上海交通大学博士学位论文

互模拟等价性验证问题研究

博 士 研 究 生：张文博

学　　号：0140379002

导　　师：傅育熙 教授

申 请 学 位：工学博士

学　　科：软件工程

所 在 单 位：电子信息与电气工程学院

答 辩 日 期：2020 年 9 月 10 日

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Doctor

ON BISIMILARITY EQUIVALENCE CHECKING PROBLEMS

Candidate:	Wenbo Zhang
Student ID:	0140379002
Supervisor:	Prof. Yuxi Fu
Academic Degree Applied for:	Doctor of Engineering
Speciality:	Software Engineering
Affiliation:	School of Electronic Information and Electrical Engineering
Date of Defence:	Sep. 10th, 2020
Degree-Conferring-Institution:	Shanghai Jiao Tong University

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，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：张文博

日期：2020年8月21日

上海交通大学

学位论文使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，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。

本学位论文属于 ☑公开论文

□内部论文，□1 年/□2 年/□3 年　解密后适用本授权书。

□秘密论文，____年（不超过 10 年）解密后适用本授权书。

□机密论文，____年（不超过 20 年）解密后适用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：张文博　　指导教师签名：

日期：2020 年 8 月 21 日　　日期：2020 年 9 月 2 日

互模拟等价性验证问题研究

摘　要

等价性验证是形式化验证的一种主要方法，其目标是判定两个系统是否等价，一个常见的应用场景是判定系统设计和系统实现是否等价。互模拟等价是验证领域中最受研究者们关注的等价关系之一。根据强弱不同的等价性要求，互模拟等价又可分为强互模拟等价，分支互模拟等价和弱互模拟等价等精细程度不同的等价关系。图灵机的等价性是不可判定的，因此相关研究都是在表达能力受限的模型上进行的。本文的研究将重点聚焦于以下两个模型的互模拟等价验证问题：(1) 下推自动机和 (2) 概率进程演算。下推自动机是一个应用广泛的模型，适用于建模带有递归调用的程序；概率进程演算在一般的进程演算中引入概率选择算子，能够适用于建模带有概率的并发模型。

本文的主要贡献有以下几个方面：

第一，我们研究了下推自动机的强互模拟等价问题的复杂性。我们证明了一般的下推自动机的强互模拟是 ACKERMANN-难的。结合此前其他研究者给出的 ACKERMANN 的上界，可以得到该问题是 ACKERMANN-完备的结论。同时，我们的下界证明也适用于下推自动机的一个子模型：赋范下推自动机。因此，赋范下推自动机的强互模拟等价问题也是一个 ACKERMANN-完备问题。

第二，我们研究了在下推自动机中，当状态数固定时的强互模拟等价问题的参数复杂性。此前的研究表明，控制状态数量是影响该问题复杂性的一个重要的参数。我们证明了在下推自动机的状态数固定为 $d \geq 4$ 时，其强互模拟等价问题是$\mathbf{F}_{d-1}$-难的。其中，$\mathbf{F}_{d-1}$ 是一个介于初等复杂性类 (Elementary complexity class) 和原始递归的复杂性类。而在两个状态的赋范下推自动机中，我们证明了其强互模拟等价问题是 EXPTIME-难的。在上界方面，当赋范下推自动机的状态数固定为

d 时，我们将其他研究者给出的$\mathbf{F}_{d+4}$ 的上界改进为$\mathbf{F}_{d+3}$ 的上界。

第三，我们研究了一个近些年来新提出的定义概率进程演算的模型无关的方法。我们以该方法为基础，以有限状态的随机通信系统演算（Randomized Calculus of Communicating Systems, RCCS）为例，研究了有限状态的概率进程演算的分支互模拟等价问题。一方面，我们提出了一个多项式时间的分支互模拟等价性判定算法；另一方面，我们设计了一个公理系统，并且证明了该公理系统的可靠性和完备性。

关键词：互模拟, 等价性验证, 下推自动机, 概率进程演算

ON BISIMILARITY EQUIVALENCE CHECKING PROBLEMS

ABSTRACT

Equivalence Checking is one of the major approaches in the area of formal verification. Equivalence checking can be used to determine when two systems (specification and implementation) should be considered as the same. Bisimulation equivalence plays a central role among different equivalence relations studied in the area of verification. There are many different kinds of bisimulation equivalence, such as strong bisimulation equivalence, branching bisimulation equivalence, and weak bisimulation equivalence. As to the considered systems, people are more interest in models which are strictly weaker than Turing machine, as bisimilarity problem for Turing machine is undecidable. This thesis focuses on the bisimulation equivalence on pushdown automata (PDA) and probabilistic process calculus. PDA are widely used in theoretical computer science and can model recursive programs naturally. Probabilistic process calculus are process calculus extend with probabilistic choice operators. Probabilistic process calculus can be used to model probabilistic concurrent programs. This thesis studies on the bisimilarity problems for pushdown automata and finite state fragment of probabilistic process calculus.

The main contributions of this thesis are as follows: Firstly, we study the strong bisimilarity problem for PDA. The problem was proven to be in ACKERMANN. And we prove that it is actually an ACKERMANN-complete problem by showing the problem has an ACKERMANN-hard lower bound. Our ACKERMANN-hardness can be generalized to normed PDA. So the strong bisimilarity problem for normed PDA is also ACKER-

MANN-complete. Secondly, we study the parametric complexity of strong bisimilarity problem for PDA when the number of control states is fixed as d. We show a $\mathbf{F}_{d-1}$-hardness result for the problem, where $\mathbf{F}_{d-1}$ is a complexity class between elementary and primitive recursive. When d is 2, we show the problem for normed PDA is EXPTIME-hard. As for the upper bound, we improve the $\mathbf{F}_{d+4}$ result to $\mathbf{F}_{d+3}$ when the PDA is normed. Lastly, we study a model independent approach for random process model proposed recently. In this thesis, we focus on the finite state RCCS (Randomized Calculus of Communicating Systems) model. We give a polynomial time algorithm to decide the branching bisimulation problem for RCCS. We also give an axiomatic system and we prove the soundness and completeness of the system.

KEY WORDS: Bisimulation, Equivalence Checking, Pushdown Automata, Probabilistic Process Calculus

目 录

插图索引

表格索引

算法索引

主要符号对照表

$\mathbb{N}$	自然数集合
$\|\cdot\|$	范数
$\mathcal{Q}$	PDA 状态集合
Γ	PDA 栈符号集合
Σ	PDA 动作符号集合
$\mathcal{R}$	PDA 规则集合
ϵ	空串
τ	内部动作
ℓ	可见动作或内部动作
λ	非确定动作或概率动作
$\sim$	强互模拟等价
$\approx$	弱互模拟等价
$\simeq$	分支互模拟等价
$\equiv$	同余关系
$\mathcal{A}$	公理系统
$\mathcal{P}$	进程集合
$\mathcal{P}_{fs}$	有限状态进程集合
$\mathcal{S}_{fs}$	有限状态项集合

第一章　引言

1.1　研究背景

现如今，计算机越来越广泛的被应用于各个领域。在一些对安全性要求极高的领域中，如何避免软件系统的缺陷是一个非常重要的课题。然而，有些软件系统的缺陷难以被发现，而当缺陷发生时，则会带来严重的后果。例如发生在上世纪八十年代的 Therac-25 事件。从 1985 年到 1987 年，美国及加拿大发生了至少六起由放射线疗法机器 Therac-25 引起的医疗事故。事故的原因是软件设计中的一个竞争条件 (race condition) 设计的缺陷，而这个缺陷由于重现的条件较为特殊，因此在测试时没有被发现。

形式化验证是一种能够有效保证软件系统安全性的方法。形式化验证的目标是根据某些形式规范或属性，对软件系统的正确性进行严格的证明，从而有效的提高系统的健壮性。目前，形式化验证的两种主要方法是模型检测（Model Checking）[1-3] 和等价性验证（Equivalence Checking）[4]。

模型检测的目标是判断一个系统是否满足一个给定的性质。这个性质可以是一个安全性质（safety property），或者是一个活性性质（liveness property）。安全性质表示坏的事情永远不会发生，例如两个进程永远不会同时访问同一个资源；活性性质表示好的事情总会发生，例如某个进程不会无限的等待，最终总会被执行。模型检测方法通常需要穷举所有的状态，因此会面临状态爆炸的问题。为此，研究者们探索出了很多方法试图解决这个问题，例如基于有序二元决策图（Ordered Binary Decision Diagrams）[5] 的符号模型检测（Symbolic Model Checking）[6]，偏序归约（Partial Order Reduction）[7, 8] 等等。目前，研究者在模型检测领域已经有了丰富的研究成果 [9]。

本文的研究聚焦于等价性验证。等价性验证的目标是判断系统设计和最终实现是否等价。等价性验证的一个关键问题是等价关系的选择，最初研究者们主要关注的等价关系是语言等价（Language Equivalence），即判定两个给定系统接受的语言是否是相同的。随着研究的深入，研究者们提出了更多的等价关系。其中受到关注最多的等价关系是互模拟等价 [10]。其基本思想是，一个进程的所有动作都可以被另一个进程模拟，并且得到的两个进程依然等价。互模拟等价是一个比语言等价更为精细的等价关系，如图1–1中的例子，进程 P_1 和 Q_1 是语言等价的，它们识别相同的语言 $\{ab, ac\}$；然而它们不是互模拟等价的，进程 Q_1 在做 a 动作时可能会进入不同的状态，而进程 P_1 做的 a 动作是确定的。

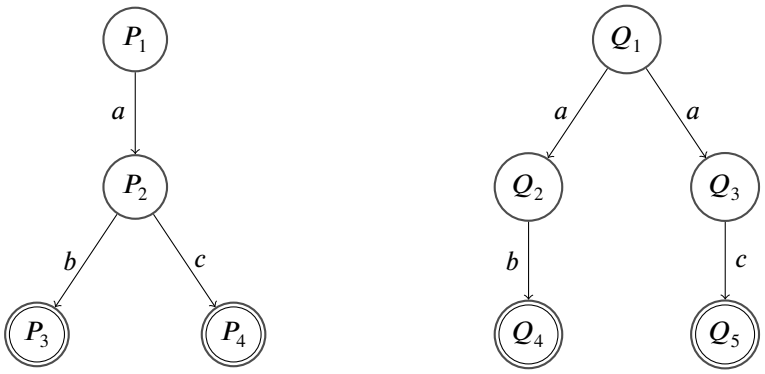


图 1–1 一个互模拟意义上不等价的例子

最早的互模拟等价由 Park 在 1981 年提出 [10]，这个定义此后被称作强互模拟等价（通常也直接叫做互模拟等价）。之后 Milner 在系统中引入内部动作，从观测者的角度提出了新的互模拟关系，称为弱互模拟等价 [11]。van Glabbeek 等人根据内部动作是否改变状态做了进一步区分，提出了分支互模拟等价 [12]。此外，更多的互模拟等价关系可以参见引文 [13]。

上述的多种等价关系在图灵机上都是不可判定的。然而，很多不是图灵完备的系统在程序验证中也有广泛的应用。例如下推自动机（Pushdown Automata, PDA）可以很自然的建模一个递归结构的程序；而 Petri 网（Petri Net, PN）[14] 则适合描述一个并发程序。等价性验证的大部分研究对象都是无限状态系统。系统可能包含的一些无界的数据类型，例如计数器，栈，队列等，会导致系统成为一个无限状态系统。另外，带参数的系统也可以看作无限状态系统。大部分无限状态系统都涵盖于 Mayr 定义的进程重写系统（Process Rewriting System, PRS）中 [15]。人们在这些不同表达能力的模型上得到了很多成果 [16]。本文的第一个重要研究对象是下推自动机，我们会重点关注下推自动机的强互模拟问题。

等价性验证也同样广泛应用在引入了并发操作的进程模型中。近些年来，概率进程模型也越来越受到研究者们的重视。很多确定算法无法有效解决的问题在引入概率后都可以得到很好的解决方案。人们在许多传统的进程演算模型中引入概率选择算子，得到了对应的概率进程演算模型，如概率 CCS（Probabilistic Calculus of Communicating Systems）[17, 18]，概率 CSP（Probabilistic Communicating Sequential Processes）[19]，概率 ACP（Probabilistic Algebra of Communicating Processes）[20]，和概率异步 π 演算（Probabilistic Asynchronous π Calculus）[21] 等等。最近，Fu 提出了一种模型无关的定义概率进程演算的方法 [22]。这类新的模型基于这样一个观点：概率进程模型中的概率选择动作应该是系统内部动作，概率可以由系统的内部运算来近似实现；而非确定选择算子无法由计算实现，是系统交

互的属性。本文的第二个研究对象是文章 [22] 中定义的随机通信系统演算（Randomized Calculus of Communicating Systems）。我们会重点关注随机通信系统演算的分支互模拟问题。

除了等价问题的可判定性，研究者们的另一个研究重点是这些问题的复杂性。而由于很多验证问题本身是非常困难的，随着输入规模的增长，计算时间的增长速度可能超越所有的初等函数。因此验证问题的研究需要引入增长很快的复杂性类。Schmitz 在 2016 年引入了一系列借用序数表示的在初等复杂性类（Elementary complexity class）之上的复杂性类 [23]，能够精确的描述很多验证领域中问题的难度。

本文的研究内容包括以下两点，一是下推自动机互模拟的算法和复杂性，以及当其控制状态数固定时的参数复杂性。二是随机通信系统演算上的分支互模拟等价的判定算法和有限公理化。

1.2 相关工作

1.2.1 语言等价验证

等价性验证最初的研究重点是语言等价。1956 年，Moore 证明了有限状态自动机的语言等价问题可判定 [24]。此后，理论科学家们开始探索其他表达能力更强的模型的语言等价问题。1961 年，Bar-Hillel 等人证明了上下文无关文法的语言等价问题是不可判定的 [25]。结合表格1–1中的 Chomsky 谱系 [26] 可以看到，此后的关键问题是确定的上下文无关语言的等价性判定问题。例1.1说明了，确定的上下文无关语言严格的包含于上下文无关语言。

例 1.1 定义如下语言：

$$L \stackrel{\text{def}}{=} \{a^n b^n : n \geq 1\} \cup \{a^n b^{2n} : n \geq 1\}$$

语言 L 是一个上下文无关语言，但不是确定的上下文无关语言。

因此文章 [25] 的不可判定结果不能直接推广到确定的上下文无关语言的等价性判定问题。并且在上下文无关文法等价性的不可判定的证明中，非确定性是很重要的一个条件 [25]，因此长期以来，人们都相信确定的上下文无关文法的语言等价问题是可判定的。但其证明经历了一段漫长的过程。

1966 年，Ginsburg 和 Greibach 研究了确定的上下文无关文法，并证明了许多性质 [27]。例如一个确定的上下文无关语言和一个正则语言是否相等是可判定的；一个确定的上下文无关语言是否包含于另一个确定的上下文无关语言是不可判定

表 1–1 Chomsky 谱系

Chomsky 层级	文法	语言	自动机
类型 0	无受限文法	递归可枚举语言	图灵机
类型 1	上下文有关文法	上下文有关语言	线性有界自动机
类型 2	上下文无关文法	上下文无关语言	下推自动机
——	确定上下文无关文法	确定上下文无关语言	确定下推自动机
类型 3	正则文法	正则语言	有限状态自动机

的。此外，他们提出了其语言等价问题是否可判定这个公开问题。为了解决这个问题，研究者们开始对确定的下推自动机的很多子模型上的语言等价问题进行探索。常见的一些子模型包括：

- 没有不可见动作 τ 的下推自动机，我们称该模型为实时下推自动机（real-time PDA）；
- 只有一个状态，确定的实时下推自动机识别的语言，我们称该语言为确定简单语言（Simple Deterministic Language）；
- 出栈行为和入栈行为只能切换有限次的下推自动机，我们称该模型为有限反转下推自动机（finite-turn PDA）；
- 只有一个栈符号的下推自动机，我们称该模型为单计数器自动机（One Counter Automata，OCA）。

1966 年，Korenjak 和 Hopcroft 证明了确定简单语言的语言等价问题可判定 [28]。1974 年，Valiant 证明了有限反转下推自动机的语言等价问题可判定 [29]。1975 年，Valiant 和 Paterson 证明了确定的单计数器自动机的语言等价问题可判定 [30]。之后，Romanovskii 和 Oyamaguchi 分别独立证明了实时下推自动机的语言等价问题可判定 [31, 32]。这些研究工作给人们提供了许多技术方面的积累和认识上的提升。直到 1997 年，Sénizergues 宣布证明了确定下推自动机的语言等价问题可判定 [33, 34]，这个杰出的结果获得了 2002 年的哥德尔奖。

而在复杂性方面，大部分问题也都已解决。非确定的有限状态自动机（Nondeterministic Finite Automata, NFA）的语言等价问题是 PSPACE-完备的 [35]。确定的有限状态自动机（Deterministic Finite Automata, DFA）的语言等价问题是 NL-完备的 [36]。确定简单语言上的语言等价问题是多项式时间可解的 [37]。目前的一个重要的公开问题，即 DPDA 的语言等价问题的复杂性。目前该问题最好的上界是 TOWER [38, 39]，而下界只是 P-难（判定一个上下文语言是否为空是一个 P-难问题，下界可以由此问题归约而来）。上下界之间还存在巨大的差距。

1.2.2 进程重写系统上的互模拟等价验证

结合近年来的研究来看，在可判定性以及复杂性上，互模拟等价通常会比语言等价有更好的结果。从复杂性的角度看，一个例子是 NFA 上的语言等价问题是 PSPACE-完备的 [35]，而互模拟等价可以在多项式时间解决 [40, 41]。而从可判定性的角度，Baeten 等人关于赋范基本进程代数（normed Basic Process Algebra, normed BPA）的结果 [42] 同样证实了这个观点。BPA 可以看作是只有一个状态的 PDA，一个系统是赋范的表示该系统中所有的进程都可以终止。Baeten 等人证明了赋范基本进程代数的互模拟可判定。而赋范基本进程代数识别的语言是上下文无关语言，因此其语言等价问题是不可判定的。最初 Baeten 等人的证明比较复杂，有很多研究者从不同角度对该证明做了简化 [43-45]。而在这几篇文章中提到的一些技术，包括互模拟基技术 [43] 以及 tableau 证明技术 [44]，在后来的互模拟等价验证的许多相关问题的研究中都得到了广泛的应用。

互模拟等价验证关注的模型中有很多都可以表示为某种重写系统（Rewriting System）。Mayr 提出了一个一般的框架称作进程重写系统 [15]。该框架涵盖了很多常见的模型，除了前文中提到的有限状态系统（Finite-state System, FS），PDA，BPA，Petri 网之外，还包括基本并行进程（Basic Parallel Process, BPP）[46]，进程代数（Process Algebra, PA）[47] 等模型。图1–2中表示了进程重写系统框架中不同的模型的相对表达能力的关系。在本节的余下部分，我们将总结图1–2中的部分模型的互模拟等价验证的研究现状。

基本进程代数（BPA） 在 Baeten 等人证明了赋范基本进程代数的强互模拟可判定 [42] 后不久，Huynh 和 Tian 在 1994 年证明了一个赋范基本进程代数的互模拟等价问题有 NP^{NP} 的上界 [48]，此后，Hirshfeld 等人给出了一个多项式时间算法 [37]。Czerwiński 和 Lasota [49] 在 2010 年给出了一个 $\mathcal{O}(n^5)$ 时间复杂度的算法，这是该问题目前已知的最好的复杂性结果。

当研究一个系统接受的语言时，系统的赋范性是一个不失一般性的要求。而当研究互模拟等价时，赋范的模型上的结果不能直接推广到一般的模型上。对于一般的基本进程代数的强互模拟问题，1995 年，Christensen [50] 给出了一个可判定性的证明。文章中用到了前文中提到的互模拟基的技术。Christensen 证明了互模拟关系等价于一组有限的互模拟基关于连接操作的同余关系。这个可判定性由两个半可判定过程组成，因此没有一个复杂性的上界。此后，Burkart 等人给出了一个初等复杂性类的上界 [51]，并且在该篇文章中，作者们声称文中的算法可以优化到 2EXPTIME。这篇文章的核心方法是将有限的互模拟基在初等函数的时间

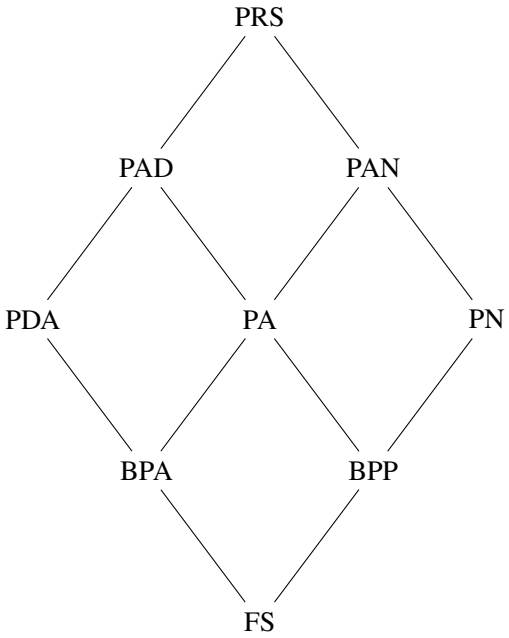


图 1–2 进程重写系统

内构造出来。此后，Jančar 给出了一个 2EXPTIME 的明确的证明 [52]。下界方面，2002 年，Srba 证明了 BPA 的互模拟等价是 PSPACE-难的 [53]。此后 Kiefer 将此下界提升至 EXPTIME-难 [54]。目前 2EXPTIME 的上界和 EXPTIME-难的下界之间的差距仍是领域内的一个重要的公开问题。

当等价关系考虑了内部动作，互模拟问题的可判定性的正面结论非常少。2013 年，Fu 证明了一个重要的结论：赋范基本进程代数的分支互模拟等价是可判定的 [55]。2015 年，He 和 Huang 证明了该问题是 EXPTIME-完备的 [56]（EXPTIME-难的详细证明于 2017 年发表在 Huang 和 Yin 的文章中 [57]）。除此之外，BPA 上的问题暂时没有其他正面结论。下界方面，2002 年，Mayr 证明了赋范基本进程代数的弱互模拟是 EXPTIME-难的 [58]。

我们在表1–2中总结了 BPA 上互模拟等价问题目前的研究现状。

基本并行进程（BPP） 1993 年，Christensen 等人证明了基本并行进程上的强互模拟是可判定的 [60]。2003 年 Jančar 给出了一个 PSPACE 的算法 [61]，而这与此前 Srba 证明的 PSPACE-难的下界 [62] 一起得到了一个完备结果。在赋范基本进程代数上，强互模拟有多项式时间算法 [63]，目前最好的上界是 $\mathcal{O}(n^3)$ [64]。

对于考虑了内部动作的等价关系，BPP 上的正面结论也是非常少的。2011 年，Czerwiński 等人证明了赋范 BPP 上的分支互模拟可判定 [65]。Srba 证明了赋范

表 1–2 基本进程代数上互模拟等价研究现状

	$\sim$	$\simeq$	$\approx$
赋范 BPA	P [37] P-难 [59]	EXPTIME [56] EXPTIME-难 [57]	? EXPTIME-难 [58]
一般 BPA	2EXPTIME [52] EXPTIME-难 [54]	? EXPTIME-难 [54]	? EXPTIME-难 [58]

BPP 上的弱互模拟等价是 PSPACE-难的 [66]。尹在其博士论文中证明了赋范 BPP 的分支互模拟也是 PSPACE-难的 [67]。

我们在表1–3中总结了 BPP 上互模拟等价问题目前的研究现状。

表 1–3 基本并行进程上互模拟等价研究现状

	$\sim$	$\simeq$	$\approx$
赋范 BPP	P [63] P-难 [59]	可判定 [65] PSPACE-难 [67]	? PSPACE-难 [66]
一般 BPP	PSPACE[61] PSPACE-难 [62]	? PSPACE-难 [62]	? PSPACE-难 [62]

下推自动机（PDA） 1996 年，Stirling 使用 tableaux 技术证明了赋范下推自动机的强互模拟可判定 [68]。1998 年，Sénizergues 证明了下推自动机的强互模拟可判定 [69]。此后，Stirling 使用 tableaux 技术给出了下推自动机的强互模拟判定性的一个新的证明 [70]。最近一段时间，Jančar 和 Schimtz 给出了判定一个下推自动机上强互模拟的 ACKERMANN 复杂性的算法 [71]。事实上该算法是在一阶文法 (First Order Grammar) 的模型上提出的。一阶文法等价于一个内部动作确定的 PDA。在下界方面，2010 年，Kučera 和 Mayr 证明了下推自动机的强互模拟是 EXPTIME-难的。此后这个结论被 Benedikt 等人提升至 TOWER-难 [72]，并且该下界对赋范下推自动机同样成立。

当等价关系考虑了内部动作后，Srba 证明了赋范下推自动机的弱互模拟不可判定 [73]。2014 年，Yin 等人证明了赋范下推自动机的分支互模拟也是不可判定的 [74]。值得一提的是，Sénizergues 在 1998 年的文章中证明的是一个更强的结论：如果 PDA 内部动作是确定的（等价于前面提到的一阶文法），其弱互模拟是可判

定的[69, 75]。此外 Fu 证明了当内部动作都是 push 操作时，其分支互模拟是可判定的 [76]。这些工作都使下推自动机上互模拟问题的可判定性的界限愈加清晰。

