,.\,k'[\,\overline{\mathtt{open}}\,.\,Q]\,] \mid k[\,\mathtt{in}\,m\,.\,\overline{\mathtt{open}}\,.\,\mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,] \\
 & & \mid w[\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P\,]\,) \\
\textbf{(UT Res In 2)} & \simeq & (\nu w,k)(m[\,\mathtt{open}\,k\,.\,k'[\,\overline{\mathtt{open}}\,.\,Q]\,] \mid k[\,\overline{\mathtt{open}}\,.\,\mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,]\,] \\
 & & \mid w[\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P\,]\,) \\
\textbf{(UT Amb Open 1)} & \simeq & (\nu w,k)(m[k'[\,\overline{\mathtt{open}}\,.\,Q]\,] \mid \mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,] \mid w[\,\overline{\mathtt{in}}\,m\,.\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P\,]\,) \\
\textbf{(UT Res In 1)} & \simeq & (\nu w,k)(w[\,\mathtt{open}\,m\,.\,\mathtt{open}\,k'\,.\,P \mid m[k'[\,\overline{\mathtt{open}}\,.\,Q]\,] \mid \overline{\mathtt{open}}\,]\,]\,) \\
\textbf{(UT Amb Open 1)} & \simeq & (\nu w,k)(w[\,\mathtt{open}\,k'\,.\,P \mid k'[\,\overline{\mathtt{open}}\,.\,Q]\,]\,]\,) \\
\textbf{(UT Amb Open 1)} & \simeq & (\nu w,k)(w[P \mid Q]\,)
\end{array}
$$

较之 SA 的实现，这里对 k 进行了约束，同时取消了对 m 的约束。SA 中对 m 所作的约束或者对 $\mathtt{in}\,m$ 动作的限制将大大影响代理在防火墙外部的意义。很多情况下，代理可以作为从防火墙中派出用于采集信息的软件对象。如果对代理进行了上述限制，就不可能使得代理从外部带回信息，最多起到在外部传播信息的作用，那么代理回到防火墙内也就没有太大的意义了。在 ROAM 的实现中，由于没有对 m 的约束， m 在进入防火墙之前，还可以与其它灰箱进行一些计算和信息交互，比起 SA 版本中对 m 的限制来，要更为合理和实用。同时对密钥 k 的约束也是合理的：在防火墙协议的实现中，灰箱 k 的唯一作用就是将代理带入防火墙，对其约束保证了密钥只为代理和防火墙所知。SA 中强调的所谓多个代理共享一个密钥一说其实是建立在对 m 使用的限制之上的。

除了仿照 MA 和 SA 中的实现外，以下还提供了两种更为简洁的实现方式。这两种实现较之以前实现方式的显著不同是利用类似 $(\nu k)w[\,\overline{\mathtt{in}}\,k\,.\,\cdots]$ 的构造来代替 $(\nu w)w[\,\overline{\mathtt{in}}\,m\,.\,\cdots]$ 构造。由于 ROAM 通过协动作的参数控制归约，这样防火墙灰箱名 w 的保密性限制便可取消，而将进入防火墙的条件完全建立在密钥 k 的传播上。

首先介绍的一种避免使用额外的灰箱 k' 的方案。在前一种实现中，使用 k 进入 m 的方案，这种方法使用了额外的灰箱 k'，来保证 Q 中的工作不会干扰 $\mathtt{in}\,w$ 的正常执行。下面采用让 m 进入 k 的方案，将更加简单。

$$
\begin{array}{lcl}
AG_2 & \triangleq & m[\,\mathtt{in}\,k\,.\,\overline{\mathtt{open}}\,.\,Q] \\
FW_2 & \triangleq & w[\,\overline{\mathtt{out}}\,k\,.\,\overline{\mathtt{in}}\,k\,.\,\mathtt{open}\,k\,.\,\mathtt{open}\,m\,.\,P \mid k[\,\mathtt{out}\,w\,.\,\overline{\mathtt{in}}\,m\,.\,\mathtt{in}\,w\,.\,\overline{\mathtt{open}}\,]\,]
\end{array}
$$

此时，同样有：

$$(\nu k)(AG_2 \mid FW_2) \simeq (\nu k)w[P \mid Q] \tag{6.13}$$

另外一种更为直观的方案使用了换名构造 $\mathtt{be}^+$ ：

$$
\begin{array}{lcl}
AG_3 & \triangleq & m[\,m\ \mathtt{be}^+\ k\,.\,\mathtt{in}\,w\,.\,Q] \\
FW_3 & \triangleq & w[\,\overline{\mathtt{in}}\,k\,.\,P]
\end{array}
$$

由于换名构造保证了 $m[m\ \mathtt{be}^+\ k\ .\ \mathtt{in}\ w\ .Q] \simeq k[\ \mathtt{in}\ w\ .Q]$，因此易知：

$$(\nu k)(AG_3 \mid FW_3) \simeq (\nu k)w[P \mid Q] \tag{6.14}$$

当然，由于取消了对防火墙灰箱名 w 的约束，实际使用时 P 和 Q 的内容都要有所限制，防止其中的 $\overline{\mathtt{in}}$ 动作将其它未经授权的灰箱带入防火墙内。

本章小结

本章综合前文对演化类型系统和进程等价性判定的研究，给出了用于判定进程等价的若干定律，包括无上下文限制的单线程定律、有上下文限制的单线程定律、定点接收定律以及作为推论的无类型等价性定律。利用这些定律可以非常简单地判定进程的等价性关系。本章最后用这些等价性定律直观地证明了换名例子和防火墙跨越例子在相应条件下的正确性，初步显示了等价性定律的应用价值。这些例子的 ROAM 实现均比 MA 和 SA 版本的实现放宽了条件限制，也进一步说明增加协动作的参数限制后， ROAM 在安全性控制方面的优越性。

第七章　π 演算的翻译

从灰箱演算提出之初，到以后对 SA 的研究，表达能力一直都作为一个重要的话题。Cardelli 和 Gordon[5] 在首次引入灰箱演算这一形式化模型时，就给出了没有本地通讯原语的纯 MA 演算对图灵机的模拟，也给出了带本地通讯原语的扩展 MA 对异步 π 演算的一个翻译，但是没有给出操作一致性的证明。Levi 和 Sangiorgi[16] 利用 SA 中没有强干扰的特性，给出了带本地通讯原语的扩展 SA 对多元异步 π 演算的翻译，同时利用文献 [16] 中的进程等价性定律证明了该翻译的操作一致性。在 ROAM 研究的前期 [58] 也曾给出了带本地通讯原语的 ROAM 演算对多元异步 π 演算的一个翻译。

上述研究中对 π 演算的翻译都无例外地使用了本地通讯原语。Zimmer[18] 首次给出了并证明了不带本地通讯原语的纯 SA 对同步 π 演算的翻译方案，使人们对灰箱演算的表达能力有了更进一步的认识。它最突出的特点是用灰箱的移动和打开这些单步的原子操作来模拟 π 演算和其变体中普遍存在的全局替换操作。为此，Zimmer 首先设计了一种没有全局替换操作的新的 π 演算变体—— π_{esc} 演算并给出了 π_{esc} 演算和 π 演算的相互翻译方法。以此为基础，Zimmer 给出了用纯 SA 对 π_{esc} 演算的翻译。最后，Zimmer 给出了用纯 SA 直接翻译 π 演算的方案并证明了该方案的操作一致性。

值得指出，文献 [18] 中对翻译方案的证明过程比较复杂。其直接原因在于，对 π 演算的翻译必然引入类型演化问题，即代表 π 通道的多线程非移动灰箱必然要打开代表 π 通讯进程的移动灰箱。虽然 Levi 和 Sangiorgi[16] 已经给出了判定 SA 进程等价的定律，但是由于文献 [16] 的类型系统尚未引入类似本文的类型演化机制，对进程的类型不能精确定义，其等价性定律无法使用在 Zimmer 翻译方案的证明中。

本章主要研究纯 ROAM 演算对 π 演算的翻译。本章采用了类似 Zimmer[18] 的方法，通过纯 ROAM 演算对 π_{esc} 演算的翻译来完成最后对 π 演算的翻译。本章首先介绍了 π 演算和 π_{esc} 演算的语法、归约语义和一些辅助的定义，并给出文献 [18] 中关于两种演算表达能力相同的结论。之后，本章给出了纯 ROAM 演算翻译 π_{esc} 的一个方案，并利用等价性定律给出了翻译过程操作一致性的证明。最后，本章给出了纯 ROAM 对 π 演算的翻译。

§7.1　π 演算

§7.1.1　语法

本文采用没有选择 (choice) 构造的同步 π 演算，下文如无特殊说明，π 演算一词均特指该演算。

为了翻译过程的需要，我们将通道（用 M 表示）区分为通道名（**Name**）和通道变量（**Var**）两种，通道名是指与约束操作对应的通道，用 m 、 n 等字母表示；通道变量特指被输入构造约束的通道，用 x 、 y 等字母表示。以下是 π 演算的语法定义：

$$P ::= \mathbf{0} \mid P \mid Q \mid\ !P \mid \overline{M}\langle M' \rangle . P \mid M(x) . P \mid (\nu n)P$$

$$M ::= n \in \mathbf{Name} \mid x \in \mathbf{Var}$$

在 $(\nu n)P$ 中，通道名 n 为约束名；在 $M(x).P$ 中，变量 x 为约束变量。对约束名和约束变量可用 α 转换进行换名，换名结果可看作与原进程等价。除约束名外的其它通道名称自由名；除约束变量外的其它变量称自由变量。进程 P 中的自由名和自由变量集合分别用 $fn(P)$ 和 $fv(P)$ 表示。

§7.1.2 归约语义

π 演算的归约语义由结构同余关系和归约规则组成。结构同余关系给出了 π 进程的等价变形能力，其规则如下：

(π **Struct Refl**)	$P \equiv P$
(π **Struct Symm**)	$Q \equiv P \Longrightarrow P \equiv Q$
(π **Struct Trans**)	$P \equiv Q,\ Q \equiv R \Longrightarrow P \equiv R$
(π **Struct Res**)	$P \equiv Q \Longrightarrow (\nu n)P \equiv (\nu n)Q$
(π **Struct Par**)	$P \equiv Q \Longrightarrow P \mid R \equiv Q \mid R$
(π **Struct Repl**)	$P \equiv Q \Longrightarrow !P \equiv !Q$
(π **Struct Output**)	$P \equiv Q \Longrightarrow \overline{M}\langle M' \rangle.P \equiv \overline{M}\langle M' \rangle.Q$
(π **Struct Input**)	$P \equiv Q \Longrightarrow M(x).P \equiv M(x).Q$
(π **Struct Par Zero**)	$P \mid \mathbf{0} \equiv P$
(π **Struct Par Comm**)	$P \mid Q \equiv Q \mid P$
(π **Struct Par Assoc**)	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
(π **Struct Res Zero**)	$(\nu n)\mathbf{0} \equiv \mathbf{0}$
(π **Struct Res Res**)	$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$
(π **Struct Res Par**)	$n \notin fn(P) \Longrightarrow (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$
(π **Struct Repl Par**)	$!P \equiv P \mid !P$
(π **Struct Repl Zero**)	$!\mathbf{0} \equiv \mathbf{0}$

π 演算进程的归约用符号 “ $\longrightarrow_\pi$ ” 表示。 π 演算的归约规则如下，其中最主要的规则是 (π **Red Comm**)，构造 $Q\{M/x\}$ 表示将 Q 中所有的自由变量 x 替换为 M ：

(π **Red Comm**)	$\dfrac{}{\overline{n}\langle m \rangle.P \mid n(x).Q \longrightarrow_\pi P \mid Q\{m/x\}}$
(π **Red Par**)	$\dfrac{P \longrightarrow_\pi P'}{P \mid Q \longrightarrow_\pi P' \mid Q}$
(π **Red Res**)	$\dfrac{P \longrightarrow_\pi P'}{(\nu n)P \longrightarrow_\pi (\nu n)P'}$

(π Red Struct) $$\frac{P \equiv P' \quad P' \longrightarrow_\pi P'' \quad P'' \equiv P'''}{P \longrightarrow_\pi P'''}$$

§7.2 π_{esc} 演算

本节简要介绍一个与 π 演算等价的 π_{esc} 演算 [18]。π_{esc} 演算指使用显式替换和通道的 π 演算变体（ π-calculus with explicit substitution and channel ）。由于 π_{esc} 演算与 π 演算表达能力相同 [18]，但不使用全局变量替换，因此更加适合使用没有本地通讯原语的纯灰箱演算进行模拟。我们采用的策略也是先翻译 π_{esc} 演算，然后再给出对 π 演算的直接翻译。

§7.2.1 树状替换

在介绍 π_{esc} 演算之前，首先引入 π_{esc} 演算中会用到的一个重要概念——树状替换。

定义 7.1 *[18]*（ **树状替换** ） 树状替换是一个映射 $\sigma : \mathbf{Var} \to \mathbf{Var} \cup \mathbf{Name}$，且满足：

(1). σ 的值域 $im(\sigma) \subseteq \mathbf{Name} \cup dom(\sigma)$；

(2). 对任意 $x \in dom(\sigma)$，存在整数 k，使得 $x\sigma^k \in \mathbf{Name}$（即：不会产生循环）。

事实上，树状替换可以图示为一个由树组成的森林结构：根节点属于 **Name** 集合，其它节点（组成 $dom(\sigma)$ ）属于 **Var** 集合，对任意非根节点，总可沿所在的树找到其对应的根节点。

用 $\emptyset$ 表示一个空的树状替换。如果 $x \notin dom(\sigma)$ 且 $M \in \mathbf{Name} \cup dom(\sigma)$，用 $\{M/x\} \uplus \sigma$ 表示将 $\{M/x\}$ 增加到 σ 后形成的新的树状替换。定义对树状替换的取自由名函数为 $fn(\sigma) \triangleq im(\sigma) \cap \mathbf{Name}$。另外， $P\sigma$ 表示将树状替换直接作用在进程上，结果为将进程中所有自由变量 x 替换为 $x\sigma$。

§7.2.2 语法

π_{esc} 演算在语法上是 π 演算的一个扩展，主要在 π 演算语法的基础上增加了显式通道构造“ $[n : S]$ ”、显式变量构造“ $(\nu x : M)$ ”和相应的辅助构造。其语法定义如下 [18]：

$$P ::= \mathbf{0} \mid P \mid Q \mid !P \mid \overline{M}\langle M' \rangle . P \mid M(x) . P \mid (\nu n)P \mid [n : S] \mid (\nu x : M)P$$

$$M ::= n \in \mathbf{Name} \mid x \in \mathbf{Var}$$

$$S ::= \epsilon \mid S \mid S' \mid \langle M \rangle . P \mid (x) . P$$

显式通道构造“ $[n : S]$ ”表示一个名字为 n 的通道，通道中的内容 S 为一组将在该通道上执行输入输出操作的进程，用 $(x) . P$ 表示输入进程、$\langle M \rangle . P$ 表示输出进程， S 的元素间也可看作是并发执行关系。

显式变量构造“ $(\nu x : M)P$ ”（ $x \neq M$ ）表示一个新的显式变量，其值为 M，作用范围为 P。进程 P 中 x 为约束变量，在归约时将被替换为 M。

§7.2.3 结构同余关系

π_{esc} 进程的结构同余关系在保留 π 演算结构同余关系的基础上，增加以下内容：

(π_{esc} **Struct Refl**) $S \equiv S$

(π_{esc} **Struct Symm**) $S \equiv S' \Longrightarrow S' \equiv S$

(π_{esc} **Struct Trans**) $S \equiv S',\ S' \equiv S'' \Longrightarrow S \equiv S''$

(π_{esc} **Struct Channel**) $S \equiv S' \Longrightarrow [n : S] \equiv [n : S']$

(π_{esc} **Struct Var**) $P \equiv Q \Longrightarrow (\nu x : M)P \equiv (\nu x : M)Q$

(π_{esc} **Struct Abs**) $S' \equiv S'' \Longrightarrow S \mid S' \equiv S \mid S''$

(π_{esc} **Struct Out Abs**) $P \equiv Q \Longrightarrow \overline{M}\langle M' \rangle . P \equiv \overline{M}\langle M' \rangle . Q$

(π_{esc} **Struct In Abs**) $P \equiv Q \Longrightarrow (x) . P \equiv (x) . Q$

(π_{esc} **Struct Abs Zero**) $S \mid \epsilon \equiv S$

(π_{esc} **Struct Abs Comm**) $S \mid S' \equiv S' \mid S$

(π_{esc} **Struct Abs Assoc**) $(S \mid S') \mid S'' \equiv S \mid (S' \mid S'')$

(π_{esc} **Struct Var Par**) $x \notin fv(P) \Longrightarrow (\nu x : M)(P \mid Q) \equiv P \mid (\nu x : M)Q$

(π_{esc} **Struct Res Var**) $(\nu n)(\nu x : M)P \equiv (\nu x : M)(\nu n)P$

(π_{esc} **Struct Var Var**) $x \neq y, x \neq M', y \neq M \Longrightarrow (\nu x : M)(\nu y : M')P \equiv (\nu y : M')(\nu x : M)P$

§7.2.4 归约规则

π_{esc} 演算归约规则的一般形式为 “ $\sigma : P \longrightarrow_{esc} Q$ ”。这里 σ 是一个树状替换，作为进程归约的环境，它包含了进程中所有自由变量对应的值。为了得到 P 中所有自由变量的值，下面的归约规则都必须满足条件 $fv(P) \subseteq dom(\sigma)$ 。 π_{esc} 演算的归约规则如下：

(π_{esc} **Red Subst Out**) $$\frac{x\sigma = M}{\sigma : \overline{x}\langle M' \rangle . P \longrightarrow_{esc} \overline{M}\langle M' \rangle . P}$$

(π_{esc} **Red Subst In**) $$\frac{x\sigma = M}{\sigma : x(y) . P \longrightarrow_{esc} M(y) . P}$$

(π_{esc} **Red Output**) $$\frac{}{\sigma : [n : S] \mid \overline{n}\langle M \rangle . P \longrightarrow_{esc} [n : S \mid \langle M \rangle . P]}$$

(π_{esc} **Red Input**) $$\frac{}{\sigma : [n : S] \mid n(x) . P \longrightarrow_{esc} [n : S \mid (x) . P]}$$

(π_{esc} **Red Comm**) $$\frac{x \neq M}{\sigma : [n : S \mid \langle M \rangle . P \mid (x) . Q] \longrightarrow_{esc} [n : S] \mid P \mid (\nu x : M)Q}$$

(π_{esc} **Red Var**) $$\frac{x \notin dom(\sigma) \quad \{M/x\} \uplus \sigma : P \longrightarrow_{esc} P'}{\sigma : (\nu x : M)P \longrightarrow_{esc} (\nu x : M)P'}$$

(π_{esc} **Red Par**) $\dfrac{P \longrightarrow_{esc} P'}{P \mid Q \longrightarrow_{esc} P' \mid Q}$

(π_{esc} **Red Res**) $\dfrac{P \longrightarrow_{esc} P'}{(\nu n)P \longrightarrow_{esc} (\nu n)P'}$

(π_{esc} **Red Struct**) $\dfrac{P \equiv P' \quad P' \longrightarrow_{esc} P'' \quad P'' \equiv P'''}{P \longrightarrow_{esc} P'''}$

在上面的归约规则中，(π_{esc} **Red Subst Out**) 和 (π_{esc} **Red Subst In**) 用于将进程中的变量替换为环境中的值。(π_{esc} **Red Output**) 和 (π_{esc} **Red Input**) 使得准备进行通讯的进程进入相应的通道。(π_{esc} **Red Comm**) 用于完成通道内输入和输出进程的通讯，通讯使得输入进程中的遗留部分被一个新的代表了变量替换的显式变量构造所约束。这样，在 π_{esc} 中，显式通道构造将原来 π 演算中的一次通讯动作划分为两个步骤，首先准备在通道上输入（或输出）的进程找到并进入相对应的显式通道，然后在显式通道中，输入输出进程相遇并完成通讯。再通过显式变量构造，原来的通讯导致的全局替换被转化为一个显式变量构造，遗留进程中的这个变量每次被使用时，将通过替换得到具体的值。(π_{esc} **Red Var**) 表示进程的显示替换操作可被放入环境中。最后是三条与 π 演算类似的推导规则。

§7.2.5 合法进程

完全按照 π_{esc} 演算语法构成的进程会出现一些无效的构造，例如进程 $\overline{n}\langle m\rangle\,.\,[p : S]$ 中的显式通道 p 必须等到输出操作完成才能投入使用；进程 $[n : S] \mid [n : S'] \mid \overline{n}\langle m\rangle\,.\,P \mid n(x)\,.\,Q$ 在归约过程中可能使输入输出进程进入不同的显式通道而无法通讯。因此，文献 [18] 中定义了以下一些断言来判断进程的合法性。

记号 $P \Downarrow_1 n$ 表示 P 中包含至少一个名为 n 的自由显式通道。记号 $P \Downarrow_2 n$ 表示 P 中包含至少两个名为 n 的自由显式通道。

定义 7.2 （ $P \Downarrow_i n,\ i = 1, 2$ ）

(π_{esc} **Pres Res**) $\dfrac{P \Downarrow_i n \quad m \neq n}{(\nu m)P \Downarrow_i n}$ (π_{esc} **Pres ParL**) $\dfrac{P \Downarrow_i n}{P \mid Q \Downarrow_i n}$

(π_{esc} **Pres ParR**) $\dfrac{Q \Downarrow_i n}{P \mid Q \Downarrow_i n}$ (π_{esc} **Pres Par 2**) $\dfrac{P \Downarrow_1 n \quad Q \Downarrow_1 n}{P \mid Q \Downarrow_2 n}$

(π_{esc} **Pres Repl**) $\dfrac{P \Downarrow_i n}{!P \Downarrow_i n}$ (π_{esc} **Pres Repl 2**) $\dfrac{P \Downarrow_1 n}{!P \Downarrow_2 n}$

(π_{esc} **Pres Output**) $\dfrac{P \Downarrow_i n}{\overline{M}\langle M'\rangle\,.\,P \Downarrow_i n}$ (π_{esc} **Pres Input**) $\dfrac{P \Downarrow_i n}{M(x)\,.\,P \Downarrow_i n}$

(π_{esc} **Pres Channel 1**) $\dfrac{}{[n : S] \Downarrow_1 n}$ (π_{esc} **Pres Channel 2**) $\dfrac{S \Downarrow_1 n}{[n : S] \Downarrow_2 n}$

(π_{esc} **Pres Channel**) $\dfrac{S \Downarrow_i m}{[n : S] \Downarrow_i m}$ (π_{esc} **Pres Var**) $\dfrac{P \Downarrow_i n}{(\nu x : M)P \Downarrow_i n}$

