





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 |

**上海交通大学**

**毕业设计（论文）原创性声明**

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

**上海交通大学**

**毕业设计（论文）版权使用授权书**

本毕业设计（论文）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索， 可以采用影印、缩印或扫描等复制手段保存和汇编本毕业设计（论文）。

**保密**□，在 年解密后适用本授权书。

本论文属于

## 不保密□。

（请在以上方框内打“**√**”）

作 者 签 名 ： 指 导 教 师 签 名 ：

日 期 ： 年 月 日 日 期 ： 年 月 日

# 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，我们证明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的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. Re- cently, 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 uni- form 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 ran- dom 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 gram- mar 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 a congruent relation. 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 analysis modern concurrent communica- tion 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, ob- servation equivalence

|  |  |  |
| --- | --- | --- |
| **目 录** | | |
| [第一章 绪论](#_bookmark1) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [1](#_bookmark1) |
| [1.1 通信系统演算](#_bookmark2) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [1](#_bookmark2) |
| [1.2 随机进程模型](#_bookmark4) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [2](#_bookmark4) |
| [1.3 研究目的](#_bookmark5)··················································································· | | [2](#_bookmark5) |
| [1.4 论文结构](#_bookmark6)··················································································· | | [2](#_bookmark6) |
| [第二章 随机传值进程模型](#_bookmark7)········································································· | | [3](#_bookmark7) |
| [2.1 传值进程模型](#_bookmark8) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [3](#_bookmark8) |
| [2.1.1](#_bookmark9) | [The Value-Passing Calculus](#_bookmark9) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [3](#_bookmark9) |
| [2.1.2](#_bookmark12) | [If Then Else 语法符号化](#_bookmark12)························································· | [4](#_bookmark12) |
| [2.2 随机传值进程模型](#_bookmark13)········································································ | | [5](#_bookmark13) |
| [2.3 随机传值进程模型中的互模拟关系](#_bookmark15) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [6](#_bookmark15) |
| [2.3.1](#_bookmark16) | [互模拟关系与观察等价性](#_bookmark16)······················································· | [6](#_bookmark16) |
| [2.3.2](#_bookmark20) | [条件等价树](#_bookmark20) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [9](#_bookmark20) |
| [2.3.3](#_bookmark31) | [随机传值进程模型的符号互模拟](#_bookmark31) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [12](#_bookmark31) |
| [2.4 随机传值进程模型的等价性](#_bookmark36)····························································· | | [15](#_bookmark36) |
| [2.5 本章小结](#_bookmark44)··················································································· | | [19](#_bookmark44) |
| [第三章 随机传值进程模型的应用](#_bookmark45) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [21](#_bookmark45) |
| [3.1 Gossip-Style Membership 协议](#_bookmark46)··························································· | | [21](#_bookmark46) |
| [3.1.1](#_bookmark47) | [Gossip 协议](#_bookmark47) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [21](#_bookmark47) |
| [3.1.2](#_bookmark48) | [组成员协议](#_bookmark48) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [21](#_bookmark48) |
| [3.2 Gossip-Style Membership 的实现](#_bookmark49) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [22](#_bookmark49) |
| [3.2.1](#_bookmark50) | [Gossip 协议的实现](#_bookmark50)······························································· | [22](#_bookmark50) |
| [3.2.2](#_bookmark52) | [Gossip 协议的等价性](#_bookmark52) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [24](#_bookmark52) |
| [3.2.3](#_bookmark55) | [Gossip-Style Membership 的实现](#_bookmark55) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [26](#_bookmark55) |
| [3.3 Gossip-Style Membership 的仿真模拟](#_bookmark60)··················································· | | [29](#_bookmark60) |
| [3.3.1](#_bookmark61) | [Go 语言与 CSP](#_bookmark61) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [29](#_bookmark61) |
| [3.3.2](#_bookmark62) | [代码实现与仿真效果](#_bookmark62) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | [30](#_bookmark62) |
| [3.4 本章小结](#_bookmark66)··················································································· | | [32](#_bookmark66) |
| [第四章 总结与展望](#_bookmark67)················································································· | | [33](#_bookmark67) |
| [附录 A 基于 Gossip-Style Membership 协议的 P2P 系统的 Go 语言实现](#_bookmark68)····················· | | [35](#_bookmark68) |
| [参考文献](#_bookmark69) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [40](#_bookmark69) |
| [致 谢](#_bookmark108) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | | [43](#_bookmark108) |

# 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，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 等程序语言中得到广泛应用[[1](#_bookmark70)]。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代理人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](#_bookmark71)]。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](#_bookmark72)]，HOARE C 提出的通信顺序进程 CSP[[4](#_bookmark73)]，BERGSTRA J 等提出的 ACP[[5](#_bookmark74)]，以及 BOLOGNESI T 等的LOTOS[[6](#_bookmark75)]。作为一种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](#_bookmark76), [8](#_bookmark77)], 概率CSP[[9](#_bookmark78)] 和概率ACP[[10](#_bookmark79)] 等。近期， FU Y 提出了一个通用方法Uniform Approach[[11](#_bookmark80)]，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 Calculus of Communicating System(CCS)[[3](#_bookmark72)] 和HOARE C 的Communicating Sequential Process(CSP)[[4](#_bookmark73)]。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](#_bookmark81)]。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。

## 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 ∣ 𝛼 .𝑇 ∣ 𝑆 ∣ 𝑇 ∣ (𝑎)𝑇 ∣ 𝜇𝑋.

∑𝑖∈𝐼 𝑖 𝑖

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

表示，𝛼𝑖 ∈ 𝐴𝑐𝑡，𝜏 表示一切静态迁移（内部动作）。∑𝑖∈𝐼 𝛼𝑖.𝑇 为非确定性选择项，若 𝐼 = ∅，

𝐶合ℎ𝐼𝑎𝑛是=有{限~~𝑎~~ ∣自𝑎然∈数𝐶ℎ集𝑎。𝑛}𝐶ℎ𝑎𝑛 为名字（或通道）的集合，其中的元素以小写字母表示，集合

，动作集合 𝐴𝑐𝑡 = 𝐶ℎ𝑎𝑛 ∪ 𝐶ℎ𝑎𝑛 ∪ {𝜏}，其中的元素用小写希腊字母

我们可以将这一项写作 0。𝑆 ∣ 𝑇 表示𝑆, 𝑇 可以并发执行。(𝑎)𝑇 为限制（Restriction）或本地化（Localization）操作子，对外隐藏 中的 通道，也可以写为 或 。 表 示 递归（Recursion）操作。若一个CCS 进程表达式 中不含自由变元，称 为进程（process）。

𝑃 𝑃

𝑇 𝑎 𝑇 \{𝑎} 𝑇 \𝑎 𝜇

CCS 的迁移语义如图 [1–1](#_bookmark3)，表示状态之间的变迁规则，其中 𝜆 ∈ 𝐴𝑐𝑡。

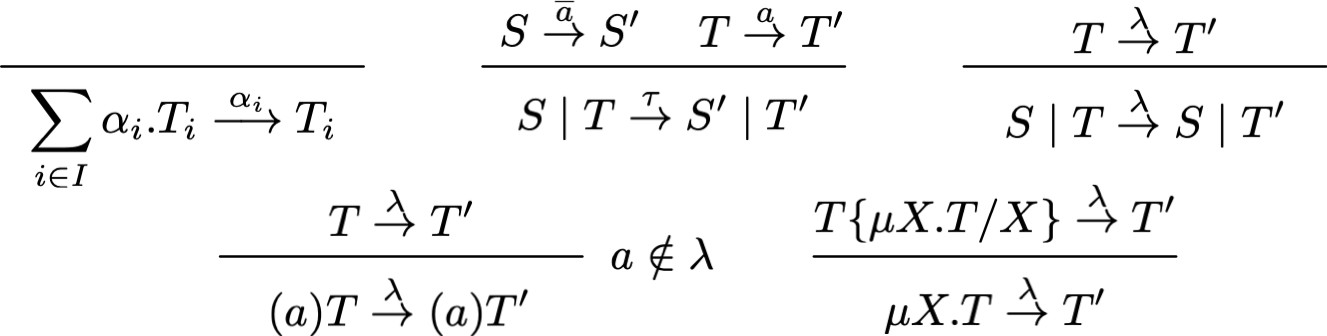


图 1–1 CCS 迁移语义

## 随机进程模型

FU Y 提供了一个模型无关的方法——An Uniform Approach to Random Process Model(Uniform Approach)[[11](#_bookmark80)]，将进程模型扩展为随机进程模型。Uniform Approach 区分了非确定行为和概率性的行为，在 CCS 的基础上建立了随机化的 CCS(后文称为 RCCS) 的语法、迁移语义并研究了 RCCS 的代数特性如互模拟关系。

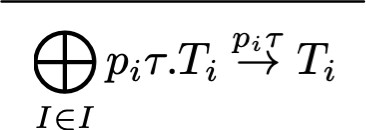
1 RCCS 在 C1CS 的基2础上增加了概率选择操作子 ⨁𝑖∈𝐼 𝑝𝑖𝜏.𝑇𝑖，其中 0 < 𝑝𝑖 < 1 ∧ ∑1𝑖∈𝐼 𝑝𝑖 =

𝑇 2

。例如，𝑆 = 3 𝜏.𝑇1 + 3 𝜏.𝑇2 意味着𝑆 经过静态迁移（内部动作）到达𝑇1 的概率为 3 ，到达

2 的概率为 3 。

对应随机选择操作子的迁移语义为：



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。

## 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。

## 论文结构

本文分为三个章节。第一章为绪论，引入了通信并发系统的概念，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。第四章为结论以及本文待改进的工作。

# 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[[3](#_bookmark72)] 中，也是使用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。我们选择传值进程模型的一种定义——The Value-Passing Calculus[[13](#_bookmark82)] 进行扩展。

## 传值进程模型

传值进程模型（Value-Passing Calculus）是一种可以将某个域中的值作为通信的内容，并且可以在特定逻辑条件执行某个动作的并发进程模型。一个经典的例子是[[3](#_bookmark72)] 中的一单元缓冲区，它可以接受、储存、发送消息：

𝐶 𝑑=𝑒𝑓 𝑖𝑛(𝑥).𝐶′(𝑥) (𝑥) 𝑑=𝑒𝑓 ~~𝑜𝑢𝑡~~(𝑥).𝐶

𝐶′

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 为例：

𝐴(𝑥)

𝐴(𝑥) = if 𝜑(𝑥) then ~~𝑎~~(𝑓 (𝑡)) else 𝐵(𝑥)

𝐴(𝑥) 在满足条件𝜑(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 无关；若不满足 ，则执行程序 。

𝑥 𝜑(𝑥) 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，通信协议的过程就可以被抽象为传值进程模型，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](#_bookmark83)]。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。

### 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 时，我们会将 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 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 将变元表达式的取值和计算全部依赖一个集合 [[15](#_bookmark84)]， LIN H 依赖一个评估 完成布尔表达式的判定和表达式的计算[[16](#_bookmark85)]，MILNER R 对CCS 的扩展 Value-Passing CCS 也没有讨论函数的计算过程和条件的可判定性[[3](#_bookmark72)]。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

𝜌

𝑉

𝑓 (𝑡)

𝐴(𝑥) 𝜑(𝑥)

神域的概念，提出了一个封闭的传值进程模型[[13](#_bookmark82)]，这个模型同时也是不针对某一应用场景的通用模型。

𝕍ℙℂ

The Value-Passing Calculus 中的传值进程模型 Th 的语法可以表示为：

𝑇 ∶= ∑𝑖∈𝐼 𝜑𝑖𝑎(𝑥).𝑇𝑖| ∑𝑖∈𝐼 𝜑𝑖~~𝑎~~(𝑡𝑖).𝑇𝑖|𝑇 |𝑇 ′|(𝑐)𝑇 |𝜑𝑇 |!𝑎(𝑥).𝑇 |!~~𝑎~~(𝑡).𝑇 (2–1)

其中𝑇 是一个𝕍ℙℂTh 项，Th 是可判定的逻辑，𝜑𝑖 是一个布尔表达式，可以通过Th 证明或

证伪，若 Th 可以给出 𝜑𝑖 的一个证明，记为 Th ⊢ 𝜑 。𝑎(𝑥) 是一个输入前缀，~~𝑎~~(𝑡 ) 是一个输𝑖 𝑖

出前缀，我们可以用 ∑𝑖∈𝐼 𝜑𝑖𝜆𝑖.𝑇𝑖 来表示 ∑𝑖∈𝐼 𝜑𝑖𝑎(𝑥).𝑇𝑖 或 ∑𝑖∈𝐼 𝜑𝑖~~𝑎~~(𝑡).𝑇𝑖。!𝑎(𝑥).𝑇 和 !~~𝑎~~(𝑡).𝑇 对应递归操作， 可以表示为 。其余的表达与第一章CCS 语法定义的解释相同。

𝜇𝑋.𝐸 (𝑐)(𝐸{𝑐(𝑧).0/𝑋}|!𝑐(𝑟).𝐸{𝑐(𝑧).0/𝑋})

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。

𝕍 ℙℂ

Th 的符号迁移语义如图 [2–1](#_bookmark11)所示。

′ ′ 𝜆 ′

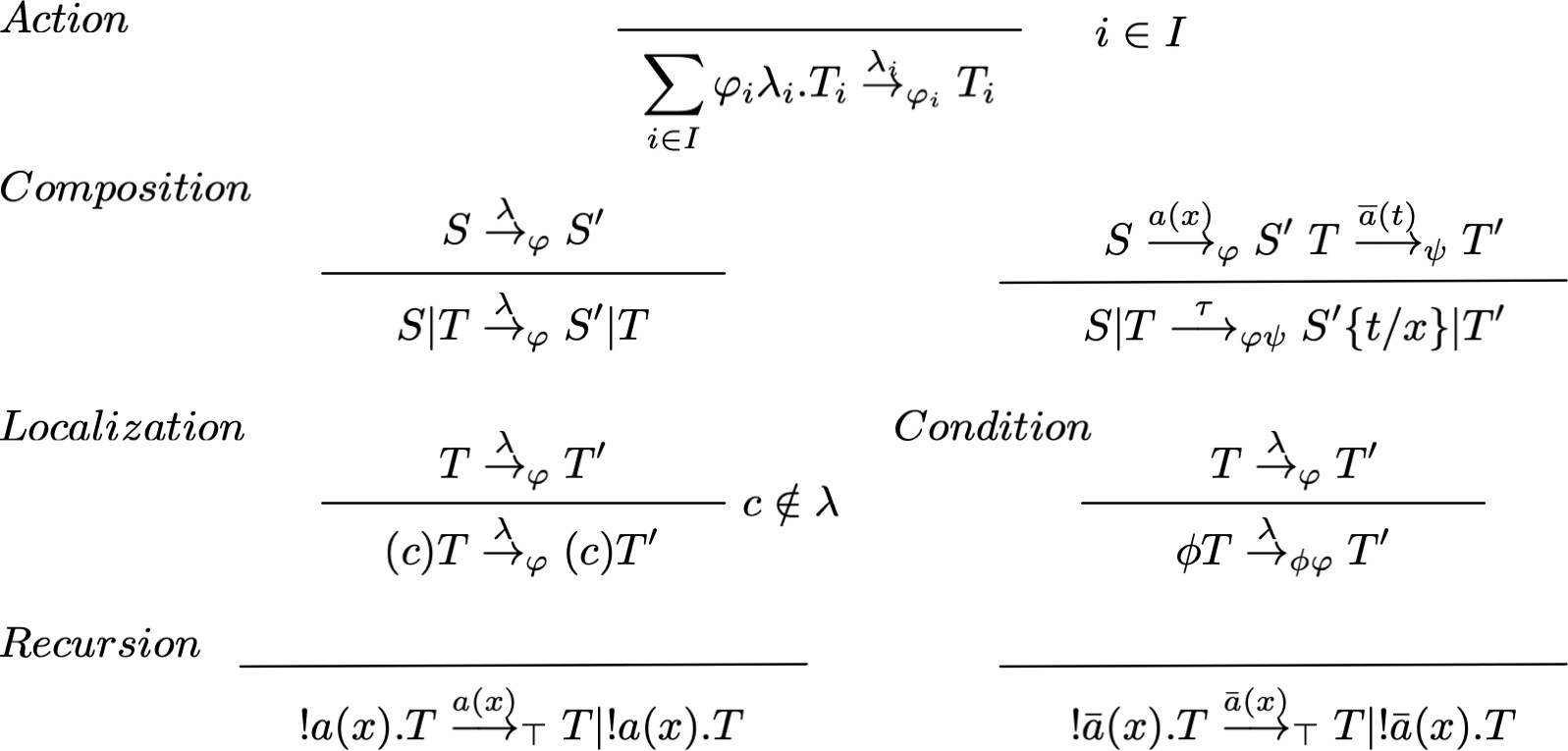


图 2–1 𝕍ℙℂTh 的符号迁移语义规则

其中，𝑇 = 𝜑𝜆.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 →𝜑 𝑇 。

符号语义中的动作集合𝐴𝑐𝑡 = {𝑎(𝑥), ~~𝑎~~(𝑡)|𝑎 ∈ 𝐶ℎ𝑎𝑛, 𝑥 ∈ V𝛴, 𝑡 ∈ T𝛴} ∪ {𝜏}，𝐴𝑐𝑡 中的元素可以用 表示，其中 V𝛴 T𝛴 为可判定逻辑 Th 中变元的集合和项的集合 [引用]。

𝜆 ,

The Value-Passing Calculus 使用可判定的一阶理论判定条件[[17](#_bookmark86)] 并提出了一个图灵完备

的数值系统（Numeric System）[[13](#_bookmark82)] 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。

### If Then Else 语法符号化

𝐴(𝑥)

进程 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](#_bookmark72)] 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，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 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 then else 可以被定义为 。以下推论依据 The Value-Passing Calculus 中对 If Then Else 语法的定义和 Th 的 Proof System，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。

𝕍ℙℂ

𝜑 𝑆 𝑇 𝜑𝑆|⌝𝜑𝑇

以下推论中 𝑆, 𝑇 ∈ 𝒯𝕍 ℙℂ 。

Th

**推论 2.1** 𝜑0 = 0

**证**0 **明** 𝜑0 = 𝜑0 + 0 = 𝜑0 + ⊤0 = 𝜑0 + (𝜑∨⌝𝜑)0 = 𝜑0 + 𝜑0+⌝𝜑0 = 𝜑0+⌝𝜑0 = (𝜑∨⌝𝜑)0 = ⊤0□=

**推论 2.2** 𝜑𝑇 = (𝜑𝑇 ∣⌝𝜑0)

**证明** 𝜑𝑇 = (𝜑𝑇 ∣ 0) = (𝜑𝑇 ∣⌝𝜑0) □

**推论 2.3** 𝑆 =⌝𝜑𝜑𝑇 ，则 𝑆 = 0。

**证明** 𝑆 =⌝𝜑𝜑𝑇 = (⌝𝜑 ∧ 𝜑)𝑇 = ⊥𝑇 = 0 □

进而我们可以得到 If Then Else 语法与 𝕍ℙℂTh 规则的对照表：

表 2–1 If Then Else 语法对照表1

𝑆 =

⌝𝜑 𝜑 𝑇 𝑆 = 0

语法

′ 𝕍ℙℂTh 规则对照

𝑆 =

if 𝜑 then 𝑇 𝑆 = (𝜑𝑇 |⌝𝜑0)

𝑆if=𝜑 then 𝑇 else 𝑇 𝑆 = (𝜑𝑇 |⌝𝜑𝑇 ′)

if then if then

## 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、生物分析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[[18](#_bookmark87)] 和生物自组装系统[[19](#_bookmark88)]。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[[20](#_bookmark89)] 等应用，但由于其是在 Value-passing CCS[[3](#_bookmark72)] 基础上的概率扩展，[[20](#_bookmark89), [21](#_bookmark90)] 仍然存在神域的问题。

𝕍ℙℂ

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 Th 的基础上，用Uniform Approach 中的随机选择 𝑖∈𝐼 𝑖 𝑖 扩展公式 [2–1](#_bookmark10)中 Th 项的定义。我们可以得到随机传值进程模型（Random VPC，记为 Th）项的定义：

ℝ𝕍ℙℂ

⨁ 𝑝 𝜏.𝑇 𝕍ℙℂ

𝑇 ∶= ⨁𝑖∈𝐼 𝑝𝑖𝜏.𝑇𝑖 ∣ ∑𝑖∈𝐼 𝜑𝑖𝜆𝑖.𝑇𝑖 ∣ 𝑇 ∣ 𝑇 ′ ∣ (𝑐)𝑇 ∣ 𝜑𝑇 ∣!𝑎(𝑥).𝑇 ∣!𝑎(𝑡).𝑇 (2–2)

其中，与 [2.1.1](#_bookmark9)中定义的一样：𝜆𝑖 ∈ 𝐴𝑐𝑡，Th 为可判定的一阶理论。随机选择操作子 ⨁𝑖∈𝐼 𝑝𝑖𝜏.𝑇𝑖

𝑇 0 < [𝑝 <](#_bookmark9) 1, ∑ 𝑝 = 1 𝑆 = ⨁ 𝑝 𝜏.𝑇 𝑆 𝑝 𝜏

中， 𝑖 𝑖∈𝐼 𝑖 。 𝑖∈𝐼 𝑖 𝑖 意味着 在 𝑖 的概率下经过内部动作 到达

𝑖 状𝒯态。

ℝ𝕍ℙℂTh 为所有 ℝ𝕍ℙℂTh 的项 𝑇 的集合。

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 ⊤，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 Th 中的条件操作子， 得到 𝑖∈𝐼 𝑖 𝑖 。

𝜑(⨁ 𝑝 𝜏.𝑇 )

