上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目： On Uniform Approach to Random Process Model

学生姓名： 孙晨鸽

学生学号： 516030910421

专　　业： 计算机科学与技术

指导教师： 傅育熙教授

学院 (系)： 计算机科学与工程系

ON UNIFORM APPROACH TO RANDOM PROCESS MODEL

摘要

并发理论是理论计算机科学中一个活跃的研究领域。随着大规模通讯系统的迅速发展，并发理论成为建模和表征现实并发系统的重要方法论。作为经典并发理论的扩展，概率进程被广泛研究，产生了很多用于不同的应用场景的各种变体。2019 年，傅育熙教授提出了一个对并发进程模型进行概率化扩展的通用方法，这一方法具有模型无关性的优点。在本文中，我们在这一通用方法的框架下对傅教授的工作进行了扩展，提出了一个传值进程演算的随机版本——随机传值进程模型。这一模型是使用傅教授的通用方法对经典传值进程演算的概率扩展。首先，我们规范化了随机传值进程模型的语法和转移语义并研究了这一模型的代数性质，例如互模拟关系。我们还证明了这一模型等价关系的同余性。其次，我们验证了随机传值进程模型可以用于具有传值特点的现实问题的建模和分析。作为应用案例，我们使用随机传值进程模型有效地建模并模拟实现了基于云计算协议 Gossip-Style Membership 协议的通信过程，证明了随机传值进程模型对于并发通信过程的建模和分析具有一定的可行性。我们的工作是傅教授的通用方法在理论和应用层面上的延伸。

关键词：并发理论，随机进程，并发传值进程，互模拟关系，观察等价性

ON UNIFORM APPROACH TO RANDOM PROCESS MODEL

ABSTRACT

Concurrency theory has been an active field of research in theoretical computer science. Recently, with the rapid development of massive communication systems, concurrency theory has become an important methodology for modeling and characterizing real concurrent systems. As an extension of classic concurrency theory, probabilistic processes have been widely studied for many years and led to lots of variants for different applications. In 2019, Yuxi Fu has proposed a uniform approach to study the probabilistic extension of concurrency processes which has the merit of being model-independent. In this work, we first extend Yuxi's original work by introducing a random version of value-passing calculus under the uniform approach framework. This new model is a randomized extension of the classic value-passing calculus. In this paper, we formalize the grammar and transition semantics of the random value-passing calculus, as well as study its algebraic properties such as bisimulation relation. We show that the new equivalence relation is congruent. Second, we show that random value-passing calculus is especially suitable for modeling and analyzing real-world applications with value-passing characteristics. As a case study, we use random value-passing calculus to efficiently model and implement a well-known communication system based on the Gossip-Style Membership Protocol, which is a cloud computing protocol. This shows that our new model is suitable for formalizing and analyzing modern concurrent communication systems. Our work extends the uniform approach to random process model in both theoretical and application aspects.

Key words: concurrency theory, probabilistic process, value-passing calculus, bisimulation, observation equivalence

目　录

第一章 绪论 …… 1
1.1 通信系统演算 …… 1
1.2 随机进程模型 …… 2
1.3 研究目的 …… 2
1.4 论文结构 …… 2

第二章 随机传值进程模型 …… 3
2.1 传值进程模型 …… 3
2.1.1 The Value-Passing Calculus …… 3
2.1.2 If Then Else 语法符号化 …… 5
2.2 随机传值进程模型 …… 5
2.3 随机传值进程模型中的互模拟关系 …… 7
2.3.1 互模拟关系与观察等价性 …… 7
2.3.2 条件等价树 …… 9
2.3.3 随机传值进程模型的符号互模拟 …… 12
2.4 随机传值进程模型的等价性 …… 15
2.5 本章小结 …… 20

第三章 随机传值进程模型的应用 …… 21
3.1 Gossip-Style Membership 协议 …… 21
3.1.1 Gossip 协议 …… 21
3.1.2 组成员协议 …… 21
3.2 基于 Gossip-Style Membership 协议的通信系统的实现 …… 22
3.2.1 基于 Gossip 协议的通信系统的实现 …… 22
3.2.2 Gossip 协议的等价关系 …… 24
3.2.3 基于 Gossip-Style Membership 协议的通信系统的实现 …… 26
3.3 基于 Gossip-Style Membership 协议的通信系统的仿真模拟 …… 29
3.3.1 Go 语言与 CSP …… 29
3.3.2 代码实现与仿真效果 …… 30
3.4 本章小结 …… 32

第四章 总结与展望 …… 33

参考文献 …… 35

附录 A 基于 Gossip-Style Membership 协议的通信系统的 Go 语言实现 …… 38

致　谢 …… 43

第一章　绪论

并发理论是理论计算机科学中一个活跃的研究领域。随着大规模通讯系统的迅速发展，并发理论成为建模和表征现实并发系统的重要方法论。并发理论的研究内容包括：如何刻画并行进程的行为，在什么情况下他们可以互相模拟，研究各种通信和同步机制和死锁、可观察性、发散性等并发现象。对并发理论的研究加深了人们对并发系统的认识，其主要的研究成果已经在 Ada，Java 等编程语言中得到广泛应用[1]。在计算机科学中，进程演算（或进程代数）是用于形式化建模并发系统的多种相关方法。进程演算提供了具体描述多个独立程序或者是多个进程之间交互、通信、同步的方法，其中包含了对进程操作和分析的描述、以及证明形式化推导进程之间存在等价关系的代数法则[2]。关于进程演算的典例主要包括 MILNER R 提出的通信系统演算 CCS(Calculus of Communicating System)[3]，HOARE C 的通信顺序进程 CSP(Communicating Sequential Process)[4]，BERGSTRA J 等的 ACP(Algebra of Communicating Processes)[5]，以及 BOLOGNESI T 等的 LOTOS[6]。作为描述并发系统的模型，进程演算受到广泛研究并被成功应用到实际系统的规范、设计、分析及验证中。

随机在现代计算机科学的研究中日趋重要，在对并发系统的研究中也备受关注。由于现代计算机系统具有开放性、分布式、交互式的特点，并发系统中通常包含复杂的行为：非确定性行为和随机性行为。为了使用简单、易用的形式化方法描述复杂的并发系统，并对并发系统进行建模和分析，我们通常会使用非确定性行为的统计行为特性。因此，在并发进程模型中引入随机性的概念是有意义的。作为经典并发进程模型的重要扩展，概率进程被广泛研究，有代表性的工作有对 CCS 的概率性扩展[7, 8]，概率 CSP[9] 和概率 ACP[10] 等。近期，FU Y 提出了一个通用方法 Uniform Approach[11]，可以用于将并发进程模型扩展为概率并发进程模型。由于其模型无关性，我们可以使用 Uniform Approach 对其他并发进程模型进行概率化扩展。

有两种主要的进程演算可以作为概率模型的基础：分别是 MILNER R 的 CCS[3] 和 HOARE C 的 CSP[4]。它们的区别在与等价的类型和建模并发系统采用的方法[12]。Uniform Approach 以 CCS 为例进行了概率扩展，为理解 Uniform Approach，我们首先对 CCS 进行简要的介绍。

1.1　通信系统演算

CCS 使用了标记变迁系统 (Labled Transition System) 来建模。CCS 的语法如下：

$$S,T ::= X \mid \sum_{i\in I} \alpha_i.T_i \mid S \mid T \mid (a)T \mid \mu X.T \tag{1–1}$$

其中 S,T 为 CCS 的进程表达式 (agent)，X 为进程表达式变元（agent variables）。索引集合 I 是有限自然数集。$Chan$ 为名字（或通道）的集合，其中的元素以小写字母表示，集合 $\overline{Chan} = \{\overline{a} \mid a \in Chan\}$，动作集合 $Act = Chan \cup \overline{Chan} \cup \{\tau\}$，其中的元素用小写希腊字母表示，$\alpha_i \in Act$，$\tau$ 表示一切静态迁移（内部动作）。$\sum_{i\in I} \alpha_i.T$ 为非确定性选择项，若 $I = \varnothing$，我们可以将这一项写作 0。$S \mid T$ 表示 S,T 可以并发执行。$(a)T$ 为限制（Restriction）或内部化（Localization）操作子，对外隐藏 T 中的 a 通道，也可以写为 $T\backslash\{a\}$ 或 $T\backslash a$。μX 表示递归（Recursion）操作。若一个 CCS 进程表达式 P 中不含自由变元，称 P 为进程（process）。

CCS 的迁移语义如图 1–1，表示状态之间的变迁规则，其中 $\lambda \in Act$。

$$\frac{}{\sum_{i\in I}\alpha_i.T_i \xrightarrow{\alpha_i} T_i} \qquad \frac{S \xrightarrow{\bar{a}} S' \quad T \xrightarrow{a} T'}{S \mid T \xrightarrow{\tau} S' \mid T'} \qquad \frac{T \xrightarrow{\lambda} T'}{S \mid T \xrightarrow{\lambda} S \mid T'}$$

$$\frac{T \xrightarrow{\lambda} T'}{(a)T \xrightarrow{\lambda} (a)T'}\ a \notin \lambda \qquad \frac{T\{\mu X.T/X\} \xrightarrow{\lambda} T'}{\mu X.T \xrightarrow{\lambda} T'}$$

图 1–1 CCS 迁移语义

1.2 随机进程模型

FU Y 提供了一个模型无关的方法 Uniform Approach(A Uniform Approach to Random Process Model)[11]，将进程模型扩展为随机进程模型。Uniform Approach 区分了非确定性行为和概率性行为，在 CCS 的基础上建立了 RCCS(Randomized CCS) 的语法、迁移语义并研究了 RCCS 的代数特性，如互模拟关系。

RCCS 在 CCS 的基础上增加了概率选择操作子 $\bigoplus_{i\in I} p_i\tau.T_i$，其中 $0 < p_i < 1 \wedge \sum_{i\in I} p_i = 1$。例如，$S = \frac{1}{3}\tau.T_1 + \frac{2}{3}\tau.T_2$ 意味着 S 经过静态迁移（内部动作）变迁为 T_1 状态的概率为 $\frac{1}{3}$，变迁为 T_2 状态的概率为 $\frac{2}{3}$。对应随机选择操作子的迁移语义如图 1–2。

$$\frac{}{\bigoplus_{i\in I} p_i\tau.T_i \xrightarrow{p_i\tau} T_i}$$

图 1–2 随机选择操作子的迁移语义

由于 Uniform Approach 具有模型无关性，我们可以用它来扩展其他的进程模型，进而使非概率模型概率化。

1.3 研究目的

本文希望使用 Uniform Approach 的通用方法，概率化扩展经典并发模型——并发传值进程模型，得到随机传值进程模型。我们可以通过对传值进程模型的扩展，一方面进一步佐证 Uniform Approach 的模型无关性；一方面随机传值进程模型可以应用于具有传值特点的并发系统的建模和分析，如通信过程、安全协议、生物系统等。我们希望随机传值进程模型的语法、迁移语义可以用于为具有传值特点现实问题建模通信、同步等机制，为这类问题的模型设计与软件开发提供方法，并可以应用随机传值进程模型的代数特性如互模拟关系、等价关系等分析模型的死锁、活性、可观察、发散等并发特性。

1.4 论文结构

本文分为四个章节。第一章为绪论，引入了进程演算的概念，介绍了通信并发演算 CCS，引入了一种对进程模型进行概率化扩展的通用方法 Uniform Approach。第二章我们使用 Uniform Approach 中的方法对传值进程模型进行概率化扩展得到随机传值进程模型。第三章我们使用随机传值进程模型对基于云计算协议 Gossip-Style Membership 协议的通信过程建模。第四章为总结以及未来的工作方向。

第二章　随机传值进程模型

由于 Uniform Approach 是模型无关的概率扩展方法，我们可以使用该方法概率化扩展其他的进程模型。本章会对传值进程模型进行概率化扩展。传值进程模型是经典的进程模型，在 MILNER R 的 Communication and Concurrency 一书中，也使用了传值的通信并发模型——消息的发送者、接受者和通信媒介，引入并发和通信的概念[3]。我们选择传值进程模型的一种定义——The Value-Passing Calculus[13] 进行扩展。

2.1　传值进程模型

传值进程模型（Value-Passing Calculus）是一种将某个域中的值作为通信的内容，并且在特定逻辑条件下执行特定动作的并发进程模型。一个经典的例子是 MILNER R 的 Communication and Concurrency 一书中的一单元缓冲区，它可以接受、储存、发送消息：

$$\begin{aligned} C &\overset{def}{=} in(x).C'(x) \\ C'(x) &\overset{def}{=} \overline{out}(x).C \end{aligned} \tag{2–1}$$

其中 CCS 进程表达式 C 接受了 in 通道输入的变元 x 的一个值时会变迁为 $C'(x)$ 状态，$C'(x)$ 通过 $\overline{out}$ 通道输出 x 的值时会变迁为 C[3]。

更普遍的传值进程模型也可以对输入进行逻辑判定，根据特定的条件执行特定的动作，通过输入控制进程的执行。我们以进程 $A(x)$ 为例：

$$A(x) = \text{if } \varphi(x) \text{ then } \overline{a}(f(t)) \text{ else } B(x) \tag{2–2}$$

$A(x)$ 在满足条件 $\varphi(x)$ 时会通过通道 a 输出函数 $f(t)$ 的值，其中 t 可能是 x 的函数，也可能与 x 无关；若不满足 $\varphi(x)$，则执行程序 $B(x)$。

传值进程模型赋予了进程传递数据的能力，我们可以通过进程之间传递的数据来控制进程的执行和进程间的交互，这种控制能力显著的增强了并发进程模型的表达能力，拓宽了并发进程模型的应用场景。在网络通信中，各种通信协议通过在主机间传值进而控制主机的行为，执行通信协议的通信系统就可以被抽象为并发传值进程模型，支持并发的编程语言也可以被解释为传值进程模型，如 Erlang[14]。很多通过传值、计算解决的现实问题也都可以抽象为传值进程模型。

2.1.1　The Value-Passing Calculus

对传值进程模型的研究很多都会依赖一个神域，这个神域是一个领域模型 (Domain Model) 或一个逻辑理论 (Logic Theory)，帮助我们进行条件的判定，函数的计算和提供变元的取值范围。如在执行 $A(x)$ 时，我们会将 $\varphi(x)$ 传给神域，神域判定是否满足条件，并将判定结果返回给我们，同样的，我们也会将 $f(t)$ 传给神域，神域帮我们计算这个函数，并将结果返回给我们。例如，WINSKEL G 等人将变元表达式的取值和计算全部依赖一个集合 V[15]，LIN H 等人依赖一个评估 ρ 完成布尔表达式的判定和表达式的计算[16]，其中最经典的并发传值模型是 MILNER R 对 CCS 的扩展 Value-Passing CCS[3]，上述两个传值进程模

型也是基于 Value-Passing CCS。MILNER R 的 Value-Passing CCS 在 CCS 中引入了带数据参数的动作和进程，使它比之 CCS 刻画实际系统的能力大大提升，Value-Passing CCS 中有下列数据类型：Val 表示数据值的集合，其元素用 v 表示，相似的 $x \in DVar$ 为数据变量，$e \in DExp$ 为数据表达式，$b \in BExp$ 为布尔表达式，ρ 为 $DVal$ 到 Val 的映射，与 CCS 相同 $a, c \in Chan$。Value-Passing CCS 的语法可以表示为：

$$S, T ::= 0 | A(e) | \sum_{i \in I} \alpha_i.T | S|T | (a)T | if\ b\ then\ T \tag{2–3}$$

其中 $\alpha_i ::= \tau | c(x) | \overline{c}(t)$，$t$ 或是 $\rho(x)$ 计算所得或与 x 无关，$A(e)$ 为进程常量[17]。然而 MILNER R 在 Value-Passing CCS 的定义过程中没有讨论函数的计算过程和 $if\ b\ then\ T$ 中条件 b 的可判定性，可以认为 MILNER R 的传值进程模型同样依赖神域帮助 Value-Passing CCS 完成这些未定义的工作。

神域通常不包括在传值进程模型中，且通常是未定义的，或只在特定场景下定义。由于神域的存在，对于这些传值进程模型表达能力的衡量变得十分困难，FU Y 在 The Value-Passing Calculus 中取缔了神域的概念，提出了一个封闭的传值进程模型[13]，这个模型同时也是不针对某一应用场景的通用模型。

The Value-Passing Calculus 中的传值进程模型 $\mathbb{VPC}_{\mathsf{Th}}$ 的语法可以表示为：

$$T ::= \sum_{i \in I} \varphi_i a(x).T_i | \sum_{i \in I} \varphi_i \overline{a}(t_i).T_i | T|T' | (c)T | \varphi T | !a(x).T | !\overline{a}(t).T \tag{2–4}$$

其中 T 是一个 $\mathbb{VPC}_{\mathsf{Th}}$ 项，Th 是可判定的逻辑，φ_i, φ 是布尔表达式，可以通过 Th 证明或证伪，若 Th 可以给出 φ_i 的一个证明，记为 $\mathsf{Th} \vdash \varphi_i$。$a(x)$ 是一个输入前缀，$\overline{a}(t_i)$ 是一个输出前缀，我们可以用 $\sum_{i \in I} \varphi_i \lambda_i.T_i$ 来表示 $\sum_{i \in I} \varphi_i a(x).T_i$ 或 $\sum_{i \in I} \varphi_i \overline{a}(t).T_i$。$!a(x).T$ 和 $!\overline{a}(t).T$ 对应递归操作，$\mu X.E$ 可以表示为 $(c)(E\{c(z).0/X\} | !\overline{c}(r).E\{c(z).0/X\})$。其余的语法与第一章 CCS 语法定义的解释相同。

The Value-Passing Calculus 提出了两种迁移语义的描述：具体语义（Concrete Semantics）和符号语义（Symbolic Semantics）。后文我们只用到了符号语义，因此我们忽略具体语义的相关内容。

$\mathbb{VPC}_{\mathsf{Th}}$ 的符号迁移语义如图 2–1所示。

$$\textit{Action}\quad \frac{}{\sum_{i \in I} \varphi_i \lambda_i.T_i \xrightarrow{\lambda_i}_{\varphi_i} T_i}\ i \in I$$

$$\textit{Composition}\quad \frac{S \xrightarrow{\lambda}_{\varphi} S'}{S|T \xrightarrow{\lambda}_{\varphi} S'|T} \qquad \frac{S \xrightarrow{a(x)}_{\varphi} S'\ T \xrightarrow{\overline{a}(t)}_{\psi} T'}{S|T \xrightarrow{\tau}_{\varphi\psi} S'\{t/x\}|T'}$$

$$\textit{Localization}\quad \frac{T \xrightarrow{\lambda}_{\varphi} T'}{(c)T \xrightarrow{\lambda}_{\varphi} (c)T'}\ c \notin \lambda \qquad \textit{Condition}\quad \frac{T \xrightarrow{\lambda}_{\varphi} T'}{\phi T \xrightarrow{\lambda}_{\phi\varphi} T'}$$

$$\textit{Recursion}\quad \frac{}{!a(x).T \xrightarrow{a(x)}_{\top} T|!a(x).T} \qquad \frac{}{!\overline{a}(x).T \xrightarrow{\overline{a}(x)}_{\top} T|!\overline{a}(x).T}$$

图 2–1 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号迁移语义规则

其中，$T = \varphi\lambda.T'$ 在 φ 条件下执行 λ 动作到达 T' 状态，在符号语义下可以写作 $T \xrightarrow{\lambda}_{\varphi} T'$。

符号语义中的动作集合 $Act = \{a(x), \overline{a}(t) | a \in Chan, x \in \mathsf{V}_\Sigma, t \in \mathsf{T}_\Sigma\} \cup \{\tau\}$，$Act$ 中的元素可以用 λ 表示，其中 $\mathsf{V}_\Sigma, \mathsf{T}_\Sigma$ 为可判定逻辑 Th 中变元的集合和项的集合[13]。

The Value-Passing Calculus 使用可判定的一阶理论[18] 判定条件并提出了一个图灵完备的数值系统（Numeric System）[13] 作为底层模型来实现可计算的函数。通过这两种方式，将 Value-Passing Calculus 从神域中解放出来。

2.1.2 If Then Else 语法符号化

进程 $A(x)$ 中出现了 If Then Else 语法，If Then Else 语法实现的分支控制在编程中也是非常重要的组成部分。在 Communication and Concurrency[3] 中也多次使用 If Then Else 语法，然而在 CCS 的定义中，我们无法通过 CCS 进程表达式的语法规则来实现这种分支控制；书中对分支控制的使用，如第五章（Bisimulation and Observation Equivalence）中对 JobShop 的建模[3] 也比较随意，没有严格的定义也没有判断 If 条件的可判定性。在 FU Y 的 The Value-Passing Calculus[13] 中，规定了 If 条件是通过一阶理论 Th 可判定的布尔表达式，并说明 if φ then S else T 可以被定义为 $\varphi S | \neg\varphi T$。以下推论依据 The Value-Passing Calculus 中对 If Then Else 语法的定义和 $\mathbb{VPC}_{\mathsf{Th}}$ 的 Proof System[13]，给出 If Then Else 的变体的符号化定义，在本文的后续章节会统一使用这些符号。

以下推论中 $S, T \in \mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$。

推论 2.1 $\varphi 0 = 0$

证明 $\varphi 0 = \varphi 0 + 0 = \varphi 0 + \mathsf{T}0 = \varphi 0 + (\varphi \vee \neg\varphi)0 = \varphi 0 + \varphi 0 + \neg\varphi 0 = \varphi 0 + \neg\varphi 0 = (\varphi \vee \neg\varphi)0 = \mathsf{T}0 = 0$ □

推论 2.2 $\varphi T = (\varphi T \mid \neg\varphi 0)$

证明 $\varphi T = (\varphi T \mid 0) = (\varphi T \mid \neg\varphi 0)$ □

推论 2.3 $S = \neg\varphi\varphi T$，则 $S = 0$。

证明 $S = \neg\varphi\varphi T = (\neg\varphi \wedge \varphi)T = \bot T = 0$ □

进而我们可以得到 If Then Else 语法与 $\mathbb{VPC}_{\mathsf{Th}}$ 规则的对照表：

表 2–1 If Then Else 语法对照表

语法	$\mathbb{VPC}_{\mathsf{Th}}$ 规则对照
$S =$ if φ then T else T'	$S = (\varphi T \mid \neg\varphi T')$
$S =$ if φ then T	$S = (\varphi T \mid \neg\varphi 0)$
$S =$ if $\neg\varphi$ then if φ then T	$S = 0$