(π_{esc} **Pres AbsL**) $\dfrac{S \Downarrow_1 n}{S \mid S' \Downarrow_1 n}$ (π_{esc} **Pres AbsR**) $\dfrac{S' \Downarrow_1 n}{S \mid S' \Downarrow_1 n}$

(π_{esc} **Pres Abs 2**) $\dfrac{S \Downarrow_1 n \quad S' \Downarrow_1 n}{S \mid S' \Downarrow_2 n}$ (π_{esc} **Pres Out Abs**) $\dfrac{P \Downarrow_i n}{\langle M\rangle\,.\,P \Downarrow_i n}$

$$(\pi_{esc}\ \textbf{Pres In Abs}) \quad \frac{P \Downarrow_i n}{(x)\,.\,P \Downarrow_i n}$$

另外定义 $pr(P) \triangleq \{n \mid n \in \textbf{Name} \wedge P \Downarrow_1 n\}$ 。显然有 $pr(P) \subseteq fn(P)$ 。

在此基础上，用判定 $\vdash P : OK$ 表示 P 中的显式通道不出现在前缀构造和复制构造内部且在名字约束构造内至多只出现一个该约束名对应的显式通道构造。判定 $\vdash P : OK$ 的推导规则如下：

$$(\pi_{esc}\ \textbf{OK Res}) \quad \frac{\vdash P : OK \quad P \not\Downarrow_2 n}{\vdash (\nu n)P : OK}$$

$$(\pi_{esc}\ \textbf{OK Zero}) \quad \frac{}{\vdash \mathbf{0} : OK}$$

$$(\pi_{esc}\ \textbf{OK Par}) \quad \frac{\vdash P : OK \quad \vdash Q : OK}{\vdash P \mid Q : OK}$$

$$(\pi_{esc}\ \textbf{OK Repl}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_1 n}{\vdash\, !P : OK}$$

$$(\pi_{esc}\ \textbf{OK Output}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_1 n}{\vdash \overline{M}\langle M' \rangle\,.\,P : OK}$$

$$(\pi_{esc}\ \textbf{OK Input}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_1 n}{\vdash M(x)\,.\,P : OK}$$

$$(\pi_{esc}\ \textbf{OK Channel}) \quad \frac{\vdash S : OK}{\vdash [n : S] : OK}$$

$$(\pi_{esc}\ \textbf{OK Var}) \quad \frac{\vdash P : OK}{\vdash (\nu x : M)P : OK}$$

$$(\pi_{esc}\ \textbf{OK Eps}) \quad \frac{}{\vdash \epsilon : OK}$$

$$(\pi_{esc}\ \textbf{OK Abs}) \quad \frac{\vdash S : OK \quad \vdash S' : OK}{\vdash S \mid S' : OK}$$

$$(\pi_{esc}\ \textbf{OK Out Abs}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_1 n}{\vdash \langle M \rangle\,.\,P : OK}$$

$$(\pi_{esc}\ \textbf{OK In Abs}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_1 n}{\vdash (x)\,.\,P : OK}$$

最后，一个合法进程（用 $\vdash P : Valid$ 表示）的定义如下：

$$(\pi_{esc}\ \textbf{Valid}) \quad \frac{\vdash P : OK \quad \forall n \in \textbf{Name}\ \ P \not\Downarrow_2 n}{\vdash P : Valid}$$

对于判定 $\vdash P : Valid$ ，文献 [18] 中给出了如下的结论：

引理 7.3 (文献 [18]) 如果 $\sigma : P \longrightarrow_{esc} Q$ 且 $\vdash P : Valid$ ，那么 $\vdash Q : Valid$ 。

§7.2.6 通道闭包

合法进程中限定了进程中显式通道出现的位置和个数，但是为了使进程可以顺利归约，还需保证在适当的地方必须提供必要的显式通道构造使得进程可以利用该通道进行通讯。通道闭包操作因此而来。首先操作 $cl(P)$ 用于为进程 P 中所有的约束名构造增加必要的显式通道：

$$cl((\nu n)P) \quad \triangleq \quad (\nu n)([n : \epsilon] \mid cl(P)) \quad \text{如果 } P \not\Downarrow_1 n$$

$$(\nu n)cl(P) \qquad \text{如果 } P \Downarrow_1 n$$

除此之外，通道闭包操作 $cl(P)$ 对于其它构造不产生作用。

在此基础上，操作 $cl_\sigma(P)$ 用于在 $cl(P)$ 的基础上补充所有 P 未提供的自由名以及 σ 中所有通道名所对应的显式通道，这样， P 归约所需的显式通道就全部补齐了。

$$cl_\sigma(P) \quad \triangleq \quad [n_1 : \epsilon] \mid \cdots \mid [n_k : \epsilon] \mid cl(P)$$

这里 $\{n_1, \cdots, n_k\} = (fn(P) \cup fn(\sigma) \backslash pr(P))$ 。

§7.2.7 π 演算和 π_{esc} 演算的关系

本节给出文献 [18] 中证明的关于 π 演算和 π_{esc} 演算的关系。

将 π 进程 P 翻译成 π_{esc} 进程的操作就是 $cl_\emptyset(P)$ （这里把 P 看作是一个 π_{esc} 进程）；将 π_{esc} 进程翻译成 π 进程的操作 $\{\!|P|\!\}$ 定义为将显式通道中的内容返还成通道的输入输出动作，并将显式变量翻译成经过变量替换后的进程。对以上翻译操作，下面的定理成立。

定理 7.4 (文献 [18])

(1). 如果 $\sigma : P \longrightarrow_{esc} Q$ ，那么 $\{\!|P|\!\} \equiv \{\!|Q|\!\}$ 或者 $\{\!|P|\!\} \longrightarrow_\pi \{\!|Q|\!\}$ ；

(2). 如果 $\{\!|P|\!\} \longrightarrow_\pi Q$ 、 $cl_\emptyset(P) \equiv P$ 、 $\vdash P : Valid$ 且 $fv(P) = \emptyset$ ，那么存在 π_{esc} 进程 P' 满足 $\emptyset : P \longrightarrow_{esc} P'$ 且 $\{\!|P'|\!\} \equiv Q$ 。

由于 π_{esc} 演算显然比 π 演算更加复杂、每个归约动作更加基本，但上述定理说明了用 π 演算可以翻译 π_{esc} 演算这一事实。因此可以认为 π 演算和 π_{esc} 演算具有相同的表达能力。

§7.3 π_{esc} 演算的翻译

本节给出了将 π_{esc} 演算翻译为 ROAM 的一种方案，并利用已有的 ROAM 进程等价性定律，给出翻译方案操作一致性的证明。

§7.3.1 翻译过程

在下面的翻译过程中，取名字 $enter, c, r, r_1, r_2, w, w_1 \notin \mathbf{Name} \cup \mathbf{Var}$ 。首先我们定义如下的一些简写 [1] 。

$$\begin{array}{lcl}
\mathbf{server}\ a\,.\,P & \triangleq & !enter[\mathtt{in}\,a\,.\,\overline{\mathtt{open}}.\,P] \\
\mathbf{request}\ n & \triangleq & \mathtt{in}\,n\,.\,\overline{\mathtt{in}}\,enter\,.\,\mathtt{open}\,enter \\
\mathbf{request}\ x & \triangleq & \mathtt{in}\,x\,.\,\overline{\mathtt{in}}\,enter\,.\,\mathtt{out}\,x\,.\,\mathtt{open}\,enter \\
\mathbf{fwd}\ \ M & \triangleq & \mathbf{server}\,r\,.\,\mathbf{request}\,M \mid \mathbf{server}\,w\,.\,\mathbf{request}\,M \\
\mathbf{allowIO} & \triangleq & !\,\overline{\mathtt{in}}\,r \mid !\,\overline{\mathtt{out}}\,r \mid !\,\overline{\mathtt{in}}\,w \mid !\,\overline{\mathtt{out}}\,w
\end{array}$$

[1] 对于定义中的 request 项，针对参数是通道名或通道变量，结果会不相同，这种方法虽然不很规范，但方便了 fwd 的定义，且使得下文的翻译更加直观。否则，必须在使用这些简写时，还要分情况考虑，显得不紧凑。

这些简写主要用来完成 client/server 模式的请求内容和返回结果，如：

$$x[\mathbf{server}\ agent\,.\,P] \mid agent[\mathbf{request}\ x \mid Q] \quad \longrightarrow^{+} \quad x[\mathbf{server}\ agent\,.\,P] \mid agent[P \mid Q] \tag{7.1}$$

在处理 π_{esc} 的读进程和写进程时，分别将其翻译为 $r[\mathbf{request}\ M \mid \cdots]$ 和 $w[\mathbf{request}\ M \mid \cdots]$，其中 M 为输入输出操作所在的目标通道。对于显式变量构造 $(\nu x : M')$，其翻译结果为 $x[\mathbf{fwd}\ \ M' \mid \mathbf{allowIO}]$，则如果读进程或写进程中的目标通道为变量 x，则通过下面的归约，可使目标通道变为显示变量 x 的值 M：

$$x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid r[\mathbf{request}\ x \mid \cdots] \quad \longrightarrow^{+} \quad x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid r[\mathbf{request}\ M \mid \cdots] \tag{7.2}$$

$$x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ x \mid \cdots] \quad \longrightarrow^{+} \quad x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ M \mid \cdots] \tag{7.3}$$

对于显式通道构造 $[n : S]$，我们将其翻译为一个名为 n 的服务灰箱。其内容分为固定的服务部分（ $\mathbf{server}\ r\,.\,P_r \mid \mathbf{server}\ w\,.\,P_w \mid \ \mathbf{allowIO}$ ）和暂存的客户部分（ S 的翻译结果），这样当外部的读写进程请求通道 n 的服务时，有：

$$\begin{aligned} & n[\mathbf{server}\ r\,.\,P_r \mid \mathbf{server}\ w\,.\,P_w \mid \mathbf{allowIO} \mid \cdots] \mid r[\mathbf{request}\ n \mid \cdots] \\ \longrightarrow^{+} \quad & n[\mathbf{server}\ r\,.\,P_r \mid \mathbf{server}\ w\,.\,P_w \mid \mathbf{allowIO} \mid \cdots \mid r[P_r \mid \cdots]] \end{aligned} \tag{7.4}$$

$$\begin{aligned} & n[\mathbf{server}\ r\,.\,P_r \mid \mathbf{server}\ w\,.\,P_w \mid \mathbf{allowIO} \mid \cdots] \mid w[\mathbf{request}\ n \mid \cdots] \\ \longrightarrow^{+} \quad & n[\mathbf{server}\ r\,.\,P_r \mid \mathbf{server}\ w\,.\,P_w \mid \mathbf{allowIO} \mid \cdots \mid w[P_w \mid \cdots]] \end{aligned} \tag{7.5}$$

注意在 **request** 的定义中，当参数为变量时包含跳出服务器灰箱的动作，而参数为通道名时则不包含跳出动作。

在下面的翻译代码中，将显式通道中的 P_r 和 P_w 设定为让灰箱 $r[P_r \mid \cdots]$ 和 $w[P_w \mid \cdots]$ 进行交互来模拟读进程和写进程在通道内的通讯，并在交互的过程中产生与归约结果相对应的翻译代码。这样就完成了用 ROAM 模拟 π_{esc} 演算操作语义的目的。

下面我们利用以上简写，给出对任何 π_{esc} 进程 P 在给定的某树状替换 σ 下的具体翻译方案 $\langle\!\langle \sigma,\ P \rangle\!\rangle$ （基于 $\langle\!\langle P \rangle\!\rangle$ 、 $\langle\!\langle S \rangle\!\rangle_n$ 和 $\langle\!\langle \sigma \rangle\!\rangle$ ）：

$$\begin{aligned}
\langle\!\langle \mathbf{0} \rangle\!\rangle &\triangleq \mathbf{0} \\
\langle\!\langle P \mid Q \rangle\!\rangle &\triangleq \langle\!\langle P \rangle\!\rangle \mid \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle\ !P \rangle\!\rangle &\triangleq\ !\langle\!\langle P \rangle\!\rangle \\
\langle\!\langle (\nu n) P \rangle\!\rangle &\triangleq (\nu n : \underline{\vee}^{\omega}) \langle\!\langle P \rangle\!\rangle \\
\langle\!\langle \overline{M}\langle M' \rangle\,.\,P \rangle\!\rangle &\triangleq w[\mathbf{request}\ M \mid c[\mathtt{out}\ r\,.\,\overline{\mathtt{open}}\,.\,\langle\!\langle P \rangle\!\rangle] \\
&\quad \mid w_1[\,\mathtt{in}\ r_1\,.\,\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M' \mid \mathbf{allowIO})]\,] \\
\langle\!\langle M(x)\,.\,P \rangle\!\rangle &\triangleq r[\mathbf{request}\ M \mid (\nu x : \underline{\vee}^{\omega})(c[\mathtt{out}\ r\,.\,\overline{\mathtt{open}}\,.\,\langle\!\langle P \rangle\!\rangle] \mid r_1[\,\overline{\mathtt{in}}\ w_1\,.\,\mathtt{in}\ x\,.\,\overline{\mathtt{open}}] \\
&\quad \mid x[\,\overline{\mathtt{in}}\ r_1\,.\,\overline{\mathtt{out}}\ r_2\,.\,\mathtt{open}\ r_1\,.\,\mathtt{open}\ w_1 \mid r_2[\mathtt{out}\ x\,.\,\overline{\mathtt{open}}]\,])\,]
\end{aligned}$$

$$
\begin{array}{lcl}
 & & \mid \mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r \\
\langle\!\langle [n : S] \rangle\!\rangle & \triangleq & n[\mathbf{allowIO} \mid \mathbf{server}\ w\ .\ \mathtt{in}\ r\ .\ \overline{\mathtt{open}} \\
 & & \mid \mathbf{server}\ r\ .\ \overline{\mathtt{in}}\ w\ .\ \mathtt{out}\ n\ .\ \overline{\mathtt{out}}\ c\ .\ \mathtt{open}\ w\ .\ \overline{\mathtt{out}}\ c\ .\ \mathtt{open}\ r_2\ .\ \overline{\mathtt{open}} \mid \langle\!\langle S \rangle\!\rangle_n] \\
 & & \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \\
 & & (\text{这里 } k \text{ 为 } S \text{ 中读进程的个数}) \\
\langle\!\langle (\nu x : M) P \rangle\!\rangle & \triangleq & (\nu x : \underline{\vee}^{\omega})(x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \langle\!\langle P \rangle\!\rangle) \\
\langle\!\langle \epsilon \rangle\!\rangle_n & \triangleq & \mathbf{0} \\
\langle\!\langle S \mid S' \rangle\!\rangle_n & \triangleq & \langle\!\langle S \rangle\!\rangle_n \mid \langle\!\langle S' \rangle\!\rangle_n \\
\langle\!\langle \langle M \rangle\ .\ P \rangle\!\rangle_n & \triangleq & w[\,\mathtt{in}\ r\ .\ \overline{\mathtt{open}} \mid c[\,\mathtt{out}\ r\ .\ \overline{\mathtt{open}}\ .\ \langle\!\langle P \rangle\!\rangle] \\
 & & \mid w_1[\,\mathtt{in}\ r_1\ .\ \overline{\mathtt{open}}\ .\ (\mathbf{fwd}\ \ M' \mid \mathbf{allowIO})]\,] \\
\langle\!\langle (x)\ .\ P \rangle\!\rangle_n & \triangleq & r[\,\overline{\mathtt{in}}\ w\ .\ \mathtt{out}\ n\ .\ \overline{\mathtt{out}}\ c\ .\ \mathtt{open}\ w\ .\ \overline{\mathtt{out}}\ c\ .\ \mathtt{open}\ r_2\ .\ \overline{\mathtt{open}} \\
 & & \mid (\nu x : \underline{\vee}^{\omega})(c[\,\mathtt{out}\ r\ .\ \overline{\mathtt{open}}\ .\ \langle\!\langle P \rangle\!\rangle] \mid r_1[\,\overline{\mathtt{in}}\ w_1\ .\ \mathtt{in}\ x\ .\ \overline{\mathtt{open}}] \\
 & & \mid x[\,\overline{\mathtt{in}}\ r_1\ .\ \overline{\mathtt{out}}\ r_2\ .\ \mathtt{open}\ r_1\ .\ \mathtt{open}\ w_1 \mid r_2[\,\mathtt{out}\ x\ .\ \overline{\mathtt{open}}]\,]\,)] \\
\langle\!\langle \sigma,\ P \rangle\!\rangle & \triangleq & \langle\!\langle \sigma \rangle\!\rangle \mid \langle\!\langle P \rangle\!\rangle \\
\langle\!\langle M_1/x_1 \uplus \cdots \uplus M_k/x_k \rangle\!\rangle & \triangleq & x_1[\mathbf{fwd}\ \ M_1 \mid \mathbf{allowIO}] \mid \cdots \mid x_k[\mathbf{fwd}\ \ M_k \mid \mathbf{allowIO}]
\end{array}
$$

§7.3.2 类型方案

上述翻译方案与文献 [18] 中的翻译方案的最大不同是移动进程的单线程性。除了通道灰箱和变量灰箱由于其非移动性，其中的进程为多线程外，其它灰箱及进程都具有单线程特性。这就使得翻译结果正确性的证明可以完全利用前文的进程等价定律完成。

为了利用基于单线程的等价性定律，我们给出翻译过程的一个单线程类型方案。令：

$$
\begin{array}{lcl}
\Gamma_{SYS} & \triangleq & enter : \curvearrowright^1 [\curvearrowright^1 [\underline{\vee}^0]],\ r : \curvearrowright^1 [\underline{\vee}^0],\ r_1 : \curvearrowright^1 [\underline{\vee}^0],\ r_2 : \curvearrowright^1 [\underline{\vee}^0] \\
 & & w : \curvearrowright^1 [\underline{\vee}^0],\ w_1 : \curvearrowright^1 [\underline{\vee}^{\omega}],\ c : \curvearrowright^1 [\underline{\vee}^0]
\end{array}
$$

同时，令：

$$
\begin{array}{lcl}
Env(\sigma) & \triangleq & \{x : \underline{\vee}^{\omega} \mid x \in dom(\sigma)\} \\
Env(P) & \triangleq & \{M : \underline{\vee}^{\omega} \mid M \in fn(P) \cup fv(P)\} \\
Env(\sigma,\ P) & \triangleq & Env(\sigma) \cup Env(P)
\end{array}
$$

此时下面的命题保证了翻译结果的类型在 Γ_{SYS} 所给出的单线程类型方案下是合法的。

命题 7.5 对任意 π_{esc} 进程 P 和树状替换 σ，如果 $\vdash P : Valid$、$cl_\sigma(P) \equiv P$ 且 $fv(P) \subseteq dom(\sigma)$，则 $Env(\sigma,\ P),\ \Gamma_{SYS} \vdash \langle\!\langle \sigma,\ P \rangle\!\rangle : \underline{\vee}^{\omega}$。

证明 对 P 的构成进行归纳。根据 $\langle\!\langle \sigma,\ P \rangle\!\rangle$ 的定义利用类型规则逐一检验易证。 ∎

§7.3.3 进程在翻译环境下的等价性

翻译过程正确性的证明有两层含义：首先，翻译结果可以模拟原进程的所有归约动作（即：$\sigma : P \longrightarrow_{esc} Q \Longrightarrow \langle\!\langle \sigma,\ P \rangle\!\rangle \longrightarrow R$ 且 R 和 $\langle\!\langle \sigma,\ Q \rangle\!\rangle$ 等价）；其次，翻译结果的任何归约动作必须对应某种原进程的归约动作（即：$\langle\!\langle \sigma,\ P \rangle\!\rangle \longrightarrow R \Longrightarrow \sigma : P \longrightarrow Q$ 且 R 和 $\langle\!\langle \sigma,\ Q \rangle\!\rangle$ 等价）。在这里，“R 和 $\langle\!\langle \sigma,\ Q \rangle\!\rangle$ 等价”中的等价性需要建立在某种等价性判定的基础上。基于类型的简单上下文等价性虽然是一种通用的方案，但是对于 π_{esc} 进程的翻译，如果采用这种等价性判定作为基础，则要求翻译的结果中所有的自由名和自由变量（包括 $dom(\sigma)$ 中的变量）都必须约束起来，否则原来 π_{esc} 进程的合法性和 σ 中关于变量的替换值的正确性就无法得到保证。例如，假设翻译结果 P 中存在灰箱 n 用于代表一个 π_{esc} 中的显式通道，由于 π_{esc} 进程的合法性，翻译结果本身不可能存在两个名为 n 的灰箱，但是当把进程 P 放到任意的上下文中时，上下文中就可能存在另一个名 n 的灰箱，这样原来通道的唯一性就被破坏，归约可能产生意想不到的结果，进程的等价性无法得到体现。

对于上述问题，一种解决方法是在最终的翻译结果外部加上所有自由通道名和自由变量的约束构造，保证外界不会存在与之同名的灰箱。但这样一来翻译结果失去了其组合性，在证明中就不便使用归纳法了。

基于上述原因，在下文对翻译结果正确性的证明中，重新定义了一个进程间特殊的等价性关系：进程在翻译环境下的等价性。首先我们定义一类进程，它们必须是某个翻译结果的组成部分。

定义 7.6 （**翻译环境子进程**） *ROAM 进程 P 称为一个翻译环境子进程，如果存在简单上下文 H 、树状替换 σ 和 π_{esc} 进程 R 满足 $\langle\!\langle \sigma,\ R \rangle\!\rangle \longrightarrow^* H(P)$ 。记为 $\mathbf{PTC} \vdash P$ 。*

对于翻译环境子进程 P ，扩展类型环境 Γ_{SYS} 使其包含 P 中代表通道的自由灰箱名，则在此扩展的类型环境下， P 具有合法的类型。为此，定义针对 P 所做的扩展类型环境如下：

$$Env_{\rho}(P) \quad \triangleq \quad \{n : \underline{\vee}^{\omega} \mid n \in fn(P) \backslash \{enter, r, r_1, r_2, w, w_1, c\}\}$$

引理 7.7 *如果 $\mathbf{PTC} \vdash P$ ，那么 $Env_{\rho}(P),\ \Gamma_{SYS} \vdash P : T$ 。*

证明 由 $\mathbf{PTC} \vdash P$ 的定义可知存在 H 、σ 和 R 满足 $\langle\!\langle \sigma,\ R \rangle\!\rangle \longrightarrow^* H(P)$ 。取 $\Gamma = Env(\sigma,\ R)$ ，由命题 7.5 可知 $\Gamma,\ \Gamma_{SYS} \vdash \langle\!\langle \sigma,\ R \rangle\!\rangle : \underline{\vee}^{\omega}$ ，再由 $\langle\!\langle \sigma,\ R \rangle\!\rangle \longrightarrow^* H(P)$ 利用定理 3.9 可得 $\Gamma,\ \Gamma_{SYS} \vdash H(P) : \underline{\vee}^{\omega}$ ，因此必有 $\Gamma,\ \Gamma_{SYS} \vdash P : T$ 。显然 $Env_{\rho}(P) \subseteq \Gamma$ 且 $fn(P) \subseteq dom(Env_{\rho}(P),\ \Gamma_{SYS})$ 。因此使用与 $\Gamma,\ \Gamma_{SYS} \vdash P : T$ 完全相同的类型推导步骤即可得到 $Env_{\rho}(P),\ \Gamma_{SYS} \vdash P : T$ 。 ∎