𝕍ℙℂ

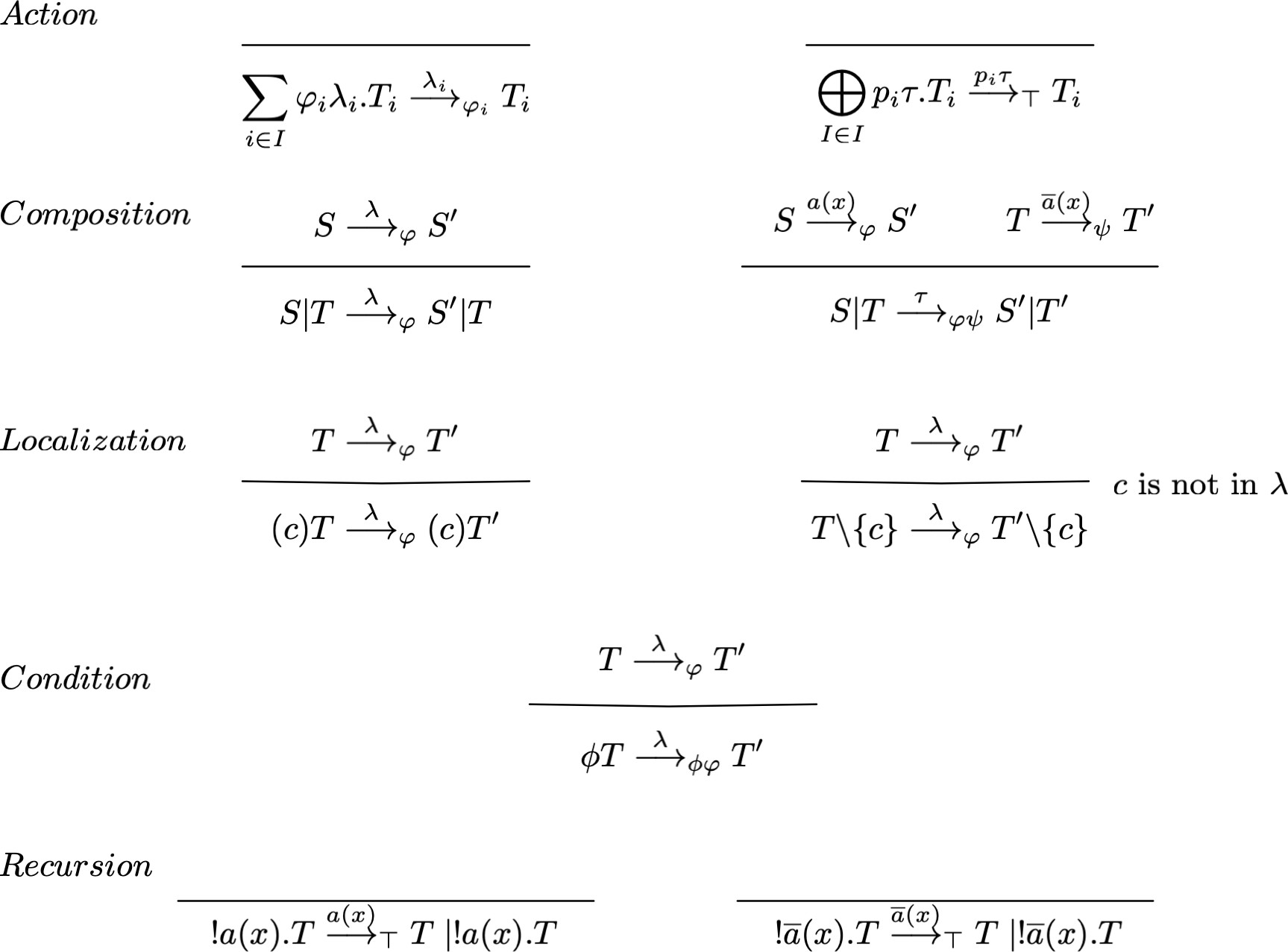


图 2–2 随机传值进程模型的符号迁移语义

对于 𝕍ℙℂTh 的前缀操作子的迁移规则：∑𝑖∈𝐼 𝜑𝑖𝜆𝑖.𝑇𝑖 →𝜆𝑖 𝜑𝑖 𝑇𝑖，它本质上代表了非确定 选择，即作出 𝑖 条件下的 𝑖 动作到达 𝑖 状态是非确定的，我们可以通过 Uniform Ap-

𝑝 𝜑 𝜆

𝜑 𝜆 𝑇

proach𝑝的𝜏.𝜑方𝜆法.𝑇将它𝑝→𝜏扩→𝜆展为𝑇一个概率性选择，即在概率 𝑖 下会作出 𝑖 条件下的 𝑖 动作：

⨁𝑖∈𝐼 𝑖

达𝜆 一个 ℝ𝕍ℙℂTh 状态: 𝜑𝑖𝜆𝑖.𝑇𝑖，若 Th ⊢ 𝜑𝑖（即一阶理论 Th 下，𝜑𝑖 为真），则我们可以经过

𝑖 𝑖 𝑖

𝑖 ⊤

𝑖 𝜑

𝑖。它的语义为：在概率 𝑝𝑖 下，我们会经过一个内部 𝜏 操作到

𝑖 操ℝ作𝕍到ℙ达ℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂTh 状态 𝑇𝑖，其中 𝜆𝑖 ∈ {𝑎(𝑥), 𝑎(𝑥) ∣ 𝑎 ∈ 𝒩 , 𝑥 ∈ V𝛴, 𝑡 ∈ T𝛴} ∪ {𝜏}。

Th 在 Th 的基础1𝜏 上扩展了前缀操作： ，其中 。类似 的

𝕍ℙℂ 𝑝𝜏. 𝑝 ∈ (0, 1) 𝐴 = 𝜏.𝐵

。若将 𝕍ℙℂ 中的此类内部操作同样看待，我们可以得出推论 [2.4](#_bookmark14)。Th

1, 𝐴1

=Th𝐵项，我们可以认为 𝐴 →⊤ 𝐵，这时也满足 𝐴 = ⨁𝑖∈𝐼 𝑝𝑖𝜏.𝐴𝑖 的定义，此时 𝐼 = {1}, 𝑝1 =

**推论 2.4** 𝕍ℙℂTh 是一种特殊的 ℝ𝕍ℙℂTh。

## 随机传值进程模型中的互模拟关系

### 互模拟关系与观察等价性

𝑅 𝐴

程序理论的基本问题是进程的等价性。等价关系定义为：设 是非空集合 上的二元关系，若 是自反的、对称的、传递的，则称 是 上的等价关系。研究等价关系的目的

𝑅 𝑅 𝐴

可利用等价类来选择测试用例[[22](#_bookmark91)]。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](#_bookmark72)]。van GLABBEEK R 和 WEIJLAND W 提出的分支互模拟 *(branching bisimulation)*[[23](#_bookmark92), [24]](#_bookmark93) 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[[25](#_bookmark94)]，仅适用于有限状态的概率进程模型的研究[[26](#_bookmark95), [27](#_bookmark96)] 等。Uniform Approach 在分支互模拟[[23](#_bookmark92), [24](#_bookmark93)] 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。

𝒫

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。𝐶𝐶𝑆

上的分支互模拟的定义如下：

**定义 2.1** ℰ𝑙 是′在 𝒫𝐶𝐶𝑆 上的对′ 称关系，若对′ 于″任意 𝐴, 𝐵𝜏∈ 𝒫𝐶𝜏𝐶𝑆 , ′𝐴ℰ𝑙 𝐵 满″ 足： ′ ′ ″

若𝐴 → 𝐴 ，则 𝑙 = 𝜏 且𝐴 ℰ𝐵 或存在𝐵 , 𝐵 使得𝐵 → ⋯ → 𝐵 → 𝐵 且𝐴ℰ𝐵 ∧𝐴 ℰ𝐵 。

则称 ℰ 是在 𝒫𝐶𝐶𝑆 上一个分支互模拟关系。

𝐴 →𝜏 内𝐴′部∼操𝐴作 𝜏 带来的状态迁移可𝜏 以称′ 为静态𝜏迁移。𝜏 若′∼ 是 𝒫𝐶𝐶𝑆 上的一个等价关𝜏

系当′

时，我们可以写为 𝐴 →∼ 𝐴 ，当 𝐴 → ⋯ → 𝐴 ∼ 𝐴 时，我们可以写为 𝐴 ⇒∼ 𝐴 。

在Uniform Appr𝜏oach 中𝜏 ，这′ 种动作称为状态保持的静态迁移（*state-pre*𝜏*serving silent transition*），

对应的，若 𝐴 → ⋯ → 𝐴 ≁ 𝐴，称为状态改变的静态迁移。⇒∼ 为 →∼ 的闭包。

Uniform Approach 中 RCCS 的分支互模拟涉及等价集的概念，我们首先看 CCS 等价集的定义：

(𝐴, 𝐵) ∈ ℰ 𝐵 ∈ [𝐴] 𝒫 /ℰ 𝒫 ℰ

𝐴**定**∈**义**[**2**𝐴**.2**]

集合。

若二元关系 ℰ 是 𝒫𝐶𝐶𝑆 上的等价关系。𝐴 关于等价关系 ℰ 的等价集写作 [𝐴]ℰ ，

ℰ ，若 ，有 ℰ 。我们用 𝐶𝐶𝑆 表示所有 𝐶𝐶𝑆 关于 的等价集的

𝒫𝑅𝐶𝐶𝑆 ，𝒯𝕍 ℙℂTh ，𝒯ℝ𝕍 ℙℂTh 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](#_bookmark18)的形式：

[𝐴]

**定义**

**2.3** ℰ 是𝒫𝐶𝐶𝑆 上的𝑙

等价关系，若𝑙对于任意动作 𝑙, 𝑙 ≠ 𝜏 和任意等价集𝒞 ∈ 𝒫𝐶𝐶𝑆 /ℰ, 𝒞 ≠

ℰ ，满足若 𝐵ℰ𝐴 ⇒ℰ → 𝒞，则 𝐵 ⇒ℰ → 𝒞，称 ℰ 是 𝒫𝐶𝐶𝑆 上的分支互模拟关系。

定义 [2.3](#_bookmark18)与定义 [2.1](#_bookmark17)虽然本质上是相同的，但是定义 [2.3](#_bookmark18)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转移之间的相互模拟。

𝜏 ′

考虑到 RCCS 在𝑝𝜏 CC′ S 的基础上增加了概率选择，状态保持的静态迁移在 𝐴 → 𝐴 ℰ𝐴

的基础上新增了 𝐴 → 𝐴 ℰ𝐴, 0 < 𝑝 < 1 的可能，为了用类似 ⇒ℰ 的方式表示这种静态迁移，

U𝐴niform Approach 提出了等价树（*Epsilon Tree*）的概念[[11](#_bookmark80)]，将概率选择作为树的分支，节点种迁移： -迁移( -transition) 和 -迁移( -transition)，分别表示等价树上的节点迁移到其他等价集动作和概率，Uniform Approach 中定义的概率版分支互模拟是对 -迁移和 -迁移的相互

模拟。

𝑙 𝑞

𝑙 𝑙 𝑞 𝑞

的关于等价关系 ℰ 的等价树上的任意节点 𝑁 ∈ [𝐴]ℰ 。同时，Uniform Approach 定义了两

示𝐴 ⇝𝐴 关→𝑙 于ℬ等价关系 ℰ 的等价树上的每一个叶子结点 𝐿，存在 𝐿 → 𝐿

**定义 2.4 (𝒍-迁移 (𝒍-transition))** 𝐴 ∈ 𝒫𝑅𝐶𝐶𝑆 ，从 𝐴 到 ℬ ∈ 𝒫𝑅𝐶𝐶𝑆 /𝑙ℰ ℬ ′≠ [𝐴]ℰ 的 𝑙-迁移表

∈ ℬ, 𝑙 ≠ 𝜏，写作

ℰ 。

关系 ℰ 的等价∐树上的𝑝 𝜏每一个叶子结点𝑝，

**定义 2.5 (𝒒-迁移 (𝒒-transition))** 𝐴 ∈ 𝒫𝑅𝐶𝐶𝑆 ，ℰ 为𝒫𝑅𝐶𝐶𝑆 上的等价关系，𝐿 为𝐴 关于等价

ℬ ∈ 𝒫𝑅𝐶𝐶𝑆 /ℰ, ℬ ≠ [𝐴]ℰ 。

定 义 P(𝐿

定义 Pℰ (𝐿 ⟶

𝑖⟶∈[𝑘]

ℬ) = ∑{𝑝𝑖|𝐿 ⟶𝑖

𝐿𝑖 ∈ ℬ ∧ 𝑖 ∈ 𝐼}。

∐𝑖∈[𝑘] 𝑝𝑖𝜏

∐𝑖∈[𝑘] 𝑝𝑖𝜏

ℬ) = P(𝐿

∐𝑖⟶∈[𝑘] 𝑝𝑖𝜏 ℬ)/(𝑞1 − P(𝐿

∐𝑖⟶∈[𝑘] 𝑝𝑖𝜏 [𝐴]ℰ ))。

当 Pℰ (𝐿 ⟶ ℬ) = 𝑞 时，记为 𝐴 ⇝ℰ → 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** 当 𝒫𝑅𝐶𝐶𝑆𝑙 上的等价关系 ℰ 满足： 𝑙

* + - * 若 𝐵ℰ𝐴 ⇝ℰ →𝑞 𝒞 ∈ 𝒫𝑅𝐶𝐶𝑆 /ℰ, 𝑙 ≠ 𝜏 ∧ 𝒞 ≠ [𝐴]ℰ ，则 𝐵 ⇝ℰ →𝑞 𝒞。
      * 若 𝐵ℰ𝐴 ⇝ℰ → 𝒞 ∈ 𝒫𝑅𝐶𝐶𝑆 /ℰ, 𝑙 ≠ 𝜏 ∧ 𝒞 ≠ [𝐴]ℰ ，则 𝐵 ⇝ℰ → 𝒞。 则等价关系 是分支互模拟关系。

ℰ

𝕍ℙℂTh 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 = ~~𝑎~~(0).𝑆

可以作出𝐴 → ⊤ 𝑆 的动作，而 𝐵 不能。但对于𝑎(𝐴0) → 𝑥=0 𝑆 和𝑎𝐴(0) → ⌝(𝑥=0) 𝑆，𝐵 却可以𝑎相(0) 应的

和 𝐵 = ((𝑥 =𝑎(00))~~𝑎~~(0).𝑆|⌝(𝑥 = 0)~~𝑎~~(0).𝑆) 在观察上应𝑎该(0)是一致的，𝑎但(0)若我们应用分支互模拟 𝐴

作出模拟的动作。这就要求 𝐵 的两个操作：𝐵 ⟶𝑥=0 𝑆 和𝐵 ⟶⌝(𝑥=0) 𝑆 共同模拟𝐴 ⟶⊤ 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[[28](#_bookmark97)]。在 The Value-Passing Cal-

𝕍ℙℂ

culus 中，FU Y 也提出了 Th 的符号互模拟[[13](#_bookmark82)]。

ℰ**定义 2.7 (符号互模拟)** ℰ 是一个 𝒯𝕍 ℙℂTh 上的二元对称关系，当 𝐴ℰ𝐵 满足下列条件时，称

是一个符号𝜏互模拟′ 关系：

𝐴 → 𝐴 𝜑 {𝜑 } {𝐵 ⇒ 𝐵 } ∀𝑖 ∈ 𝐼 ⊢

* 若 𝜑 ，则存在 的划分 ′ 𝑖 𝑖∈𝐼 和集合𝜏 ′𝜓𝑖 𝑖 𝑖∈𝐼 ，对于 ，Th′ ′ ′

𝜑𝑖 ⇒𝑎𝜓(𝑡)𝑖 ∧ 𝜑𝑖𝐴ℰ𝜑𝑖𝐵𝑖，且满足𝜑𝑖𝐴 ℰ𝜑𝑖𝐵𝑖 或 𝐵𝑖 →𝜓𝑖′ 𝐵𝑖 ∧ T𝑎(h𝑡 )⊢ 𝜑𝑖 ⇒ 𝜓𝑖 ∧ 𝜑𝑖𝐴 ℰ𝜑𝑖𝐵𝑖 。

* + - * 若 𝐴′ →𝜑 𝐴′，则存在 𝜑 的划分{𝜑𝑖}𝑖∈𝐼 和集′ 合{𝐵′ ⇒𝜓𝑖 𝐵𝑖

→𝑖 𝜓𝑖′ 𝐵𝑖′}，满足 Th ⊢ (𝜑𝑖 ⇒

𝜓𝑖𝜓𝑖 )𝑎(∧𝑥)(𝜑𝑖 ⇒′ 𝑡 = 𝑡𝑖) 且 (𝜑𝑖𝐴ℰ𝜑𝑖𝐵𝑖) ∧ (𝜑𝑖𝐴 ℰ𝜑𝑖𝐵𝑖 )。

𝑎(𝑥) ′

* + - * 若𝜓 𝜓𝐴′ → 𝜑 𝐴 ，则存在 𝜑 的划′ 分{𝜑′ 𝑖}𝑖∈𝐼 和集合{𝐵 ⇒𝜓𝑖 𝐵𝑖 → 𝜓𝑖′ 𝐵𝑖 }，满足 Th ⊢ 𝜑𝑖 ⇒

𝑖 𝑖 且 (𝜑𝑖𝐴ℰ𝜑𝑖𝐵𝑖𝑠) ∧ (𝜑𝑖𝐴 ℰ𝜑𝑖𝐵𝑖 )。

互模拟关系的全集记为 ≊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 = ~~𝑎~~(0).𝑆 和 𝐵 = ((𝑥 = 0)~~𝑎~~(0).𝑆|⌝(𝑥 = 0)~~𝑎~~(0).𝑆) 是符号互模拟的。

**证明** 定义等价关系 ℰ = {(𝐴, 𝐵)}∪ ≡，其中 ≡ 为绝对等价关系，题目等价于证明 ℰ 是一个

符号互模拟关𝑎(0系) 。

对于 𝐴 → ⊤ 𝑆，存在 ⊤ 的划分 {(𝑥 = 0), ⌝(𝑥 = 0)} 和集合 {𝐵 ⟶𝑥=0 𝑆, 𝐵 ⟶⌝(𝑥=0) 𝑆}，

𝑎(0)

𝑎(0)

且 (𝑥 = 0)𝑆ℰ𝑎((𝑥0)= 0)𝑆。

𝑎(0)

𝑎(0)

对于 𝐵 ⟶𝑥=0 𝑆，存在 𝐴 ⟶𝑥=0 𝑆 与之符号互模拟。𝐵 ⟶⌝(𝑥=0) 𝑆 同理。 □

2.3.2 条件等价树

到 𝐴 和 𝐴

5)~~𝑎~~( 根据 [2.3.1](#_bookmark16)中′ 定义 [2.7](#_bookmark19)中定义的 𝕍ℙℂTh 的符号互模拟关系，我们观察 𝐴(𝑥) = (𝑥 ≥

作是 𝐴(𝑥) 无法完成的，所以不存在 𝑥 ≥ 3 的划分，′使得 𝐴(𝑥) 可以给出一个迁移的集合′

s(𝑥)).0)′ 和 𝐴 (𝑥) = [(𝑥](#_bookmark19) ≥ 3)~~𝑎~~(s(𝑥)).0，′ 根据𝑎( 𝕍ℙℂTh 的符号互模拟的定义，我们容易得

来𝐵′模(𝑥)拟=整(个(𝑥 𝑥> ≥5)3∧。(但𝑥 反≥ 观3))𝐵~~𝑎~~((𝑥) = (𝑥 > 5)𝐴(𝑥),𝐵 (𝑥) = (𝑥 > 5)𝐴 (𝑥)，我们可以得到

是不符号互模拟的，对于 𝐴 (𝑡)

⟶s(𝑡))𝑥≥3 0，因为 3 < 𝑥 < 5 的这一部分动

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：

s(𝑥)).0 = (𝑥 > 5)𝑎(s(𝑥)).0，同理 𝐵(𝑥) = (𝑥 > 5)′~~𝑎~~(s(𝑥)).0，

**定**𝐴′**义**∈ **2**[𝐴**.8**]**(条件等价集)** 𝐴, 𝐴′ ∈ 𝒯ℝ𝕍 ℙℂTh ，ℰ 是 ℝ𝕍ℙℂTh 上的等价关系，若 𝜑𝐴ℰ𝜑𝐴′，则 所有 Th 关于布尔表达式 和等价关系 的条件等价集的集合。

ℝ𝕍ℙℂ 𝜑 ℰ

𝜑ℰ 。[𝐴]𝜑ℰ 称为等价关系 ℰ 在条件 𝜑 下包含 𝐴 的等价集。我们用 𝒯 /𝜑ℰ 表示ℝ𝕍 ℙℂTh

[𝐴(𝑥)由] 于 𝕍ℙℂTh 是特殊𝑠的 ℝ𝕍ℙℂTh，条件等价集的定义同样适用 𝕍ℙℂTh。显然，𝐴′(𝑥) ∈

条件 ⊤ 具有一定的特殊性：对于无条件的操作我们认为操作是在 ⊤ 条件下执行的，因

(𝑥>5)ℰ ， 其 中 ℰ =≊ 。Th

此 ⊤ 可以看作是最强的条件。

**推论 2.5** 𝐴, 𝐵 ∈ 𝒯𝕍 ℙℂTh ，若 𝐴 ≊𝑠Th 𝐵，则对布尔表达式 𝜑，𝜑𝐴 ≊T𝑠h 𝜑𝐵。

**证明** 对于 𝐴𝜏 可以′ 执行的′操作𝜏 ，𝜑′𝐴′ 也会有对应的版本：

* + - * 若 𝐴 →𝑎(𝑥𝜑) 𝐴 ，′ 则 𝜑 𝐴 →𝑎(𝜑𝑥)′𝜑 𝐴 。
* 若 𝐴 𝑎→(𝑡) 𝜑 𝐴 ，则 𝜑′𝐴𝑎→(𝑡) 𝜑′𝜑 𝐴′。

′• 若𝑎(𝑥𝐴)

→𝜑 𝐴′，则 𝜑′𝐴 𝑠→𝜑′𝜑 𝐴′。

𝑎(𝑥)

对𝐵′}𝜑 𝐴 → 𝜑′𝜑 𝐴′，根据 𝐴 ≊T′h 𝐵，我们𝑠 知道存在 𝜑′的划𝑠 分 {𝜑′ 𝑖}𝑖∈𝐼 和集合 {𝐵 ⇒𝜓𝑖 𝐵𝑖 → 𝜓𝑖′

𝑖 𝑖∈𝐼 ，满足 Th′ ⊢ 𝜑𝑖𝑠 ⇒ 𝜓′ 𝑖𝜓𝑖 ∧ 𝜑𝑖𝐴 ≊Th 𝜑𝑖𝐵𝑖 ∧ 𝜑𝑖𝐴 ≊Th 𝜑𝑖𝐵𝑖 。′ ′ 𝑠

若要证明 𝜑 𝐴 ≊Th 𝜑 𝐵，我们可以构造等价关系 𝒮 = {(𝜑 𝐴, 𝜑 𝐵)|(𝐴, 𝐵) ∈≊Th}，证明是符号互模拟𝑠 的，其中 为任意 𝕍 ℙℂ′ Th 。