2.2 随机传值进程模型

由于随机性的可计算性，许多具有传值性质的通信过程、生物现象等现实问题的建模和分析也被引入了概率的思想，如 SWIM 协议[19] 和生物自组装系统[20]。我们可以在传值进

程模型中引入随机性用于对这些具有随机性、传值特点的问题，或者可以用随机分布近似表示非确定性行为的系统进行建模和分析。目前已经存在使用随机传值进程模型建模和分析网络安全[21] 等应用，但由于其是在 Value-passing CCS[3] 基础上的概率扩展，以文献 [21, 22] 为例的模型仍然存在神域的问题。

为了避免神域带来的问题，我们可以将随机传值进程模型定义在 $\mathbb{VPC}_{\mathsf{Th}}$ 的基础上，用 Uniform Approach 中的随机选择 $\bigoplus_{i\in I} p_i\tau.T_i$ 扩展公式 2–4中 $\mathbb{VPC}_{\mathsf{Th}}$ 项的定义。我们可以得到随机传值进程模型 (Randomized VPC)，记为 $\mathbb{RVPC}_{\mathsf{Th}}$ 项的定义：

$$T ::= \bigoplus_{i\in I} p_i\tau.T_i \mid \sum_{i\in I} \varphi_i\lambda_i.T_i \mid T \mid T' \mid (c)T \mid \varphi T \mid !a(x).T \mid !\bar{a}(t).T \tag{2–5}$$

其中，与 2.1.1中定义的一样：$\lambda_i \in Act$，Th 为可判定的一阶理论。随机选择操作子 $\bigoplus_{i\in I} p_i\tau.T_i$ 中，$0 < p_i < 1, \sum_{i\in I} p_i = 1$。$S = \bigoplus_{i\in I} p_i\tau.T_i$ 意味着 S 在 p_i 的概率下经过内部动作 τ 到达 T_i 状态。$\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 为所有 $\mathbb{RVPC}_{\mathsf{Th}}$ 的项 T 的集合，其中若 $P \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 中不含自由变量，则称 P 为 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的进程，进程的全集记为 $\mathcal{P}_{\mathbb{RVPC}_{\mathsf{Th}}}$。

在 2.1.1中 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号语义的基础上，增加图 1–2定义的随机选择操作子的迁移规则，我们可以得到随机传值进程模型的符号语义，其中随机选择的条件为 T，表明该操作在任意条件下可执行；若随机选择也需要在特定条件下执行，我们可以配合 $\mathbb{VPC}_{\mathsf{Th}}$ 中的条件操作子，得到 $\varphi(\bigoplus_{i\in I} p_i\tau.T_i)$。

Action

$$\frac{}{\sum_{i\in I} \varphi_i\lambda_i.T_i \xrightarrow{\lambda_i}_{\varphi_i} T_i} \qquad \frac{}{\bigoplus_{i\in I} p_i\tau.T_i \xrightarrow{p_i\tau}_{\top} T_i}$$

Composition

$$\frac{S \xrightarrow{\lambda}_{\varphi} S'}{S|T \xrightarrow{\lambda}_{\varphi} S'|T} \qquad \frac{S \xrightarrow{a(x)}_{\varphi} S' \qquad T \xrightarrow{\bar{a}(x)}_{\psi} T'}{S|T \xrightarrow{\tau}_{\varphi\psi} S'|T'}$$

Localization

$$\frac{T \xrightarrow{\lambda}_{\varphi} T'}{(c)T \xrightarrow{\lambda}_{\varphi} (c)T'} \qquad \frac{T \xrightarrow{\lambda}_{\varphi} T'}{T\backslash\{c\} \xrightarrow{\lambda}_{\varphi} T'\backslash\{c\}}\ c \text{ is not in } \lambda$$

Condition

$$\frac{T \xrightarrow{\lambda}_{\varphi} T'}{\phi T \xrightarrow{\lambda}_{\phi\varphi} T'}$$

Recursion

$$\frac{}{!a(x).T \xrightarrow{a(x)}_{\top} T \mid !a(x).T} \qquad \frac{}{!\bar{a}(x).T \xrightarrow{\bar{a}(x)}_{\top} T \mid !\bar{a}(x).T}$$

图 2–2 随机传值进程模型的符号迁移语义

对于 $\mathbb{VPC}_{\mathsf{Th}}$ 的前缀操作子的迁移规则：$\sum_{i\in I} \varphi_i\lambda_i.T_i \xrightarrow{\lambda_i}_{\varphi_i} T_i$，它本质上代表了非确定选择，即作出 φ_i 条件下的 λ_i 动作到达 T_i 状态是非确定的，我们可以通过 Uniform Approach 的方法将它扩展为一个概率性选择，即在概率 p_i 下会作出 φ_i 条件下的 λ_i 动作：

$\bigoplus_{i\in I} p_i\tau.\varphi_i\lambda_i.T_i \xrightarrow{p_i\tau}_{\mathsf{T}} \xrightarrow{\lambda_i}_{\varphi_i} T_i$。它的语义为：在概率 p_i 下，我们会经过一个内部 τ 操作到达一个 $\mathbb{RVPC}_{\mathsf{Th}}$ 状态: $\varphi_i\lambda_i.T_i$，若 $\mathsf{Th}\vdash\varphi_i$（即一阶理论 Th 下，φ_i 为真），则我们可以经过 λ_i 操作到达 $\mathbb{RVPC}_{\mathsf{Th}}$ 状态 T_i，其中 $\lambda_i\in\{a(x),\bar{a}(x)\mid a\in Chan, x\in\mathsf{V}_\Sigma, t\in\mathsf{T}_\Sigma\}\cup\{\tau\}$。

$\mathbb{RVPC}_{\mathsf{Th}}$ 在 $\mathbb{VPC}_{\mathsf{Th}}$ 的基础上扩展了前缀操作：$p\tau.$，其中 $p\in(0,1)$。类似 $A=\tau.B$ 的 $\mathbb{RVPC}_{\mathsf{Th}}$ 项，我们可以认为 $A\xrightarrow{1\tau}_{\mathsf{T}} B$，这时也满足 $A=\bigoplus_{i\in I} p_i\tau.A_i$ 的定义，此时 $I=\{1\}, p_1=1, A_1=B$。若将 $\mathbb{VPC}_{\mathsf{Th}}$ 中的此类内部操作同样看待，我们可以得出推论 2.4。

推论 2.4 $\mathbb{VPC}_{\mathsf{Th}}$ 是一种特殊的 $\mathbb{RVPC}_{\mathsf{Th}}$。

2.3 随机传值进程模型中的互模拟关系

2.3.1 互模拟关系与观察等价性

程序理论的基本问题是进程的等价性。等价关系的定义为：设 R 是非空集合 A 上的二元关系，若 R 是自反的、对称的、传递的，则称 R 是 A 上的等价关系。研究等价关系的目的在于将集合中的元素进行分类，选取每类的代表元素来降低问题的复杂度，如软件测试时，可利用等价类来选择测试用例[23]。对通信并发系统程序等价的定义和验证，现在已有很多研究。互模拟等价性为进程描述语言提供了有效的语义理论。在提出 CCS 时 MILNER R 提出了观察等价 *(observation equivalence)* 与弱互模拟 *(weak bisimulation)*[3]。Van GLABBEEK R 和 WEIJLAND W 提出的分支互模拟 *(branching bisimulation)*[24, 25] 也是十分著名的研究。

对于概率模型的等价性，目前有对全概率进程模型，即概率选择代替非确定性选择的进程模型的弱互模拟及分支互模拟的研究[26]，仅适用于有限状态的概率进程模型的研究[27, 28]等。Uniform Approach 在分支互模拟[24, 25] 的基础上给出了 RCCS 分支互模拟关系的定义，以及等价关系同余性的证明[11]。

RCCS 的分支互模拟是 CCS 分支互模拟的扩展，我们首先来看 CCS 的分支互模拟。$\mathcal{P}_{CCS}$ 上的分支互模拟的定义如下：

定义 2.1 $\mathcal{E}$ 是在 $\mathcal{P}_{CCS}$ 上的对称关系，若对于任意 $A,B\in\mathcal{P}_{CCS}, A\mathcal{E}B$ 满足：
若 $A\xrightarrow{l}A'$，则 $l=\tau$ 且 $A'\mathcal{E}B$ 或存在 B', B'' 使得 $B\xrightarrow{\tau}\cdots\xrightarrow{\tau}B'\xrightarrow{l}B''$ 且 $A\mathcal{E}B'\wedge A'\mathcal{E}B''$。
则称 $\mathcal{E}$ 是在 $\mathcal{P}_{CCS}$ 上一个分支互模拟关系。

内部操作 τ 带来的状态迁移可以称为静态迁移。若 $\sim$ 是 $\mathcal{P}_{CCS}$ 上的一个等价关系，当 $A\xrightarrow{\tau}A'\sim A$ 时，我们可以写为 $A\xrightarrow{\tau}_{\sim}A'$，当 $A\xrightarrow{\tau}\cdots\xrightarrow{\tau}A'\sim A$ 时，我们可以写为 $A\overset{\tau}{\Rightarrow}_{\sim}A'$。在 Uniform Approach 中，这种动作称为状态保持的静态迁移（*state-preserving silent transition*），对应的，若 $A\xrightarrow{\tau}\cdots\xrightarrow{\tau}A'\nsim A$，称为状态改变的静态迁移（*state-changing silent transition*）。$\Rightarrow_{\sim}$ 为 $\xrightarrow{\tau}_{\sim}$ 的闭包[11]。

Uniform Approach 中 RCCS 的分支互模拟涉及等价集的概念，我们首先看 $\mathcal{P}_{CCS}$ 上的等价集的定义：

定义 2.2 若二元关系 $\mathcal{E}$ 是 $\mathcal{P}_{CCS}$ 上的等价关系。A 关于等价关系 $\mathcal{E}$ 的等价集写作 $[A]_{\mathcal{E}}$，$A\in[A]_{\mathcal{E}}$，若 $(A,B)\in\mathcal{E}$，有 $B\in[A]_{\mathcal{E}}$。我们用 $\mathcal{P}_{CCS}/\mathcal{E}$ 表示所有 $\mathcal{P}_{CCS}$ 关于 $\mathcal{E}$ 的等价集的集合。

$\mathcal{P}_{RCCS}$，$\mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$，$\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价集的定义都是相似的，因此不做赘述。

引入了等价集的概念后，我们可以将分支互模拟的形式简化为定义 2.3的形式：

定义 2.3 $\mathcal{E}$ 是 $\mathcal{P}_{CCS}$ 上的等价关系，若对于任意动作 $l, l \neq \tau$ 和任意等价集 $C \in \mathcal{P}_{CCS}/\mathcal{E}, C \neq [A]_\mathcal{E}$，满足若 $B\mathcal{E}A \Rightarrow_\mathcal{E} \xrightarrow{l} C$，则 $B \Rightarrow_\mathcal{E} \xrightarrow{l} C$，称 $\mathcal{E}$ 是 $\mathcal{P}_{CCS}$ 上的分支互模拟关系。

定义 2.3与定义 2.1虽然本质上是相同的，但是定义 2.3更好的揭示了分支互模拟的核心思想：一系列状态保持的静态模拟后的状态改变的静态迁移之间的相互模拟。

考虑到 RCCS 在 CCS 的基础上增加了概率选择，状态保持的静态迁移在 $A \xrightarrow{\tau} A'\mathcal{E}A$ 的基础上新增了 $A \xrightarrow{p\tau} A'\mathcal{E}A, 0 < p < 1$ 的可能，为了用类似 $\Rightarrow_\mathcal{E}$ 的方式表示这种静态迁移，Uniform Approach 提出了等价树（*Epsilon Tree*）的概念，将概率选择作为树的分支，节点 A 的关于等价关系 $\mathcal{E}$ 的等价树上的任意节点 $N \in [A]_\mathcal{E}$。同时，Uniform Approach 定义了两种迁移：l-迁移 (l-transition) 和 q-迁移 (q-transition)，分别表示等价树上的节点迁移到其他等价集的动作和概率，Uniform Approach 中定义的概率版分支互模拟是对 l-迁移和 q-迁移的相互模拟[11]。

定义 2.4 (l-迁移 (l-transition)) $A \in \mathcal{P}_{RCCS}$，从 A 到 $\mathcal{B} \in \mathcal{P}_{RCCS}/\mathcal{E}, \mathcal{B} \neq [A]_\mathcal{E}$ 的 l-迁移表示 A 关于等价关系 $\mathcal{E}$ 的等价树上的每一个叶子结点 L，存在 $L \xrightarrow{l} L' \in \mathcal{B}, l \neq \tau$，写作 $A \rightsquigarrow_\mathcal{E} \xrightarrow{l} \mathcal{B}$。

定义 2.5 (q-迁移 (q-transition)) $A \in \mathcal{P}_{RCCS}$，$\mathcal{E}$ 为 $\mathcal{P}_{RCCS}$ 上的等价关系，L 为 A 关于等价关系 $\mathcal{E}$ 的等价树上的每一个叶子结点，$\mathcal{B} \in \mathcal{P}_{RCCS}/\mathcal{E}, \mathcal{B} \neq [A]_\mathcal{E}$。

定义 $\mathsf{P}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) = \sum\{p_i | L \xrightarrow{p_i\tau} L_i \in \mathcal{B} \wedge i \in I\}$。

定义 $\mathsf{P}_\mathcal{E}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) = \mathsf{P}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B})/(1 - \mathsf{P}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} [A]_\mathcal{E}))$。

当 $\mathsf{P}_\mathcal{E}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) = q$ 时，记为 $A \rightsquigarrow_\mathcal{E} \xrightarrow{q} \mathcal{B}$，即 q-迁移。

现在我们终于可以引入 Uniform Approach 中定义的 RCCS 的分支互模拟关系了！

定义 2.6 (分支互模拟) 当 $\mathcal{P}_{RCCS}$ 上的等价关系 $\mathcal{E}$ 满足：

(1) 若 $B\mathcal{E}A \rightsquigarrow_\mathcal{E} \xrightarrow{l} C \in \mathcal{P}_{RCCS}/\mathcal{E}, l \neq \tau \wedge C \neq [A]_\mathcal{E}$，则 $B \rightsquigarrow_\mathcal{E} \xrightarrow{l} C$。

(2) 若 $B\mathcal{E}A \rightsquigarrow_\mathcal{E} \xrightarrow{q} C \in \mathcal{P}_{RCCS}/\mathcal{E}, l \neq \tau \wedge C \neq [A]_\mathcal{E}$，则 $B \rightsquigarrow_\mathcal{E} \xrightarrow{q} C$。

则等价关系 $\mathcal{E}$ 是分支互模拟关系。

$\mathbb{VPC}_{\mathsf{Th}}$ 与 CCS 的区别主要体现在前缀操作子和条件操作子，这也是传值的进程模型与非传值的进程模型的主要区别。对于前缀操作子，相似的定义互模拟关系是比较容易的，只需要保证通道一致、值一致即可。而条件操作子给互模拟的定义带来了麻烦，比如：$A = \overline{a}(0).S$ 和 $B = ((x = 0)\overline{a}(0).S | \neg(x = 0)\overline{a}(0).S)$ 在观察上应该是一致的，但若我们应用分支互模拟 A 可以作出 $A \xrightarrow{a(0)}_\mathsf{T} S$ 的动作，而 B 不能。但对于 $A \xrightarrow{a(0)}_{x=0} S$ 和 $A \xrightarrow{a(0)}_{\neg(x=0)} S$，$B$ 却可以相应的作出模拟的动作。这就要求 B 的两个操作：$B \xrightarrow{\overline{a}(0)}_{x=0} S$ 和 $B \xrightarrow{\overline{a}(0)}_{\neg(x=0)} S$ 共同模拟 $A \xrightarrow{\overline{a}(0)}_\mathsf{T} S$。

为了上述问题 HENNESSY M 和 LIN H 为传值进程模型提出了符号迁移图 (Symbolic Transition Graph) 和符号互模拟 (Symbolic Bisimulation) 的概念[29]。在 The Value-Passing Calculus 中，FU Y 也提出了 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号互模拟[13]。

定义 2.7 (符号互模拟) $\mathcal{E}$ 是一个 $\mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$ 上的二元对称关系，当任意 $A\mathcal{E}B$ 满足下列条件时，称 $\mathcal{E}$ 是一个符号互模拟关系：

(1) 若 $A \xrightarrow{\tau}_{\varphi} A'$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \Rightarrow_{\psi_i} B_i\}_{i\in I}$，对于 $\forall i \in I$，$\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i \wedge \varphi_i A \mathcal{E} \varphi_i B_i$，且满足 $\varphi_i A' \mathcal{E} \varphi_i B_i$ 或 $B_i \xrightarrow{\tau}_{\psi_i'} B_i' \wedge \mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i' \wedge \varphi_i A' \mathcal{E} \varphi_i B_i'$。

(2) 若 $A \xrightarrow{\overline{a}(t)}_{\varphi} A'$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \Rightarrow_{\psi_i} B_i \xrightarrow{\overline{a}(t_i)}_{\psi_i'} B_i'\}$，满足 $\mathsf{Th} \vdash (\varphi_i \Rightarrow \psi_i\psi_i') \wedge (\varphi_i \Rightarrow t = t_i)$ 且 $(\varphi_i A \mathcal{E} \varphi_i B_i) \wedge (\varphi_i A' \mathcal{E} \varphi_i B_i')$。

(3) 若 $A \xrightarrow{a(x)}_{\varphi} A'$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \Rightarrow_{\psi_i} B_i \xrightarrow{a(x)}_{\psi_i'} B_i'\}$，满足 $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\psi_i'$ 且 $(\varphi_i A \mathcal{E} \varphi_i B_i) \wedge (\varphi_i A' \mathcal{E} \varphi_i B_i')$。

互模拟关系的全集记为 $\approx^s_{\mathsf{Th}}$。

根据符号互模拟的定义，我们可以解决前述条件操作子带来的麻烦，我们可以通过下面的例子验证这一点。

例 2.1 证明 $A = \overline{a}(0).S$ 和 $B = ((x = 0)\overline{a}(0).S | \neg(x = 0)\overline{a}(0).S)$ 是符号互模拟的。

证明 定义等价关系 $\mathcal{E} = \{(A, B)\} \cup \equiv$，其中 $\equiv$ 为绝对等价关系，题目等价于证明 $\mathcal{E}$ 是一个符号互模拟关系。

对于 $A \xrightarrow{\overline{a}(0)}_{\mathsf{T}} S$，存在 T 的划分 $\{(x = 0), \neg(x = 0)\}$ 和集合 $\{B \xrightarrow{\overline{a}(0)}_{x=0} S, B \xrightarrow{\overline{a}(0)}_{\neg(x=0)} S\}$，且 $(x = 0)S \mathcal{E} (x = 0)S$。

对于 $B \xrightarrow{\overline{a}(0)}_{x=0} S$，存在 $A \xrightarrow{\overline{a}(0)}_{x=0} S$ 与之符号互模拟。$B \xrightarrow{\overline{a}(0)}_{\neg(x=0)} S$ 同理。 □

2.3.2 条件等价树

根据 2.3.1中定义 2.7中定义的 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号互模拟关系，我们观察 $A(x) = (x \geq 5)\overline{a}(\mathsf{s}(x)).0$ 和 $A'(x) = (x \geq 3)\overline{a}(\mathsf{s}(x)).0$，根据 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号互模拟的定义，我们容易得到 A 和 A' 是不符号互模拟的，对于 $A'(t) \xrightarrow{\overline{a}(\mathsf{s}(t))}_{x\geq 3} 0$，因为 $3 < x < 5$ 的这一部分动作是 $A(x)$ 无法完成的，所以不存在 $x \geq 3$ 的划分，使得 $A(x)$ 可以给出一个迁移的集合来模拟整个 $x \geq 3$。但反观 $B(x) = (x > 5)A(x), B'(x) = (x > 5)A'(x)$，我们可以得到 $B'(x) = ((x > 5) \wedge (x \geq 3))\overline{a}(\mathsf{s}(x)).0 = (x > 5)\overline{a}(\mathsf{s}(x)).0$，同理 $B(x) = (x > 5)\overline{a}(\mathsf{s}(x)).0$，$B(x), B'(x)$ 不仅是符号互模拟，甚至是绝对等价的。对于 $(x > 5)A(x)$ 和 $(x > 5)A'(x)$ 的这种关系，我们给出条件等价集的定义，可以使等价集内部对某个条件透明：

定义 2.8 (条件等价集) $A, A' \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，$\mathcal{E}$ 是 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的等价关系，若 $\varphi A \mathcal{E} \varphi A'$，则 $A' \in [A]_{\varphi\mathcal{E}}$。$[A]_{\varphi\mathcal{E}}$ 称为等价关系 $\mathcal{E}$ 在条件 φ 下包含 A 的等价集。我们用 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}/\varphi\mathcal{E}$ 表示所有 $T \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 关于布尔表达式 φ 和等价关系 $\mathcal{E}$ 的条件等价集的集合。

由于 $\mathbb{VPC}_{\mathsf{Th}}$ 是特殊的 $\mathbb{RVPC}_{\mathsf{Th}}$，条件等价集的定义同样适用 $\mathbb{VPC}_{\mathsf{Th}}$。显然，$A'(x) \in [A(x)]_{(x>5)\mathcal{E}}$，其中 $\mathcal{E} = \approx^s_{\mathsf{Th}}$。

条件 T 具有一定的特殊性：对于无条件的操作我们认为操作是在 T 条件下执行的，因此 T 可以看作是最强的条件。

推论 2.5 $A, B \in \mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$，若 $A \approx^s_{\mathsf{Th}} B$，则对布尔表达式 φ，$\varphi A \approx^s_{\mathsf{Th}} \varphi B$。