我们在表1–4中总结了 PDA 上互模拟等价问题目前的研究现状。

表 1–4 下推自动机上互模拟等价研究现状

	$\sim$	$\simeq$	$\approx$
赋范 PDA	ACKERMANN [71] TOWER-难 [72]	—— 不可判定 [74]	—— 不可判定 [73]
一般 PDA	ACKERMANN [71] TOWER-难 [72]	—— 不可判定 [74]	—— 不可判定 [73]

其他模型 在进程重写系统其他模型上的相关结果比较少。赋范进程代数（PA）的强互模拟有 2NEXPTIME 的上界 [77]。除此之外，大部分都是否定结果。例如早些时候进程代数的弱互模拟被证明是不可判定的 [73]。而进程重写系统中表达能力强过 BPA 和 BPP 的模型的分支互模拟都是不可判定的 [74]。Petri 网的强互模拟也是不可判定的 [78]。

1.2.3 概率进程演算

研究者们对概率并发系统的建模有过很多尝试。1990 年，Giacalone 等人提出了一种建模方式 [17]，其想法是在标记迁移系统上，为每一条边标记一个 0 到 1 的值，并要求从一个状态 P 出发的所有动作的边的概率之和为 0 或 1。此后不久，Larsen 和 Skou 提出了一种相似的方法 [79]，区别是要求从一个状态 P 出发的做相同动作的边的概率之和为 0 或 1。此后 van Glabeek 等人对这两种建模方式做了总结 [80]。给第一种模型一个术语称作自生模型（generative model），给第二种模型一个术语称作响应模型（reactive model）。图1–3中左边是自生模型中从一个状态出发的动作，右边是响应模型中从一个状态出发的动作。在自生模型中，一个进程只要没有被外界限制，可以主动的去选择做某个概率动作。而在响应模型中，进程只能被动的等待响应外界的交互，然后确定下一步的概率动作。

对于概率系统上互模拟等价关系的定义，Larsen 和 Skou 做了开创性工作 [79]。此后，又有许多研究者在其他概率模型上定义了互模拟关系 [81-83]。而在这些不同的模型上，研究者们对互模拟等价的判定算法 [84-87] 以及有限公理化都已经有了很多成果 [88-93]。

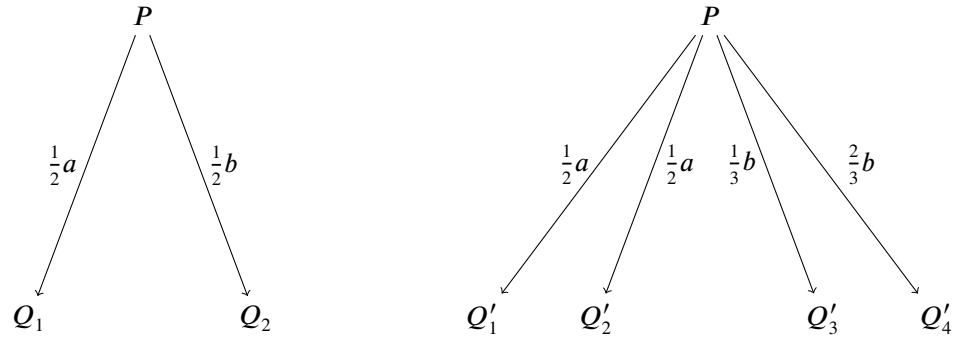


图 1–3 自生模型和响应模型

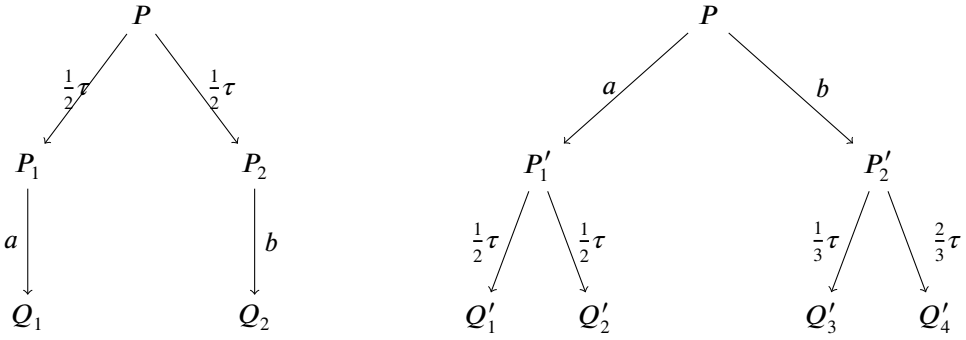


图 1–4 新模型与自生模型和响应模型的对应

区别于自生模型和响应模型，Fu 提出了一个模型无关的定义概率模型的方式[22]。其主要特点是概率动作都表现为一个内部动作。从现实的角度出发，概率都是可以通过计算来模拟的，而计算在交互系统中通常可以抽象为一个内部动作。与之相对，非确定是可能体现在交互中的属性，不能通过计算来实现。自生模型和响应模型都可以通过该新模型表示。例如图1–3中的两个进程可以分别对应到图1–4中的两个进程。

此外，文章 [22] 同样给出了一个模型独立的定义概率系统互模拟等价的方法。新的互模拟等价可以满足很多好的性质。例如，区别于传统的使用概率标记迁移系统（Probabilistic Labelled Transition System, pLTS）[94] 定义的互模拟等价关系，文章 [22] 中的方法定义的互模拟关系在复合（composition），限名（localization），递归（recursion）算子的作用下都是封闭的；并且，新的互模拟关系可以使进程在发散（divergence）性质上同步。

在新的概率系统中，互模拟等价的验证算法以及有限公理化等问题都还有待解决。

1.3 本文贡献

本文研究了下推自动机上的强互模拟等价问题和随机通信系统演算上的分支互模拟等价问题。主要贡献包括以下几点：

- 首先，本文证明了（赋范）下推自动机上的强互模拟等价问题是ACKERMANN-难问题，结合其他研究者的上界结果，（赋范）下推自动机上的互模拟等价问题是一个 ACKERMANN-完备问题。此结果为这个已经公开了数十年的问题画上了句号。这也说明，从 Sénizergues 的开创性工作 [69], 到 Stirling, Jančar 等人关于该问题的持续探索和改进 [70, 71, 95, 96]，该问题目前的算法从计算复杂性的角度来说已经是最优的了。
- 其次，本文研究了当控制状态数固定为 d 时，下推自动机的强互模拟等价问题的上下界。本文给出了一个（赋范）下推自动机上的强互模拟等价 $\mathbf{F}_{d-1}$-难的计算复杂性下界。而当 $d = 2$ 且下推自动机满足赋范性时，本文给出了一个 EXPTIME-难的下界。在上界方面，当下推自动机满足赋范性时，我们将之前$\mathbf{F}_{d+4}$ 的上界 [71] 改进到了$\mathbf{F}_{d+3}$。
- 再次，本文基于 Fu 提出的模型独立的概率进程演算的框架 [22]，以有限状态的随机通信系统演算为例，研究了其分支互模拟等价问题。主要贡献包括两点，一是给出了一个多项式时间的等价性判定算法，二是设计了一个随机通信系统演算上的分支互模拟等价的公理系统。本文中的算法和公理系统也可以很容易的推广到其他通过该模型独立的方法定义的概率进程演算系统中。
- 最后，本文在随机通信系统演算的算法部分的工作中，提出了 ϵ-图的概念来保证算法终止；在公理化部分的工作中，提出了概率被守卫的（probabilistically guarded）来解决传统的被守卫的（guarded）和强守卫的（strongly guarded）两个概念都不能应用在新的概率模型中的问题。此为本文在技术方面的贡献。

1.4 章节安排

本文的后续章节安排如下。

第二章将介绍一些准备知识。首先，我们将介绍标记迁移系统（Labeled Transition System, LTS），这是描述后文中其他模型的语义的重要工具。接下来，我们将会分别给出下推自动机和随机通信系统演算的严格定义。之后，我们将会介绍互模拟关系及其博弈刻画，并介绍防御者力迫技术，以及概率模型中的互模拟关系定义。最后，我们将简要介绍 Schmitz 定义的快速增长复杂性类 [23]。

第三章将重点研究下推自动机的强互模拟问题的复杂性。我们首先将会介绍一个 ACKERMANN-难的问题，即重置 Petri 网的可覆盖问题，我们将此问题归约到下推自动机上的强互模拟等价问题，从而证明了下推自动机上强互模拟问题是一个 ACKERMANN-难的问题。结合 Jančar 和 Schmitz 关于此问题给出的 ACKERMANN 的算法 [71]，我们可以说明此问题是一个 ACKERMANN-完备问题。这个下界还可以推广到赋范下推自动机。并且这个归约也给出了一个参数复杂性下界：当状态数量为 $d \geq 4$ 时，下推自动机的强互模拟问题是$\mathbf{F}_{d-1}$-难的。

第四章将会研究两个状态的赋范下推自动机上的强互模拟问题。本章将会给出一个从 hit-or-run 博弈问题到该问题的归约，从而证明该问题是 EXPTIME-难的。

第五章将会研究赋范下推自动机的参数复杂性的上界。相对于之前的$\mathbf{F}_{d+4}$ 的上界，我们将利用赋范下推自动机的一些特殊性质，将上界改进到$\mathbf{F}_{d+3}$。

第六章将会研究随机通信系统演算上的分支互模拟问题。本章的前半部分将会给出一个该问题的多项式时间的判定算法。后半部分将会给出一个随机通信系统演算上的分支互模拟的有限公理化，并证明其可靠性和完备性。

第七章将对全文做出总结。

第二章　准备知识

本章中，我们将会介绍本文的研究中用到的一些准备知识。在2.1节中，我们介绍标记迁移系统的概念。在2.2节和2.3节中，我们将分别介绍本文中重点研究的两个模型：下推自动机和随机通信系统演算。在2.4节中，我们将给出互模拟等价的定义以及互模拟等价的一个博弈刻画，并且我们还将介绍一个常用的技术，称作防御者力迫技术。2.5节将会介绍由 Schmitz 定义的复杂性类层级。

2.1　标记迁移系统

我们通常会用标记迁移系统来描述一个系统的语义。在本节中，我们给出标记迁移系统的定义。

定义 2.1　一个*标记迁移系统*（Labeled Transition System, LTS）$\mathcal{L}$ 是一个三元组 $(S, \Sigma, \rightarrow)$。

- S 是一个可数的状态集合；
- Σ 是一个有限的动作集合；
- $\rightarrow \subseteq S \times \Sigma \times S$ 是一个迁移规则集合。

标记迁移系统的定义和有限状态自动机的定义 [97] 的主要区别是标记迁移系统的状态数量可以是无限的，并且我们不会强调它的初始状态和终止状态。我们会将一个进程对应于标记迁移系统中的一个状态，将进程之间的迁移理解为迁移系统的动作。这样，本文中涉及到的不同的模型都可以翻译为一个标记迁移系统。

在本文中，我们将会用符号 P, Q 来表示标记迁移系统的状态；用符号 a, b, c 来表示 Σ 中的可见动作，用 τ 来表示 Σ 中的内部动作，用 ℓ 来表示一个可见动作或者内部动作。规则 $(P, \ell, Q) \in \rightarrow$ 可以直观的写作 $P \xrightarrow{\ell} Q$。我们用 $P \xrightarrow{\ell}$ 表示存在某个 $Q \in S$ 使得 $P \xrightarrow{\ell} Q$；如果不存在这样的 Q，我们表示为 $P \overset{\ell}{\nrightarrow}$。

我们用 u, v 表示一串连续的动作。对于 $u = \ell_1 \ell_2 \cdots \ell_k$，一条*路径*（path）$P \xrightarrow{u} Q$ 表示存在状态 $P_1, P_2, \cdots P_k \in S$，使得：

$$P \xrightarrow{\ell_1} P_1 \xrightarrow{\ell_2} \cdots P_{k-1} \xrightarrow{\ell_k} Q$$

我们也会用 $P \rightarrow^* Q$ 表示存在一条从 P 到 Q 的路径。

2.2 下推自动机

本小节我们介绍本文中的一个重要的研究对象：下推自动机。下推自动机可以表示为一个四元组 $\mathcal{A} = (\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$。其中

- $\mathcal{Q}$ 是一个有限的状态集合；
- Γ 是一个有限的栈符号集合；
- Σ 是一个有限的动作符号集合；
- $\mathcal{R} \subseteq \mathcal{Q} \times \Gamma \times \Sigma \times \mathcal{Q} \times \Gamma^*$ 是一个有限的规则集合。

这个定义采用了文章 [98] 中对 PDA 的定义。和经典的定义稍有不同的是我们不强调起始状态和终止状态。因为在互模拟等价的验证中，我们会将下推自动机看作一个迁移系统而非一个语言接收器 [97]。

我们用 $p, q \in \mathcal{Q}$ 来表示 PDA 的控制状态；用 $X, Y \in \Gamma$ 来表示 PDA 的栈符号；和 LTS 类似，我们也用 $a, b, c \in \Sigma$ 来表示 PDA 的可见动作，用 τ 来表示 Σ 中的内部动作，用 ℓ 来表示一个可见动作或者内部动作。如果 $pX \xrightarrow{\tau} q_1\alpha_1 \in \mathcal{R}$，$pX \xrightarrow{\tau} q_2\alpha_2 \in \mathcal{R}$ 可以推出 $q_1 = q_2, \alpha_1 = \alpha_2$，我们称 $\mathcal{A}$ 是一个*内部动作确定的*下推自动机。此外，我们用 $\alpha, \beta \in \Gamma^*$ 表示一串连续的栈符号，用 ϵ 表示一个空串。两个串 α 和 β 的连接可以直接表示为 $\alpha\beta$，我们约定 $\epsilon\alpha = \alpha\epsilon = \alpha$。我们用 $P, Q \in \mathcal{Q} \times \Gamma^*$ 来表示下推自动机的一个进程。下推自动机 $\mathcal{A}$ 的进程集合用 $\mathcal{P}_{\mathcal{A}}$ 来表示。我们用符号 $|\alpha|$ 表示 α 的长度。对于进程 $P = p\alpha$，我们定义 $|P| \stackrel{\text{def}}{=} |\alpha|$。

对于规则 $r = (p, X, \ell, q, \alpha) \in \mathcal{R}$，我们写作 $pX \xrightarrow{\ell} q\alpha$。在本文中，我们也可能会用 $P \xrightarrow{r} Q$ 来表示一条迁移规则。其中 r 是一条规则而不是一个动作，这样做的好处是可以确定的表示一条路径。我们定义映射 $\mathsf{lab}(r) = \ell$，这个定义可以推广到 $\mathsf{lab} : \mathcal{R}^* \to \Sigma^*$。

下推自动机的语义可以通过如下规则来说明：

$$\frac{pX \xrightarrow{\ell} q\alpha \in \mathcal{R}}{pX\beta \xrightarrow{\ell} q\alpha\beta} \tag{2–1}$$

根据规则 (2–1)，我们可以从一个 PDA $\mathcal{A} = (\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$ 自然的诱导出一个标记迁移系统 $\mathcal{L} = (\mathcal{P}_{\mathcal{A}}, \Sigma, \rightarrow_{\mathcal{A}})$。

例 2.1 在下推自动机 $(\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$ 中，$\mathcal{Q} = \{p, q\}$，$\Gamma = \{X\}$，$\Sigma = \{a, b, c\}$。$\mathcal{R}$ 包含以下规则：

$$pX \xrightarrow{a} pXX \qquad pX \xrightarrow{b} q\epsilon \qquad qX \xrightarrow{c} q\epsilon$$

我们可以诱导出标记迁移系统如图2–1所示。

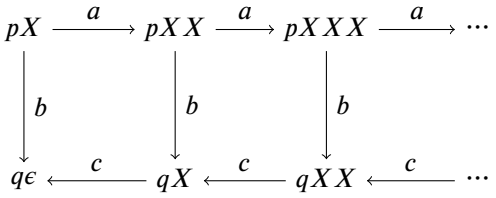


图 2–1 下推自动机诱导的标记迁移系统示例

接下来，我们定义一个重要的概念，称作进程的赋范性（normed）。赋范性表示系统内的任何进程应该在有限步内终止。

定义 2.2 对于一个下推自动机的进程 P，如果存在某个状态 $p_i \in Q$，并且存在一条迁移路径：

$$P \xrightarrow{\ell_1} P_1 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_n} p_i\epsilon \tag{2–2}$$

那么我们称进程 P 是赋范的。

我们定义 $\|P\|$ 为 P 的范数，表示最短的能够将进程 P 的栈做空的路径长度。如果不存在这样的路径，我们记 $\|P\| = \infty$。如果一个下推自动机中的所有进程都是赋范的，那么我们称其为一个赋范下推自动机。在赋范下推自动机中，对于任意状态 p 和任意栈符号 X，我们可以用动态规划在多项式时间内计算出所有的 $\|pX\|$。我们定义：

$$d_0 \stackrel{\text{def}}{=} \max\left\{\|pX\| : p \in Q, X \in \Gamma\right\}$$

不难证明，

$$d_0 < 2^{|Q|\cdot|\Gamma|}$$

d_0 是一个在 PDA 上的互模拟等价验证中被广泛用到的常数 [39, 96]。根据范数的定义，我们有以下一些基本的性质：

命题 2.1 对于一个赋范下推自动机中的进程 P，$\|P\| > 0$，

1. 存在一条迁移规则 $P \xrightarrow{\ell} Q$ 使得 $\|Q\| = \|P\| - 1$；
2. 对于任意迁移规则 $P \xrightarrow{\ell} Q$，我们有 $\|P\| - 1 \le \|Q\| < \|P\| + 2d_0$。

对于一个赋范进程，它的栈不断做空的过程就是范数逐渐减小的过程，我们定义如下概念：

定义 2.3 (范数下降路径) 如果一条路径

$$\pi = P_1 \xrightarrow{\ell_1} P_1 \xrightarrow{\ell_2} \cdots P_{k-1} \xrightarrow{\ell_k} P_k$$

满足对于 $1 \le i \le k-1$，$\|P_{i+1}\| = \|P_i\| - 1$ 成立，我们称 π 为一条范数下降路径。

根据命题2.1，对于任意一个赋范进程 P，都存在一条长度为 $\|P\|$ 的范数下降路径。

2.3 随机通信系统演算

本小节，我们介绍本文中的另一个重要的研究对象：随机通信系统演算（Random Calculus of Communicating System, RCCS）。

首先，我们用小写字母 a, b, c 来表示通道名，$\mathcal{C}$ 表示一个通道名的集合。定义 $\overline{\mathcal{C}} = \{\overline{a} : a \in \mathcal{C}\}$。我们用 $\mathcal{A}_n = \mathcal{C} \cup \overline{\mathcal{C}} \cup \{\tau\}$ 来表示非确定动作名的集合。小写希腊字母 α 表示集合 $\mathcal{A}_n$ 中的一个动作。我们用 $\mathcal{A}_p = \{q\tau : 0 < q < 1\}$ 来表示概率动作名的集合。$\mathcal{A} = \mathcal{A}_n \cup \mathcal{A}_p$ 是所有动作集合。

随机通信系统演算的语法如下：

$$S, T \ ::= X \ \Big| \ \sum_{i\in I} \alpha_i.T_i \ \Big| \ \mu X.T \ \Big| \ \bigoplus_{i\in I} p_i\tau.T_i \ \Big| \ S \mid T \ \Big| \ (a)T \tag{2–3}$$

在 (2–3) 中，

- X 表示一个变量；
- $\sum_{i\in I} \alpha_i.T_i$ 表示一个非确定项；
- $\mu X.T$ 表示一个不动点项；
- $\bigoplus_{i\in I} p_i\tau.T_i$ 表示一个概率项；
- $S \mid T$ 表示 S 和 T 的并行复合；
- $(a)T$ 表示限制名操作。

在非确定项和概率项中，下标集合 I 是有限的。我们在文中也会采用 $\alpha_1.T_1 + \alpha_2.T_2 + \alpha_3.T_3$ 以及 $p_1\tau.T_1 \oplus p_2\tau.T_2 \oplus p_3\tau.T_3$ 的写法。在概率项中，我们要求 $\sum_{i\in I} p_i = 1$。在非确定项中，如果集合 I 为空，那么该非确定项可以记作 $\mathbf{0}$。通常我们会省略进程中的 $\mathbf{0}$。为了后文叙述的方便，我们定义如下符号表示从一个项出发可以做的所有概率动作的集合：

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

如果一个变量 X 出现在 $\sum_{i\in I} \alpha_i.T_i$ 或 $\bigoplus_{i\in I} p_i\tau.T_i$ 中，我们称变量 X 是**被守卫的**（guarded）变量，这些变量需要做一些动作之后才能够被看到。出现在可见动作之后的变量是**强守卫的**（strongly guarded），这些变量需要做一些可见动作之后才能够被看到。我们要求在不动点项 $\mu X.T$ 中，所有在 T 中出现的变量 X 都是被守卫的。如果一个变量 X 没有出现在不动点算子 μX 之后，我们称其为**自由变**

量。我们用 fv(T) 来表示 T 中的所有自由变量的集合。如果一个项 T 中不包含自由变量，那么 T 是一个进程（process）。

我们定义 RCCS 的操作语义。令 $\lambda \in \mathcal{A}$：

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

$$\frac{S \xrightarrow{\overline{\alpha}} S', T \xrightarrow{\alpha} T'}{S|T \xrightarrow{\tau} S'|T'} \qquad \frac{T \xrightarrow{\lambda} T'}{S|T \xrightarrow{\lambda} S|T'} \qquad \frac{S \xrightarrow{\lambda} S'}{S|T \xrightarrow{\lambda} S'|T} \tag{2–4}$$

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

因为 CCS 是图灵完备的 [11]，RCCS 作为一个在 CCS 上的扩展，自然也是图灵完备的。所以我们无法判定其等价性问题。通用的做法是考虑有限状态的进程演算上的等价性验证问题 [99, 100]。我们将在语法 (2–3) 中去掉并行复合操作和限制名操作。有限状态的随机通信系统演算（记作 RCCS$_{fs}$）的语法如下：

$$S, T \ ::= X \ \Big|\ \sum_{i\in I}\alpha_i.T_i \ \Big|\ \mu X.T \ \Big|\ \bigoplus_{i\in I} p_i\tau.T_i \tag{2–5}$$

相应的，语义规则可以简化为：

$$\frac{}{\sum_{i\in I}\alpha_i.T_i \xrightarrow{\alpha_i} T_i} \qquad \frac{}{\bigoplus_{i\in I} p_i\tau.T_i \xrightarrow{p_i\tau} T_i} \qquad \frac{T\{\mu X.T/X\} \xrightarrow{\lambda} T'}{\mu X.T \xrightarrow{\lambda} T'} \tag{2–6}$$

在 RCCS$_{fs}$ 中，我们依旧沿用 RCCS 中的符号规范。我们分别用符号 $\mathcal{P}_{fs}$ 和 $\mathcal{S}_{fs}$ 表示 RCCS$_{fs}$ 中所有的进程集合和项集合。如果一个项可以立即做一个非确定动作，我们称其为非确定项，我们用 $\mathcal{S}_n$ 表示所有的非确定项的集合。如果一个项可以立即做一个概率动作，我们称其为概率项，我们用 $\mathcal{S}_r$ 表示所有的概率项的集合。单个变量 X 称为变量项，我们用 $\mathcal{S}_v$ 表示所有的变量项的集合。显然我们有 $\mathcal{S}_{fs} = \mathcal{S}_n \cup \mathcal{S}_r \cup \mathcal{S}_v$。

2.4 互模拟等价

2.4.1 下推自动机上的互模拟等价

本小节我们将会介绍下推自动机上的强互模拟，弱互模拟以及分支互模拟等价。首先，我们给出强互模拟等价的定义 [10]。

定义 2.4 (强互模拟等价) 给定一个进程集合 $\mathcal{P}$ 上的二元等价关系 $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$。如果对于所有的 $(P, Q) \in \mathcal{R}$ 都满足以下性质：

- 如果 $P \xrightarrow{\ell} P'$，那么存在迁移规则 $Q \xrightarrow{\ell} Q'$，并且满足 $(P', Q') \in \mathcal{R}$；
- 如果 $Q \xrightarrow{\ell} Q'$，那么存在迁移规则 $P \xrightarrow{\ell} P'$，并且满足 $(P', Q') \in \mathcal{R}$。

那么我们称关系 $\mathcal{R}$ 是一个强互模拟关系。

如果对于进程 P, Q，存在一个强互模拟关系 $\mathcal{R}$ 使得 $(P, Q) \in \mathcal{R}$，我们称 P 和 Q 是强互模拟的，记作 $P \sim Q$。我们用 $\sim$ 表示所有强互模拟关系的并集，$\sim$ 是最大的强互模拟关系。

可以看到，在强互模拟的定义中，内部动作 τ 和其他动作没有区别。因此在后文中，当我们讨论强互模拟时，也可以不失一般性的假定系统中没有 τ 动作，即系统是一个实时系统。这个假定不会对问题本身造成实质性的影响。

对于两个不互模拟的进程，我们可以定义其*等价步数*（Equivalence Level）的概念。比如在例2.1中，进程 pX 和 qX 在一步之内就可以看出它们不相等：进程 pX 可以做动作 b 而进程 qX 不可以。而要说明进程 pX 和 pXX 不等则至少需要两步：

1. 首先进程 pX 执行迁移规则 $pX \xrightarrow{b} q\epsilon$，进程 pXX 可以通过 $pXX \xrightarrow{b} qX$ 模拟；
2. 可以很容易看到第一步的结果 $q\epsilon$ 和 qX 不互模拟。