基于翻译环境子进程的定义，我们通过限定等价性判断使用的上下文，定义进程在 π 翻译环境下的等价性如下。

定义 7.8 *ROAM 进程在 π 翻译环境下的等价： $\mathbf{PTC} \triangleright P \simeq Q \iff \mathbf{PTC} \vdash P$ 并且对任意 H 和灰箱名 n ，如果 $\mathbf{PTC} \vdash H(P)$ ，那么 $H(P) \Downarrow_n \iff H(Q) \Downarrow_n$ 。*

上述等价性显然包含在前文定义的具有类型限定的等价关系中。

命题 7.9 如果 $\mathbf{PTC} \vdash P$ 且 $Env_\rho(P),\ \Gamma_{SYS} \triangleright P \simeq Q : T$，那么 $\mathbf{PTC} \triangleright P \simeq Q$。

证明 对任意 H 和灰箱名 n 满足 $\mathbf{PTC} \vdash H(P)$，由引理 7.7 可知 $Env_\rho(H(P)),\ \Gamma_{SYS} \vdash H(P) : T'$，而由 $Env_\rho(P) \subseteq Env_\rho(H(P))$ 可知 H 为 $(Env_\rho(P),\ \Gamma_{SYS}/T)$ 型简单上下文，利用条件 $Env_\rho(P),\ \Gamma_{SYS} \triangleright P \simeq Q : T$ 可知，$H(P) \Downarrow_n \iff H(Q) \Downarrow_n$，因此 $\mathbf{PTC} \triangleright P \simeq Q$。 ∎

上述命题说明，第六章中的等价性定律同样适用于 π 翻译环境下的等价性。

§7.3.4 翻译方案的正确性证明

方便书写和阅读，定义如下简写：

$$
\begin{array}{lcl}
\mathbf{wHead} & \triangleq & \mathtt{in}\, r\,.\,\overline{\mathtt{open}} \\
\mathbf{wTail}(M,\ P) & \triangleq & c[\mathtt{out}\, r\,.\,\overline{\mathtt{open}}\,.\,\langle\!\langle P \rangle\!\rangle] \mid w_1[\mathtt{in}\, r_1\,.\,\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})] \\
\mathbf{rHead} & \triangleq & \overline{\mathtt{in}}\, w\,.\,\mathtt{out}\, n\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \\
\mathbf{rTail}(x,\ P) & \triangleq & (\nu x : \underline{\vee}^{\omega})(c[\mathtt{out}\, r\,.\,\overline{\mathtt{open}}\,.\,\langle\!\langle P \rangle\!\rangle] \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}} \\
 & & \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]\,])] \\
\mathbf{Channel} & \triangleq & \mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead}
\end{array}
$$

下面引理中的等价性结果表明了翻译结果在归约时不会产生意外的结果。在这些等价性结果的证明过程中，所涉及进程为翻译结果子进程（$\mathbf{PTC} \vdash P$）这一点很容易通过翻译方案得出，在证明过程中将直接作为已知条件使用。

引理 7.10

(Enter Input)

(1). $\mathbf{PTC} \triangleright n[\mathbf{Channel} \mid r[\overline{\mathtt{in}}\, enter\,.\,\mathtt{open}\, enter \mid \mathbf{rTail}(x,\ P)]]$
$\simeq n[\mathbf{Channel} \mid r[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{rHead}] \mid \mathbf{rTail}(x,\ P)]]$

(2). $\mathbf{PTC} \triangleright n[\mathbf{Channel} \mid r[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{rHead}] \mid \mathbf{rTail}(x,\ P)]]$
$\simeq n[\mathbf{Channel} \mid r[\mathbf{rHead} \mid \mathbf{rTail}(x,\ P)]]$

(Enter Output)

(1). $\mathbf{PTC} \triangleright n[\mathbf{Channel} \mid w[\overline{\mathtt{in}}\, enter\,.\,\mathtt{open}\, enter \mid \mathbf{wTail}(M,\ P)]]$
$\simeq n[\mathbf{Channel} \mid w[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{wHead}] \mid \mathbf{wTail}(M,\ P)]]$

(2). $\mathbf{PTC} \triangleright n[\mathbf{Channel} \mid w[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{wHead}] \mid \mathbf{wTail}(M,\ P)]]$
$\simeq n[\mathbf{Channel} \mid w[\mathbf{wHead} \mid \mathbf{wTail}(M,\ P)]]$

证明 以 **(Enter Input)** 为例（**(Enter Output)** 同理可证）。

首先考察结果 (2)，令 $P = n[\mathbf{Channel} \mid r[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{rHead}] \mid \mathbf{rTail}(x,\ P)]]$，由 $\mathbf{PTC} \vdash P$ 知 **(ST Open 1)** 所需的所有类型条件，包括 $Env_\rho(P),\ \Gamma_{SYS} \vdash P : T$，$r$、$enter$ 为单线程，均可满足，且 $\mathbf{rTail}(x,\ P) \overset{enter^{[\overline{\mathtt{open}}]}}{\not\Longrightarrow}$，因此直接使用 **(ST Open 1)** 可得。

对于结果 (1)，主要是检查等价性定律 **(Rec In)** 的适用性。令 $P = n[\mathbf{Channel} \mid r[\overline{\mathtt{in}}\, enter\,.\,\mathtt{open}\, enter \mid \mathbf{rTail}(x,\ P)]]$，由 $\mathbf{PTC} \vdash P$ 知所有的类型条件，包括 $Env_\rho(P),\ \Gamma_{SYS} \vdash$

$P:T$ 、 n 为非移动和 r 为单线程，均可满足，现在关键是保证对任意给定满足 $\mathbf{PTC} \vdash H(P)$ 的 H ，均不会在 n 中产生干扰 $enter$ 进入 r 的额外的 $enter^{[\mathtt{in}\ r]}$ 进程。

在翻译环境下，考察 n 中可能出现的顶层子灰箱。根据翻译方案的具体规则可知，任何显式通道 n 对应的灰箱中只有两类处于复制构造下的 $enter$ 灰箱和对应输入输出操作的灰箱 r 和 w ，并且 r 和 w 在 n 中只是发生让 $enter$ 进入和让 w 进入 r 的移动操作，不会被打开，也不会释放出新的 $enter$ 灰箱。通过以上分析可知，任何显式通道中的复制构造 $enter[\mathtt{in}\, r\,.\,\overline{\mathtt{open}}\,.\,\mathbf{rHead}]$ 在任何翻译环境子进程中具有定点接收特性，因此可应用定律 **(Rec In)** 得到结论。 ∎

引理 7.11

(Subst Input)

(1). $\mathbf{PTC} \triangleright\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO} \mid r[\overline{\mathtt{in}}\, enter\,.\,\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid \mathbf{rTail}(y,\ P)]]$

$\simeq\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}$

$\mid r[\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{rTail}(y,\ P)]]$

(2). $\mathbf{PTC} \triangleright\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}$

$\mid r[\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{rTail}(y,\ P)]]$

$\simeq\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}] \mid r[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{rTail}(y,\ P)]$

(3). $\mathbf{PTC} \triangleright\ r[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{rTail}(y,\ P)]$

$\simeq\ w[\mathbf{request}\ M \mid \mathbf{rTail}(y,\ P)]$

(Subst Output)

(1). $\mathbf{PTC} \triangleright\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO} \mid w[\overline{\mathtt{in}}\, enter\,.\,\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid \mathbf{wTail}(M',\ P)]]$

$\simeq\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}$

$\mid w[\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{wTail}(M',\ P)]]$

(2). $\mathbf{PTC} \triangleright\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}$

$\mid w[\mathtt{out}\, x\,.\,\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{wTail}(M',\ P)]]$

$\simeq\ x[\mathbf{fwd}\ M \mid \mathbf{allowIO}] \mid w[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{wTail}(M',\ P)]$

(3). $\mathbf{PTC} \triangleright\ w[\mathtt{open}\, enter \mid enter[\overline{\mathtt{open}}\,.\,\mathbf{request}\, M] \mid \mathbf{wTail}(M',\ P)]$

$\simeq\ w[\mathbf{request}\ M \mid \mathbf{wTail}(M',\ P)]$

证明 以 **(Subst Input)** 为例。结论 (1) 、 (3) 的证明过程类似引理 7.10 。结论 (2) 可以使用 **(Rec Co-out)** ，该定律的应用主要难点是动作 $x^{[\overline{\mathtt{out}}\ r]}$ 在任何翻译环境子进程中定点接收特性的保证。分析灰箱 x 内可执行的动作可知， x 自身可能激发的顶层动作只有两类： $\overline{\mathtt{in}}$ 和 $\overline{\mathtt{out}}$ 。由于没有 $\mathtt{open}$ 动作的存在，所以 x 在归约过程中也不可能引入任何新的顶层动作，也即不可能出现新的 $\overline{\mathtt{out}}\ r\,.\,P$ 干扰原 $!\overline{\mathtt{out}}\ r$ 的归约，因此上述的定点接收条件可以被保证。 ∎

引理 7.12

(Communicate)

(1). $\mathbf{PTC} \triangleright\ n[\mathbf{Channel} \mid \langle\!\langle S \rangle\!\rangle \mid r[\mathtt{out}\, n\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}}$

$\mid w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)] \mid \mathbf{rTail}(x,\ P_2)]]$

$\simeq\ n[\mathbf{Channel} \mid \langle\!\langle S \rangle\!\rangle] \mid r[\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}}$

$\mid w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)] \mid \mathbf{rTail}(x,\ P_2)]$

$$
\begin{aligned}
&(2).\ \mathbf{PTC} \rhd r[\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)] \mid \mathbf{rTail}(x,\ P_2)]\\
&\quad \simeq (\nu x : \underline{\vee}^{\omega})(r[\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)]\\
&\qquad \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}]\\
&\qquad \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\qquad \mid c[\overline{\mathtt{open}}\,.\,\langle\!\langle P_2 \rangle\!\rangle])\\
&(3).\ \mathbf{PTC} \rhd r[\mathtt{open}\, w\,.\,\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)]\\
&\qquad \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}] \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\quad \simeq r[\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1) \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}]\\
&\qquad \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&(4).\ \mathbf{PTC} \rhd r[\overline{\mathtt{out}}\, c\,.\,\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1) \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}]\\
&\qquad \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\quad \simeq r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid w_1[\mathtt{in}\, r_1\,.\,\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]\\
&\qquad \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}] \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\qquad \mid c[\overline{\mathtt{open}}\,.\,\langle\!\langle P_1 \rangle\!\rangle]\\
&(5).\ \mathbf{PTC} \rhd r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid w_1[\mathtt{in}\, r_1\,.\,\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]\\
&\qquad \mid r_1[\overline{\mathtt{in}}\, w_1\,.\,\mathtt{in}\, x\,.\,\overline{\mathtt{open}}] \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\quad \simeq r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid r_1[\mathtt{in}\, x\,.\,\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]\\
&\qquad \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&(6).\ \mathbf{PTC} \rhd r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid r_1[\mathtt{in}\, x\,.\,\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]\\
&\qquad \mid x[\overline{\mathtt{in}}\, r_1\,.\,\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\quad \simeq r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid x[\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1\\
&\qquad \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]] \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&(7).\ \mathbf{PTC} \rhd r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid x[\overline{\mathtt{out}}\, r_2\,.\,\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1\\
&\qquad \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]] \mid r_2[\mathtt{out}\, x\,.\,\overline{\mathtt{open}}]]\\
&\quad \simeq r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid r_2[\overline{\mathtt{open}}]\\
&\qquad \mid x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]]\\
&(8).\ \mathbf{PTC} \rhd r[\mathtt{open}\, r_2\,.\,\overline{\mathtt{open}} \mid r_2[\overline{\mathtt{open}}]\\
&\qquad \mid x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]]\\
&\quad \simeq r[\overline{\mathtt{open}} \mid x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]]\\
&(9).\ \mathbf{PTC} \rhd \mathtt{open}\, r \mid r[\overline{\mathtt{open}} \mid x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]]\\
&\quad \simeq x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]\\
&(10).\ \mathbf{PTC} \rhd x[\mathtt{open}\, r_1\,.\,\mathtt{open}\, w_1 \mid r_1[\overline{\mathtt{open}} \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]]\\
&\quad \simeq x[\mathtt{open}\, w_1 \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]\\
&(11).\ \mathbf{PTC} \rhd x[\mathtt{open}\, w_1 \mid w_1[\overline{\mathtt{open}}\,.\,(\mathbf{fwd}\ \ M \mid \mathbf{allowIO})]]\\
&\quad \simeq x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}]\\
&(12).\ \mathbf{PTC} \rhd \mathtt{open}\, c \mid c[\overline{\mathtt{open}}\,.\,\langle\!\langle P \rangle\!\rangle\\
&\quad \simeq \langle\!\langle P \rangle\!\rangle
\end{aligned}
$$

证明 证明过程类似引理 7.10 、 7.11 ，简单分析如下：

(1). 使用 **(Rec Co-out)**，类似引理 7.11 的结论 (2)；
(2). 使用 **(ST Out)**；
(3). 使用 **(ST Open 1)**；
(4). 使用 **(ST Out)**；
(5). 使用 **(ST In)**，显然 r 中不会出现第二个 w_1 或 r_1 灰箱；
(6). 使用 **(ST Aux In)**，第三种情况；
(7). 使用 **(ST Out)**；
(8). 使用 **(ST Open 1)**；
(9). 使用 **(Rec Open)**，在翻译环境中，$\mathtt{open}\ r$ 动作只可能以 $\mathtt{open}\ r\,.\,\mathbf{0}$ 的方式出现在最外层，因此可以看作是一种特殊的有限个数的定点接收情况（且服务的个数严格等于客户的个数），此外，不会有其它形式的 $\mathtt{open}\ r$ 出现；
(10). 使用 **(ST Open 1)**；
(11). 使用 **(ST Open 1)**；
(12). 使用 **(Rec Open)**，类似情况 9 。

∎

在上述引理的基础上，下面的命题 7.13 、 7.14 说明了上述 ROAM 演算对 π_{esc} 演算翻译方案的操作一致性（ operational correspondence ）。

命题 7.13 如果 $\sigma : P \longrightarrow_{esc} Q$，那么 $\langle\!\langle \sigma,\ P \rangle\!\rangle \longrightarrow^* R$ 且 $R \equiv \langle\!\langle \sigma,\ Q \rangle\!\rangle$ 。

证明 对 $\sigma : P \longrightarrow_{esc} Q$ 的推导过程进行归纳。

(π_{esc} **Red Subst Out**) 此时有 $P = \overline{x}\langle M' \rangle\,.\,P_1$ 、 $Q = \overline{M}\langle M' \rangle\,.\,P_1$ 且 $x\sigma = M$ 。由 $x\sigma = M$ 可知 $\sigma = \{M/x\} \uplus \sigma'$，因此 $\langle\!\langle \sigma,\ P \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid w[\mathbf{request}\ x \mid \mathbf{wTail}(M',\ P_1)] \equiv \langle\!\langle \sigma' \rangle\!\rangle \mid x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ x \mid \mathbf{wTail}(M',\ P_1)] \longrightarrow^* \langle\!\langle \sigma' \rangle\!\rangle \mid x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ M \mid \mathbf{wTail}(M',\ P_1)] \equiv \langle\!\langle \sigma,\ Q \rangle\!\rangle$ 。

(π_{esc} **Red Subst In**) 类似 (π_{esc} **Red Subst Out**) 可证。

(π_{esc} **Red Output**) 此时 $P = [n : S] \mid \overline{n}\langle M \rangle\,.\,P_1$ 且 $Q = [n : S \mid \langle M \rangle\,.\,P_1]$ 。则 $\langle\!\langle \sigma,\ P \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S \rangle\!\rangle] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \mid w[\mathbf{request}\ n \mid \mathbf{wTail}(M,\ P_1)]$(其中 k 为 S 中读进程的个数)，所以 $\langle\!\langle \sigma,\ P \rangle\!\rangle \longrightarrow^* \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S \rangle\!\rangle \mid w[\mathbf{wHead} \mid \mathbf{wTail}(M,\ P_1)]] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \equiv \langle\!\langle \sigma,\ Q \rangle\!\rangle$ （最后一步需要有此条件的支持： $S \mid \langle M \rangle\,.\,P_1$ 中读进程的个数仍为 k）。

(π_{esc} **Red Input**) 类似 (π_{esc} **Red Output**) 可证。

(π_{esc} **Red Comm**) 此时 $P = [n : S \mid \langle M \rangle\,.\,P_1 \mid (x)\,.\,P_2]$ 且 $Q = [n : S] \mid P_1 \mid (\nu x : M)P_2$ 。则 $\langle\!\langle \sigma,\ P \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S \rangle\!\rangle \mid w[\mathbf{wHead} \mid \mathbf{wTail}(M,\ P_1)] \mid r[\mathbf{rHead} \mid \mathbf{rTail}(x,\ P_2)]] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \longrightarrow^* \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S \rangle\!\rangle] \mid \langle\!\langle P_1 \rangle\!\rangle \mid (\nu x : \underline{\vee}^{\omega})(x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \langle\!\langle P_2 \rangle\!\rangle) \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^{k-1} \equiv \langle\!\langle \sigma,\ Q \rangle\!\rangle$

(π_{esc} **Red Var**) 易证。

(π_{esc} **Red Res**) 易证。

(π_{esc} **Red Par**) 易证。

(π_{esc} **Red Struct**) 基于 $P \equiv_{esc} Q \Longrightarrow \langle\!\langle \sigma,\ P \rangle\!\rangle \equiv \langle\!\langle \sigma,\ Q \rangle\!\rangle$ 易证。

∎

命题 7.14 如果 $\langle\!\langle \sigma,\ P \rangle\!\rangle \longrightarrow Q$，则存在 π_{esc} 进程 R 满足 $\sigma : P \longrightarrow_{esc} R$ 且 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R \rangle\!\rangle \simeq Q$。

证明 对 P 的构成进行归纳。

- $P = \mathbf{0}$：此时 $\langle\!\langle \sigma,\ \mathbf{0} \rangle\!\rangle \equiv \langle\!\langle \sigma \rangle\!\rangle$ 不可能再进行归约。
- $P = (\nu n)P_1$：此时有 $\langle\!\langle \sigma,\ (\nu n)P_1 \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid (\nu n : \underline{\vee}^{\omega})\langle\!\langle P_1 \rangle\!\rangle \equiv (\nu n : \underline{\vee}^{\omega})(\langle\!\langle \sigma \rangle\!\rangle \mid \langle\!\langle P_1 \rangle\!\rangle) = (\nu n : \underline{\vee}^{\omega})(\langle\!\langle \sigma,\ P_1 \rangle\!\rangle) \longrightarrow Q$，因此必有 $\langle\!\langle \sigma,\ P_1 \rangle\!\rangle \longrightarrow Q_1$ 且 $Q \equiv (\nu n : \underline{\vee}^{\omega})Q_1$。由归纳假设可知，存在 R_1 满足 $\sigma : P_1 \longrightarrow_{esc} R_1$ 且 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R_1 \rangle\!\rangle \simeq Q_1$。令 $R = (\nu n)R_1$，则易知有 $\sigma : P \longrightarrow_{esc} R$。同时由 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R_1 \rangle\!\rangle \simeq Q_1$ 可知 $\mathbf{PTC} \triangleright (\nu n : \underline{\vee}^{\omega})\langle\!\langle \sigma,\ R_1 \rangle\!\rangle \simeq (\nu n : \underline{\vee}^{\omega})Q_1$。因此由 $\langle\!\langle \sigma,\ R \rangle\!\rangle \equiv (\nu n : \underline{\vee}^{\omega})\langle\!\langle \sigma,\ R_1 \rangle\!\rangle$ 和 $Q \equiv (\nu n : \underline{\vee}^{\omega})Q_1$，即得 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R \rangle\!\rangle \simeq Q$。
- $P = (\nu x : M)P_1$：此时有 $\langle\!\langle \sigma,\ (\nu x : M)P_1 \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid (\nu x : \underline{\vee}^{\omega})(x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \langle\!\langle P_1 \rangle\!\rangle) \equiv (\nu x : \underline{\vee}^{\omega})(\langle\!\langle \sigma \rangle\!\rangle \mid x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \langle\!\langle P_1 \rangle\!\rangle) = (\nu x : \underline{\vee}^{\omega})(\langle\!\langle \sigma \uplus \{M/x\},\ P_1 \rangle\!\rangle) \longrightarrow Q$，利用归纳假设类似上一种情况 ($P = (\nu n)P_1$) 可证。
- $P = [n : S]$：此时有 $\langle\!\langle \sigma,\ [n : S] \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S \rangle\!\rangle] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k \longrightarrow Q$。此时，只可能有 $S \equiv \langle M \rangle\,.\,P_1 \mid (x)\,.\,P_2 \mid S_1$ 且 $Q \equiv \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S_1 \rangle\!\rangle \mid r[w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)] \mid \mathtt{out}\ n\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ w\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ r_1\,.\,\overline{\mathtt{open}} \mid \mathbf{rTail}(x,\ P_2)]] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k$。令 $R = [n : S_1] \mid P_1 \mid (\nu x : M)P_2$，则显然有 $\sigma : P \longrightarrow_{esc} R$。同时利用引理 7.12 中的结论可知 $\mathbf{PTC} \triangleright \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S_1 \rangle\!\rangle] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^{k-1} \mid \langle\!\langle P_1 \rangle\!\rangle \mid (\nu x : \underline{\vee}^{\omega})(x[\mathbf{fwd}\ \ M \mid \mathbf{allowIO}] \mid \langle\!\langle P_2 \rangle\!\rangle) \simeq \langle\!\langle \sigma \rangle\!\rangle \mid n[\mathbf{allowIO} \mid \mathbf{server}\ w\,.\,\mathbf{wHead} \mid \mathbf{server}\ r\,.\,\mathbf{rHead} \mid \langle\!\langle S_1 \rangle\!\rangle \mid r[w[\overline{\mathtt{open}} \mid \mathbf{wTail}(M,\ P_1)] \mid \mathtt{out}\ n\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ w\,.\,\overline{\mathtt{out}}\ c\,.\,\mathtt{open}\ r_1\,.\,\overline{\mathtt{open}} \mid \mathbf{rTail}(x,\ P_2)]] \mid (\mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r)^k$，即 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R \rangle\!\rangle \simeq Q$。
- $P = \overline{M}\langle M' \rangle\,.\,P_1$：此时有 $\langle\!\langle \sigma,\ \overline{M}\langle M' \rangle\,.\,P_1 \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid w[\mathbf{request}\ M \mid \mathbf{wTail}(M',\ P_1)] \longrightarrow Q$。只可能有 $\sigma = \sigma_1 \uplus \{M_1/M\}$ 且 $Q \equiv \langle\!\langle \sigma_1 \rangle\!\rangle \mid M[\mathbf{fwd}\ \ M_1 \mid \mathbf{allowIO} \mid w[\overline{\mathtt{in}}\ enter\,.\,\mathtt{out}\ M\,.\,\mathtt{open}\ enter \mid \mathbf{wTail}(M',\ P_1)]]$。令 $R = \overline{M_1}\langle M' \rangle\,.\,P$，首先由 $\sigma M = M_1$ 可知 $\sigma : P \longrightarrow_{esc} R$，同时由引理 7.11 中的结论可知 $\mathbf{PTC} \triangleright \langle\!\langle \sigma_1 \rangle\!\rangle \mid M[\mathbf{fwd}\ \ M_1 \mid \mathbf{allowIO}] \mid w[\mathbf{request}\ M_1 \mid \mathbf{wTail}(M',\ P_1)] \simeq \langle\!\langle \sigma_1 \rangle\!\rangle \mid M[\mathbf{fwd}\ \ M_1 \mid \mathbf{allowIO} \mid w[\overline{\mathtt{in}}\ enter\,.\,\mathtt{out}\ M\,.\,\mathtt{open}\ enter \mid \mathbf{wTail}(M',\ P_1)]]$，即 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R \rangle\!\rangle \simeq Q$。
- $P = M(x)\,.\,P_1$：类似上一种情况可证。
- $P = P_1 \mid P_2$：此时有 $\langle\!\langle \sigma,\ P_1 \mid P_2 \rangle\!\rangle = \langle\!\langle \sigma \rangle\!\rangle \mid \langle\!\langle P_1 \rangle\!\rangle \mid \langle\!\langle P_2 \rangle\!\rangle \longrightarrow Q$，分以下情况讨论：
  - $\langle\!\langle \sigma \rangle\!\rangle \mid \langle\!\langle P_1 \rangle\!\rangle \longrightarrow Q_1$ 且 $Q \equiv Q_1 \mid \langle\!\langle P_2 \rangle\!\rangle$：此时，利用归纳假设易证。
  - $\langle\!\langle \sigma \rangle\!\rangle \mid \langle\!\langle P_2 \rangle\!\rangle \longrightarrow Q_2$ 且 $Q \equiv \langle\!\langle P_1 \rangle\!\rangle \mid Q_2$：类似可证。
  - $\langle\!\langle P_1 \rangle\!\rangle \mid \langle\!\langle P_2 \rangle\!\rangle \longrightarrow Q_3$ 且 $Q \equiv \langle\!\langle \sigma \rangle\!\rangle \mid Q_3$：此时又可分以下情况讨论：