证明 对于 A 可以执行的操作，$\varphi' A$ 也会有对应的版本：

(1) 若 $A \xrightarrow{\tau}_{\varphi} A'$，则 $\varphi' A \xrightarrow{\tau}_{\varphi'\varphi} A'$。

(2) 若 $A \xrightarrow{a(x)}_{\varphi} A'$，则 $\varphi' A \xrightarrow{a(x)}_{\varphi'\varphi} A'$。

(3) 若 $A \xrightarrow{\overline{a}(t)}_{\varphi} A'$，则 $\varphi' A \xrightarrow{\overline{a}(t)}_{\varphi'\varphi} A'$。

对 $\varphi' A \xrightarrow{a(x)}_{\varphi'\varphi} A'$，根据 $A \approx^{s}_{\mathsf{Th}} B$，我们知道存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \Rightarrow_{\psi_i} B_i \xrightarrow{a(x)}_{\psi_i'} B_i'\}_{i\in I}$，满足 $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\psi_i' \wedge \varphi_i A \approx^{s}_{\mathsf{Th}} \varphi_i B_i \wedge \varphi_i A' \approx^{s}_{\mathsf{Th}} \varphi_i B_i'$。

若要证明 $\varphi' A \approx^{s}_{\mathsf{Th}} \varphi' B$，我们可以构造等价关系 $S = \{(\varphi' A, \varphi' B) | (A, B) \in \approx^{s}_{\mathsf{Th}}\}$，证明 S 是符号互模拟的，其中 A, B 为任意 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$。

通过 $A \approx^{s}_{\mathsf{Th}} B$ 我们可以得到集合 $\{\varphi'\varphi_i\}_{i\in I}$，对于任意 i, j，由于 $\varphi_i \wedge \varphi_j = \bot$，我们可以得到 $(\varphi'\varphi_i) \wedge (\varphi'\varphi_j) = \bot$；又 $\bigvee_{i\in I} \varphi_i = \varphi$，$\bigvee_{i\in I} \varphi'\varphi_i = \varphi'\varphi$，因此 $\{\varphi'\varphi_i\}_{i\in I}$ 为 $\varphi'\varphi$ 的划分。我们还可以得到集合 $\{\varphi' B \Rightarrow_{\varphi'\psi_i} B_i \xrightarrow{a(x)}_{\psi_i'} B_i'\}$，满足 $\mathsf{Th} \vdash \varphi'\varphi_i \Rightarrow \varphi'\psi_i\psi_i'$，而 $(\varphi'\varphi_i A, \varphi'\varphi_i B_i), (\varphi'\varphi_i A', \varphi'\varphi_i B_i') \in S$，因此我们可以用 $\{\varphi' B \Rightarrow_{\varphi'\psi_i} B_i \xrightarrow{a(x)}_{\psi_i'} B_i'\}$ 模拟 $\varphi' A \xrightarrow{a(x)}_{\varphi'\varphi} A'$。

对称的一边和其他两种情况的证明是相似的。 □

推论 2.6 $A, B \in \mathbb{RVPC}_{\mathsf{Th}}$，$\mathcal{E}$ 是 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的等价关系，φ 是任意布尔表达式，若 $B \in [A]_{\mathcal{E}}$，则 $B \in [A]_{\varphi\mathcal{E}}$。

推论 2.6可以简单理解为：一个 $\mathbb{RVPC}_{\mathsf{Th}}$ 项在任意条件下成立的等价集是所有条件等价集的交集。我们也可以给出一个简单的证明。

证明 $A, B \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，$fv(\varphi)$ 为 φ 中的自由变元的集合，V 为 $fv(\varphi)$ 的全部取值的集合，对于 $fv(\varphi)$ 的每一个赋值 $v \in V$，我们可以得到一个无自由变元的布尔表达式 φ_v：

(1) 若 $\mathsf{Th} \vdash \varphi_v$，$\varphi_v A = A\mathcal{E}B = \varphi_v B$。

(2) 若 $\mathsf{Th} \nvdash \varphi_v$，$\varphi_v A = 0 = \varphi_v B$。 □

将 T 推广到其他布尔表达式，我们可以得到更为广泛的结论：

推论 2.7 $A, B \in \mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$，若 $\varphi A \approx^{s}_{\mathsf{Th}} \varphi B$，则对布尔表达式 $\varphi', \mathsf{Th} \vdash \varphi' \Rightarrow \varphi$，$\varphi' A \approx^{s}_{\mathsf{Th}} \varphi' B$。

推论 2.8 $A, B \in \mathbb{RVPC}_{\mathsf{Th}}$，$\mathcal{E}$ 是 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的等价关系，φ 是任意布尔表达式，若 $B \in [A]_{\varphi\mathcal{E}}$，则对于布尔表达式 $\varphi', \mathsf{Th} \vdash \varphi' \Rightarrow \varphi$，$B \in [A]_{\varphi'\mathcal{E}}$。

推论 2.9 $L \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，$L$ 关于布尔表达式 φ 和等价关系 $\mathcal{E}$ 的等价集为 $C = [L]_{\varphi\mathcal{E}}$，对任意 $\mathsf{Th} \vdash \varphi' \Rightarrow \varphi$，$L$ 关于 $\varphi'\mathcal{E}$ 的等价集 $[L]_{\varphi'\mathcal{E}}$ 是 C 的超集，可以将 $[L]_{\varphi'\mathcal{E}}$ 记为 $[C]_{\varphi'\mathcal{E}}$。

推论 2.9可以由推论 2.8得出。

定义 2.7给出了 $\mathbb{VPC}_{\mathsf{Th}}$ 的互模拟的定义，观察定义 2.7中的 $A \xrightarrow{a(x)}_{\varphi} A'$，我们用 $\{B \Rightarrow_{\psi_i} B_i' \xrightarrow{a(x)}_{\psi_i'} B_i'\}$ 模拟，其中 $(\varphi_i A \mathcal{E} \varphi_i B_i) \wedge (\varphi_i A' \mathcal{E} \varphi_i B_i')$，根据条件等价集的定义，我们可以得到 $B_i \in [A]_{\varphi\mathcal{E}}$，$B_i' \in [A']_{\varphi\mathcal{E}}$，由于 $B_i \approx^{s}_{\mathsf{Th}} A$，$B_i \approx^{s}_{\mathsf{Th}} B$，那么 $B \Rightarrow_{\psi_i} B_i \in [B]_{\varphi\mathcal{E}}$ 实际上也是状态保持的静态迁移。

对于我们概率扩展后的 $\mathbb{RVPC}_{\mathsf{Th}}$，对上述例子状态保持静态迁移多了 $B \xrightarrow{q\tau}_{\psi_i} B_i$ 的情况，其中 $q \in (0, 1)$，表示概率的状态保持的静态迁移。我们希望可以找到 $\mathbb{RVPC}_{\mathsf{Th}}$ 中类似 $\Rightarrow_{\mathcal{E}}$ 的方式来表达这种状态保持的静态迁移。我们可以使用 Uniform Approach 中等价树的方法来刻画这种状态保持的静态迁移，在 $\mathbb{RVPC}_{\mathsf{Th}}$ 中称为条件等价树。

在定义条件等价树之前，我们首先关注一种条件无关的静态迁移树：

定义 2.9 (静态迁移树) 若 $A \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$, A 的静态迁移树 t 满足如下定义：

(1) 每一个节点都被标记成 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 的一个项，A 是根节点。

(2) 节点间的边被标记成 (φ, p)，其中 $p \in (0,1]$，φ 是一个布尔表达式。如果一条从 A' 到 A'' 的有向边被标记成 (φ, p)，表示 $A' \xrightarrow{p\tau}_{\varphi} A''$。

静态迁移树包含了状态改变的静态迁移和状态保持的静态迁移，条件等价树需要对特定条件下的状态改变的静态迁移进行剪枝：

定义 2.10 (条件等价树) φ 是一个布尔表达式，$\mathcal{E}$ 是 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系，$A \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，当下列条件成立时，$A$ 的静态迁移树 t 称为是一个关于 $\varphi\mathcal{E}$ 的条件等价树（$\varphi\mathcal{E}$-tree），记作 $t^{A}_{\varphi\mathcal{E}}$。

(1) t 上的所有节点 N，$N \in [A]_{\varphi\mathcal{E}}$，并被重新标记为 φN。

(2) 若 t 的边被标记为 (ψ, q)，则 $\mathsf{Th} \vdash \varphi \Rightarrow \psi$。

(3) 若 B, B' 是 t 上的节点，$B \overset{(\psi,q)}{\rightarrow} B', q \in (0,1)$，则存在 $B \xrightarrow[\psi]{\coprod_{i\in[k]}} \coprod_{i\in[k]} B_i$，$B_i, i \in [k]$ 是 t 上的节点，且 B 有且仅有 $B_1, \cdots, B_k$ 这 k 个儿子节点，且根据 (2)，有 $\mathsf{Th} \vdash \varphi \Rightarrow \psi$。

(4) 若 B, B' 是 t 上的节点，$B \overset{(\psi,1)}{\rightarrow} B'$，则有 $B \xrightarrow{\tau}_{\psi} B'$ 且 B 有且仅有 B' 一个儿子节点，且根据 (2)，有 $\mathsf{Th} \vdash \varphi \Rightarrow \psi$。

条件等价树实质上是概率的状态保持的静态迁移。

从定义上看条件等价树的定义比较抽象，我们可以看几个简单有趣的例子：

例 2.2 $A(x) \overset{def}{=} (x > 5)(\frac{1}{2}\tau.B(x) \oplus \frac{1}{2}\tau.C(x)) | (x \leq 5)(\frac{1}{3}\tau.B(x) \oplus \frac{2}{3}\tau.C(x))$。若 $[A(x)]_{\mathcal{E}} = [B(x)]_{\mathcal{E}} = [C(x)]_{\mathcal{E}}$，我们可以得到 $A(x)$ 的静态迁移树如图2–3，$A(x)$ 关于 $(x > 7)\mathcal{E}$ 的条件等价树如图2–4。

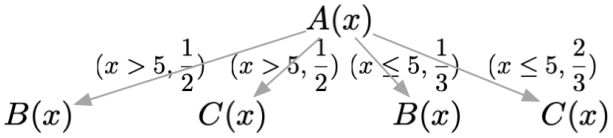

图 2–3 $A(x)$ 的静态迁移树

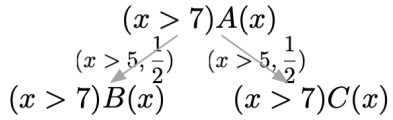

图 2–4 $A(x)$ 关于 $(x > 7)\mathcal{E}$ 的条件等价树

例 2.3 $H(x) \overset{def}{=} (x \leq 3)(\frac{1}{3}\tau.(x \leq 1)G(x) \oplus \frac{1}{3}\tau.(x \leq 2)G(x) \oplus \frac{1}{3}\tau.(x \leq 3)G(x))$

$[H(x)]_{\mathcal{E}_1} = [G(x)]_{\mathcal{E}_1}$，$H(x), G(x) \in \mathbb{RVPC}_{\mathsf{Th}}$，其中 $\mathsf{Th} = \mathsf{PA}$[18]。

$H(x)$ 的静态迁移树如图 2–5，它包含了 $H(x)$ 所有的静态迁移。

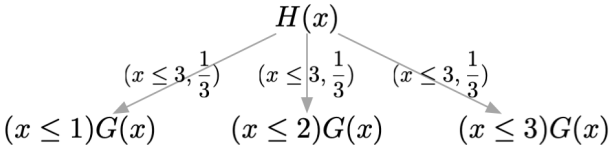

图 2–5 $H(x)$ 的静态迁移树

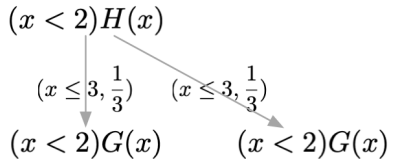

图 2–6 $H(x)$ 的 $(x < 2)\mathcal{E}_1$ 条件等价树

因为 $\mathsf{Th} \vdash \mathsf{T} \nRightarrow (x \leq 3)$，$H(x)$ 的 $\mathsf{T}\mathcal{E}_1$-tree 只有一个根节点 $H(x)$。

同理，$H(x)$ 的 $(x > 3)\mathcal{E}_1$-tree 只有一个根节点 $H(x)$。

$H(x)$ 的 $(x < 2)\mathcal{E}_1$ 条件等价树如图 2–6，其中 $\mathsf{Th} \vdash (x < 2) \Rightarrow (x \leq 3)$，由于 $(x < 2) \wedge (x \leq 2)G(x) = (x < 2)G(x)$，$G(x) \in [H(x)]_{\mathcal{E}_1}$，根据推论 2.6，$(x < 2)G(x) \in [H(x)]_{(x<2)\mathcal{E}_1}$。另一个

分支同理。此时实际有 $(x<2)H(x) \xrightarrow{\frac{2}{3}\tau}_{\mathsf{T}} [(x<2)G(x)]_{(x<2)\mathcal{E}_1}$，因此在条件树内部，我们可以将节点和边都视为状态无关。

例 2.4 假设我们有一个不太行的下课铃系统，每一时刻它坏掉的可能性是 $\frac{1}{2}$，可以通过内部自动校准系统 (静态迁移：τ 操作) 修复，它的内部有一个计时器，每隔定长时间（τ 操作）就会从 a 通道广播录好的一段下课铃，录好的下课铃可以用变元 x 表示。这个下课铃系统可以被抽象为一个 $\mathbb{RVPC}_{\mathsf{Th}}$，其中 Th 可以认为是 PA[18]，我们可以用 $G(x)$ 来表示这个系统：

$$G(x) \overset{def}{=} \mu X.(\frac{1}{2}\tau.X \oplus \frac{1}{2}\tau.\overline{a}(x).X) \tag{2–6}$$

$G(x)$ 在 $\mathcal{E}_2$ 的等价集 $[G(x)]_{\mathsf{T}\mathcal{E}_2} = [\overline{a}(x).G(x)]_{\mathsf{T}\mathcal{E}_2}$。$G(x)$ 的 $\mathsf{T}\mathcal{E}_2$ 条件等价树如图 2–7。

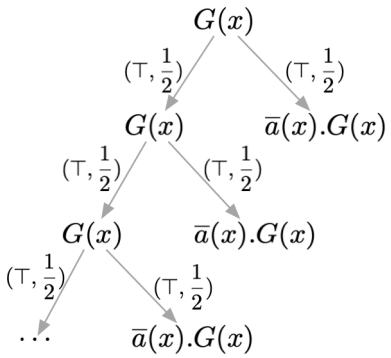

图 2–7 $G(x)$ 的 $\mathsf{T}\mathcal{E}_2$ 条件等价树

例 2.5 假设例 2.4的下课铃系统经过岁月的磨练，年久失修，每自动校准一次音量就会减弱，我们用变元 y 来表示音量，音量必须满足 $y>1$ 才可以播放，当 $y \le 0$ 时下课铃系统音量就无法减弱了。为了方便建模，我们用 $\mathsf{p}(x) = x-1$ 来表示这种音量减弱。校长出于节约经费的考虑，只要下课铃还能在他办公室听见 $(y>3)$ 就可以继续使用，现在的下课铃系统依然是一个 $\mathbb{RVPC}_{\mathsf{Th}}$，我们可以用修改的 $G'(x,y)$ 来表示这个系统：

$$G'(x,y) \overset{def}{=} (y>3)(\frac{1}{2}\tau.((y>0)\tau.G'(x,\mathsf{p}(y))|\neg(y>0)\tau.G'(x,y)) \oplus \frac{1}{2}\tau.((y>1)\overline{a}(x).G'(x,y))) \tag{2–7}$$

$G'(x,y)$ 在 $(y>3)\mathcal{E}_3$ 的等价集 $[G'(x,y)]_{(y>3)\mathcal{E}_3} = [G'(x,y')]_{(y'>3)\mathcal{E}_3} = [\overline{a}(x).G'(x,y'')]_{(y''>3)\mathcal{E}_3}$，其中 y, y', y'' 不一定相等。

$G'(x,y)$ 的 $(y>3)\mathcal{E}_3$ 条件等价树如图 2–8。其中 $\mathsf{p}(x)$ 的实现可以通过 FU Y 的 The Value-Passing Calculus 中的 Numeric System[13] 实现。

2.3.3 随机传值进程模型的符号互模拟

定义了概率的状态保持的静态迁移，我们现在可以用 Uniform Approach 的方法得到 $\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟。我们首先需要定义随机传值进程模型条件版的 l-迁移和 q-迁移。

定义 2.11 (条件 l-迁移 (φl-transition)) $A \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, \mathcal{B} \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}/\varphi'\mathcal{E}\backslash[A]_{\varphi'\mathcal{E}}$，其中 φ' 是一个布尔表达式，$\mathcal{E}$ 是 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系，若 A 的条件等价树 $t^A_{\varphi'\mathcal{E}}$ 的所有叶子结点 L，存在 $L \xrightarrow{l}_{\varphi} L' \in \mathcal{B}, l \neq \tau$，称为 φ' 条件下 A 到 $\mathcal{B}$ 的 φl-迁移，记作 $A \rightsquigarrow_{\varphi'\mathcal{E}} \xrightarrow{l}_{\varphi} \mathcal{B}$。

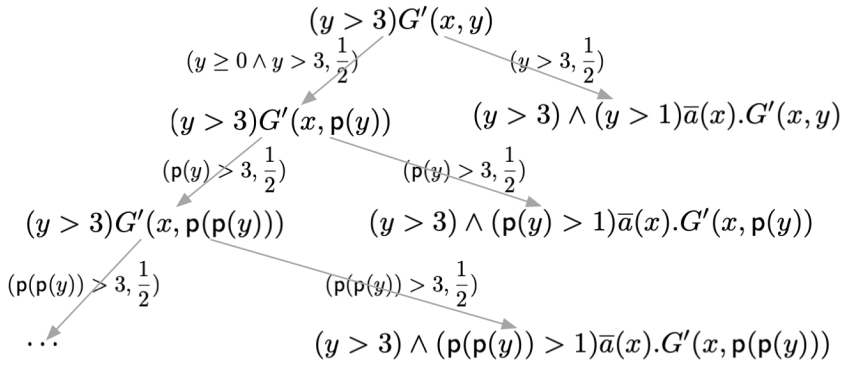

图 2–8 $G'(x,y)$ 的 $(y>3)\mathcal{E}_3$ 条件等价树

定义 2.12 (条件 q-迁移 (φq-transition)) $A\in\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, \mathcal{B}\in(\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}/\varphi'\mathcal{E})\backslash[A]_{\varphi'\mathcal{E}}$，其中 φ' 是一个布尔表达式，$\mathcal{E}$ 是 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系，对 $t^A_{\varphi'\mathcal{E}}$ 的每一个叶子结点 L 和布尔表达式 φ，若 $L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\coprod_{i\in[k]}L_i, i\in I, L_i\in\mathcal{B}$，且 $\mathsf{Th}\vdash\varphi\Rightarrow\psi_i$：

定义 $\mathsf{P}_{\varphi}(L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\ \mathcal{B})=\sum\{p_i\mid L\xrightarrow{p_i\tau}_{\psi_i}L_i\in\mathcal{B}\wedge i\in I\wedge\mathsf{Th}\vdash\varphi\Rightarrow\psi_i\}$。

定义 $\mathsf{P}_{\varphi,\varphi'\mathcal{E}}(L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\ \mathcal{B})=\mathsf{P}_{\varphi}(L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\ \mathcal{B})/(1-\mathsf{P}_{\varphi}(L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\ [A]_{\varphi\mathcal{E}}))$。

当 $\mathsf{P}_{\varphi,\varphi'\mathcal{E}}(L\xrightarrow{\coprod_{i\in I}p_i\tau}\coprod_{i\in I}\psi_i\ \mathcal{B})=q$ 时，称 φ' 条件下 A 到 $\mathcal{B}$ 存在 φq-迁移。写作 $A\rightsquigarrow_{\varphi'\mathcal{E}}\xrightarrow{q}_{\varphi}\mathcal{B}$。

符号互模拟实质上是保证 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系满足条件 l-迁移和条件 q-迁移的互模拟。

定义 2.13 (符号互模拟) $\mathcal{E}$ 是一个 $\mathcal{T}_{RVPC}$ 上的二元对称关系，当 $A\mathcal{E}B$ 且满足下列条件时，称 $\mathcal{E}$ 是一个符号互模拟 (Symbolic Bisimulation)：

(1) 若 $A\rightsquigarrow_{\varphi\mathcal{E}}\xrightarrow{a(x)}_{\varphi}L\notin[A]_{\varphi\mathcal{E}}$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$，和集合 $\{B\rightsquigarrow_{\varphi_i\mathcal{E}}\xrightarrow{a(x)}_{\psi_i}L'\in[L]_{\varphi_i\mathcal{E}}\}$，使得对 $i\in I$，$\mathsf{Th}\vdash\varphi_i\Rightarrow\psi_i$。

(2) 若 $A\rightsquigarrow_{\varphi\mathcal{E}}\xrightarrow{\bar{a}(t)}_{\varphi}L\notin[A]_{\varphi\mathcal{E}}$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$，和集合 $\{B\rightsquigarrow_{\varphi_i\mathcal{E}}\xrightarrow{\bar{a}(t_i)}_{\psi_i}L'\in[L]_{\varphi_i\mathcal{E}}\}$，使得对 $i\in I$，$\mathsf{Th}\vdash(\varphi_i\Rightarrow\psi_i)\wedge(\varphi_i\Rightarrow(t=t_i))$。

(3) 若 $A\rightsquigarrow_{\varphi\mathcal{E}}\xrightarrow{q}_{\varphi}C\in\mathcal{T}/\mathcal{E}, C\neq[A]_{\varphi\mathcal{E}}$，则存在 φ 的划分 $\{\varphi_i\}_{i\in I}$，和集合 $\{B\rightsquigarrow_{\varphi_i\mathcal{E}}\xrightarrow{q_i}_{\varphi_i}[C]_{\varphi_i\mathcal{E}}\}$，使得 $i\in I$，$q_i=q$。

如果 RCCS 的分支互模拟可以看作是用等价树模拟等价树，$\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟可以看作是等价森林模拟等价树。我们可以给出等价森林的递归定义。

定义 2.14 (条件等价森林) $A\in\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，$\mathcal{E}$ 是 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系，φ 是一个布尔表达式，对于 φ 的一个划分 $\{\varphi_i\}_{i\in I}$，A 关于 $\varphi\mathcal{E}$ 的条件等价森林有且仅有 A 关于 $\varphi_i\mathcal{E}$ 的条件等价树或条件等价森林，$i\in I$。

在证明 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 的两项符号互模拟时，我们通常可以构造一个含有该项的等价集，并证明该等价集是一个符号互模拟关系。构建一个 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 项的条件 l-迁移和条件 q-迁移时，我们可以首先构建该项的静态迁移树。

例 2.6 假设学校负责看管设备的老师发现例 2.5中的下课铃系统其实只有在 $y \geq 5$ 时才能被所有教室的同学们听到，为了不违抗校长的决定又同时让同学们都听到下课铃，他决定当 $(y < 5)$ 时用自己的电脑通过 a 通道播放下课铃，这时校长以为下课铃系统被工人修好成例 2.4中的下课铃系统，决定一直使用这个下课铃。校长的感觉是错觉吗?