我们可以定义其等价步数来刻画说明一对进程不等至少需要经过多少步。首先我们递归的定义一个进程集合 $\mathcal{P}$ 上的关系 $\sim_i$，$i \in \mathbb{N}$：

- $\sim_0 \stackrel{\text{def}}{=} \mathcal{P} \times \mathcal{P}$；
- 如果对于一对进程 $P, Q \in \mathcal{P}$，满足以下两点：
  - 所有从 P 起始的迁移规则 $P \xrightarrow{a} P'$，存在 $Q \xrightarrow{a} Q'$，并且满足 $(P', Q') \in \sim_i$；
  - 所有从 Q 起始的迁移规则 $Q \xrightarrow{a} Q'$，存在 $P \xrightarrow{a} P'$，并且满足 $(Q', P') \in \sim_i$。

  那么 $(P, Q) \in \sim_{i+1}$。

由以上的定义，我们可以看到：$\sim = \bigcap_{i \in \mathbb{N}} \sim_i$。对于一对进程 $P, Q \in \mathcal{P}$，如果 $P \nsim Q$，我们定义进程 P 和 Q 的等价步数（用 EL 表示）：

$$\mathsf{EL}(P, Q) \stackrel{\text{def}}{=} \max\{i : P \sim_i Q\}$$

如果 $P \sim Q$，我们令 $\mathsf{EL}(P, Q) \stackrel{\text{def}}{=} \infty$。根据此定义，在例2.1中，$\mathsf{EL}(pX, qX) = 0$，$\mathsf{EL}(pX, pXX) = 1$。

类似于命题2.1，我们有如下命题：

命题 2.2 对于一对进程 $P,Q\in\mathcal{P}$，如果 $0<\mathsf{EL}(P,Q)<\infty$，那么：

1. 存在一条迁移规则 $P\xrightarrow{\ell}P'$（或者 $Q\xrightarrow{\ell}Q'$），使得对于任意的 $Q\xrightarrow{\ell}Q'$（或者 $P\xrightarrow{\ell}P'$），都满足 $\mathsf{EL}(P',Q')\leq\mathsf{EL}(P,Q)-1$，
2. 对于任意的迁移规则 $P\xrightarrow{\ell}P'$（或者 $Q\xrightarrow{\ell}Q'$），都存在 $Q\xrightarrow{\ell}Q'$（或者 $P\xrightarrow{\ell}P'$），使得 $\mathsf{EL}(P',Q')\geq\mathsf{EL}(P,Q)-1$。

同样的，类似于定义2.3，我们也可以定义等价步数下降迁移规则对：

定义 2.5 (等价步数下降的迁移规则对) 对于一对进程 $P\backsim Q$，且 $\mathsf{EL}(P,Q)>0$，我们令 $r_1=P\xrightarrow{\ell}P'$（或者 $r_1=Q\xrightarrow{\ell}Q'$）为满足命题2.2.1 的迁移规则，令 $r_2=Q\xrightarrow{\ell}Q'$（或者 $r_2=P\xrightarrow{\ell}P'$）为确定了 r_1 之后，满足命题2.2.2 的迁移规则。我们称 (r_1,r_2) 为从 (P,Q) 起始的*等价步数下降的迁移规则对*。

根据命题2.2，我们有 $\mathsf{EL}(P',Q')=\mathsf{EL}(P,Q)-1$。我们可以不断的寻找下一对进程，从而得到两条长度为 $\mathsf{EL}(P,Q)$ 的路径。

定义 2.6 (等价步数下降路径) 对于路径：

$$\pi:(P_1,Q_1)\xrightarrow{\ell_1}(P_2,Q_2)\xrightarrow{\ell_2}\cdots\xrightarrow{\ell_{k-1}}(P_k,Q_k)$$

对于 $1\leq i\leq k-1$，我们令 $r_i=P_i\xrightarrow{\ell_i}P_{i+1}$，$r_i'=Q_i\xrightarrow{\ell_i}Q_{i+1}$。如果对于任意 $1\leq i\leq k-1$，(r_i,r_i') 是 (P_i,Q_i) 起始的等价步数下降的迁移规则对，那么我们称路径 π 为 (P_1,Q_1) 起始的*等价步数下降路径*。

如果 $P\backsim Q$，我们可以找到一条从 (P,Q) 起始的长度为 $\mathsf{EL}(P,Q)$ 的等价步数下降路径。

除了强互模拟外，另一个受到广泛关注的互模拟关系是弱互模拟 [11]。弱互模拟等价又叫做观测等价（observational equivalence），即只关注系统可以被外界观测到的动作。我们用 $\Longrightarrow$ 表示 $\xrightarrow{\tau}$ 的自反传递闭包。对于可见动作 a，我们用 $\overset{a}{\Longrightarrow}$ 表示一串动作 $\Longrightarrow\xrightarrow{a}\Longrightarrow$。我们给出弱互模拟的定义。

定义 2.7 (弱互模拟等价) 给定一个进程集合 $\mathcal{P}$ 上的二元关系 $\mathcal{R}\subseteq\mathcal{P}\times\mathcal{P}$。如果对于所有的 $(P,Q)\in\mathcal{R}$ 都满足以下性质：

- 如果 $P\xrightarrow{a}P'$，那么存在 Q' 使得 $Q\overset{a}{\Longrightarrow}Q'$，并且满足 $(P',Q')\in\mathcal{R}$；
- 如果 $P\xrightarrow{\tau}P'$，那么存在 Q' 使得 $Q\Longrightarrow Q'$，并且满足 $(P',Q')\in\mathcal{R}$；

- 如果 $Q \xrightarrow{a} Q'$，那么存在 P' 使得 $P \overset{a}{\Longrightarrow} P'$，并且满足 $(Q', P') \in \mathcal{R}$。
- 如果 $Q \xrightarrow{\tau} Q'$，那么存在 P' 使得 $P \Longrightarrow P'$，并且满足 $(Q', P') \in \mathcal{R}$。

那么我们称关系 $\mathcal{R}$ 是一个弱互模拟关系。

如果对于进程 P, Q，存在一个弱互模拟关系 $\mathcal{R}$ 使得 $(P, Q) \in \mathcal{R}$，我们称 P 和 Q 是弱互模拟的，记作 $P \approx Q$。我们用 $\approx$ 表示所有弱互模拟关系的并集，$\approx$ 是最大的弱互模拟关系。

弱互模拟对于内部动作没有任何限制。为了更精确的刻画内部动作的行为，van Glabeek 提出了分支互模拟 [12]。分支互模拟要求只有不改变状态的内部动作可以忽略，而改变状态的内部动作还是要被模拟的，我们给出它的严格定义：

定义 2.8 (分支互模拟等价) 给定一个进程集合 $\mathcal{P}$ 上的二元对称关系 $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$。如果对于所有的 $(P, Q) \in \mathcal{R}$ 都满足以下性质：

- 如果 $P \xrightarrow{a} P'$，那么存在 Q' 使得 $Q \Longrightarrow Q'' \xrightarrow{a} Q'$，并且满足 $(P', Q') \in \mathcal{R}$，$(P, Q'') \in \mathcal{R}$；
- 如果 $P \xrightarrow{\tau} P'$，那么
  - 或者 $(P', Q) \in \mathcal{R}$；
  - 或者存在 Q' 使得 $Q \Longrightarrow Q'' \xrightarrow{\tau} Q'$，并且满足 $(P', Q') \in \mathcal{R}$，$(P, Q'') \in \mathcal{R}$。

那么我们称关系 $\mathcal{R}$ 是一个弱互模拟关系。

如果对于进程 P, Q，存在一个分支互模拟关系 $\mathcal{R}$ 使得 $(P, Q) \in \mathcal{R}$，我们称 P 和 Q 是分支互模拟的，记作 $P \simeq Q$。我们用 $\simeq$ 表示所有分支互模拟关系的并集，$\simeq$ 是最大的分支互模拟关系。

2.4.2 互模拟等价的博弈刻画

互模拟等价关系有一个非常自然的博弈刻画。本节我们将以强互模拟等价为例介绍其博弈刻画。

给定一对进程 (P_0, Q_0)，可以定义 (P_0, Q_0) 上的强互模拟博弈。该博弈有两个玩家，攻击者（Attacker）和防御者（Defender）。攻击者的目标是证明进程 P_0 和 Q_0 不等，防御者的目标是证明进程 P_0 和 Q_0 相等。这个博弈将进行若干轮。假设在第 i 轮开始时，当前的格局为 (P_{i-1}, Q_{i-1})，接下来博弈按照以下步骤进行：

1. 攻击者可以从 P_{i-1} 和 Q_{i-1} 中选择一个进程，并从该进程出发选择一个迁移规则 $P_{i-1} \xrightarrow{a_{i-1}} P_i$（或 $Q_{i-1} \xrightarrow{a_{i-1}} Q_i$），如果攻击者没有可以选择的动作，防御者赢得博弈；

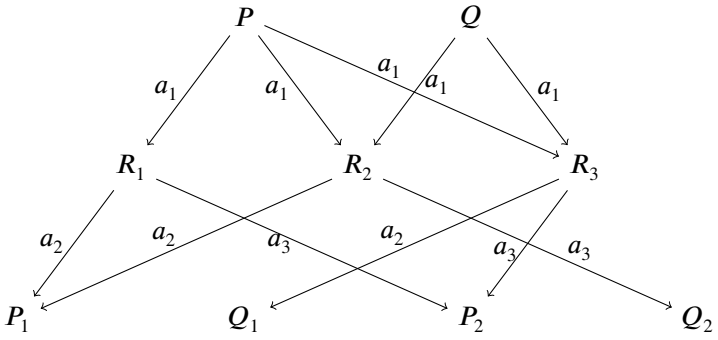


图 2–2 防御者力迫技术

2. 第 1 步结束后，防御者需要在另一个进程上做出一个对应的迁移规则 $Q_{i-1} \xrightarrow{a_{i-1}} Q_i$（或 $P_{i-1} \xrightarrow{a_{i-1}} P_i$），如果防御者无法模拟攻击者的动作，那么攻击者赢得博弈；
3. 如果两位玩家都能做出选择，博弈将以格局 (P_i, Q_i) 开始第 $i+1$ 轮。我们可以用符号 $(P_{i-1}, Q_{i-1}) \xrightarrow{a_{i-1}} (P_i, Q_i)$ 来表示这一轮的两个动作。

如果博弈能够一直进行下去，那么防御者赢得博弈。

命题 2.3 对于一对进程 P 和 Q，$P \sim Q$ 当且仅当防御者在 (P, Q) 上的强互模拟博弈有必胜策略。

回顾互模拟博弈的定义，我们发现整个博弈一直是由攻击者来主导。攻击者主动的去选择一个进程和它要做的迁移规则，而防御者要做的事情就是模拟这个迁移规则。然而，我们可以引入一些规则来使防御者也能在某些时候占据主动权。这个技术称作防御者力迫技术 [101]。这个技术在证明互模拟等价的不可判定性以及复杂性下界方面取得了很多成功 [54, 72, 78]。

图2–2的例子说明了防御者力迫技术的思想。假设博弈从进程 (P, Q) 开始，攻击者希望证明进程 P 和进程 Q 不互模拟。

1. 在第 1 轮中，攻击者唯一可以选择的迁移规则是 $P \xrightarrow{a_1} R_1$，如果攻击者选择其他迁移规则，那么防御者总可以使当前格局的两个进程相同。当格局的两个进程相同时，防御者只要跟随攻击者的选择就一定可以获得博弈的胜利。
2. 在攻击者选择 $P \xrightarrow{a_1} R_1$ 后，防御者可以有以下两个选择：
   - 如果防御者选择 $Q \xrightarrow{a_1} R_2$，那么博弈第 2 轮的格局为 (R_1, R_2)。此时，攻击者将被迫选择 $R_1 \xrightarrow{a_3} P_2$ 或者 $R_2 \xrightarrow{a_3} Q_2$，防御者选择相应的迁移规则，博弈进入 (P_2, Q_2)；

- 如果防御者选择 $Q \xrightarrow{a_1} R_3$，那么博弈第 2 轮的格局为 (R_1, R_3)。此时，攻击者将被迫选择 $R_1 \xrightarrow{a_2} P_1$ 或者 $R_3 \xrightarrow{a_2} Q_1$，防御者选择相应的迁移规则，博弈进入 (P_1, Q_1)。

综合以上分析，防御者在该博弈中可以主动的选择让博弈进入 (P_1, Q_1) 或者 (P_2, Q_2)，因此防御者只要能够赢得其中一个博弈，他就能赢得整个博弈。我们有以下引理：

引理 2.4 在图2–2中，$P \sim Q$ 当且仅当 $P_1 \sim Q_1$ 或者 $P_2 \sim Q_2$。

2.4.3 概率系统上的分支互模拟等价

本节中，我们将会回顾 Fu 在文章 [22] 中定义的随机通信系统演算上的分支互模拟等价关系。由于本节的很多定义技术性较强，我们会在附录A中给出一些例子，读者可以对照例子理解本节中的定义。

我们回顾一般模型上的分支互模拟定义（定义2.8），要定义随机通信系统演算上的分支互模拟，有以下两个关键点：

1. 如何将一般模型上的一串连续的不改变状态的内部动作[1]$\Longrightarrow$ 推广到概率模型上；
2. 相对于一般模型只有一种非确定的动作，概率模型有概率动作和非确定动作两种完全不同的动作，该如何定义一步动作的对应。

首先对于第一点，因为概率模型中涉及到了概率分支，因此，一条路径 $P \Longrightarrow$ 可能会分叉成一棵以 P 为根节点的树。从树上的每个非叶子节点出发，可能会是一组概率动作 $\coprod_{i\in I} p_i\tau$，也可能是一个非确定动作 τ。而相对应地，我们会要求整棵树上的动作都不改变状态。接下来，我们将会介绍 ϵ-树 [22] 的定义，直观上，ϵ-树就是路径 $\Longrightarrow$ 在概率模型上的推广。

对于一个 $\mathcal{P}_{fs}$ 上的等价关系 $\mathcal{R}$，我们用 $A\mathcal{R}B$ 表示 $(A, B) \in \mathcal{R}$. 符号 $\mathcal{P}_{fs}/\mathcal{R}$ 表示由等价关系 $\mathcal{R}$ 划分的等价类。$[A]_{\mathcal{R}}$ 表示包含 A 的等价类。

定义 2.9 (ϵ-树 [22]) 给定一个进程 $A \in \mathcal{P}_{fs}$ 和一个 $\mathcal{P}_{fs}$ 上的二元关系 $\mathcal{R}$，一个 A 关于 $\mathcal{R}$ 的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$（*An ϵ-tree of A with regard to $\mathcal{R}$*）是一个满足下列条件的带标记的树。

- 树 $\mathcal{T}_{\mathcal{R}}^{A}$ 上的每一个节点都被 $[A]_{\mathcal{R}}$ 中的元素标记，根结点被 A 标记。
- 树的边被一个实数 i 标记，$0 < i \leq 1$.

[1]这里所说的动作 $P \xrightarrow{\tau} P'$ 不改变状态，是指 P 和 P' 在同一个等价类中。

- 如果存在一条从节点 B 到节点 B' 的边由实数 p 标记，$0 < p < 1$，那么存在概率迁移规则
$$B \xrightarrow{\coprod_{1\le i\le k} p_i\tau} \coprod_{1\le i\le k} B_i$$
使得对于任意的 i 满足 $1 \le i \le k$，$\mathcal{T}_{\mathcal{R}}^{A}$ 中存在一条从 B 到 B_i 的标记为 p_i 的边，并且 $B_1, \cdots, B_k$ 是 B 的所有子节点。
- 如果存在一条从节点 B 到节点 B' 的边由实数 1 标记，那么存在非确定迁移规则 $B \xrightarrow{\tau} B'$，并且在 $\mathcal{T}_{\mathcal{R}}^{A}$ 中，B' 是 B 的唯一子节点。

后文中，如果 A 和 $\mathcal{R}$ 不做强调，我们也会把 $\mathcal{T}_{\mathcal{R}}^{A}$ 简写为 $\mathcal{T}$。

对于前面所说的第二个关键点，即概率模型中有两种不同的动作：非确定动作和概率动作，文章 [22] 分别引入了 ℓ-迁移和 q-迁移两个概念来对应这两种情况。在这个定义中，需要注意的几点有，

- 因为概率动作都是内部动作，如果一组概率动作 $\coprod_{i\in I} p_i\tau$ 全部都不改变状态，那么就类似于一般模型中的一步不改变状态的 τ 动作。这样一步动作是不需要被模拟的。因此 q-迁移定义中的一个条件是要求存在改变状态的概率 τ 动作。
- 在一组概率 τ 动作中，可能有一部分是改变状态的，一部分是不改变状态的。例如图2–3中的例子，假设图中的动作除了自环都是改变状态的。虽然 E_1 比 E_2 有更大的概率做不改变状态的动作回到自身，但是最终它们都会以各一半的概率进入 F_1 和 F_2。因此 E_1 和 E_2 应该是等价的。所以在计算概率时，我们需要计算一个条件概率。如果在改变状态的条件下，两个进程以相同的概率进入相同的状态，那么两个进程应该被认为是等价的。

接下来，我们逐步给出 ℓ-迁移和 q-迁移的严格定义。在一棵 ϵ-树 $\mathcal{T}$ 上，一个**分支**（branch）π 可以是一条从根节点到叶节点的路径或者一条无限路径。我们定义 $|\pi|$ 为分支中边的数量；如果 π 是一条无限路径，则定义 $|\pi| = \infty$。对于正整数 $1 \le i \le |\pi|$，$\pi(i)$ 表示 π 的第 i 条边。我们定义 π 的概率：

$$P(\pi) \stackrel{\text{def}}{=} \begin{cases} 1 & |\pi| = 0 \\ \prod_{i\le|\pi|} \pi(i) & 0 < |\pi| < \infty \\ \lim_{k\to\infty} \prod_{i\le k} \pi(i) & |\pi| = \infty \end{cases}$$

给定一个 ϵ-树 $\mathcal{T}$，其有限分支的概率定义为：

$$P^f(\mathcal{T}) \stackrel{\text{def}}{=} \lim_{k\to\infty} \sum \left\{ P(\pi) \mid \pi \text{ 是 } t \text{ 中的一个有限的分支，使得} |\pi| \le k \right\} \qquad (2\text{–}7)$$

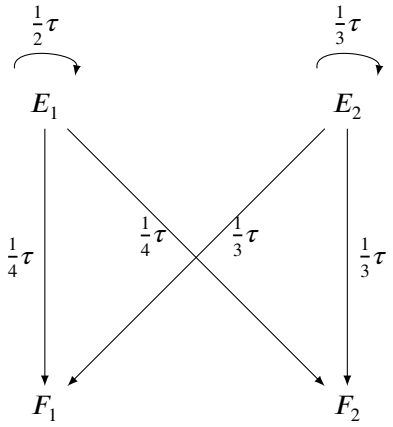


图 2–3 随机通信系统演算分支互模拟的例子

对于一个 ϵ-树 $\mathcal{T}^A_{\mathcal{R}}$，如果 $P^f(\mathcal{T}^A_{\mathcal{R}}) = 1$，我们称它是正则的（regular）。

定义 2.10 (ℓ-迁移 [22]) 给定一个非确定动作 $\ell \in \mathcal{A}_n$，一个进程 $A \in \mathcal{P}_{fs}$，和一个 $\mathcal{P}_{fs}$ 上的等价关系 $\mathcal{R}$。令 $\mathcal{B} \in \mathcal{P}_{fs}/\mathcal{R}$ 为一个 $\mathcal{R}$ 划分的等价类。假定 $\ell \neq \tau \vee \mathcal{B} \neq [A]_{\mathcal{R}}$，一个从进程 A 到等价类 $\mathcal{B}$ 的关于 $\mathcal{R}$ 的 ℓ-迁移包含两部分：

- 一个关于 $\mathcal{R}$ 的正则的 ϵ-树 $\mathcal{T}^A_{\mathcal{R}}$；
- 从 $\mathcal{T}^A_{\mathcal{R}}$ 中的每个叶子 L 出发都有的一条迁移规则 $L \xrightarrow{\ell} L' \in \mathcal{B}$。

如果存在一个从进程 A 到等价类 $\mathcal{B}$ 的关于 $\mathcal{R}$ 的 ℓ-迁移，我们记作 $A \rightsquigarrow_{\mathcal{R}} \xrightarrow{\ell} \mathcal{B}$。

直观上，ℓ-迁移就是经典模型中的路径 $P \Longrightarrow \xrightarrow{\ell} Q$ 在概率模型上的推广。

接下来，我们定义 q-迁移。首先假定 $L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]} L_i$，其中，符号 $[k] \stackrel{\text{def}}{=} \{1, 2, \cdots, k\}$。如果在 1 到 k 中存在至少一个 i，满足 $[L_i]_{\mathcal{R}} \neq [L]_{\mathcal{R}}$。我们可以定义：

$$P(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) \stackrel{\text{def}}{=} \sum \left\{ p_i : L \xrightarrow{p_i\tau} L_i \in \mathcal{B}, 0 \leq i \leq k \right\}$$

定义条件概率：

$$P_{\mathcal{R}}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) = \frac{P(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B})}{1 - P(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} [L]_{\mathcal{R}})}$$

定义 2.11 (q-迁移 [22]) 给定一个实数 $0 < q \leq 1$，一个进程 $A \in \mathcal{P}_{fs}$ 和一个 $\mathcal{P}_{fs}$ 上的等价关系 $\mathcal{R}$。令 $\mathcal{B} \in \mathcal{P}_{fs}/\mathcal{R}$ 为一个 $\mathcal{R}$ 划分的等价类。一个从进程 A 到等价类 $\mathcal{B}$ 的关于 $\mathcal{R}$ 的 q-迁移包含两部分：

- 一个关于 $\mathcal{R}$ 的正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$；
- 从 $\mathcal{T}_{\mathcal{R}}^{A}$ 中的每一个叶子 L 出发，都有的一个概率迁移规则的集合

$$L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \coprod_{i\in[k]} L_i$$

使得

$$P_{\mathcal{R}}(L \xrightarrow{\coprod_{i\in[k]} p_i\tau} \mathcal{B}) = q$$

如果存在一个从进程 A 到等价类 $\mathcal{B}$ 的关于 $\mathcal{R}$ 的 q-迁移，我们记作 $A \rightsquigarrow_{\mathcal{R}} \xrightarrow{q} \mathcal{B}$。

直观上，q-迁移描述了在执行了一些不改变状态的内部动作之后，在保证改变状态的前提下，有 q 的条件概率达到另外一个等价性类。

定义 2.12 (Fu [22]) $\mathcal{R}$ 是进程集合 $\mathcal{P}_{fs}$ 上的一个等价关系，如果以下两点成立：

1. 如果 $B\mathcal{R}A \rightsquigarrow_{\mathcal{R}} \xrightarrow{\ell} C \in \mathcal{P}/\mathcal{R}$，使得 $\ell \neq \tau \vee C \neq [A]_{\mathcal{R}}$，那么 $B \rightsquigarrow_{\mathcal{R}} \xrightarrow{\ell} C$.
2. 如果 $B\mathcal{R}A \rightsquigarrow_{\mathcal{R}} \xrightarrow{q} C \in \mathcal{P}/\mathcal{R}$ 使得 $C \neq [A]_{\mathcal{R}}$，那么 $B \rightsquigarrow_{\mathcal{R}} \xrightarrow{q} C$.