(1). $P_1 \mid P_2 \equiv [n : S] \mid \overline{n}\langle M\rangle . P_3 \mid P_4$ 且 $Q_3 \equiv n[\mathbf{Channel} \mid w[\overline{\mathtt{in}}\ enter . \mathtt{open}\ enter \mid \mathbf{wTail}(M,\ P_3)]] \mid \langle\!\langle P_4\rangle\!\rangle$；此时令 $R = [n : S \mid \langle M\rangle . P_3] \mid P_4$，由引理 7.10 易知 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R\rangle\!\rangle \simeq \langle\!\langle \sigma\rangle\!\rangle \mid Q_3$，即 $\mathbf{PTC} \triangleright \langle\!\langle \sigma,\ R\rangle\!\rangle \simeq Q$。

(2). $P_1 \mid P_2 \equiv [n : S] \mid n(x) . P_3 \mid P_4$ 且 $Q_3 \equiv n[\mathbf{Channel} \mid r[\overline{\mathtt{in}}\ enter . \mathtt{open}\ enter \mid \mathbf{rTail}(x,\ P_3)]] \mid \langle\!\langle P_4\rangle\!\rangle$；类似上一种情况可证。

(3). $P_1 \mid P_2 \equiv (\nu n)P_3$：此时可归入对情况 $P = (\nu n)P_1$ 的证明。

(4). $P_1 \mid P_2 \equiv (\nu x : M)P_3$：此时可归入对情况 $P = (\nu x : M)P_1$ 的证明。

- $P = !P_1$：此时有 $\langle\!\langle \sigma,\ !P\rangle\!\rangle = \langle\!\langle \sigma\rangle\!\rangle \mid !\langle\!\langle P_1\rangle\!\rangle \longrightarrow Q$。这只可能有 $\langle\!\langle \sigma\rangle\!\rangle \mid \langle\!\langle P_1\rangle\!\rangle \longrightarrow Q_1$ 且 $Q \equiv Q_1 \mid !\langle\!\langle P_1\rangle\!\rangle$，利用归纳假设易证。

通过以上对所有情况的分析可知，结论成立。 ∎

§7.4 π 演算的翻译

在上节给出的 π_{esc} 演算翻译的基础上， ROAM 对 π 演算的翻译方案可直接定义为：

$$\{\{P\}\} = \langle\!\langle \emptyset,\ cl_{\emptyset}(P)\rangle\!\rangle$$

具体来说，对 π 进程 P 的翻译操作 $\{\{P\}\}$ 定义如下（下面的翻译只针对于没有自由通道的 π 进程，否则，需要在最终的结果中为每个自由通道 m 增加一个并置的 $m[\mathbf{Channel}]$ 进程）：

$$\begin{array}{lcl}
\{\{\mathbf{0}\}\} & \triangleq & \mathbf{0} \\
\{\{P \mid Q\}\} & \triangleq & \{\{P\}\} \mid \{\{Q\}\} \\
\{\{\ !P\}\} & \triangleq & \ !\{\{P\}\} \\
\{\{(\nu n)P\}\} & \triangleq & (\nu n : \underline{\vee}^{\omega})(n[\mathbf{Channel}] \mid \{\{P\}\}) \\
\{\{\overline{M}\langle M'\rangle . P\}\} & \triangleq & w[\mathbf{request}\ M \mid c[\mathtt{out}\ r . \overline{\mathtt{open}} . \{\{P\}\}] \\
 & & \mid w_1[\mathtt{in}\ r_1 . \overline{\mathtt{open}} . (\mathbf{fwd}\ \ M' \mid \mathbf{allowIO})]] \\
\{\{M(x) . P\}\} & \triangleq & r[\mathbf{request}\ M \mid (\nu x : \underline{\vee}^{\omega})(c[\mathtt{out}\ r . \overline{\mathtt{open}} . \{\{P\}\}] \\
 & & \mid r_1[\overline{\mathtt{in}}\ w_1 . \mathtt{in}\ x . \overline{\mathtt{open}}] \\
 & & \mid x[\overline{\mathtt{in}}\ r_1 . \overline{\mathtt{out}}\ r_2 . \mathtt{open}\ r_1 . \mathtt{open}\ w_1 \mid r_2[\mathtt{out}\ x . \overline{\mathtt{open}}]])] \\
 & & \mid \mathtt{open}\ c \mid \mathtt{open}\ c \mid \mathtt{open}\ r \\
\mathbf{Channel} & \triangleq & \mathbf{allowIO} \mid \mathbf{server}\ w . \mathtt{in}\ r . \overline{\mathtt{open}} \\
 & & \mid \mathbf{server}\ r . \overline{\mathtt{in}}\ w . \mathtt{out}\ n . \overline{\mathtt{out}}\ c . \mathtt{open}\ w . \overline{\mathtt{out}}\ c . \mathtt{open}\ r_2 . \overline{\mathtt{open}}
\end{array}$$

由于 π_{esc} 演算和 π 演算之间可以相互翻译 [18] 的特殊关系，上述翻译的正确性很容易得到验证。

本章小结

本章在上文对演化类型系统研究和 ROAM 中等价性定律研究的基础上，给出了用纯 ROAM 演算对同步 π 演算的一个单线程翻译方案，对该翻译方案的证明过程中利用了 ROAM 进程的等

价性定律。这不但说明了 ROAM 演算具有类似安全灰箱演算的表达能力，而且说明了演化类型系统在精确刻划进程类型中的突出作用，同时体现了进程等价性定律在进程等价性证明中的作用。

本章的结论说明，由于灰箱演算可以用更加基本的三个局部性动作来表达 π 演算中最基本的本地通讯原语，因此可以认为，灰箱演算具有比基于名字通讯的系列演算具有更加原子的构造。

第八章　结语

§8.1　论文工作的创新性

本论文针对灰箱演算的一个变体—— ROAM 做了较为深入的研究和探索。在研究过程中，除了将灰箱演算中已有的理论和工具运用到 ROAM 中进行证实之外，对以下一些方面还作出了创新性的贡献。

- **提出并研究了灰箱演算的一个变体—— ROAM**
  本文对利用协动作参数加强灰箱演算的安全性这一命题进行了研究，提出了灰箱演算的变体 ROAM 。本文的研究结论初步表明，通过利用协动作参数加强交互双方的彼此控制， ROAM 在安全控制方面比起 SA 来有一定优势。同时，对协动作参数进行控制后， ROAM 并没有丧失 SA 所具有的表达能力。
- **提出了灰箱演算的演化类型系统**
  在众多灰箱系统类型问题的研究工作中，类型的演化问题一直没有得到妥善地解决，灰箱在被打开前后的类型变化得不到体现。 Cardelli 等人 [28] 引入了被动移动构造及其类型描述，但是被动移动本身具有很多副作用 [5] ，因此价值不大。文献 [32] 中也针对该问题，提出了一种解决方法，但是遗憾的该类型系统只允许同层进程中包含一个 $\overline{\texttt{open}}$ 动作。本文的类型演化系统可以说是首次专门针对这个问题提出了一套完整的解决方法，并且在设计时考虑到类型系统中所刻划属性的增减，专门使用一个准类型集合进行表示。刻划不同的属性，可以有不同的准类型集合。

  在本文的演化类型系统 ETS-MT中，对线程数的精确刻划也是特色之一。在灰箱演算进程的代数性质研究中，进程和灰箱的线程数限制显得尤为重要，在 SA 演算的代数性质研究中 [16] ，就首次引入了单线程的概念。本文的前期工作 [58] 以及对 ETS-MT的研究是对灰箱演算类型系统研究中线程数类型属性的进一步完善，首次提出了具有子类型特性（ $0 \leq 1 \leq \omega$ ）的完善的线程数刻划体系。

  这里需要提到与本文 ETS-MT研究类似的 Amtoft 等人的多态类型研究工作 [54] 。在文献 [54] 中， Amtoft 等人提出了多种消息通讯类型的多态性，其中包括通讯类型随灰箱被打开而发生的变化问题。他们的解决方法可以看作是对消息通讯类型多态性的研究成果，而本文的研究着重于灰箱演算中固有的移动性和线程数的类型演化问题。虽然 Amtoft 等人的工作更早，但是本文的工作没有受其任何影响。本文在类型演化问题的整体解决方案上和在对移动性和线程数的刻划上作出了创新性的工作。虽然本文给出的 ETS-MT是专门为 ROAM 设计的，但是其中包含的演化类型思想和移动性、线程数两个概念在各种灰箱演算体系中都可以得到应用。

  在最新的文献 [33] 中， Amtoft 提出了一种对灰箱演算每个能力执行过程进行跟踪的类型系统，可以看作是对灰箱演算类型系统的又一个重要进步。在该类型系统中，无需通过协动作给出明显的线索，即可跟踪灰箱的移动和打开。其缺点是这种跟踪需要建立很多辅助的信息之上的。 Amtoft 在文献 [33] 中对我们的工作也给予了肯定。

- **对 ROAM 中类型、代数性质和表达能力等理论的综合与贯通**

  本文对 ROAM 的标号转移语义、等价性判定、等价性定律、π 演算翻译等方面的工作都不是独创性的，对标号转移语义的研究和进程等价性判断的研究采用了文献 [20] 中的方法，对等价性定律的研究受到文献 [16] 的启发，对 π 演算的翻译使用了文献 [18] 中的方法。但是本文将以前在这些方面的这些孤立研究贯穿起来，将它们融合在一个整体中，集中表现在以下两点。

  （一）用基于硬化关系所得的简单上下文等价性一般判定方法证明进程的等价性定律。

  文献 [20] 中关于 MA 进程的等价性研究止于给出了基于简单上下文等价的进程等价性一般判定方法（ activity lemma ），对其应用仅仅是证明了类似防火墙跨越的几个例子。文献 [16] 中对进程等价性定律的证明采用基于互模拟的方法。本文中基于文献 [20] 的方法证明进程的等价性定律尚属首次。

  （二）用进程的等价性定律完成对 π 演算翻译的代数方法证明。

  文献 [16] 中的等价性定律采用的单线程和非移动类型并不支持类型的演化，这就使得 [16] 中对等价性定律的应用止于证明带本地通讯原语的 SA 演算对 π 演算的翻译。并且使得文献 [18] 中用纯 SA 翻译 π 演算的方案只能使用比较复杂的非代数方法进行。本文结合演化类型系统 ETS-MT中对单线程和非移动进程的更精确刻划，使得进程的等价性定律适用范围更加广泛。其直接后果为对 π 演算翻译的代数方法证明。

另外，本人在博士论文期间收集整理的关于灰箱演算的参考文献和研究人员的网页 [60] 已经被许多研究人员（如 Luca Cardelli[1]， Gérard Boudol[2] ）和著名的研究资料网站（如： Calculi for Mobile Processes[3] ）链接。

§8.2 进一步的工作

本文对 ROAM 演算的研究只是在寻找适合于广域网络移动计算模型的形式化描述工作中的一小步。进一步的研究工作可以在以下一些方面展开.

- **ROAM 演算的扩展**

  本文研究没有本地通讯原语的纯 ROAM 演算，在其基础上可以进行各种扩展，使其内容更加丰富。例如：增加本地通讯原语、仿照 Amtoft 等人增加函数语言的抽象和应用两个基本构造等，并利用本文的研究成果和经验，完成该扩展演算的类型系统、代数性质等研究工作。

- **ROAM 类型系统在安全性上的扩展和分布式类型验证的研究与算法实现**

  对广域网移动计算模型的研究归根到底要解决安全性问题。本文中 ROAM 的演化类型系统主要针对 ROAM 的代数性质研究。针对安全性问题，可以在 ROAM 的演化类型系统中增加组的概念和组间安全性的包含偏序关系和打开偏序关系，以此控制整个移动计算系统内的安全性不被破坏。对该类型系统的研究重点在于类型的分布式验证。由于在分布式移动系统很难做到对分布在各节点的子系统进行集中的类型验证，这种类型验证只能由单个节点根据其所了解的局部信息进行。其难点就在于对网络中前来访问的灰箱的类型检验。如何得到该灰箱

[1] http://www.luca.demon.co.uk/

[2] http://www-sop.inria.fr/mimosa/personnel/Gerard.Boudol.html

[3] http://lampwww.epfl.ch/mobility/

的类型信息？如何保证该信息的正确性？不同节点的类型信息如何进行交流？这些问题都是在一个实用的移动计算模型中需要回答的。

- **对 MA 、 SA 、 ROAM 和 SAP 等多种演算在安全性、易用性、代数性质和表达能力方面的比较**

  灰箱体系结构的精巧构造和强大的描述能力使得这方面的研究工作异常活跃。在对其协动作的研究方面就有最初的 MA 、 SA 到本文的 ROAM 及最新的 SAP 。对灰箱演算的研究最终将走向何方？移动计算的核心模型到底是不是灰箱演算？这些问题也许可以从这些演算在安全性、易用性、代数性质和表达能力各方面的比较得到一些回答。

参考文献

[1] R. S. Gray. Agent Tcl: A flexible and secure mobile agent system. In Proc. of the Fourth Annual Tcl/Tk Workshop, Monterey, July 1996, pages 9-13, Berkeley, 1996. Also available as Dartmouth Computer Science Technical Report TR98-327.

[2] M. Strafier, J. Baumann, and F. Hohl. Mole - A Java Based Mobile Agent System. In: M. Muhlhauser: (ed.), Special Issues in Object Oriented Programming, pp. 301-308, Springer-Verlag 1997.

[3] Aglets.org: `http://www.aglets.org/`

[4] A. Tripathi, N. Karnik, R. Singh, T. Ahmed, J. Eberhard, and A. Prakash. Development of Mobile Agent Applications in Ajanta. Technical report, Department of Computer Science, University of Minnesota, May 1999.

[5] L. Cardelli and A. D. Gordon. Mobile ambients. In M. Nivat(ed.), Proc. FOSSACS'98, International Conference on Foundations of Software Science and Computation Structures, volume 1378 of Lecture Notes in Computer Science, pp. 140-155. Springer-Verlag, 1998.

[6] A. Church. The Calculi of Lambda Conversion. Princeton University Press, 1941.

[7] R. Milner. A calculus of communicating systems. LNCS 92, pages 1–171, Springer-Verlag. 1980.

[8] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts 1-2. Information and Computation, 100(1), pp. 1-77, 1992. (First appeared in 1989 as a LFCS report)

[9] D. Sangiorgi. Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. PhD thesis CST-99-93, Department of Computer Science, University of Edinburgh, 1992.

[10] L. Cardelli. Abstractions for mobile computation. In J. Vitek and C. Jensen (eds), Secure Internet Programming: Security Issues for Mobile and Distributed Objects, volume 1603 of Lecture Notes in Computer Science, pp. 51-94. Springer-Verlag, 1999.

[11] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In Proc. 3rd International Workshop on High-Level Concurrent Languages (HLCL'98), vol. 16(3) of Electronic Notes in Theoretical Computer Science, Elsevier, 1998.

[12] R. M. Amadio. An asynchronous model of locality, failure, and process mobility. In Proc. COORDINATION 97, volume 1282 of Lecture Notes in Computer Science, pp. 374-391, Springer-Verlag, Berlin, 1997.

[13] P. Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In Proc. ICALP'98, volume 1443 of Lecture Notes in Computer Science, pp. 695-706, Springer-Verlag, 1998.

[14] T. Sekiguchi and A. Yonezawa. A calculus with code mobility. In H. Bowman and J. Derrick (Eds), Proc. Second IFIP Intl. Conf. on FMOODS, pp.21-36, Chapman Hall, 1997.

[15] C. Fournet, G. Gonthier, J. Levy, L. Maranget, and D.Remy. A calculus of mobile agents. In Proc. 7th Intl. Conf. on Concurrency Theory(CONCUR'96), pp.406-421, 1996.

[16] F. Levi, D. Sangiorgi. Controlling interference in ambients. In Proc. POPL'00, pages 352–364, Boston, Massachusetts, 2000.

[17] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In Internet Programming Languages, 1999.
Avaliable at: `http://www.cs.purdue.edu/homes/jv/pub/wipl-book99.ps.gz`

[18] P. Zimmer. On the expressiveness of pure mobile ambients. In L. Aceto and B. Victor, editors, Proc. EXPRESS '00, pages 81–204. To apper in Electronic Notes in Theoretical Computer Science, Vol 39,

Elsevier.

[19] L. Cardelli and A. D. Gordon. Mobile ambients - annex, unpublished notes, 1998.
Avaliable at: http://research.microsoft.com/Users/luca/Papers/MobileAmbientsAnnex.A4.ps

[20] A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. In Proc. FoSSaCS'99, volume 1578 of Lecture Notes in Computer Science, pp. 212-226. Springer-Verlag, 1999.

[21] U. Nestmann and B. C. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, Proc. of the 7th Int. Conf. on Concurrency Theory (CONCUR 96), number 1119 in LNCS, pages 179–194. Springer-Verlag, 1996.

[22] V. T. Vasconcelos. The call-by-value lambda-calculus, the SECD machine, and the pi-calculus. Technical Report DI/FCUL TR-00-3. Department of Computer Science, University of Lisbon. May 2000.
Avaliable at: http://www.di.fc.ul.pt/biblioteca/tech-reports/00-3.pdf

[23] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. In Conference Record of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press, 1999.

[24] C. Fournet and G. Gonthier. The reflexive CHAM and the join-calculus. In Proc. 23rd Annual ACM Symposium on Princples of Programming Languges(POPL'96), pp. 372-385, 1996.

[25] G. Berry and G. Boudol. The chemical abstract machine. Theoretical Computer Science, 96(1), pp. 217-248, 1992.

[26] C. Fournet. The Join-calculus: a calculus for distributed mobile programming. PhD thesis, Ecole Polytechnique, November 1998, published by INRIA.
Avaliable at: http://www.research.microsoft.com/~fournet/papers/dissertation.ps.gz

[27] L. Cardelli and A. D. Gordon. Types for mobile ambients. In Proc. POPL'99, pp. 79-92. ACM Press. 1999.

[28] L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for mobile ambients. In Proc. ICALP'99, volume 1644 of Lecture Notes in Computer Science, pp. 230-239. Spinger-Verlag, 1999.

[29] P. Zimmer. Subtyping and typing algorithms for mobile ambients. In Proc. FoSSaCS'2000, LNCS 1784:375-390, Springer-Verlag, 2000.

[30] L. Cardelli, G. Ghelli, A. D. Gordon. Ambient groups and mobility types. In Proc. TCS2000, LNCS 1872:333–347, Springer-Verlag, 2000.

[31] M. Bugliesi, G. Castagna. Secure safe ambients. In Conf. Rec. POPL'01: 28th ACM Symp. Pronc. of Prog. Langs., pp.222-235, 2001.

[32] M. Dezani-Ciancaglini and I. Salvo. Security types for mobile safe ambients. In Proc. ASIAN'00, volume 1961 of LNCS, pages 215–236, Springer-Verlag, 2000.

[33] T. Amtoft. Casual type system for ambient movements. Draft.
Avaliable from: http://www.cs.bu.edu/associates/tamtoft/papers.html

[34] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. In Proceedings of International Conference on Concurrency Theory (CONCUR), volume 1877 of LNCS, pages 365–379. Springer, 2000.

[35] L. Cardelli, G. Ghelli, and A. D. Gordon. Types for the Ambient Calculus. To Appear in Information and Computation special issue on TCS'2000.

[36] L. Cardelli. Ambit. http://www.luca.demon.co.uk/Ambit/Ambit.html, 1997.

[37] L. Cardelli. Mobile ambient synchronization. SRC Technical Note 1997-013. DEC System Research Center, July 1997.

Avaliable at: http://research.microsoft.com/Users/luca/Notes/SRCNote97-13.A4.ps

[38] F. Le Fessant. The Jocaml system prototype. http://join.inria.fr/jocaml, 1998.

[39] C. Fournet, J. L é vy, and A. Schmitt. An asynchronous, distributed implementation of mobile ambients. In Proc. IFIP TCS'2000, pages 348–364, Japan, 2000.

[40] L. Cardelli and A. D. Gordon. Anytime, anywhere, modal logics for mobile ambients. In Proc. 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00), pp 365-377, Boston, Massachusetts, Jan. 19-21, 2000.