此时，看管设备的老师和例 2.5中的下课铃系统依旧是一个 $\mathbb{RVPC}_{\mathsf{Th}}$，我们可以用 $G''(x,y)$ 来表示这个系统。

$G''(x,y) = (\frac{1}{2}\tau.((y>0)\tau.G''(x,\mathsf{p}(y))|\neg(y>0)\tau.G''(x,y)) \oplus \frac{1}{2}\tau.((y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y))$。

我们只需要证明 $G(x)$ 与 $G''(x,y)$ 符号互模拟即可。

证明 构造等价集

$$\begin{aligned} S = \{&(G(x), G''(x,y)),\\ &(G(x), (y>0)\tau.G''(x,\mathsf{p}(y))|\neg(y>0)\tau.G''(x,y)),\\ &(\overline{a}(x).G(x), (y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y))\}\\ &\cup\{(G(t), G''(t,y))|t\in V\}\end{aligned} \tag{2–8}$$

其中 V 是 x 的取值范围。我们只需证明 S 是一个符号互模拟关系即可。

(1) $\overline{a}(x).G(x)$ 和 $(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y)$ 关于 $(x=t)S$ 的等价树只有一个根节点，由于不涉及概率，这里实际可以使用 $\mathbb{VPC}_{\mathsf{Th}}$ 的符号互模拟证明，但由于 $\mathbb{VPC}_{\mathsf{Th}}$ 是 $\mathbb{RVPC}_{\mathsf{Th}}$ 的特例，此处依然可以使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟证明。

(a) 对于 x 的每一个赋值 $t\in V$，$\overline{a}(x).G(x) \rightsquigarrow_{(x=t)S} \xrightarrow{\overline{a}(t)}_{(x=t)} G(t)$。我们可以得到 $(x=t)$ 的一个划分：$(x=t) = ((y>1)\wedge(x=t)) \vee ((y\leq 1)\wedge(x=t))$。

这时存在集合

$$\{(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y) \rightsquigarrow_{(x=t)\wedge(y>1)S} \xrightarrow{\overline{a}(t')}_{(y>1)} G''(t',y),$$

$$(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y) \rightsquigarrow_{(x=t)\wedge(y\leq 1)S} \xrightarrow{\overline{a}(t')}_{(y<5)} G''(t',y)\}$$

可以模拟上述操作，其中 $\mathsf{Th}\vdash(x=1)\wedge(y\leq 1)\Rightarrow(y<5)$ 且 $\mathsf{Th}\vdash(x=t)\wedge(y\leq 1)\Rightarrow(t=t')$，$G''(t,y)\in[G(t)]_{(x=t)\wedge(y\leq 1)S}$。$(x=t)\wedge(y>1)$ 相同。

(b) 对 x 的每一个赋值 $t\in V$，

$$(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y) \rightsquigarrow_{(x=t)\wedge(y>1)S} \xrightarrow{\overline{a}(t)}_{(x=t)\wedge(y>1)} G''(t,y)$$

存在集合 $\{\overline{a}(x).G(x) \rightsquigarrow_{(x=t)(y>1)S} \xrightarrow{\overline{a}(t')}_{(x=t)} G(t')\}$ 可以模拟上述操作，其中 $\mathsf{Th}\vdash((x=t)\wedge(y>1))\Rightarrow(x=t)\wedge\mathsf{Th}\vdash((x=t)\wedge(y>1))\Rightarrow t=t'$，$G(t)\in[G''(t,y)]_{(x=t)\wedge(y>1)S}$。

对 x 的每一个赋值 $t\in V$，

$$(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y) \rightsquigarrow_{(x=t)\wedge(y<5)S} \xrightarrow{\overline{a}(t)}_{(x=t)\wedge(y<5)} G''(t,y)$$

存在集合 $\{\overline{a}(x).G(x) \rightsquigarrow_{(x=t)(y<5)S} \xrightarrow{\overline{a}(t')}_{(x=t)} G(t')\}$ 可以模拟上述操作，其中 $\mathsf{Th}\vdash((x=t)\wedge(y<5))\Rightarrow(x=t)\wedge\mathsf{Th}\vdash((x=t)\wedge(y<5))\Rightarrow t=t'$，$G(t)\in[G''(t,y)]_{(x=t)\wedge(y<5)S}$。

(2) $(G(t),(y>0)\tau.G''(x,\mathsf{p}(y))|\neg(y>0)\tau.G''(x,y))$ 的这一对等价关系也是常规的 $\mathbb{VPC}_{\mathsf{Th}}$ 证明，此处不做赘述。

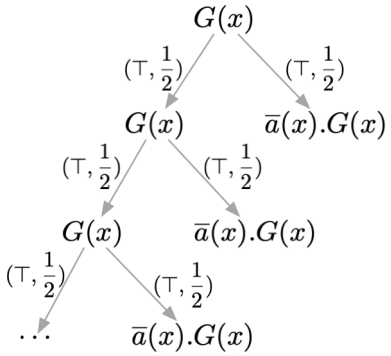

图 2–9 $G(x)$ 的静态迁移树

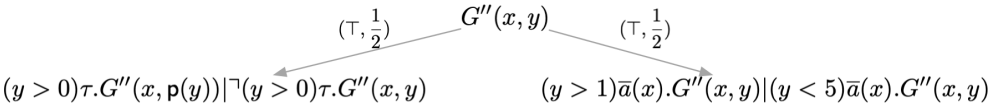

图 2–10 $G''(x,y)$ 的静态迁移树

(3) $G(x)$ 的静态迁移树如图 2–9，$G''(x,y)$ 的静态迁移树如图 2–10。

对于 $G(x) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\frac{1}{2}}_{\mathsf{T}} [\overline{a}(x).G(x)]_{\mathsf{T}\mathcal{S}}$，存在集合

$$\{G''(x,y) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\frac{1}{2}}_{\mathsf{T}} [(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y)]_{\mathsf{T}\mathcal{S}}\}$$

其中 $\mathsf{Th} \vdash (\top \Rightarrow \top)$ 且 $\frac{1}{2} = \frac{1}{2}, (\overline{a}(x).G(x), (y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y)) \in \mathcal{S}$。

对于 $G''(x,y) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\frac{1}{2}}_{\mathsf{T}} [(y>1)\overline{a}(x).G''(x,y)|(y<5)\overline{a}(x).G''(x,y)]_{\mathsf{T}\mathcal{S}}$，我们可以对称地用集合 $\{G(x) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\frac{1}{2}}_{\mathsf{T}} [\overline{a}(x).G(x)]_{\mathsf{T}\mathcal{S}}\}$ 模拟。

对于 $G''(x,y) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\frac{1}{2}}_{\mathsf{T}} [(y>0)\tau.G''(x,\mathsf{p}(y))|\neg(y>0)\tau.G''(x,y)]_{\mathsf{T}\mathcal{S}}$，由于等价关系具有传递性，$G''(x,y) \in [(y>0)\tau.G''(x,\mathsf{p}(y))|\neg(y>0)\tau.G''(x,y)]_{\mathsf{T}\mathcal{S}} = [G''(x,y)]_{\mathsf{T}\mathcal{S}}$。

因此我们可以得出校长的判断是正确的。 □

例 2.6是一个简单的例子，给出了同一系统的两种不同实现，这两种实现在外界观察者的视角中是效果相同的，即具有观察等价性。符号互模拟给观察等价性提供了严格的刻画，为观察者的主观感觉提供了客观依据。

2.4 随机传值进程模型的等价性

如 2.3.3节中所述，符号互模拟描述了 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性，因此我们可以通过符号互模拟定义随机传值进程模型的等价性。

我们首先证明符号互模拟关系是一个等价关系，众所周知一个等价关系应具有自反性、对称性、传递性。根据定义 2.13，容易得到符号互模拟具有自反性和对称性，我们接下来证明符号互模拟的传递性。

引理 2.10 符号互模拟具有传递性。

证明 证明符号互模拟的传递性等价于证明若 $\mathcal{E}$ 是一个符号互模拟，且 $A\mathcal{E}B, B\mathcal{E}C$，则 $A\mathcal{E}C$。

(1) 若 $A \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{RVPC}/\varphi\mathcal{E}, C \neq [A]_{\varphi\mathcal{E}}$，由于 $A\mathcal{E}B$，根据定义存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $S = \{B \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{\lambda}_{\psi_i} [C]_{\varphi_i\mathcal{E}} | \mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\}_{i\in I}$。对于 B 关于 $\varphi_i\mathcal{E}$ 的条件等价树 $t^B_{\varphi_i\mathcal{E}}$ 上的节点 L，因为 $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i$，若 $L \xrightarrow{\lambda}_{\psi_i} L' \in [C]_{\varphi_i\mathcal{E}}$，则 $L \xrightarrow{\lambda}_{\varphi_i} L' \in [C]_{\varphi_i\mathcal{E}}$，因此存在集合 $S' = \{B \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{\lambda}_{\varphi_i} [C]_{\varphi_i\mathcal{E}}\}_{i\in I}$，根据定义 2.13，$S'$ 也可以作为 $A \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{RVPC}/\varphi\mathcal{E}, C \neq [A]_{\varphi\mathcal{E}}$ 的模拟。

由于 $B\mathcal{E}C$，对于每一个 $B \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{\lambda}_{\varphi_i} [C]_{\varphi_i\mathcal{E}}$，根据定义存在 φ_i 的划分 $\{\varphi_{i,j}\}_{j\in J}$ 和集合 $S_i = \{C \rightsquigarrow_{\varphi_{i,j}\mathcal{E}} \xrightarrow{\lambda}_{\psi_{i,j}} [C]_{\varphi_{i,j}\mathcal{E}} | \mathsf{Th} \vdash \varphi_{i,j} \Rightarrow \psi_{i,j}\}_{j\in J}$，和集合 $S_i = \{C \rightsquigarrow_{\varphi_{i,j}\mathcal{E}} \xrightarrow{\lambda}_{\varphi_{i,j}} [C]_{\varphi_{i,j}\mathcal{E}}\}_{j\in J}$。

进而，存在 φ 的划分 $Con = \bigcup_{i\in I}\{\varphi_{i,j}\}_{j\in J}$ 和集合 $S' = \bigcup_{i\in I} S'_i$，使得 C 可以符号模拟 A。对称的证明是完全一致的。

(2) 若 $A \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{q}_{\varphi} C \in \mathcal{T}_{RVPC}/\varphi\mathcal{E}, C \neq [A]_{\varphi\mathcal{E}}, q \in (0,1]$，由于 $A\mathcal{E}B$，存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{q_i}_{\psi_i} [C]_{\varphi_i\mathcal{E}} | \mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i \wedge q = q_i\}$，因此存在 $\{B \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{q_i}_{\varphi_i} [C]_{\varphi_i\mathcal{E}}\}$ 可以模拟 A 的条件 q-迁移，余下部分与 l-迁移的证明相同。 □

证明了符号互模拟的传递性后我们可以得到定理 2.11。

定理 2.11 符号互模拟关系是等价关系。

接下来我们希望通过符号互模拟定义 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性，一种比较直接的想法是将 $\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟的全集定义为 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性，我们首先需要证明符号互模拟关系的全集仍然是符号互模拟关系。

引理 2.12 如果每一个 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的关系 $\mathcal{E}_i$ 都是符号互模拟关系，那么 $(\bigcup_{i\in I}\mathcal{E}_i)^*$ 是符号互模拟关系。

证明 令 $\mathcal{E} = (\bigcup_{i\in I}\mathcal{E}_i)^*$。给定 $A_0\mathcal{E}_{i_1}A_1\mathcal{E}_{i_2}A_2\cdots A_{k-1}\mathcal{E}_{i_k}A_k$，由于 $\mathcal{E} = (\bigcup_{i\in I}\mathcal{E}_i)^*$，我们可以得到 $A_1\mathcal{E}A_k$，若 $\mathcal{E}$ 是一个符号互模拟关系，则对于条件 l-迁移：$A_0 \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$，存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{A_k \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{\lambda}_{\psi_i} [C]_{\varphi_i\mathcal{E}} | \mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\}$ 可模拟该操作。我们可以通过归纳的证明 $A_0\mathcal{E}A_1$，A_1 可以模拟 A_0；$A_1\mathcal{E}A_2$，A_2 可以模拟 A_1；…；最终证明 A_k 可以模拟 A_0。

对于 $A_0 \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$，我们需要给出 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}} \xrightarrow{\lambda}_{\psi_i} [C]_{\varphi_i\mathcal{E}}\}$ 来模拟 A_0 的动作。

考虑 $A_0 \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$，由等价集的定义，我们可以得出 $\varphi\mathcal{E}$ 上的等价集 C 可以被划分成不相交的 $\varphi\mathcal{E}_{i_1}$ 上的等价集 $\{C^{i_1}_j\}$。假设 t_{A_0} 是 A_0 关于 $\varphi\mathcal{E}$ 的条件等价树，我们可以递归的构建出 A_1 关于 $\varphi\mathcal{E}$ 的条件等价森林。

(1) t_{A_0} **的根节点** A_0 **只有一个儿子节点** A'_0。若 $A'_0 \in [A_0]_{\varphi\mathcal{E}_{i_1}}$，我们将以 $\varphi A'_0$ 作为下次递归的根节点，构建 φA_1 条件等价森林（这样递归是因为在 A_1 的 $\varphi\mathcal{E}$ 条件等价森林中的所有节点的动作都满足条件 φ，A_1 和 φA_1 是等价的）。若 $A'_0 \notin [A_0]_{\varphi\mathcal{E}_{i_1}}$，根据 $A_0\mathcal{E}_{i_1}A_1$，我们可以得到 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \xrightarrow{\tau}_{\psi_i} [C]_{\varphi_i\mathcal{E}_{i_1}} | \mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\}_{i\in I}$。

根据推论 2.10的证明经验，我们其实可以用 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \xrightarrow{\tau}_{\varphi_i} [C]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$ 模拟 $A_0 \rightsquigarrow_{\varphi\mathcal{E}} \xrightarrow{\tau}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$，为了方便证明，后续的证明中我们将直接

使用 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{\lambda}{\to}_{\varphi_i} [C]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$ 的形式。

$\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{\tau}{\to}_{\varphi_i} [C]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$ 实际构建出了一个条件等价树的集合 $\{t^i_{A_1}\}_{i\in I}$，对于每个条件等价树 $t^i_{A_1}$ 的叶子节点 B，$B \overset{\tau}{\to}_{\varphi_i} B' \in [A'_0]_{\varphi_i\mathcal{E}_{i_1}}$，我们根据 $\varphi_i A'_0$ 构建每一个 $\varphi_i B'$ 关于 $\varphi_i\mathcal{E}$ 的等价森林。此时将 B 所在的 $\varphi_i\mathcal{E}$ 等价树 $t^i_{A_1}$ 复制并与每个 B' 的 $\varphi_i\mathcal{E}$ 的等价森林中的等价树相连，我们可以得到 A_1 关于 $\varphi\mathcal{E}$ 的条件等价森林。

(2) t_{A_0} **的根节点** A_0 **有** h **个儿子节点** $A^1_0, \cdots, A^h_0$。根据定义 $A_0 \overset{\amalg_{i\in[h]} p_j\tau}{\to}_{\psi} \coprod_{j\in[h]} A^j_0$，$A_0$ 到 A^j_0 的边被标记为 (ψ, p_j)，其中 $\varphi \Rightarrow \psi$。

若所有 $A^j_0 \in [A_0]_{\varphi\mathcal{E}_{i_1}}$，我们将根据 φA^1_0 构建 φA_1 的条件等价森林。

若存在 $A^j_0 \notin [A_0]_{\varphi\mathcal{E}_{i_1}}$，不妨设为 A^1_0，则 $A_0 \rightsquigarrow_{\varphi\mathcal{E}_{i_1}} \overset{q}{\to}_{\varphi} [A^1_0]_{\varphi\mathcal{E}_{i_1}}$，根据 $A_0\mathcal{E}_{i_1}A_1$，我们可以得到 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{q}{\to}_{\varphi_i} [A^1_0]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$，这其中关系到 A_1 关于 $\varphi_i\mathcal{E}_{i_1}$ 的条件等价树 t^i_1，根据条件 q-迁移的定义，存在 $\mathsf{P}_{\varphi_i,\varphi_i\mathcal{E}_{i_1}}(N \overset{\amalg_{i'\in[h']} p_{i'}\tau}{\longrightarrow}_{\varphi_i} N' \in [A^1_0]_{\varphi_i\mathcal{E}_{i_1}}) = q$，我们根据 $\varphi_i A^1_0$ 构建每个 $\varphi_i N'$ 关于 $\varphi_i\mathcal{E}$ 的条件等价森林。

(3) t_{A_0} **的根节点** A_0 **可以作** $A_0 \overset{\lambda}{\to}_{\varphi} L'$。根据定义存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{\lambda}{\to}_{\varphi_i} [L']_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$。

上述递归的构建可以通过图 2–11更加形象的表述。图 2–11中，第一层描述了根节点只

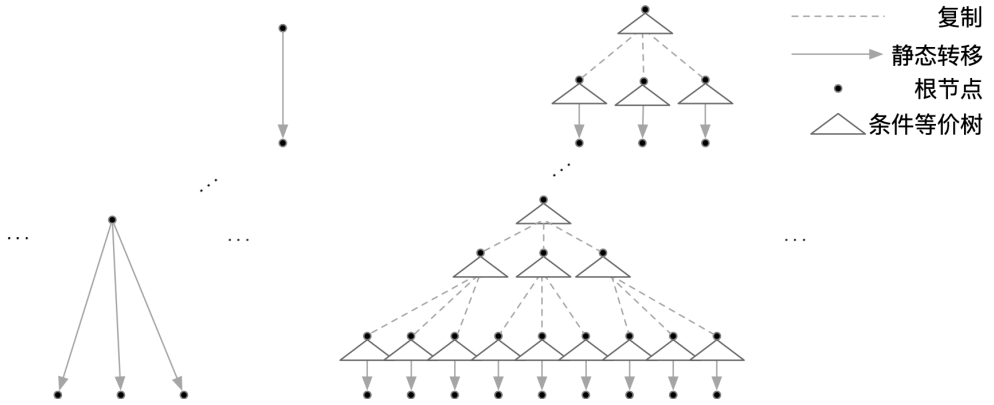

图 2–11 递归构建图解

有一个儿子节点情况的构建方法，对于每一个 $i \in I$，右侧的条件等价树都会复制一份，全部的条件等价树构成 $\varphi\mathcal{E}$ 条件等价森林，作 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{\tau}{\to}_{\varphi_i} [A'_0]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$ 对 $A_0 \overset{\tau}{\to}_{\varphi\mathcal{E}_{i_1}} A'_0$ 的模拟。

第二层描述了根节点有多个儿子节点情况的构建方法，对于每一个 $A_0 \rightsquigarrow_{\varphi\mathcal{E}_{i_1}} \overset{q}{\to}_{\varphi}$，右侧的条件等价树会复制一份，在对每一个 $A_0 \rightsquigarrow_{\varphi\mathcal{E}_{i_1}} \overset{q}{\to}_{\varphi} [A^1_0]_{\varphi\mathcal{E}_{i_1}}$ 模拟时，对模拟该动作的集合 $\{A_1 \rightsquigarrow_{\varphi_i\mathcal{E}_{i_1}} \overset{q}{\to}_{\varphi_i} [A^1_0]_{\varphi_i\mathcal{E}_{i_1}}\}_{i\in I}$ 中的每一个元素都复制一份条件等价树，再进行对应的模拟。

通过上述构建方式，我们最终会构建出 A_1 关于 $\varphi\mathcal{E}$ 的条件等价树或条件等价森林，来模拟 $A_0 \rightsquigarrow_{\varphi\mathcal{E}} \overset{\lambda}{\to}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$。归纳地，我们可以构建出 A_2 对 A_1 关于 $\varphi\mathcal{E}$ 的条件等价树或条件等价森林中的每一个条件等价树的迁移的条件等价森林……最终构建出 A_k 模拟 $A_0 \rightsquigarrow_{\varphi\mathcal{E}} \overset{\lambda}{\to}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}, C \neq [A_0]_{\varphi\mathcal{E}}$ 的条件等价森林。

对于条件 q-迁移：$A_0 \rightsquigarrow_{\varphi\mathcal{E}} \overset{q}{\to}_{\varphi} C$ 的证明方法是相同的。 □

我们希望定义符号互模拟关系的全集为 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的观察等价性，但这样定义仍会存在问题：考虑 $A \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，若 A 在 $\varphi\mathcal{E}$ 下的条件等价树是无限延伸的，即没有叶子节点，例如：

$$A(x) \overset{def}{=} (\overline{a}(x).0|B|C)\backslash\{a,b,c\}$$

$$B \overset{def}{=} a(x).B(x) \qquad B(x) \overset{def}{=} \frac{1}{2}\tau.\overline{b}(x).B \oplus \frac{1}{2}\tau.\overline{c}(x).B$$

$$C \overset{def}{=} b(x).C(x) \qquad D \overset{def}{=} c(x).D(x)$$

$$C(x) \overset{def}{=} \overline{a}(x).C \qquad D(x) \overset{def}{=} \overline{a}(x).D$$