我们称 $\mathcal{R}$ 是一个 $\mathcal{P}_{fs}$ 上的*分支互模拟*。

$\mathcal{P}_{fs}$ 上最大的分支互模拟记作 $\simeq_{\mathrm{RCCS}_{fs}}$。在不引起歧义的情况下，我们也会简写作 $\simeq$。文章 [22] 中的定理 17 证明了：

命题 2.5 (Fu [22]) $\simeq_{\mathrm{RCCS}_{fs}}$ 是一个同余关系。

2.5 快速增长复杂性类

因为很多验证问题本身是非常困难的，需要引入合适的复杂性类来描述它们。本文我们引入 Schmitz 定义的一个以序数为下标的复杂性类层级 [23]：

$$\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \cdots, \mathbf{F}_\omega, \mathbf{F}_{\omega+1}, \cdots \mathbf{F}_{\omega^2}, \cdots \mathbf{F}_{\omega^\omega} \cdots$$

其中 $\mathbf{F}_3 = \text{TOWER}$ 是最小的非初等复杂性类（Non-elementary）；$\mathbf{F}_\omega = \text{ACKERMANN}$ 是最小的非原始递归（Non-primitive recursive）复杂性类。

我们对上述复杂性类层级的定义做一个简要回顾。Schimtz 首先定义了一系列以序数为下标的函数 $F_\alpha : \mathbb{N} \to \mathbb{N}$：

$$F_0(x) \overset{\text{def}}{=} x+1, \qquad F_{\alpha+1}(x) \overset{\text{def}}{=} \underbrace{F_\alpha(\cdots(F_\alpha}_{x+1\text{ 次}}(x))\cdots), \qquad F_\lambda(x) \overset{\text{def}}{=} F_{\lambda(x)}(x) \tag{2–8}$$

其中 $\alpha+1$ 表示一个后继序数，λ 表示一个极限序数。$\lambda(x)$ 采用标准的基本序列（fundamental sequence）的定义：

$$(\gamma+\omega^{\beta+1})(x) \overset{\text{def}}{=} \gamma+\omega^{\beta}\cdot(x+1), \qquad (\gamma+\omega^{\lambda})(x) \overset{\text{def}}{=} \gamma+\omega^{\lambda(x)}$$

我们可以尝试计算这列函数的前面几项：$F_1(x) \overset{\text{def}}{=} 2x+1$，$F_2(x) \overset{\text{def}}{=} 2^{x+1}(x+1)-1$ 等等。以此列函数为基础，可以定义复杂性类：

$$\mathbf{F}_{\alpha} \overset{\text{def}}{=} \bigcup_{p\in\mathcal{F}_{<\alpha}} \mathrm{DTIME}(F_{\alpha}(p(n))).$$

其中，函数 p 是一个 $\mathcal{F}_{<\alpha}$ 中的函数。$\mathcal{F}_{<\alpha}$ 的定义这里不再展开。粗略地说，p 是一个比 F_{α} 增长严格慢的函数。

此外，Schmitz 定义了相对层级（relativised hierarchies）。给定一个单调递增函数 $h:\mathbb{N}\to\mathbb{N}$，可以定义相对函数：

$$F_{h,0}(x) \overset{\text{def}}{=} h(x), \qquad F_{h,\alpha+1}(x) \overset{\text{def}}{=} \underbrace{F_{h,\alpha}(\cdots(F_{h,\alpha}}_{x+1\text{次}}(x))\cdots), \qquad F_{h,\lambda}(x) \overset{\text{def}}{=} F_{h,\lambda(x)}(x) \tag{2–9}$$

类似的，可以定义相对复杂性类：

$$\mathbf{F}_{h,\alpha} \overset{\text{def}}{=} \bigcup_{p\in\mathcal{F}_{<\alpha}} \mathrm{DTIME}(F_{h,\alpha}(p(n)))$$

我们介绍如下引理来结束本节：

引理 2.6 (Schmitz [23]) h 是一个严格递增函数，α,β 是两个序数。如果 $h\le\mathbf{F}_{\beta}$，那么 $\mathbf{F}_{h,\alpha}\subseteq\mathbf{F}_{\beta+\alpha}$。

第三章　下推自动机互模拟等价下界

本章中，我们将会证明如下定理。

定理 3.1　判定下推自动机的强互模拟等价是 ACKERMANN-难的。状态数给定为 $d \geq 4$ 的下推自动机的强互模拟等价问题是$\mathbf{F}_{d-1}$-难的。

在3.1节中，我们将介绍重置 Petri 网以及该模型上可覆盖问题的定义，这个问题是一个 ACKERMANN-难问题。在3.2节中，我们将重置 Petri 网的可覆盖问题归约到下推自动机的强互模拟等价问题，从而证明定理3.1。这个归约的思路类似于 Jančar 证明一阶文法的互模拟是 ACKERMANN-难的思路 [39]。在3.3节中，我们将讨论 PDA 和一阶文法的关系，并证明一阶文法在强互模拟的意义上比 PDA 严格的强。3.4节对本章内容做出总结。

3.1　重置 Petri 网

定义 3.1 (重置 Petri 网)　一个*重置 Petri 网* $\mathcal{N}$ 是一个三元组 $(S, C, \mathcal{R})$。其中，

- S 是一个有限的状态集合；
- $C = \{c_1, c_2, \cdots, c_d\}$ 是一个有限的计数器集合；
- $\mathcal{R} \subseteq S \times \mathcal{O} \times S$ 是一个有限的规则集合，其中

$$\mathcal{O} = \left\{ \mathsf{INC}(c_i), \mathsf{DEC}(c_i), \mathsf{RESET}(c_i) : i = 1, 2, \cdots, d \right\}$$

是对计数器的所有操作的集合。

重置 Petri 网在某一时刻的*格局*形如 $(s, n_1, n_2, \cdots, n_d)$，其中 $s \in S$ 表示当前的状态，$n_1, n_2, \cdots, n_d$ 分别表示当前 d 个计数器的值。如果 $\mathcal{R}$ 中包含一条规则 (s, op, t)，在满足下列条件时，我们可以得到重置 Petri 网的格局之间的迁移 $(s, n_1, n_2, \cdots, n_d) \to (t, n'_1, n'_2, \cdots, n'_d)$：

- 如果 $op = \mathsf{INC}(c_i)$，那么 $n'_i = n_i + 1$，且对于 $j \neq i$，有 $n'_j = n_j$；
- 如果 $op = \mathsf{DEC}(c_i)$，$n_i > 0$，那么 $n'_i = n_i - 1$，且对于 $j \neq i$，有 $n'_j = n_j$；
- 如果 $op = \mathsf{RESET}(c_i)$，那么 $n'_i = 0$，且对于 $j \neq i$，有 $n'_j = n_j$。

我们用 $\to^*$ 表示 $\to$ 的自反传递闭包。

我们可以在重置 Petri 网的格局上定义偏序关系 $\leq$：

$$(s, n_1, n_2, \cdots, n_d) \leq (s', n'_1, n'_2, \cdots, n'_d) \quad \text{当且仅当} \quad s = s', n_i \leq n'_i, 1 \leq i \leq n$$

对于格局 σ_1, σ_2，如果存在格局 σ 使得 $\sigma_1 \to^* \sigma$，$\sigma_2 \leq \sigma$，那么我们称 σ_2 可以被 σ_1 覆盖。

重置 Petri 网的可覆盖问题
输入：一个重置 Petri 网 $\mathcal{N}$ 和两个格局 σ_1, σ_2。 问题：σ_2 是否可以被 σ_1 覆盖?

Schnoebelen 证明了重置 Petri 网的可覆盖问题是 ACKERMANN-难的 [102]；另一方面，通过对 Dickson 引理的复杂性分析，可以得到 ACKERMANN 的复杂性上界 [103]。结合这两个结果，我们可以得到如下定理：

定理 3.2 (Schnoebelen 等 [102, 103]) 判定重置 Petri 网的可覆盖问题是一个 ACKERMANN-完备问题。

重置 Petri 网中关键的参数是计数器的个数，如果固定计数器的个数 $d \geq 3$，可以得到一个参数复杂性结果。

定理 3.3 (Schmitz [104, 105]) 固定重置 Petri 网的计数器的个数 $d \geq 3$，可覆盖问题是一个$\mathbf{F}_d$-完备问题。

3.2 下推自动机互模拟下界分析

本节中，我们将会给出一个下推自动机互模拟 ACKERMANN-难的复杂性下界。重置 Petri 网的可覆盖问题是一个已知 ACKERMANN-难的问题，我们通过把重置 Petri 网的可覆盖问题归约到下推自动机的互模拟问题，从而给出下推自动机互模拟的 ACKERMANN-难的复杂性下界。

在 2014 年，Jančar 证明了如下定理：

定理 3.4 (Jančar[39]) 判定一阶文法的强互模拟等价是 ACKERMANN-难的。

本节中给出的证明类似于 Jančar 证明的思路，最主要的不同点在于我们采用了新的测零方式以避免引入内部动作。在3.3节中，我们还将进一步讨论一阶文法和下推自动机的关系。

3.2.1 证明思路

给定一个重置 Petri 网 RPN $\mathcal{N} = (S, C, \mathcal{R})$，一个初始格局 σ_s，一个目标格局 σ_f，我们将会构造一个下推自动机 $\mathcal{A} = (Q, \Gamma, \Sigma, \mathcal{R})$ 和 $\mathcal{A}$ 上的两个进程 P, Q，[1]使得下述命题成立：

$$\sigma_s \text{ 覆盖 } \sigma_f \quad \text{当且仅当} \quad P \nsim Q$$

回顾在2.4.2中介绍的互模拟的博弈刻画，在对应的互模拟博弈中，攻击者将会试图证明 σ_s 可以覆盖 σ_f，防御者试图证明 σ_s 不能覆盖 σ_f。

我们的归约有以下几个关键点。

- 对于 $\mathcal{N}$ 中的一条路径

  $$\sigma_s \to \sigma_1 \to \sigma_2 \to \cdots \to \sigma_k$$

  互模拟博弈的每一轮都能够与其对应：

  $$(P, Q) \xrightarrow{a_1} (P_1, Q_1) \xrightarrow{a_2} (P_2, Q_2) \xrightarrow{a_3} \cdots \xrightarrow{a_k} (P_k, Q_k)$$

  其中，$k \in \mathbb{N}$。我们将会把从起始格局 σ_s 到某个格局 σ_i 中经历的所有对计数器的操作直接记录在 P_i, Q_i 的栈中。同时将第 σ_i 的状态记录在 P_i 和 Q_i 的栈顶。我们会用两个不同的栈顶符号来区分 P_i 和 Q_i。
- 在重置 Petri 网中，当一个计数器为 0 时就不能再做减法操作了。同样的，归约中也需要保证攻击者不能试图去执行一个不合法的减法操作。这一点将会借助防御者力迫技术的技术来实现。每当攻击者要做一个减法操作时，防御者都能够有一次测 0 的机会，如果当前操作的计数器的值为 0，那么防御者将会直接获得博弈胜利。通过这一点可以保证攻击者做的每一步操作都是合法的。
- 最后一点是覆盖性检查，当 PDA 栈顶记录的格局 σ_k 的状态和目标格局 σ_f 的状态一样时，攻击者有机会执行覆盖性检查，如果 $\sigma_k \geq \sigma_f$，我们构造的 PDA 中要使攻击者赢得博弈。我们会引入一个特殊的动作 f，当覆盖性检查通过后，博弈最终会进入一对进程 (P', Q')，其中 P' 可以做动作 f，但 Q' 不能做，由此攻击者可以赢得博弈。

[1]这个构造将会是一个指数时间构造，这足够证明我们的结果，因为 ACKERMANN-难在原始递归的归约下都是封闭的。此外，本节的归约也可以采用4.3.2节的技术修改为多项式时间的。

3.2.2 准备工作

当给定重置 Petri 网 $\mathcal{N}=(S,C,\mathcal{R})$，一个起点格局 $\sigma_s=(t_s,n_1,n_2,\cdots,n_d)$，一个终点格局 $\sigma_f=(t_f,m_1,m_2,\cdots,m_d)$ 后，我们将要构造 $\mathcal{A}=(Q,\Gamma,\Sigma,\mathcal{R})$ 和两个 $\mathcal{A}$ 上的进程 P,Q。

首先，我们引入如下 $d+1$ 个状态：

$$Q \stackrel{\text{def}}{=} \{p,q_1,q_2,\cdots,q_d\}$$

其中，$q_1,q_2,\cdots,q_d$ 分别对应于 C 中的 d 个计数器 $c_1,c_2,\cdots,c_d$。此外，我们还需要一个额外的状态 p，状态 p 将作为一个标准，用来帮助判断一个计数器是否为 0。

我们引入一些栈符号来记录计数器上的操作：

$$\Gamma_C \stackrel{\text{def}}{=} \{X_i^+,X_i^-,X_i^0 : i=1,2,\cdots,d\} \subseteq \Gamma$$

我们引入一些栈符号来记录 S 中的状态：

$$\Gamma_S \stackrel{\text{def}}{=} \{Y_s,Y_s' : s\in S\} \subseteq \Gamma$$

利用前面引入的状态以及栈符号，$\mathcal{N}$ 的初始格局 $\sigma_s=(t_s,n_1,n_2,\cdots,n_d)$ 可以被编码为一对进程：

$$P = pY_{t_s}(X_1^+)^{n_1}\cdots(X_d^+)^{n_d} \quad Q = pY_{t_s}'(X_1^+)^{n_1}\cdots(X_d^+)^{n_d}$$

在这对进程互模拟的过程中，我们会始终让栈顶符号去编码 $\mathcal{N}$ 当前的状态，栈中的其他符号记录了所有对计数器的操作。因此，通过栈中的符号我们就可以计算出当前计数器的值。我们定义一个映射 $\mathsf{count} : \Gamma_S^* \to \mathbb{Z}^d$。[1] 对于 $\alpha = X_nX_{n-1}\cdots X_1 \in \Gamma_S^*$，令 K_i 表示最大的满足 $X_j = X_i^0$ 的下标 j。如果 X_i^0 在 α 中没有出现，则令 $K_i=0$。令 I_i 和 D_i 分别表示 $X_nX_{n-1}\cdots X_{K_i+1}$ 中 X_i^+ 和 X_i^- 出现的次数，我们定义：

$$\mathsf{count}(\alpha) \stackrel{\text{def}}{=} (I_1-D_1,I_2-D_2,\cdots I_d-D_d) \tag{3–1}$$

直观上，函数 count 计算了 d 个计数器在经历了当前栈符号中记录的操作后，计数器的最终取值。我们定义 $\mathsf{count}_i(\alpha) \stackrel{\text{def}}{=} I_i-D_i$，即第 i 个计数器的最终取值。

例 3.1 $\mathsf{count}_1(X_2^+X_2^0X_1^-X_2^+X_1^+)=0$，$\mathsf{count}_2(X_2^+X_2^0X_1^-X_2^+X_1^+)=1$

[1] 事实上，重置 Petri 网中的计数器的值不能为负，定义域中的一些使计数器成为负数的符号串（例如 $X_1^-X_1^-X_1^+$）在模拟过程中是不会出现的。这里为了方便起见，我们让函数 count 定义在了所有 Γ_S^* 中的符号串上，同时值域定义在 $\mathbb{Z}^d$ 上而不是 $\mathbb{N}^d$ 上。

3.2.3 计数器操作

对于集合 $\mathcal{R}$ 中的规则，我们将会在下推自动机 $\mathcal{A}$ 中引入对应的规则，来保证互模拟博弈能够模拟 $\mathcal{N}$ 的运行。其中需要特别注意的是计数器的减法操作。我们首先定义一些规则来实现2.4.2小节中介绍的防御者力迫技术。为了使后文中的表述更加简洁，我们这里采用了 Benedikt 等人定义的宏规则的符号[72]。

我们定义宏规则 $(pX, qY) \xrightarrow{ATT} \{(p_1\alpha_1, q_1\beta_1), (p_2\alpha_2, q_2\beta_2)\}$ 表示如下迁移规则的集合：

$$pX \xrightarrow{a} p_1\alpha_1 \qquad qY \xrightarrow{a} q_1\beta_1$$
$$pX \xrightarrow{b} p_2\alpha_2 \qquad qY \xrightarrow{b} q_2\beta_2$$

这里动作 a 和 b 在 $\mathcal{A}$ 中的其他规则中都不会出现。在这条宏规则中，攻击者是占据主动的一方。他可以主动的选择动作 a 或者 b，从而使博弈进入 $(p_1\alpha_1, q_1\beta_1)$ 或者 $(p_2\alpha_2, q_2\beta_2)$。

有时我们不强调攻击者的选择权，我们引入宏规则 $(pX, qY) \xrightarrow{ATT} (p_1\alpha_1, q_1\beta_1)$ 表示如下迁移规则的集合：

$$pX \xrightarrow{a} p_1\alpha_1 \qquad qY \xrightarrow{a} q_1\beta_1$$

接下来，我们定义一个防御者能够占据主动的宏规则，我们用 $(pX, qY) \xrightarrow{DEF} \{(p_1\alpha_1, q_1\beta_1), (p_2\alpha_2, q_2\beta_2)\}$ 表示如下迁移规则的集合：

$$pX \xrightarrow{a_1} pZ_1 \quad pX \xrightarrow{a_1} pZ_2 \quad pX \xrightarrow{a_1} pZ_3$$
$$qY \xrightarrow{a_1} pZ_2 \quad qY \xrightarrow{a_1} pZ_3$$
$$pZ_1 \xrightarrow{a_2} p_1\alpha_1 \quad pZ_1 \xrightarrow{a_3} p_2\alpha_2 \quad pZ_2 \xrightarrow{a_2} p_1\alpha_1 \quad pZ_2 \xrightarrow{a_3} q_2\beta_2$$
$$pZ_3 \xrightarrow{a_2} q_1\beta_1 \quad pZ_3 \xrightarrow{a_3} p_2\alpha_2$$

这里的动作 a_1, a_2, a_3 和栈符号 Z_1, Z_2, Z_3 在 $\mathcal{A}$ 中的其他规则中都不会出现。这些规则就是图2–2的实现，可以看到防御者能够用这些规则迫使攻击者做出特定的动作，从而占据博弈的主动。

下面的引理说明了这些宏规则的作用。因为我们没有限制 (pX, qY) 是否还有其他动作，所以该引理的叙述不像引理2.4那么强。

引理 3.5 我们有以下两点成立：

- 引入宏规则 $(pX, qY) \xrightarrow{ATT} \{(p_1\alpha_1, q_1\beta_1), (p_2\alpha_2, q_2\beta_2)\}$ 之后，如果 $p_1\alpha_1\gamma \sim q_1\beta_1\gamma$ 或者 $p_2\alpha_2\gamma \sim q_2\beta_2\gamma$，那么 $pX\gamma \sim qY\gamma$。

- 引入宏规则 $(pX, qY) \xhookrightarrow{DEF} \{(p_1\alpha_1, q_1\beta_1), (p_2\alpha_2, q_2\beta_2)\}$ 之后，如果 $pX\gamma \sim qY\gamma$，那么 $p_1\alpha_1\gamma \sim q_1\beta_1\gamma$ 或者 $p_2\alpha_2\gamma \sim q_2\beta_2\gamma$。

借助于这些宏规则，我们可以在 PDA 中引入规则来模拟重置 Petri 网中的规则。对于规则 $(s, op, t) \in \mathcal{R}$，有以下几种情况：

- 如果 $op = \mathsf{INC}(c_i)$，引入宏规则 $(pY_s, pY_s') \xhookrightarrow{ATT} (pY_tX_i^+, pY_t'X_i^+)$；
- 如果 $op = \mathsf{RESET}(c_i)$，引入宏规则 $(pY_s, pY_s') \xhookrightarrow{ATT} (pY_tX_i^0, pY_t'X_i^0)$；
- 如果 $op = \mathsf{DEC}(c_i)$，引入宏规则 $(pY_s, pY_s') \xhookrightarrow{DEF} \{(pY_tX_i^-, pY_t'X_i^-), (p, q_i)\}$

通过引入的规则我们可以看到，当互模拟博弈的格局为 $(pY_s\alpha, pY_s'\alpha)$ 时，$\mathcal{N}$ 中的 $\mathsf{INC}(c_i)$ 和 $\mathsf{RESET}(c_i)$ 操作是可以随时进行的，PDA 直接将操作记录到栈中。但是当攻击者想去做一个 $\mathsf{DEC}(c_i)$ 操作时，防御者有机会将博弈引入另外一个分支 $(p\alpha, q_i\alpha)$。防御者选择进入这个分支的条件是计数器 c_i 当前的值为 0。我们将在下一小节中引入规则保证如下引理成立：

引理 3.6 对于 $\alpha \in \Gamma_S^*$，$q_i\alpha \sim p\alpha$ 当且仅当 $\mathsf{count}_i(\alpha) = 0$。

3.2.4 合法性检查

在对重置 Petri 网迁移路径的模拟过程中，我们将所有计数器上的所有操作全部放在栈上。因此为实现引理3.6，需要解决的一个重要问题是当验证某个计数器 i 的值时，如何屏蔽对其他计数器的操作，从而反映出某个特定的计数器的大小。这里我们将放弃符号之间的严格对应，而是将栈中的内容当作一个整体来考虑。为此，我们会要求进程在此后只能做一种动作 b_p，而进程之间是否互模拟就取决于其能够做的动作的总个数是否相等。

为了使后文中的表述更加简洁，我们引入一个新的宏规则 $pX \overset{u}{\hookrightarrow} q\alpha$，其中 $u = a_1, a_2, \cdots, a_n$，该宏规则表示如下的一系列连续的迁移规则的集合：

$$pX \xrightarrow{a_1} pX_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} pX_{n-1} \xrightarrow{a_n} q\alpha$$

其中，动作 $a_1, a_2, \cdots, a_n$ 和栈符号 $X_1, X_2, \cdots, X_{n-1}$ 都不会出现在 $\mathcal{A}$ 的其他规则中。

控制状态 p 将作为一个标准，我们引入如下规则：

$$pX \xhookrightarrow{b_pb_p} p \qquad \text{对于所有的} X \in \Gamma_C$$

在状态 p 下，所有的 Γ_C 中的栈符号在出栈的过程中都会做两个 b_p 动作。而对于

控制状态 q_i，我们引入如下规则：

$$
\begin{aligned}
&q_i X_i^- \xhookrightarrow{b_p} q_i \\
&q_i X_i^+ \xhookrightarrow{b_p b_p b_p} q_i \\
&q_i X_i^0 \xhookrightarrow{b_p b_p} p \\
&q_i X \xhookrightarrow{b_p b_p} q_i \qquad \text{对于所有的} X \in \Gamma_C \setminus \{X_i^-, X_i^+, X_i^0\}
\end{aligned}
$$

我们可以看到，当状态 q_i 遇到其他计数器上的操作时，和状态 p 一样，每个栈符号在出栈的过程也会做两个 b_p 动作。而当状态 q_i 遇到与第 i 个计数器相关的操作时，会有和状态 p 不同的行为：当遇到 $\mathsf{INC}(c_i)$ 时，状态 q_i 控制下的进程会多做一个 b_p 动作；当遇到 $\mathsf{DEC}(c_i)$ 时，状态 q_i 控制下的进程会少做一个 b_p 动作；当遇到 $\mathsf{RESET}(c_i)$ 时，状态 q_i 控制下的进程会进入状态 p。

借助这些规则，我们可以证明引理3.6。

证明 (引理3.6的证明) 可以看到 $q_i\alpha$ 和 $p\alpha$ 是否互模拟取决于从它们开始做的 b_p 动作的次数是否相等。在遇到第一个栈符号 X_i^0 之前，进程 $q_i\alpha$ 中每遇到一个符号 X_i^+，会做连续 3 个 b_p 动作；每遇到一个符号 X_i^-，会做 1 个 b_p 动作；遇到其他的符号会做连续 2 个 b_p 动作。当遇到符号 X_i^0 后，会进入状态 p，此后所有的符号都做连续 2 个 b_p 动作。因此 $q_i\alpha$ 和 $p\alpha$ 互模拟当且仅当

$$3I_i + D_i + 2(|\alpha| - I_i - D_i) = 2|\alpha|$$

这里 I_i 和 D_i 的定义参见 (3–1)。因为 $\mathsf{count}_i(\alpha) = I_i - D_i$，引理得证。 □

3.2.5 覆盖性检查

通过之前的构造，我们保证了互模拟博弈将会忠实的反映重置 Petri 网的一条执行路径。当互模拟博弈到达格局 $(pY_{t_f}\alpha, pY'_{t_f}\alpha)$ 时，即对应的 $\mathcal{N}$ 进入状态 t_f 时，攻击者有机会去验证当前的格局是否覆盖目标格局 $\sigma_f = (t_f, m_1, m_2, \cdots, m_d)$。

我们引入栈符号 $(Z_1, Z'_1), (Z_2, Z'_2), \cdots, (Z_{d+1}, Z'_{d+1})$ 和栈符号 $(Z_{1,m_1}, Z'_{1,m_1}), (Z_{2,m_2}, Z'_{2,m_2}), \cdots, (Z_{d,m_d}, Z'_{d,m_d})$，并引入以下规则：

$$
\begin{aligned}
&(pY_{t_f}, pY'_{t_f}) \xrightarrow{ATT} (pZ_1, pZ'_1) \\
&(pZ_i, pZ'_i) \xrightarrow{DEF} \left\{(pZ_{i,m_i}, pZ'_{i,m_i}), (pZ_{i+1}, pZ'_{i+1})\right\},\ i = 1, 2, \cdots, d \\
&pZ_{d+1} \xrightarrow{f} p, \qquad pZ'_{d+1} \xrightarrow{b_{pop}} p
\end{aligned}
$$

攻击者可以通过执行 $(pY_{t_f}, pY'_{t_f}) \xrightarrow{ATT} (pZ_1, pZ'_1)$ 来声称当前格局可以覆盖格局 σ_f。此后，防御者要避免博弈进入 (pZ_{d+1}, pZ'_{d+1})，因为 pZ_{d+1} 的 f 动作不能被 pZ'_{d+1} 模拟。因此，防御者将从以下格局中选择一个继续下去：

$$(pZ_{1,m_1}\alpha, pZ'_{1,m_1}\alpha),\ (pZ_{2,m_2}\alpha, pZ'_{2,m_2}\alpha), \cdots,\ (pZ_{d,m_d}\alpha, pZ'_{d,m_d}\alpha)$$

当防御者选择格局 $(pZ_{i,m_i}\alpha, pZ'_{i,m_i}\alpha)$，表示防御者质疑在第 i 维上，当前计数器的值比目标值小。因此我们将会设计规则使得下述性质成立：

引理 3.7 对于 $\alpha \in \Gamma_S^*$，$pZ_{i,m_i}\alpha \sim pZ'_{i,m_i}\alpha$ 当且仅当 $\mathsf{count}_i(\alpha) < m_i$。

为了保证引理3.7成立，对于 $i = 1, 2, \cdots, d$，我们引入栈符号 $(Z_{i,0}, Z'_{i,0})$, …, (Z_{i,m_i}, Z'_{i,m_i}) 并引入以下规则：

$$\begin{aligned} &(pZ_{i,j}, pZ'_{i,j}) \xrightarrow{DEF} \left\{(pZ_{i,j-1}X_i^-, pZ'_{i,j-1}X_i^-),\ (p, q_i)\right\},\ \ j = 1, 2, \cdots, m_i \\ &pZ_{i,0} \xrightarrow{f} p, \qquad\qquad pZ'_{i,0} \xrightarrow{b_{pop}} p \end{aligned} \tag{3–2}$$