𝐴, 𝐵 𝒯

𝒮

通过 𝐴 ′≊Th 𝐵 我′ 们可以得到集合 {𝜑 𝜑𝑖}𝑖∈𝐼 ，对于任′ 意 𝑖, 𝑗，′ 由于 𝜑𝑖 ∧ 𝜑′ 𝑗 = ⊥，我′们

可以得到 (𝜑 𝜑𝑖) ∧ (𝜑 𝜑𝑗 ) = ⊥；又′ ⋁𝑖∈𝐼 𝜑𝑖 = 𝜑，𝑎(𝑥⋁) 𝑖∈𝐼 𝜑 𝜑𝑖 = 𝜑 𝜑，因此 {𝜑 𝜑𝑖}𝑖∈𝐼 为 𝜑 𝜑

的划分。我们还可以得到集合 {𝜑 𝐵 ⇒𝜑′𝜓𝑖 𝐵𝑖 → 𝜓𝑖′ 𝐵𝑖′}，满足 Th ⊢ 𝜑′𝜑𝑖 𝑎⇒(𝑥) 𝜑′𝜓𝑖𝜓𝑖′，而

(𝜑′𝜑𝑎𝑖𝐴(𝑥,) 𝜑′𝜑𝑖𝐵𝑖), (𝜑′𝜑𝑖𝐴′, 𝜑′𝜑𝑖𝐵𝑖′) ∈ 𝒮，因此我们可以用 {𝜑′𝐵 ⇒𝜑′𝜓𝑖 𝐵𝑖 → 𝜓𝑖′ 𝐵𝑖′} 模拟

𝜑′𝐴 → 𝜑′𝜑 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 ∈ ℝ𝕍ℙℂTh，ℰ 是ℝ𝕍ℙℂTh 上的等价关系，𝜑 是任意布尔表达式，若 𝐵 ∈ [𝐴]ℰ ，则 𝜑ℰ 。

𝐵 ∈ [𝐴]

推论 [2.6](#_bookmark21)可以简单理解为：一个ℝ𝕍ℙℂTh 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。

𝑓 𝑣(𝜑) 𝑣 ∈ 𝑉 𝜑

**证明** 𝐴, 𝐵 ∈ 𝒯ℝ𝕍 ℙℂTh ，𝑓𝑣(𝜑) 为 𝜑 中的自由变元的集合，𝑉 为 𝑓 𝑣(𝜑) 的全部取值的集合， 对于 的每一个赋值 ，我们可以得到一个无自由变元的布尔表达式 𝑣：

* + - * 若 Th ⊢ 𝜑𝑣，𝜑𝑣𝐴 = 𝐴ℰ𝐵 = 𝜑𝑣𝐵。

⊬ 𝜑 𝜑 𝐴 = 0 = 𝜑 𝐵 □

* + - * 若 Th 𝑣， 𝑣 𝑣 。

将 ⊤ 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 ∈ 𝒯𝕍 ℙℂTh ，若𝜑𝐴 ≊𝑠Th 𝜑𝐵，则对布尔表达式𝜑′, Th ⊢ 𝜑′ ⇒ 𝜑，𝜑′𝐴 ≊T𝑠h 𝜑′𝐵。

**推论 2.8** 𝐴, 𝐵 ∈ ℝ𝕍′ℙℂTh，ℰ 是′ ⇒ℝ𝜑𝕍 ℙℂTh 上的等价关系，𝜑 是任意布尔表达式，若𝐵 ∈ [𝐴]𝜑ℰ ，

则对于布尔表达式 𝜑 , Th ⊢ 𝜑

，𝐵 ∈ [𝐴]𝜑′ℰ 。

**推论 2.**′**9** 𝐿 ∈ 𝒯ℝ𝕍 ℙℂTh ，′ 𝐿 关于布尔表达式𝜑 和等价关系ℰ 的等价集为𝒞 = [𝐿]𝜑ℰ ，对任意

Th ⊢ 𝜑 ⇒ 𝜑，𝐿 关于 𝜑 ℰ 的等价集 [𝐿]𝜑′ℰ 是 𝒞 的超集，可以将 [𝐿]𝜑′ℰ 记为 [𝒞]𝜑′ℰ 。

推论 [2.9](#_bookmark24)可以由推论 [2.8](#_bookmark23)得出。

𝐵′ ⟶𝑎(定𝑥) 义 𝐵[2.](#_bookmark19)′[7}](#_bookmark19)给出了 [𝕍](#_bookmark23)ℙℂTh 的互模拟的定义，观察定义 [2.7](#_bookmark19)中的𝐴 ⟶𝜑 𝐴 ，我们用{𝐵 ⇒𝜓𝑖

𝑎(𝑥) ′

𝑖 𝜓𝑖′

𝑖 模拟′，其中′

(𝜑𝑖𝐴ℰ𝜑𝑖𝐵𝑖) ∧ (𝑠𝜑𝑖𝐴′ℰ𝜑𝑖𝐵𝑖′)𝑠，根据条件等价集的定义，我们可以得

到 𝐵𝑖 ∈ [𝐴]𝜑ℰ ，𝐵𝑖 ∈ [𝐴 ]𝜑ℰ ，由于 𝐵𝑖 ≊Th 𝐴，𝐵𝑖 ≊Th 𝐵，那么 𝐵 ⇒𝜓𝑖 𝐵𝑖 ∈ [𝐵]𝜑ℰ 实际上也

是状态保持的静态迁移。

𝑞𝜏

对于我们概率扩展后的ℝ𝕍ℙℂTh，对上述例子状态保持静态迁移多了 𝐵 →𝜓𝑖 𝐵𝑖 的情况，其中 ，表示概率的状态保持的静态迁移。我们希望可以找到 Th 中类似 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 Th 中称为条件等价树。

ℝ𝕍ℙℂ

𝑞 ∈ (0, 1) ℝ𝕍ℙℂ ⇒

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 ∈ 𝒯ℝ𝕍 ℙℂTh , 𝐴 的静态迁移树 𝑡 满足如下定义：

* + - * 每一个节点都被标记成 𝒯ℝ𝕍 ℙℂTh 的一个项，𝐴 是根节点。 ′
      * 𝐴节″点间的边被标记成 (𝜑, 𝑝)，其中 𝑝 ∈′(0𝑝𝜏, 1]，″𝜑 是一个布尔表达式。如果一条从 𝐴 到

的有向边被标记成 (𝜑, 𝑝)，表示 𝐴 →𝜑 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，ℰ 是 𝒯ℝ𝕍 ℙℂTh 上的等价关系，𝐴 ∈ 𝒯ℝ𝕍 ℙℂTh ， 当下列条件成立时， 的静态迁移树 称为是一个关于 的条件等价树（ -tree) ，记作

𝑡𝐴 𝐴 𝑡 𝜑ℰ 𝜑ℰ

𝜑ℰ 。

(2) 若 𝑡 的边被标记为 (𝜓, 𝑞)，(则𝜓,𝑞)Th ⊢ 𝜑 ⇒ 𝜓。

(1) 𝑡 上的所有节点 𝑁，𝑁 ∈ [𝐴]𝜑ℰ ，并被重新标记为 𝜑𝑁。

∐𝑖∈[𝑘]

(3) 若𝑡 𝐵, 𝐵′ 是𝑡 上的节点，𝐵 → 𝐵′, 𝑞 ∈ (0, 1)，则存在 𝐵 ⟶ 𝜓 ∐𝑖∈[𝑘] 𝐵𝑖，𝐵𝑖, 𝑖 ∈ [𝑘] 是

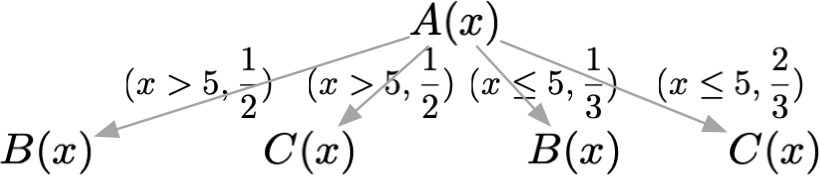
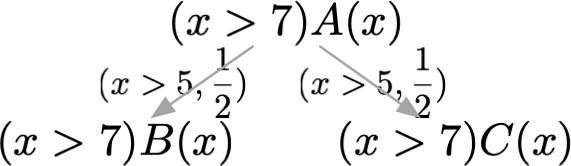
上的节′ 点，且𝐵 有且仅有(𝜓𝐵,1), ⋯′, 𝐵𝑘 这𝑘 个𝜏儿子节′ 点，且根据(2)，′有 Th ⊢ 𝜑 ⇒ 𝜓。

(4) 若 𝐵, 𝐵 是 𝑡 上的节点，𝐵 → 𝐵 ，则有 𝐵 →𝜓 𝐵 且 𝐵 有且仅有 𝐵 一个儿子节点，

且根据 (2)，有 Th ⊢ 𝜑 ⇒ 𝜓。

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**𝐶(**2**𝑥**.**)**2**]

图 2–3 图 2–4

𝐴(𝑥) = (𝑥 > 5)( 12 𝜏.𝐵(𝑥)⊕ 12 𝜏.𝐶(𝑥))|(𝑥 ≤ 5)( 13 𝜏.𝐵(𝑥)⊕ 23 𝜏.𝐶(𝑥))。若[𝐴(𝑥)]ℰ = [𝐵(𝑥)]ℰ =

ℰ ，我们可以得到 𝐴(𝑥) 的静态迁移树如图[2–3](#_bookmark25)，𝐴(𝑥) 关于(𝑥 > 7)ℰ 的条件等价树如图[2–](#_bookmark25)

[4](#_bookmark25)。 𝑑𝑒𝑓 1 1 1

**例 2.**[**3**𝐻(𝐻𝑥)(]𝑥) ==[𝐺(𝑥(𝑥≤)]3)( 3 𝜏.(𝑥 ≤ 1)𝐺(𝑥) ⊕ 3 𝜏.(𝑥 ≤ 2)𝐺(𝑥) ⊕ 3 𝜏.(𝑥 ≤ 3)𝐺(𝑥))

的静态迁移树如图 [2–5](#_bookmark26)，它包含了 𝐻(𝑥) 所有的静态迁移。

𝐻(𝑥) ℰ1

ℰ1 ，𝐻(𝑥), 𝐺(𝑥) ∈ ℝ𝕍ℙℂTh，其中 Th = PA[cite]。

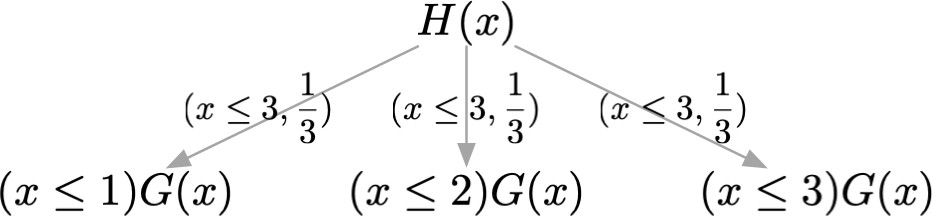
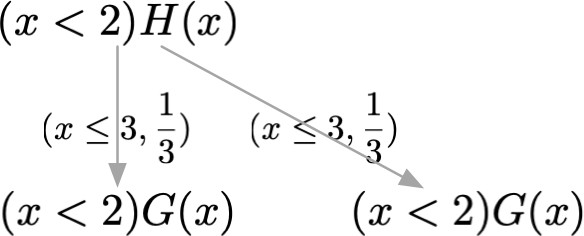
 

图 2–5 图 2–6

因为 Th ⊢ ⊤ ⇏ (𝑥 ≤ 3)，𝐻(𝑥) 的 ⊤ℰ1-tree 只有一个根节点 𝐻(𝑥)。

𝐻同(理𝑥)，𝐻(𝑥) 的 (𝑥 > 3)ℰ1-tree 只有一个根节点 𝐻(𝑥)。

2)𝐺(𝑥) = (𝑥的<(𝑥2)𝐺<(𝑥2)ℰ1-tree 如图 [2–6](#_bookmark26)，其中 Th ⊢ (𝑥 < 2) ⇒ (𝑥 ≤ 3)，由于 (𝑥 < 2) ∧ (𝑥 ≤

分支同理。此时(𝑥 < 2)𝐻(𝑥) →3

，𝐺(𝑥) ∈2[𝜏𝐻[(𝑥)]](#_bookmark26)ℰ1 ，根据推论 [2.6](#_bookmark21)，(𝑥 ≤ 2)𝐺(𝑥) ∈ [𝐻(𝑥)](𝑥<2)ℰ1 。另一个

和边都视为状态无关。

⊤ [(𝑥 < 2)𝐺(𝑥)](𝑥<2)ℰ1 ，因此在条件树内部，我们可以将节点

**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 12 ，可以通过内部自动校准系统(静态迁移： 操作) 修复，它的内部有一个计时器，每隔定长时间（ 操作） 就会从 通道广播录好的一段下课铃，录好的下课铃可以用变元 表示。这个下课铃系统可以被抽象为一个 Th，其中 Th 可以认为是 PA，我们可以用 来表示这个系统：

ℝ𝕍ℙℂ 𝐺(𝑥)

𝑎 𝑥

𝜏 𝜏

~~2~~ ~~2~~

𝐺(𝑥)

𝐺(𝑥) 𝑑=𝑒𝑓 𝜇𝑋.( 1 𝜏.𝑋 ⊕ 1 𝜏.𝑎(𝑥).𝑋)

在 ℰ2 的等价集 [𝐺(𝑥)]⊤ℰ2 = [~~𝑎~~(𝑥).𝐺(𝑥)]⊤ℰ2 。𝐺(𝑥) 的 ⊤ℰ2-tree 如图 [2–7](#_bookmark29)。

**例 2.5** 假设例 [2.4](#_bookmark27)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](#_bookmark27) 来表示音量，音量必须满足 𝑦 > 1 才可以播放，当 𝑦 ≤ 0 时下课铃系统音

量就无法减弱了。为了方便建模，我们用p(𝑥) = 𝑥 − 1 来表示这种音量减弱。校长出于节约依然是一个 ℝ𝕍ℙℂTh，我们可以用修改的 𝐺 (𝑥, 𝑦) 来表示这个系统：

1 1

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

𝐺′(𝑥, 𝑦) 𝑑=𝑒𝑓 (𝑦 > 3)( 2 𝜏.((𝑦 > 0)𝜏.𝐺′(𝑥, p(𝑦))|⌝(𝑦 > 0)𝜏.𝐺′(𝑥, 𝑦)) ⊕ ~~2~~ 𝜏.((𝑦 > 1)~~𝑎~~(𝑥).𝐺′(𝑥, 𝑦)))

𝐺′(𝑥, 𝑦) 在′ (″𝑦 > 3)ℰ3 的等价集 [𝐺′(𝑥, 𝑦)](𝑦>3)ℰ3 = [𝐺′(𝑥, 𝑦′)](𝑦′>3)ℰ3 = [~~𝑎~~(𝑥).𝐺′(𝑥, 𝑦″)](𝑦″>3)ℰ3 ，

其中𝐺𝑦,′(𝑦𝑥, 𝑦) 不一定相等。

的 (𝑦 > 3)ℰ3-tree 如图 [2–8](#_bookmark30)。

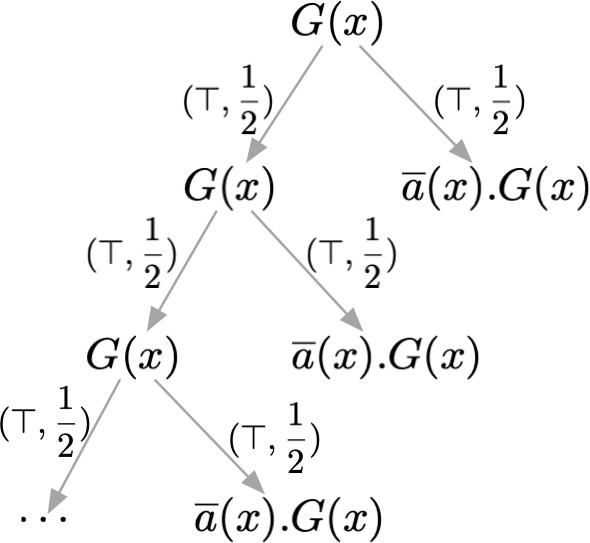


图 2–7

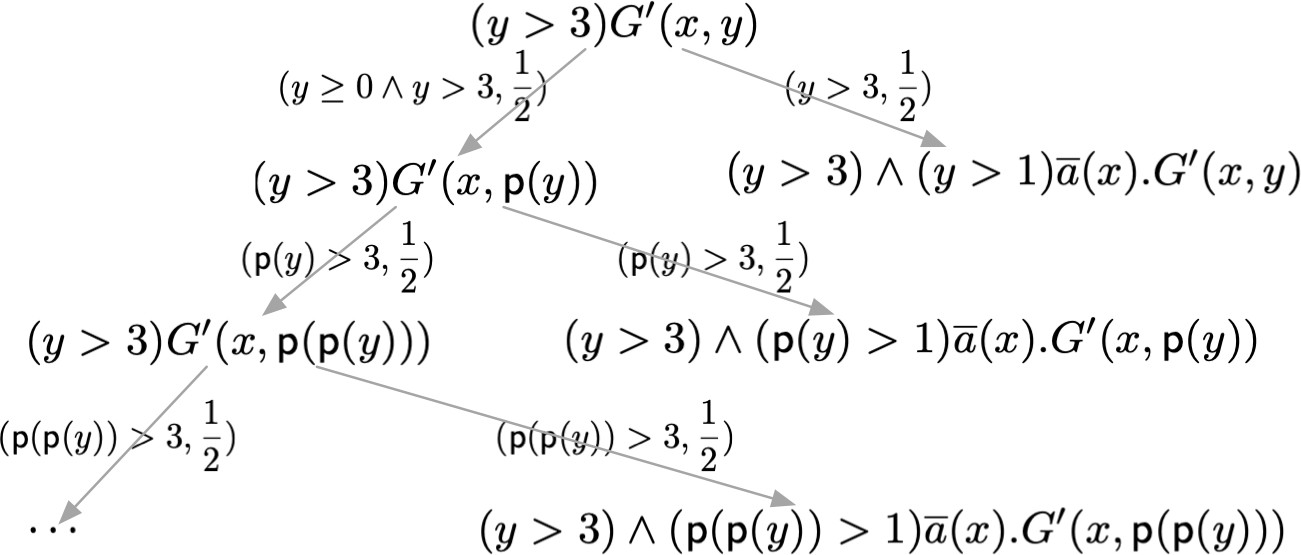


图 2–8

### 随机传值进程模型的符号互模拟

ℝ𝕍 ℙ定ℂ 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

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。Th

**定义 2.11 (条件 𝒍-迁移 (𝝋𝒍-transition))** 𝐴 ∈ 𝒯ℝ𝕍 ℙℂTh , ℬ ∈ 𝒯ℝ𝕍 ℙℂTh /𝜑′𝐴ℰ \{[𝐴]𝜑′ℰ }，其中 𝜑′ 是 存在 𝐿 →𝜑 𝐿′ ∈ ℬ, 𝑙 ≠ 𝜏，称为 𝜑′ 条件下 𝐴 到 ℬ 的 𝜑𝑙-迁移，记作 𝐴 ⇝𝜑′ℰ →𝜑 ℬ。

一个布尔𝑙 表达式，ℰ 是 𝒯ℝ𝕍 ℙℂTh 上的等价关系，若 𝐴 的条件等价树 𝑡𝜑′ℰ 的所有叶子结点 𝐿，𝑙

**定义 2.12 (条件 𝒒-迁移 (𝝋𝒒-transition))** 𝐴 ∈ 𝒯ℝ𝕍 ℙℂTh , ℬ𝐴 ∈ (𝒯ℝ𝕍 ℙℂTh /𝜑′ℰ)\{[𝐴]𝜑′ℰ }，其中 𝜑′

是一个布尔表达式，ℰ 是 𝒯ℝ𝕍 ℙℂTh 上的等价关系，对 𝑡𝜑′ℰ 的每一个叶子结点 𝐿 和布尔表达

∐ 𝑝 𝜏

式 𝜑，若 𝐿

⟶𝑖∈𝐼

𝑖 ∐𝑝𝑖∈𝜏𝐼 𝜓𝑖 ∐𝑖∈[𝑘] 𝐿𝑖, 𝑖 ∈ 𝐼, 𝐿𝑖 ∈ ℬ，且 Th ⊢ 𝜑 ⇒ 𝜓𝑖：

定义 P𝜑(𝐿 ∐⟶𝑖∈𝐼 𝑖 ∐𝑝𝑖∈𝜏𝐼 𝜓𝑖 ℬ) = ∑{𝑝𝑖 ∣ 𝐿 𝑝→𝑖𝜏𝜓𝑖 𝑝𝐿𝜏𝑖 ∈ ℬ ∧ 𝑖 ∈ 𝐼 ∧ Th ⊢ ∐𝜑 ⇒𝑝𝜓𝜏 𝑖}。

定义 P𝜑,𝜑′ℰ (𝐿 ∐⟶𝑖∈𝐼

𝑖 ∐𝑖∈𝐼 𝜓𝑖 ℬ) = P𝜑(𝐿 ∐⟶

𝑖 ∐𝑖∈𝐼 𝜓𝑖 ℬ)/(1 − P𝜑(𝐿

⟶𝑖∈𝐼

𝑖 ∐𝑖∈𝐼 𝜓𝑖 [𝐴]𝜑ℰ ))。

当P𝜑,𝜑′ℰ (𝐿 ∐⟶𝑖∈𝐼 𝑝𝑖𝜏∐𝑖∈𝐼 𝜓𝑖 ℬ) = 𝑞 时，称 𝜑′ 条件下𝐴 到ℬ 存在𝜑𝑞-迁移。写作 𝐴 ⇝𝜑′ℰ →𝑞 𝜑

ℬ

。

符号互模拟实质上是保证 𝒯ℝ𝕍 ℙℂTh 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)** ℰ 是一个 𝒯𝑅𝑉 𝑃 𝐶 上的二元对称关系，当 𝐴ℰ𝐵 且满足下列条件时， 称 是一个符号互模拟 (Symbolic Bisimulation)：

ℰ

𝜑𝑖ℰ

，使𝑎(̄ 得𝑡)

对 𝑖 ∈ 𝐼，Th ⊢ 𝜑𝑖 ⇒ 𝜓𝑖。

𝑎(𝑡𝑖) ′

* + - * 若[𝐿]𝐴 ⇝} 𝜑ℰ →𝜑 𝐿 ∉ [𝐴]𝜑ℰ ，则存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 ，和集合 {𝐵 ⇝𝜑𝑖ℰ → 𝜓𝑖 𝐿 ∈
* 若[𝐿]𝐴 ⇝} 𝜑ℰ 𝑎→(𝑥)𝜑 𝐿 ∉ [𝐴]𝜑ℰ ，则存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 ，和集合 {𝐵 ⇝𝜑𝑖ℰ 𝑎→(𝑥)𝜓𝑖 𝐿′ ∈