若定义等价关系 $[A(x)]_{\mathcal{E}} = [B(x)|C|D]_{\mathcal{E}} = [B|C(x)|D]_{\mathcal{E}} = [B|C|D(x)]_{\mathcal{E}}$，这时 $A(x)$ 的 $\top\mathcal{E}$ 条件等价树如图 2–12。在观察等价性的角度，我们认为 $E(x) \overset{def}{=} \tau.E(x)$ 与 $A(x)$ 应该是观察

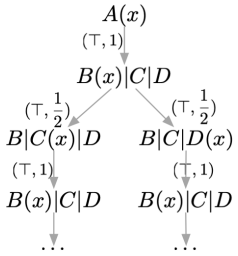

图 2–12 $A(x)$ 的 $\top\mathcal{E}$ 条件等价树

互模拟的，然而，这时找到 $A(x)$ 的条件 l 转移和条件 q 转移就会变得困难，因此我们引入 Uniform Approach 中的共发散（codivergent）[11] 的相关概念，并将其扩展为 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 下的共发散来解决这个问题。

定义 2.15 (发散) 对于 $A \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$，若 A 在 $\varphi\mathcal{E}$ 下的条件等价树为 t，定义树 t 的分支 π 是 t 从根节点开始到叶节点结束的一条路径，定义 $\pi(i)$ 为分支 π 上第 i 条边的标记中的概率部分，$\pi(i) \in (0,1]$。

定义这条分支的概率 $\mathsf{P}(\pi) = \prod\{\pi(i)|i \in [|\pi|]\}$，若这条分支是无穷的，则 $\mathsf{P}(\pi) = \lim_{j\to\infty}\prod_{i=0}^{j}\pi(i)$。

定义条件等价树的概率 $\mathsf{P}(t) = \sum\{\mathsf{P}(\pi)|\pi\text{是}t\text{的一条分支}\}$。

定义 t 的前 k 层分支的概率为 $\mathsf{P}^k(t) = \sum\{\mathsf{P}(\pi)|\pi\text{是}t\text{的一条分支且}|\pi| \le k\}$。

定义 t 的有限长分支的概率 $\mathsf{P}^f(t) = \lim_{k\to\infty}\mathsf{P}^k(t)$。

当条件等价树 t 的有限长分支的概率 $\mathsf{P}^f(t) = 0$ 时，t 称为发散的条件等价树，我们可以理解为 t 没有有限长的分支。

通过定义 2.15，我们定义了使用符号互模拟关系的全集作为观察等价性时，存在问题的情况，我们使用共发散的概念对这种情况的条件等价树的观察等价性单独定义。

定义 2.16 (共发散) 若 $\mathcal{E}$ 是 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的等价关系，$\mathcal{E}$ 是一个共发散的等价关系当且仅当：当对于布尔表达式 φ，条件等价集 $\mathcal{C} \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}/\varphi\mathcal{E}$，$\mathcal{C}$ 中的每一个元素关于 $\varphi\mathcal{E}$ 的条件等价森林中的每一个棵树都是发散的条件等价树；或 $\mathcal{C}$ 中的每一个元素关于 $\varphi\mathcal{E}$ 的条件等价森林中的每一个棵树都不是发散的条件等价树。

定义 2.17 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的观察等价性 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 定义为 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 上的共发散的符号互模拟关系的全集。

由于 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 仍然是一个符号互模拟关系，因此我们可以得到定理 2.13。

定理 2.13 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 是一个等价关系。

在数学特别是抽象代数中，同余关系或简称同余是相容于某个代数运算的等价关系。设 $A =< S, *, \delta >$ 是一个代数系统，$\sim$ 是载体 S 上的等价关系，若 $\sim$ 在 A 上的所有运算下都是可保持的，则称 $\sim$ 为代数系统 A 上的同余关系[30]。同余关系使得元素所在的等价类在运算上可以作为一个整体来看待。我们希望证明我们定义的 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 是一个同余关系。

定理 2.14 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 具有同余性。

证明 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 对于随机选择和非确定性选择的可保持性比较容易证明。由推论 2.7易知 $A =_{\mathbb{RVPC}_{\mathsf{Th}}} B$ 则 $\varphi A =_{\mathbb{RVPC}_{\mathsf{Th}}} \varphi B$。为方便证明，以下 $=_{\mathbb{RVPC}_{\mathsf{Th}}}$ 简记为 $=$，绝对等价性记为 $\equiv$，定义记为 $\stackrel{def}{=}$。

对于内部化操作子（Localization），对于 $A = B, (a)A \rightsquigarrow_{\varphi=} \xrightarrow{\lambda}_{\varphi} (a)A' \notin [(a)A]_{\varphi=}, a \notin \lambda$，易知有 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{(a)B \rightsquigarrow_{\varphi_i=} \xrightarrow{\lambda}_{\varphi_i} [(a)A]_{\varphi=}\}$。

对于并发操作子（Composition），考虑等价关系 $\mathcal{R} \stackrel{def}{=} \{(A|C, B|D) | (A, B) \in =, (C, D) \in =\}$，令 $\mathcal{R}' = (\mathcal{R} \cup =)^*$，我们证明 $\mathcal{R}'$ 是一个符号互模拟关系。对于条件 l-迁移：$(A|C) \rightsquigarrow_{\varphi\mathcal{R}'} \xrightarrow{\lambda}_{\varphi} C \in \mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}/\varphi\mathcal{R}', C \neq [(A|C)]_{\varphi\mathcal{R}'}$，我们可以用引理 2.12中的方法递归的构造 $(B|C)$ 的 $\varphi\mathcal{E}$ 条件等价森林来模拟 $(A|C)$ 的动作，令 $t_{(A|C)}$ 为 $(A|C)$ 关于 $\varphi\mathcal{R}'$ 的条件等价树。

(1) $t_{(A|C)}$ 上的边：$A|C \xrightarrow{(\psi,1)} A'|C, \mathsf{Th} \vdash \varphi \Rightarrow \psi$ 是由于 $A \xrightarrow{\tau}_{\varphi} A'$。

若 $A' \in [A]_{\varphi=}$，那么 $(A'|C) \in [(A|C)]_{\varphi\mathcal{R}'} \equiv [(B|D)]_{\varphi\mathcal{R}'}$。实际上，对于 $A \xrightarrow{\coprod_{i\in I} p_i\tau}_{\varphi} \coprod_{i\in I} A_i = A$，$A_i|C \in [B|D]_{\varphi\mathcal{R}'}$。我们可以根据 $\varphi(A_0|C)$ 的 $\varphi\mathcal{R}'$ 条件等价树构建 $(B|D)$ 的 $\varphi\mathcal{R}'$ 条件等价森林。

若 $A' \notin [A]_{\varphi=}$，那么根据定义存在 $\{B \rightsquigarrow_{\varphi_i=} \xrightarrow{\tau}_{\varphi_i} B_i \in [A']_{\varphi_i=}\}_{i\in I}$，模拟 $A \rightsquigarrow_{\varphi=} \xrightarrow{\tau}_{\varphi} A'$，其中 $\{\varphi_i\}_{i\in I}$ 是 φ 的划分。我们根据 $\varphi_i A'$ 构建 $\varphi_i B_i$ 的 $\varphi_i\mathcal{R}'$ 条件等价森林。

(2) $t_{(A|C)}$ 上的边：$A|C \xrightarrow{(\psi,1)} A'|C', \mathsf{Th} \vdash \varphi \Rightarrow \psi$ 是由于 $A \xrightarrow{\overline{a}(t)}_{\psi'} A', C \xrightarrow{a(x)}_{\psi''} C', \psi = \psi'\psi''$。

根据 $A = B, C = D$，$A \rightsquigarrow_{\varphi=} \xrightarrow{\overline{a}(t)}_{\varphi} A'$ 可以由 $\{B \rightsquigarrow_{\varphi_i=} \xrightarrow{\overline{a}(t)}_{\varphi_i} B_i \in [A']_{\varphi_i=}\}_{i\in I}$ 模拟，其中 $\{\varphi_i\}_{i\in I}$ 是 φ 的划分。$C \rightsquigarrow_{\varphi=} \xrightarrow{a(x)}_{\varphi} C'$ 可以由 $\{D \rightsquigarrow_{\varphi_j=} \xrightarrow{a(x)}_{\varphi_j} D_j \in [C']_{\varphi_j=}\}_{j\in J}$ 模拟，其中 $\{\varphi_j\}_{j\in J}$ 是 φ 的划分。

对于所有的 $i \in I, j \in J, \mathsf{Th} \vdash \varphi_i\varphi_j \not\Rightarrow \bot$，$(B_i \mid D_j) \in [A' \mid C']_{\varphi_i\varphi_j\mathcal{R}'}$，我们根据 $\varphi_i\varphi_j(A'|C')$ 的 $\varphi_i\varphi_j\mathcal{R}'$ 条件等价树构建 $\varphi_i\varphi_j(B_i|D_j)$ 的 $\varphi_i\varphi_j\mathcal{R}'$ 条件等价森林。

(3) $t_{(A|C)}$ 上的边：$A|C \xrightarrow{(\psi,q)} A'|C, \mathsf{Th} \vdash \varphi \Rightarrow \psi, q \in (0,1)$，则存在 $A \xrightarrow{\coprod_{i\in I} p_i\tau}_{\varphi} \coprod_{i\in I} A_i$，对所有的 $i \in I$，$A|C \xrightarrow{(\psi,p_i)} A_i|C, \mathsf{Th} \vdash \varphi \Rightarrow \psi$。

若对所有的 $i \in I, A_i \in [A]_{\varphi=}$，我们可以根据 $(A_i|C)$ 关于 $\varphi\mathcal{R}'$ 的条件等价树构建 $(B|D)$ 的 $\varphi\mathcal{R}'$ 条件等价森林。

若存在 $A_i \notin [A]_{\varphi=}$，不妨设为 A_0，若有 $A_1 \neq A_0 \wedge A_1|C \notin [A|C]_{\varphi=} \wedge A_1|C \in [A_0|C]_{\varphi=}$，此时有 $A \rightsquigarrow_{\varphi=} \xrightarrow{q_0}_{\varphi} [A_0]_{\varphi=}$，$A \rightsquigarrow_{\varphi=} \xrightarrow{q_1}_{\varphi} [A_1]_{\varphi=}$，$A|C \rightsquigarrow_{\varphi=} \xrightarrow{q_0+q_1}_{\varphi} [A_0|C]_{\varphi=}$。

根据 $A = B$，存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B \rightsquigarrow_{\varphi_i=} \xrightarrow{q_0}_{\varphi_i} B_i \in [A_0]_{\varphi_i=}\}_{i\in I}$ 模拟 $A \rightsquigarrow_{\varphi=} \xrightarrow{q_0}_{\varphi} [A_0]_{\varphi=}$。存在 φ 的划分 $\{\varphi_j\}_{j\in J}$ 和集合 $\{B \rightsquigarrow_{\varphi_j=} \xrightarrow{q_1}_{\varphi_j} B_j \in [A_1]_{\varphi_j=}\}_{i\in I}$

模拟 $A \rightsquigarrow_{\varphi=} \xrightarrow{q_1}_{\varphi} [A_1]_{\varphi=}$。对所有的 $i \in I, j \in J, \mathsf{Th} \vdash \varphi_i\varphi_j \not\Rightarrow \bot$，我们有 $(B|D) \rightsquigarrow_{\varphi_i\varphi_j=} \xrightarrow{q_0+q_1}_{\varphi_i\varphi_j} (B'|D) \in [A_0|C]_{\varphi_i\varphi_j=}$，我们根据 $\varphi_i\varphi_j(A_0|C)$ 的 $\varphi_i\varphi_j\mathcal{R}'$ 条件等价树构建 $\varphi_i\varphi_j(B'|D)$ 的 $\varphi_i\varphi_j\mathcal{R}'$ 条件等价森林。

(4) 若 $(A|C) \xrightarrow{\lambda}_{\varphi} (A'|C), \lambda \neq \tau$，则 $A \xrightarrow{\lambda}_{\varphi} A'$，根据定义存在 φ 的划分 $\{\varphi_i\}_{i\in I}$ 和集合 $\{B|D \rightsquigarrow_{\varphi_i=} \xrightarrow{\lambda}_{\varphi_i} (B'|D) \in [(A'|C)]_{\varphi_i=}\}_{i\in I}$。

对于条件 q-迁移：$(A|C) \rightsquigarrow_{\varphi=} \xrightarrow{q}_{\varphi} C$ 的证明方法是相同的。

在上述证明中，若 $(A, B) \in =$，且 A 关于 $\varphi =$ 的条件等价树是发散的，根据定义可知 B 对应的条件等价森林中的条件等价树也是发散的，我们可以根据同样的扩展方法证明共发散对上述操作子的封闭性。 □

2.5 本章小结

本章我们介绍了一种经典的并发进程模型——传值进程算子（The Value-Passing Calculus）和它的一种实现 $\mathbb{VPC}_{\mathsf{Th}}$，我们分析了 $\mathbb{VPC}_{\mathsf{Th}}$ 对比其他传值进程模型的优势——不依赖神域，是一个封闭的模型，因此具有很强的表达能力。考虑到 $\mathbb{VPC}_{\mathsf{Th}}$ 的优势和表达能力，我们在 $\mathbb{VPC}_{\mathsf{Th}}$ 的基础上进行概率扩展获得随机传值进程模型。

首先，我们使用 Uniform Approach 中的方法为 $\mathbb{VPC}_{\mathsf{Th}}$ 添加随机选择操作子，扩展成为随机传值进程模型 $\mathbb{RVPC}_{\mathsf{Th}}$，并给出了随机传值进程模型 $\mathbb{RVPC}_{\mathsf{Th}}$ 的语法和迁移语义。

其次，为了给出 $\mathbb{RVPC}_{\mathsf{Th}}$ 观察等价性的定义，我们介绍了分支互模拟，分析了 Uniform Approach 得到分支互模拟的随机版本的关键是将非概率的状态保持的静态迁移扩展为了概率下的状态保持的静态迁移树。同时，我们引入了 Uniform Approach 中的等价集、$\mathbb{VPC}_{\mathbb{Th}}$ 中的符号互模拟的概念，由于 $\mathbb{RVPC}_{\mathsf{Th}}$ 中涉及到条件操作子，我们不能直接使用 Uniform Approach 中的扩展方法。因此我们将等价集的概念扩展为条件等价集，参考 Uniform Approach 中等价树的概念，定义了 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 的条件等价树。为了解决条件操作子的问题，我们还提出了条件等价森林的概念，进而提出使用条件等价森林模拟条件等价树的思想，并使用条件等价树的概念给出了 $\mathcal{T}_{\mathbb{RVPC}_{\mathsf{Th}}}$ 的符号互模拟关系的定义，也就是符号互模拟的概率版本。为了解决无限的条件等价树无法得到符号互模拟关系的问题，我们根据 Uniform Approach 共发散的概念提出了 $\mathbb{RVPC}_{\mathsf{Th}}$ 上共发散的定义，我们将共发散的符号互模拟的全集定义为 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性，并证明了 $\mathbb{RVPC}_{\mathsf{Th}}$ 观察等价性是一个同余关系。

第三章　随机传值进程模型的应用

如前文所述，传值进程模型可以用于对通信协议的形式刻画，我们可以使用随机传值进程模型对引入了随机性的通信协议进行形式刻画。本章中我们使用随机传值进程模型对基于云计算协议 Gossip-Style Membership 协议的通信过程进行建模和模拟实现，作为应用案例，以证明 $\mathbb{RVPC}_{\mathsf{Th}}$ 对并发通信系统的建模和分析具有一定的可行性。

3.1　Gossip-Style Membership 协议

3.1.1　Gossip 协议

Gossip 协议，也称为流言协议、传染病协议 (Epidemic Protocol) 是一个基于传染病传播方式的点对点通信协议[31]。

Gossip 协议可以被解释为办公室流言的传播：每一个职员会随机和另一个职员分享最近的流言，被分享者得知流言后会随机的和其他人分享，在一次分享中，被分享者或许已经知道了这个流言。比如有一天，小赵传出老板的一个谣言，他在一次会议结束时将这个谣言告诉了小钱；小钱得知了谣言后，又在一次会议后将它告诉了小李；小李在一次会议后告诉小孙时，发现小孙已经从小王那里知道了这个谣言。

Gossip 协议主要作分布式数据库系统中各个副本节点同步数据之用，这种场景的一个最大特点就是组成的网络的节点都是对等节点，是非结构化网络。2015 年之前，Bitcoin 就是使用了 Gossip 协议来传播交易信息[32]。

Gossip 协议中，节点之间的通信方式有三种：

(1) **Push Gossip:**
消息的发送者周期性的随机选择 k 个目标节点发送 Gossip Message。接收到 Gossip Message 的节点可以根据本地时间，周期性的选择 k 个目标节点发送 Gossip Message。在发送过程中，已经拥有 Gossip Message 的节点仍然可以被选为目标节点。

(2) **Pull Gossip:**
每个节点周期性的向 k 个目标节点发送 Gossip Query，收到 Gossip Query 的节点若拥有 Gossip Message 的节点会向发送 Query 的节点返回 Gossip Message 的拷贝。

(3) **Push/Pull Gossip:**
在超过 $\frac{n}{2}$ 的节点拥有 Gossip Message 时，可以证明此时选择 Pull Gossip 会比 Push Gossip 传播的更快[31]。因此在使用 Gossip 协议时，常使用 Push Gossip 与 Pull Gossip 的混合：在消息传播 $\frac{n}{2}$ 节点之前使用 Push Gossip，在消息传播 $\frac{n}{2}$ 之后使用 Pull Gossip。

3.1.2　组成员协议

组成员协议 (Group Membership Protocol，后文简称 Membership 协议)[33] 为集群中的每个节点提供了一个本地的列表，称为 Membership List，用来维护集群中其他节点的信息。Membership 协议提供了两个主要的服务：

(1) 检测失效的节点

(2) 传播消息，如：告知其他节点失效信息

常见的 Membership 协议有 Heartbeating Protocol[34], Gossip-Style Membership Protocol[35], SWIM Failure Detector Protocol[19] 等。

3.2 基于 Gossip-Style Membership 协议的通信系统的实现

我们使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 来实现 Gossip-Style Membership 协议[35]，来作为使用随机传值进程模型建模通信过程的示例，类似的，也可以使用同样的建模方法建模其他现实问题。

由于 Membership 可以看作节点内部的功能，我们可以首先实现 Gossip 协议，将网络建立起来。

3.2.1 基于 Gossip 协议的通信系统的实现

由于 Push Gossip 的机制比较简单，并且单一的 Push Gossip 协议仍然可以达到 $O(\log N)$ 时间复杂度的消息传播[31]，本节我们使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 来建模以 Push Gossip 作为通信机制的 Gossip 协议。

我们首先来建模以 Gossip 协议为通信的 Peer-to-Peer Sysyem 的节点。根据根据 Gossip 协议的定义，我们可以从单一节点周期性的发送消息，这意味着我们需要一个计时器机制，我们选择在节点外部使用计时器周期性的通知节点发送消息，节点的 $\overline{time}$ 端口用于向计时器设置计时开始，$timeout$ 端口用于接收计时器传来的 timeout 信号。由于节点本身不产生和存储消息，我们需要使用 $accept$ 端口接收 P2P 系统外界消息，使用 $\overline{deliver}$ 向 P2P 系统外界传递节点在 P2P 系统中从其他节点收到的消息。由于现实网络是复杂的，我们将每一对节点之间的消息传播路径抽象为单独的、单工的通道，如节点 $Node_i$ 向 $Node_j$ 传播消息时，消息会在 $trans_{i,j}$ 通道中，从 $Node_i$ 节点的 $\overline{trans_{i,j}}$ 端口发出，从 $Node_j$ 节点的 $trans_{i,j}$ 端口接收。这里的消息传输方式实际上是信息的逻辑传输路径，信息的物理传输路径可能是十分复杂且动态的，我们可以将系统中节点的编号简单的理解为 Socket 通信中的 IP 地址：端口号，每一个节点代表了某台主机上的某个进程。

Gossip 单节点的通道示意图如图 3–1和以 Gossip 协议作为通信协议的四节点的 Peer-to-Peer Sysyem 的示意图如图 3–2。

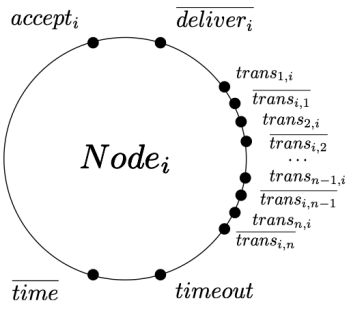

图 3–1 Gossip 节点示意图

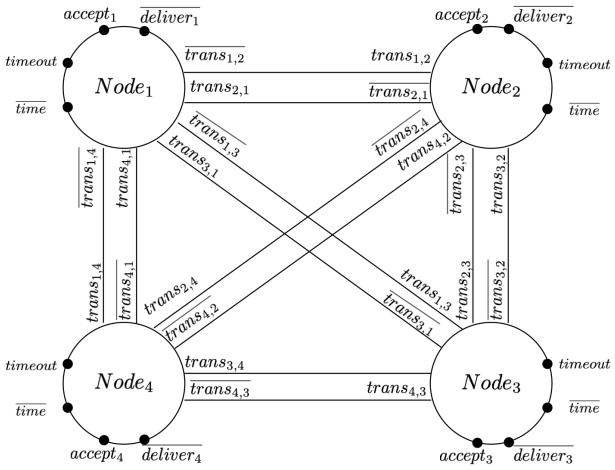

图 3–2 基于 Gossip 协议的四节点 P2P 系统结构

Gossip 节点 Node 的状态有以下几种，分别对应 Gossip 协议中节点的状态:

表 3–1 节点状态对应表

节点状态	Gossip 状态
Node	可接受系统外信息（除此状态其余状态均无法接收外界信息）
DeliveringNode	可向系统外传递信息
UnInfectiousNode	未获取 Gossip Message
InfectiousNode	已获取 Gossip Message
GossipingNode	可向系统内部特定 k 个其他节点发送 Gossip Message

基于以上定义的节点结构和节点状态，我们定义一个基于 Gossip 协议的 P2P 系统。

$$
\begin{aligned}
Node_i &\overset{def}{=} accept_i(x).DeliveringNode_i(x)\\
DeliveringNode_i(x) &\overset{def}{=} \overline{deliver_i}(x).InfectiousNode_i(x)\\
InfectiousNode_i(x) &\overset{def}{=} timeout.(\bigoplus_{perm\in \mathsf{PERM}_i} p_{perm}\tau.GossipingNode_{i,perm}(x))\\
GossipingNode_{i,perm}(x) &\overset{def}{=} \overline{trans_{i,perm_1}}(x).\cdots\overline{trans_{i,perm_k}}(x).\overline{time}.InfectiousNode_i(x)\\
UnInfectiousNode_i &\overset{def}{=} \sum_{j\in \mathsf{N}/\{i\}} trans_{j,i}(x).DeliveringNode_i(x)\\
GossipSystem_{\mathsf{N}} &\overset{def}{=} (Node_1 \mid UnInfectiousNode_2 \mid \cdots \mid UnInfectiousNode_n)\\
&\quad \backslash\{trans_{i,j} \mid i\in \mathsf{N}\wedge j\in \mathsf{N}\wedge i\neq j\}\cup\{time, timeout\}
\end{aligned}
\tag{3–1}
$$

在这个 P2P 系统中共有 n 个节点，标号为 $\mathsf{N}=\{1,2,\cdots,n\}$，每一个拥有 Gossip Message 的节点周期性的向 $\mathsf{k}(\mathsf{k}\le n-1)$ 个其他节点发送 Gossip Message。其中 PERM_i 为 $\mathsf{N}/\{i\}$ 中任选 k 个元素的全排列，p_{perm} 为向一个特定的排列 $perm$ 发送 Gossip Message 的概率，为了方便建模，我们可以假设一个节点选择任意其他节点作为 Gossip 的目标节点的概率是相同的，即 $p_{perm}=\frac{1}{A^{\mathsf{k}}_{n-1}}=\frac{(n-\mathsf{k}-1)!}{(n-1)!}$ 为固定值，当 $\mathsf{k}=2$ 时，$p_{perm}=\frac{1}{(n-1)(n-2)}$。

我们可以看到节点 $Node_i$ 接受到外界消息 x 后，迁移为 $DeliveringNode_i(x)$ 的状态，向外界递送这个消息，进而迁移为 $InfectiousNode_i(x)$ 的状态，可以在接受到 $timeout$ 信号时以概率 p_{perm} 的概率选定 $\mathsf{N}/\{i\}$ 中任选 k 个元素的全排列中的一个排列 $perm$ 作为发送目标，进而迁移为 $GossipingNode_{i,perm}(x)$，向 $perm$ 中的节点发送 Gossip Message，发送完成后回到 $InfectiousNode_i(x)$ 的状态。在一个 $GossipSystem_{\mathsf{N}}$ 中，还有状态为 $InfectiousNode_i$ 的节点，这些节点在接受了来自其他节点的 Gossip Message 时会迁移到 $DeliveringNode_i(x)$ 的状态。为了后面的分析，上述定义基于一系列理想条件的假设：

(1) 网络传输可靠。然而现实中的网络传输存在丢包、延迟、比特反转等问题，我们可以通过 ACK 机制来解决。对不可靠网络下的 Gossip 协议，我们可以在上述理想条件下增加对网络传输的过程的建模，建模过程在 MILNER R 的 CCS 中有提及[3]。

(2) 节点不会损坏。在现实中节点（主机）在运行了一定时间后就会出现问题，对于多个节点构成的 P2P 系统，存在节点失效的概率只会更高[36]，Membership 协议就是用来探测节点失效的一种方式。

(3) 系统中只存在一个消息的传输。对于单个消息源的多个消息，我们仍然可以看作单个 Gossip Message 一起发送；对于多个消息源，我们可以为每个节点提供消息队列机

制来保存多个消息，后文对 Membership 的建模提供了一个多消息源的解决思路。

3.2.2 Gossip 协议的等价关系

Gossip 协议的目的是为了在系统内对节点进行多播，我们可以定义一个多播的规约，证明 3.2.1节中我们使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 实现的基于 Gossip 协议的 P2P 系统满足这个规约，在进程演算中，我们需要证明二者互模拟。由于我们在 3.2.1节中实现的是单消息源单个消息的传播，在多播的定义中同样我们只关注单消息源单个消息的多播。

定义 3.1 单消息源的多播规约定义如下：

$$\begin{aligned} MulticastSpec_{\mathsf{N}} &\overset{def}{=} accept_1(x).\overline{deliver_1}(x).Multicasting_{\mathsf{N},\{1\}}(x) \\ Multicasting_{\mathsf{N},\mathsf{KNOWN}}(x) &\overset{def}{=} \overline{deliver_i}(x).Multicasting_{\mathsf{N},\mathsf{KNOWN}\cup\{i\}}(x), \\ & i \in \mathsf{N} - \mathsf{KNOWN} \wedge |\mathsf{N}| \neq |\mathsf{KNOWN}| \end{aligned} \tag{3–2}$$

其中 KNOWN 为已经得到此消息的节点编号集合，节点可以通过 $deliver$ 通道告知通信系统外（如本地的其他进程）节点已收到信息。若一个节点对系统外通过 $deliver$ 通道传递了这个消息，我们将这个节点加入 KNOWN，表示这个节点已经收到了消息。

定义 3.2 由于 $GossipSystem_{\mathsf{N}}$ 中的节点可能处于：$InfectiousNode$，$DeliveringNode$，$UnInfectiousNode$ 三种状态（我们将 $GossipingNode$ 的状态合并进了 $InfectiousNode$），在后续的证明过程中表示起来比较冗长，为了方便后续的证明，我们给出特定状态下的 $GossipSystem_{\mathsf{N}}$ 的记法，其中并发操作子符合交换律。

$$\begin{aligned} &GossipSystem_{\mathsf{N},(a,b,c)}(x) \\ &= (\overbrace{InfectiousNode(x) \mid \cdots \mid InfectiousNode(x)}^{a} \\ &\overbrace{|DeliveringNode(x) \mid \cdots \mid DeliveringNode(x)}^{b} \\ &\overbrace{|UnInfectiousNode \mid \cdots \mid UnInfectiousNode)}^{c} \\ &\backslash\{trans_{i,j} \mid i \in \mathsf{N} \wedge j \in \mathsf{N} \wedge i \neq j\} \cup \{time, timeout\} \end{aligned}$$

其中，$GossipSystem_{\mathsf{N},(a,b,c)}(x)$ 表示在系统的 $n = |\mathsf{N}|$ 个节点中，有 a 个处于 $InfectiousNode(x)$ 的状态，有 b 个处于 $DeliveringNode(x)$ 的状态，有 c 个处于 $UnInfectiousNode$ 的状态，且满足 $a+b+c=n$。很显然，初始的 $GossipSystem_{\mathsf{N}}$ 中唯一的 $Node$ 状态的节点此时已经从外界接收了某个消息 x。

接下来我们来证明，我们实现的基于 Gossip 协议的 P2P 系统满足多播系统的规约。由于 $GossipSystem_{\mathsf{N}}$ 和 $MulticastSpec_{\mathsf{N}}$ 中的每一个节点能且仅能执行一次 $deliver$ 操作，且 $deliver$ 的顺序不影响功能，所以我们可以规定 $A \overset{\overline{deliver_i}(t_1)}{\longrightarrow}_{\mathsf{T}} B$ 和 $C \overset{\overline{deliver_j}(t_2)}{\longrightarrow}_{\mathsf{T}} B$ 是互模拟的当且仅当 $t_1 = t_2$，对下标是否一致不作要求。因此在后文的证明中忽略了下标。

定理 3.1 $GossipSystem_{\mathsf{N}} =_{\mathbb{RVPC}_{\mathsf{Th}}} MulticastSpec_{\mathsf{N}}$，Th 是可判定的逻辑。

证明 我们可以通过构建等价集，并证明等价集是一个符号互模拟关系来证明 $GossipSystem_{\mathsf{N}} =_{\mathbb{RVPC}_{\mathsf{Th}}} MulticastSpec_{\mathsf{N}}$。

构造等价集

$$\begin{aligned}\mathcal{S} = \{&(GossipSystem_{\mathsf{N}}, MulticastSpec_{\mathsf{N}}),\\ &(GossipSystem_{\mathsf{N},(n,0,0)}(x), Multicasting_{\mathsf{N},\mathsf{N}}(x))\}\\ \cup \{&(GossipSystem_{\mathsf{N},(a,b,c)}(x), Multicasting_{\mathsf{N},\mathsf{KNOWN}}(x)) \mid |\mathsf{KNOWN}| = a \neq n\}\end{aligned} \tag{3–3}$$

首先需要声明，根据等价关系的传递性，我们可以得到 $GossipSystem_{\mathsf{N},(a,b,c)}(x) \in [GossipSystem_{\mathsf{N},(a,b',c')}(x)]_{\mathsf{T}\mathcal{S}}, b+c = b'+c'$。我们来依次证明 $\mathcal{S}$ 中的每一对等价关系为符号互模拟关系:

(1) $GossipSystem_{\mathsf{N},(n,0,0)}(x)$ 和 $Multicasting_{\mathsf{N},\mathsf{N}}(x)$ 的 $\mathsf{T}\mathcal{S}$ 条件等价树分别只有一个根节点，且不能转移至其他等价集，他们的互模拟是显然的，实际上 $GossipSystem_{\mathsf{N},(n,0,0)}(x) = Multicasting_{\mathsf{N},\mathsf{N}}(x) = 0$。

(2) 对于 $(GossipSystem_{\mathsf{N},(a,b,c)}(x), Multicasting_{\mathsf{N},\mathsf{KNOWN}}(x)) \mid |\mathsf{KNOWN}| = a \neq n$:

(a) 若 $a < n-1$：$GossipSystem_{\mathsf{N},(a,b,c)}(x)$ 的 $\mathsf{T}\mathcal{S}$ 条件等价树 t 如图 3–3所示。当 t 上的

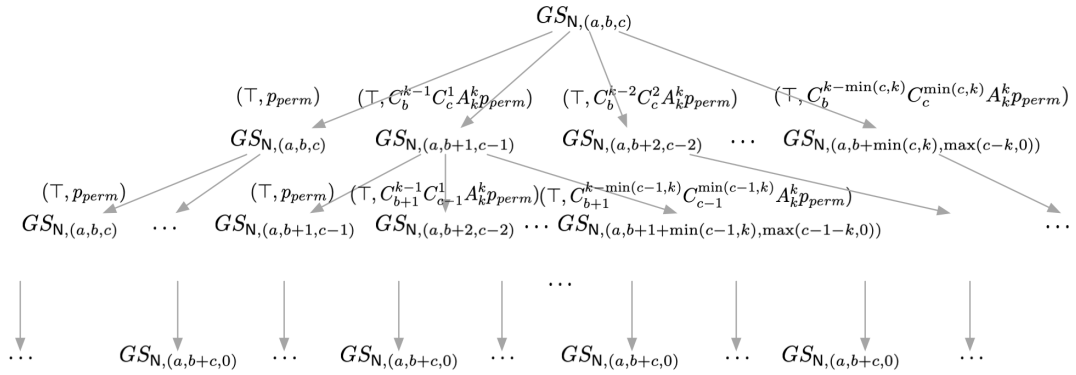

图 3–3 $GossipSystem_{\mathsf{N},(a,b,c)}(x)$ 关于 $\mathsf{T}\mathcal{S}$ 的条件等价树 t

节点进入 $GossipSystem_{\mathsf{N},(a,b+c,0)}(x)$ 状态时，状态保持的静态迁移（τ 动作）就会终止，因此 $GossipSystem_{\mathsf{N},(a,b+c,0)}(x)$ 为 t 的叶子结点。很显然，对 t 上的每一个非叶结点，都会经过 $Node_j \xrightarrow{p_j'}_{\mathsf{T}} GossipSystem_{\mathsf{N},(a,b+c,0)}(x), p_j' = p_{j_1}p_{j_2}\cdots p_{j_m} < 1$。对 t 的叶子结点有 $GossipSystem_{\mathsf{N},(a,b+c,0)}(x) \xrightarrow{\overline{deliver}(x)}_{\mathsf{T}} GossipSystem_{\mathsf{N},(a+1,b+c-1,0)}(x)$。即

$$GossipSystem_{\mathsf{N},(a,b,c)}(x) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\overline{deliver}(x)}_{\mathsf{T}} [GossipSystem_{\mathsf{N},(a+1,b',c')}(x)]_{\mathsf{T}\mathcal{S}}$$

其中 $b'+c' = b+c-1$。我们可以用

$$Multicasting_{\mathsf{N},\mathsf{KNOWN}} \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\overline{deliver}(x)}_{\mathsf{T}} Multicasting_{\mathsf{N},\mathsf{KNOWN}'}, |\mathsf{KNOWN}'| = a+1$$

来模拟上述状态迁移。对称的模拟依然成立。

(b) 若 $a = n-1$:

若 $b = 1$，$GossipSystem_{\mathsf{N},(n-1,1,0)}(x)$ 的 $\mathsf{T}\mathcal{S}$ 条件等价树 t 只有一个根节点，对于 $(GossipSystem_{\mathsf{N},(n-1,1,0)}(x) \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\overline{deliver}(x)}_{\mathsf{T}} [GossipSystem_{\mathsf{N},(n,0,0)}(x)]_{\mathsf{T}\mathcal{S}}$，我们可以用 $Multicasting_{\mathsf{N},\mathsf{KNOWN}} \rightsquigarrow_{\mathsf{T}\mathcal{S}} \xrightarrow{\overline{deliver}(x)}_{\mathsf{T}} Multicasting_{\mathsf{N},\mathsf{N}}$ 来模拟。

若 $b = 0$，$GossipSystem_{\mathsf{N},(n-1,0,1)}(x)$ 的 $\mathsf{T}\mathcal{S}$ 条件等价树 t 如图 3–4所示。

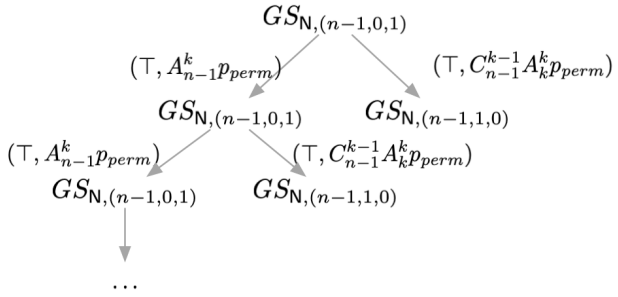

图 3–4 $GossipSystem_{\mathsf{N},(a,b,c)}(x)$ **关于** $\top\mathcal{S}$ **的条件等价树** t

$GossipSystem_{\mathsf{N},(n-1,1,0)}$ 为 t 的叶节点，且有 $(GossipSystem_{\mathsf{N},(n-1,0,1)}(x) \overset{\overline{deliver}(x)}{\twoheadrightarrow_{\top\mathcal{S}}} \top$ $[GossipSystem_{\mathsf{N},(n,0,0)}(x)]_{\top\mathcal{S}}$。我们可以用 $Multicasting_{\mathsf{N},\mathsf{KNOWN}} \overset{\overline{deliver}(x)}{\twoheadrightarrow_{\top\mathcal{S}}} \top$ $Multicasting_{\mathsf{N},\mathsf{N}}$ 来模拟。

(3) 对 $(GossipSystem_{\mathsf{N}}, MulticastSpec_{\mathsf{N}})$，我们可以用

$$GossipSystem_{\mathsf{N}} \overset{accept(x).\overline{deliver}(x)}{\longrightarrow}_{\top} GossipSystem_{\mathsf{N},(1,0,0)}(x)$$

$$MulticastSpec_{\mathsf{N}} \overset{accept(x).\overline{deliver}(x)}{\longrightarrow}_{\top} Multicasting_{\mathsf{N},\{1\}}$$

来相互模拟。 □

3.2.3 基于 Gossip-Style Membership 协议的通信系统的实现

Van RENESSE R 提出了一种基于 Gossip 协议的错误检测机制：在集群中的每一个节点会维护一个列表 Membership List，列表中包含已知的其他节点的地址和整数表示的心跳。在每一个 Gossip 的周期，节点会增加自己的心跳，并且随机的向另一个节点发送自己的 Membership List；若节点收到了其他节点发送过来的 Membership List，它会将两个列表合并，保留对应地址心跳最大的项。若节点的 Membership List 的一项经过 T_{fail} 时间没有更新，则认为这一项对应的节点失效，我们称为 Gossip-Style Membership 协议[35]。

在本次的实现中，我们规定在每一个 Gossip 周期中，节点可以随机选择 k 个节点发送自己的 Membership List。

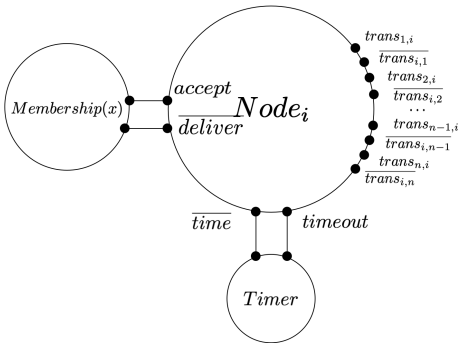

图 3–5 Gossip-Style Membership 节点示意图

由于 Gossip-Style Membership 无需与外界的输入输出，并且需要提供储存、更新 Membership List 的函数，我们需要对之前的节点结构进行调整。同时，系统内的节点也会有失效

的可能，我们可以用 p_{fail} 来定义一个节点失效的概率，同时经过一个静态迁移 τ，这个节点就会被修复。另外，由于在系统中的所有节点都是消息源，且 Membership List 在系统中不停的更新，因此也没有了感染者与被感染者的角色区分，也需要对名称进行了修改。在 Gossip System 节点的基础上，将原本对外暴露的 $accept, deliver$ 通道用于连接 $Membership$，作为链接网络的节点获取和更新本地 Membership List 的通道，修改后的节点如图 3–5 所示。

Gossip-Style Membership 协议中的 P2P 系统定义如下，

$$\begin{aligned}
FragileNode_i &\overset{def}{=} p_{fail}\tau.BadNode_i \oplus (1-p_{fail})\tau.Node_i \\
BadNode_i &\overset{def}{=} \tau.FragileNode_i \\
Node_i &\overset{def}{=} timeout.accept(x).(\bigoplus_{perm\in \mathrm{PERM}_i} p_{perm}\tau.GossipingNode_{i,perm}(x)) \\
&\quad + \sum_{j\in \mathsf{N}/\{i\}} trans_{j,i}(x).\overline{deliver}(x).FragileNode_i \\
GossipingNode_{i,perm}(x) &\overset{def}{=} \overline{trans_{i,perm_1}}(x).\cdots\overline{trans_{i,perm_k}}(x).\overline{time}.FragileNode_i \\
GossipSystem_{\mathsf{N}} &\overset{def}{=} (FragileNode_1 \mid FragileNode_2 \mid \cdots \mid FragileNode_n) \\
&\quad \backslash\{trans_{i,j} \mid i\in \mathsf{N}\wedge j\in \mathsf{N}\wedge i\neq j\}\cup\{time, timeout, accept, deliver\}
\end{aligned} \tag{3–4}$$

其中，每一个节点定义为 $FragileNode$，每一时刻它会有 p_{fail} 的概率成为失效节点 $BadNode$，和 $1-p_{fail}$ 的概率成为可发送列表和接受列表的正常节点 $Node$。在 $Node$ 状态的节点，可以在外界计时器发送 $timeout$ 信号时通过 $accept$ 通道获取自身的 Membership List，以概率 p_{perm} 的概率选定 $\mathsf{N}/\{i\}$ 中任选 k 个元素的全排列中的一个排列 $perm$ 作为发送目标，进而迁移为 $GossipingNode_{i,perm}(x)$，向 $perm$ 中的节点发送 Membership List，发送完成后回到 $FragileNode$ 的状态。一个有 $n=|\mathsf{N}|$ 个节点的基于 Gossip-Style Membership 协议的 P2P 系统定义为 $GossipSystem_{\mathsf{N}}$，它由 n 个 $FragileNode$ 状态的并发的节点构成。

此外，我们还需要定义每个节点本地的 Membership 系统，来处理 Membership List 的获取和更新。Membership List 的内容一般为表 3–2中的内容。其中 Address 表示系统中其他节

表 3–2 Membership List 样表

Address	HeartBeat	LocalTime
1	10120	66
2	10103	62
3	10098	63

点的地址，HeartBeat 表示对应节点的心跳计数，LocalTime 表示 Membership List 中这一项记录最近一次的更新时间，当当前时间与某一项的 LocalTime 相差超过 T_{fail}，我们认为这一节点失效。

因此 Membership 系统需要有一个本地计时器 Timer、一个心跳计数器 Counter，一个本地 $MembershipList(X)$，X 应为一个 (Address, HeartBeat, LocalTime) 的三元组的数组，一个记录本地地址的 $AddrInfo(address)$（地址应在加入网络时由 DNS 分配，此处不考虑它的分配过程）。定义 $x_i[Address]$ 为取 Address 的值的操作子，$x_i[HeartBeat]$ 同理。n 为