证明 (引理3.7的证明) 根据 (3–2) 中引入的规则，如果博弈进入 $(pZ_{i,0}, pZ'_{i,0})$，那么攻击者获胜。所以防御者一定要在某个时间点选择测零。而防御者在选择测零之前，一共有 m_i 次机会将符号 X_i^- 压入栈中。

如果 $\mathsf{count}_i(\alpha) < m_i$，那么防御者可以选择将 $\mathsf{count}_i(\alpha)$ 个 X_i^- 符号压入栈中。博弈将会进入格局 $(p\gamma\alpha, q_i\gamma\alpha)$，使得 $\mathsf{count}_i(\gamma\alpha) = 0$。根据引理3.6，防御者获胜。

如果 $\mathsf{count}_i(\alpha) \geq m_i$，那么无论防御者如何选择，最终进入测零时的格局 $(p\gamma\alpha, q_i\gamma\alpha)$，一定有 $\mathsf{count}_i(\gamma\alpha) > 0$。根据引理3.6，攻击者获胜。 □

3.2.6 结论证明

我们已经完成了 $\mathcal{A}$ 的构造，接下来我们可以证明归约的正确性。

引理 3.8 d 个计数器的重置 Petri 网的可覆盖问题可以（在指数时间）归约到 $d+1$ 个状态的下推自动机的互模拟问题。

证明 如果格局 $(t_s, n_1, \cdots, n_d)$ 可以覆盖格局 $(t_f, m_1, \cdots, m_d)$，那么存在一个格局 $(t_f, m'_1, \cdots, m'_d)$，使得 $(t_s, n_1, \cdots, n_d) \to^* (t_f, m'_1, \cdots, m'_d) \geq (t_f, m_1, \cdots, m_d)$。那么攻击者只要简单的按照路径 $(t_s, n_1, \cdots, n_d) \to^* (t_f, m'_1, \cdots, m'_d)$ 来选择动作即可。因为每一个减法操作都是合法的，防御者不能通过测零来赢得胜利。博弈可以到达 $(pY_{t_f}\alpha, pY'_{t_f}\alpha)$，并且对于 $1 \leq i \leq d$，$\mathsf{count}_i(\alpha) \geq m'_i$。攻击者可以进入覆盖性检查并赢得博弈。

如果格局 $(t_s, n_1, \cdots, n_d)$ 不能覆盖 $(t_f, m_1, \cdots, m_d)$，防御者只需要跟随攻击者的动作，并且在攻击者试图做不合法的减法操作时选择测零。那么当博弈进入格局 $(pY_{t_f}\beta, pY'_{t_f}\beta)$ 时，一定存在某个 i，使得 $\mathsf{count}_i(\beta) < m_i$。根据引理 3.7，如果攻击者要进入覆盖性检查，那么防御者将获胜。如果攻击者始终不进入覆盖性检查，博弈可以一直进行下去，防御者也将获胜。

本章中描述的归约是指数时间的。导致指数时间的原因是起始格局和目标格局可以是二进制编码的，因此对其的处理可能会引入指数多条规则。利用4.3.2小节的技术，也可以将此归约改进为多项式时间。 □

根据引理3.8，定理3.2和定理3.3，我们可以得到结论：

定理 3.1 判定下推自动机的强互模拟等价是 ACKERMANN-难的。状态数给定为 $d \geq 4$ 的下推自动机的强互模拟等价问题是$\mathbf{F}_{d-1}$-难的。

为了保证构造的下推自动机是一个赋范下推自动机，我们可以再引入一些规则。对于 $s \in \mathcal{S}$，我们有：

$$pY_s \xrightarrow{b_p} p \quad pY'_s \xrightarrow{b_p} p$$

这些规则的引入不会影响归约的正确性，因为攻击者不会主动去选择这些新引入的规则，否则防御者做对应的规则可以赢得博弈。因此我们可以将此结果推广到赋范下推自动机。

定理 3.9 判定赋范下推自动机的强互模拟等价是 ACKERMANN-难的。状态数给定为 $d \geq 4$ 的赋范下推自动机的互模拟等价问题是$\mathbf{F}_{d-1}$-难的。

3.3 一阶文法和下推自动机的比较

一阶文法是下推自动机互模拟等价研究中提出的一个重要概念。有很多关于下推自动机的结论都是在一阶文法的框架下给出的 [39, 71, 95, 106]。一阶文法等价于内部动作确定的下推自动机。事实上，一阶文法的定义中没有内部动作。一阶文法中的一个进程可以用一个图来表示，在内部动作确定的 PDA 转换为一阶文法的过程中，通过图上节点的坍缩，确定的内部动作都会被吸收掉。而不确定的内部动作无法吸收，因此一般的 PDA 无法转换成一阶文法。

具体而言，一个一阶文法中定义的进程 T 可以被编码到一个内部动作确定的 PDA 的进程 P，使得 T 和 P 是弱互模拟的 [71]。Caucal 在文章 [107] 的命题 5.8 中提到，在考虑强互模拟时，一阶文法要严格的比下推自动机强。不过 Caucal 没有给出证明。在本节中，我们将给出一个证明。

我们将证明如下定理：

定理 3.10 存在一个内部动作确定的 PDA 的进程 P，对于任意实时 PDA 进程 O，$P \not\approx O$。

有了定理3.10，我们再通过文章 [71] 中给出的从内部动作确定的 PDA 到一阶文法的转换方法，可以得到一个一阶文法的进程 T，使得 $T \approx P$。并且按照一阶文法的定义，T 中没有内部动作，因此 T 不能和任意实时 PDA 进程强互模拟。我们可以有如下推论：

推论 3.11 存在一个一阶文法中定义的进程 T，对于任意实时 PDA 进程 O，$O \not\sim T$.

我们给出一个满足定理3.10的内部动作确定的 PDA 的一个例子：

例 3.2 我们定义一个内部动作确定的 $\mathcal{A}$，它包含以下规则：

- $pX_0 \xrightarrow{a} pX$，　$pX \xrightarrow{a} pXX$，　$pX \xrightarrow{b} pYX$，　$pY \xrightarrow{b} pYY$；
- $pY \xrightarrow{c} p_1Y$，　$pY \xrightarrow{d} p_2Y$；
- $p_1Y \xrightarrow{\tau} p_1$，　$p_1X \xrightarrow{a} p_1$，　$p_2Y \xrightarrow{b} p_2$，　$p_2X \xrightarrow{\tau} p_2$.

直观上，pX_0 可以通过动作 $pX_0 \xrightarrow{a^m b^n} pY^m X^n$ 在栈中记录 m 和 n 的信息。执行动作 c 之后，进程可以做动作 a^m，执行动作 d 之后，再通过一段出栈的内部动作，进程可以做 b^n。我们将证明实时下推自动机无法做到这一点。这也可以说明为什么一阶文法下界的证明 [39] 无法直接用在实时下推自动机上。

引理 3.12 对于例子3.2中的进程 pX_0，和任意一个实时下推自动机的进程 O，一定有 $O \not\approx pX_0$。

本节的余下部分将会证明引理 3.12。

首先，我们定义一些路径上的性质。给定一条路径

$$\xi : P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} P_k \tag{3–3}$$

如果对于 $1 \le i \le k$ 都满足 $|P_i| \ge |P_0|$，那么我们称 ξ 是非下降（non-sinking）的。如果 ξ 是非下降的，那么我们可以将 (3–3) 写作：

$$\xi : qX\alpha \xrightarrow{a_1} Q_1\alpha \xrightarrow{a_2} \cdots \xrightarrow{a_k} Q_k\alpha \tag{3–4}$$

其中 $|Q_i| > 0$。直观上，非下降的路径要求整条路径的动作都源自初始进程的状态和栈顶符号。

定义 3.2 (可泵的) 给定一条路径 ξ，如果满足以下两点：

- ξ 是非下降的；
- 存在状态 $q \in Q$，栈符号 $X \in \Gamma$，$\alpha \in \Gamma^*$ 和 $\beta \in \Gamma^+$，使得 $P_0 = qX\alpha$，$P_k = qX\beta\alpha$。

我们称 ξ 是 可泵的（pumpable）的。

如果 ξ 是可泵的，那么我们可以将 (3–4) 写作：

$$\xi : qX\alpha \xrightarrow{a_1} Q_1\alpha \xrightarrow{a_2} \cdots \xrightarrow{a_k} qX\beta\alpha \tag{3–5}$$

我们可以不失一般性的假定在下推自动机中的每一条规则 $(p, X, a, q, \alpha) \in \mathcal{R}$ 都满足 $|\alpha| \leq 2$。任意一个下推自动机都可以通过引入一些辅助的栈符号来转换成一个满足上述要求的下推自动机。

引理 3.13 给定下推自动机 $\mathcal{A} = (Q, \Gamma, \Sigma, \mathcal{R})$，$P_0$ 是 $\mathcal{A}$ 的一个进程。如果路径：

$$\xi : P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} P_n \tag{3–6}$$

- $n \geq |P_0| \cdot (|Q| \cdot |\Gamma| + 1)^{|Q|\cdot|\Gamma|+1}$；
- 对于任意的两个 P_i, P_j，$i \neq j$，都满足 $P_i \neq P_j$。

那么 ξ 中一定包含一段可泵的路径。

证明 引理中对 ξ 没有任何限制。我们将首先证明如果一条路径是非下降的，那么当这段路径足够长之后，一定可以找到一段可泵的路径。之后再将该性质推广到任意路径，从而完成引理的证明。

首先，给定：

$$\xi' : Q_0 \xrightarrow{b_1} Q_1 \xrightarrow{b_2} \cdots \xrightarrow{b_m} Q_m \tag{3–7}$$

为一段非下降的路径，并且如果 $i \neq j$，那么 $Q_i \neq Q_j$。我们用 $\mathsf{type}(\xi')$ 来表示 ξ' 出现的所有进程的状态和栈顶字符组成的二元组：

$$\mathsf{type}(\xi') \stackrel{\text{def}}{=} \left\{(q, Z) : \xi' \text{ 中存在进程} qZ\beta\right\}$$

我们首先证明如下性质，记作 $(\star)$：

如果 ξ' 的长度 $m \geq (|\mathsf{type}(\xi')| + 1)^{|\mathsf{type}(\xi')|+1}$，那么 ξ' 中包含一段可泵的路径。$(\star)$

我们按照集合 $\mathsf{type}(\xi')$ 中的元素个数 $|\mathsf{type}(\xi')|$ 做归纳。

- $|\mathsf{type}(\xi')| = 1$，那么 ξ' 本身就是可泵的。

- $|\mathsf{type}(\xi')| = k+1$。根据假定，$\xi'$ 的长度至少为 $(k+2)^{k+2}$。因为 ξ' 是非下降的并且 ξ' 中任意两个进程都不相等，因此只有不超过 k 个其他进程的栈高度和 Q_0 相等，如果将这些栈高度最短的进程去掉，那么 (3–7) 将被分为不超过 $k+1$ 段，其中至少应该有一段的长度超过 $(k+1)^{k+1}$，我们将该段路径记作 ξ''。因为我们可以不失一般性的假定 $\mathcal{A}$ 的一条迁移规则最多只会让栈的高度增加 1，因此 ξ'' 也是非下降的。
  我们将进程 Q_0 写成 $qZ\alpha$，可以分两种情况讨论：(1) 如果 $(q,Z) \in \mathsf{type}(\xi'')$，那么在 ξ'' 中存在一个进程 $Q_j = qZ\beta\alpha$，路径 $Q_0 \xrightarrow{b_1} \cdots \xrightarrow{b_j} Q_j$ 是可泵的；(2) 如果 $(q,Z) \notin \mathsf{type}(\xi'')$，那么我们有 $|\mathsf{type}(\xi'')| = k$。可以根据归纳假设，$\xi''$ 包含一段可泵的路径。

接下来，我们将性质 ($\star$) 推广到任意路径 ξ。我们考察 (3–6) 中从起始进程 P_0 出发，长度为 $(|\mathcal{Q}| \cdot |\Gamma| + 1)^{|\mathcal{Q}|\cdot|\Gamma|+1}$ 的一段子路径 ξ' 中，(1) 如果 ξ' 是非下降的，那么根据性质 ($\star$)，我们可以在 ξ' 中得到一段可泵的路径；(2) 如果 ξ' 不是非下降的，那么在 ξ' 中存在某个 P'，$|P'| = |P_0| - 1$。我们可以将进程 P' 做为新的起点。因为初始的进程 P_0 的栈高度是有限的，最多经历 $|P_0| \cdot (|\mathcal{Q}||\Gamma| + 1)^{|\mathcal{Q}||\Gamma|+1}$ 步之后，我们一定能够找到一段可泵的路径。 □

有了上述引理之后，我们就可以证明引理3.12。

证明 (引理 3.12的证明) 我们用反证法。假定存在一个实时下推自动机 $\mathcal{A}' = (\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$ 和 $\mathcal{A}'$ 的一个进程 O，满足 $O \approx pX_0$。不失一般性的，我们假定 $\mathcal{R}$ 中所有的规则 $q_1X \xrightarrow{a} q_2\alpha \in \mathcal{R}$ 满足 $|\alpha| \leq 2$ 。为了区分，当我们用 p 表示例子3.2中的状态，用 q 表示 $\mathcal{A}'$ 中的状态。定义常数 $N = |\mathcal{Q}| \cdot |\Gamma| + 1$。假定在例3.2中，从进程 pX_0 开始连续做 m 个 a 动作和 n 个 b 动作，其中 $m \geq 1$：

$$pX_0 \xrightarrow{a^m} pX^m \xrightarrow{b} pYX^m \xrightarrow{b} \cdots \xrightarrow{b} pY^nX^m. \tag{3–8}$$

因为 $O \approx pX_0$ 并且 $\mathcal{A}'$ 是实时 PDA，因此进程 O 对 (3–8) 中动作的模拟的过程可以写成如下路径：

$$O \xrightarrow{a^m} O_{m,0} \xrightarrow{b} O_{m,1} \xrightarrow{b} \cdots \xrightarrow{b} O_{m,n} \tag{3–9}$$

其中，对于 $0 \leq i \leq n$，我们有 $O_{m,i} \approx pY^iX^m$。又因为对于 $i \neq j$，有 $pY^iX^m \not\approx pY^jX^m$，因此 $O_{m,i} \not\sim O_{m,j}$。我们可以令 $n = |O_{m,0}|N^N = |O_{m,0}| \cdot (|\mathcal{Q}| \cdot |\Gamma| + 1)^{|\mathcal{Q}|\cdot|\Gamma|+1}$。根据引理 3.13，存在 $q \in \mathcal{Q}$，$Z \in \Gamma$，$0 \leq s < t \leq n$，使得：

1. $qZ\alpha = O_{m,s}$，$qZ\beta\alpha = O_{m,t}$。其中，$\alpha \in \Gamma^*$，$\beta \in \Gamma^+$；

2. 以下路径是可泵的：

$$O_{m,s} \xrightarrow{b} O_{m,s+1} \xrightarrow{b} \cdots \xrightarrow{b} O_{m,t}$$

可泵的路径的特点是这一段是可以不断重复执行的。令 $k = t - s$，我们有：

$$qZ\alpha \xrightarrow{b^k} qZ\beta\alpha \xrightarrow{b^k} qZ\beta\beta\alpha \xrightarrow{b^k} \cdots \tag{3–10}$$

因为 $qZ\alpha = O_{m,s} \approx pY^sX^m$，路径(3–10)中的动作可以被进程 pY^sX^m 模拟：

$$pY^sX^m \xrightarrow{b^k} pY^{s+k}X^m \xrightarrow{b^k} pY^{s+2k}X^m \xrightarrow{b^k} \cdots \tag{3–11}$$

根据互模拟的定义，对于 $i \geq 0$，$qZ\beta^i\alpha \approx pY^{s+ki}X^m$。

我们可以选取两个不同的 m 的取值：m_1 和 m_2，我们分别得到两组(3–10)和(3–11)中的执行路径。当然，其中涉及到的 $s > 0, k > 0, q \in Q, Z \in \Gamma, \alpha \in \Gamma^*, \beta \in \Gamma^+$ 都可能不同。因为 q 和 Z 都只有有限多个，根据鸽巢原理（Pigeonhole principle），我们可以选定 $1 \leq m_1 < m_2 \leq N$ 使得(3–10)中的 q 和 Z 相同。其他不同的符号我们可以分别记作 $s_1, k_1, \alpha_1, \beta_1$ 和 $s_2, k_2, \alpha_2, \beta_2$。不难看出，

$$qZ\alpha_1 = O_{m_1,s_1} \approx pY^{s_1}X^{m_1} \qquad qZ\alpha_2 = O_{m_2,s_2} \approx pY^{s_2}X^{m_2}$$

此外，$qZ \xrightarrow{b^{k_1}} qZ\beta_1$，$qZ \xrightarrow{b^{k_2}} qZ\beta_2$。

因为 $O_{m_1,s_1} \approx pY^{s_1}X^{m_1}$，动作序列 $O_{m_1,s_1} \xrightarrow{b^{k_1\cdot N}} qZ\beta_1^N\alpha_1$ 可以被 $pY^{s_1}X^{m_1} \xrightarrow{b^{k_1\cdot N}} pY^{s_1+k_1\cdot N}X^{m_1}$ 模拟。因为进程 O_{m_2,s_2} 也可以执行从 O_{m_1,s_1} 开始的可泵的路径，并且 $O_{m_2,s_2} \approx pY^{s_2}X^{m_2}$，所以动作序列 $O_{m_2,s_2} \xrightarrow{b^{k_1\cdot N}} qZ\beta_1^N\alpha_2$ 可以被 $pY^{s_2}X^{m_2} \xrightarrow{b^{k_1\cdot N}} pY^{s_2+k_1\cdot N}X^{m_2}$ 模拟。我们可以得到

$$pY^{s_1+k_1\cdot N}X^{m_1} \approx qZ\beta_1^N\alpha_1 \qquad pY^{s_2+k_1\cdot N}X^{m_2} \approx qZ\beta_1^N\alpha_2$$

考虑下面一段从 $pY^{s_2+k_1\cdot N}X^{m_2}$ 开始的迁移路径：

$$pY^{s_2+k_1\cdot N}X^{m_2} \xrightarrow{c} p_1Y^{s_2+k_1\cdot N}X^{m_2} \overset{\tau}{\Longrightarrow} p_1X^{m_2} \xrightarrow{a^{m_1+1}} p_1X^{m_2-m_1-1}$$

$qZ\beta_1^N\alpha_2$ 将执行互模拟动作：$qZ\beta_1^N\alpha_2 \xrightarrow{ca^{m_1+1}} Q_1$。因为 $qZ\beta_1^N\alpha_2$ 是实时下推自动机中的进程，并且 $N \geq m_1 + 1$，因此从 $qZ\beta_1^N\alpha_2$ 出发执行的动作 ca^{m_1+1} 只和 $qZ\beta_1^N$ 有关，而和 α_2 无关。因此进程 $qZ\beta_1^N\alpha_1$ 出发也能够执行动作：$qZ\beta_1^N\alpha_1 \xrightarrow{ca^{m_1+1}}$

Q_2。然而，与进程 $qZ\beta_1^N\alpha_1$ 弱互模拟的进程 $pY^{s_1+k_1\cdot N}X^{m_1}$ 却不能执行 $\xrightarrow{ca^{m_1+1}}$。从而我们得到矛盾。引理得证。

□

3.4 本章小结

下推自动机的强互模拟问题是验证领域的核心问题之一。最近，Jančar 和 Schimtz 证明了：

定理 3.14 (Jančar 等 [71]) 判定下推自动机的强互模拟等价是一个 ACKERMANN 问题。判定状态数给定为 d 的下推自动机的强互模拟等价是$\mathbf{F}_{d+4}$ 问题。

而在本章中，我们证明了：

定理 3.1 判定下推自动机的强互模拟等价是 ACKERMANN-难的。状态数给定为 $d \geq 4$ 的下推自动机的强互模拟等价问题是$\mathbf{F}_{d-1}$-难的。

该证明还可以推广到赋范下推自动机：

定理 3.9 判定赋范下推自动机的强互模拟等价是 ACKERMANN-难的。状态数给定为 $d \geq 4$ 的赋范下推自动机的互模拟等价问题是$\mathbf{F}_{d-1}$-难的。

因此，我们有以下结论：

推论 3.15 （赋范）下推自动机的强互模拟等价是一个 ACKERMANN-完备问题。

此前，Jančar 等人证明了一个比实时下推自动机更强的模型：一阶文法的强互模拟等价也是一个 ACKERMANN-完备问题 [39, 71]。我们在本章中探讨了一阶文法和实时下推自动机的关系，证明了在强互模拟的意义下，一阶文法比实时下推自动机严格的强。

通过以上结论我们可以看到，下推自动机的状态数量在其互模拟等价的研究中是一个重要的参数。在第四章和第五章中，我们将分别探索下推自动机状态数量固定时的参数复杂性的下界和上界。

第四章　两个状态的赋范下推自动机互模拟等价下界

本章我们将会证明如下定理。

定理 4.1　两个状态的赋范下推自动机的强互模拟等价有 EXPTIME-难的下界。

这个证明的思路类似于 Kiefer 关于一般的基本进程代数的互模拟等价 EXPTIME-难的下界的证明 [54]。我们将会构造一个从“判定 Hit-or-Run 博弈的胜者问题”到“两个状态的赋范下推自动机的互模拟等价问题”的归约。在4.1节中，我们回顾了此前下推自动机强互模拟的参数复杂性的研究情况，总结了此前的证明技术的局限性，并提出可能的研究思路。在4.2节中，我们给出 Hit-or-Run 博弈的定义。在4.3节中，我们回顾了基本进程代数的互模拟等价下界的证明思路，这将为定理4.1的证明打下基础。在4.4节中，我们构造了定理4.1的证明。4.5节对本章内容做出总结。

4.1　研究思路

第三章中，我们证明了一个下推自动机强互模拟的一个参数复杂性下界：当状态数量为 $d \geq 4$ 时，下推自动机的强互模拟问题是$\mathbf{F}_{d-1}$-难的。这也是该问题的第一个参数复杂性下界。

这个下界结论来自于重置 Petri 网可覆盖问题的归约。重置 Petri 网可覆盖问题有一个非常好的参数复杂性：当一个重置 Petri 网的计数器的个数 $d \geq 3$ 时，可覆盖问题是一个$\mathbf{F}_d$-完备问题。这里的 $d \geq 3$ 的限制源于其下界证明技巧的局限性。其具体做法是编码一个有界的 Minsky 机 [108]。Minsky 机的定义中至少有两个计数器，而为了模拟 Minsky 机中的测零操作，还需要引入一个额外的计数器。因此，通过这个方法证明的下界至少需要模拟三个计数器。第三章的归约中，我们用了 $d+1$ 个状态来模拟 d 个计数器。对于不超过三个状态的 PDA，这个归约暂时无法得到任何复杂性的下界。因此，我们需要探索其他的技术。

当 PDA 只有一个状态时，它等价于 BPA 模型。2013 年，Kiefer 通过一个 Hit-or-Run 博弈的胜者判定问题的归约证明了 BPA 的强互模拟是 EXPTIME-难的。本章中，我们对定理4.1的证明将会借鉴 Kiefer 的证明思路。除此之外，2018 年，Jančar 证明了，给定一个二进制表示的单计数器网（One Counter Net, OCN），其强互模拟问题是 EXPSPACE-难的 [109]。OCN 等价于只有一个栈符号的下推自动机。这个结论很有希望推广到有少量控制状态的 PDA 的强互模拟问题上。一个可能的思路

是将 OCN 的状态编码到 PDA 栈顶，将 OCN 的计数器编码到栈中，同时用第三章的技术，借助两个状态对 OCN 的减法操作做出限制，从而实现对 OCN 的编码。不过这个证明还有一些细节上的问题需要克服。

4.2 Hit-or-Run 博弈

我们首先介绍 Hit-or-Run 博弈的定义：

定义 4.1 (Hit-or-Run 博弈) 一个 Hit-or-Run 博弈在两个玩家（玩家 0 和玩家 1）之间展开，它可以表示为一个六元组 $(S_0, S_1, \to, s_\vdash, s_\dashv, k_\dashv)$。其中

- S_0 是玩家 0 的状态集合；
- S_1 是玩家 1 的状态集合，$S_0 \cap S_1 = \varnothing$，定义 $S \stackrel{\text{def}}{=} S_0 \cup S_1$；
- $\to \subseteq S \times \mathbb{N} \times S \cup \{s_\dashv\}$ 是迁移规则集合；
- $s_\vdash \in S$ 是起始状态；
- $s_\dashv \notin S$ 是终止状态；
- $k_\dashv \in \mathbb{N}$ 是目标值；

对于规则 $(s, n, t) \in \to$，我们也写作 $s \xrightarrow{n} t$。博弈的一个格局是 $\sigma = (s, k) \in (S \cup \{s_\dashv\}) \times \mathbb{N}$。博弈从格局 $(s_\vdash, 0)$ 开始。每一轮按照如下规则进行：

- 如果当前的格局 $(s, k) \in S_0 \times \mathbb{N}$，那么玩家 0 将会选择一条规则 (s, n, t)，格局被更新为 $(t, k + n)$；
- 如果当前的格局 $(s, k) \in S_1 \times \mathbb{N}$，那么玩家 1 将会选择一条规则 (s, n, t)，格局被更新为 $(t, k + n)$；
- 如果当前的格局 (s, k) 满足 $s = s_\dashv$，博弈终止。

如果博弈终止时，当前格局为 $(s_\dashv, k), k \neq k_\dashv$，那么玩家 1 将获胜；如果博弈一直保持在格局集合 $\{(s_\dashv, k_\dashv)\} \cup S \times \mathbb{N}$ 中，玩家 0 将获胜。可以看到，博弈的名字“Hit-or-Run”就表示玩家 0 的获胜条件，或者击中（hit）格局 $(s_\dashv, k_\dashv)$，或者避开（run from）状态 $s_\dashv$。

Kiefer 将一个多项式空间的交替式图灵机（Alternating Turing Machine, ATM）的接受问题归约到 Hit-or-Run 博弈的胜者判定问题。借助计算复杂性理论中的结论 APSPACE=EXPTIME，可以得到如下结论：

命题 4.2 (Kiefer [54]) 给定一个 Hit-or-Run 博弈，判定博弈的胜者是一个 EXPTIME-完备问题。

4.3 基本进程代数的互模拟等价下界

4.3.1 基本进程代数

定义 4.2 一个基本进程代数系统 $\mathcal{B}$ 是一个三元组 $(\mathcal{N}, \Sigma, \mathcal{R})$。

- $\mathcal{N}$ 是一个有限的非终止符号集合；
- Σ 是一个有限的动作集合；
- $\mathcal{R} \subseteq \mathcal{N} \times \Sigma \times \mathcal{N}^*$ 是一个迁移规则集合。

对于 $(X, a, \beta) \in \mathcal{R}$，我们也写作 $X \xrightarrow{a} \beta$。一个基本进程代数 $(\mathcal{N}, \Sigma, \mathcal{R})$ 可以按照规则 (4–1) 诱导出一个 $\mathcal{N}^*$ 上的标记迁移系统 $(\mathcal{N}^*, \Sigma, \rightarrow)$。

$$\frac{X \xrightarrow{a} \alpha \in \mathcal{R}}{X\beta \xrightarrow{a} \alpha\beta \in \ \rightarrow} \tag{4–1}$$

基本进程代数上的进程用符号 $P, Q \in \mathcal{N}^*$ 表示。基本进程代数上的强互模拟等价定义如下：

BPA 上的强互模拟等价问题
输入：一个基本进程代数 $\mathcal{B} = (\mathcal{N}, \Sigma, \mathcal{R})$ 和两个进程 P 和 Q， 问题：判定 P 和 Q 是否强互模拟?