[41] D. Sangiorgi. Extensionality and intensionality of the ambient logic. In Proc. POPL'01, pp.4-17, ACM Press, 2001.

[42] W. Charatonik, S. Dal-Zilio, A. D. Gordon, S. Mukhopadhyay and J. Talbot. The complexity of model checking ambients. In Proc. FOSSACS'01, volumn 2032 of Lecture Notes in Computer Science, pp. 152-167, Springer-Verlag, 2001.

[43] W. Charatonik and J. Talbot. The decidability of model checking mobile ambients. To appear in Proc. CSL'01, the 15th Annual Conference of the European Association for Computer Science Logic.

[44] R. R. Hansen, J. G. Jensen, F. Nielson, H. R. Nielson. Abstract interpretation of mobile ambients. In Proc. SAS'99, volume 1694 of Lecture Notes in Computer Science, pp. 134-148, Springer-Verlag, 1999.

[45] H. R. Nielson, F. Nielson. Shape analysis for mobile ambients. To appear in Proc. 27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00), Boston, Massachusetts, Jan. 19-21, 2000.

[46] F. Nielson, H. R. Nielson, R. R. Hansen, J. G. Jensen. Validating firewalls in mobile ambients. In J. Baeten and S. Mauw(eds.), Proc. CONCUR'99, volumn 1664 of Lecture Notes in Computer Science, pp. 463-477. Springer-Verlag, 1999.

[47] F. Levi and C. Bodei. Security analysis for mobile ambients. In Proceedings of the Workshop on Issues in the Theory of Security, (co-located with ICALP'2000), pages 18–23, University of Geneva, Switzerland, 2000.

[48] J. Feret. Abstract Interpretation-Based Static Analysis of Mobile Ambients. In Proc. SAS'01, volumn 2126 of Lecture Notes in Computer Science, pp. 412-430, Springer-Verlag, 2001.

[49] F. Levi, S. Maffeis. An Abstract Interpretation Framework for Analysing Mobile Ambients. In Proc. SAS'01, volumn 2126 of Lecture Notes in Computer Science, pp. 395-411, Springer-Verlag, 2001.

[50] L. Cardelli and G. Ghelli. A query language for semistructed data based on the ambient logic. In Proc. ESOP'01, volume 2028 of Lecture Notes in Computer Science, pages 1–22, Springer-Verlag, 2001.

[51] P. Stanski and A. Zaslavsky. Expressing dynamics of mobile agent systems using ambient calculus. In Proc. MDDS Workshop at DEXA'98, pages 434–439, Vienna, 1998.

[52] D. Sangiorgi, A. Valente. A distributed abstract machine for safe ambients. In Proc. ICALP'01, LNCS 2076:408–420, Springer-Verlag, 2001.

[53] P. Degano, F. Levi and C. Bodei. Safe ambients: control flow analysis and security. In Proc. ASIAN'00, volume 1961 of LNCS, pages 199–214, Springer-Verlag, 2000.

[54] T. Amtoft, A. J. Kfoury, and S. M. Pericas-Geertsen. What are polymorphically-typed ambients? In D. Sands, ed., ESOP 2001, Genova, vol. 2028 of LNCS, pp.206-220. Springer-Verlag, Apr. 2001. An extended version appears as Technical Report BUCS-TR-2000-021, Comp. Sci. Dept., Boston U., 2000.

[55] M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In TACS 2001, LNCS 2215:38–63, Springer-Verlag, 2001.

[56] M. Merro and M. Hennessy. Bisimulation congruences in safe ambients. To appear in POPL'02, ACM Press, 2002.

[57] B. C. Pierce. Type-Theoretic Fundations for Programming Languages. The MIT Press, 2002, forth coming.

[58] X. Guan, Y. Yang, J. You. Making ambients more robust. In Proc. Intl. Conf. on Software: Theory and Practice, pages 377–384, Beijing, China, 2000.

[59] X. Guan, Y.Yang, J. You. Typing evolving ambients. Information Processing Letters, 80(5), pp 265-270, Nov. 2001.

[60] Ambient Calculi Online. http://www.wikipedia.com/wiki/AmbientCalculiOnline. 2002.

附录 1　引理 3.8 和定理 3.9 的证明

本附录证明第三章关于 ETS-MT类型系统的同余一致性（ subject congruence ）和归约一致性（ subject reduction ）。由于子类型关系的自反和传递性，在下面引理 3.8 和定理 3.9 的证明过程中，如果不作特殊声明，我们假定 **(ET Sub)** 规则在另一类型判定规则运用后只被使用了一次。

引理 3.8 (Subject congruence)　如果 $\Gamma \vdash P : T$ 且 $P \equiv Q$，那么 $\Gamma \vdash Q : T$。

证明　对 $P \equiv Q$ 和 $Q \equiv P$ 的推导过程进行双向归纳证明下列两个假设成立：

(1). 如果 $\Gamma \vdash P : T$ 且 $P \equiv Q$，那么 $\Gamma \vdash Q : T$；

(2). 如果 $\Gamma \vdash Q : T$ 且 $P \equiv Q$，那么 $\Gamma \vdash P : T$。

(Struct Refl) 此时 P 和 Q 完全相同，易知假设 (1) 、 (2) 成立。

(Struct Symm) 此时对于假设 (1) 有 $Q \equiv P$，由归纳假设 (2) 可知假设 (1) 成立。假设 (2) 可同样由归纳假设 (1) 得到。

(Struct Trans) 此时对于假设 (1) 有 $P \equiv R$ 且 $R \equiv Q$。假定 $\Gamma \vdash P : T$，利用归纳假设 (1) 两次可相继得到 $\Gamma \vdash R : T$ 和 $\Gamma \vdash Q : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res) 此时对于假设 (1) 有 $P = (\nu n : T_n)P_1$ 、 $Q = (\nu n : T_n)Q_1$ 且 $P_1 \equiv Q_1$。假定 $\Gamma \vdash (\nu n : T_n)P_1 : T$，这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma, n : T_n \vdash P_1 : T'$ 且 $T' \leq T$。由 $P_1 \equiv Q_1$ 利用归纳假设 (1) 可知 $\Gamma, n : T_n \vdash Q_1 : T'$，再利用 **(ET Res)** 和 **(ET Sub)** 可得 $\Gamma \vdash (\nu n : T_n)Q_1 : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par) 此时对于假设 (1) 有 $P = P_1 \mid R$ 、 $Q = Q_1 \mid R$ 且 $P_1 \equiv Q_1$。假定 $\Gamma \vdash P_1 \mid R : T$，这只能通过 **(ET Sub)** 以及 **(ET Par)** 得到，且有 $\Gamma \vdash P_1 : T_1$ 、 $\Gamma \vdash R : T_2$ 且 $T_1 \mid_t T_2 \leq T$。由 $P_1 \equiv Q_1$ 利用归纳假设 (1) 可知 $\Gamma \vdash Q_1 : T_1$，再利用 **(ET Par)** 和 **(ET Sub)** 可得 $\Gamma \vdash Q_1 \mid R : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Repl) 此时对于假设 (1) 有 $P = !P_1$ 、 $Q = !Q_1$ 且 $P_1 \equiv Q_1$。假定 $\Gamma \vdash !P_1 : T$，这只能通过 **(ET Sub)** 和 **(ET Repl)** 得到，且有 $\Gamma \vdash P_1 : T_1$ 且 $T_1 \mid_t T_1 \leq T$。由 $P_1 \equiv Q_1$ 利用归纳假设 (1) 可知 $\Gamma \vdash Q_1 : T_1$，再利用 **(ET Repl)** 和 **(ET Sub)** 可得 $\Gamma \vdash !Q_1 : T$，因此假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Amb) 此时对于假设 (1) 有 $P = n[P_1]$ 、$Q = n[Q_1]$ 且 $P_1 \equiv Q_1$。假定 $\Gamma \vdash n[P_1] : T$，这只能通过 **(ET Sub)** 和 **(ET Amb)** 得到，且有 $\Gamma \vdash n : T_1$ 、$\Gamma \vdash P_1 : T_1$ 且 $\underline{\vee}^0 \leq T$。由 $P_1 \equiv Q_1$ 利用归纳假设 (1) 可知 $\Gamma \vdash Q_1 : T_1$，再利用 **(ET Amb)** 和 **(ET Sub)** 可得 $\Gamma \vdash n[Q_1] : T$，因此假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Act) 此时对于假设 (1) 有 $P = M.P_1$ 、 $Q = M.Q_1$ 且 $P_1 \equiv Q_1$。假定 $\Gamma \vdash M.P_1 : T$，这只能通过 **(ET Sub)** 和 **(ET Act)** 得到，且有 $\Gamma \vdash M : W$ 、 $\Gamma \vdash P_1 : T_1$ 且 $W(T_1) \leq T$。由 $P_1 \equiv Q_1$ 利用归纳假设 (1) 可知 $\Gamma \vdash Q_1 : T_1$，再利用 **(ET Act)** 和 **(ET Sub)** 可得 $\Gamma \vdash M.Q_1 : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par Zero) 此时有 $P = Q \mid \mathbf{0}$。对于假设 (1)，假定 $\Gamma \vdash Q \mid \mathbf{0} : T$，这只能通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma \vdash Q : T_1$ 、 $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$ 且 $T_1 \mid_t \underline{\vee}^0 \leq T$，再由性质 3.2 中的 **(Par**

Zero T) 可得 $T_1 \leq T$，利用 **(ET Sub)** 可得 $\Gamma \vdash Q : T$，假设 (1) 成立。对于假设 (2)，假定 $\Gamma \vdash Q : T$，由引理 3.5 可得 $\Gamma \vdash \diamondsuit$，利用 **(ET Inact)** 有 $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$，再利用 **(ET Par)** 可得 $\Gamma \vdash Q \mid \mathbf{0} : T \mid_t \underline{\vee}^0$，即 $\Gamma \vdash Q \mid \mathbf{0} : T$，假设 (2) 成立。

(Struct Par Comm) 此时有 $P = P_1 \mid Q_1$ 且 $Q = Q_1 \mid P_1$。对于假设 (1)，假定 $\Gamma \vdash P_1 \mid Q_1 : T$，这只能通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma \vdash P_1 : T_1$、$\Gamma \vdash Q_1 : T_2$ 且 $T_1 \mid_t T_2 \leq T$。通过 **(ET Par)** 可知 $\Gamma \vdash Q_1 \mid P_1 : T_2 \mid_t T_1$。而由性质 3.2 中的 **(Par Symm** T**)** 可知 $T_2 \mid_t T_1 = T_1 \mid_t T_2$，因此利用 **(ET Sub)** 可得 $\Gamma \vdash Q_1 \mid P_1 : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par Assoc) 此时有 $P = (P_1 \mid Q_1) \mid R$ 且 $Q = P_1 \mid (Q_1 \mid R)$。对于假设 (1)，假定 $\Gamma \vdash (P_1 \mid Q_1) \mid R : T$，这只能通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma \vdash P_1 \mid Q_1 : T_1$、$\Gamma \vdash R : T_2$ 且 $T_1 \mid_t T_2 \leq T$，而前者也只能通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma \vdash P_1 : T_3$、$\Gamma \vdash Q_1 : T_4$ 且 $T_3 \mid_t T_4 \leq T_1$。利用两次 **(ET Par)** 可得 $\Gamma \vdash P_1 \mid (Q_1 \mid R) : T_3 \mid_t (T_4 \mid_t T_2)$。由性质 3.2 中的 **(Par Assoc** T**)** 知 $T_3 \mid_t (T_4 \mid_t T_2) = (T_3 \mid_t T_4) \mid_t T_2$，因此利用 **(ET Sub)** 可得 $\Gamma \vdash P_1 \mid (Q_1 \mid R) : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res Zero) 此时有 $P = (\nu n : T_n)\mathbf{0}$ 且 $Q = \mathbf{0}$。对于假设 (1)，假定 $\Gamma \vdash (\nu n : T_n)\mathbf{0} : T$，这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma,\ n : T_n \vdash \mathbf{0} : T_1$ 且 $T_1 \leq T$。由 $n \notin fn(\mathbf{0})$ 根据引理 3.6 可知 $\Gamma \vdash \mathbf{0} : T_1$，再利用 **(ET Sub)** 可得 $\Gamma \vdash \mathbf{0} : T$，假设 (1) 成立。对于假设 (2)，假定 $\Gamma \vdash \mathbf{0} : T$，通过约束名换名可保证 $n \notin dom(\Gamma)$，利用引理 3.7 可知 $\Gamma,\ n : T_n \vdash \mathbf{0} : T$，再利用 **(ET Res)** 可得 $\Gamma \vdash (\nu n : T_n)\mathbf{0} : T$，假设 (2) 成立。

(Struct Res Res) 此时有 $P = (\nu n : T_n)(\nu m : T_m)R$ 且 $Q = (\nu m : T_m)(\nu n : T_n)R$。对于假设 (1)，假定 $\Gamma \vdash (\nu n : T_n)(\nu m : T_m)R : T$，这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma,\ n : T_n \vdash (\nu m : T_m)R : T_1$ 且 $T_1 \leq T$。同样这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma,\ n : T_n,\ m : T_m \vdash R : T_2$ 且 $T_2 \leq T_1$。由类型环境为一集合可知，其元素位置无关，因此利用两次 **(ET Res)** 和 **(ET Sub)** 可得 $\Gamma \vdash (\nu m : T_m)(\nu n : T_n)R : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res Par) 此时有 $P = (\nu n : T_n)(P_1 \mid Q_1)$、$Q = P_1 \mid (\nu n : T_n)Q_1$ 且 $n \notin fn(P_1)$。对于假设 (1)，假定 $\Gamma \vdash (\nu n : T_n)(P_1 \mid Q_1) : T$，这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma,\ n : T_n \vdash P_1 \mid Q_1 : T_1$ 且 $T_1 \leq T$。同样这只能通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma,\ n : T_n \vdash P_1 : T_2$、$\Gamma,\ n : T_n \vdash Q_1 : T_3$ 且 $T_2 \mid_t T_3 \leq T_1$。由 $n \notin fn(P)$ 根据引理 3.6 可知 $\Gamma \vdash P_1 : T_2$。又由 **(ET Res)** 知 $\Gamma \vdash (\nu n : T_n)Q_1 : T_3$，利用 **(ET Par)** 和 **(ET Sub)** 可得 $\Gamma \vdash P_1 \mid (\nu n : T_n)Q_1 : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 过程相反，并使用了引理 3.7，此处略。

(Struct Res Amb) 此时有 $P = (\nu n : T_n)m[R]$、$Q = m[(\nu n : T_n)R]$ 且 $n \neq m$。对于假设 (1)，假定 $\Gamma \vdash (\nu n : T_n)m[R] : T$，这只能通过 **(ET Sub)** 和 **(ET Res)** 得到，且有 $\Gamma,\ n : T_n \vdash m[R] : T_1$ 且 $T_1 \leq T$。同样这只能通过 **(ET Sub)** 和 **(ET Amb)** 得到，且有 $\Gamma,\ n : T_n \vdash m : T_m$、$\Gamma,\ n : T_n \vdash R : T_m$ 且 $\underline{\vee}^0 \leq T_1$。由 $n \neq m$ 根据引理 3.6 可知 $\Gamma \vdash m : T_m$。又由 **(ET Res)** 知 $\Gamma \vdash (\nu n : T_n)R : T_m$，利用 **(ET Amb)** 和 **(ET Sub)** 可得 $\Gamma \vdash m[(\nu n : T_n)R] : T$，因此假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 过程相反，并使用了引理 3.7，此处略。

(Struct Repl Par) 此时有 $P = !R$ 且 $Q = R \mid !R$。对于假设 (1)，假定 $\Gamma \vdash\ !R : T$，这只能通过 **(ET Sub)** 和 **(ET Repl)** 得到，且有 $\Gamma \vdash R : T_1$ 且 $T_1 \mid T_1 \leq T$。利用 **(ET Par)** 可知 $\Gamma \vdash R \mid !R : T_1 \mid_t (T_1 \mid_t T_1)$。由 $\mid_t$ 的定义可知 $T_1 = Z^Y$，再由 $\mid_t$ 的定义可得 $Z^Y \mid_t Z^Y = Z^Y \mid_t (Z^Y \mid_t Z^Y)$，利用 **(ET Sub)** 可得 $\Gamma \vdash R \mid !R : T$，假设 (1) 成立。对于假设 (2)，假定 $\Gamma \vdash R \mid !R : T$，这只能通过 **(ET Sub)** 和 **(ET Par)**，且有 $\Gamma \vdash R : T_1$、$\Gamma \vdash\ !R : T_2$ 且 $T_1 \mid_t T_2 \leq T$，再由 **(ET Repl)** 可知 $T_1 = Z^Y$. 因此可利用性质 3.2 中的 **(Par Zero T)** 和 **(Par Strict T)** 得到 $T_2 \leq T$。最后利用 **(ET Sub)** 可得 $\Gamma \vdash\ !R : T$，假设 (2) 成立。

(Struct Repl Zero) 此时有 $P = !\mathbf{0}$ 且 $Q = \mathbf{0}$。对于假设 (1)，假定 $\Gamma \vdash !\mathbf{0} : T$，这只能通过 **(ET Sub)** 和 **(ET Repl)** 得到，且有 $\Gamma \vdash \mathbf{0} : T_1$ 且 $T_1 \mid_t T_1 \leq T$。由 $\mid_t$ 的定义知 $T_1 = Z^Y$，因此可利用性质 3.2 中的 **(Par Zero T)** 和 **(Par Strict T)** 得到 $T_1 \leq T_1 \mid_t T_1$。利用 **(ET Sub)** 可得 $\Gamma \vdash \mathbf{0} : T$，假设 (1) 成立。对于假设 (2)，假定 $\Gamma \vdash \mathbf{0} : T$，这只能通过 **(ET Sub)** 和 **(ET Inact)** 得到，且有 $\Gamma \vdash \mathbf{0} : \underline{\vee}^0$ 且 $\underline{\vee}^0 \leq T$，利用 **(ET Repl)** 和 **(ET Sub)** 可得 $\Gamma \vdash\ \mathbf{0} : T$，假设 (2) 成立。

(Struct Empty) 此时有 $P = \epsilon\,.\,Q$。对于假设 (1)，假定 $\Gamma \vdash \epsilon\,.\,Q : T$，这只能通过 **(ET Sub)** 和 **(ET Act)** 得到，且有 $\Gamma \vdash \epsilon : W$、$\Gamma \vdash Q : T'$ 且 $W(T') \leq T$，而由 **(ET Empty)** 知 $W = -$，因此有 $W(T') = T'$，利用 **(ET Sub)** 可得 $\Gamma \vdash Q : T$，假设 (1) 成立。对于假设 (2)，假定 $\Gamma \vdash Q : T$，利用引理 3.5 知 $\Gamma \vdash \diamondsuit$，再由 **(ET Empty)** 可得 $\Gamma \vdash \epsilon : -$，因此利用 **(ET Act)** 可得 $\Gamma \vdash \epsilon\,.\,Q : -(T)$，即 $\Gamma \vdash \epsilon\,.\,Q : T$，假设 (2) 成立。

(Struct Path) 此时有 $P = (M_1\,.\,M_2)\,.\,R$ 且 $Q = M_1\,.\,(M_2\,.\,R)$。对于假设 (1)，假定 $\Gamma \vdash (M_1\,.\,M_2)\,.\,R : T$，这只能通过 **(ET Sub)** 和 **(ET Act)** 得到，且有 $\Gamma \vdash M_1\,.\,M_2 : W_1$、$\Gamma \vdash R : T'$ 且 $W_1(T') \leq T$，而由前者只能通过 **(ET Path)** 得到，且有 $\Gamma \vdash M_1 : W_2$、$\Gamma \vdash M_2 : W_3$ 且 $W_1 = W_2(W_3)$，利用两次 **(ET Act)** 可知 $\Gamma \vdash M_1\,.\,(M_2\,.\,R) : W_2(W_3(T'))$。而 $W_2(W_3(T')) = (W_2(W_3))(T') = W_1(T')$，因此利用 **(ET Sub)** 有 $\Gamma \vdash M_1\,.\,(M_2\,.\,R) : T$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 基本类似，此处略。

通过以上对所有可能情况的分析可知，结论成立. ∎

下面在另外一个引理的基础上，我们证明类型判定规则的正确性。

引理 A.1 假定 $\Gamma \vdash A : Z^Y \cdot_t -$，以下两个结论成立：

(1). 如果 $\Gamma \vdash A\,.\,P : T$，那么 $\Gamma \vdash P : T'$ 且 $T' \leq T$；

(2). 如果 $\Gamma \vdash A\,.\,P \mid Q : T$，那么 $\Gamma \vdash P \mid Q : T'$ 且 $T' \leq T$。

证明 首先证明 (1)，再利用 (1) 的结论证明 (2)。

(1). 已知条件 $\Gamma \vdash A\,.\,P : T$ 必然通过 **(ET Sub)** 和 **(ET Act)** 得到，且有 $\Gamma \vdash A : W$、$\Gamma \vdash P : T'$ 且 $W(T') \leq T$。而由已知 $\Gamma \vdash A : Z^Y \cdot_t -$，因此有 $Z^Y \cdot_t T' \leq T$，利用性质 3.1 可知 $T' \leq Z^Y \cdot_t T'$，即 $T' \leq T$。

(2). 条件 $\Gamma \vdash A\,.\,P \mid Q : T$ 必然通过 **(ET Sub)** 和 **(ET Par)** 得到，且有 $\Gamma \vdash A\,.\,P : T_1$、$\Gamma \vdash Q : T_2$ 且 $T_1 \mid_t T_2 \leq T$。由结论 (1) 知 $\Gamma \vdash P : T_1'$ 且 $T_1' \leq T_1$。令 $T' = T_1 \mid T_2$, 利用 **(ET Par)** 可得 $\Gamma \vdash P \mid Q : T'$ 且 $T' \leq T$。

∎

定理 3.9 (Subject reduction) 如果 $\Gamma \vdash P : T$ 且 $P \longrightarrow Q$，那么 $\Gamma \vdash Q : T'$ 且 $T' \leq T$。

证明 对 $P \longrightarrow Q$ 的推导过程进行归约。

(Red In) 即：已知 $\Gamma \vdash m[\,\mathtt{in}\, n \,.\, P_1 \mid P_2] \mid n[\,\overline{\mathtt{in}}\, m \,.\, Q_1 \mid Q_2] : T$，要证 $\Gamma \vdash n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] : T'$ 且 $T' \leq T$。证明过程如下：