MembershipList 的最大容量。

我们定义的 Membership 系统的实现如图 3–6所示，可以看到，我们使用 n 个

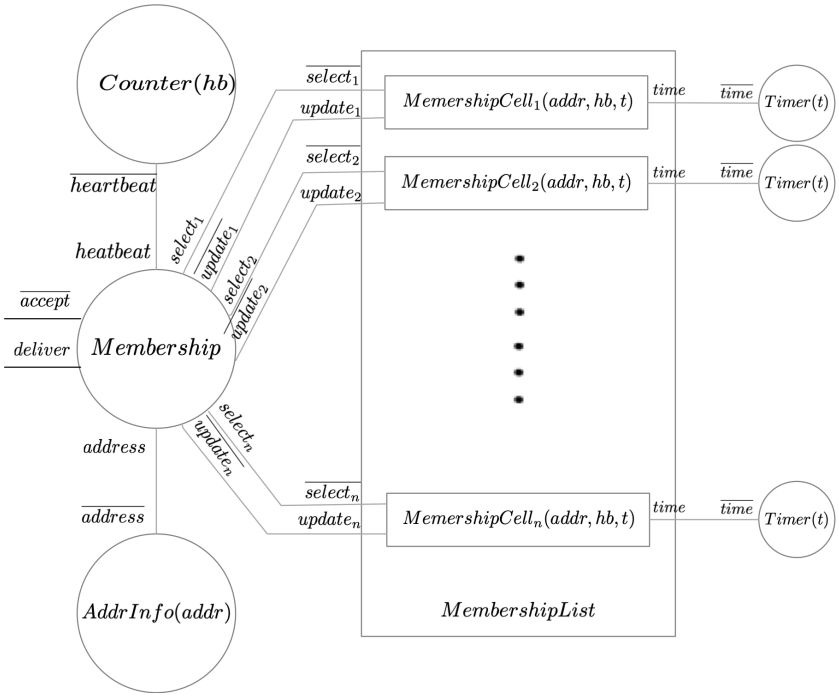

图 3–6 Membership 示意图

$MembershipCell$ 来存储 Membership List 中的每一项，$Membership$ 可以通过 $select_i$ 通道获取 $MembershipCell_i$ 中的数据，也可以通过 $update_i$ 通道更新 $MembershipCell_i$ 中的数据，每一个 $MembershipCell$ 连接一个 Timer，用来获取本地时间，设置 LocalTime 字段，实际上，对于一个节点它可能只有一个 Timer，这里的连接也是指逻辑连接。$MembershipCell$ 的定义如下：

$$\begin{aligned} &MembershipCell_i(addr, hb, t) \stackrel{def}{=} \\ &\overline{select_i}(\{Address : addr, HeartBeat : hb\}).MembershipCell_i(addr, hb, t) \\ &+ update_i(\{Address : addr', HeartBeat : hb'\}).time(t').MembershipCell_i(addr', hb', t') \end{aligned} \tag{3–5}$$

n 个并发的 $MembershipCell$ 组成的 $MembershipList$ 定义如下，在初始状态时，每一个 $MembershipCell$ 中的数据为空 (ϵ)：

$$MembershipList \stackrel{def}{=} (MembershipCell_1(\epsilon) \mid \cdots MembershipCell_n(\epsilon)) \tag{3–6}$$

此外，我们定义了一个 $Counter$ 维护和更新节点的心跳计数，在每一次 Gossip 周期 $hb = hb + 1$，$Membership$ 可以通过 $heartbeat$ 通道获取心跳。

$$Counter(hb) \stackrel{def}{=} \overline{heartbeat}(hb).Counter(\mathsf{s}(hb)) \tag{3–7}$$

对于节点本地地址的维护，我们定义了一个 $AddrInfo$ 来储存本地地址，$Membership$ 可以通过 $address$ 通道获取本地地址。

$$AddrInfo(addr) \stackrel{def}{=} \overline{address}(addr) \tag{3–8}$$

定义完以上的辅助工具后，我们可以定义 $Membership$ 的逻辑了！

$$
\begin{aligned}
&Membership \overset{def}{=} address(addr).heartbeat(hb).select_1(x_1)\cdots select_n(x_n).\\
&\quad \overline{accept}(\{Address : addr, HeartBeat : hb\}, x_1, \cdots, x_n).Membership\\
&\quad + deliver(x_1, x_2, \cdots, x_n).Processing(x_1, x_2, \cdots, x_n)\\
&Processing(x_1, x_2, \cdots, x_n) \overset{def}{=} (Check(x_1) \mid \cdots \mid Check(x_n) \mid Membership)\\
&Check(x) \overset{def}{=} address(addr).((addr = x[Address])0 | \neg(addr = x[Address])FindAndUpdate_1(x))\\
&FindAndUpdate_i(x) \overset{def}{=} select_i(x_i).((x_i = \epsilon)(\overline{update_i}(x).0)\\
&\quad | \neg(x_i = \epsilon)((x_i[Address] = x[Address])((x_i[HeartBeat] < x[HeartBeat])(\overline{update_i}(x).0)\\
&\quad | \neg(x_i[HeartBeat] < x[HeartBeat])0)\\
&\quad | \neg(x_i[Address] = x[Address])FindAndUpdate_{i+1}(x))), i \leq n\\
&FindAndUpdate_i(x) \overset{def}{=} 0, i > n
\end{aligned}
\tag{3–9}
$$

其中 $\mathsf{s}(x)$ 表示 $\mathsf{s}(x) = x + 1$，$\mathsf{s}(x)$ 的实现可以通过 FU Y 在 The Value-Passing Calculus 中定义的 Numeric System 实现[13]。

$Membership$ 可以获取 $MembershipList$ 中所有 Cell 中储存的信息，添加本地地址和心跳后，将这一组信息经 $accept$ 通道发出，根据 $Node$ 的定义，我们知道 $Node$ 从 $accept$ 通道接受这些信息后会随机的发送给 k 个其他节点。另外，$Membership$ 从 $deliver$ 通道接收了其他节点发送的 Membership List 后，将迁移到状态 $Processing(x_1, x_2, \cdots, x_n)$。$Processing(x_1, x_2, \cdots, x_n)$ 可以并发的运行 $Check(x_i)$，在 $Check(x_i)$ 状态下，若 x_i 的地址为当前节点的地址，不做处理，若不为当前节点的地址，则从第一个 $MembershipCell$ 开始对比 x_i 的地址与 $MembershipCell$ 的地址，若地址相同且 x_i 中的心跳计数大于 $MembershipCell$ 中的心跳计数，更新 $MembershipCell$ 的数据；若所有有值的 $MembershipCell$ 均无 x_i 的地址，用 x_i 直接更新第一个空的 Cell。

3.3 基于 Gossip-Style Membership 协议的通信系统的仿真模拟

在本节我们会根据本章的模型来实现一个基于 Gossip-Style Membership 协议的 P2P 系统，由于资源限制，不能实际部署在多个主机构成的集群，我们将使用多进程来模拟多个主机，实际上进程理论本就是用来刻画进程的并发，多个主机上的程序的本质也是进程。

3.3.1 Go 语言与 CSP

我们将使用 Go 语言实现这个基于 Gossip-Style Membership 协议的通信系统，Go 语言实现了两种并发模型，一种是 C++，Java 使用的多线程，一种是 CSP[4] 并发模型。如我们在第 一章中提到的，CSP 也是一种经典的进程演算，它与 CCS 的区别在于等价的类型和建模并发系统采用的方法，有关 CCS 和 CSP 对比的研究也有很多[12, 37-39]。Go 语言使用了 CSP 理论中的 Process/Channel，对应到语言特性中的 goroutine/channel[40]。Goroutine 是一种轻量级线程，channel 用于协程间的通信。我们使用 Go 语言的并发特性可以更加直观的展示出对前述模型的代码实现。

3.3.2 代码实现与仿真效果

由于代码的解释比较枯燥，本小节只展示关键部分的代码实现，完整的代码实现及下载、运行方式可以参考附录 A。在编写代码的过程中，考虑到代码的可读性和字符的限制，我们对前述模型通道和进程状态的名称有所修改，但本质没有变化。

在 Go 语言中，我们可以用 channel 特性来实现 $\mathbb{RVPC}_{\mathsf{Th}}$ 中的通道。如在实现 Gossip-Style Membership 的节点结构时，Node 结构体中，chan 类型的字段分别代表了图 3–5中的相应名字的通道。其中，我们在 3.2.1小节中提到 $trans_{i,j}$ 通道用于节点 i 向节点 j 传输信息，这一通道实质上表达的是信息传输的逻辑路径，我们在实现时考虑信息传输的代码层面的物理路径，只需要每一个节点设置一个 *trans* 通道用于接收集群中其他节点的信息即可，向其他节点发送信息时，我们可以通过 Others 字段向 Others[i].trans 通道发送消息。对于 Node 节点的状态迁移，我们可以使用 bool 类型的字段 isbad 表示节点状态是否为 BadNode。

```
type Node struct {
   isbad bool
   trans chan []Message
   time chan Nil
   timeout chan Nil
   accept chan []Message
   deliver chan []Message
   Others []*Node
}
```

我们可以使用基于 select 的多路复用实现非确定性选择，如我们在 3–4中实现的系统中的 $Node_i$，这里 $Node_i$ 可以做非确定的选择：接收到 timeout 信号后从 accept 通道获取本地的 Membership List 随机选择 k 个其他节点发送；或从其他节点接收到 Membership List 消息，通过 deliver 通道向本地的 Membership 进程传递该消息。

```
select {
   //timeout.accept(x).bigoplus p tau.GossipingNode
   case <- node.timeout:
      messages := <- node.accept
      //随机生成 k 个其他节点的排列 targets
      node.Gossiping(messages, targets)
   // sum trans(x).deliver(x).FragileNode
   case message := <- node.trans:
      node.deliver <- message
}
```

select 的语法与 switch 比较相似，每一个 case 代表一个通信操作，select 会等待 case 中有能够执行的 case 时执行该 case：当条件满足时，select 才会通信并执行 case 后的语句块，其他通信不执行。

在 Go 语言中，每一个并发执行的单元称为一个 goroutine，我们同样使用 goroutine 来实现 $\mathbb{RVPC}_{\mathsf{Th}}$ 中的并发。如对 $GossipSystem_{\mathsf{N}} \overset{def}{=} (FragileNode_1 \mid FragileNode_2 \mid \cdots \mid FragileNode_n)$，的实现：

```
for i:=0;i<NODE_NUM;i++ {
     go nodes[i].Fragile()
 }
```

实际上在我们的系统中，对一个节点本地的 Timer，Membership 和接入通信网络的 Node 同样应该是并发的。

本节仿真实现的基于 Gossip-Style Membership 协议的 P2P 系统参数如表 3–3。

表 3–3 基于 Gossip-Style Membership 协议的 P2P 系统参数表

参数名称	参数值	参数含义
n	5	系统节点数量
k	2	每 Gossip 周期发送的节点数
T_{Gossip}	1s	Gossip 周期
T_{repair}	4s	失效节点恢复时间
T_{fail}	4s	节点有效期超时时间
p_{fail}	0.1	节点失效概率

我们使用上述参数运行我们实现的系统，我们使用 html 为我们的系统做了一个前端界面，我们可以在这个界面观察每个节点 Membership List 的更新过程。

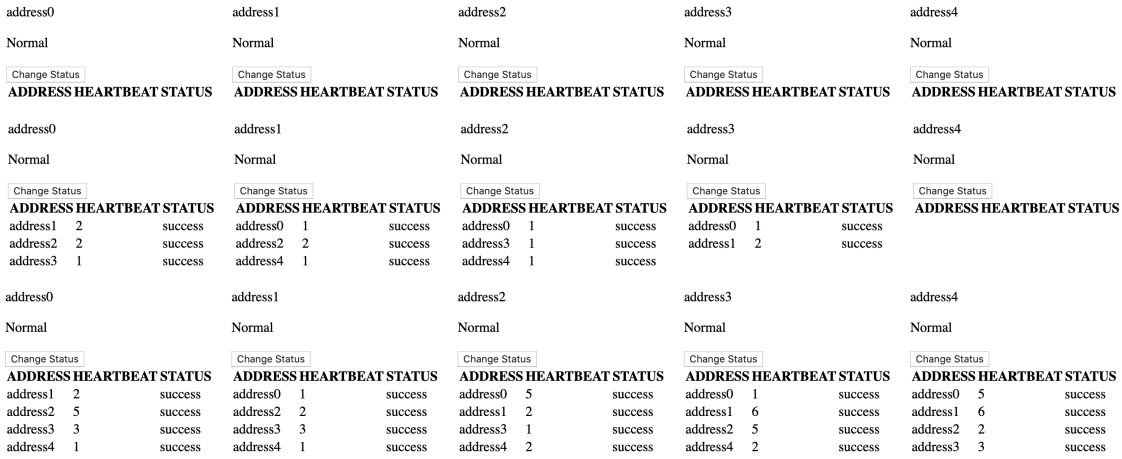

图 3–7 基于 Gossip-Style Membership 协议的 5 节点 P2P 系统的 Membership List 扩展过程

图 3–7展示了每个节点的 Membership List 的扩展过程，其中第一行为节点的地址，我们使用字符串表示，也可以根据需要修改成其他的形式，第二行为对应节点的状态，Bad 表示节点失效，对应 $BadNode$ 状态，Normal 表示节点正常，对应 $FragileNode$ 和 $Node$ 状态，第三行的按钮可以改变节点的状态，接下来的是对应节点的 Membership List，可以看到我们展示了 Address，Heartbeat, Status 字段，其中 Status 字段是由 LocalTime 计算而得，当当前时间与 Membership List 项的 LocalTime 字段相差超过 T_{fail} 时，我们认为这一项对应的节点处于失效状态。在 3–7的第一个图中，我们可以看到初始状态所有节点的 Membership List 为空，第二张图中 Membership List 进行了扩张，第三张图中全部节点的 Membership List 包含了所有其他节点。

我们同样可以展示系统的失效检测。在图 3–8中，第一张图是正常状态，我们在第二张图中手动改变了地址为 address1 节点状态，使该节点失效，可以看到第三张图中 address0,address3 检测到了 address1 的失效，第四张图所有节点都检测到了 address1 的失效。

我们也可以通过设置 p_{fail} 使系统中的节点以一定概率自动失效和恢复，动态效果可以参考附录 A。

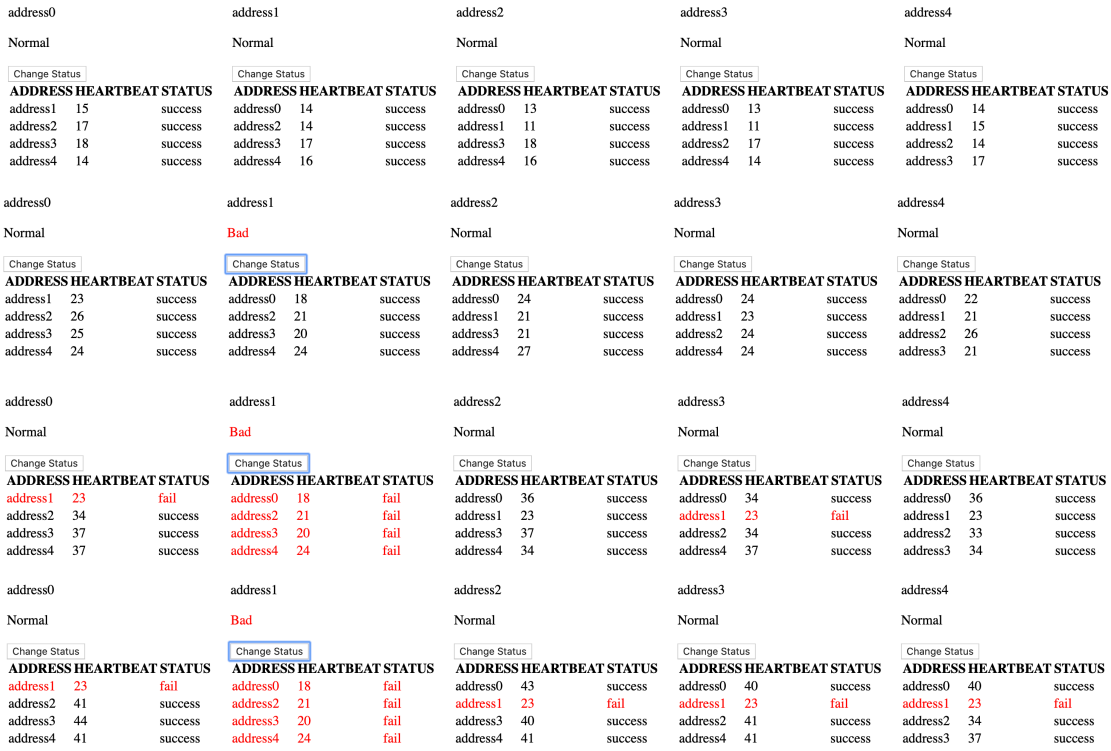

图 3–8 基于 Gossip-Style Membership 协议的 5 节点 P2P 系统的错误检测

3.4 本章小结

本章我们以一种基于云计算失效检测协议 Gossip-Style Membership 协议的通信系统的 $\mathbb{RVPC}_{\mathsf{Th}}$ 实现作为本文中提出的随机传值进程模型的应用案例，提供了使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 建模及实现通信协议以至其他现实问题的思路。

我们首先介绍了 Gossip 协议和 Membership 协议，并使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 实现了基于 Gossip 协议的 P2P 系统，使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟证明了 Gossip 协议与多播规约的观察等价性。在基于 Gossip 协议的 P2P 系统的基础上，我们修改实现使其成为基于 Gossip-Sytle Membership 协议的 P2P 系统，并给出了 GO 语言的代码实现和仿真模拟。仿真结果证明了我们的随机传值进程模型在对并发通信过程的建模和分析中有一定的可行性。

第四章　总结与展望

随着随机和交互在现代计算机科学中的应用日渐广泛，概率进程模型的研究也备受关注。而作为适用范围颇为广泛的一种经典进程模型，传值进程模型的概率扩展也有很多研究。在已有工作中，对传值进程模型的扩展往往局限于某一个特定的应用场景，或因为作为基础的传值进程模型的局限性，其模型完整性尚有欠缺，普遍意义上的概率传值进程模型的研究尚显不足。本文致力于在已有的通用概率扩展方法和传值进程模型的基础上探索具有普适性的传值进程模型的概率扩展。

本文在传值进程模型 $\mathbb{VPC}_{\mathsf{Th}}$ 的基础上，使用一种模型无关的概率扩展方法，扩展了 $\mathbb{VPC}_{\mathsf{Th}}$ 的语法和迁移语义，增加了随机选择操作子，使其除非确定性选择之外还可以做随机选择，我们称这一模型为随机传值进程模型，记为 $\mathbb{RVPC}_{\mathsf{Th}}$。

同时，为了给出 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性的定义，我们使用 Uniform Approach 的方法定义了条件等价集、条件等价树（森林）、条件 l 转移和条件 q 转移，进而定义了 $\mathbb{RVPC}_{\mathsf{Th}}$ 下的符号互模拟。在 Uniform Apprach 中，条件等价树的本质是概率化了 $\mathbb{VPC}_{\mathsf{Th}}$ 中的状态保持的静态迁移 $A \Rightarrow_{\varphi} A' = A$，$\mathbb{RVPC}_{\mathsf{Th}}$ 中的符号互模拟的本质是用条件等价树中条件 φ 对应的划分 $\{\varphi_i\}_{i\in I}$ 所对应的条件等价森林模拟条件等价树中的条件 l 转移和条件 q 转移。我们将共发散的符号互模拟关系的全集定义为 $\mathbb{RVPC}_{\mathsf{Th}}$ 的观察等价性，并给出了观察等价性的同余性证明。

最后，我们以基于云计算协议 Gossip-Style Membership 协议的一个通信系统为例，给出了使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 建模通信过程，乃至其他现实问题的方法和过程，并使用 $\mathbb{RVPC}_{\mathsf{Th}}$ 的符号互模拟给出了 Gossip 协议与我们定义的多播规约的观察等价性。给出基于 Gossip-Style Membership 协议的通信系统模型的同时，我们使用 Go 语言实现了这个模型，并分析了 Go 语言特性对应部分 $\mathbb{RVPC}_{\mathsf{Th}}$ 操作子的实现。事实证明 $\mathbb{RVPC}_{\mathsf{Th}}$ 在建模和分析通信过程时具有一定的可行性。我们希望 $\mathbb{RVPC}_{\mathsf{Th}}$ 可以为更多含有传值、计算性质的现实问题的建模和分析提供有效、可行的方法。

我们认为下一步的研究工作主要可以从以下几个方面入手：

第一，随机传值进程模型中的随机性可以体现在两个方面：内容随机性和通道随机性，内容随机性体现在值的随机性，外界向进程传的值可能会在某一个值域中符合某种概率分布，或近似符合某种概率分布；而通道随机性体现在进程对通信通道的选择是随机的，如在第 三章中，Gossip 协议会周期性的随机选择通信的对象。由于 Uniform Approach 中只关注了通道的随机性，在本文对随机传值进程模型的定义中，我们也只关注了通道的随机性。这样做有一定的合理性：对于一个与具体问题无关的形式化方法，我们无法给定值的一个具体的随机分布。对于具体的问题，我们可以在 $\mathbb{RVPC}_{\mathsf{Th}}$ 的基础上考虑值的随机分布。

但遗憾的是，在第 三章中我们给出的应用，包括在第 二章中的简单的例子，实际上都是 $\mathbb{RVPC}_{\mathsf{Th}}$ 上的进程，由于系统没有与外界的值的通信，整个系统中没有自由变元，因此依旧是不需要考虑内容随机性的模型。这是本文在构思上一个欠考虑的地方，实际上建模一个与外界有值传递的系统对于本文的意义会更大一些。进一步的研究可以关注内容随机性。对于内容的随机性的实现，MILNER R 在 Communication and Concurrency 中提供了一个思路：为了建模传值的 CCS，MILNER R 将 Value-passing CCS 转化为 CCS 时将 $S = a(x).T$ 转为

$S = \sum_{v \in V} a_v.T\{v/x\}$，其中 V 是给定值的集合[3]。我们也可以用 $S = \bigoplus_{v \in V} p_v.\tau.a(v).T\{v/x\}$ 的方法来建模值的随机性。

第二，我们希望本文提出的随机传值进程模型也是一种通用的模型，不只局限于本文第 三章中的应用场景，甚至不局限于通信过程的建模，而是能够适用于很多具有传值特点的并发过程。我们希望应用场景甚至可以是跨学科的，包括生物过程、生产流程等等。进一步的工作也可以是在这些领域中使用随机传值进程模型解决一些问题。

参考文献

[1] 程序理论[M/OL]. 见: 计算机科学技术百科全书. 清华大学出版社, 2005: 65-67. https://books.google.com.hk/books?id=n79Zh2JzBhYC.