目前为止基本进程代数的强互模拟问题最好的下界是 Kiefer 在 2013 年给出的 EXPTIME-难 [54]。具体做法是构造一个从 Hit-or-Run 博弈的胜者判定问题到基本进程代数互模拟问题归约。

定理 4.3 (Kiefer [54]) 基本进程代数的强互模拟是 EXPTIME-难的。

我们将会简要回顾定理4.3的证明思路。我们在构造定理4.1的证明中也会用到类似的思路。

4.3.2 二进制编码

由于 Hit-or-Run 博弈中涉及到了自然数的加法操作，为了保证归约的过程是多项式时间的，有必要在 BPA 上实现一个二进制的编码。在本小节中，我们将介绍 Kiefer 文中的编码。

对于给定的 Hit-or-Run 博弈 $(S_0, S_1, \rightarrow, s_{\vdash}, s_{\dashv}, k_{\dashv})$，我们令 k 为博弈输入中出现的最大的数字，即 $k \stackrel{\text{def}}{=} \max\left\{\{k_{\dashv}\} \cup \{m : (s,m,t) \in \rightarrow\}\right\}$。令 n 为最小的满足 $2^n \geq k$ 的自然数。可以引入非终止符集合：

$$\mathcal{N}_b = \{X_0, X_1, X_2, \cdots, X_n\} \subseteq \mathcal{N}$$

其中 X_i 将用来表示数字 2^i。类似于第三章。我们可以定义一个从 $\mathcal{N}_b^*$ 到 $\mathbb{N}$ 的映射 count，对于 $\alpha = X_{j_1} X_{j_2} \cdots X_{j_n} \in \mathcal{N}_b^*$，

$$\mathsf{count}(\alpha) \stackrel{\text{def}}{=} \sum_{i=1}^{n} 2^{j_i}$$

例 4.1 对于进程 $\alpha = X_0 X_2 X_4 X_0$，$\mathsf{count}(\alpha) = 2^0 + 2^2 + 2^4 + 2^0 = 22$。

对于 $X_i \in \mathcal{N}_b$，我们引入如下规则：

$$\begin{aligned} X_i &\xrightarrow{b_p} X_0 X_1 \cdots X_{i-1} \quad 0 < i \leq n \\ X_0 &\xrightarrow{b_p} \epsilon \end{aligned} \tag{4–2}$$

规则 (4–2) 也是非终止符号 X_i 可以做的唯一一条迁移规则。借助规则 (4–2)，我们可以实现进程与自然数的对应：一个表示自然数 j 的进程，可以做连续 j 个 b_p 动作。例如，对于例4.1中的进程 α，可以验证 $\alpha \xrightarrow{b_p^{22}} \epsilon$。我们有如下性质：

命题 4.4 如果两个 BPA 进程 $\alpha, \beta \subseteq \mathcal{N}_b^*$，那么：

$$\alpha \sim \beta \iff \mathsf{count}(\alpha) = \mathsf{count}(\beta)$$

注 第三章中的归约也可以利用本节中的二进制编码的技巧优化为一个多项式时间的归约。

4.3.3 归约思路

完成归约需要构造基本进程代数 $\mathcal{B} = (\mathcal{N}, \Sigma, \mathcal{R})$ 以及两个进程 P 和 Q，使得

$$P \sim Q \quad \text{当且仅当} \quad \text{玩家 0 将会赢得 Hit-or-Run 博弈}$$

回顾在2.4.2小节中定义的互模拟的博弈刻画，攻击者将会试图去证明 $P \nsim Q$，因此攻击者将会和 Hit-or-Run 博弈中的玩家 1 有相同的获胜目标，而防御者则对应于玩家 0 的获胜目标。

首先，对于 Hit-or-Run 博弈中的每一个状态，Kiefer 引入了一对非终止符号来记录这个状态：

$$\mathcal{N}_s = \{Y_s, Y_s' : s \in S \cup \{s_\dashv\}\}$$

因此，初始的进程对 (P, Q) 被定义为：

$$P \stackrel{\text{def}}{=} Y_{s_\vdash} \quad Q \stackrel{\text{def}}{=} Y'_{s_\vdash}$$

在 Kiefer 构造的归约中，对于 Hit-or-Run 博弈中的规则的模拟是通过类似于3.2.3小节中定义的宏规则 $\xrightarrow{DEF}$ 和 $\xrightarrow{ATT}$ 来实现的。$\xrightarrow{DEF}$ 用来模拟 S_0 中的状态出发的规则，$\xrightarrow{ATT}$ 用来模拟 S_1 中的状态出发的规则。Kiefer 引入了一些规则把格局中计数器的值记录在进程中。例如在格局 (s, k) 时，对应的两个进程 $(P', Q') = Y_s\alpha, Y_s'\alpha$，其中 $\alpha \in \mathcal{N}_b^*$ 并且满足 $\mathsf{count}(\alpha) = k$。当互模拟进程对的第一个非终止符号分别为 $(Y_{s_\dashv}, Y'_{s_\dashv})$ 时，攻击者将有机会选择是否要进行验证，验证的目标是当前进程记录的数值是否为目标值 $k_\dashv$。如果为目标值，防御者将获胜，否则攻击者将获胜。在验证目标值的过程中，Kiefer 选择了一个进程，加入非终止符号序列 γZ，其中 Z 不能做任何动作，$\mathsf{count}(\gamma) = k_\dashv$，最终的进程对将会是 $(\gamma Z\beta, \beta)$。

以下命题是显然的。

命题 4.5 如果非终止符号 Z 不能做任何动作，那么对于任意的 $\alpha, \beta \in \mathcal{N}^*$，$\alpha Z\beta$ 和 α 都是互模拟的。

根据命题4.4和命题4.5，$\gamma Z\beta \sim \beta$ 当且仅当 $\mathsf{count}(\beta) = \mathsf{count}(\gamma)$。因此防御者在击中目标值时可以获得互模拟博弈的胜利。而如果攻击者始终没有选择验证目标值，那么防御者也将获得互模拟博弈的胜利。

4.4 归约过程

本节中我们将给出一个从 Hit-or-Run 博弈的胜者判定问题到两个状态的赋范下推自动机互模拟问题的多项式时间归约，从而完成定理4.1的证明。

相对于4.3节中介绍的证明，本节中我们将要求构造的下推自动机满足赋范性。4.3节中的归约引入了一个不能做任何动作的符号来帮助验证目标值，这会导致进程成为非赋范的进程，我们需要避免引入这样的栈符号。我们知道基本进程代数等价于只有一个状态的下推自动机，因此本节中的归约的重点是用一个额外的状态来帮助实现目标值的验证。这里的思路有些类似第三章中的技巧，在验证过程

中，进程只能做一种动作，所以不需要保证栈符号之间的对应，两个进程是否互模拟只取决它们能做的动作的总数是否相等。当格局进入 $(P', Q') = Y_s\alpha, Y'_s\alpha$，攻击者想要验证目标值时，我们在其中一个进程 P' 中加入我们要验证的值，之后验证 P' 中的数字是否是 Q' 的两倍。这个验证可以利用两个状态来实现。

本小节的余下部分将给出一个完整的归约构造。

4.4.1 准备工作

我们将构造一个包含两个状态的赋范下推自动机 $\mathcal{A} = (\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$。其中，状态集合为：

$$\mathcal{Q} = \{p, q\}$$

我们沿用4.3.2节中的二进制编码技巧，引入下列栈符号以及规则 (4–2)。

$$\Gamma_c \stackrel{\text{def}}{=} \{X_0, X_1, \cdots, X_n\} \subseteq \Gamma$$

这里的自然数 n 的定义参见4.3.2节。

我们沿用4.3.2节对映射 count 的定义。此外，我们定义一个新的映射 encode：$\{0, 1, \cdots, 2^n\} \to \Gamma_c^*$，对于 $0 \le i \le 2^n$，存在唯一的序列 $i_1, i_2, \cdots, i_j$ 满足 $0 \le i_1 < i_2 < \cdots < i_j \le n$ 并且 $i = 2^{i_1} + 2^{i_2} + \cdots + 2^{i_j}$。我们定义

$$\mathsf{encode}(i) \stackrel{\text{def}}{=} X_{i_1} X_{i_2} \cdots X_{i_j}$$

对于 Hit-or-Run 博弈中的每一个状态，我们引入栈符号：

$$\Gamma_s = \left\{Y_s, Y'_s : s \in S \cup \{s_\dashv\}\right\} \subseteq \Gamma$$

初始的进程对 (P, Q) 被定义为：

$$P \stackrel{\text{def}}{=} pY_{s_\vdash} \quad Q \stackrel{\text{def}}{=} pY'_{s_\vdash}$$

4.4.2 记录 Hit-or-Run 博弈中的格局

接下来我们的目标是引入一些规则将 Hit-or-Run 博弈中的格局的计数器的值记录在栈中。

在 Hit-or-Run 博弈中，不失一般性的，我们可以假设从每一个 S 中的状态出发都有两条规则。给定状态 $s \in S_j$，其中 $j = 0$ 或 1，假如从状态 s 出发有 $d \ge 3$ 条规则，分别计作 $s \xrightarrow{n_i} t_i$，$i \in \{1, 2, \cdots, d\}$，如图4–1(a) 所示，那么我们可以引入

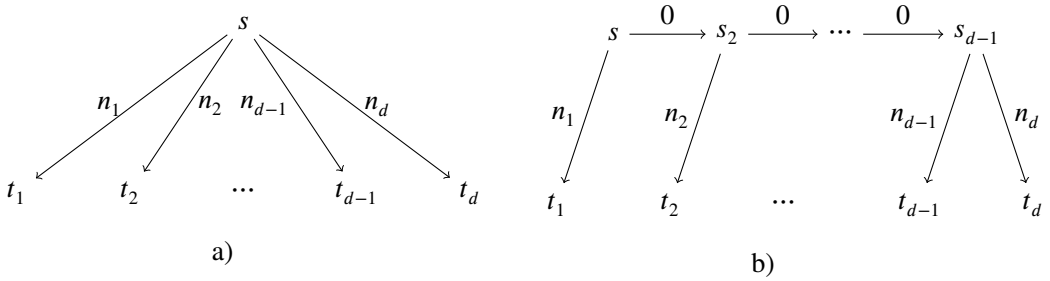


图 4–1 辅助状态示例

一些辅助状态 $s_2, \cdots, s_{d-1} \in S_j$，如图4–1(b) 所示，保证所有的状态出发最多只有两条规则。对于只有一条规则的状态，我们可以加入这条规则的一个复制。我们将会去掉没有规则的状态。以上这些修改都不会影响到博弈的最终结果。

我们对 $s \in S$ 做如下讨论：

- 对于 S_0 中的状态 s，假设从 s 出发的两条规则是 $s \xrightarrow{n_1} t_1$ 和 $s \xrightarrow{n_2} t_2$。令 $\alpha_1 = \mathsf{encode}(n_1)$，$\alpha_2 = \mathsf{encode}(n_2)$，我们在 $\mathcal{R}$ 中引入规则：

$$(pY_s, pY'_s) \xrightarrow{DEF} \left\{(pY_{t_1}\alpha_1, pY'_{t_1}\alpha_1), (pY_{t_2}\alpha_2, pY'_{t_2}\alpha_2)\right\} \tag{4–3}$$

- 对于 S_1 中的状态 s，假设从 s 出发的两条规则是 $s \xrightarrow{n_1} t_1$ 和 $s \xrightarrow{n_2} t_2$。令 $\alpha_1 = \mathsf{encode}(n_1)$，$\alpha_2 = \mathsf{encode}(n_2)$，我们在 $\mathcal{R}$ 中引入规则：

$$(pY_s, pY'_s) \xrightarrow{ATT} \left\{(pY_{t_1}\alpha_1, pY'_{t_1}\alpha_1), (pY_{t_2}\alpha_2, pY'_{t_2}\alpha_2)\right\} \tag{4–4}$$

通过规则 (4–3) 和 (4–4)，我们可以保证下述引理成立。

引理 4.6 在 Hit-or-Run 博弈中，如果玩家 0（玩家 1）能够进入使博弈进入格局 (s, k)，那么在对应的互模拟博弈中，防御者 (攻击者) 能够让博弈进入到一个进程对 $(P, Q) = (pY_s\alpha, pY'_s\alpha)$，其中，$\mathsf{count}(\alpha) = k$。

4.4.3 验证目标值

当互模拟博弈进入 $(pY_{s_\dashv}\alpha, pY'_{s_\dashv}\alpha)$ 时，我们希望防御者可以在 $\mathsf{count}(\alpha) = k_\dashv$ 时赢得博弈，攻击者在 $\mathsf{count}(\alpha) \neq k_\dashv$ 时赢得博弈。我们引入规则：

$$(pY_{s_\dashv}, pY'_{s_\dashv}) \overset{b}{\hookrightarrow} (p, q\beta) \tag{4–5}$$

其中 $\beta = \mathsf{encode}(k_\dashv)$。通过规则 (4–5)，互模拟博弈将会进入到 $(p\alpha, q\beta\alpha)$。我们希望引入规则使得如下引理成立。

引理 4.7 在下推自动机 $\mathcal{A}$ 中，$\beta = \mathsf{encode}(k_{\dashv})$，$\alpha \subseteq \Gamma_c^*$。那么：

$$p\alpha \sim q\beta\alpha \iff \mathsf{count}(\alpha) = k_{\dashv}$$

在状态为 q 时的出栈操作类似于规则 (4–2)，我们在这里引入规则：

$$\begin{aligned} qX_i &\xrightarrow{b_p} qX_0X_1\cdots X_{i-1} \quad 0 < i \le n \\ qX_0 &\xrightarrow{b_p} q\epsilon \end{aligned} \tag{4–6}$$

而在状态为 p 时，每次出栈操作将会做两个 b_p 动作。我们引入规则：

$$\begin{aligned} pX_i &\xrightarrow{b_pb_p} pX_0X_1\cdots X_{i-1} \quad 0 < i \le n \\ pX_0 &\xrightarrow{b_pb_p} p\epsilon \end{aligned} \tag{4–7}$$

根据这些规则，我们能够使引理4.7成立。

证明 (引理4.7的证明) 因为从进程 $p\alpha$ 和进程 $q\beta\alpha$ 出发只能做唯一一种动作 b_p，并且该动作是确定的。因此，$p\alpha \sim q\beta\alpha$ 当且仅当它们在成为空进程之前，可以做的动作 b_p 的个数相等。根据规则 (4–6)，$q\beta\alpha$ 可以做 $k_{\dashv} + \mathsf{count}(\alpha)$ 个 b_p 动作，根据规则 (4–7)，$p\alpha$ 可以做 $2\mathsf{count}(\alpha)$ 个 b_p 动作，因此 $p\alpha \sim q\beta\alpha$ 当且仅当 $\mathsf{count}(\alpha) = k_{\dashv}$。引理得证。 □

目前为止，我们已经完成了 $\mathcal{A}$ 的构造，我们证明如下引理。

引理 4.8 给定一个 Hit-or-Run 博弈 $(S_0, S_1, \to, s_{\vdash}, s_{\dashv}, k_{\dashv})$，我们可以在多项式时间内构造一个有两个状态的赋范下推自动机 $\mathcal{A} = (\mathcal{Q}, \Gamma, \Sigma, \mathcal{R})$ 以及 $\mathcal{A}$ 中的两个进程 P 和 Q。使得

$$P \sim Q \quad \text{当且仅当} \quad \text{玩家 0 将会赢得 Hit-or-Run 博弈}$$

证明 如果在 Hit-or-Run 博弈中，玩家 0 有必胜策略，即博弈的格局一直保持在集合 $\{(s_{\dashv}, k_{\dashv})\} \cup S \times \mathbb{N}$ 中。当格局在集合 $S \times \mathbb{N}$ 中时，互模拟博弈可以继续进行下去，而当格局为 $(s_{\dashv}, k_{\dashv})$ 时，根据引理4.6，互模拟博弈将会进入到进程对 $(P, Q) = (pY_s\alpha, pY_s'\alpha)$，其中，$\mathsf{count}(\alpha) = k$。又根据引理4.7，我们有 $P \sim Q$。

如果在 Hit-or-Run 博弈中，玩家 1 有必胜策略，那么经过有限次迁移后，玩家 1 总能够使博弈进入格局 $(s_{\dashv}, k)$，其中 $k \neq k_{\dashv}$。根据引理4.6，互模拟博弈将会进入根据 $(P, Q) = (pY_s\alpha, pY_s'\alpha)$，其中，$\mathsf{count}(\alpha) \neq k$。之后，攻击者可以选择验证目标值，使互模拟博弈进入格局 $(p\alpha, q\beta\alpha)$，其中 $\beta = \mathsf{encode}(k_{\dashv})$。又根据引理4.7，我们有 $P \not\sim Q$。 □

结合定理4.3和引理4.8，我们完成了定理4.1的证明。

4.5 本章小结

结合最近的研究结果来看 [71]，下推自动机的状态数量在互模拟等价验证中是一个重要的参数。Kiefer 证明了当 PDA 只有一个状态时的互模拟等价是 EXPTIME-难的。而截止目前，还没有关于固定状态数的赋范下推自动机的复杂性下界的研究。

在本章中，我们将 Hit-or-Run 博弈的胜者判定问题归约到两个状态的赋范下推自动机强互模拟问题，从而证明了定理4.1:

定理 4.1 两个状态的赋范下推自动机的强互模拟等价有 EXPTIME-难的下界。

关于下一步研究的思路，一个可以借鉴的工作是 Jančar 在二进制表示的 OCN 上的强互模拟问题 EXPSPACE-难的下界证明 [109]。借助目前积累的一些在 PDA 上对计数器编码的技巧，或许可以证明一个更好的下界。

第五章　固定状态数赋范下推自动机互模拟等价上界

本章我们将会证明如下定理：

定理 5.1　状态数给定为 d 的赋范下推自动机的互模拟等价问题是$\mathbf{F}_{d+3}$ 问题。

在本节中，我们将优化 Jančar 和 Schmitz 关于固定状态数的下推自动机的复杂性结果 [71]。在赋范下推自动机中，我们将之前的$\mathbf{F}_{d+4}$ 优化到$\mathbf{F}_{d+3}$。

在第5.1节中我们将介绍一些基本的定义。在第5.2节我们回顾 Jančar 和 Schmitz 的算法。在第5.3节中我们给出赋范下推自动机的一些性质，从而证明定理5.1。第5.4节对本章做出总结。

5.1　基本定义以及性质

5.1.1　范数的推广和互模拟同余

在第2.2节中我们定义了赋范 PDA 进程的范数。进程 P 的范数 $\|P\|$ 表示最短的能够将进程 P 的栈做空的路径长度。而在本章中，我们需要用到更加精细的定义。假如 PDA 中有 d 个状态，我们定义 $|||P||| = (n_1, n_2, \cdots, n_d)$，其中 n_i 表示最短的从 P 到 $p_i\epsilon$ 的路径长度。如果没有从 P 到 $p_i\epsilon$ 的路径，那么 $n_i \stackrel{\text{def}}{=} \infty$。此外，我们定义 $|||p, X|||_i \stackrel{\text{def}}{=} n_i$。

类似于定义2.3，我们可以定义进程 P 关于状态 p_i 的范数下降路径。

定义 5.1 (P 关于状态 p_i 的范数下降路径)　如果一条路径

$$\pi = P_1 \xrightarrow{\ell_1} P_1 \xrightarrow{\ell_2} \cdots P_{k-1} \xrightarrow{\ell_k} P_k$$

满足对于 $1 \le j \le k-1$，$\left|\left|\left|P_{j+1}\right|\right|\right|_i = \left|\left|\left|P_j\right|\right|\right|_i - 1$ 成立，我们称 π 为一条P 关于 p_i 的范数下降路径。

对于 PDA 进程上的一个等价关系 $\mathcal{R}$，如果 $\mathcal{R}$ 满足：

$$p\alpha \ \mathcal{R}\ q\beta \ \Rightarrow\ \forall \delta \in \Gamma^*, p\alpha\delta \ \mathcal{R}\ q\beta\delta$$

我们称等价关系 $\mathcal{R}$ 满足同余性（congruence）。可以从例5.1中看到，下推自动机上的互模拟关系是不满足同余性的。

例 5.1 如果下推自动机 $\mathcal{A}$ 中包含如下转移规则，

$$pX \xrightarrow{a} p\epsilon, \quad qX \xrightarrow{a} q'X, \quad q'X \xrightarrow{a} q\epsilon$$

那么我们有 $pXX \sim qX$，而 $pXXX \nsim qXX$。

在下推自动机的互模拟的定义中，两个栈为空但是状态不同的进程是互模拟的，例如在例5.1中，$p\epsilon \sim q\epsilon$。这也解释了为什么 PDA 上的互模拟关系不满足同余性。基于此，Stirling 定义了一个互模拟同余关系 [110]。这个关系在互模拟的基础上，要求两个栈为空的进程的状态也应该相同。

定义 5.2 (互模拟同余 [110]) $\mathcal{R}$ 是下推自动机 $\mathcal{A}$ 上的进程对之间的一个二元关系，对于任意的 $(p\alpha, q\beta) \in \mathcal{R}$：

- 如果 $\alpha = \epsilon$，那么 $\beta = \epsilon$，并且 $q = p$;
- 如果 $\beta = \epsilon$，那么 $\alpha = \epsilon$，并且 $p = q$;
- 如果 $p\alpha \xrightarrow{a} p'\alpha'$，存在迁移规则 $q\beta \xrightarrow{a} q'\beta'$，并且 $(p'\alpha', q'\beta') \in \mathcal{R}$;
- 如果 $q\beta \xrightarrow{a} q'\beta'$，存在迁移规则 $p\alpha \xrightarrow{a} p'\alpha'$，并且 $(p'\alpha', q'\beta') \in \mathcal{R}$。

那么我们称 $\mathcal{R}$ 是互模拟同余关系。

如果进程 $p\alpha$ 和 $q\beta$ 满足互模拟同余，我们记作 $p\alpha \equiv q\beta$。互模拟同余关系满足同余性。我们可以将 $\mathsf{EL}(P, Q)$ 的定义推广到 $\mathsf{EL}_{\equiv}(P, Q)$。后文中，我们将会看到 $\equiv$ 关系在判定算法中的应用。

5.1.2 简单常量和平衡操作

在下推自动机互模拟的判定算法中，一个经常用到的操作是对进程进行复合或分解。而由于状态的存在，我们不能简单的将栈分为两部分。研究者们通常采用常量（constant）的技术 [110] 来帮助表示进程的复合或分解。所有的常量都可以被视为栈符号，我们用 Γ_C 表示所有的常量的集合。

常量可以分为简单常量（simple constant）和递归常量（recursive constant），我们分别用 Γ_{SC} 和 Γ_{RC} 表示所有的简单常量和递归常量的集合。本小节中，我们将会介绍简单常量，在小节5.1.3中我们将会介绍递归常量。