𝜑𝑖ℰ

* 若[𝒞]𝐴 ⇝} 𝜑ℰ →𝜑 𝒞 ∈ 𝒯 /ℰ, 𝒞 ≠ [𝐴]𝜑ℰ ，则存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 ，和集合 {𝐵 ⇝𝜑𝑖ℰ →𝜑𝑖

，使𝑞 得对 𝑖 ∈ 𝐼，Th ⊢ (𝜑𝑖 ⇒ 𝜓𝑖) ∧ (𝜑𝑖 ⇒ (𝑡 = 𝑡𝑖))。

𝑞𝑖

𝜑𝑖ℰ

，使得 𝑖 ∈ 𝐼，𝑞𝑖 = 𝑞。

如果 RCCS 的分支互模拟可以看作是用等价树模拟等价树，ℝ𝕍ℙℂTh 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 ∈ 𝒯ℝ𝕍 ℙℂTh ，ℰ 是 𝒯ℝ𝕍 ℙℂTh 上的等价关系，𝜑 是一个布尔表达 式，对于 的一个划分 𝑖 𝑖∈𝐼 ， 关于 的条件等价森林有且仅有 关于 𝑖 的条件等价树或条件等价森林， 。

𝑖 ∈ 𝐼

𝜑 {𝜑 } 𝐴 𝜑ℰ 𝐴 𝜑 ℰ

在证明𝒯ℝ𝕍 ℙℂTh 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 ℝ𝕍 ℙℂTh 项的条件 -迁移和条件 -迁移时，我们可以首先构建该项的静态迁移树。

𝒯 𝑙 𝑞

**例 2.6** 假设学校负责看管设备的老师发现例 [2.5](#_bookmark28)中的下课铃系统其实只有在 𝑦 ≥ 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) 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

时用自己的电脑通过

例 [2.4](#_bookmark27)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？

𝐺″(𝑥来, 𝑦表) 示= 这( 1个𝜏.(系(𝑦统>。0)𝜏.𝐺″(𝑥, p(𝑦))|⌝(𝑦 > 0)𝜏.𝐺″(𝑥, 𝑦)) ⊕ 1 𝜏.((𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 <

𝐺″(𝑥此, 𝑦时)

，看管设备的老师和例 [2.5](#_bookmark28)中的下课铃系统依旧是一个 ℝ𝕍ℙℂTh，我们可以用

5)𝑎(𝑥).𝐺″(𝑥, 𝑦)) 2 ″ 2

我们只需要证明 𝐺(𝑥) 𝐺

。

与

(𝑥, 𝑦) 符号互模拟即可。

**证明** 构造等价集

𝒮 = {(𝐺(𝑥), 𝐺″(𝑥, 𝑦″)), ″

(𝐺(𝑥), (𝑦 > 0)𝜏.𝐺 (𝑥, p(𝑦″))|⌝(𝑦 > 0)𝜏.𝐺 (𝑥, 𝑦))″,

(𝑎(𝑥).𝐺(″𝑥), (𝑦 > 1)𝑎(𝑥).𝐺 (𝑥, 𝑦)|(𝑦 < 5)𝑎(𝑥).𝐺 (𝑥, 𝑦)),

(𝐺(𝑡), 𝐺 (𝑡, 𝑦))|𝑡 ∈ 𝑉 }

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。

* + - * ~~𝑎~~(𝑥).𝐺(𝑥) 和 (𝑦 > 1)~~𝑎~~(𝑥).𝐺 (𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺 (𝑥, 𝑦) 关于 (𝑥 = 𝑡)𝒮 的等价树只有一

个𝕍 ℙ根ℂ节点，由于不涉及概率，这里实际可以使用 𝕍ℙℂTh 的符号互模拟证明，但由于

Th 是 ℝ𝕍ℙℂTh 的特例，此处依然可以使用 ℝ𝑎𝕍(𝑡ℙ) ℂTh 的符号互模拟证明。

(𝑥 = 𝑡) = ((𝑦 > 1) ∧ (𝑥 = 𝑡)) ∨ ((𝑦 ≤ 1) ∧ (𝑥 = 𝑡))

**–** 对于 𝑥 的每一个赋值 𝑡 ∈ 𝑉 ，~~𝑎~~(𝑥).𝐺(𝑥) ⇝(𝑥=𝑡)𝒮 ⟶(𝑥=𝑡) 𝐺(𝑡)。我们可以得到 (𝑥 = 𝑡)

的一个划分： 。

这时存在集合

{(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦) ⇝(𝑥=𝑡)∧(𝑦>1)𝒮 𝑎→(𝑡′)(𝑦>1) 𝐺″(𝑡′, 𝑦),

(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦) ⇝(𝑥=𝑡)∧(𝑦≤1)𝒮 𝑎→(𝑡′)(𝑦<5) 𝐺″(𝑡′, 𝑦)}

可1) 以⇒模(𝑡拟= 上𝑡′)述操″作，其中 Th ⊢ (𝑥 = 1) ∧ (𝑦 ≤ 1) ⇒ (𝑦 < 5) 且 Th ⊢ (𝑥 = 𝑡) ∧ (𝑦 ≤

，𝐺 (𝑡, 𝑦) ∈ [𝐺(𝑡)](𝑥=𝑡)∧(𝑦≤1)𝒮 。(𝑥 = 𝑡) ∧ (𝑦 > 1) 相同。

**–** 对 𝑥 的每一个赋值 𝑡 ∈ 𝑉 ，

(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦) ⇝(𝑥=𝑡)∧(𝑦>1)𝒮 𝑎→(𝑡)(𝑥=𝑡)∧(𝑦>1) 𝐺″(𝑡, 𝑦)

， 𝑎(𝑡′) ′

𝑡存) ∧在(𝑦集>合1){)𝑎⇒(𝑥)(.𝑥𝐺(=𝑥)𝑡)⇝∧(𝑥=𝑡)(𝑦>1)𝒮 ⟶(𝑥=𝑡) 𝐺(𝑡 )} 可以模′拟上述操作″，其中 Th ⊢ ((𝑥 =

Th ⊢ ((𝑥 = 𝑡) ∧ (𝑦 > 1)) ⇒ 𝑡 = 𝑡 ，𝐺(𝑡) ∈ [𝐺

𝑥 𝑡 ∈ 𝑉

对 的每一个赋值 ，

(𝑡, 𝑦)](𝑥=𝑡)∧(𝑦>1)𝒮 。

(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦) ⇝(𝑥=𝑡)∧(𝑦<5)𝒮 𝑎→(𝑡)(𝑥=𝑡)∧(𝑦<5) 𝐺″(𝑡, 𝑦)

， 𝑎(𝑡′) ′

𝑡存)∧在(𝑦集<合5){)𝑎⇒(𝑥)(.𝑥𝐺=(𝑥𝑡))⇝∧ (𝑥=𝑡)(𝑦<5)𝒮 ⟶(𝑥=𝑡) 𝐺(𝑡 )} 可以模′ 拟上述操作″，其中 Th ⊢ ((𝑥 =

″ Th ⊢ ((𝑥 = 𝑡)″∧(𝑦 < 5)) ⇒ 𝑡 = 𝑡 ，𝐺(𝑡) ∈ [𝐺

(𝑡, 𝑦)](𝑥=𝑡)∧(𝑦<5)𝒮 。

* + - * (𝐺(𝑡), (𝑦 > 0)𝜏.𝐺 (𝑥, p(𝑦))|⌝(𝑦 > 0)𝜏.𝐺

(𝑥, 𝑦)) 的这一对等价关系也是常规的𝕍ℙℂTh 证

的静态迁移树如图[2–9](#_bookmark34)，

(𝑥, 𝑦) 的静态迁移树如图 [2–10](#_bookmark35)。

图 2–9

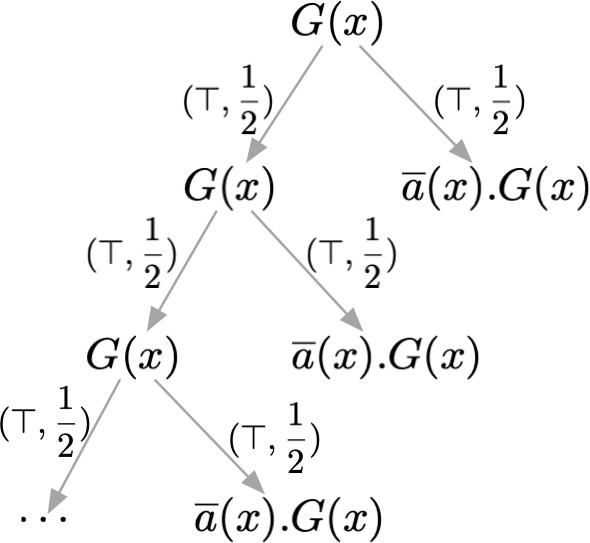
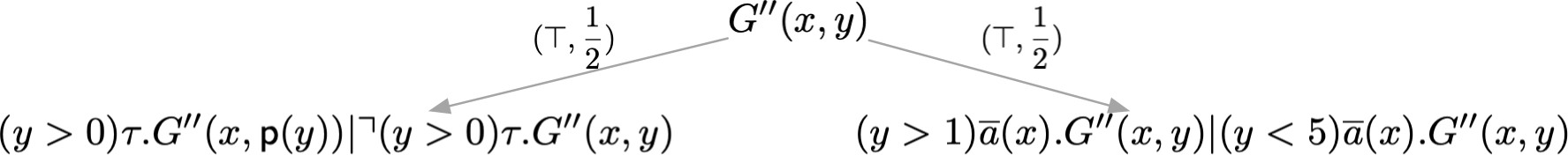


图 2–10



对于 𝐺(𝑥) ⇝⊤𝒮 →⊤ [𝑎(𝑥).𝐺(𝑥)]⊤𝒮 ，存在集合

{𝐺″(𝑥, 𝑦) ⇝⊤𝒮 →12 ⊤ [(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)]⊤𝒮 }

12

其中Th ⊢ (⊤ ⇒ ⊤) 且 12 = 21 ，(~~𝑎~~(𝑥).𝐺(𝑥), (𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)) ∈ 𝒮。

对于 𝐺″(𝑥, 𝑦) ⇝⊤𝒮 →12 ⊤ [(𝑦 > 1)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)|(𝑦 < 5)~~𝑎~~(𝑥).𝐺″(𝑥, 𝑦)]⊤𝒮 ，我们可以对称地 用集合 {𝐺(𝑥) ⇝⊤𝒮 →⊤ [𝑎(𝑥).𝐺(𝑥)]⊤𝒮 } 模拟。

12

12

对于𝐺″(𝑥, ″𝑦) ⇝⊤𝒮 →⊤ [(𝑦 > 0)𝜏″.𝐺″(𝑥, p(𝑦))|⌝(𝑦 > 0)𝜏″.𝐺″(𝑥, 𝑦)]⊤𝒮 ，由″ 于等价关系具有

传递性，𝐺 (𝑥, 𝑦) ∈ [(𝑦 > 0)𝜏.𝐺 (𝑥, p(𝑦))|⌝(𝑦 > 0)𝜏.𝐺 (𝑥, 𝑦)]⊤𝒮 = [𝐺 (𝑥, 𝑦)]⊤𝒮 。

因此我们可以得出校长的判断是正确的。 □

例 [2.6](#_bookmark33)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。

## 随机传值进程模型的等价性

ℝ𝕍ℙℂ

如 [2.3.3](#_bookmark31)节中所述，符号互模拟描述了 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](#_bookmark32)，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𝜆 模拟的传递性等价于证明若ℰ 是一个符号互模拟，且𝐴ℰ𝐵, 𝐵ℰ𝐶，则𝐴ℰ𝐶。

* 若𝐴 ⇝𝜑ℰ →𝜑 𝒞 ∈ 𝒯𝑅𝑉 𝑃𝜆 𝐶 /𝜑ℰ, 𝒞 ≠ [𝐴]𝜑ℰ ，由于 𝐴ℰ𝐵，根据定义存在 𝜑 的划分{𝜑𝑖}𝑖∈𝐼

和集合 𝑆 = {𝐵 ⇝𝜑𝑖ℰ →𝜓𝑖 [𝒞]𝜑𝑖ℰ |Th ⊢ 𝜑𝑖 ⇒𝜆 𝜓𝑖}𝑖∈𝐼 。对于 𝐵 关于 𝜑𝑖ℰ 的条件等价树𝜆

𝑡𝐵 𝐿 ⊢ 𝜑 ⇒ 𝜓 𝐿 → 𝐿′ ∈ [𝒞] 𝐿 → 𝐿′ ∈ [𝒞]

𝜑𝑖ℰ 上的节点 ，因为 Th 𝜆 𝑖 𝑖，若 𝜓𝑖 𝜑𝑖ℰ ，则 𝜑𝑖 𝜑𝑖ℰ ，因𝜆

此存在集合 𝑆′ = {𝐵 ⇝𝜑𝑖ℰ →𝜑𝑖 [𝒞]𝜑𝑖ℰ }𝑖∈𝐼 ，根据定义 [2.13](#_bookmark32)，𝑆′ 也可以作为 𝐴 ⇝𝜑ℰ →𝜑

𝒞 ∈ 𝒯𝑅𝑉 𝑃 𝐶 /𝜑ℰ, 𝒞 ≠ [𝐴]𝜑ℰ 的模拟。𝜆

𝐵ℰ𝐶 𝐵 ⇝ → [𝒞] 𝜑 {𝜑 }

由于 ，对于每𝜆一个 𝜑𝑖ℰ 𝜑𝑖 𝜑𝑖ℰ ，根据定义存在 𝑖 的划分 𝑖,𝑗 𝑗∈𝐽 和集𝜆

合[𝒞]𝑆𝑖 =} {𝐶 ⇝𝜑𝑖,𝑗 ℰ →𝜓𝑖,𝑗 [𝒞]𝜑𝑖,𝑗 ℰ |Th ⊢ 𝜑𝑖,𝑗 ⇒ 𝜓𝑖,𝑗 }𝑗∈𝐽 ，和集合 𝑆𝑖 = {𝐶 ⇝𝜑𝑖,𝑗 ℰ →𝜑𝑖,𝑗

𝜑𝑖,𝑗 ℰ

𝑗∈𝐽 。 ′ ′

进而，存在𝜑 的划分𝐶𝑜𝑛 = ⋃𝑖∈𝐼 {𝜑𝑖,𝑗 }𝑗∈𝐽 和集合𝑆

𝐴

拟 。对称𝑞 的证明是完全一致的。

* 若𝐴 ⇝𝜑ℰ →𝜑 𝒞 ∈𝑞𝑖 𝒯𝑅𝑉 𝑃 𝐶 /𝜑ℰ, 𝒞 ≠ [𝐴]𝜑ℰ , 𝑞 ∈ (0, 1]，由于 𝐴ℰ𝐵，存在 𝜑 的𝑞 划分{𝜑𝑖}𝑖∈𝐼

= ⋃𝑖∈𝐼 𝑆𝑖 ，使得𝐶 可以符号模

和集合{𝐵 ⇝𝜑𝑖ℰ →𝜓𝑖 [𝒞]𝜑𝑖ℰ |Th ⊢ 𝜑𝑖 ⇒ 𝜓𝑖 ∧ 𝑞 = 𝑞𝑖}，因此存在{𝐵 ⇝𝜑𝑖ℰ →𝑖 𝜑𝑖 [𝒞]𝜑𝑖ℰ } 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](#_bookmark38)。

**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 ℝ𝕍ℙℂTh 的观察等价性，一种比较直接的想法是将 Th 的符号互模拟的全集定义为 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。

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

**引理 2.12** 如果每一个𝒯ℝ𝕍 ℙℂTh 上的关系ℰ𝑖 都是符号互模拟关系，那么 (⋃𝑖∈𝐼 ℰ𝑖)∗ 是符号互 模拟关系。

**证明** 令ℰ = (⋃𝑖∈𝐼 ℰ𝑖)∗。给定 𝐴0ℰ𝑖1 𝐴1ℰ𝑖2 𝐴2 ⋯ 𝐴𝑘−1ℰ𝑖𝑘 𝐴𝑘，由于 ℰ = (⋃𝜆𝑖∈𝐼 ℰ𝑖)∗，我们可以得

到𝐴1ℰ𝐴𝑘，若ℰ 是一个符号互模拟关系，则对于条𝜆 件𝑙-迁移：𝐴0 ⇝𝜑ℰ →𝜑 𝒞 ∈ 𝒯ℝ𝕍 ℙℂTh , 𝒞 ≠

[𝐴0]𝜑ℰ ，存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 {𝐴𝑘 ⇝𝜑𝑖ℰ →𝜓𝑖 [𝒞]𝜑𝑖ℰ |Th ⊢ 𝜑𝑖 ⇒ 𝜓𝑖} 可模拟该操作。 我们可以通过归纳的证明 0 1， 1 可以模拟 0; 1 2， 2 可以模拟 1；…；最终证

明 𝐴𝑘 可以模拟 𝐴0。𝜆

𝐴 ℰ𝐴 𝐴 𝐴 𝐴 ℰ𝐴 𝐴

对于 𝜆𝐴0 ⇝𝜑ℰ →𝜑 𝒞 ∈ 𝒯ℝ𝕍 ℙℂTh , 𝒞 ≠ [𝐴0]𝜑ℰ ，我们需要给出 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合

{𝐴1 ⇝𝜑𝑖ℰ →𝜓𝑖 [𝒞]𝜑𝑖𝜆ℰ } 来模拟 𝐴0 的动作。

考虑 𝐴0 ⇝𝜑ℰ →𝜑 𝒞 ∈ 𝒯ℝ𝕍 ℙℂTh , 𝒞 ≠ [𝐴0]𝜑ℰ ，由等𝑖1 价集的定义，我们可以得出𝜑ℰ 上的等

树，我们可以递归的构建出 𝐴1 关于 𝜑ℰ 的条′ 件等价′ 森林。 ′

价集 𝒞 可以被划分成不相交的 𝜑ℰ𝑖 上的等价集 {𝒞𝑗 }。假设 𝑡𝐴 是 𝐴0 关于 𝜑ℰ 的条件等价1 0

* 𝑡𝐴0 **的根节点** 𝐴0 **只有一个儿子节点** 𝐴0**。**若 𝐴0 ∈ [𝐴0]𝜑ℰ𝑖1 ，我们将以 𝜑𝐴0 作为下次递归

的根节点，构建𝜑𝐴1 条件等价森林（这样递归是因为在 𝐴′1 的𝜑ℰ 条件等价森林中的

𝜑 𝐴 𝜑𝐴 𝐴 ∉ [𝐴 ] 𝐴 ℰ 𝐴

所有节点的动作都满足条件 ， 1 和 1 是等价的𝜏）。若 0 0 𝜑ℰ𝑖1 ，根据 0 𝑖1 1，

我们可以得到 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 {𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜓𝑖 [𝒞]𝜑𝑖ℰ𝑖1 |𝜏Th ⊢ 𝜑𝑖 ⇒ 𝜓𝑖}𝑖∈𝐼 。

根𝐴 据⇝推论→

[2.10](#_bookmark37)的证明经验， 我们其实

[𝒞 ∈](#_bookmark37) 𝒯 , 𝒞 ≠ [𝐴 ] }

可以用 {𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜑𝑖 [𝒞]𝜑𝑖ℰ𝑖1 }𝑖∈𝐼 模拟

0 𝜑ℰ 𝜑

𝜆ℝ𝕍 ℙℂTh 0 𝜑ℰ

𝑖∈𝐼 ，为了方便证明，后续的证明中我们将直

接使用 {𝐴1 𝜏⇝𝜑𝑖ℰ𝑖1 →𝜑𝑖 [𝒞]𝜑𝑖ℰ𝑖1 }𝑖∈𝐼 的形式。 𝑖

{𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜑𝑖 𝑖 [𝒞]𝜑𝑖ℰ𝑖1 }𝑖∈𝐼 实际构建𝜏出了一′ 个条件′ 等价树的集合 {𝑡𝐴1 }𝑖∈𝐼 ，对于每个

条件等价树 𝑡𝐴1 的叶子节点 𝐵，𝐵 →𝜑𝑖 𝐵 ∈ [𝐴0]𝜑𝑖ℰ𝑖1 ，我们根据 𝜑𝑖𝐴0 构建每一个

𝜑 𝐵′ 𝑖 ′

𝑖 关于𝜑𝑖ℰ 的等价森林。此时将 𝐵 所在的𝜑𝑖ℰ 等价树𝑡𝐴1 复制并与每个𝐵 的𝜑𝑖ℰ

的等价森林中的等价树相连，我们可以得到 𝐴1 关于 𝜑ℰ 的∐条件等价森林。

* 𝑡𝐴𝐴𝑗0 **的根节点** 𝐴0 **有** ℎ **个儿子节点** 𝐴10, ⋯ , 𝐴ℎ0 **。**根据定义𝐴0

0 的边被𝑗 标记为 (𝜓, 𝑝𝑗 )，其中 𝜑 ⇒ 𝜓。1

𝑖∈→[ℎ]𝑝𝑗 𝜏 𝜓 ∐𝑗∈[ℎ] 𝐴𝑗0，𝐴0 到

𝐴 ∈ [𝐴 ] 𝜑𝐴 𝜑𝐴

若所有 𝑗0 0 𝜑ℰ𝑖1 ，我们将根据1 0 构建 1𝑞的条件等价森林。

若存在 0 0 𝜑ℰ𝑖1 ，不妨设为 0，则 0 𝑞 𝜑ℰ𝑖1 1𝜑 0 𝜑ℰ𝑖1 ，根据 0 𝑖1 1，我们可

𝐴 ∉ [𝐴 ] 𝐴 𝐴 ⇝ → [𝐴1] 𝐴 ℰ 𝐴

以得到 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 {𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜑𝑖 [𝐴0]𝜑𝑖ℰ𝑖1 }𝑖∈𝐼 ，这其∐中关系𝑝到𝜏 𝐴1 关于

[𝜑𝐴𝑖ℰ1]𝑖1 的条) =件𝑞等价树 𝑡𝑖1，根据条1

件 𝑞-迁移的定′义，存在 P𝜑𝑖,𝜑𝑖ℰ𝑖1 (𝑁

𝑖′∈⟶[ℎ′]

𝑖′

𝜑𝑖 𝑁′ ∈

0 𝜑𝑖ℰ𝑖1

，我们根据 𝜑𝑖𝐴0 构𝜆建每个 𝜑𝑖𝑁

关于 𝜑𝑖ℰ 的条件等价森林。

* 𝑡𝐴0 **的根节**𝜆**点** 𝐴0′ **可以作** 𝐴0 →𝜑 𝐿′**。**根据定义存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 上述递归的构建可以通过图 [2–11](#_bookmark40)更加形象的表述。图 [2–11](#_bookmark40)中，第一层描述了根节点只