[2] BAETEN J. A brief history of process algebra[J/OL]. Theoretical Computer Science, 2005, 335(2): 131-146. http://www.sciencedirect.com/science/article/pii/S0304397505000307. DOI: https://doi.org/10.1016/j.tcs.2004.07.036.

[3] MILNER R. Communication and Concurrency[M]. USA: Prentice-Hall, Inc., 1989.

[4] HOARE C A R. Communicating Sequential Processes[J/OL]. Commun. ACM, 1978, 21(8): 666-677. https://doi.org/10.1145/359576.359585. DOI: 10.1145/359576.359585.

[5] BERGSTRA J, KLOP J. Algebra of communicating processes with abstraction[J/OL]. Theoretical Computer Science, 1985, 37: 77-121. http://www.sciencedirect.com/science/article/pii/030439758590088X. DOI: https://doi.org/10.1016/0304-3975(85)90088-X.

[6] BOLOGNESI T, BRINKSMA E. Introduction to the ISO specification language LOTOS[J/OL]. Computer Networks and ISDN Systems, 1987, 14(1): 25-59. http://www.sciencedirect.com/science/article/pii/0169755287900857. DOI: https://doi.org/10.1016/0169-7552(87)90085-7.

[7] GIACALONE A, JOU C C, SMOLKA S A. Algebraic Reasoning for Probabilistic Concurrent Systems[C]. in: Proc. IFIP TC2 Working Conference on Programming Concepts and Methods. North-Holland, 1990: 443-458.

[8] HANSSON H, JONSSON B. A calculus for communicating systems with time and probabilities[C]. in: [1990] Proceedings 11th Real-Time Systems Symposium. 1990: 278-287.

[9] LOWE G. Probabilities and Priorities in Timed CSP[D]. 1993.

[10] ANDOVA S. Process Algebra with Probabilistic Choice[C]. in: KATOEN J P. Formal Methods for Real-Time and Probabilistic Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999: 111-129.

[11] FU Y. A Uniform Approach to Random Process Model[J/OL]. CoRR, 2019, abs/1906.09541. arXiv: 1906.09541. http://arxiv.org/abs/1906.09541.

[12] FIDGE C. A Comparative Introduction to CSP, CCS and LOTOS[R/OL]. Software Verification Research Centre, Department Of Computer Science, University Of Queensland. 1994. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.6397&rep=rep1&type=pdf.

[13] FU Y. The Value-Passing Calculus[M/OL]. in: LIU Z, WOODCOCK J, ZHU H. Theories of Programming and Formal Methods: Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013: 166-195. https://doi.org/10.1007/978-3-642-39698-4_11. DOI: 10.1007/978-3-642-39698-4_11.

[14] HUANG H, YANG F. An interpretation of Erlang into value-passing Calculus[J]. Journal of Networks, 2013, 8(7): 1504-1513.

[15] WINSKEL G. A presheaf semantics of value-passing processes[C]. in: MONTANARI U, SASSONE V. CONCUR '96: Concurrency Theory. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996: 98-114.

[16] INGÃ³LFSDÃ³TTIR A, LIN H. CHAPTER 7 - A Symbolic Approach to Value-Passing Processes[G/OL]. in: BERGSTRA J, PONSE A, SMOLKA S. Handbook of Process Algebra. Amsterdam: Elsevier Science, 2001: 427-478. http://www.sciencedirect.com/science/article/pii/B9780444828309500254. DOI: https://doi.org/10.1016/B978-044482830-9/50025-4.

[17] 邓维佳. 一种扩展的并发传值进程抽象模型[D]. 中国科学院研究生院（软件研究所）, 2005.

[18] PRESBURGER M. Uber die vollstandigkeiteines gewissen systems der arithmetik ganzer zahlen[C]. in: Welchen die addition als einzige operation hervortritt. Comptes-Rendus du ler Congres des Mathematiciens des Pays Slavs, 1929.

[19] DAS A, GUPTA I, MOTIVALA A. SWIM: scalable weakly-consistent infection-style process group membership protocol[C]. in: Proceedings International Conference on Dependable Systems and Networks. 2002: 303-312.

[20] 施若愚. 生物自组装的概率进程演算模型[D]. 上海交通大学, 2007.

[21] ZHANG Q, JIANG Y, DING L. Modelling and Analysis of Network Security - aÂ Probabilistic Value-passing CCS Approach[C]. in: QING S, OKAMOTO E, KIM K, et al. Information and Communications Security. Cham: Springer International Publishing, 2016: 295-302.

[22] VARACCA D, VÖLZER H, WINSKEL G. Probabilistic Event Structures and Domains[C]. in: GARDNER P, YOSHIDA N. CONCUR 2004 - Concurrency Theory. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 481-496.

[23] 陈建明, 曾明, 刘国荣. 离散的数学结构[M]. 西安交通大学出版社, 2004.

[24] Van GLABBEEK R J, WEIJLAND W P. Branching Time and Abstraction in Bisimulation Semantics[J/OL]. J. ACM, 1996, 43(3): 555-600. https://doi.org/10.1145/233551.233556. DOI: 10.1145/233551.233556.

[25] Van GLABBEEK R J, WEIJLAND W P. Branching Time and Abstraction in Bisimulation Semantics[J/OL]. J. ACM, 1996, 43(3): 555-600. https://doi.org/10.1145/233551.233556. DOI: 10.1145/233551.233556.

[26] BAIER C, HERMANNS H. Weak bisimulation for fully probabilistic processes[C]. in: GRUMBERG O. Computer Aided Verification. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997: 119-130.

[27] PHILIPPOU A, LEE I, SOKOLSKY O. Weak Bisimulation for Probabilistic Systems[C]. in: PALAMIDESSI C. CONCUR 2000 — Concurrency Theory. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000: 334-349.

[28] ANDOVA S, WILLEMSE T A. Branching bisimulation for probabilistic systems: Characteristics and decidability[J/OL]. Theoretical Computer Science, 2006, 356(3): 325-355. http://www.sciencedirect.com/science/article/pii/S0304397506001459. DOI: https://doi.org/10.1016/j.tcs.2006.02.010.

[29] HENNESSY M, LIN H. Symbolic bisimulations[J/OL]. Theoretical Computer Science, 1995, 138(2): 353-389. http://www.sciencedirect.com/science/article/pii/030439759400172F. DOI: https://doi.org/10.1016/0304-3975(94)00172-F.

[30] 曹晓东. 离散数学及算法第二版[M]. 机械工业出版社, 2013.

[31] DEMERS A, et al. Epidemic Algorithms for Replicated Database Maintenance[J]. Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, 1987, 87: 1-12. DOI: 10.1145/41840.41841.

[32] FANTI G, VISWANATH P. Anonymity Properties of the Bitcoin P2P Network[Z]. 2017. arXiv: 1703.08761 [cs.CR].

[33] GUREVICH Y, MANI R. Group Membership Protocol: Specification and Verification[M]. in: Specification and Validation Methods. USA: Oxford University Press, Inc., 1995: 295-328.

[34] KAWAZOE AGUILERA M, CHEN W, TOUEG S. Heartbeat: A timeout-free failure detector for quiescent reliable communication[C]. in: MAVRONICOLAS M, TSIGAS P. Distributed Algorithms. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997: 126-140.

[35] Van RENESSE R, MINSKY Y, HAYDEN M. A Gossip-Style Failure Detection Service[C]. in: DAVIES N, JOCHEN S, RAYMOND K. Middleware'98. London: Springer London, 1998: 55-70.

[36] LIENIG J, BRUEMMER H. Reliability Analysis[M/OL]. in: Fundamentals of Electronic Systems Design. Cham: Springer International Publishing, 2017: 45-73. https://doi.org/10.1007/978-3-319-55840-0_4. DOI: 10.1007/978-3-319-55840-0_4.

[37] HATZEL M, WAGNER C, PETERS K, et al. Encoding CSP into CCS (Extended Version)[Z]. 2015. arXiv: 1508.01127 [cs.LO].

[38] Van GLABBEEK R. Musings on Encodings and Expressiveness[J/OL]. Electronic Proceedings in Theoretical Computer Science, 2012, 89: 81-98. http://dx.doi.org/10.4204/EPTCS.89.7. DOI: 10.4204/eptcs.89.7.

[39] BROOKES S D. On the relationship of CCS and CSP[C]. in: DIAZ J. Automata, Languages and Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983: 83-96.

[40] PRABHAKAR R, KUMAR R. Concurrent programming with Go[R/OL]. Google, Tech. 2011. http://www.golang.org.

附录 A　基于 Gossip-Style Membership 协议的通信系统的 Go 语言实现

附录 A中仅展示与 $\mathbb{RVPC}_{\mathbb{Th}}$ 有关的部分，若要运行完整代码 (包含前端展示界面) 和更多运行结果，可以移步Github。

Node 的 Go 语言实现。

```go
package simple

import (
  "fmt"
  "math/rand"
  "time"
)

const (
  P_FAIL=0.1
  GOSSIP_INTERVAL=time.Second //1sGossip 一次
  REPAIR_TIME=4*time.Second //BadNode 恢复时间
  NODE_NUM = 5 //节点数目
  BUFSIZE = 4 //channel buffer size 一般设置为数据中心节点的数目即可
  K = 2
  IS_AUTO = false //展示用
)

type Nil struct {}

type Node struct {
  isbad bool
  trans chan []Message
  time chan Nil
  timeout chan Nil
  accept chan []Message
  deliver chan []Message
  Others []*Node
  //仅打日志及图形化显示使用
  address string
  membership *Membership
}

func NewNode(address string) (instance *Node) {
  instance = new(Node)
  instance.accept = make(chan []Message)
  instance.deliver = make(chan []Message)
  instance.trans = make(chan []Message, BUFSIZE)
  instance.time = make(chan Nil)
  instance.timeout = make(chan Nil)
  instance.isbad = false
  instance.address = address
  instance.Others = make([]*Node, 0)
  instance.membership = NewMembership(address, instance.accept, instance.deliver)
  fmt.Printf("Initialized Node %p\n", instance)
  return
```

```
}

//多路复用实现 Nondeterminated Choice
func (node *Node) Fragile() {
  fmt.Printf("Node %p Starts\n", node)
  go node.timer()
  go node.membership.Running()
  node.time <- Nil{}
  for {
    if (IS_AUTO) {
      node.Bad()
    }
    if (node.isbad) {
      time.Sleep(REPAIR_TIME)
      continue
    }
    select {
    case message := <- node.trans:
      node.deliver <- message
      for len(node.trans) > 0 {
        message = <- node.trans
        node.deliver <- message
      }
    case <- node.timeout:
      messages := <- node.accept
      rand.Seed(time.Now().UnixNano())
      perm := rand.Perm(len(node.Others))[:K]
      var targets []*Node
      for _, p := range perm {
        targets = append(targets, node.Others[p])
      }
      node.Gossiping(messages, targets)
    }
  }
}
func (node *Node) Bad() {
  rand.Seed(time.Now().UnixNano())
    r := rand.Float32()
    if r < P_FAIL {
      node.isbad = true
    } else {
      node.isbad = false
    }
}

func (node *Node) timer() {
  for {
    select {
    case <- node.time:
      time.Sleep(GOSSIP_INTERVAL)
      node.timeout <- Nil{}
    }
  }
}

func (node *Node) Gossiping(messages []Message, targets []*Node) {
  var str string
  for _, t := range targets {
```

```
    if (len(t.trans) == BUFSIZE) {
      continue
    }
    t.trans <- messages
    str+=(t.address+" ")
  }
  fmt.Printf("[SEND] From: %s; To: %s\n", node.address, str)
  node.time <- Nil{}
}

func transmitting(messages []Message, targets []*Node) string {
  var str string
  for _, t := range targets {
    if (len(t.trans) == BUFSIZE) {
      continue
    }
    t.trans <- messages
    str+=(t.address+" ")
  }
  return str
}

func (node *Node)ChangeStatus() {
  node.isbad = !node.isbad
}

func (node *Node)Address() string {
  return node.address
}
```

main 的 Go 语言实现，主要包含了多个节点并发执行的逻辑。

```
package main

import (
  "./simple"
  "strconv"
)

var nodes []*simple.Node

func main() {
  nodes = make([]*simple.Node, 0)
  for i:=0;i<simple.NODE_NUM;i++ {
    nodes = append(nodes, simple.NewNode("address"+strconv.Itoa(i)))
  }

  for i:=0;i<simple.NODE_NUM;i++ {
    nodes[i].Others = append(nodes[i].Others, nodes[:i]...)
    nodes[i].Others = append(nodes[i].Others, nodes[i+1:]...)
    go nodes[i].Fragile()
  }

  for {
  }
}
```

Membership 的 Go 语言实现。由于 Node，main 的 Go 语言实现已经可以很好的起到 Demo 的作用，且考虑到代码的效率，没有必要使小规模且本地顺序执行的的 Membership

中的每一个 Cell 并发执行，Membership 的具体实现与文中定义不完全相同。

```
package simple

import (
  "time"
  "fmt"
)

type Message struct {
  Address string
  Heartbeat int
}

type Cell struct {
  Message Message
  LocalTime int64
}

type Membership struct {
  address string
  heartbeat int
  membershipList map[string]Cell
  accept chan []Message
  deliver chan []Message
}

func NewMembership(address string, accept chan []Message, deliver chan[]Message) (instance *
 Membership){
  instance = new(Membership)
  instance.address = address
  instance.heartbeat = 0
  instance.membershipList = make(map[string]Cell,0)
  instance.accept = accept
  instance.deliver = deliver
  fmt.Printf("Initialzed Membership %p\n", instance)
  return
}

func (membership *Membership) Running() {
  for {
    select {
    case messages := <- membership.deliver:
      membership.Deliver(messages)
    case membership.accept <- membership.Accept():

    }
  }
}

func (membership *Membership) Deliver(messages []Message) {
  list := membership.membershipList
  for _, message := range messages {
    if message.Address == membership.address {
      continue
    }
    if cell, ok := list[message.Address]; !ok || (cell.Message.Heartbeat < message.Heartbeat)
  {
      //MembershipList 中没有这个节点的信息或信息是旧的，增加或更新
```

```
      if message.Address == membership.address {
        continue
      }
      if cell, ok := list[message.Address]; !ok || (cell.Message.Heartbeat < message.Heartbeat)
    {
        //MembershipList 中没有这个节点的信息或信息是旧的，增加或更新
        list[message.Address] = Cell{Message:Copy(message), LocalTime:time.Now().UnixNano()}
      }
    }
    membership.PrintUpdate()
  }

  func (membership *Membership) Accept() (messages []Message) {
    list := membership.membershipList
    messages = make([]Message, 0)
    membership.heartbeat++
    messages = append(messages, Message{membership.address, membership.heartbeat})
    for _, cell := range list {
      messages = append(messages, Copy(cell.Message))
    }
    return
  }
```

致　谢

感谢我的导师傅育熙教授，龙环副教授。龙环老师在我做毕设的过程中每次都会认真回复邮件解答我的疑问，帮我梳理工作的方向，给予了我很多非常有建设性意义的建议与指导。

感谢 F1603303 的同学和其他计算机系的同学，在远程毕设期间，大家相互提醒时间节点和注意事项，给予了我很大的帮助。

感谢制作毕业设计 LaTeX 模板 SJTUThesis 的同学，感谢其他帮助过我的同学，感谢父母、朋友们的支持和鼓励。

ON UNIFORM APPROACH TO RANDOM PROCESS MODEL

With the rapid development of massive communication systems, concurrency theory has become an important methodology for modeling and characterizing real concurrent systems. Research on concurrency theory deepens people's understanding of concurrent systems. Some of the research outcomes have already been used in mainstream programming languages with concurrency features, like Ada and Java. Process calculus is a kind of formal method using algebraic methods to study concurrent systems, such as CCS, CSP, and ACP. As a model for describing concurrent systems, process calculus has been extensively studied and successfully applied to the specification, design, analysis, and verification of actual systems.

Modern computer systems, which are open, distributed, and interactive, have both nondeterministic behaviors and random choices. To use simple, easy-to-use formal methods to describe complex concurrent systems, and to model and analyze concurrent systems, we usually use statistical behavioral characteristics of non-deterministic behaviors. Therefore, it is meaningful to introduce the concept of randomness in concurrent process models. As an important extension to concurrency theory, the probabilistic process has been widely researched. Recently, Fu proposes a uniform approach for turning process models into their randomized extensions. Since it is a model-independent method, we can use it to derive probabilistic extensions of any other process models. Fu expresses the uniform approach by demonstrating how to define the grammar and syntax of RCCS, a randomized version of CCS, and provides us a way to define bisimulation and equivalence on RCCS as well.

A value-passing calculus is a process calculus where the content of communications are values chosen from some data domain, and the propositions appearing in the conditionals are formulas constructed from logic. It can be applied to modeling and analyzing communicating processes, biological processes, and other real-world problems with value-passing characteristics. In most studies of value-passing calculus, there exist some oracles providing data domain, logic decision, and even computation of functions in value-passing processes. Those oracles usually remain undefined, making it hard to analyze the expressiveness of those value-passing models. However, Fu proposes a closed value-passing calculus, called $\mathbb{VPC}_{\mathsf{Th}}$, using a first-order theory to decide the bool expressions and a Turing complete numeric system to calculate the outcomes of functions, which successfully avoids such oracles.

Considering $\mathbb{VPC}_{\mathsf{Th}}$'s great expressiveness, we decide to use Fu's uniform approach to extend $\mathbb{VPC}_{\mathsf{Th}}$ into a probabilistic version, called $\mathbb{RVPC}_{\mathsf{Th}}$. Hopefully, $\mathbb{RVPC}_{\mathsf{Th}}$ could help to model and analyze some meaningful real-world processes such as communicating processes and network security, etc. Intending to benefit mankind, we apply Fu's uniform approach to $\mathbb{VPC}_{\mathsf{Th}}$ and get the grammar and symbolic transition syntax of $\mathbb{RVPC}_{\mathsf{Th}}$, specifically, we add a random choice operator to $\mathbb{VPC}_{\mathsf{Th}}$ as well as a transition rule to random choice.

The equivalence of processes is a basic topic of the theory of programs. In the field of process

calculus, we usually use bisimulation to describe the equivalence of process models. Bisimulation has been studied for many years since process calculus came into being. Representative work includes Milner's weak bisimulation and van Glabbeek and Weijland's branching bisimulation. As for the equivalence of the probabilistic process, there already exists some research on full probabilistic processes, finite states' probabilistic processes, etc. However, it is hard to use branching bisimulation or weak bisimulation to describe the equivalence of the value-passing process because of the conditional operator. In this way, scholars have come up with a symbolic bisimulation for value-passing calculus. Fu also has defined the symbolic bisimulation of $\mathbb{VPC}_{\mathsf{Th}}$.

Fu's uniform approach proposes a random version of branching bisimulation to describe the equivalence of RCCS. Similarly, we can use the method in Fu's uniform approach to define a random version of symbolic bisimulation for our $\mathbb{RVPC}_{\mathsf{Th}}$. The core idea to derive a random version of a certain bisimulation is to find out a random version of state-preserving silent transition. Fu's uniform approach uses the Epsilon tree to construct a random branching bisimulation. The main difficulty of using the uniform approach to define $\mathbb{RVPC}_{\mathsf{Th}}$'s symbolic bisimulation is still the conditional operator. To eliminate the influence of the conditional operator, we propose a conditional equivalence class and conditional Epsilon tree in this paper. After defining the conditional l-transition and q-transition, we propose a random symbolic bisimulation. The essence of our symbolic bisimulation is that we can find a division of a certain condition and use conditional epsilon trees corresponding to each element in the division to simulate a conditional epsilon tree. As for some special processes that have a conditional epsilon tree with all branches infinite, we define the codivergence of $\mathbb{RVPC}_{\mathsf{Th}}$ for those processes. Then we define the observance equivalence of $\mathbb{RVPC}_{\mathsf{Th}}$ as the largest co-divergent and symbolic bisimulation relation. We manage to prove the congruence of our observance equivalence as well. Meanwhile, we also provide some interesting examples to annotate the concepts mentioned above.

In addition, for showing that our newly proposed random value-passing process model is feasible to model and analyze some real-world processes, we decide to model a real-world process as a demo. Since gossip protocol is interesting, widely known, and easy to understand and failure detection is a mainstream topic of cloud computing, we choose to model and analyze a communicating process based on a failure detection mechanism, Gossip-Style Membership Protocol. There's something with randomization inside gossip protocol that for every gossip time, each node will pick k other nodes in a cluster as targets to send gossip messages. It is a good property for us to apply our random value-passing process model. We firstly construct a peer-to-peer system where nodes all use gossip protocol to communicate with each other. Since gossip is used to implement multicast in a group, we also define a specification of a multicasting system and prove that the multicasting system is symbolic bisimulated with our peer-to-peer system based on gossip protocol. Next, we adjust the definition of the former peer-to-peer system and using $\mathbb{RVPC}_{\mathsf{Th}}$ to define a membership system interacting with nodes connected to the communicating network of the peer-to-peer system. Finally, we use Golang to implement our peer-to-peer system and use HTML and javascript to show the result. The result shows that our peer-to-peer system implemented by using $\mathbb{RVPC}_{\mathsf{Th}}$ runs correctly, which implies that $\mathbb{RVPC}_{\mathsf{Th}}$ is suitable to be used in modeling and analyzing real-world processes with value-passing characteristics, such as communicating processes and biology processes, etc. Hopefully, after we define and prove several concepts of $\mathbb{RVPC}_{\mathsf{Th}}$ and demonstrate with an application of

it, modeling and analyzing real-world processes with value-passing characteristics could be easier and even routine work.

Although there still exists something not well considered in our model, our extension of the value-passing process model, for one hand, further supports the model independence of the uniform approach. On the other hand, it proves the feasibility of applying the random value-passing process model to the modeling and analysis of concurrent systems with value-passing characteristics.