我们用 U 来表示一个简单常量。假设 $\mathcal{A}$ 的状态集合 $Q = \{p_1, p_2, \cdots, p_k\}$，那么简单常量 U 被定义为：

$$p_1U = Q_1, \quad p_2U = Q_2, \quad \cdots, \quad p_kU = Q_k$$

我们可以将 U 表示为 $(Q_1, Q_2, \cdots, Q_k)$，我们记 $U(i) = Q_i$。其中，Q_i 中的栈符号不能包含其他的简单常量（但可以包含递归常量）。我们可以定义 $|U| = \max\{|Q_1|, \cdots |Q_k|\}$。

接下来我们介绍一个在上界算法中常用的技术，称为平衡操作。

我们回顾下面的定义：

$$d_0 \stackrel{\text{def}}{=} \max\{\|pX\|, p \in Q, X \in \Gamma\}$$

对于一对进程 (P, Q)，假设 $P = pX\alpha, Q = M\beta$，$|M| = d_0$，如果在 (P, Q) 的等价步数下降路径中进程 P 在 d_0 步之内始终保持 α 不变，d_0 步操作之后的结果为 $(P', Q') = (T\alpha, M'\beta)$，那么我们可以做如下的平衡操作。

平衡操作

1. 对于每一个状态 $p_i \in Q$，我们固定一个 pX 关于 p_i 的范数下降路径 $pX \xrightarrow{u_i} p_i$。
2. 对于 $i = 1, 2, \cdots, k$，我们找到一个使得 $\mathsf{EL}(p_i\alpha, M_i\beta)$ 最大的模拟路径 $M\beta \xrightarrow{u_i} M_i\beta$。
3. 定义简单常量:
$$U = (M_1, M_2, \cdots, M_k)$$
在从 (P, Q) 开始的等价步数下降路径 $(P, Q) \xrightarrow{u} (P', Q') = (T\alpha, M'\beta)$ 中，我们可以将 $(T\alpha, M'\beta)$ 改写为 $(TU\beta, M'\beta)$。

我们将用 $\odot$ 表示一次平衡操作：

$$(pX\alpha, M\beta) \xrightarrow{u} (T\alpha, M'\beta) \odot (TU\beta, M'\beta) \tag{5–1}$$

式子 (5–1) 也称为左平衡操作，类似的，形如 (5–2) 的平衡操作称为右平衡操作。

$$(M\alpha, qY\beta) \xrightarrow{u} (M'\alpha, T\beta) \odot (M'\alpha, TU\alpha) \tag{5–2}$$

下面引理说明了，平衡操作并不会影响等价步数，

引理 5.2 (Jančar[96]) 在平衡操作 $(pX\alpha, M\beta) \xrightarrow{u} (T\alpha, M'\beta) \odot (TU\beta, M'\beta)$ 中，$\mathsf{EL}(T\alpha, M'\beta) = \mathsf{EL}(TU\beta, M'\beta)$。

平衡操作的好处是使两个进程都有一个相同的后缀，并且其不同的“前缀” (TU, M') 的长度也有一个上界，这将会有助于估计等价步数的界。以下引理说

明了，平衡操作产生的“前缀”的大小是有界的。因此，在平衡操作的次数足够多时，其“前缀”必定会重复。

引理 5.3 (Jančar[96]) 在平衡操作 $(pX\alpha, M\beta) \xrightarrow{u} (T\alpha, M'\beta) \odot (TU\beta, M'\beta)$ 中，$|p_iU| \leq d_0$，$|T| \leq d_0$，$|M'| \leq 2d_0$。

5.1.3 递归常量和切割操作

本小节中我们介绍另外一种常量，称作递归常量。我们用 V 来表示一个递归常量。递归常量 V 被定义为：

$$p_1V = R_1,\ p_2V = R_2,\ \cdots,\ p_kV = R_k$$

我们记 $V(i) = R_i$。其中，R_i 有两种选择，

- R_i 可以是 $p_j\epsilon$，并且我们要求 $0 \leq i \leq j$；
- R_i 可以是 $p_j\alpha_iV$，其中 α 中不包含其他递归常量。

我们可以定义 $|V| = \max\{|\alpha_i| : R_i = p_j\alpha_iV\} \cup \{0\}$。

注 在后文算法中用到的递归常量 V 中，如果 $V(i) = p_j\alpha V$，那么 α 中最多只包含一个简单常量。如果 $V(i) = p_j\epsilon$，我们保证 $i \leq j$。

为了简化表述，我们有时会用 $\{p_i, R_i\}$ 来表示递归常量：

$$p_1V = p_1\epsilon,\ \cdots,\ p_iV = R_i,\ \cdots,\ p_kV = p_k\epsilon$$

引理 5.4 (Jančar [96]) 如果 $\mathsf{EL}_{\equiv}(E, F) = k < l = \mathsf{EL}_{\equiv}(E\alpha, F\alpha)$，那么存在一个状态 p_i，一个进程 $H \neq p_i\epsilon$，一个动作序列 $w \in \Sigma^*$，$|w| \leq k$ 使得，

- $E \xrightarrow{w} p_i\epsilon, F \xrightarrow{w} H$ 或者 $F \xrightarrow{w} p_i\epsilon, E \xrightarrow{w} H$；
- $p_i\alpha \sim_{l-k} H\alpha$。

并且我们有 $\mathsf{EL}_{\equiv}(E\alpha, F\alpha) = \mathsf{EL}_{\equiv}(EV\alpha, FV\alpha)$，其中 $V = \{p_i, H\}$。

引理5.4描述了这样的一个直观，如果 $\mathsf{EL}_{\equiv}(E, F)$ 严格比 $\mathsf{EL}_{\equiv}(E\alpha, F\alpha)$ 小，那么在从 (E, F) 开始的等价步数下降路径中，最后一定要将其中一个进程的栈做空。否则在 $(E\alpha, F\alpha)$ 的博弈中，攻击者可以按照 (E, F) 博弈的策略，在 $\mathsf{EL}_{\equiv}(E, F)$ 步能赢得博弈。这就意味着 $(E\alpha, F\alpha)$ 和 (E, F) 的等价步数相等，与假设矛盾。所以每当出现 $\mathsf{EL}_{\equiv}(E, F)$ 严格比 $\mathsf{EL}_{\equiv}(E\alpha, F\alpha)$ 小的情况时，我们就可以定义一个递归常量。这里我们复用平衡操作的符号 $\odot$：

$$(E\alpha, F\alpha) \odot (EV\alpha, FV\alpha)$$

对于一个给定的递归常量 V，定义

$$\mathrm{Var}(V) = \{i : V(i) = p_i\epsilon\}$$

如果 $|\mathrm{Var}(V)| = 0$，我们称 V 是完全的。

引理 5.5 如果递归常量 V 是完全的，那么对于任意的 $\alpha \in \Gamma^*$，$PV\alpha \sim PV$。

引理5.5说明，如果一个递归常量 V 是完全的，我们可以考虑将 V 之后的部分去掉。事实上这一部分不会再被访问到了。我们称这个操作为切割操作。

5.2 下推自动机互模拟上界分析

在文章 [96] 和 [71] 中，算法的分析是在一阶文法的框架下叙述的。在本节中，我们首先给出一个该证明在下推自动机框架下的重述。我们固定一个下推自动机 $\mathcal{A} = (Q, \Gamma, \Sigma, \mathcal{R})$。我们将回顾 Jančar 的定理 [96]：

定理 5.6 状态数给定为 d 的下推自动机的互模拟等价问题是$\mathbf{F}_{d+4}$ 中的问题。

5.2.1 平衡策略

本小节将会介绍在下推自动机互模拟等价研究中的一个经典的平衡策略 [34][76][96][111]。如果从进程 P, Q 开始，攻击者选择 $P \xrightarrow{r} P'$（$Q \xrightarrow{r'} Q'$），防御者选择 $Q \xrightarrow{r'} Q'$（$P \xrightarrow{r} P'$），我们将这样一轮操作记作：

$$\begin{array}{ccc} P & \xrightarrow{r} & P' \\ Q & r' & Q' \end{array}$$

我们的目标是控制如下等价步数下降的路径的长度：

$$\pi_0 : \begin{array}{ccccccc} P_0 & \xrightarrow{r_1} & P_1 & \xrightarrow{r_2} & \cdots & \xrightarrow{r_m} & P_m \\ Q_0 & r'_1 & Q_1 & r'_2 & & r'_m & Q_m \end{array} \tag{5–3}$$

我们将会对 π_0 做若干次平衡操作，在第 j 次操作结束后得到新的路径 π_j。第 j 次平衡操作将会进行在 π_{j-1} 的一个后缀 π'_{j-1} 上。我们令 $\pi'_0 = \pi_0$。

第 j 次平衡操作

1. π'_{j-1} 起始进程对记作 (S_0, T_0)，如果 S_0 中有简单常量，那么直到该简单常量出栈前，我们将不允许右平衡操作；如果是 T_0 中有简单常量，那么直到该简单常量出栈前，那么我们将不允许左平衡操作。这么做的目的是防止出现简单常量的嵌套。我们找到最早出现的可以进行平衡操作的进程对 (S_i, T_i)，并进行平衡操作。

$$\rho_j : \begin{matrix} S_0 \\ T_0 \end{matrix} \xrightarrow[u']{u} \begin{matrix} S_i \\ T_i \end{matrix} \xrightarrow[v']{v} \begin{matrix} S'_i \\ T'_i \end{matrix} \odot \begin{matrix} S''_i \\ T''_i \end{matrix}$$

2. 接下来，我们将从 (S''_i, T''_i) 开始确定一个等价步数下降路径作为 π'_j。本轮平衡操作后，新的路径为：

$$\pi_j : \rho_1 \rho_2 \cdots \rho_j \pi'_j$$

由于引理5.2，π_j 的长度始终都等于 π_0。因此不管做多少次平衡操作，我们的目标始终是给出路径长度的上界。

5.2.2 等价步数的上界分析

对于路径 π_j 的长度，Jančar 做了细致的分析 [96]。本小节中，我们将会介绍一些直观。

首先，我们可以列出每次平衡操作的结果：

$$(E_1\beta_1, F_1\beta_1),\ (E_2\beta_2, F_2\beta_2),\ \cdots,\ (E_k\beta_n, F_k\beta_n) \tag{5–4}$$

在序列 (5–4) 中，虽然 $(E_i\beta_i, F_i\beta_i)$ 和 $(E_{i+1}\beta_{i+1}, F_{i+1}\beta_{i+1})$ 之间的步数没有一个上界，然而我们可以观察到，因为我们的平衡策略是尽可能早的做平衡操作，所以除非两个进程都在不断的做出栈操作，否则我们可以很快的达成做平衡操作的条件。如果两次平衡操作间隔很长，那么之前必定要有很多次的间隔很短的平衡的操作积累出足够的栈的高度。因此可以借助栈的高度做一个均摊分析，可以给出两次平衡操作的平均间隔一个上界。由此，我们只需要分析出序列 (5–4) 的长度的界，就能够给出路径 (5–3) 长度的界。

序列 (5–4) 中的每一个后缀 β_i 都不同，这给分析它的界带来了困难。然而，如果 β_i 本身就是 β_{i+1} 的后缀，即存在 α 使得 $\beta_i\alpha = \beta_{i+1}$，那么我们可以将

$(E_{i+1}\beta_{i+1}, F_{i+1}\beta_{i+1})$ 重写为 $(E'_{i+1}\beta_i, F'_{i+1}\beta_i)$，其中 $E'_{i+1} = E_{i+1}\alpha$，$F'_{i+1} = F_{i+1}\alpha$。这样，序列 (5–4) 可以被分为若干段，其中每一段都可以重写为：

$$(S_1\beta, T_1\beta),\ (S_2\beta, T_2\beta),\ \cdots,\ (S_z\beta, T_z\beta) \tag{5–5}$$

我们的划分方式是让总段数尽可能少。可以证明，总共的段数和要验证的互模拟进程的初始栈高度有关，不超过初始栈高度的双指数。因此如果给出了序列 (5–5) 长度的上界，就可以给出序列 (5–4) 长度的上界。

需要注意的一点是，在序列 (5–5) 中，由于我们为了保证后缀一致，在 (5–4) 的基础上对其进行了重写，因此 S_i 和 T_i 的大小可能超出引理5.3中给出的界，但是起始进程 S_1 和 T_1 的大小依然满足这个界。并且 S_i 和 T_i 的栈高度的增长速度也是有界的，如(5–6)所示。这对之后分析序列 (5–5) 的长度起到关键作用。

$$|S_i, T_i| \leq h + g(i-1) \tag{5–6}$$

其中 h 是输入的多项式，而 g 是输入的一个指数。

我们将序列 (5–5) 的长度的上界记作 $\mathcal{E}$。下面的引理总结了我们这一小节中介绍的直观分析。

引理 5.7 (Jančar[96]) 对于下推自动机 $\mathcal{A} = (Q, \Gamma, \Sigma, \mathcal{R})$ 和给定的两个进程 P,Q，如果 $P \sim Q$，那么：

$$\mathsf{EL}(P, Q) \leq c \cdot (\mathcal{E} \cdot |P, Q| + |P, Q|^2)$$

其中 $c \leq f(|\mathcal{A}|, |P|, |Q|)$，$f$ 是一个双指数函数。

在下一节中，我们的研究重点就是分析序列 (5–5) 的长度，给出 $\mathcal{E}$ 的一个上界。

5.2.3 平衡结果序列长度上界分析

本节中，我们将会引入一个递归常量 V_0。首先我们将 V_0 初始化为：

$$p_1V_0 = p_1\epsilon,\ p_2V_0 = p_2\epsilon,\ \cdots,\ p_kV_0 = p_k\epsilon$$

可以看到，V_0 可以插入一个 PDA 进程的任意位置而不对进程做任何的改变。我们可以将等价步数下降序列 (5–5) 重写为：

$$(S_1V_0\beta, T_1V_0\beta),\ (S_2V_0\beta, T_2V_0\beta),\ \cdots,\ (S_zV_0\beta, T_zV_0\beta) \tag{5–7}$$

下面我们简述如何给出序列(5–7)的长度 z。根据5.2.2小节的分析，序列(5–7)中最初始的一对进程 S_1V_0, T_1V_0 的大小有一个上界，记作 h_0。因此我们可以定义如下集合：

$$\mathrm{PAIRS}_{\mathrm{S}:n} \stackrel{\text{def}}{=} \left\{(EV, FV) : E, F \in \mathcal{Q} \times (\Gamma \cup \Gamma_{SC})^*,\ \mathrm{Var}(V) = n\right\}$$

$$\mathrm{PAIRS}_{\mathrm{H}\leq h} \stackrel{\text{def}}{=} \left\{(EV, FV) : E, F \in \mathcal{Q} \times (\Gamma \cup \Gamma_{SC})^*,\ |EV| \leq h, |FV| \leq h\right\}$$

那么 (S_1V_0, T_1V_0) 的所有可能取值的集合为：

$$\mathcal{P}_0 = \mathrm{PAIRS}_{\mathrm{S}:n} \cap \mathrm{PAIRS}_{\mathrm{H}\leq h_0}$$

我们把集合 $\mathcal{P}_0$ 中所有不互模拟同余的进程对最大的等价步数记作：

$$e_0 = \max\{\mathsf{EL}_{\equiv}(E, F) : E \nsim F, (E, F) \in \mathcal{P}_0\}$$

根据引理5.4，如果序列(5–7)的长度超过了 e_0，那么一定存在 p_i 和 H，$H \neq p_i$，使得 $p_iV_0\beta \sim_{l-k} HV_0\beta \sim_{l-k} HV_1\beta$，其中 $V_1 = V_0\{(p_i, H)\}$。我们可以将序列 (5–7) 从第 $e+2$ 项开始的子序列

$$(S_{e+2}V_0\beta, T_{e+2}V_0\beta),\ (S_{e+3}V_0\beta, T_{e+3}V_0\beta),\ \cdots,\ (S_zV_0\beta, T_zV_0\beta)$$

改写为

$$(S_{e+2}V_1\beta, T_{e+2}V_1\beta), (S_{e+3}V_1\beta, T_{e+3}V_1\beta), \cdots, (S_zV_1\beta, T_zV_1\beta) \tag{5–8}$$

其中，$V_1 = V_0\{(p_i, H)\}$。我们可以给现状序列的起始进程对 $(S_{e+2}V_1, T_{e+2}V_1)$ 的大小一个上界：

$$h_1 = h_0 + e_0 \cdot g$$

而 $\mathrm{Var}(V_1)$ 相比 $\mathrm{Var}(V_0)$ 都会严格的下降 1。直到 V_0 成为一个完全的递归常量。当下推自动机有 n 个状态时，因此递归常量的更新最多进行 n 次。序列 (5–7) 可以被分成 $n+1$ 段。

$$\begin{array}{llll}
(S_1V_0\beta, T_1V_0\beta), & (S_2V_0\beta, T_2V_0\beta), & \cdots, & (S_{i_1}V_0\beta, S_{i_1}V_0\beta)\ \odot \\
(S_{i_1}V_1\beta, T_{i_1}V_1\beta), & (S_{i_1+1}V_1\beta, T_{i_1+1}V_1\beta), & \cdots, & (S_{i_2}V_1\beta, S_{i_2}V_1\beta)\ \odot \\
\cdots & & & \\
(S_{i_n}V_n\beta, T_{i_n}V_n\beta), & (S_{i_n+1}V_n\beta, S_{i_n+1}V_n\beta), & \cdots, & (S_zV_n\beta, T_zV_n\beta)
\end{array}$$

我们可以给出序列 (5–7) 长度的上界：

$$\mathcal{E} \leq \sum_{0\leq i\leq n} (e_i + 1) \tag{5–9}$$

而计算出这个上界 $\mathcal{E}$ 的关键是如何分别找到集合 $\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_n$ 中等价步数最大的一对进程。并且如何确定不存在等价步数更大的一对进程。Jančar 和 Schmitz 给出了一个算法，见算法5–1。

算法 5–1 计算 $\mathcal{E}$

输入: 下推自动机 $\mathcal{A} = (Q, \Gamma, \Sigma, \mathcal{R})$

输出: $\mathcal{E}$

$\mathcal{B} \coloneqq 0$;
for $i \coloneqq 0$ **to** n **do**
  $e_i \coloneqq 0$;
end
$h_0 \coloneqq 2 \cdot d_0$;
for $i \coloneqq 1$ **to** n **do**
  $h_i \coloneqq 2h_{i-1} + g$;
end
$\mathcal{E} = n + 1$;
for $i \coloneqq 0$ **to** n **do**
  $\mathcal{P}_i \coloneqq \mathrm{PAIRS}_{\mathrm{S}:i} \cap \mathrm{PAIRS}_{\mathrm{H}\leq h_i}$;
end
while $\exists i \in [0, n], \exists (E, F) \in \mathcal{P}_i, \mathit{EL}_{\equiv}(\mathcal{E}, E, F) < \infty$ **do**
  $e \coloneqq \mathsf{EL}_{\equiv}(\mathcal{E}, E, F)$;
  $\mathcal{P}_i \coloneqq \mathcal{P}_i \setminus \{(E, F)\}$;
  $\mathcal{B} \coloneqq \mathcal{B} \cup \{(E, F)\}$;
  if $e > e_i$ **then**
    for $j \coloneqq i + 1$ **to** n **do**
      $h_j = h_{j-1} + g \cdot e_{j-1}$;
      $e_j = \max\{\mathsf{EL}_{\equiv}(E, F) : (E, F) \in \mathcal{B} \cap \mathrm{PAIRS}_{\mathrm{S}:j} \cap \mathrm{PAIRS}_{\mathrm{H}\leq h_j}\}$;
      $\mathcal{P}_j = (\mathrm{PAIRS}_{\mathrm{S}:j} \cap \mathrm{PAIRS}_{\mathrm{H}\leq h_j}) \setminus \mathcal{B}$;
    end
    $\mathcal{E} = \sum_{0 \leq j \leq n} (e_j + 1)$;
  end
end

算法中定义了:

$$\mathsf{EL}_{\equiv}(\mathcal{E}', P, Q) \stackrel{\text{def}}{=} \begin{cases} \mathsf{EL}_{\equiv}(P, Q) & \text{如果} \mathsf{EL}_{\equiv}(P, Q) \leq c \cdot (\mathcal{E}' \cdot |P, Q| + |P, Q|^2) \\ \infty & \text{如果} \mathsf{EL}_{\equiv}(P, Q) > c \cdot (\mathcal{E}' \cdot |P, Q| + |P, Q|^2) \end{cases}$$

其中 $c \leq f(|\mathcal{A}|, |E|, |F|)$，$f$ 是一个双指数函数。

算法中变量 $\mathcal{B}$ 的作用是收集所有算法中涉及到的不互模拟同余的进程对。算法的一个关键点在于计算一对进程的等价步数。给定一对进程 (P, Q)，没有一个简单的算法能够计算 $\mathsf{EL}_{\equiv}(P, Q)$，但是我们可以通过简单的穷举计算出 $\mathsf{EL}_{\equiv}(\mathcal{E}', P, Q)$，因为 $\mathsf{EL}_{\equiv}(\mathcal{E}', P, Q)$ 的定义中设定了一个搜索的上限。Jančar 和 Schmitz 证明了这样一个上限的设定不会影响算法5–1的正确性，因为在找出所有的不互模拟同余的进程对之前，总会存在至少一对进程的等价步数是在这个界之下的（文章 [71] 的定理 5）。

接下来我们讨论算法5–1的计算复杂性。最重要的问题是算法第13行的循环会执行多少遍。每次在集合 $\mathcal{P}_i$ 中找到一个新的不互模拟同余的进程时，都会更新集合 $\mathcal{P}_{i+1}, \cdots \mathcal{P}_n$。可以定义一个序数：

$$\lambda = \omega^n \cdot |\mathcal{P}_0| + \omega^{n-1} \cdot |\mathcal{P}_1| + \cdots \omega^0 \cdot |\mathcal{P}_n|$$

这个序数会随着每一次循环严格的下降，因此算法5–1的终止性可以保证。而要进一步的研究复杂性，就要对对应的序数序列做更多的控制。对于序数 $\lambda = \omega^n \cdot c_n + \omega^{n-1} \cdot c_{n-1} + \cdots \omega^0 \cdot c_0$，定义：

$$\|\lambda\| \stackrel{\text{def}}{=} \max \left\{ n, \max_{0 \le i \le n} \{c_i\} \right\}$$

对于一列序数 $\lambda_0, \lambda_1, \lambda_2, \cdots$，如果对于一个单调递增函数 $h : \mathbb{N} \to \mathbb{N}$，满足 $\|\lambda_k\| \le h^k(\lambda_0)$，我们称 h 是该列序数的控制函数。根据前面的分析可知，序列(5–7)对应的序数序列有一个双指数的控制函数。而根据 [112] 的定理 3.3 可知，(5–7)的长度：

$$\mathcal{E} \le F_{h,n+1}(|S_1, T_1|)$$

最后，根据引理2.6可以得到：

定理 5.6 状态数给定为 d 的下推自动机的互模拟等价问题是$\mathbf{F}_{d+4}$ 中的问题。

5.3 赋范下推自动机互模拟上界分析

在本节中，我们将分析当下推自动机是赋范的情况下，算法可能的优化。我们可以看到，在 PDA 上界的分析中用到了快速增长复杂性类的相对层级，其中，一部分是控制函数，控制每一步增长的大小，这里是一个不超过$\mathbf{F}_3$ 的函数；第二部分是最大序数，控制一共有多少步，这里是$\mathbf{F}_{d+1}$。在本节中，我们将证明，如果下推自动机是赋范的，那么最大序数只需要是$\mathbf{F}_d$ 即可。因此可以将之前$\mathbf{F}_{d+4}$ 的上界降低为$\mathbf{F}_{d+3}$。

在赋范下推自动机中，所有进程都是赋范的。对于生成的递归常量 V，$V(i)$ 也是赋范的，下述命题5.8是显然的。

命题 5.8 在赋范下推自动机中生成的递归常量都是赋范的。

对于赋范下推自动机，除了栈为空的时候，一个系统永远都可以做动作。因此两个进程是否互模拟和两个进程的范数也有一些关系。

命题 5.9 P, Q 是赋范下推自动机的两个进程。

1. 如果 $P \sim Q$，那么 $\|P\| = \|Q\|$。
2. 如果 $\|P\| \neq \|Q\|$，那么 $\mathsf{EL}(P, Q) \leq \min\{\|P\|, \|Q\|\}$。

证明 对于第一点，我们用反证法，假设 $\|P\| \neq \|Q\|$。不失一般性的，我们可以假定 $\|P\| < \|Q\|$，令 $\|P\| = k$。那么存在一条从进程 P 开始的范数下降路径：

$$P = P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} P_k \tag{5–10}$$

满足

- $|P_k| = 0$，即 P_k 的栈为空；
- 对于 $0 \leq i \leq k-1$，$\|P_{i+1}\| = \|P_i\| - 1$。

因为 $P \sim Q$，所以 Q 可以模拟 (5–10) 中的动作 $Q \xrightarrow{a_1 \cdots a_k} Q'$。因为 $\|Q\| > k$，所以 Q' 的栈不为空。因为 Q' 也是一个赋范进程，因此它可以继续做动作，而 P_k 不能做任何动作，所以 $P_k \not\sim Q'$，与 $P \sim Q$ 矛盾。