(1)	$\Gamma \vdash m[\,\mathtt{in}\, n \,.\, P_1 \mid P_2] \mid n[\,\overline{\mathtt{in}}\, m \,.\, Q_1 \mid Q_2] : T$	已知条件
(2.1)	$\Gamma \vdash m[\,\mathtt{in}\, n \,.\, P_1 \mid P_2] : T_1$	(1),仅能使用**(ET Sub)**
(2.2)	$\Gamma \vdash n[\,\overline{\mathtt{in}}\, m \,.\, Q_1 \mid Q_2] : T_2$	+**(ET Par)**
(2.3)	$T_1 \mid_t T_2 \leq T$	
(3.1)	$\Gamma \vdash m : T_m$	(2.1),仅能使用**(ET Sub)**
(3.2)	$\Gamma \vdash \mathtt{in}\, n \,.\, P_1 \mid P_2 : T_m$	+**(ET Amb)**
(3.3)	$\underline{\vee}^0 \leq T_1$	
(4.1)	$\Gamma \vdash n : T_n$	(2.2),仅能使用**(ET Sub)**
(4.2)	$\Gamma \vdash \overline{\mathtt{in}}\, m \,.\, Q_1 \mid Q_2 : T_n$	+**(ET Amb)**
(4.3)	$\underline{\vee}^0 \leq T_2$	
(5)	$\Gamma \vdash \mathtt{in}\, n : \curvearrowright^1 \cdot_t -$	(4.1)+**(ET Cap Mbl)**
(6)	$\Gamma \vdash P_1 \mid P_2 : T_m$	(3.2)(5)+引理A.1.(2)加**(ET Sub)**
(7)	$\Gamma \vdash m[P_1 \mid P_2] : \underline{\vee}^0$	(3.1)(6)+**(ET Amb)**
(8)	$\Gamma \vdash \overline{\mathtt{in}}\, m : \underline{\vee}^1 \cdot_t -$	(3.1)+**(ET Cap Imm)**
(9)	$\Gamma \vdash Q_1 \mid Q_2 : T_n$	(4.2)(8)+引理A.1.(2)加**(ET Sub)**
(10)	$\Gamma \vdash m[P_1 \mid P_2] \mid Q_1 \mid Q_2 : \underline{\vee}^0 \mid_t T_n$	(7)(9)+**(ET Par)**
(11)	$\Gamma \vdash m[P_1 \mid P_2] \mid Q_1 \mid Q_2 : T_n$	(10)+性质3.2
(12)	$\Gamma \vdash n[m[P_1 \mid P_2] \mid Q_1 \mid Q_2] : \underline{\vee}^0$	(4.1)(11)+**(ET Amb)**
(13)	$\underline{\vee}^0 \leq T$	(2.3)(3.3)(4.3)+性质3.2
(14)	（结论成立）	(12)(13), 取 $T' = \underline{\vee}^0$

(Red Out) 即：已知 $\Gamma \vdash n[m[\,\mathtt{out}\, n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\, m \,.\, Q_1 \mid Q_2] : T$，要证 $\Gamma \vdash m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] : T'$ 且 $T' \leq T$。证明过程如下：

(1)	$\Gamma \vdash n[m[\,\mathtt{out}\, n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\, m \,.\, Q_1 \mid Q_2] : T$	已知条件
(2.1)	$\Gamma \vdash n : T_n$	(1),仅能使用**(ET Sub)**
(2.2)	$\Gamma \vdash m[\,\mathtt{out}\, n \,.\, P_1 \mid P_2] \mid \overline{\mathtt{out}}\, m \,.\, Q_1 \mid Q_2 : T_n$	+**(ET Amb)**
(2.3)	$\underline{\vee}^0 \leq T$	
(3.1)	$\Gamma \vdash m[\,\mathtt{out}\, n \,.\, P_1 \mid P_2] : T_1$	(2.2),仅能使用**(ET Sub)**
(3.2)	$\Gamma \vdash \overline{\mathtt{out}}\, m \,.\, Q_1 \mid Q_2 : T_2$	+**(ET Par)**
(3.3)	$T_1 \mid_t T_2 \leq T_n$	
(4.1)	$\Gamma \vdash m : T_m$	(3.1),仅能使用**(ET Sub)**

(4.2)	$\Gamma \vdash \mathtt{out}\ n\ .\ P_1 \mid P_2 : T_m$	**+(ET Amb)**
(4.3)	$\underline{\vee}^0 \leq T_1$	
(5)	$T_2 \leq T_n$	(4.3)(3.3)+性质3.2
(6)	$\Gamma \vdash \overline{\mathtt{out}}\ m\ .\ Q_1 \mid Q_2 : T_n$	(3.2)(5)+**(ET Sub)**
(7)	$\Gamma \vdash \overline{\mathtt{out}}\ m : \underline{\vee}^1 \cdot_t -$	(4.1)+**(ET Cap Imm)**
(8)	$\Gamma \vdash Q_1 \mid Q_2 : T_n$	(6)(7)+引理A.1.(2)加**(ET Sub)**
(9)	$\Gamma \vdash n[Q_1 \mid Q_2] : \underline{\vee}^0$	(2.1)(8)+**(ET Amb)**
(10)	$\Gamma \vdash \mathtt{out}\ n : \curvearrowright^1 \cdot_t -$	(2.1)+**(ET Cap Mbl)**
(11)	$\Gamma \vdash P_1 \mid P_2 : T_m$	(4.2)(10)+引理A.1.(2)加**(ET Sub)**
(12)	$\Gamma \vdash m[P_1 \mid P_2] : \underline{\vee}^0$	(4.1)(11)+**(ET Amb)**
(13)	$\Gamma \vdash m[P_1 \mid P_2] \mid n[Q_1 \mid Q_2] : \underline{\vee}^0$	(12)(9)+**(ET Par)**
(14)	（结论成立）	(13)(2.3), 取 $T' = \underline{\vee}^0$

(Red Open) 即：已知 $\Gamma \vdash \mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2] : T$，要证 $\Gamma \vdash P \mid Q_1 \mid Q_2 : T'$ 且 $T' \leq T$。

证明过程如下：

(1)	$\Gamma \vdash \mathtt{open}\ n\ .\ P \mid n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2] : T$	已知条件
(2.1)	$\Gamma \vdash \mathtt{open}\ n\ .\ P : T_1$	(1),仅能使用**(ET Sub)**
(2.2)	$\Gamma \vdash n[\overline{\mathtt{open}}\ .\ Q_1 \mid Q_2] : T_2$	**+(ET Par)**
(2.3)	$T_1 \mid_t T_2 \leq T$	
(3.1)	$\Gamma \vdash n : T_n$	(2.2),仅能使用**(ET Sub)**
(3.2)	$\Gamma \vdash \overline{\mathtt{open}}\ .\ Q_1 \mid Q_2 : T_n$	**+(ET Amb)**
(3.3)	$\underline{\vee}^0 \leq T_2$	
(4)	$T_1 \leq T$	(3.3)(2.3)+性质3.2
(5.1)	$\Gamma \vdash \overline{\mathtt{open}}\ .\ Q_1 : T_{n1}$	(3.2),仅能使用**(ET Sub)**
(5.2)	$\Gamma \vdash Q_2 : T_{q2}$	**+(ET Par)**
(5.3)	$T_{n1} \mid_t T_{q2} \leq T_n$	
(6.1)	$\Gamma \vdash \overline{\mathtt{open}} : W$	(5.1),仅能使用**(ET Sub)**
(6.2)	$\Gamma \vdash Q_1 : T_{q1}$	**+(ET Act)**
(6.3)	$W(T_{q1}) \leq T_{n1}$	
(7)	$W = \underline{\vee}^1[-]$	(6.1),仅能使用**(ET Co-open)**
(8)	$\underline{\vee}^1[T_{q1}] \mid_t T_{q2} \leq T_n$	(7)(6.3)(5.3)+性质3.2
(9)	$\underline{\vee}^1 \mid_t T_{q2}[T_{q1} \mid_t T_{q2}] \leq T_n$	(8)+$\mid_t$的定义
(10.1)	$T_n = Z^Y[T'_n]$	(9)+$\leq$的定义
(10.2)	$T_{q1} \mid_t T_{q2} \leq T'_n$	
(11)	$\Gamma \vdash Q_1 \mid Q_2 : T_{q1} \mid_t T_{q2}$	(6.2)(5.2)+**(ET Par)**
(12)	$\Gamma \vdash Q_1 \mid Q_2 : T'_n$	(11)(10.2)+**(ET Sub)**

(13)	$\Gamma \vdash \mathtt{open}\ n : \underline{\vee}^1 \cdot_t (T'_n \mid_t -)$	(10.1)+**(ET Open)**
(14.1)	$\Gamma \vdash P : T_p$	(2.1),仅能使用**(ET Sub)**
(14.2)	$\underline{\vee}^1 \cdot_t (T'_n \mid_t T_p) \leq T_1$	+**(ET Act)**
(15)	$\Gamma \vdash P \mid Q_1 \mid Q_2 : T_p \mid T'_n$	(14.1)(12)+**(ET Par)**
(16)	$T_p \mid T'_n \leq T$	(14.2)(4)+性质3.2, 3.1
(17)	（结论成立）	(15)(16), 取 $T' = T_p \mid T'_n$

(Red Par) 即：已知 $\Gamma \vdash P \mid Q : T$ 且 $P \longrightarrow P'$，要证 $\Gamma \vdash P' \mid Q : T'$ 且 $T' \leq T$。证明过程如下：

(1)	$\Gamma \vdash P \mid Q : T$	已知条件
(2.1)	$\Gamma \vdash P : T_p$	(1),仅能使用**(ET Sub)**
(2.2)	$\Gamma \vdash Q : T_q$	+**(ET Par)**
(2.3)	$T_p \mid_t T_q \leq T$	
(3)	$P \longrightarrow P'$	已知条件
(4.1)	$\Gamma \vdash P' : T'_p$	(2.1)(3)+归纳假设
(4.2)	$T'_p \leq T_p$	
(5)	$\Gamma \vdash P' \mid Q : T_p \mid_t T_q$	(4.1)(4.2)+**(ET Sub)(ET Par)**
(6)	（结论成立）	(5)(2.3), 取 $T' = T_p \mid_t T_q$

(Red Res) 即：已知 $\Gamma \vdash (\nu n : T_n)P : T$ 且 $P \longrightarrow P'$，要证 $\Gamma \vdash (\nu n : T_n)P' : T'$ 且 $T' \leq T$。证明过程如下：

(1)	$\Gamma \vdash (\nu n : T_n)P : T$	已知条件
(2.1)	$\Gamma,\ n : T_n \vdash P : T_1$	(1),仅能使用**(ET Sub)**
(2.2)	$T_1 \leq T$	+**(ET Res)**
(3)	$P \longrightarrow P'$	已知条件
(4.1)	$\Gamma,\ n : T_n \vdash P : T_2$	(2.1)(3)+归纳假设
(4.2)	$T_2 \leq T_1$	
(5)	$\Gamma \vdash (\nu n : T_n)P : T_1$	(4.1)(4.2)+**(ET Res)(ET Sub)**
(6)	（结论成立）	(5)(2.2), 取 $T' = T_1$

(Red Amb) 即：已知 $\Gamma \vdash n[P] : T$ 且 $P \longrightarrow P'$，要证 $\Gamma \vdash n[P'] : T'$ 且 $T' \leq T$。证明过程如下：

(1)	$\Gamma \vdash n[P] : T$	已知条件
(2.1)	$\Gamma \vdash n : T_n$	(1),仅能使用**(ET Sub)**
(2.2)	$\Gamma \vdash P : T_n$	+**(ET Amb)**
(2.3)	$\underline{\vee}^0 \leq T$	
(3)	$P \longrightarrow P'$	已知条件
(4.1)	$\Gamma \vdash P' : T'_n$	(2.2)(3)+归纳假设
(4.2)	$T'_n \leq T_n$	

(5)	$\Gamma \vdash n[P'] : \underline{\vee}^0$	(4.1)(4.2)+**(ET Sub)(ET Amb)**
(6)	（结论成立）	(5)(2.3), 取 $T' = \underline{\vee}^0$

(Red Struct) 即：已知 $\Gamma \vdash P : T$ 、 $P \equiv P'$ 、 $P' \longrightarrow P''$ 且 $P'' \equiv P'''$ ，要证 $\Gamma \vdash P''' : T'$ 且 $T' \leq T$ 。证明过程如下：

(1)	$\Gamma \vdash P : T$	已知条件
(2)	$P \equiv P'$	已知条件
(3)	$\Gamma \vdash P' : T$	(1)(2)+引理3.8
(4)	$P' \longrightarrow P''$	已知条件
(5.1)	$\Gamma \vdash P'' : T_1$	(3)(4)+归纳假设
(5.2)	$T_1 \leq T$	
(6)	$P'' \equiv P'''$	已知条件
(7)	$\Gamma \vdash P''' : T_1$	(5)(6)+引理3.8
(8)	（结论成立）	(7)(5.2), 取 $T' = T_1$

通过以上对所有可能情况的分析可知，结论成立. ∎

附录 2　引理 4.14 的证明

引理 4.14 指出同余进程存在活体和残余部分均为同余的硬化结果。本附录给出其具体的证明过程。为方便行文，首先引入以下一些附加的定义和结论。

定义 B.1　*活体的结构同余关系“ $\hat{\equiv}$ ”定义为：*

(1). $n[P] \mathrel{\hat{\equiv}} n[Q]$，*如果* $P \equiv Q$；

(2). $A.P \mathrel{\hat{\equiv}} A.Q$，*如果* $P \equiv Q$，*这里* $A \in \mathbf{Action}$；

引理 B.2　*对任意活体* P、Q *和* R *有：*

(1). $P \mathrel{\hat{\equiv}} P$；

(2). $P \mathrel{\hat{\equiv}} Q \Longrightarrow Q \mathrel{\hat{\equiv}} P$；

(3). $P \mathrel{\hat{\equiv}} Q,\ Q \mathrel{\hat{\equiv}} R \Longrightarrow P \mathrel{\hat{\equiv}} R$；

(4). $P \mathrel{\hat{\equiv}} Q \Longrightarrow P \equiv Q$。

证明　通过 $\hat{\equiv}$ 的定义易证，此处略。 ∎

定义 B.3　*硬化结果集* $\mathcal{C}$ *上的结构同余关系“* $\equiv$ *”定义为：*

$$C \equiv D \quad \stackrel{\triangle}{=} \quad C = (\nu\vec{r})\langle P'\rangle P'',\ D = (\nu\vec{r})\langle Q'\rangle Q'',\ P' \mathrel{\hat{\equiv}} Q',\ P'' \equiv Q''$$

通过上述 $\mathcal{C}$ 上结构同余关系的定义，引理 4.14 的证明可转化为证明个同余进程存在相互同余的硬化结果。本附录最后将证明这一等价结论（引理 B.6 ）。

引理 B.4　*如果* $C \equiv D$，*则* $\overline{(\nu n)}C \equiv \overline{(\nu n)}D$。

证明　由 $C \equiv D$ 的定义知 $C = (\nu\vec{r})\langle P'\rangle P''$、$D = (\nu\vec{r})\langle Q'\rangle Q''$、$P' \mathrel{\hat{\equiv}} Q'$ 且 $P'' \equiv Q''$。分情况讨论如下：

(1). 若 $n \notin fn(P')$，由引理 B.2.(4) 和引理 4.3 知，$n \notin fn(Q')$，因此有 $\overline{(\nu n)}C = (\nu\vec{r})\langle P'\rangle(\nu n)P'' \equiv (\nu\vec{r})\langle Q'\rangle(\nu n)Q'' = \overline{(\nu n)}D$，结论成立。

(2). 若 $n \in fn(P')$，则分以下两种情况：

(a) 若 $P' = m[P'_1]$、$m \neq n$ 且 $n \notin fn(P'')$，则 $\overline{(\nu n)}C = (\nu\vec{r})\langle m[(\nu n)P'_1]\rangle P''$。而由 $P' \mathrel{\hat{\equiv}} Q'$ 的定义知，$Q' = m[Q'_1]$ 且 $P'_1 \equiv Q'_1$，再利用 $P'' \equiv Q''$ 根据引理 4.3 可得 $n \notin fn(Q'')$，因此，$\overline{(\nu n)}D = (\nu\vec{r})\langle m[(\nu n)Q'_1]\rangle Q''$，易知 $\overline{(\nu n)}C \equiv \overline{(\nu n)}D$，结论成立。

(b) 否则，易知 $\overline{(\nu n)}C = (\nu n,\vec{r})\langle P'\rangle P''$ 且 $\overline{(\nu n)}D = (\nu n,\vec{r})\langle Q'\rangle Q''$，结论成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 B.5　$\overline{(\nu n)(\nu m)}C \equiv \overline{(\nu m)(\nu n)}C$。

证明　通过类似引理 B.4 的分析易证，此处略。 ∎

结合上文 $\mathcal{C}$ 上结构同余关系的定义，下面的引理 B.6 给出了与引理 4.14 完全相同的结论。

引理 B.6 如果 $P \equiv Q$ 且 $Q > D$，则存在 C 满足 $P > C$ 且 $C \equiv D$。

证明 通过对 $P \equiv Q$ 的推导过程进行归纳证明以下两个假设成立：如果 $P \equiv Q$，那么

(1). 若 $P > C$，则存在 D 满足 $Q > D$ 且 $C \equiv D$；

(2). 若 $Q > D$，则存在 C 满足 $P > C$ 且 $C \equiv D$。

(Struct Refl) 此时 $P = Q$，易知假设成立。

(Struct Symm) 此时 $Q \equiv P$，假设 (1) 可由归纳假设 (2) 得到，假设 (2) 可由归纳假设 (1) 得到。

(Struct Trans) 此时有 $P \equiv R$ 且 $R \equiv Q$。对于假设 (1)，假定 $P > (\nu\vec{p})\langle P'\rangle P''$，由归纳假设 (1) 知，有 $R > (\nu\vec{r})\langle R'\rangle R''$、$P' \mathrel{\widehat{\equiv}} R'$ 且 $P'' \equiv R''$。再由归纳假设 (1) 知，有 $Q > (\nu\vec{q})\langle Q'\rangle Q''$、$R' \mathrel{\widehat{\equiv}} Q'$ 且 $R'' \equiv Q''$。由 $\mathrel{\widehat{\equiv}}$ 和 $\equiv$ 的传递性可得 $P' \mathrel{\widehat{\equiv}} Q'$ 且 $P'' \equiv Q''$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res) 此时有 $P = (\nu n)P'$、$Q = (\nu n)Q'$ 且 $P' \equiv Q'$。对于假设 (1)，假定 $(\nu n)P > C$，这只能通过规则 **(Harden Res)** 得到，即有 $P' > C'$ 且 $C = \overline{(\nu n)}C'$。由归纳假设 (1) 知，存在 D' 满足 $Q' > D'$ 且 $C' \equiv D'$。利用 **(Harden Res)** 可得 $Q = (\nu n)Q' > \overline{(\nu n)}D'$。由引理 B.4 知 $\overline{(\nu n)}C' \equiv \overline{(\nu n)}D'$，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par) 此时有 $P = P_1 \mid R$、$Q = Q_1 \mid R$ 且 $P_1 \equiv Q_1$。对于假设 (1)，假定 $P_1 \mid R > (\nu\vec{r})\langle P'\rangle P''$，该结果必定来自以下两条规则之一：

(Harden Par 1) 此时有 $P_1 > (\nu\vec{r})\langle P'\rangle P'''$、$P'' = P''' \mid R$ 且 $\{\vec{r}\} \cap fn(R) = \emptyset$。由归纳假设 (1) 知，有 $Q_1 > (\nu\vec{r})\langle Q'\rangle Q'''$、$P' \mathrel{\widehat{\equiv}} Q'$ 且 $P''' \equiv Q'''$，利用 **(Harden Par 1)** 得 $Q_1 \mid R > (\nu\vec{r})\langle Q'\rangle (Q''' \mid R)$，由 $P''' \mid R \equiv Q''' \mid R$ 知，假设 (1) 成立。

(Harden Par 2) 此时有 $R > (\nu\vec{r})\langle P'\rangle P'''$、$P'' = P_1 \mid P'''$ 且 $\{\vec{r}\} \cap fn(P_1) = \emptyset$。由引理 4.3 知，$fn(P_1) = fn(Q_1)$，因此 $\{\vec{r}\} \cap fn(Q_1) = \emptyset$。利用 **(Harden Par 1)** 得 $Q_1 \mid R > (\nu\vec{r})\langle P'\rangle (Q_1 \mid P''')$，由 $P_1 \mid P''' \equiv Q_1 \mid P'''$ 知，假设 (1) 成立。

假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Repl) 此时有 $P = !P_1$、$Q = !Q_1$ 且 $P_1 \equiv Q_1$。对于假设 (1)，假定 $!P_1 > (\nu\vec{r})\langle P'\rangle P''$，这只能通过 **(Harden Repl)** 得到，即有 $P_1 > (\nu\vec{r})\langle P'\rangle P'''$ 且 $P'' = P''' \mid !P_1$。由归纳假设 (1) 知，有 $Q_1 > (\nu\vec{r})\langle Q'\rangle Q'''$、$P' \mathrel{\widehat{\equiv}} Q'$ 且 $P''' \equiv Q'''$，利用 **(Harden Repl)** 得，$!Q_1 > (\nu\vec{r})\langle Q'\rangle (Q''' \mid !Q_1)$，由 $P''' \mid !P_1 \equiv Q''' \mid !Q_1$ 知，假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Amb) 此时有 $P = n[P_1]$、$Q = n[Q_1]$ 且 $P_1 \equiv Q_1$。对于假设 (1)，假定 $n[P_1] > (\nu\vec{r})\langle P'\rangle P''$，这只能通过 **(Harden Amb)** 得到，即有 $P' = n[P_1]$、$\{\vec{r}\} = \emptyset$ 且 $P'' = \mathbf{0}$。而利用 **(Harden Amb)** 有，$n[Q_1] > (\nu)\langle n[Q_1]\rangle \mathbf{0}$，由已知 $P_1 \equiv Q_1$ 知，$n[P_1] \mathrel{\widehat{\equiv}} n[Q_1]$，因此假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Action) 此时有 $P = M.P_1$、$Q = M.Q_1$ 且 $P_1 \equiv Q_1$。对于假设 (1)，假定 $M.P_1 > C$，由引理 4.11，只能存在以下两种情况：

(1). $M > A.N$、$C = (\nu)\langle A.P'\rangle \mathbf{0}$ 且 $P' \equiv N.P_1$。此时，由引理 4.12 知，$M.Q_1 > (\nu)\langle A.Q'\rangle \mathbf{0}$ 且 $Q' \equiv N.Q_1$，由 $P_1 \equiv Q_1$ 可得 $P' \equiv N.P_1 \equiv N.Q_1 \equiv Q'$，因此 $A.P' \mathrel{\widehat{\equiv}} A.Q'$，假设 (1) 成立。

(2). $M = \epsilon$ 且 $P_1 > C$。此时根据归纳假设 (1)，存在 D 满足 $Q_1 > D$ 且 $C \equiv D$。由引理 4.13 知 $M.Q_1 > D$，假设 (1) 成立。