有一个儿子节点情况的构建方法，对于每一个 𝑖 ∈ 𝐼，右侧的′条件等价树都𝜏会复制′一份，全

{𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜑𝑖 [𝐿 ]𝜑𝑖ℰ𝑖1 }。

部的条件等价树构成𝜑ℰ 条件等价森林，作 {𝐴1 ⇝𝜑𝑖ℰ𝑖1 →𝜏 𝜑𝑖 [𝐴0]𝜑𝑖ℰ𝑖1 } 对𝐴0 →𝜑ℰ𝑖1 𝐴0 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𝐴0 ⇝𝜑ℰ𝑖1 →𝑞 𝜑，右侧

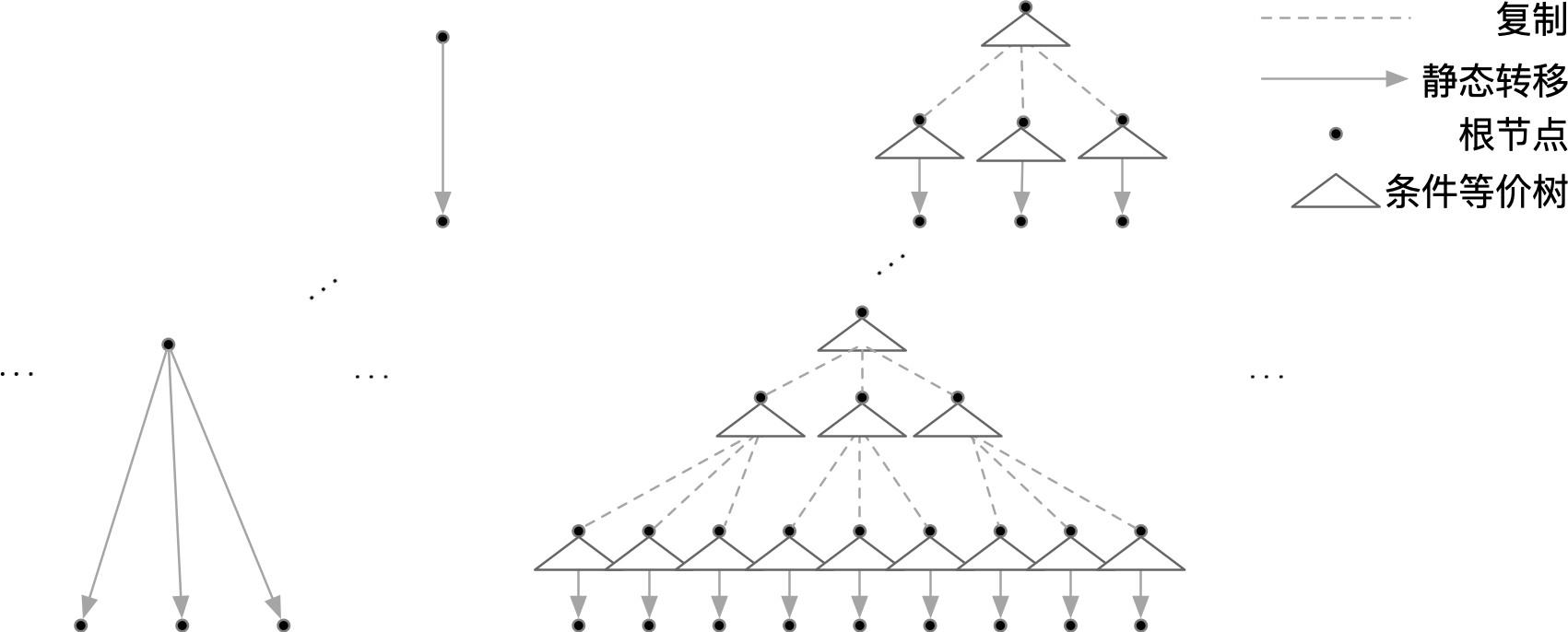


图 2–11 递归构建图解

的条件等价树会复制一份，在对每一个 𝐴0 ⇝𝜑ℰ𝑖1 →𝑞 𝜑 [𝐴01]𝜑ℰ𝑖1 模拟时，对模拟该动作的集合

{𝐴 ⇝ →𝑞 [𝐴1] }

1 𝜑𝑖ℰ𝑖1 𝜑𝑖 0 𝜑𝑖ℰ𝑖1 𝑖∈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𝜆构建方式，我们最终会构建出 𝐴1 关于 𝜑ℰ 的条件等价树或条件等价森林，来

模拟𝐴0 ⇝𝜑ℰ →𝜑 𝒞 ∈ 𝒯ℝ𝕍 ℙℂTh , 𝒞 ≠ [𝐴0]𝜑ℰ 。归纳地，我们可以构建出𝐴2 对𝐴1 关于𝜑ℰ 的条

件等价树或条𝜆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 𝐴𝑘

𝐴 ⇝ → 𝒞 ∈ 𝒯 , 𝒞 ≠ [𝐴 ]

模拟 0 𝜑ℰ 𝜑 ℝ𝕍ℙℂTh 𝑞 0 𝜑ℰ 的条件等价森林。

对于条件 𝑞-迁移：𝐴0 ⇝𝜑ℰ →𝜑 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 ℝ𝕍ℙℂTh 上的观察等价性，但这样定义仍会存在问题：考虑 ℝ𝕍 ℙℂTh ，若 在 下的条件等价树是无限延伸的，即没有叶子节点， 例如：

𝐴 ∈ 𝒯 𝐴 𝜑ℰ

𝐴(𝑥) 𝑑=𝑒𝑓 (~~𝑎~~(𝑥).0|𝐵(𝜖)|𝐶)\{𝑎, 𝑏, 𝑐}

𝐵 𝑑=𝑒𝑓 𝑎(𝑥).𝐵(𝑥)

𝐶 𝑑=𝑒𝑓 𝑏(𝑥).𝐶(𝑥)

𝐶(𝑥) 𝑑=𝑒𝑓 ~~𝑎~~(𝑥).𝐶

若定义等价关系 [𝐴(𝑥)]ℰ = [𝐵(𝑥)|𝐶|𝐷]ℰ = [𝐵|𝐶(𝑥)|𝐷]ℰ = [𝐵|𝑑𝐶𝑒𝑓|𝐷(𝑥)]ℰ ，这时 𝐴(𝑥) 的 ⊤ℰ 条

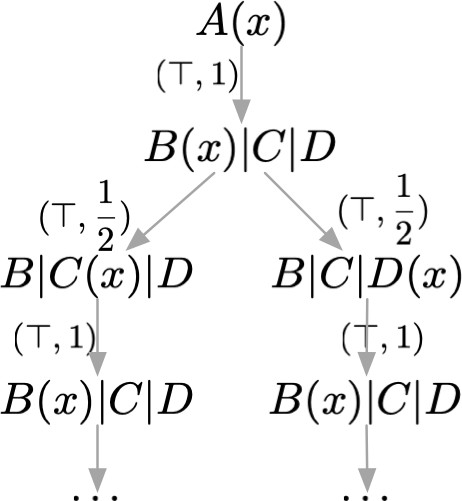
𝐵(𝑥) 𝑑=𝑒

21 𝜏.~~𝑏~~(𝑥).𝐵 ⊕ 1~~2~~ 𝜏.~~𝑐~~(𝑥).𝐵

𝐷 𝑑=𝑒𝑓 𝑐(𝑥).𝐷(𝑥)

𝐷(𝑥) 𝑑=𝑒𝑓 ~~𝑎~~(𝑥).𝐷

件等价树如图 [2–12](#_bookmark41)。在观察等价性的角度，我们认为 𝐸(𝑥) = 𝜏.𝐸(𝑥) 与 𝐴(𝑥) 应该是观察

图 2–12 𝐴(𝑥) 的 ⊤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 codivergent 的相关概念来解决这个问题。

**定义 2.15** 对于 𝐴 ∈ 𝒯ℝ𝕍 ℙℂTh ，若 𝐴 在 𝜑ℰ 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 为分支 上第 条边的标记中的概率部分，

。

lim𝑗→∞ ∏𝑖=0 𝜋(𝑖)。

𝜋(𝑖) 𝜋 𝑖

𝜋(𝑖) ∈ (0, 1]

定义条件等价树的概率 P(𝑡) = ∑𝑘{P(𝜋)|𝜋是𝑡的一条分支}。

定义这𝑖=𝑗条分支的概率 P(𝜋) = ∏{𝜋(𝑖)|𝑖 ∈ [|𝜋|]}，若这条分支是无穷的，则 P(𝜋) =

定义 𝑡 的有限长分支的概率 P (𝑡) = lim𝑓𝑘→∞ P (𝑡)。

定义 𝑡 的前 𝑘 层分支的概率为𝑓P (𝑡) = ∑{P(𝜋)𝑘|𝜋是𝑡的一条分支且|𝜋| ≤ 𝑘}。

当条件等价树𝑡 的有限长分支的概率P

𝑡

理解为 没有有限长的分支。

(𝑡) = 0 时，𝑡 称为发散的条件等价树，我们可以

通过定义 [2.15](#_bookmark42)，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 (codivergent) 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** 若ℰ 是𝒯ℝ𝕍 ℙℂTh 上的等价关系，ℰ 是一个共发散的等价关系当且仅当：当对于所 有的布尔表达式 ，对条件等价集 ℝ𝕍 ℙℂTh ， 中的每一个元素关于 的条件等价森林中的每一个棵树都是发散的等价树；或 中的每一个元素关于 的条件等价森林中的每一个棵树都不是是发散的等价树。

𝒞 𝜑ℰ

𝜑 𝒞 ∈ 𝒯 /𝜑ℰ 𝒞 𝜑ℰ

**定义 2.17** 𝒯ℝ𝕍 ℙℂTh 上的观察等价性 =ℝ𝕍 ℙℂTh 定义为 𝒯ℝ𝕍 ℙℂTh 上的共发散的符号互模拟关系 的全集。

由于 =ℝ𝕍 ℙℂTh 仍然是一个符号互模拟关系，因此我们可以得到定理 [2.13](#_bookmark43)。

**定理 2.13** =ℝ𝕍 ℙℂTh 是一个等价关系。

是一个代数系统，∼ 是载体 𝑆 上的等价关系，若 ∼ 在 𝐴 上的所有运算下都是可保持的，则称 为代数系统 上的同余关系[[29](#_bookmark98)]。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 ℝ𝕍 ℙℂTh 是一个同余关系。

=

∼ 𝐴

**定理 2.14** =ℝ𝕍 ℙℂTh 具有同余性。

𝐴**证明**= =ℝ𝕍 ℙ𝐵ℂTh 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](#_bookmark22)易知

≡ ℝ𝕍ℙℂTh

𝑑则𝑒𝑓 𝜑𝐴 =ℝ𝕍 ℙℂTh 𝜑𝐵。为方便证明，以下 =ℝ𝕍 ℙℂTh 简记为 =，绝对等价性记为

，定义记为 = 。

对于本地化操作子（Localization），对于𝜆 𝜑= 𝜑

𝜆 ′

𝐴 = 𝐵, (𝑎)𝐴 ⇝ → (𝑎)𝐴

易知有 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 {(𝑎)𝐵 ⇝𝜑𝑖=→𝜑𝑖 [(𝑎)𝐴]𝜑𝑑=𝑒𝑓}。

∉ [(𝑎)𝐴]𝜑=, 𝑎 ∉ 𝜆，

} 对于′ 并发操作∗子（Compositio′n），考虑等价关系 ℛ

= {(𝐴|𝐶, 𝐵|𝐷)|(𝐴, 𝐵) ∈=, (𝐶, 𝐷) ∈𝜆 =

𝒞，∈令𝒯ℛ

= (ℛ/𝜑∪ℛ=′,)𝒞，≠我[们(𝐴证|𝐶明)] ℛ

是一个符号互模拟关系。对于条件 𝑙-迁移：(𝐴|𝐶) ⇝𝜑ℛ′ →𝜑

ℝ𝕍ℙℂTh

′

𝜑ℛ′ ，我们可以用引理 [2.12](#_bookmark39)中的方′ 法递归的构造 (𝐵|𝐶) 的 𝜑ℰ

的

条件等价森林来模拟 (𝐴|𝐶)(𝜓的,1)动作′ ，令 𝑡(𝐴|𝐶) 为 (𝐴|𝐶[)](#_bookmark39) 关于 𝜑𝜏 ℛ 条件等价树。

* 𝑡(𝐴|𝐶) 上的边：𝐴|𝐶 → 𝐴 |𝐶, Th ⊢ 𝜑 ⇒ 𝜓 是由于 𝐴 →𝜑 𝐴 。

∐(𝐵𝑖|∈𝐷𝐼) 𝑖

若 𝐴′𝐴∈ [=𝐴]𝐴𝜑=，那么 (𝐴′|𝐶) ∈ [(𝐴|𝐶)]𝜑ℛ′ ≡ [(𝐵|𝐷)]𝜑ℛ′ 。实际上′，对于 𝐴 ∐𝑖∈→𝐼 𝑝𝑖𝜏𝜑

′，𝐴𝑖|𝐶 ∈ [𝐵|𝐷]𝜑ℛ′ 。我们可以根据 𝜑(𝐴0|𝐶) 的 𝜑ℛ

条件等价树构建

′ 的 𝜑ℛ

条件等价森林。 𝜏 ′

𝜏 ′

若𝐴 ∉ [𝐴]𝜑=，那么根据定义存在 {𝐵 ⇝𝜑′𝑖=→𝜑𝑖 𝐵𝑖 ∈ [𝐴 ]𝜑𝑖=′}𝑖∈𝐼 ，模拟 𝐴 ⇝𝜑=→𝜑 𝐴 ，

其中 {𝜑𝑖}𝑖∈𝐼 是 𝜑 的(划𝜓,1分) 。′我们′ 根据 𝜑𝑖𝐴 构建 𝜑𝑖𝐵𝑖 的 𝜑𝑎𝑖ℛ(𝑡) 条件′等价𝑎森(𝑥)林。 ′

* 𝑡𝜓(𝐴′|𝜓𝐶″) 上的边：𝐴|𝐶 → 𝐴 |𝐶 , Th ⊢ 𝜑 ⇒ 𝜓 是由于 𝐴 ⟶𝜓′ 𝐴 , 𝐶

⟶𝜓″ 𝐶 , 𝜓 =

。 𝑎(𝑡) ′ 𝑎(𝑡) ′

根据 ， 𝜑= 𝜑𝑎(𝑥) 可以由 𝜑𝑖= 𝜑𝑖 𝑖 𝜑𝑖= 𝑖∈𝐼 模拟，其𝑎(𝑥)

中 {𝜑𝑖}𝑖∈𝐼 是 𝜑 的划分。𝐶 ⇝𝜑=⟶𝜑 𝐶′ 可以由 {𝐷 ⇝𝜑𝑗=⟶𝜑𝑗 𝐷𝑗 ∈ [𝐶′]𝜑𝑗=}𝑗∈𝐽 模

𝐴 = 𝐵, 𝐶 = 𝐷 𝐴 ⇝ ⟶ 𝐴 {𝐵 ⇝ ⟶ 𝐵 ∈ [𝐴 ] }

𝜑对𝜑于(所𝐴′有|𝐶的′) 𝑖 ∈ 𝐼, 𝑗 ∈′ 𝐽 , Th ⊢ 𝜑𝑖𝜑𝑗 ⇏ ⊥，(𝐵𝑖 ∣ 𝐷𝑗 ) ∈ [𝐴′

拟，其中 {𝜑𝑗 }𝑗∈𝐽 是 𝜑 的划分。

∣′ 𝐶′]𝜑𝑖𝜑𝑗ℛ′ ，我们根据

𝑖 𝑗

的 𝜑𝑖𝜑𝑗(ℛ𝜓,𝑞)条件等价树构建 𝜑𝑖𝜑𝑗 (𝐵𝑖|𝐷𝑗 ) 的 𝜑𝑖𝜑𝑗 ℛ

条∐ 件𝑝等𝜏 价森林。

* 𝑡(𝐴|𝐶) 上的边：𝐴|𝐶 (𝜓→,𝑝𝑖)𝐴′|𝐶, Th ⊢ 𝜑 ⇒ 𝜓, 𝑞 ∈ (0, 1)，则存在𝐴

⟶𝑖∈𝐼

𝑖 𝜑 ∐𝑖∈𝐼 𝐴𝑖，对

所有的 𝑖 ∈ 𝐼，𝐴|𝐶 → 𝐴𝑖|𝐶, Th ⊢ 𝜑 ⇒ 𝜓。 ′

若(𝐵对|𝐷所)

有的 𝑖′ ∈ 𝐼, 𝐴𝑖 ∈ [𝐴]𝜑=，我们可以根据 (𝐴𝑖|𝐶) 关于 𝜑ℛ 的条件等价树构建

的 𝜑ℛ 条件等价森林。 𝑞0

若[𝐴存] 在𝐴𝑖 ∉ [𝐴]𝜑𝑞=，不妨设为 𝐴0，若有𝑞 𝐴+𝑞1 ∉ [𝐴]𝜑= ∧ 𝐴1 ∈ [𝐴0]𝜑=，此时有 𝐴 ⇝𝜑=→𝜑

0 𝜑=，𝐴 ⇝𝜑=→1 𝜑 [𝐴0]𝜑=，𝐴|𝐶 ⇝𝜑= ⟶0 1𝜑 [𝐴1|𝐶]𝑞𝜑0=。

𝑞0

𝐴 = 𝐵 𝜑 {𝜑 } {𝐵 ⇝ → 𝐵 ∈ [𝐴 ] } 𝐴 ⇝ →

[根𝐴据] ，存在 的划分 𝑖 𝑖∈𝐼 和集合 𝜑𝑖𝑞=1 𝜑𝑖 𝑖 0 𝜑𝑖= 模拟 𝜑= 𝑞1

[𝐴0]𝜑=。存在 𝜑 的划分 {𝜑𝑗 }𝑗∈𝐽 和集合 {𝐵 ⇝𝜑𝑗=→𝜑𝑗 𝐵𝑗 ∈ [𝐴1]𝜑𝑗=𝑞} +模𝑞 拟 𝐴 ⇝𝜑=→𝜑

[𝐴1|𝐶𝜑=]。对所有的𝑖 ∈ 𝐼, 𝑗 ∈ 𝐽 , Th ⊢ 𝜑𝑖𝜑𝑗 ⇏ ⊥′，我们有(𝐵|𝐷) ⇝𝜑𝑖𝜑𝑗 = ⟶0 ′ 1𝜑𝑖𝜑𝑗 (𝐵′|𝐷) ∈′

0 𝜑𝑖𝜑𝑗 =，我们根据 𝜑𝑖𝜑𝑗 (𝐴0|𝐶) 的𝜑𝑖𝜑𝑗 ℛ 条件等价树构建𝜑𝑖𝜑𝑗 (𝐵 |𝐷) 的𝜑𝑖𝜑𝑗 ℛ

条件等价森𝜆 林。′

𝜆 ′

* 若 (𝐴|𝐶) →𝜆𝜑 (𝐴 |𝐶), 𝜆 ≠ 𝜏，则 𝐴 →𝜑 𝐴 ，根据定义存在 𝜑 的划分 {𝜑𝑖}𝑖∈𝐼 和集合 对于条件 𝑞-迁移：(𝐴|𝐶) ⇝𝜑=→𝜑 𝒞 的证明方法是相同的。

{𝐵|𝐷 ⇝𝜑𝑖=→𝜑𝑖 [(𝐴′|𝐶)]𝑞𝜑𝑖=}。

(𝐴, 𝐵) ∈= 𝐴 𝜑 = 𝐵

在上述证明中，若 ，且 关于 的条件等价树是发散的，根据定义可知 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同余性。

□

## 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 Calcu- lus）和它的一种实现 Th，我们分析了 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 Th 的优势和表达能力，我

们在 𝕍ℙℂTh 的基础上进行概率扩展获得随机传值进程模型。

𝕍ℙℂ

𝕍ℙℂ 𝕍ℙℂ

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

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

其次，为了给出 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转移扩展为了概率下的状态保持的静态转移树。同时，我们引入了 𝕋 𝕙 中的等价集、符号互模拟的概念，由于 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。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 ℝ𝕍 ℙℂTh 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 ℝ𝕍 ℙℂTh 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

𝒯

𝒯

ℝ𝕍ℙℂ

𝕍ℙℂ

ℝ等𝕍价ℙ树ℂ 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性，并证明了 Th 观察等价性是一个同余关系。

ℝ𝕍ℙℂ

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 ℝ𝕍ℙℂTh 的观察等价

# 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。

## Gossip-Style Membership 协议

### 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[[30](#_bookmark99)]。

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 协议主要用在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[[31](#_bookmark100)]。

Gossip 协议中，节点之间的通信方式有三种：

#### Push Gossip:

𝑘

消息的发送者周期性的随机选择 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 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。

𝑘

#### Pull Gossip:

𝑘

每个节点周期性的向 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。

#### Push/Pull Gossip:

𝑛

在超过 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 [cite]。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 2 节点之前使用Push Gossip，在消息传播 2 之后使用Pull Gossip。

𝑛 𝑛

### 组成员协议

组成员协议 (Group Membership Protocol，后文简称 Membership 协议)[[32](#_bookmark101)] 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：

* + - * 检测失效的节点
      * 传播消息，如：告知其他节点失效信息

常见的Membership 协议有Heartbeating Protocol[[33](#_bookmark102)], Gossip-Style Membership Protocol[[34](#_bookmark103)], SWIM Failure Detector Protocol[[18](#_bookmark87)] 等。

## Gossip-Style Membership 的实现

ℝ𝕍ℙℂ

我们使用 Th 来实现 Gossip-Style Membership 协议[[34](#_bookmark103)]，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。

### Gossip 协议的实现

𝑂( 𝑁)

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 log

ℝ𝕍ℙℂ

时间复杂度的消息传播，本节我们使用 Th 来建模以Push Gossip 作为通信机制的Gos- sip 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 ~~𝑡𝑖𝑚𝑒~~ 端口用于向计时

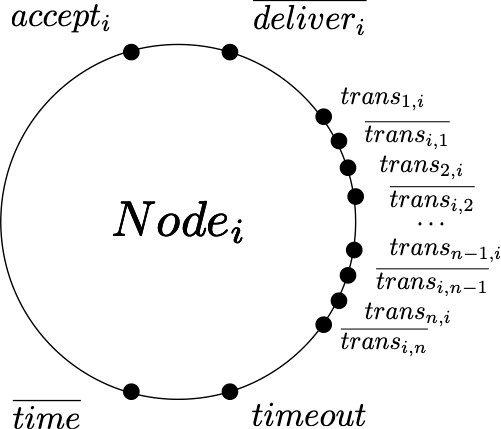
器设置计时开始，𝑡𝑖𝑚𝑒𝑜𝑢𝑡 端口用于接收计时器传来的timeout 信号。由于节点本身不产生和存储消息，我们需要使用 端口接收 P2P 系统外界消息，使用 向 P2P 系统外界传递节点在 P2P 系统中从其他节点收到的消息。由于现实网络是复杂的，我们将每一对

𝑎𝑐𝑐𝑒𝑝𝑡 ~~𝑑𝑒𝑙𝑖𝑣𝑒𝑟~~

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 𝑁𝑜𝑑𝑒𝑖 向𝑁𝑜𝑑𝑒 传播消息时，𝑗

消息会在𝑡𝑟𝑎𝑛𝑠𝑖,𝑗 通道中，从 𝑁𝑜𝑑𝑒𝑖 节点的𝑡𝑟𝑎𝑛𝑠𝑖,𝑗 端口发出，从 𝑁𝑜𝑑𝑒𝑗 节点的𝑡𝑟𝑎𝑛𝑠𝑖,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](#_bookmark51)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](#_bookmark51)。



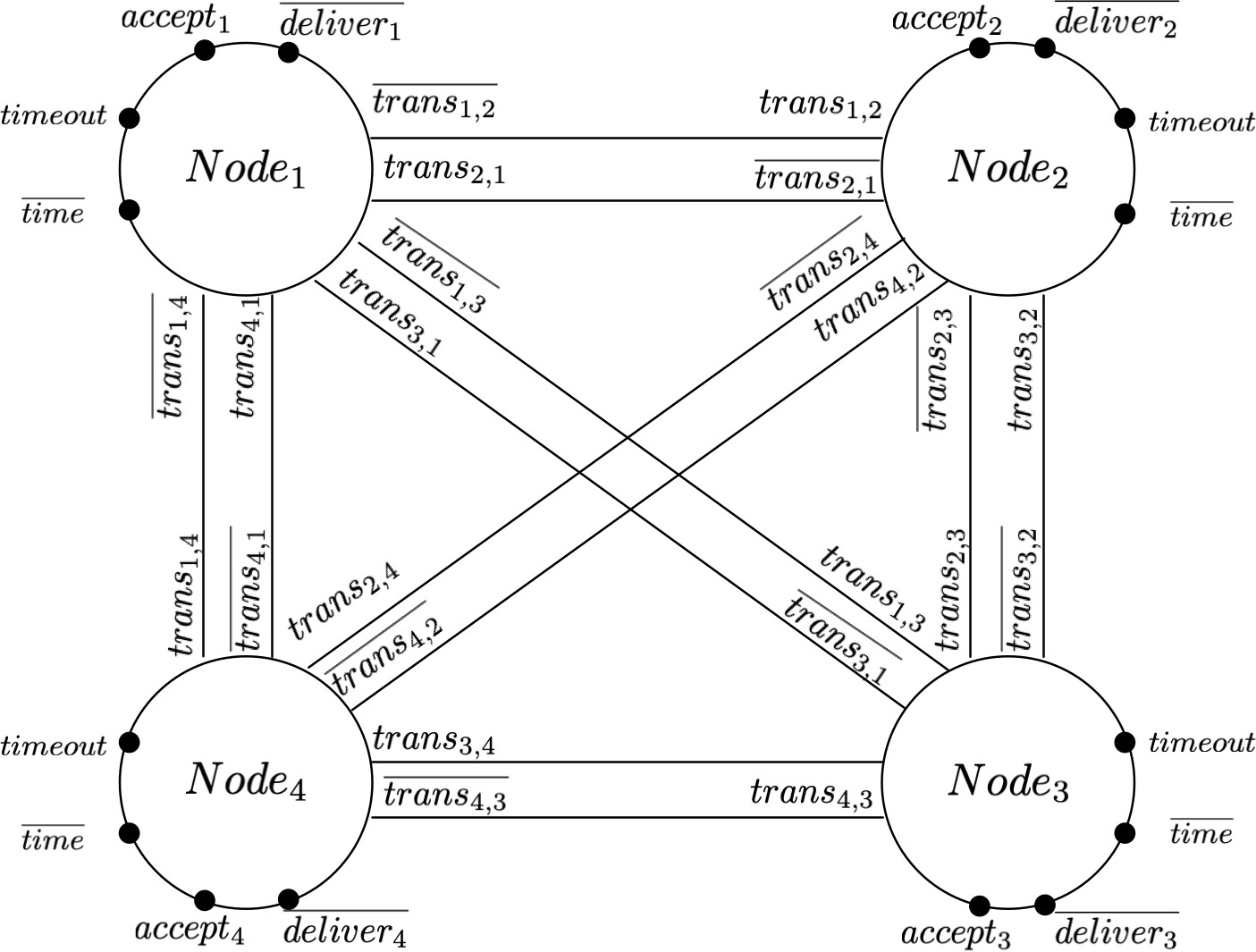
图 3–1 Gossip 节点示意图

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

|  |  |
| --- | --- |
| 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 b 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。

𝑁𝑜𝑑𝑒 𝑑=𝑒𝑓 𝑎𝑐𝑐𝑒𝑝𝑡 (𝑥).𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒 (𝑥)

𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁 𝑖 𝑖

𝑜𝑑𝑒𝑖(𝑥) 𝑑=𝑒𝑓 𝑑𝑒𝑙𝑖𝑣𝑒𝑟𝑖(𝑥).𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑖(𝑥)

𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑖(𝑥) 𝑑=𝑒𝑓 𝑡𝑖𝑚𝑒𝑜𝑢𝑡.(𝑝𝑒𝑟𝑚⨁∈ 𝑝𝑝𝑒𝑟𝑚𝜏.𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖,𝑝𝑒𝑟𝑚(𝑥))

PERM𝑖

𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒 (𝑥) 𝑑=𝑒𝑓 ~~𝑡𝑟𝑎𝑛𝑠~~ (𝑥). ⋯ 𝑡𝑟𝑎𝑛𝑠 (𝑥)~~.𝑡~~𝑖𝑚𝑒.𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒 (𝑥)

𝑖,𝑝𝑒𝑟𝑚

𝑑𝑒𝑓

𝑖,𝑝𝑒𝑟𝑚1

𝑖,𝑝𝑒𝑟𝑚𝑏 𝑖

𝑈 𝑛𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑖 = 𝑗∈∑N/{𝑖} 𝑡𝑟𝑎𝑛𝑠𝑗,𝑖(𝑥).𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖(𝑥)

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 𝑑=𝑒𝑓 (𝑁𝑜𝑑𝑒1 ∣ 𝑈 𝑛𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒2 ∣ ⋯ ∣ 𝑈 𝑛𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑛)

\{𝑡𝑟𝑎𝑛𝑠𝑖,𝑗 ∣ 𝑖 ∈ N ∧ 𝑗 ∈ N ∧ 𝑖 ≠ 𝑗} ∪ {𝑡𝑖𝑚𝑒, 𝑡𝑖𝑚𝑒𝑜𝑢𝑡}

在这个 P2P 系统中共有 𝑛 个节点，标号为 N = {1, 2, ⋯ , 𝑛}，每一个拥有 Gossip Message 的节点周期性的向 k k 个其他节点发送 Gossip Message。其中 PERM𝑖 为 N 中任选k 个元素的全排列， perm 为向一个特定的排列 发送 Gossip Message 的概率，为了方

𝑝 𝑝𝑒𝑟𝑚

( ≤ 𝑛 − 1) /{𝑖}

便建模，我1 们假(𝑛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

我们可以看到节点𝑁𝑜𝑑𝑒𝑖 接受到外界消息𝑥 后，迁移为 𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖(𝑥) 的状态，向

𝑝𝑝𝑒𝑟𝑚 = 𝐴k𝑛−1 = (𝑛−k−1)1!)! 为固定值，当 k = 2 时，𝑝𝑝𝑒𝑟𝑚 = (𝑛−1)1(𝑛−2) 。

外界递送这个消息，进而迁移为 𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒)𝑖(𝑥) 的状态，可以在接受到 𝑡𝑖𝑚𝑒𝑜𝑢𝑡 信号

时以概率 𝑝perm 的概率选定 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 𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖,𝑝𝑒𝑟𝑚(𝑥)，向 𝑝𝑒𝑟𝑚 中的节点发送Gossip Message，发送完成后

回到𝐼𝑛𝑓𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑖(𝑥) 的状态。在一个𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 中，还有状态为𝐼𝑛𝑓𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒𝑖

𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒 (𝑥)

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 𝑖

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](#_bookmark72)]。
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[[35](#_bookmark104)]，Membership 协议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。

### Gossip 协议的等价性

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](#_bookmark50)节中我们使用 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](#_bookmark50)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。

ℝ𝕍ℙℂ

**定义 3.1** 单消息源的多播规约定义如下：

𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N 𝑑=𝑒𝑓 𝑎𝑐𝑐𝑒𝑝𝑡1(𝑥).𝑑𝑒𝑙𝑖𝑣𝑒𝑟1(𝑥).𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,{1}(𝑥)

𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN(𝑥) 𝑑=𝑒𝑓 𝑑𝑒𝑙𝑖𝑣𝑒𝑟𝑖(𝑥).𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN∪{𝑖}(𝑥),

𝑖 ∈ N − KNOWN ∧ |N| ≠ |KNOWN|

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 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。

| |

𝑑𝑒𝑙𝑖𝑣𝑒𝑟

**定**𝑈 𝑛**义**𝐼𝑛**3**𝑓**.2**𝑒𝑐𝑡𝑖由𝑜𝑢𝑠于𝑁𝑜𝐺𝑑𝑒𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 中的节点可能处于：𝐼𝑛𝑓𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒，𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒，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

三种状态（我们将 𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒 的状态合并进了 𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒），

N 的记法，其中 Composition 算子符合交换律。

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥) 𝑎

= (𝐼⏞𝑛⏞𝑓⏞⏞⏞𝑒⏞𝑐⏞⏞𝑡⏞𝑖⏞⏞𝑜⏞𝑢⏞⏞𝑠⏞⏞𝑁⏞⏞⏞𝑜⏞⏞𝑑⏞⏞𝑒⏞⏞(⏞𝑥⏞⏞)⏞⏞∣⏞⋯ ∣⏞⏞𝐼⏞⏞𝑛⏞⏞𝑓⏞⏞⏞𝑒⏞𝑐⏞⏞𝑡⏞𝑖⏞𝑜⏞⏞𝑢⏞⏞𝑠⏞𝑁⏞⏞⏞⏞𝑜⏞𝑑⏞⏞⏞𝑒⏞(⏞⏞𝑥)

∣𝐷⏞𝑒⏞⏞𝑙⏞𝑖⏞𝑣⏞⏞𝑒⏞⏞𝑟⏞𝑖⏞𝑛⏞⏞𝑔⏞⏞𝑁⏞⏞⏞𝑜⏞⏞𝑑⏞⏞𝑒⏞⏞(⏞𝑥⏞⏞)⏞⏞∣⏞⋯𝑏

∣⏞𝑈 𝑛⏞⏞𝐼⏞⏞𝑛⏞⏞𝑓⏞⏞⏞𝑒⏞𝑐⏞⏞𝑡⏞𝑖⏞𝑜⏞⏞𝑢⏞⏞𝑠⏞𝑁⏞⏞⏞⏞𝑜⏞𝑑⏞⏞⏞𝑒⏞⏞∣⏞⋯𝑐

∣⏞⏞𝐷⏞⏞⏞𝑒⏞⏞𝑙⏞𝑖⏞𝑣⏞⏞𝑒⏞𝑟⏞⏞𝑖⏞𝑛⏞⏞𝑔⏞⏞𝑁⏞⏞⏞𝑜⏞⏞𝑑⏞⏞𝑒⏞⏞(⏞𝑥)

∣⏞⏞𝑈⏞⏞⏞𝑛⏞⏞𝐼⏞⏞𝑛⏞⏞𝑓⏞⏞𝑒⏞⏞𝑐⏞⏞𝑡⏞𝑖⏞𝑜⏞⏞𝑢⏞𝑠⏞⏞𝑁⏞⏞⏞𝑜⏞⏞𝑑⏞𝑒)

\{𝑡𝑟𝑎𝑛𝑠𝑖,𝑗 ∣ 𝑖 ∈ N ∧ 𝑗 ∈ N ∧ 𝑖 ≠ 𝑗} ∪ {𝑡𝑖𝑚𝑒, 𝑡𝑖𝑚𝑒𝑜𝑢𝑡}

𝑎 + 𝑏 + 𝑐 = 𝑛 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚

其𝐼𝑛𝑓中𝑒，𝑐𝑡𝑖𝐺𝑜𝑢𝑜𝑠𝑠𝑁𝑠𝑖𝑝𝑜𝑑𝑆𝑒𝑦(𝑠𝑥𝑡)𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥) 表示在系统的 𝑛 = |N| 个节点中， 有 𝑎 个处于

𝑈 𝑛𝐼𝑛𝑓 𝑒𝑐𝑡𝑖𝑜𝑢𝑠𝑁𝑜𝑑𝑒

的状态， 有 𝑏 个处于 𝐷𝑒𝑙𝑖𝑣𝑒𝑟𝑖𝑛𝑔𝑁𝑜𝑑𝑒(𝑥) 的状态， 有 𝑐 个处于的状态，且满足 。很显然，初始的 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 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 和 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N 中的每一个𝑑节𝑒𝑙𝑖𝑣点𝑒𝑟能(𝑡 )且仅能执行𝑑𝑒𝑙一𝑖𝑣𝑒次𝑟 (𝑡𝑑)𝑒𝑙𝑖𝑣𝑒𝑟 操作，且

𝑑𝑒𝑙𝑖𝑣𝑒𝑟 的顺序不影响功能，所以我们可以规定 𝐴

⟶𝑖

1 ⊤ 𝐵 和𝐶

⟶𝑗

2 ⊤ 𝐵 是互模拟的

当且仅当 𝑡1 = 𝑡2，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** 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N =ℝ𝕍 ℙℂTh 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N，Th 是可判定的逻辑。

N

构造等价集

**证**𝐺𝑜**明**𝑠𝑠𝑖𝑝𝑆我𝑦𝑠们𝑡𝑒𝑚可 以= 通过 构𝑀建𝑢𝑙等𝑡𝑖𝑐𝑎价𝑠𝑡集𝑆𝑝，𝑒𝑐 并证明等价集是一个符号互模拟关系来证明

ℝ𝕍ℙℂTh N。

𝒮 = {(𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N, 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N), (𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛,0,0)(𝑥), 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,N(𝑥))}

∪ {(𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥), 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN(𝑥)) ∣ |KNOWN| = 𝑎 ≠ 𝑛}

我们来依次证明 𝒮 中的每一对等价关系为符号互模拟关系:

* + - * 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛,0,0)(𝑥) 和𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,N(𝑥) 的⊤𝒮 条件等价树分别只有一个根节点，

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚 (𝑥) =

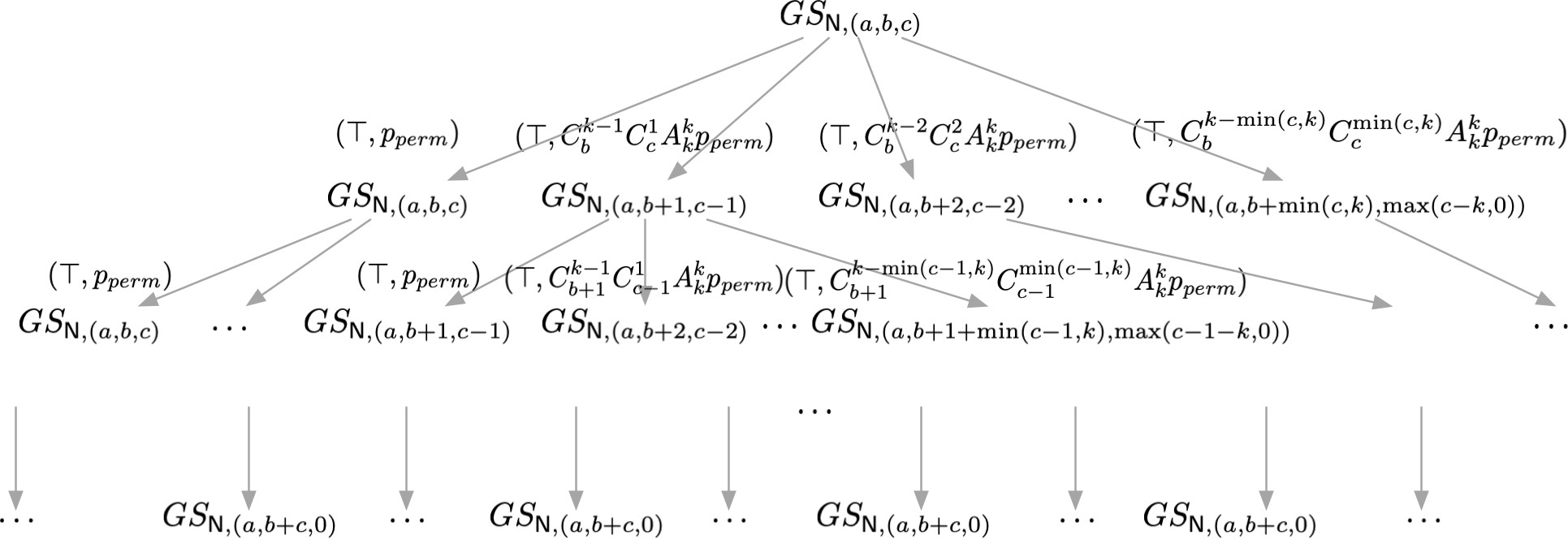
且𝑀不𝑢𝑙𝑡能𝑖𝑐𝑎转𝑠𝑡移𝑖𝑛至𝑔 其(他𝑥)等=价0 集，他们的互模拟是显然的，实际上 N,(𝑛,0,0)

N,N 。

* + - * 对于 (𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥), 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN(𝑥)) ∣ |KNOWN| = 𝑎 ≠ 𝑛:

𝑎 < 𝑛−1 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚 (𝑥) ⊤𝒮 𝑡 𝑡

**– Case 1: 。** N,(𝑎,𝑏,𝑐) 的 条件等价树 如图 [3–3](#_bookmark53)所示。当 上

图 3–3 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥) 关于 ⊤𝒮 的条件等价树 𝑡

的节点进入𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(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 ,(𝑎,𝑏+𝑐,0) 为 的叶子结点。很显然，对 上的每一个非

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚 (𝑥) 𝑡 𝑡

N𝑝′

叶结点，都会经过𝑁𝑜𝑑𝑒𝑗 →𝑗 ⊤ 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑑𝑡𝑒𝑒𝑙𝑖𝑚𝑣𝑒N𝑟(,(𝑥𝑎),𝑏+𝑐,0)(𝑥), 𝑝𝑗′ = 𝑝𝑗1 𝑝𝑗2 ⋯ 𝑝𝑗𝑚 < 1。对𝑡

的叶子结点有𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏+𝑐,0)(𝑥) ⟶ ⊤ 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎+1,𝑏+𝑐−1,0)(𝑥)。即

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥) ⇝⊤𝒮 𝑑𝑒𝑙⟶𝑖𝑣𝑒𝑟(𝑥)⊤ [𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎+1,𝑏′,𝑐′)(𝑥)]⊤𝒮

其中 𝑏′ + 𝑐′ = 𝑏 + 𝑐 − 1。我们可以用

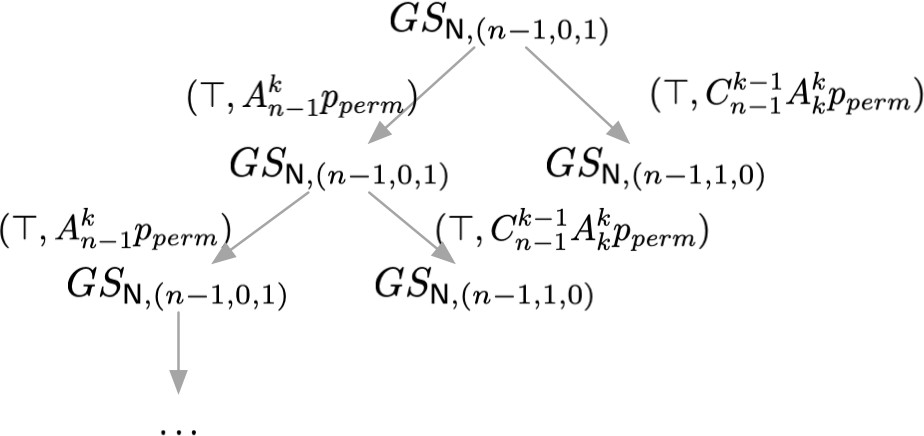
𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN ⇝⊤𝒮 𝑑𝑒𝑙⟶𝑖𝑣𝑒𝑟(𝑥)⊤ 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN′

来模拟上述状态迁移。对称的模拟依然成立。

**。**

**– Case 2:** 𝑎 = 𝑛 − 1

∗ **Case 2.1:** 𝑏 = 1。𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛−1,1,0)(𝑥) 的⊤𝒮 条件等价树𝑡 只有一个根节点， 对于 (𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛−1,1,0)(𝑥) ⇝⊤𝒮 𝑑𝑒𝑙⟶𝑖𝑣𝑒𝑟(𝑥)⊤ [𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛,0,0)(𝑥)]⊤𝒮 ，我们 可以用 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN ⇝⊤𝒮 𝑑𝑒𝑙⟶𝑖𝑣𝑒𝑟(𝑥)⊤ 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,N 来模拟。

图 3–4 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑎,𝑏,𝑐)(𝑥) 关于 ⊤𝒮 的条件等价树 𝑡

𝑏 = 0 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚 (𝑥) ⊤𝒮 𝑡

∗ **Case 2.1:** 。 N,(𝑛−1,0,1) 的 条件等价树 如图 [3–4](#_bookmark54)所𝑑𝑒示𝑙𝑖𝑣𝑒。𝑟(𝑥)

N, 来模拟。N

𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛−1,1,0) 为𝑡 的叶节点，且有(𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛−1,0,1)(𝑥) ⇝⊤𝑑𝒮𝑒𝑙𝑖𝑣⟶𝑒𝑟(𝑥) ⊤

[𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(𝑛,0,0)(𝑥)]⊤𝒮 。我们可以用 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,KNOWN

* + - * 对 (𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N, 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N)，我们可以用

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 𝑎𝑐𝑐𝑒𝑝𝑡(𝑥⟶).~~𝑑𝑒𝑙𝑖𝑣𝑒𝑟(𝑥)~~⊤ 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N,(1,0,0)(𝑥)

𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑆𝑝𝑒𝑐N 𝑎𝑐𝑐𝑒𝑝𝑡(𝑥⟶)~~.𝑑𝑒𝑙𝑖𝑣𝑒𝑟(𝑥)~~⊤ 𝑀𝑢𝑙𝑡𝑖𝑐𝑎𝑠𝑡𝑖𝑛𝑔N,{1}

⇝⊤𝒮 ⟶ ⊤

来相互模拟。 □

### Gossip-Style Membership 的实现

* + - 1. Gossip Style Membership Protocol

Renesse 等人提出了一种基于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 𝑓 𝑎𝑖𝑙 时间没有更新，则认为这一项对应的节点失效。

𝑇

𝑘

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 个节点发送自己的 Membership List。

### 对 GossipSystem 的调整

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Mem- bership 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 𝑓 𝑎𝑖𝑙 来定义一个节点失效的概率，同时经过一个静态迁移 ，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。

𝑝 𝜏

𝑎𝑐𝑐𝑒𝑝𝑡, 𝑑𝑒𝑙𝑖𝑣𝑒𝑟

在 Gossip System 节点的基础上，将原本对外暴露的

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝

通道用于连接

，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](#_bookmark57) 所示。

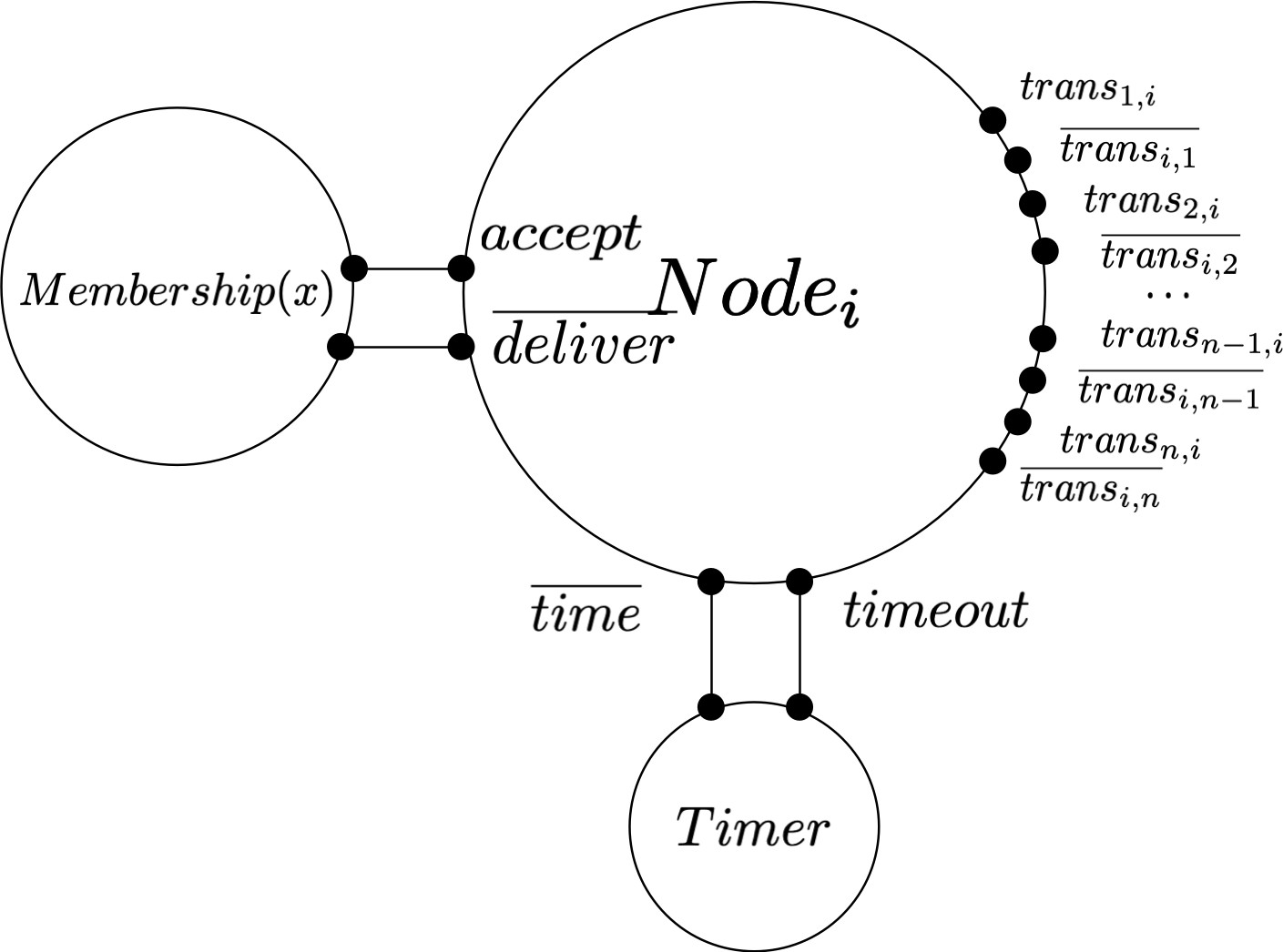


图 3–5 **Gossip-Style Membership Protocol** 节点示意图

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

𝑑𝑒𝑓

𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑖 = 𝑝𝑓 𝑎𝑖𝑙𝜏.𝐵𝑎𝑑𝑁𝑜𝑑𝑒𝑖 ⊕ (1 − 𝑝𝑓 𝑎𝑖𝑙)𝜏.𝑁𝑜𝑑𝑒𝑖

𝑑𝑒𝑓

𝐵𝑎𝑑𝑁𝑜𝑑𝑒𝑖 = 𝜏.𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑖

𝑁𝑜𝑑𝑒𝑖 𝑑=𝑒𝑓 𝑡𝑖𝑚𝑒𝑜𝑢𝑡.𝑎𝑐𝑐𝑒𝑝𝑡(𝑥).(𝑝𝑒𝑟𝑚⨁∈ 𝑝𝑝𝑒𝑟𝑚𝜏.𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖,𝑝𝑒𝑟𝑚(𝑥))

+ 𝑗∈∑N/{𝑖}

𝑡𝑟𝑎𝑛𝑠𝑗,𝑖(𝑥).𝑑𝑒𝑙𝑖𝑣𝑒𝑟(𝑥PE)R.𝐹M𝑖𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑖

𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖,𝑝𝑒𝑟𝑚(𝑥) 𝑑=𝑒𝑓 𝑡𝑟𝑎𝑛𝑠𝑖,𝑝𝑒𝑟𝑚1 (𝑥). ⋯ 𝑡𝑟𝑎𝑛𝑠𝑖,𝑝𝑒𝑟𝑚𝑏 (𝑥)~~.𝑡~~𝑖𝑚𝑒.𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑖

𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N 𝑑=𝑒𝑓 (𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒1 ∣ 𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒2 ∣ ⋯ ∣ 𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑛)

\{𝑡𝑟𝑎𝑛𝑠𝑖,𝑗 ∣ 𝑖 ∈ N ∧ 𝑗 ∈ N ∧ 𝑖 ≠ 𝑗} ∪ {𝑡𝑖𝑚𝑒, 𝑡𝑖𝑚𝑒𝑜𝑢𝑡, 𝑎𝑐𝑐𝑒𝑝𝑡, 𝑑𝑒𝑙𝑖𝑣𝑒𝑟}

其𝐵𝑎中𝑑𝑁，𝑜每𝑑𝑒一个节点定义为 𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒，每一时刻它会有 𝑝𝑓 𝑎𝑖𝑙 的概率成为失效节点

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

，和 1 − 𝑝𝑓 𝑎𝑖𝑙 的概率成为可发送列表和接受列表的正常节点 𝑁𝑜𝑑𝑒。在 𝑁𝑜𝑑𝑒 状

List，以概率 𝑝perm 的概率选定 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 𝑖,𝑝𝑒𝑟𝑚 ，向 中的节点发送Membership List，发送完

成后回到𝐹𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒 的状态。一个有𝑛 = |N| 个节点的基于Gossip-Style Membership 协议

𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒 (𝑥) 𝑝𝑒𝑟𝑚

的 P2P 系统定义为 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N，它由 𝑛 个 𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒 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](#_bookmark58)中的内容。其中Address 表示系统中其他节

表 3–2 Membership List

|  |  |  |
| --- | --- | --- |
| Address | HeartBeat | LocalTime |
| 1  2  3 | 10120  10103  10098 | 66  62  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 𝑓 𝑎𝑖𝑙，我们认为这一节点失效。

𝑇

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 ， 应为一个 (Address, HeartBeat, LocalTime) 的三元组的数组，一

个记录本地地址的 𝐴𝑑𝑑𝑟𝐼𝑛𝑓 𝑜(𝑎𝑑𝑑𝑟𝑒𝑠𝑠) （地址应在加入网络时由 DNS 分配，此处不考虑

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐿𝑖𝑠𝑡(𝑋)

它的分配过程）。定义 𝑥𝑖[𝐴𝑑𝑑𝑟𝑒𝑠𝑠] 为取 Address 的值的操作子，𝑥𝑖[𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡] 同理。𝑛 为

MembershipList 的最大容量。

𝑛

我们定义的 Membership 系统的实现如图 [3–6](#_bookmark59)所示，可以看到， 我们使用 个

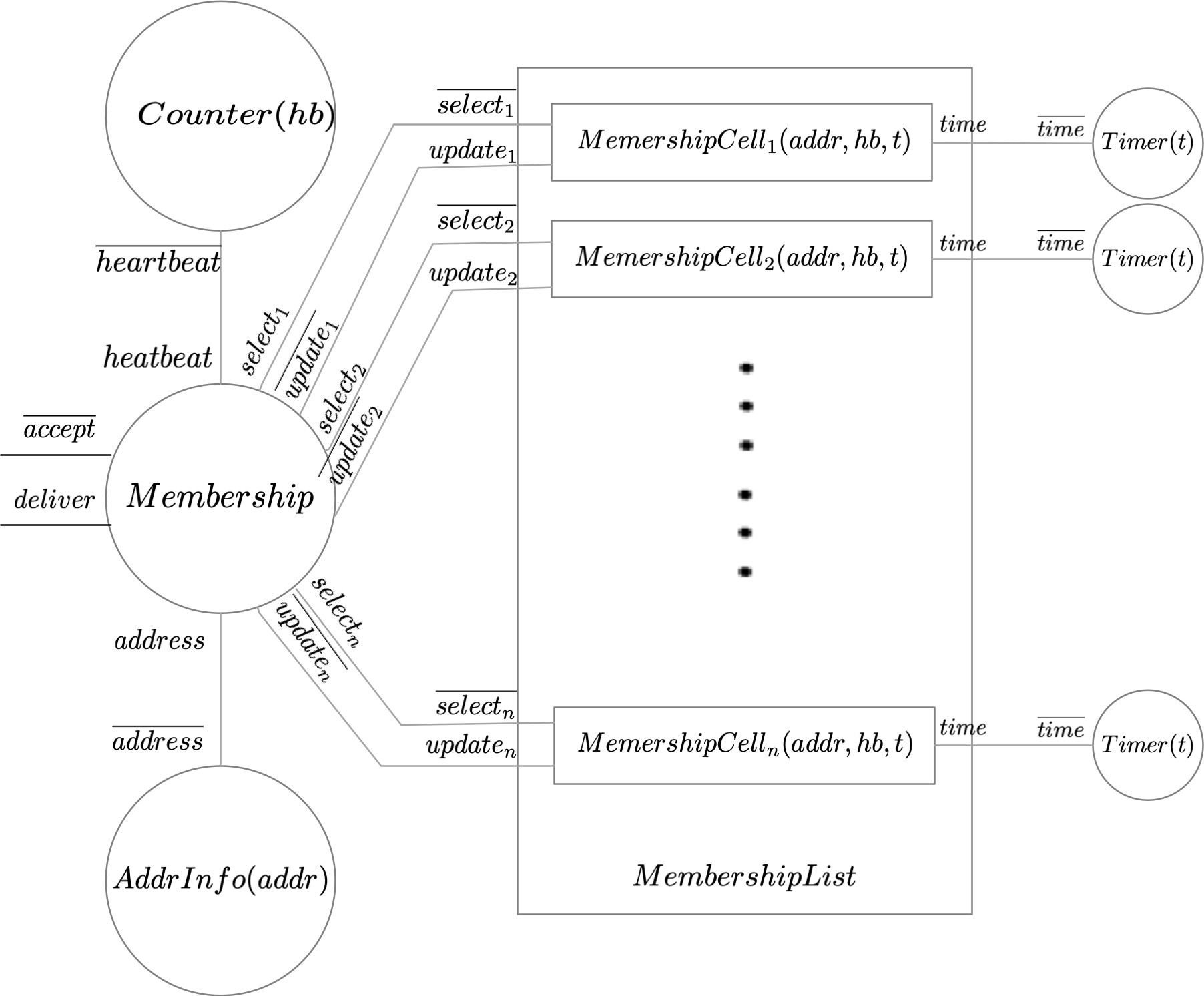


图 3–6 **Membership** 示意图

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙 来存储 Membership List 中的每一项，𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 可以通过 𝑠𝑒𝑙𝑒𝑐𝑡𝑖 通道获取 𝑖 中的数据，也可以通过 𝑖 通道更新 𝑖 中的数据， 每一个 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。 的定义如下：

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙 𝑢𝑝𝑑𝑎𝑡𝑒 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙𝑖(𝑎𝑑𝑑𝑟, ℎ𝑏, 𝑡) 𝑑=𝑒𝑓

𝑠𝑒𝑙𝑒𝑐𝑡𝑖({𝐴𝑑𝑑𝑟𝑒𝑠𝑠 ∶ 𝑎𝑑𝑑𝑟, 𝐻𝑒′𝑎𝑟𝑡𝐵𝑒𝑎𝑡 ∶ ℎ𝑏}).𝑀′𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖′𝑝𝐶𝑒𝑙𝑙𝑖(𝑎𝑑𝑑𝑟, ℎ𝑏, 𝑡)

′ ′ ′

+ 𝑢𝑝𝑑𝑎𝑡𝑒𝑖({𝐴𝑑𝑑𝑟𝑒𝑠𝑠 ∶ 𝑎𝑑𝑑𝑟 , 𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡 ∶ ℎ𝑏 }).𝑡𝑖𝑚𝑒(𝑡 ).𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙𝑖(𝑎𝑑𝑑𝑟 , ℎ𝑏 , 𝑡 )

𝑛𝑀𝑒个𝑚𝑏𝑒𝑀𝑟𝑠𝑒ℎ𝑚𝑖𝑝𝑏𝐶𝑒𝑟𝑒𝑠𝑙ℎ𝑙 𝑖𝑝𝐶𝑒𝑙𝑙 组成的 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐿𝑖𝑠𝑡 定义如下， 在初始状态时， 每一个

中的数据为空 (𝜖)：

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐿𝑖𝑠𝑡 𝑑=𝑒𝑓 (𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙1(𝜖) ∣ ⋯ 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙𝑛(𝜖))

ℎ𝑏 +此1 外，我们定义了一个 𝐶𝑜𝑢𝑛𝑡𝑒𝑟 维护和更新节点的心跳，在每一次 Gossip 周期 ℎ𝑏 =

，𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 可以通过 ℎ𝑒𝑎𝑟𝑡𝑏𝑒𝑎𝑡 通道获取心跳。

𝐶𝑜𝑢𝑛𝑡𝑒𝑟(ℎ𝑏) 𝑑=𝑒𝑓 ~~ℎ𝑒𝑎𝑟𝑡~~𝑏𝑒𝑎𝑡(s(ℎ𝑏)).𝐶𝑜𝑢𝑛𝑡𝑒𝑟(s(ℎ𝑏))

 On Uniform Approach To Random Process Model

对于节点本地地址的维护，我们定义了一个 𝐴𝑑𝑑𝑟𝐼𝑛𝑓 𝑜，𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 可以通过 𝑎𝑑𝑑𝑟𝑒𝑠𝑠 通道获取本地地址。

𝐴𝑑𝑑𝑟𝐼𝑛𝑓 𝑜(𝑎𝑑𝑑𝑟) 𝑑=𝑒𝑓 ~~𝑎𝑑𝑑𝑟𝑒~~𝑠𝑠(𝑎𝑑𝑑𝑟)

定义完以上的辅助工具后，我们可以定义 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 的逻辑了！

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 𝑑=𝑒𝑓 𝑎𝑑𝑑𝑟𝑒𝑠𝑠(𝑎𝑑𝑑𝑟).ℎ𝑒𝑎𝑟𝑡𝑏𝑒𝑎𝑡(ℎ𝑏).𝑠𝑒𝑙𝑒𝑐𝑡1(𝑥1) ⋯ 𝑠𝑒𝑙𝑒𝑐𝑡𝑛(𝑥𝑛).

𝑎𝑐𝑐𝑒𝑝𝑡({𝐴𝑑𝑑𝑟𝑒𝑠𝑠 ∶ 𝑎𝑑𝑑𝑟, 𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡 ∶ ℎ𝑏}, 𝑥1, ⋯ , 𝑥𝑛).𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝

+ 𝑑𝑒𝑙𝑖𝑣𝑒𝑟(𝑥1, 𝑥2, ⋯ , 𝑥𝑛).𝑃 𝑟𝑜𝑐𝑒𝑠𝑠𝑖𝑛𝑔(𝑥1, 𝑥2, ⋯ , 𝑥𝑛)

𝑃 𝑟𝑜𝑐𝑒𝑠𝑠𝑖𝑛𝑔(𝑥1, 𝑥2, ⋯ , 𝑥𝑛) 𝑑=𝑒𝑓 (𝐶ℎ𝑒𝑐𝑘(𝑥1) ∣ ⋯ ∣ 𝐶ℎ𝑒𝑐𝑘(𝑥𝑛) ∣ 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝)

𝐶ℎ𝑒𝑐𝑘(𝑥) 𝑑=𝑒𝑓 𝑎𝑑𝑑𝑟𝑒𝑠𝑠(𝑎𝑑𝑑𝑟).((𝑎𝑑𝑑𝑟 = 𝑥[𝐴𝑑𝑑𝑟𝑒𝑠𝑠])0|⌝(𝑎𝑑𝑑𝑟 = 𝑥[𝐴𝑑𝑑𝑟𝑒𝑠𝑠])𝐹 𝑖𝑛𝑑𝐴𝑛𝑑𝑈 𝑝𝑑𝑎𝑡𝑒1(𝑥))

𝐹 𝑖𝑛𝑑𝐴𝑛𝑑𝑈 𝑝𝑑𝑎𝑡𝑒𝑖(𝑥) 𝑑=𝑒𝑓 𝑠𝑒𝑙𝑒𝑐𝑡𝑖(𝑥𝑖).( (𝑥𝑖 = 𝜖)(𝑢𝑝𝑑𝑎𝑡𝑒𝑖(𝑥).0)

|⌝(𝑥𝑖 = 𝜖)(

(𝑥𝑖[𝐴𝑑𝑑𝑟𝑒𝑠𝑠] = 𝑥[𝐴𝑑𝑑𝑟𝑒𝑠𝑠])(

(𝑥𝑖[𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡] < 𝑥[𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡])(𝑢𝑝𝑑𝑎𝑡𝑒𝑖(𝑥).0)

∣⌝(𝑥𝑖[𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡] < 𝑥[𝐻𝑒𝑎𝑟𝑡𝐵𝑒𝑎𝑡])0)

∣⌝(𝑥𝑖[𝐴𝑑𝑑𝑟𝑒𝑠𝑠] = 𝑥[𝐴𝑑𝑑𝑟𝑒𝑠𝑠])𝐹 𝑖𝑛𝑑𝐴𝑛𝑑𝑈 𝑝𝑑𝑎𝑡𝑒𝑖+1(𝑥))), 𝑖 ≤ 𝑛

𝐹 𝑖𝑛𝑑𝐴𝑛𝑑𝑈 𝑝𝑑𝑎𝑡𝑒𝑖(𝑥) 𝑑=𝑒𝑓 0, 𝑖 > 𝑛

其中𝑀s(𝑥𝑒𝑚) 表𝑏𝑒示𝑟𝑠ℎs𝑖(𝑝𝑥) = 𝑥 + 1。

可以获取 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐿𝑖𝑠𝑡 中所有 Cell 中储存的信息，添加本地地址

和𝑎𝑐𝑐心𝑒𝑝跳𝑡

𝑘 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝 𝑑𝑒𝑙𝑖𝑣𝑒𝑟

后，将这一组信息经 𝑎𝑐𝑐𝑒𝑝𝑡 通道发出，根据 𝑁𝑜𝑑𝑒 的定义，我们知道 𝑁𝑜𝑑𝑒 从通道接受这些信息后会随机的发送给 个其他节点。另外， 从

𝑃通𝑟道𝑜𝑐接𝑒𝑠𝑠收𝑖𝑛了𝑔(𝑥其,他𝑥 节, ⋯点, 发𝑥 送)

当𝑥 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

的 Membership List 后，将迁移到状态 𝑃 𝑟𝑜𝑐𝑒𝑠𝑠𝑖𝑛𝑔(𝑥1, 𝑥2, ⋯ , 𝑥𝑛)。

1 2 𝑛

可以并发的运行 𝐶ℎ𝑒𝑐𝑘(𝑥𝑖)，在 𝐶ℎ𝑒𝑐𝑘(𝑥𝑖) 状态下，若 𝑥𝑖 的地址为

𝑖 的地址与𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙 的地址，若地址相同且𝑥𝑖 中的心跳计数大于𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙中的心跳计数，更新 的数据；若所有有值的 均无 𝑖 的地址，用 𝑖 直接更新第一个空的 Cell。

𝑥

𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙 𝑀𝑒𝑚𝑏𝑒𝑟𝑠ℎ𝑖𝑝𝐶𝑒𝑙𝑙

## Gossip-Style Membership 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 Protocol 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。

### Go 语言与 CSP

我们将使用 Go 语言实现 Gossip-Style Membership Protocol，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 并发模型。如我们在第 [一](#_bookmark1)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](#_bookmark81), [36-38]](#_bookmark106)。Go 语言使用了CSP 理论中的Process/Channel， 对应到语言中的 goroutine/channel[[39](#_bookmark107)]。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。

### 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](#_bookmark68)。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。

ℝ𝕍ℙℂ

在 Go 语言中，我们可以用 channel 特性来实现 Th 中的通道。如在实现 Gossip-

Style Membership 协议的节点结构时，Node 结构体中，chan 类型的字段分别代表了图 [3–5](#_bookmark57)中的相应名字的通道。其中，我们在 [3.2.1](#_bookmark50)小节中提到 𝑖,𝑗 通道用于节点 向节点 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 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

}

𝑡𝑟𝑎𝑛𝑠

𝑡𝑟𝑎𝑛𝑠 𝑖 𝑗

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 [3.2.3.2](#_bookmark56)中实现的系统中的：