假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par Zero) 此时有 $P = Q \mid \mathbf{0}$。对于假设 (1)，假定 $P > C$，这只能通过以下之一得到：

(Harden Par 1) 此时有 $Q > (\nu\vec{q})\langle Q'\rangle Q''$ 且 $C = (\nu\vec{q})\langle Q'\rangle (Q'' \mid \mathbf{0})$，由 $Q'' \mid \mathbf{0} \equiv Q''$ 知，假设 (1) 成立。

(Harden Par 2) 此时必须有规则使得 $\mathbf{0} > C$ 成立，但这显然是不可能用任何硬化规则得到的，因此该情况不会出现。

对于假设 (2)，假定 $Q > (\nu\vec{q})\langle Q'\rangle Q''$，则利用 **(Harden Par 1)** 得，$Q \mid \mathbf{0} > (\nu\vec{q})\langle Q'\rangle (Q'' \mid \mathbf{0})$，由 $Q'' \mid \mathbf{0} \equiv Q''$ 知假设 (2) 成立。

(Struct Par Comm) 此时有 $P = R_1 \mid R_2$ 且 $Q = R_2 \mid R_1$。对于假设 (1)，假定 $P = R_1 \mid R_2 > C$，这只能通过以下之一得到：

(Harden Par 1) 此时有 $R_1 > (\nu\vec{r})\langle R_1'\rangle R_1''$、$C = (\nu\vec{r})\langle R_1'\rangle (R_1'' \mid R_2)$ 且 $\{\vec{r}\} \cap fn(R_2) = \emptyset$。利用 **(Harden Par 2)** 可得，$Q = R_2 \mid R_1 > (\nu\vec{r})\langle R_1'\rangle (R_2 \mid R_1'')$，由 $R_1'' \mid R_2 \equiv R_2 \mid R_1''$ 知，假设 (1) 成立。

(Harden Par 2) 证明过程与 **(Harden Par 1)** 对称，此处略。

假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Par Assoc) 此时有 $P = (R_1 \mid R_2) \mid R_3$ 且 $Q = R_1 \mid (R_2 \mid R_3)$。对于假设 (1)，假定 $P = (R_1 \mid R_2) \mid R_3 > C$，这只能通过以下之一得到：

(Harden Par 1) 此时有 $(R_1 \mid R_2) > (\nu\vec{r})\langle R'\rangle R''$、$C = (\nu\vec{r})\langle R'\rangle (R'' \mid R_3)$ 且 $\{\vec{r}\} \cap fn(R_3) = \emptyset$。而 $(R_1 \mid R_2) > (\nu\vec{r})\langle R'\rangle R''$ 同样只能通过以下之一得到：

(Harden Par 1) 此时有 $R_1 > (\nu\vec{r})\langle R'\rangle R_1''$、$R'' = (R_1'' \mid R_2)$ 且 $\{\vec{r}\} \cap fn(R_2) = \emptyset$。结合 $\{\vec{r}\} \cap fn(R_3) = \emptyset$ 可知 $\{\vec{r}\} \cap fn(R_2 \mid R_3) = \emptyset$，利用 **(Harden Par 1)**，由 $R_1 > (\nu\vec{r})\langle R'\rangle R_1''$ 可得，$Q = R_1 \mid (R_2 \mid R_3) > (\nu\vec{r})\langle R'\rangle (R_1'' \mid (R_2 \mid R_3))$，而 $R'' \mid R_3 = (R_1'' \mid R_2) \mid R_3 \equiv R_1'' \mid (R_2 \mid R_3)$，因此假设 (1) 成立。

(Harden Par 2) 此时有 $R_2 > (\nu\vec{r})\langle R'\rangle R_2''$、$R'' = (R_1 \mid R_2'')$ 且 $\{\vec{r}\} \cap fn(R_1) = \emptyset$。利用 **(Harden Par 1)**，由 $\{\vec{r}\} \cap fn(R_3) = \emptyset$ 和 $R_2 > (\nu\vec{r})\langle R'\rangle R_2''$ 可得 $R_2 \mid R_3 > (\nu\vec{r})\langle R'\rangle (R_2'' \mid R_3)$，再根据条件 $\{\vec{r}\} \cap fn(R_1) = \emptyset$ 利用 **(Harden Par 1)** 可得 $Q = R_1 \mid (R_2 \mid R_3) > (\nu\vec{r})\langle R'\rangle (R_1 \mid (R_2'' \mid R_3))$，而 $R'' \mid R_3 = (R_1 \mid R_2'') \mid R_3 \equiv R_1 \mid (R_2'' \mid R_3)$，因此假设 (1) 成立。

(Harden Par 2) 证明过程与 **(Harden Par 1)** 对称，此处略。

假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res Zero) 此时 $P = (\nu n)\mathbf{0}$ 且 $Q = \mathbf{0}$。由于不存在 $P > C$ 和 $Q > D$，因此假设 (1)、(2) 直接成立。

(Struct Res Res) 此时有 $P = (\nu n)(\nu m)R$ 且 $Q = (\nu m)(\nu n)R$。这里假定 $n \neq m$，否则假设显然成立。对于假设 (1)，假定 $P > C$，这只能通过使用两次 **(Harden Res)** 得到，且有 $R > C'$ 且 $C = \overline{(\nu n)(\nu m)}C'$。使用两次 **(Harden Res)** 可得，$Q > \overline{(\nu m)(\nu n)}C'$。根据引理 B.5 知 $\overline{(\nu n)(\nu m)}C' \equiv \overline{(\nu m)(\nu n)}C'$，因此假设 (1) 成立。假设 (2) 的证明过程与假设 (1) 对称，此处略。

(Struct Res Par) 此时有 $P=(\nu n)(R_1 \mid R_2)$ 、 $Q=R_1 \mid (\nu n)R_2$ 且 $n \notin fn(R_1)$ 。对于假设 (1) ，假定 $P>C$ ，这只能通过 **(Harden Res)** 得到，且有 $C=\overline{(\nu n)}C'$ 且 $R_1 \mid R_2 > C'$ ，这里 $C'=(\nu \vec{r})\langle R' \rangle R''$ 。由于 $\vec{r}$ 为约束名向量，因此可设 $n \notin \{\vec{r}\}$ 。下面分情况讨论 $\overline{(\nu n)}C'$ 的具体定义。

- 若 $n \in fn(R')$ ，则 $R_1 \mid R_2 > (\nu \vec{r})\langle R' \rangle R''$ 必定通过以下之一得到：

  (Harden Par 1) 此时有 $R_1 > (\nu \vec{r})\langle R' \rangle R_1''$ 、 $R''=R_1'' \mid R_2$ 且 $\{\vec{r}\} \cap fn(R_2)=\emptyset$ 。由引理 4.4 可知 $R_1 \equiv (\nu \vec{r})(R' \mid R_1'')$ ，因此 $fn(R_1)=(fn(R') \cup fn(R_1'')) - \{\vec{r}\}$ ，由 $n \in fn(R')$ 和 $n \notin \{\vec{r}\}$ 可得 $n \in fn(R_1)$ 。这与已知条件矛盾，因此该情况不可能发生。

  (Harden Par 2) 此时有 $R_2 > (\nu \vec{r})\langle R' \rangle R_2''$ 、 $R''=R_1 \mid R_2''$ 且 $\{\vec{r}\} \cap fn(R_1)=\emptyset$ 。此时，分 (a) 、 (b) 两种情况讨论。 (a) ：如果 $R'=m[R_m]$ 、 $m \neq n$ 且 $n \notin fn(R'')$ ，即： $n \notin fn(R_1) \cup fn(R_2'')$ 且 $C=(\nu \vec{r})\langle m[(\nu n)R_m] \rangle (R_1 \mid R_2'')$ ，则由 **(Harden Res)** 可知 $(\nu n)R_2 > (\nu \vec{r})\langle m[(\nu n)R_m] \rangle R_2''$ ，再由 **(Harden Par 2)** 可得 $R_1 \mid (\nu n)R_2 > (\nu \vec{r})\langle m[(\nu n)R_m] \rangle (R_1 \mid R_2'')$ ，因此假设 (1) 成立。 (b) ：否则 $C=(\nu n, \vec{r})\langle R' \rangle R''$ 。由 **(Harden Res)** 可知 $(\nu n)R_2 > (\nu n, \vec{r})\langle R' \rangle R_2''$ ，再由 $n \notin fn(R_1)$ 利用 **(Harden Par 2)** 可得 $R_1 \mid (\nu n)R_2 > (\nu n, \vec{r})\langle R' \rangle (R_1 \mid R_2'')$ ，因此假设 (1) 成立。

- 若 $n \notin fn(R')$ ：则 $C=(\nu \vec{r})\langle R' \rangle (\nu n)R''$ 且 $R_1 \mid R_2 > (\nu \vec{r})\langle R' \rangle R''$ 必定通过以下之一得到：

  (Harden Par 1) 此时有 $R_1 > (\nu \vec{r})\langle R' \rangle R_1''$ 、 $R''=R_1'' \mid R_2$ 且 $\{\vec{r}\} \cap fn(R_2)=\emptyset$ 。由 $\{\vec{r}\} \cap fn(R_2)=\emptyset$ 知， $\{\vec{r}\} \cap fn((\nu n)R_2)=\emptyset$ ，因此利用 **(Harden Par 1)** 可得 $R_1 \mid (\nu n)R_2 > (\nu \vec{r})\langle R' \rangle (R_1'' \mid (\nu n)R_2)$ 。由引理 4.4 可知 $R_1 \equiv (\nu \vec{r})(R' \mid R_1'')$ ，因此 $fn(R_1)=(fn(R') \cup fn(R_1'')) - \{\vec{r}\}$ ，根据已知 $n \in fn(R_1)$ 和 $n \notin \{\vec{r}\}$ 可得 $n \in fn(R_1'')$ ，因此有 $(\nu n)R''=(\nu n)(R_1'' \mid R_2) \equiv R_1'' \mid (\nu n)R_2$ ，假设 (1) 成立。

  (Harden Par 2) 此时有 $R_2 > (\nu \vec{r})\langle R' \rangle R_2''$ 、 $R''=R_1 \mid R_2''$ 且 $\{\vec{r}\} \cap fn(R_1)=\emptyset$ 。由条件 $n \notin \{\vec{r}\} \cup fn(R')$ 利用 **(Harden Res)** 可知 $(\nu n)R_2 > (\nu \vec{r})\langle R' \rangle (\nu n)R_2''$ ，再由 $\{\vec{r}\} \cap fn(R_1)=\emptyset$ 利用 **(Harden Par 2)** 可得 $R_1 \mid (\nu n)R_2 > (\nu \vec{r})\langle R' \rangle (R_1 \mid (\nu n)R_2'')$ ，而已知 $n \notin fn(R_1)$ 保证了 $(\nu n)R''=(\nu n)(R_1 \mid R_2'') \equiv R_1 \mid (\nu n)R_2''$ ，因此假设 (1) 成立。

假设 (2) 的证明过程与假设 (1) 类似，此处略。

(Struct Res Amb) 此时有 $P=(\nu n)m[R]$ 、 $Q=m[(\nu n)R]$ 且 $n \neq m$ 。对于假设 (1) ，假定 $P>C$ ，这只能通过 **(Harden Res)** 得到，且有 $m[R] > C'$ 且 $C=\overline{(\nu n)}C'$ 。而前者只能通过 **(Harden Amb)** 得到，且有 $C'=(\nu)\langle m[R] \rangle \mathbf{0}$ ，这样 $C=\overline{(\nu n)}(\nu)\langle m[R] \rangle \mathbf{0}$ 。根据 n 是否属于 $fn(R)$ 分以下两种情况讨论：

- 若 $n \in fn(R)$ ，则 $C=(\nu)\langle m[(\nu n)R] \rangle \mathbf{0}$ 。由 **(Harden Amb)** 可知 $Q > (\nu)\langle m[(\nu n)R] \rangle \mathbf{0}$ ，假设 (1) 成立。
- 若 $n \notin fn(R)$ ，则 $C=(\nu)\langle m[R] \rangle (\nu n)\mathbf{0}$ 。由 **(Harden Amb)** 可知 $Q > (\nu)\langle m[(\nu n)R] \rangle \mathbf{0}$ ，然而由 $n \notin fn(R)$ 易知 $R \equiv (\nu n)R$ ，因此 $m[R] \mathrel{\widehat{=}} m[(\nu n)R]$ ，又由 **(Struct Res Zero)** 知 $(\nu n)\mathbf{0} \equiv \mathbf{0}$ ，因此假设 (1) 成立。

假设 (2) 的证明过程与假设 (1) 类似，此处略。

(Struct Empty) 此时 $P = \epsilon.Q$。对于假设 (1)，假定 $P = \epsilon.Q > C$，这只能通过 **(Harden Empty)** 得到，且 $Q > C$，因此假设 (1) 成立。对于假设 (2)，假定 $Q > C$，利用 **(Harden Empty)** 可得 $P = \epsilon.Q > C$，假设 (2) 成立。

(Struct Path) 此时有 $P = (M.M').R$ 且 $Q = M.(M'.R)$。

对于假设 (1)，假定 $P = (M.M').R > C$，这只能通过 **(Harden Path)** 得到，且有 $M.(M'.R) > C$，即 $Q > C$，因此假设 (1) 成立。

对于假设 (2)，假定 $Q = M.(M'.R) > D$，直接利用 **(Harden Path)** 即得 $P = (M.M').R > D$，假设 (1) 成立。

(Struct Repl Par) 此时有 $P = !R$ 且 $Q = R \mid !R$。

对于假设 (1)，假定 $P = !R > (\nu\vec{r})\langle R'\rangle R''$，这只能通过 **(Harden Repl)** 得到，且有 $R > (\nu\vec{r})\langle R'\rangle R'''$ 且 $R'' = R''' \mid !R$。由引理 4.4 可知 $!R \equiv (\nu\vec{r})(R' \mid R'')$，再利用引理 4.3 可得 $fn(!R) = (fn(R') \cup fn(R'')) - \{\vec{r}\}$，因此 $\{\vec{r}\} \cap fn(!R) = \emptyset$，利用 **(Harden Par 1)** 可得 $Q = R \mid !R > (\nu\vec{r})\langle R'\rangle (R''' \mid !R) = (\nu\vec{r})\langle R'\rangle R''$，因此假设 (1) 成立。

对于假设 (2)，假定 $Q = R \mid !R > (\nu\vec{r})\langle R'\rangle R''$，这只能通过以下之一得到：

(Harden Par 1) 此时有 $R > (\nu\vec{r})\langle R'\rangle R'''$、$R'' = R''' \mid !R$ 且 $\{\vec{r}\} \cap fn(!R) = \emptyset$，由 **(Harden Repl)** 可得 $!R > (\nu\vec{r})\langle R'\rangle R''' \mid !R = (\nu\vec{r})\langle R'\rangle R''$，假设 (2) 成立。

(Harden Par 1) 此时有 $!R > (\nu\vec{r})\langle R'\rangle R'''$、$R'' = R \mid R'''$ 且 $\{\vec{r}\} \cap fn(R) = \emptyset$。而 $!R > (\nu\vec{r})\langle R'\rangle R'''$ 只能由 **(Harden Repl)** 得到，且有 $R > (\nu\vec{r})\langle R'\rangle R_1$ 且 $R''' = R_1 \mid !R$，因此 $R''' = R_1 \mid !R \equiv R_1 \mid !R \mid R \equiv R \mid R''' \equiv R''$，假设 (2) 成立。

(Struct Repl Zero) 此时 $P = !\mathbf{0}$ 且 $Q = \mathbf{0}$。由于不存在 $P > C$ 和 $Q > D$，因此假设 (1)、(2) 直接成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

附录 3　引理 5.31 的证明

本附录证明简单上下文等价同余性中关于复制构造的引理。

引理 C.1　*如果* $!P > (\nu\vec{p})\langle Q\rangle R$，*则存在* P' *满足* $P > (\nu\vec{p})\langle Q\rangle P'$、$R = P' \mid !P$ *且* $\{\vec{p}\} \cap fn(P) = \emptyset$。

证明　结论 $!P > (\nu\vec{p})\langle Q\rangle R$ 能且仅能通过 **(Harden Repl)** 规则得到，且有前提 $P > (\nu\vec{p})\langle Q\rangle P'$ 且 $R = P' \mid !P$。由引理 4.3 和引理 4.4 可知 $fn(P) = fn((\vec{p})(Q \mid P'))$，因此 $\{\vec{p}\} \cap fn(P) = \emptyset$。∎

引理 C.2　*如果* $!P \xrightarrow{A} Q$，*则存在* R *满足* $P \xrightarrow{A} R$ *且* $Q \equiv R \mid !P$。

证明　结论 $!P \xrightarrow{A} Q$ 能且仅能通过 **(Trans Cap)** 规则得到，且有前提 $!P > (\nu\vec{p})\langle A\,.\,P'\rangle P''$、$fn(A) \cap \{\vec{p}\} = \emptyset$ 且 $Q = (\nu\vec{p})(P' \mid P'')$。由引理 C.1 可知，存在 P''' 使得 $P > (\nu\vec{p})\langle A\,.\,P'\rangle P'''$、$P'' = P''' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$。令 $R = (\nu\vec{p})(P' \mid P''')$，则由 **(Trans Cap)** 知 $P \xrightarrow{A} R$ 且 $Q = (\nu\vec{p})(P' \mid P'') = (\nu\vec{p})(P' \mid P''' \mid !P) \equiv R \mid !P$。∎

引理 C.3　*如果* $!P \xrightarrow{\tau} Q$，*则存在* R *满足* $P \mid P \xrightarrow{\tau} R$ *且* $Q \equiv R \mid !P$。

证明　对 $!P \xrightarrow{\tau} Q$ 的推导过程进行归纳。

(Trans Amb) 此时有 $!P > (\nu\vec{p})\langle n[P_1]\rangle P_2$、$P_1 \xrightarrow{\tau} P_1'$ 且 $Q = (\nu\vec{p})(n[P_1'] \mid P_2)$。由 $!P > (\nu\vec{p})\langle n[P_1]\rangle P_2$ 利用引理 C.1 可知存在 R' 满足 $P > (\nu\vec{p})\langle n[P_1]\rangle R'$、$P_2 = R' \mid !P$ 且 $fn(P) \cap \{\vec{p}\} = \emptyset$。再利用 **(Harden Par 1)** 可得 $P \mid P > (\nu\vec{p})\langle n[P_1]\rangle (R' \mid P)$。再由 **(Trans Amb)** 可知 $P \mid P \xrightarrow{\tau} R$，这里 $R = (\nu\vec{p})(n[P_1'] \mid R' \mid P)$。又由 $Q = (\nu\vec{p})(n[P_1'] \mid P_2) = (\nu\vec{p})(n[P_1'] \mid R' \mid !P) \equiv (\nu\vec{p})(n[P_1'] \mid R' \mid P) \mid !P = R \mid !P$，因此，结论成立。

(Trans In) 此时有 $!P > (\nu\vec{p})\langle n[P_1]\rangle P_2$、$P_1 \xrightarrow{\overline{\text{in}}\, m} P_1'$、$P_2 > (\nu\vec{r})\langle m[P_3]\rangle P_4$、$P_3 \xrightarrow{\text{in}\, n} P_3'$、$\{\vec{r}\} \cap fn(n[P_1]) = \emptyset$、$\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 且 $Q = (\nu\vec{p},\vec{r})(n[m[P_3'] \mid P_1'] \mid P_4)$。由 $!P > (\nu\vec{p})\langle n[P_1]\rangle P_2$ 利用引理 C.1 可知存在 R' 使得 $P > (\nu\vec{p})\langle n[P_1]\rangle R'$、$P_2 \equiv R' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$。由 $P_2 > (\nu\vec{r})\langle m[P_3]\rangle P_4$ 和 $P_2 \equiv R' \mid !P$ 利用引理 4.14 可知存在 $R' \mid !P > (\nu\vec{r})\langle m[P_5]\rangle P_6$ 且 $P_3 \equiv P_5$、$P_4 \equiv P_6$。然而，只有以下两种情况可以推导出 $R' \mid !P > (\nu\vec{r})\langle m[P_5]\rangle P_6$：

(Harden Par 1) 此时有 $R' > (\nu\vec{r})\langle m[P_5]\rangle P_7$、$P_6 = P_7 \mid !P$ 且 $\{\vec{r}\} \cap fn(\ !P) = \emptyset$。(1) 由 $P > (\nu\vec{p})\langle n[P_1]\rangle R'$ 和 $\{\vec{p}\} \cap fn(P) = \emptyset$ 利用 **(Harden Par 1)** 可知 $P \mid P > (\nu\vec{p})\langle n[P_1]\rangle (R' \mid P)$。(2) 再由 $R' > (\nu\vec{r})\langle m[P_5]\rangle P_7$ 和 $\{\vec{r}\} \cap fn(P) = \{\vec{r}\} \cap fn(\ !P) = \emptyset$ 利用 **(Harden Par 1)** 可得 $R' \mid P > (\nu\vec{r})\langle m[P_5]\rangle (P_7 \mid P)$。(3) 再由 $P_5 \equiv P_3$ 和 $P_3 \xrightarrow{\text{in}\, n} P_3'$ 利用引理 4.15 可知存在 P_5' 使得 $P_5 \xrightarrow{\text{in}\, n} P_5'$ 且 $P_5' \equiv P_3'$。综合 (1),(2),(3) 和已知条件 $P_1 \xrightarrow{\overline{\text{in}}\, m} P_1'$、$\{\vec{r}\} \cap fn(n[P_1]) = \emptyset$、$\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 利用 **(Trans In)** 可得 $P \mid P \xrightarrow{\tau} (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid P_7 \mid P) = R$，而由 $fn(P) \cap \{\vec{p},\vec{r}\} = \emptyset$ 易知 $Q = (\nu\vec{p},\vec{r})(n[m[P_3'] \mid P_1'] \mid P_4) \equiv (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid P_6) \equiv (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid P_7 \mid !P) \equiv R \mid !P$，因此结论成立。

(Harden Par 2) 此时有 $!P > (\nu\vec{r})\langle m[P_5]\rangle P_7$、$P_6 = R' \mid P_7$ 且 $\{\vec{r}\} \cap fn(R') = \emptyset$。由 $!P > (\nu\vec{r})\langle m[P_5]\rangle P_7$ 利用引理 C.1 可知存在 R'' 使得 $P > (\nu\vec{r})\langle m[P_5]\rangle R''$、$P_7 = R'' \mid !P$

且 $\{\vec{p}\} \cap fn(P) = \emptyset$。(1) 由 $P > (\nu\vec{p})\langle n[P_1]\rangle R'$ 和 $\{\vec{p}\} \cap fn(P) = \emptyset$ 利用 **(Harden Par 1)** 可知 $P \mid P > (\nu\vec{p})\langle n[P_1]\rangle (R' \mid P)$。(2) 再由 $P > (\nu\vec{r})\langle m[P_5]\rangle R''$ 和 $\{\vec{r}\} \cap fn(R') = \emptyset$ 利用 **(Harden Par 2)** 可得 $R' \mid P > (\nu\vec{r})\langle m[P_5]\rangle (R' \mid R'')$。(3) 再由 $P_5 \equiv P_3$ 和 $P_3 \xrightarrow{\text{in } n} P_3'$ 利用引理 4.15 可知存在 P_5' 使得 $P_5 \xrightarrow{\text{in } n} P_5'$ 且 $P_5' \equiv P_3'$。综合 (1),(2),(3) 和已知条件 $P_1 \xrightarrow{\overline{\text{in}}\, m} P_1'$、$\{\vec{r}\} \cap fn(n[P_1]) = \emptyset$、$\{\vec{r}\} \cap \{\vec{p}\} = \emptyset$ 利用 **(Trans In)** 可得 $P \mid P \xrightarrow{\tau} (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid R' \mid R'') = R$，而由 $fn(P) \cap \{\vec{p},\vec{r}\} = \emptyset$ 易知 $Q = (\nu\vec{p},\vec{r})(n[m[P_3'] \mid P_1'] \mid P_4) \equiv (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid P_6) \equiv (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid R' \mid P_7) \equiv (\nu\vec{p},\vec{r})(n[m[P_5'] \mid P_1'] \mid R' \mid R'' \mid !P) \equiv R \mid !P$，因此结论成立。

(Trans Out)(Trans Open) 采用类似 **(Trans In)** 的方法易知假设成立，此处略。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 C.4 如果 $H(\,!P) \longrightarrow R$，则存在 H' 满足 $R \equiv H'(\,!P)$ 且对任意非负整数 k 都有 $H(P^{k+2}) \longrightarrow H'(P^k)$。

证明 由定理 5.20 可知，$H(\,!P) \longrightarrow R$ 意味着必有以下情况之一成立：

(Act Proc) 此时有 $!P \longrightarrow P'$ 且 $R \equiv H(P')$。由 $!P \longrightarrow P'$ 利用引理 C.3 和定理 4.17 可知存在 Q 使得 $P \mid P \longrightarrow Q$ 且 $P' \equiv Q \mid !P$。令 $H' = H(Q \mid -)$，有 $R \equiv H(Q \mid !P) = H'(\,!P)$。另外，对任意非负整数 k 有 $P \mid P \mid P^k \longrightarrow Q \mid P^k$，因此有 $H(P^{k+2}) \longrightarrow H(Q \mid P^k)$，即 $H(P^{k+2}) \longrightarrow H'(P^k)$，结论成立。

(Act Har) 此时有 $H \longrightarrow H''$ 且 $R \equiv H''(\,!P)$。令 $H' = H''(P \mid P \mid -)$，则 $R \equiv H''(P \mid P \mid !P) = H'(\,!P)$，且对任意非负整数 k 有 $H(P^{k+2}) \longrightarrow H''(P^{k+2}) = H'(P^k)$，结论成立。

(Act Inter) 此时，存在 H_0 和 $\vec{r}$ 满足 $\{\vec{r}\} \cap fn(P) = \emptyset$ 且以下之一成立：

(Inter In) 此时有 $H \equiv (\nu\vec{r})H_0(m[- \mid R_1] \mid n[R_2])$、$!P \xrightarrow{\text{in } n} P'$、$R_2 \xrightarrow{\overline{\text{in}}\, m} R_2'$ 且 $R \equiv (\nu\vec{r})H_0(n[m[P' \mid R_1] \mid R_2'])$。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\text{in } n} Q$ 且 $P' \equiv Q \mid !P$。令 $H' = (\nu\vec{r})H_0(n[m[- \mid P \mid Q \mid R_1] \mid R_2'])$，则 $R \equiv (\nu\vec{r})H_0(n[m[Q \mid !P \mid R_1] \mid R_2']) \equiv (\nu\vec{r})H_0(n[m[\,!P \mid P \mid Q \mid R_1] \mid R_2']) = H'(\,!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(m[P^k \mid P \mid P \mid R_1] \mid n[R_2]) \longrightarrow (\nu\vec{r})H_0(n[m[P^k \mid P \mid Q \mid R_1] \mid R_2']) = H'(P^k)$，结论成立。

(Inter Out) 此时有 $H \equiv (\nu\vec{r})H_0(n[m[- \mid R_1] \mid R_2])$、$!P \xrightarrow{\text{out } n} P'$、$R_2 \xrightarrow{\overline{\text{out}}\, m} R_2'$ 且 $R \equiv (\nu\vec{r})H_0(m[P' \mid R_1] \mid n[R_2'])$。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\text{out } n} Q$ 且 $P' \equiv Q \mid !P$。令 $H' = (\nu\vec{r})H_0(m[- \mid P \mid Q \mid R_1] \mid n[R_2'])$，则 $R \equiv (\nu\vec{r})H_0(m[Q \mid !P \mid R_1] \mid n[R_2']) \equiv (\nu\vec{r})H_0(m[\,!P \mid P \mid Q \mid R_1] \mid n[R_2']) = H'(\,!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(n[m[P^k \mid P \mid P \mid R_1] \mid R_2]) \longrightarrow (\nu\vec{r})H_0(m[P^k \mid P \mid Q \mid R_1] \mid n[R_2']) = H'(P^k)$，结论成立。

(Inter Open) 此时有 $H \equiv (\nu\vec{r})H_0(- \mid n[R_1])$、$!P \xrightarrow{\text{open } n} P'$、$R_1 \xrightarrow{\overline{\text{open}}} R_1'$ 且 $R \equiv (\nu\vec{r})H_0(P' \mid R_1')$。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\text{open } n} Q$ 且 $P' \equiv Q \mid !P$。令 $H' = (\nu\vec{r})H_0(- \mid P \mid Q \mid R_1')$，则 $R \equiv (\nu\vec{r})H_0(Q \mid !P \mid R_1') \equiv (\nu\vec{r})H_0(\,!P \mid P \mid Q \mid R_1') = H'(\,!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(P^k \mid P \mid P \mid n[R_1]) \longrightarrow (\nu\vec{r})H_0(P^k \mid P \mid Q \mid R_1') = H'(P^k)$，结论成立。

(Inter Co-in) 此时有 $H \equiv (\nu\vec{r})H_0(m[R_1] \mid n[- \mid R_2])$ 、 $!P \xrightarrow{\overline{\text{in}}\, m} P'$ 、 $R_1 \xrightarrow{\text{in}\, n} R_1'$ 且 $R \equiv (\nu\vec{r})H_0(n[m[R_1'] \mid P' \mid R_2])$ 。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\overline{\text{in}}\, m} Q$ 且 $P' \equiv Q \mid !P$ 。令 $H' = (\nu\vec{r})H_0(n[m[R_1'] \mid - \mid P \mid Q \mid R_2])$ ，则 $R \equiv (\nu\vec{r})H_0(n[m[R_1'] \mid Q \mid !P \mid R_2]) \equiv (\nu\vec{r})H_0(n[m[R_1'] \mid !P \mid P \mid Q \mid R_2]) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(m[R_1] \mid n[P^k \mid P \mid P \mid R_2]) \longrightarrow (\nu\vec{r})H_0(n[m[R_1'] \mid P^k \mid P \mid Q \mid R_2]) = H'(P^k)$ ，结论成立。

(Inter Co-out) 此时有 $H \equiv (\nu\vec{r})H_0(n[m[R_1] \mid - \mid R_2])$ 、 $!P \xrightarrow{\overline{\text{out}}\, m} P'$ 、 $R_1 \xrightarrow{\text{out}\, n} R_1'$ 且 $R \equiv (\nu\vec{r})H_0(m[R_1'] \mid n[P' \mid R_2])$ 。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\overline{\text{out}}\, m} Q$ 且 $P' \equiv Q \mid !P$ 。令 $H' = (\nu\vec{r})H_0(m[R_1'] \mid n[- \mid P \mid Q \mid R_2])$ ，则 $R \equiv (\nu\vec{r})H_0(m[R_1'] \mid n[Q \mid !P \mid R_2]) \equiv (\nu\vec{r})H_0(m[R_1'] \mid n[\,!P \mid P \mid Q \mid R_2]) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(n[m[R_1] \mid P^k \mid P \mid P \mid R_2]) \longrightarrow (\nu\vec{r})H_0(m[R_1'] \mid n[P^k \mid P \mid Q \mid R_2]) = H'(P^k)$ ，结论成立。

(Inter Co-open) 此时有 $H \equiv (\nu\vec{r})H_0(n[- \mid R_1] \mid R_2)$ 、 $!P \xrightarrow{\overline{\text{open}}} P'$ 、 $R_2 \xrightarrow{\text{open}\, n} R_2'$ 且 $R \equiv (\nu\vec{r})H_0(P' \mid R_1 \mid R_2')$ 。由引理 C.2 可知存在 Q 满足 $P \xrightarrow{\overline{\text{open}}} Q$ 且 $P' \equiv Q \mid !P$ 。令 $H' = (\nu\vec{r})H_0(- \mid P \mid Q \mid R_1 \mid R_2')$ ，则 $R \equiv (\nu\vec{r})H_0(Q \mid !P \mid R_1 \mid R_2') \equiv (\nu\vec{r})H_0(\,!P \mid P \mid Q \mid R_1 \mid R_2') = H'(!P)$ 且对任意非负整数 k 有 $H(P^{k+2}) \equiv (\nu\vec{r})H_0(n[P^k \mid P \mid P \mid R_1] \mid R_2) \longrightarrow (\nu\vec{r})H_0(P^k \mid P \mid Q \mid R_1 \mid R_2') = H'(P^k)$ ，结论成立。

(Inter Amb In) 此时有 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、$Q \xrightarrow{\text{in}\, m} Q'$ 、$H \equiv (\nu\vec{r})H_0(- \mid m[R_1])$ 、$R_1 \xrightarrow{\overline{\text{in}}\, n} R_1'$ 、$\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(m[n[Q'] \mid R_1'] \mid P'))$ 。由 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 利用引理 C.1 可知存在 P'' 满足 $P > (\nu\vec{p})\langle n[Q]\rangle P''$ 、$P' = P'' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$ 。令 $H' = (\nu\vec{r})H_0(- \mid P \mid (\nu\vec{p})(m[n[Q'] \mid R_1'] \mid P''))$ ，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(m[n[Q'] \mid R_1'] \mid P'' \mid !P)) \equiv (\nu\vec{r})H_0(\,!P \mid P \mid (\nu\vec{p})(m[n[Q'] \mid R_1'] \mid P'')) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(P^k \mid P \mid P \mid m[R_1]) \longrightarrow (\nu\vec{r})H_0(P^k \mid P \mid (\nu\vec{p})(m[n[Q'] \mid R_1'] \mid P'')) = H'(P^k)$ ，结论成立。

(Inter Amb Co-in) 此时有 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、$Q \xrightarrow{\overline{\text{in}}\, m} Q'$ 、$H \equiv (\nu\vec{r})H_0(- \mid m[R_1])$ 、$R_1 \xrightarrow{\text{in}\, n} R_1'$ 、$\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P'))$ 。由 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 利用引理 C.1 可知存在 P'' 满足 $P > (\nu\vec{p})\langle n[Q]\rangle P''$ 、$P' = P'' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$ 。令 $H' = (\nu\vec{r})H_0(- \mid P \mid (\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P''))$ ，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P'' \mid !P)) \equiv (\nu\vec{r})H_0(\,!P \mid P \mid (\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P'')) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(P^k \mid P \mid P \mid m[R_1]) \longrightarrow (\nu\vec{r})H_0(P^k \mid P \mid (\nu\vec{p})(n[m[R_1'] \mid Q'] \mid P'')) = H'(P^k)$ ，结论成立。

(Inter Amb Out 1) 此时有 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 、$Q \xrightarrow{\text{out}\, m} Q'$ 、$H \equiv (\nu\vec{r})H_0(m[- \mid R_1])$ 、$R_1 \xrightarrow{\overline{\text{out}}\, n} R_1'$ 、$\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P' \mid R_1']))$ 。由 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 利用引理 C.1 可知存在 P'' 满足 $P > (\nu\vec{p})\langle n[Q]\rangle P''$ 、$P' = P'' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$ 。令 $H' = (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[- \mid P \mid P'' \mid R_1']))$ ，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P'' \mid !P \mid R_1'])) \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[\,!P \mid P \mid P'' \mid R_1'])) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(m[P^k \mid P \mid P \mid R_1]) \longrightarrow (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P^k \mid P \mid P'' \mid R_1'])) = H'(P^k)$ ，结论成立。

(Inter Amb Out 2) 此时有 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$、$Q \xrightarrow{\mathtt{out}\ m} Q'$、$P' \xrightarrow{\overline{\mathtt{out}}\ n} P''$、$H \equiv (\nu\vec{r})H_0(m[-\mid R_1])$、$\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P'' \mid R_1]))$。由 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 利用引理 C.1 可知存在 P_1 满足 $P > (\nu\vec{p})\langle n[Q]\rangle P_1$、$P' = P_1 \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$。由 $P' = P_1 \mid !P \xrightarrow{\overline{\mathtt{out}}\ n} P''$ 利用引理 5.18 可知必有以下情况之一成立：

- $P_1 \xrightarrow{\overline{\mathtt{out}}\ n} P_1'$ 且 $P'' \equiv P_1' \mid !P$：此时令 $H' = (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[- \mid P \mid P_1' \mid R_1]))$，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P_1' \mid !P \mid R_1])) \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[!P \mid P \mid P_1' \mid R_1])) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(m[P^k \mid P \mid P \mid R_1]) \longrightarrow (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P^k \mid P \mid P_1' \mid R_1])) = H'(P^k)$，结论成立。
- $!P \xrightarrow{\overline{\mathtt{out}}\ n} P_2$ 且 $P'' \equiv P_1 \mid P_2$：此时有根据引理 C.2 可知存在 P_2' 满足 $P \xrightarrow{\overline{\mathtt{out}}\ n} P_2'$ 且 $P_2 \equiv P_2' \mid !P$。令 $H' = (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[- \mid P_1 \mid P_2' \mid R_1]))$，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P_1 \mid P_2' \mid !P \mid R_1])) \equiv H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(m[P^k \mid P \mid P \mid R_1]) \longrightarrow (\nu\vec{r})H_0((\nu\vec{p})(n[Q'] \mid m[P^k \mid P_1 \mid P_2' \mid R_1])) = H'(P^k)$，结论成立。

(Inter Amb Co-open) 此时有 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$、$Q \xrightarrow{\overline{\mathtt{open}}} Q'$、$H \equiv (\nu\vec{r})H_0(- \mid R_1)$、$R_1 \xrightarrow{\mathtt{open}\ n} R_1'$、$\{\vec{p}\} \cap fn(m[R_1]) = \emptyset$ 且 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(Q' \mid P' \mid R_1'))$。由 $!P > (\nu\vec{p})\langle n[Q]\rangle P'$ 利用引理 C.1 可知存在 P'' 满足 $P > (\nu\vec{p})\langle n[Q]\rangle P''$、$P' = P'' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$。令 $H' = (\nu\vec{r})H_0(- \mid P \mid (\nu\vec{p})(Q' \mid P'' \mid R_1'))$，则 $R \equiv (\nu\vec{r})H_0((\nu\vec{p})(Q' \mid P'' \mid !P \mid R_1')) \equiv (\nu\vec{r})H_0(!P \mid P \mid (\nu\vec{p})(Q' \mid P'' \mid R_1')) = H'(!P)$ 且对任意非负整数 k 有 $H(P^{K+2}) \equiv (\nu\vec{r})H_0(P^k \mid P \mid P \mid R_1) \longrightarrow (\nu\vec{r})H_0(P^k \mid P \mid (\nu\vec{p})(Q' \mid P'' \mid R_1')) = H'(P^k)$，结论成立。

通过以上对所有可能情况的分析可知，结论成立。 ∎

引理 5.31 如果 $H(!P) \Downarrow_n$，那么存在整数 k 使得 $H(P^k) \Downarrow_n$。

证明 对 $H(!P) \Downarrow_n$ 的推导过程进行归纳。

(Conv Exb) 此时有 $H(!P) \downarrow_n$。由引理 5.28 可知以下之一成立；

(1). 对任意 Q，$H(Q) \downarrow_n$：此时取 $k = 1$ 即有 $H(P) \downarrow_n$。

(2). $!P \downarrow_n$ 且对任意 $Q \downarrow_n$ 有 $H(Q) \downarrow_n$：此时由引理 5.26 可知存在 $\vec{p}$、P' 和 P'' 使得 $!P > (\nu n[P'])\langle P\rangle''$ 且 $n \notin \{\vec{p}\}$。由引理 C.1 可得存在 P''' 满足 $P > (\nu\vec{p})\langle n[P']\rangle P'''$、$P'' = P''' \mid !P$ 且 $\{\vec{p}\} \cap fn(P) = \emptyset$。再由引理 5.26 可得 $P \downarrow_n$。取 $k = 1$ 即有 $H(P) \downarrow_n$。

(3). $!P \xrightarrow{A} P'$、$A \in \mathbf{BarbedAction}$ 且对任意 $Q \xrightarrow{A'} Q'$、$A' \in \mathbf{BarbedAction}$ 有 $H(Q) \downarrow_n$，由引理 C.2 可知存在 R 满足 $P \xrightarrow{A} R$，因此取 $k = 1$ 即有 $H(P) \downarrow_n$。

(Conv Red) 此时有 $H(!P) \longrightarrow Q$ 且 $Q \Downarrow_n$。由引理 C.4 可知存在 H' 满足 $Q \equiv H'(!P)$ 且对任意非负整数 j 有 $H(P^{j+2}) \longrightarrow H'(P^j)$。由 $Q \Downarrow_n$ 且 $Q \equiv H'(!P)$ 利用引理 5.6 可知 $H'(!) \Downarrow_n$，且该结果可通过与 $Q \Downarrow_n$ 同样的推导过程得到。由归纳假设知，存在整数 k 使得 $H'(P^k) \Downarrow_n$，再由已知 $H(P^{k+2}) \longrightarrow H'(P^k)$ 利用 **(Conv Red)** 可知 $H(P^{k+2}) \Downarrow_n$。

通过以上对所有可能情况的分析可知，结论成立。 ∎

致　　谢

在论文完成之际，我首先要特别感谢我的导师尤晋元教授。三年前，我有幸能成为尤教授领导的上海分布计算技术中心的一份子。在此期间，尤教授一直在学习、科研和生活上给予我们最大可能的帮助。在学术上，尤教授始终坚持高标准、严要求的指导方针，鼓励我们大胆探索、勇于创新，支持我们向国内外最高级别的学术刊物和会议上投稿，并不时给我们提供大量的信息。在工作条件上，尤教授尽可能的完善实验室的科研设备和工作环境，为我们每人配备了最好的电脑和宽敞的工作区域，使我们能够毫无顾忌地从事学术上的钻研和探索。在生活上，尤教授时常问寒问暖，每逢节日实验室都要组织一起聚餐和活动。能够遇上尤教授作为我的导师，是我毕生的幸福。如果没有尤教授的指导和帮助，难以想象如何可以在有限的时间内了解最新的学术动态、把握发展的方向、并把自己的想象变成现实。

其次我要感谢孙永强教授。是孙教授的函数式程序设计语言课程把我带进了神奇的计算机程序理论世界。同时，孙教授在我从事研究和论文写作的过程中也给了我无私的帮助，孙教授对我的勉励和鞭策为我进一步的研究探索带来了强大的动力。

我还要感谢傅育熙教授。傅教授以他深厚的学术功底和大量高质量的学术论文创作象指路的灯塔一般使我在计算机科学理论海洋中航行中看到前进的希望，告诉我中国人也可以在国际计算机学术研究中占据一席之地。与傅教授的合作更进一步使我对傅教授的博学、高产和严谨的科学作风万分敬佩。

我还要感谢分布计算技术中心的荣震华教授、姜丽红教授、饶若楠教授和陈英老师对我学术上和工作上的帮助。

另外，我还要感谢 Pascal Zimmer 博士对我在理解 π 演算翻译过程和论文书写上的帮助，以及其他仅仅通过 Email 接触过的国内外学者，特别是 Roland Backhouse 教授和 Assaf J. Kfoury 教授身上所体现的那种严谨治学和崇尚科学真理的精神对我的鼓励。

感谢朱鹏师兄、杨萍师姐、王斌师兄、毛卫良师兄对我在学习和科研上的帮助和实验室的刘福岩、刘琼波、俞茂元、夏顺东、费斐、张炜权等师兄弟对我各方面的帮助。

最后，我要感谢我的亲人。感谢我的妻子对我的不懈鼓励和无比的爱，在每一次挫折时让我看到前路的光明，在每一次成功时让我认清自己的渺小，多年来一直与我共同分享的每一次成功的喜悦，并肩分担每次失败的创伤。感谢我的母亲一直以来对我的理解和关怀，是她甘愿忍受这十年多的骨肉分离把我送进大学，是她永远在家乡为我默默祝福并竭尽全力让我没有牵挂。没有她们的爱，我不可能有今天。

攻读学位期间发表的学术论文目录

(1). 管旭东，杨怡玲，尤晋元．灰箱演算中强干扰问题的进一步控制．软件学报．录用．

(2). Xudong Guan, Yiling Yang, Jinyuan You. Typing evolving ambients. Info. Proc. Letters. 80(5):265-270, 2001, Elsevier.

(3). Xudong Guan, Yiling Yang, Jinyuan You. Making ambients more robust. In Proc. Int'l. Conf. on Software: Theory and Practice, pp.377-384, Beijing, China, Aug. 2000.

(4). Xudong Guan, Yiling Yang, Jinyuan You. POM - A mobile agent security model against malicious hosts. In Proc. HPC-Asia'2000, pp.1165-1166, Beijing, China, May 2000.

(5). Yiling Yang, Xudong Guan, Jinyuan You. Improving the interestingness of web usage mining. J. Shanghai Jiao Tong Univ.(English), 2001. To appear.

(6). 杨怡玲，管旭东，尤晋元．基于页面内容和站点结构的页面聚类挖掘算法．软件学报．录用．

(7). 杨怡玲，管旭东，陆丽娜，尤晋元． Web 日志挖掘预处理中的 Frame 页面过滤算法，计算机工程， 27(2): 76-77, 2001.

(8). 杨怡玲，管旭东，尤晋元．一个简单的 Web 日志挖掘系统．上海交通大学学报， 34(7):932-935, 2000.

(9). Yiling Yang, Xudong Guan, Jinyuan You. Frame filtering in the data preprocessing for web usage mining. In Proc. Int'l. Conf. on Intelligent Info. Processing, pp. 507-511, Beijing, China, Aug. 2000.

(10). Yiling Yang, Xudong Guan, Jinyuan You. Enhanced algorithm for mining frequently visited page groups. In Proc. ICDCS'2000 Workshops, pp. F54-F57, Taipei, Taiwan, April, 2000.