𝑁𝑜𝑑𝑒𝑖 𝑑=𝑒𝑓 𝑡𝑖𝑚𝑒𝑜𝑢𝑡.𝑎𝑐𝑐𝑒𝑝𝑡(𝑥).(𝑝𝑒𝑟𝑚⨁∈ 𝑝𝑝𝑒𝑟𝑚𝜏.𝐺𝑜𝑠𝑠𝑖𝑝𝑖𝑛𝑔𝑁𝑜𝑑𝑒𝑖,𝑝𝑒𝑟𝑚(𝑥))

+ 𝑗∈∑N/{𝑖}

𝑡𝑟𝑎𝑛𝑠𝑗,𝑖(𝑥).𝑑𝑒𝑙𝑖𝑣𝑒𝑟(𝑥PE)R.𝐹M𝑖𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒𝑖

这里 𝑁𝑜𝑑𝑒𝑖 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 Mem- bership List 随机选择 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

}

𝑘

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𝑑一𝑒𝑓个 goroutine，我们同样使用 goroutine 来

𝐹实𝑟现𝑎𝑔𝑖ℝ𝑙𝑒𝕍𝑁ℙ𝑜ℂ𝑑T𝑒h )中的并发。如对 𝐺𝑜𝑠𝑠𝑖𝑝𝑆𝑦𝑠𝑡𝑒𝑚N = (𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒1 ∣ 𝐹 𝑟𝑎𝑔𝑖𝑙𝑒𝑁𝑜𝑑𝑒2 ∣ ⋯ ∣

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](#_bookmark63)。

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

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数值 | 参数含义 |
| 𝑘𝑛 | 5 | 系统节点数量 |
| 𝑝𝑓 𝑎𝑖𝑙 | 2  0.1 | 每 Gossip 周期发送的节点数节点失效概率 |

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。

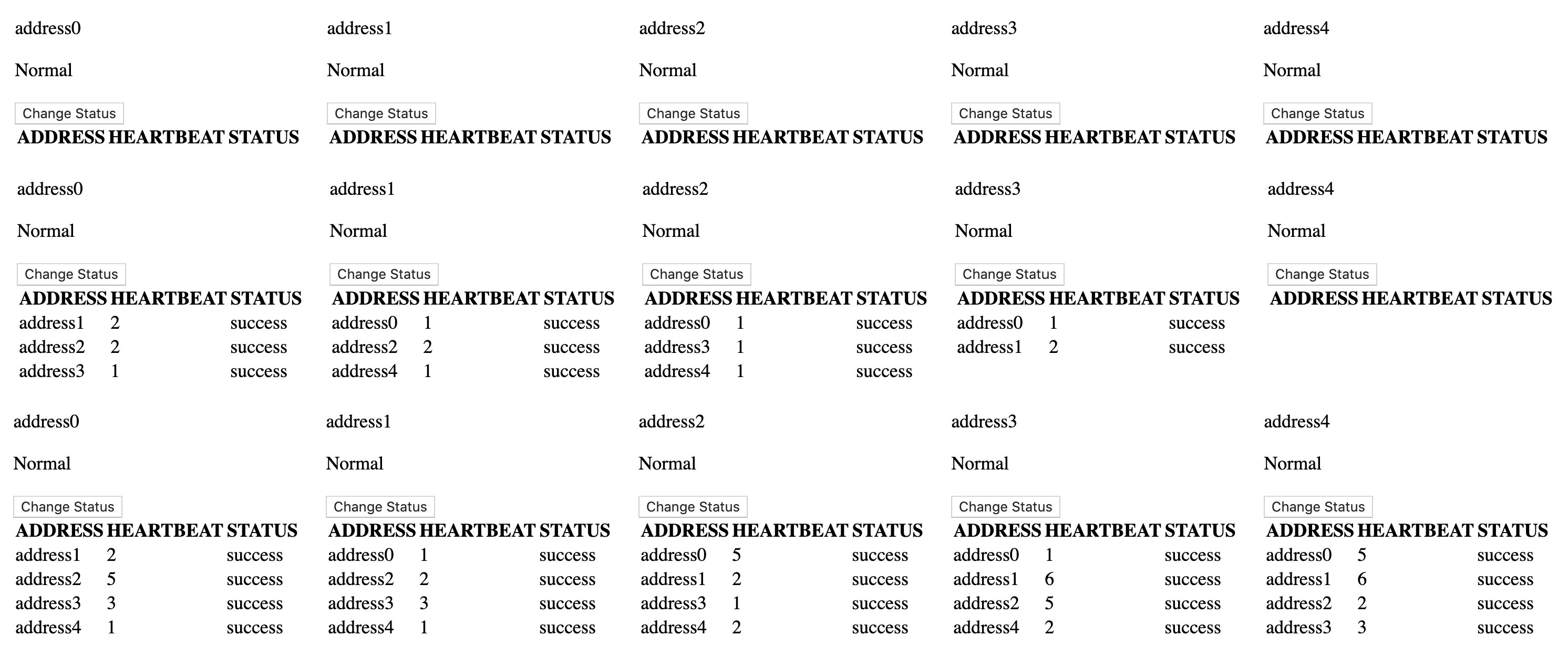


图 3–7 基于 Gossip-Style Membership 协议的 5 节点 P2P 系统的 Membership List 扩展过程

图 [3–7](#_bookmark64)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

𝑇

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 𝑓 𝑎𝑖𝑙 时，我们认为这一项对应的节 点处于失效状态。在 [3–7](#_bookmark64)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](#_bookmark65)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 ad- dress0,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。

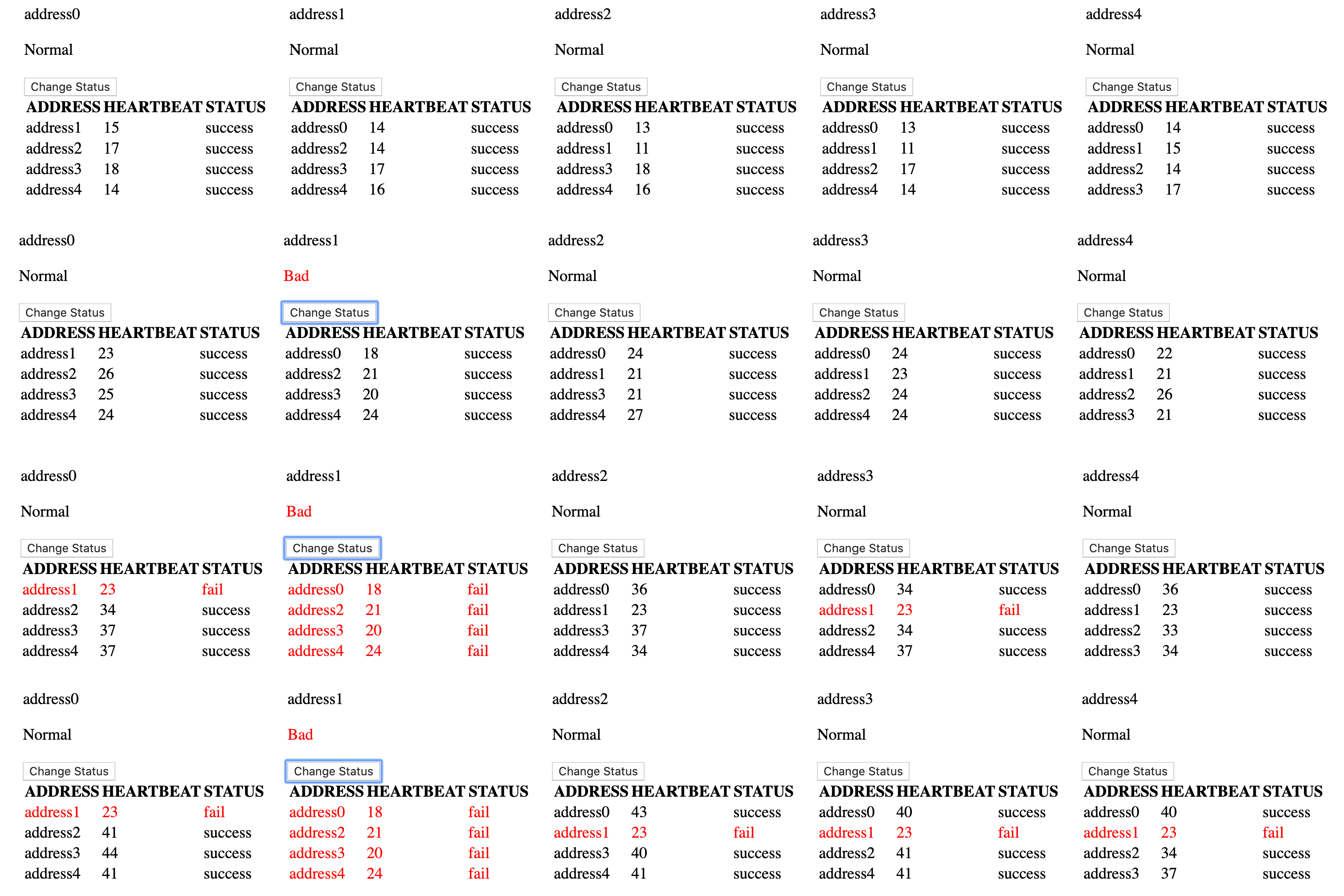


图 3–8 基于 Gossip-Style Membership 协议的 5 节点 P2P 系统的错误检测

我们也可以通过设置 𝑝𝑓 𝑎𝑖𝑙 使系统中的节点以一定概率自动失效和恢复，动态效果可以通过附录 [A](#_bookmark68)中的方式获得。

## 本章小结

ℝ𝕍ℙℂ

本章我们以一种云计算失效检测协议 Gossip-Style Membership 协议的 Th 实现作为本文中提出的随机传值进程模型的应用，提供了使用 Th 建模及实现通信协议以至其他现实问题的思路。

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

我们首先介绍了Gossip 协议和Membership 协议，并使用

Th 实现了基于Gossip 协议的P2P 系统，使用 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。

ℝ𝕍ℙℂ

# 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。

𝕍ℙℂ 𝕍ℙℂ

本文在传值进程模型 Th 的基础上，使用一种模型无关的概率扩展方法，扩展了

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 Th

ℝ𝕍ℙℂ

同时，为了给出 ℝ𝕍ℙℂTh

义了条件等价集、条件等价树（森林）、条件 𝑙 𝑞 ℝ𝕍ℙℂ

。

Uniform Approach 的方法定

的观察等价性的定义，我们使用

转移和条件 转移，进而定义了 Th 下

𝕍ℙℂ

的静态迁移 𝐴 ⇒𝜑 𝐴 = 𝐴，ℝ𝕍ℙℂTh 中的符号互模拟的本质是用条件等价树中条件 𝜑 对应

的符号互模拟。在 Un′ iform Apprach 中，条件等价树的本质是概率化了 Th 中的状态保持

的划分 {𝜑𝑖}𝑖∈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 Th 的观察等价性，并给出了观察等价性的同余性证明。

ℝ𝕍ℙℂ

ℝ𝕍 ℙ最ℂ 后，我们以一种云计算协议 Gossip-Style Membership Protocol 为例，给出了使用拟给出了 Gossip 协议与我们定义的多播规约的观察等价性。给出 Gossip-Style Membership Protocol 的模型的同时，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 Th 操作子的实现。事实证明 Th 在建模和分析通信过程时具有一定的可行性。我们希望 Th 可以为含有传值、计算性质的现实问题的建模和分析提供有效、可行的方法。

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

Th 建模通信过程，乃至其他现实问题的方法和过程，并使用 ℝ𝕍ℙℂTh 的符号互模

然而，目前的模型仍有一定的不足，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](#_bookmark45)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 Th 的基础上考虑值的随机分布。

ℝ𝕍ℙℂ

但遗憾的是，在第 [三](#_bookmark45)章中我们给出的应用，包括在第 [二](#_bookmark7)章中的简单的例子，实际上

都是 ℝ𝕍 ℙℂ

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[[3](#_bookmark72)] 中在将

书中的提供了一个思路：为了建模传值的 CCS，MILNER R 将 Value-passing CCS 转化为

𝑆 = 𝑥 𝑆 = ∑ 𝑎 .𝑇 {𝑣/𝑥} 𝑉

𝑆CC=S 时将 𝑝 .𝜏.𝑎(𝑣).𝑇 {转𝑣/𝑥为} 𝑣∈𝑉 𝑣 ，其中 是给定值的集合。我们也可以用

⨁𝑣∈𝑉 𝑣

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](#_bookmark45)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。

# 附录 A 基于 Gossip-Style Membership 协议的 P2P 系统的Go 语言实现

附录 [A](#_bookmark68)中仅展示与 ℝ𝕍ℙℂ𝕋 𝕙 有关的部分，若要运行完整代码，可以移步<https://github.com/Angeladadd/AGossipStyleMembershipProtocolImplementedUsingRVPC>。Node 的 Go 语言实现。

**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** ( "encoding/json" "fmt" "./simple" "strconv"

"log" "net/http" "io/ioutil"

)

**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

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**

}

# 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://b ooks.google.com.hk/books?id=n79Zh2JzBhYC](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](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](https://doi.org/10.1145/359576.359585).
5. BERGSTRA J, KLOP J. Algebra of communicating processes with abstraction[J/OL]. The- oretical Computer Science, 1985, 37: 77-121. [http://www.sciencedirect.com/science/article](http://www.sciencedirect.com/science/article/pii/030439758590088X)

[/pii/030439758590088X](http://www.sciencedirect.com/science/article/pii/030439758590088X). DOI: https://doi.org/10.1016/0304-3975(85)90088-X.

1. BOLOGNESI T, BRINKSMA E. Introduction to the ISO specification language LO- TOS[J/OL]. Computer Networks and ISDN Systems, 1987, 14(1): 25-59. [http:// www. sci encedirect.com/science/article/pii/0169755287900857](http://www.sciencedirect.com/science/article/pii/0169755287900857). DOI: https://doi.org/10.1016/0169-7 552(87)90085-7.
2. 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.
3. HANSSON H, JONSSON B. A calculus for communicating systems with time and probabil- ities[C]. in: [1990] Proceedings 11th Real-Time Systems Symposium. 1990: 278-287.
4. LOWE G. Probabilities and Priorities in Timed CSP[D]. 1993.
5. ANDOVA S. Process Algebra with Probabilistic Choice[C]. in: KATOEN J P. Formal Meth- ods for Real-Time and Probabilistic Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999: 111-129.
6. FU Y. A Uniform Approach to Random Process Model[J/OL]. CoRR, 2019, abs/1906.09541. arXiv: [1906.09541](https://arxiv.org/abs/1906.09541). <http://arxiv.org/abs/1906.09541>.
7. FIDGE C. A Comparative Introduction to CSP, CCS and LOTOS[R/OL]. Software Verifi- cation 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>.
8. 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.or g/10.1007/978-3-642-39698-4\_11](https://doi.org/10.1007/978-3-642-39698-4_11). DOI: [10.1007/978-3-642-39698-4\_11](https://doi.org/10.1007/978-3-642-39698-4_11).
9. HUANG H, YANG F. An interpretation of Erlang into value-passing Calculus[J]. Journal of Networks, 2013, 8(7): 1504-1513.
10. WINSKEL G. A presheaf semantics of value-passing processes[C]. in: MONTANARI U, SASSONE V. CONCUR ’96: Concurrency Theory. Berlin, Heidelberg: Springer Berlin Hei- delberg, 1996: 98-114.
11. INGÃ³LFSDÃ³TTIR A, LIN H. CHAPTER 7 - A Symbolic Approach to Value-Passing Pro- cesses[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](http://www.sciencedirect.com/science/article/pii/B9780444828309500254)

[/pii/B9780444828309500254](http://www.sciencedirect.com/science/article/pii/B9780444828309500254). DOI: https://doi.org/10.1016/B978-044482830-9/50025-4.

1. 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.
2. DAS A, GUPTA I, MOTIVALA A. SWIM: scalable weakly-consistent infection-style pro- cess group membership protocol[C]. in: Proceedings International Conference on Dependable Systems and Networks. 2002: 303-312.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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](https://doi.org/10.1145/233551.233556).
8. 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](https://doi.org/10.1145/233551.233556).
9. BAIER C, HERMANNS H. Weak bisimulation for fully probabilistic processes[C]. in: GRUMBERG O. Computer Aided Verification. Berlin, Heidelberg: Springer Berlin Heidel- berg, 1997: 119-130.
10. 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.
11. ANDOVA S, WILLEMSE T A. Branching bisimulation for probabilistic systems: Charac- teristics and decidability[J/OL]. Theoretical Computer Science, 2006, 356(3): 325-355. [http](http://www.sciencedirect.com/science/article/pii/S0304397506001459)

[://www.sciencedirect.com/science/article/pii/S0304397506001459](http://www.sciencedirect.com/science/article/pii/S0304397506001459). DOI: https://doi.org/10

.1016/j.tcs.2006.02.010.

1. HENNESSY M, LIN H. Symbolic bisimulations[J/OL]. Theoretical Computer Science, 1995, 138(2): 353-389. [http:// www. sciencedirect. com/ science/ article/ pii/ 030439759400 172F](http://www.sciencedirect.com/science/article/pii/030439759400172F). DOI: https://doi.org/10.1016/0304-3975(94)00172-F.
2. 曹晓东. 离散数学及算法第二版[M]. 机械工业出版社, 2013.
3. DEMERS A, et al. Epidemic Algorithms for Replicated Database Maintenance[J]. Proceed- ings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, 1987, 87: 1-12. DOI: [10.1145/41840.41841](https://doi.org/10.1145/41840.41841).
4. FANTI G, VISWANATH P. Anonymity Properties of the Bitcoin P2P Network[Z]. 2017. arXiv: [1703.08761 [cs.CR]](https://arxiv.org/abs/1703.08761).
5. GUREVICH Y, MANI R. Group Membership Protocol: Specification and Verification[M]. in: Specification and Validation Methods. USA: Oxford University Press, Inc., 1995: 295-328.
6. 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.
7. 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.
8. LIENIG J, BRUEMMER H. Reliability Analysis[M/OL]. in: Fundamentals of Electronic Systems Design. Cham: Springer International Publishing, 2017: 45-73. [https://doi.org/10.1 007/978-3-319-55840-0\_4](https://doi.org/10.1007/978-3-319-55840-0_4). DOI: [10.1007/978-3-319-55840-0\_4](https://doi.org/10.1007/978-3-319-55840-0_4).
9. HATZEL M, WAGNER C, PETERS K, et al. Encoding CSP into CCS (Extended Version)[Z]. 2015. arXiv: [1508.01127 [cs.LO]](https://arxiv.org/abs/1508.01127).
10. Van GLABBEEK R. Musings on Encodings and Expressiveness[J/OL]. Electronic Proceed- ings in Theoretical Computer Science, 2012, 89: 81-98. [http://dx.doi.org/10.4204/EPTCS.89](http://dx.doi.org/10.4204/EPTCS.89.7)

[.7](http://dx.doi.org/10.4204/EPTCS.89.7). DOI: [10.4204/eptcs.89.7](https://doi.org/10.4204/eptcs.89.7).

1. 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.
2. PRABHAKAR R, KUMAR R. Concurrent programming with Go[R/OL]. Google, Tech. 2011. [http://www.golang.org](http://www.golang.org/).

# 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。

感谢交大制作毕业设计 LATEX 模板 [SJTUTHESIS](https://github.com/sjtug/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 parallel and distributed computer system, concurrency theory has become an important branch of the theory of programs. Research on concurrency theory deepens people’s understanding of concurrent system. 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 methods using algebraic methods to study concurrent system, 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 nondetermin- istic behaviors and random choices. In order 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 behavior. Therefore, it is meaningful to introduce the concept of randomness in the concurrent process model. As an important extant to concurrency the- ory, probabilistic process has been widely researched. Recently, Fu proposes a Uniform Approach used to turn a process model into a randomized extension. Since it is a model independent approach, we can use it to derive a probabilistic extension of any other process model. Fu expresses 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 val- ues chosen from some data domain, and the propositions appearing in the conditionals are formulas constructed from a logic. It can be applied to modeling and analyzing communicating processes, bi- ological processes and other real-world problems with value-passing characteristics. In most studies of value-passing calculus, there exits some oracles providing data domain, logic decision and even computation of functions in value-passing processes. Those oracles are usually remained undefined making it hard to analyze the expressiveness of those models. However, Fu proposes a value-passing calculus, called Th, using a first order theory to decide the bool expression and a turing complete numeric system to derive the outcomes of functions, which successfully avoid such oracles.

𝕍ℙℂ

𝕍ℙℂ 𝕍ℙℂ

Consider Th’s great expressiveness, we decide to use Uniform Approach to extend Th into a probabilistic version, called Th. Hopefully, Th could help to model and analyze some meaningful real-world process such as communication and network security etc. With the aim of benefit mankind, we apply Uniform Approach to Th and get the grammar and symbolic transition syntax of Th, specifically, we add random choice operator to Th as well as a transition rule to random choice.

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

𝕍ℙℂ

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 in-

cludes Milner’s weak bisimulation and van Glabbeek and Weijland’s branching bisimulation. As for equivalence of probabilistic process, there already exits some research on full probabilistic pro- cesses, finite states probabilistic processes etc. However, it is hard to use branching bisimulation or weak bisimulation to describe the equivalence of value-passing process because of the conditional operator. In this way, scholars came up with a symbolic bisimulation for value-passing calculus. Fu also defined the symbolic bisimulation of Th.

𝕍ℙℂ

Uniform Approach proposes a random version of branching bisimulation to describe the equiv- alence of RCCS. Similarly, we can use the method in Uniform Approach to define a random version of symbolic bisimulation for our 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. Uniform Approach uses Epsilon tree to construct a random branching bisimulation. The main difficulty of using Uni- form Approach to define Th’s symbolic bisimulation is still the conditional operator. With the aim of eliminate the influence of conditional operator, we propose the conditional equivalence class and conditional Epsilon tree. 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 ele- ment 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 Th for those processes. Then we define the observance equivalence of Th as the largest codivergent and symbolic bisimulating relation. We manage to prove the congruence of our observance equivalence. Meanwhile, we also provide some interesting examples to annotate the concepts mentioned above. In addition, for showing 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 every gossip time, each node will pick k other nodes in a cluster to send gossip message. 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 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. Result shows that our peer-to-peer system implemented using Th runs correctly, which implies that Th is feasible to be used in modeling and analyzing real-world process with value-passing characteristics, such as communicating processes and biology processes etc. Hopefully, after we define and prove several concepts of Th and demonstrate with an application of it, modeling and analyzing

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ

ℝ𝕍ℙℂ ℝ𝕍ℙℂ

real-world process with value-passing characteristics could be easier and even a routine work.

Although there still exists something not well considered in our model, our expansion 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.