对于第二点，也是同样的原因，攻击者可以选择 (5–10) 中的动作，在 k 步之后，有不能模拟的动作。 □

需要说明的是，对于一般的下推自动机来说，命题5.9不一定成立。比如一个 PDA 中有两条规则：$pX \xrightarrow{a} pX, pX \xrightarrow{b} p$。那么 $pX \sim pXZ$，但是 $\|pXZ\| = \infty$，而 $\|pX\| = 1$。

引理 5.10 在赋范下推自动机中，$\mathcal{E}$ 的上界(5–9)可以改进为：

$$\mathcal{E} \leq (1 + d_0) \sum_{0 \leq j \leq n-1} \mathsf{EL}_{\equiv}(e_j + 1) \tag{5–11}$$

证明 回顾递归常量生成的过程。对于序列 (5–7) 中的 β，$|||\beta||| = \{n_1, n_2, \cdots, n_k\}$。我们令 $i = \max\{j : n_j = \min\{n_1, n_2, \cdots, n_k\}\}$，即范数最小的维度中下标最大的那

一维。当递归常量 V 要更新第 i 维时，假设 $p_i\beta = HV\beta$。根据我们对递归常量的要求，$|H| > 0$。因此一定有：

$$\|p_iV\beta\| = \|p_i\beta\| < \|HV\beta\|$$

根据性质5.9，我们知道 $\mathsf{EL}(p_iV\beta, HV\beta) \leq \|p_i\beta\| \leq d_0|\beta|$，而 $|\beta|$ 的值不超过此前的序列长度。因此，我们将会把之前式子 (5–9) 中的界改进到：

$$\mathcal{E} \leq (1 + d_0) \sum_{0 \leq j \leq n-1} \mathsf{EL}_{\equiv}(e_j + 1)$$

□

根据引理5.10，我们在算法5–1中，只需要记录 $\mathcal{P}_0, \mathcal{P}_1, \cdots, \mathcal{P}_{n-1}$ 一共 n 个集合，将算法第23行更新为(5–11)，之后的分析和5.2节一样。控制函数依旧是一个$\mathbf{F}_3$ 的函数，最大序数为 ω^d。根据引理2.6，可以证明如下定理。

定理 5.1 状态数给定为 d 的赋范下推自动机的互模拟等价问题是$\mathbf{F}_{d+3}$ 问题。

5.4 本章小结

在本章中，我们对 Jančar 和 Schmitz 关于一阶文法强互模拟复杂性的证明 [71] 在下推自动机模型上做了重述。当赋范下推自动机的状态数固定为 d 时，我们将之前$\mathbf{F}_{d+4}$ 的复杂性的上界改进为$\mathbf{F}_{d+3}$。

通过本章的重述，也为将来进一步研究其复杂性上界打下基础。因为实时下推自动机是一个比一阶文法更弱的模型，也许更容易在实时下推自动机上找到一些更好的性质。目前上界的分析考虑了很多最坏情况，例如每次找到一个新的不互模拟同余的进程对，$\mathcal{E}$ 的增长的分析都是按照最大可能计算的。而事实上，当找到一个新的不互模拟同余的进程对 (P, Q)，并且使得 $\mathcal{E}$ 增加一个很大的数字时，会有大量的等价步数低于 (P, Q) 的进程对不能再更新 $\mathcal{E}$ 的值。通过某种均摊分析或许可以得到一个更好的上界。而这一点对于一步动作后进程的栈高度变化不大的实时下推自动机模型来说可能更容易分析。

第六章　有限状态随机通信系统演算上的分支互模拟等价

本章我们研究有限状态的随机通信系统演算。在6.1节中，我们给出一个判定有限状态随机通信系统演算的分支互模拟的算法，并分析其复杂性。在6.2节中，我们给出一个公理系统，通过该公理系统可以得到所有满足分支互模拟关系的进程对。我们证明了该公理系统的可靠性和完备性。6.3节对本章内容做出总结。

6.1　判定算法

本节我们给出一个算法。该算法的一个关键问题是 ϵ-树有可能是无限大的，但是 ϵ-树有一个等价的有限表示，我们称作 ϵ-图，它将帮助我们设计出互模拟的判定算法。我们将会用到一个经典的算法：划分-细化算法（Partition-Refinement Algorithm）[40]。

6.1.1　树到图的转化

定义 6.1　给定一个进程 A 关于关系 $\mathcal{R}$ 的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$，我们可以定义一个由 $\mathcal{T}_{\mathcal{R}}^{A}$ 导出的 ϵ-图 $G_{\mathcal{R}}^{A}$。$G_{\mathcal{R}}^{A}$ 是一个有向带权图 (V, E)。

- $V = \{A : A \text{ 是} \mathcal{T}_{\mathcal{E}}^{A} \text{ 中的一个节点。}\}$
- $E = \{(A_1, p, A_2) : \mathcal{T}_{\mathcal{E}}^{A} \text{ 中存在从节点} A_1 \text{ 到} A_2, \text{并且标记为} p \text{的边。}\}$

如果 $G_{\mathcal{R}}^{A}$ 中的一个节点出度为 0，那么称该节点是一个下沉（sink）节点。我们用 $\text{sink}(G_{\mathcal{R}}^{A})$ 表示 $G_{\mathcal{R}}^{A}$ 中所有的下沉节点。

我们用一个例子来看 ϵ-树和 ϵ-图的对应。

例 6.1　定义进程：

$$H \overset{\text{def}}{=} \mu X.\left(\frac{1}{2}\tau.(a + \tau.X) \oplus \frac{1}{2}\tau.(b + \tau.X)\right)$$

定义等价关系：

$$\mathcal{R} = \{(H, a + \tau.H), (H, b + \tau.H), (a + \tau.H, b + \tau.H)\}$$

在图6–1(a) 中，左边一部分是 H 关于 $\mathcal{R}$ 的 ϵ-树 $\mathcal{T}_1$。右边一部分是 $\mathcal{T}_1$ 对应的 ϵ-图 G_1。G_1 中有一个下沉节点 $b + \tau.H$。在图6–1(b) 中是另一棵 H 关于 $\mathcal{R}$ 的 ϵ-树 $\mathcal{T}_2$，对应的 ϵ-图 G_2 中有一个下沉节点 $a + \tau.H$。而在图6–1(c) 中的 ϵ-树 $\mathcal{T}_3$，对应的 ϵ-图 G_3 中没有下沉节点。

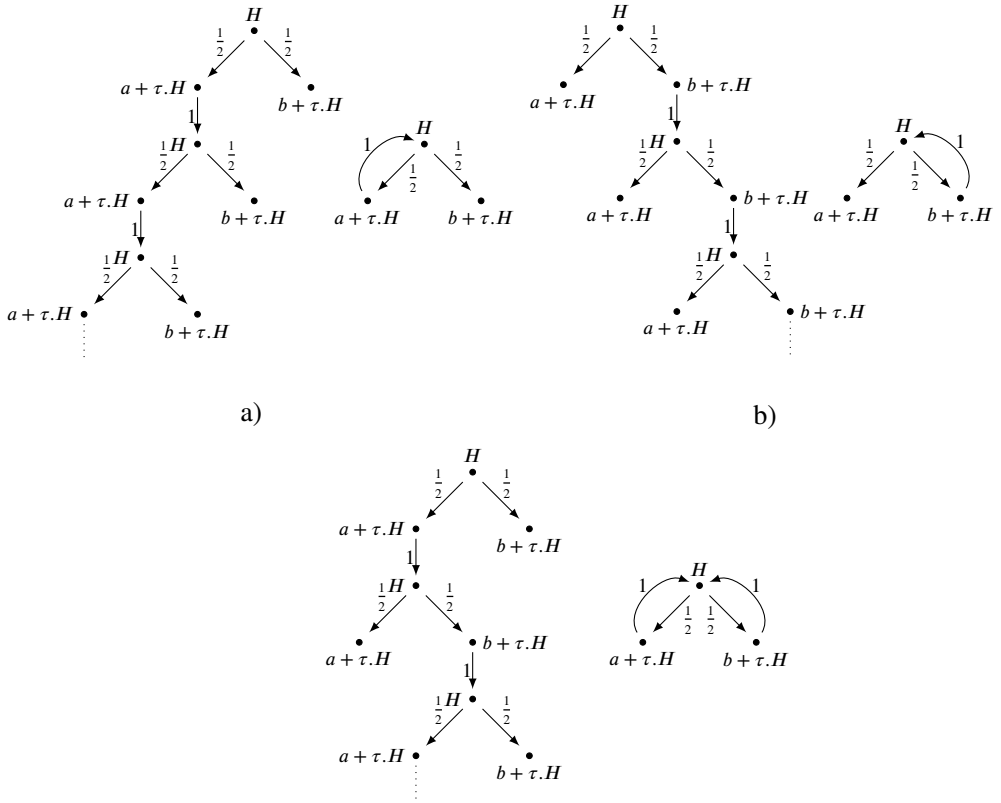


图 6–1 ϵ-树和对应的 ϵ-图的例子

定义 6.2 (正则的 ϵ-图) 对于一个 ϵ-图 G，如果 G 中的每一个节点都可以到达一个下沉节点，我们称 G 是正则的（regular）。

注 ϵ-树的正则性和 ϵ-图的正则性并不能完全保证一致。例如在图6–1(c) 中，ϵ-树满足正则性，但是对应的 ϵ-图不满足正则性。但是这不影响后续算法的正确性。可以保证的一点是，当我们找到一个正则的 ϵ-图时，我们一定可以按照如下方法构建一棵正则的 ϵ-树：如果一个正则的 ϵ-图从一个节点出发有一条边，那么 ϵ-树中对应节点一定会引入这条边。

ϵ-图和 ϵ-树在随机通信系统演算的互模拟定义中可以起到相同的作用。以 ℓ-迁移为例：存在一个从进程 A 到等价类 $\mathcal{B}$ 的关于 $\mathcal{R}$ 的 ℓ-迁移，当且仅当存在一个正则的 ϵ-图 $G^A_{\mathcal{R}}$，满足从 $G^A_{\mathcal{R}}$ 中的所有下沉节点 S 出发都有的一条迁移规则 $S \xrightarrow{\ell} S' \in \mathcal{B}$。

6.1.2 划分-细化算法

我们首先计算从某个进程 A 起始所有可达的进程集合 $R(A)$。按照 S 的语法深度归纳，可以很容易的证明如下命题。

命题 6.1 给定一个进程 $S \in \mathcal{P}_{fs}$，R_S 是一个有限集合。

我们给出算法6–1，它会根据输入进程的语法结构递归的将所有可达的进程都计算出来。

算法 6–1 计算 R_A

输入: 下推自动机 $A \in \mathcal{P}_{fs}$
输出: R_A

$R_A := \emptyset, R' := \{A\}$;
while $R' \neq \emptyset$ **do**
  Choose a process B from R', $R' := R' - \{B\}$, $R_A := R_A \cup \{B\}$;
  if $B = \sum_{i \in I} \alpha_i.T_i$ **then**
    $R' := R' \cup \{T_i : i \in I\} \setminus R_A$;
  end
  else if $B = \bigoplus_{i \in I} p_i\tau.T_i$ **then**
    $R' := R' \cup \{T_i : i \in I\} \setminus R_A$;
  end
  else if $B = \mu X.T$ **then**
    $R' := R' \cup \{T\{\mu X.T/X\}\} \setminus R_A$;
  end
end
return R_A;

例 6.2 我们以例6.1中的进程 H 为例，在算法6–1中输入 H，将会返回：

$$R_H = \left\{ H, \frac{1}{2}\tau.(a + \tau.H) \oplus \frac{1}{2}\tau.(b + \tau.H), a + \tau.H, b + \tau.H, \mathbf{0} \right\}$$

在介绍划分-细化算法前，我们先给出一些符号的定义。一个进程集合 $\mathcal{P}$ 的一个划分（partition）$\mathcal{X}$ 是一个集合，其中的元素是两两不相交的 $\mathcal{P}$ 的子集，并且保证对于任意一个元素 $A \in \mathcal{P}$，存在一个子集 $C \in \mathcal{X}$ 使得 $A \in C$。我们用 $\mathcal{E}_{\mathcal{X}}$ 表示由 $\mathcal{X}$ 诱导出的等价关系。而其中包括某个进程 A 的等价类表示为 $[A]_{\mathcal{X}}$。

给定同一个进程集合的两个划分 $\mathcal{X}_1$ 和 $\mathcal{X}_2$。如果 $\mathcal{X}_2$ 中的每一个集合都是 $\mathcal{X}_1$ 中某个集合的子集，我们说 $\mathcal{X}_1$ 是一个比 $\mathcal{X}_2$ 更粗略的划分（或者说 $\mathcal{X}_2$ 是一个比

$\mathcal{X}_1$ 更精细的划分）。直观上，我们的算法从一个包含了所有可达进程对的有限集合开始（最粗略的划分），不断的对它细化，直到不能再细化为止，最后得到的划分中，每一个集合中的元素在互模拟意义下都是相等的。

我们引入一个符号：$\hat{\varphi}\tau$。这个符号表示某个概率 $p\tau$，$0 < p \leq 1$。也就是说，当我们要强调一个动作是概率动作，而不关心具体的概率值时，我们将会用符号 $\hat{\varphi}\tau$ 来表示该动作。

定义 6.3 我们用 $\mathcal{X}$ 表示进程集合 $\mathcal{P}$ 的一个划分。一个划分的划分器（splitter）是一个三元组 (C_1, λ, C_2)，其中 $C_1, C_2 \in \mathcal{X}$ 是两个等价类，$\lambda \in S_n \cup \{\hat{\varphi}\tau\}$。并且满足下述两点之一：

1. 如果 $\lambda = \hat{\varphi}\tau$，并且 $C_1 \neq C_2$。那么存在 $A, A' \in C_1$ 和 $q \in (0, 1]$，使得进程 A 有一个 ϵ-图 $G^A_{\mathcal{X}}$，对于任意 $T \in \text{sink}(G^A_{\mathcal{X}})$，$P_{\mathcal{E}_{\mathcal{X}}}(T \xrightarrow{\coprod_{i\in[k]} p_i\tau} C_2) = q$。而从进程 A' 开始不存在这样的 ϵ-图 $G^{A'}_{\mathcal{X}}$。
2. 如果 $\lambda \in S_n$（当 $\lambda = \tau$ 时，$C_1 \neq C_2$）。那么存在两个进程 $A, A' \in C_1$，使得进程 A 有一个 ϵ-图 $G^A_{\mathcal{X}}$，满足 $\text{sink}(G^A_{\mathcal{X}})$ 中的所有节点都可以直接做 ℓ 动作到等价类 C_2。而从进程 A' 开始不存在这样的 ϵ-图 $G^{A'}_{\mathcal{X}}$。

如果存在一个划分器，说明我们找到了某一个集合中不互模拟的两个进程。我们可以依据该划分器来细化划分 $\mathcal{X}$。根据划分器定义的两种情况，我们分别给出对应的 Refine 函数的定义。

第一种情况，如果 $\mathcal{X}$ 上存在划分器 $(C_1, \hat{\varphi}\tau, C_2)$。也就是说 C_1 中存在两个不同的进程，它们到达等价类 C_2 的条件概率是不一样的。我们需要对等价类 C_1 进行细化。细化的过程如图6–2所示。我们用 $\text{tn}(C_1, \mathcal{X})$ 表示 C_1 中所有可以执行一步概率 τ 动作到一个由 $\mathcal{X}$ 划分的不同的等价性类的进程集合。我们先按照到达等价类 C_2 的概率来划分 $\text{tn}(C_1, \mathcal{X})$ 中的点。对于其余的点，如果它能做不改变状态的动作到达某个等价类，那么也将被划分入该类中。不能到任何等价类的点将被单独作为一类。

接下来我们给出严格的定义。对于 C_1 中的两个进程 A, A'，当 A 和 A' 满足：

$$P_{\mathcal{E}_{\mathcal{X}}}(A \xrightarrow{\coprod_{i\in[k]} p_i\tau} C_2) = P_{\mathcal{E}_{\mathcal{X}}}(A' \xrightarrow{\coprod_{i\in[k]} p_i\tau} C_2)$$

我们记作 $A =_p A'$。

我们将 $C_1 \cap \text{tn}(\mathcal{P}, \mathcal{X})$ 划分为 $(C_1 \cap \text{tn}(\mathcal{P}, \mathcal{X})) / =_p$。之后，对于每一个等价类 $\mathcal{B} \in (C_1 \cap \text{tn}(\mathcal{P}, \mathcal{X})) / =_p$，我们会在 $\mathcal{B}$ 中加入满足下列条件（记作 Δ_1）的进程 B：

1. $B \in C_1 \setminus \text{tn}(\mathcal{P}, \mathcal{X})$;

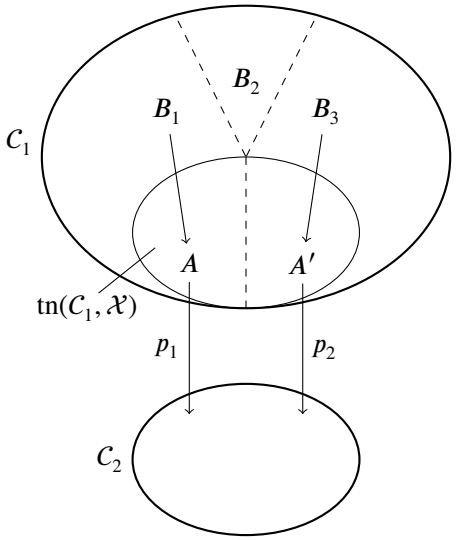


图 6–2 细化过程示意

2. 存在一个 ϵ-图 $G^B_{\mathcal{X}}$，$\mathrm{sink}(G^B_{\mathcal{X}}) \subseteq \mathcal{B}$;
3. 对于任意其他等价性类 $\mathcal{B}' \in (C_1 \cap \mathrm{tn}(\mathcal{P}, \mathcal{X}))\ /=_p$，不存在 ϵ-图 $G^B_{\mathcal{X}}$，使得 $\mathrm{sink}(G^B_{\mathcal{X}}) \subseteq \mathcal{B}'$。

我们用 $\overline{\mathcal{B}}$ 表示在 $\mathcal{B}$ 中加入了这些新的进程后的等价类。

最后，我们将其余进程划分为一类：

$$Res(C_1) \stackrel{\text{def}}{=} \left\{C \in C_1 : \text{对于任意} \mathcal{B} \in (C_1 \cap \mathrm{tn}(\mathcal{P}, \mathcal{X}))\ /=_p, C \text{ 都不满足性质 } \Delta_1\right\}$$

综合以上定义，如果一个划分 $\mathcal{X}$ 存在一个划分器 $(C_1, \hat{\varphi}\tau, C_2)$，我们可以对 $\mathcal{X}$ 按如下方法细化：

$$\mathrm{Refine}\left(\mathcal{X}, (C_1, \hat{\varphi}\tau, C_2)\right) \stackrel{\text{def}}{=} \left(\mathcal{X} \setminus \{C_1\}\right) \cup \left\{\overline{\mathcal{B}} : (\mathcal{B} \in C_1 \cap \mathrm{tn}(\mathcal{P}, \mathcal{X}))\ /=_p\right\} \cup \left(\{Res(C_1)\} \setminus \{\varnothing\}\right)$$

第二种情况，如果 $\mathcal{X}$ 有一个划分器 (C_1, λ, C_2)，$\lambda \in \mathcal{A}_n$，如果 $\lambda = \tau$ 时，要求 $C_1 \neq C_2$。即内部动作要求改变状态。这种情况要简单一些，我们直接将 C_1 中的进程按照是否能做一个 λ-迁移来将其分为两部分。

对于 C_1 中的一个进程 B，定义如下性质（记作 Δ_2）：

存在 ϵ-图 $G^B_{\mathcal{X}}$, 所有节点 $\mathrm{sink}(G^B_{\mathcal{X}})$ 可以直接做 λ 动作到等价类 C_2 中。

定义：$\mathcal{D} \stackrel{\text{def}}{=} \left\{B \in C_1 : B \text{ 满足性质 } \Delta_2\right\}$。如果一个划分 $\mathcal{X}$ 存在一个划分器 (C_1, λ, C_2)，我们可以对 $\mathcal{X}$ 按如下方法细化：

$$\mathrm{Refine}\left(\mathcal{X}, (C_1, \lambda, C_2)\right) \stackrel{\text{def}}{=} \left(\mathcal{X} \setminus \{C_1\}\right) \cup \mathcal{D} \cup (C_1 \setminus \mathcal{D})$$

每做一次细化操作，得到的新的划分会比原来的划分严格的精细。直到得到划分 $\mathcal{P}/\simeq$ 为止。我们有下述性质：

命题 6.2 $\mathcal{X}$ 是进程集合 $\mathcal{P}$ 的划分，如果 $\mathcal{X}$ 不能再被细化，那么 $\mathcal{X} = \mathcal{P}/\simeq$.

我们给出验证算法6–2。命题6.2说明了算法的正确性。

算法 6–2 随机通信系统演算分支互模拟算法

输入: 进程 A , B
输出: $A \simeq B$ 是否为真

```
1  计算 R := R_A ∪ R_B;
2  X := {R};
3  while X 上有一个 划分器 (C_1, λ, C_2) do
4  |   X := Refine(X, (C_1, λ, C_2));
5  end
6  if [A]_X = [B]_X then
7  |   返回 true;
8  end
9  else
10 |   返回 false;
11 end
```

复杂性分析

- 在第1行中，会调用算法6–1。存在常数 c，使得 $|R| \leq c \cdot (|A| + |B|)$，时间复杂度为 $\mathcal{O}((|A| + |B|))$。
- 在第3行中，循环条件需要判断一个划分是否可以细化。对于每一个 λ，我们会依次对每个等价类 C_1 做深度优先搜索。搜索策略是当遇到一个可以直接做动作 λ 的进程时回溯。如果 $\lambda = \hat{\varphi}\tau$ 或者 $\lambda = \tau$ 时，回溯的条件还需要动作结果在另外一个等价类中。我们可以标记每一个节点能够到达的等价类有哪些。以此来判定是否存在不互模拟的两个进程。在此过程中，λ 可以有 $\mathcal{O}(|A| + |B|)$ 种不同的取值；对所有等价类中节点的深度优先搜索一共需要 $\mathcal{O}(|A| + |B|)$ 时间；而比较两个进程是否能到相同的等价类需要 $\mathcal{O}(|A| + |B|)$ 时间。因此总的时间复杂度是 $\mathcal{O}((|A| + |B|)^3)$。
- 有了在算法第3行中的标记工作，算法第4行中只需要按照 Refine 的定义将划分细化。这一步需要 $\mathcal{O}((|A| + |B|))$ 时间。

- 每次循环过后，得到的新的划分会比原来的划分严格精细。总的循环次数不超过 $\mathcal{O}((|A|+|B|))$ 次，因此总的时间复杂度为 $\mathcal{O}((|A|+|B|)^4)$。

通过以上分析，我们有以下定理：

定理 6.3 有限状态的随机通信系统演算上的分支互模拟等价可以在 $\mathcal{O}(n^4)$ 的时间内判定。

6.2 有限公理化

在本节中，我们将会构建一个有限随机通信系统演算分支互模拟的有限公理系统。在6.2.1小节中，我们介绍一个新的概念：概率被守卫的。在6.2.2小节中，我们给出一个有限随机通信系统演算分支互模拟的有限公理系统 $\mathcal{A}$。在6.2.3和6.2.4小节中，我们将分别证明该公理系统的可靠性和完备性。

6.2.1 概率被守卫的

在带不动点算子的进程演算弱互模拟和分支互模拟的公理化中，通常第一步是将一个任意的项转换成一个强守卫的项 [99, 100]。在强互模拟的公理化中，只需要将一个任意的项转换成一个弱守卫的项即可 [90]。

但是在概率模型中，有一些项不能转换成强守卫的项，主要原因是在概率模型中，一个 τ 动作（可能包括概率 τ 动作）如果构成循环，并不能保证每一个动作都在 $\simeq$ 下不改变状态。

例 6.3 定义一进程 G：

$$G = \mu X.\left(\frac{1}{2}\tau.(\frac{1}{3}\tau.X \oplus \frac{2}{3}\tau.a) \oplus \frac{1}{2}\tau.b\right)$$

该进程无法转换成一个强守卫的项。

命题 6.4 对于任意的强守卫的项 P，$P \not\simeq G$。

证明 我们采用反证法，假定存在强守卫的项 P，并且有 $P \simeq G$。

- 如果 P 中没有变量，那么 P 将在有限步终止。令 $F = \frac{1}{3}\tau.G \oplus \frac{2}{3}\tau.a$, $F \not\simeq G$，那么 P 无法模拟

$$G \rightsquigarrow_{\simeq} \xrightarrow{\frac{1}{2}} [F] \rightsquigarrow_{\simeq} \xrightarrow{\frac{1}{3}} [G] \rightsquigarrow_{\simeq} \xrightarrow{\frac{1}{2}} [F] \rightsquigarrow_{\simeq} \xrightarrow{\frac{1}{3}} [G] \cdots$$

表 6–1 不同“被守卫的”概念的例子

	弱守卫的	概率被守卫的	强守卫的
X	✗	✗	✗
$\frac{1}{2}\tau.X \oplus \frac{1}{2}\tau.X$	✓	✗	✗
$\frac{1}{2}\tau.a \oplus \frac{1}{2}\tau.X$	✓	✓	✗
$a.X$	✓	✓	✓

- 如果 P 中有变量，因为 P 是强守卫的，所以 P 的形式为 $\mu X.(\cdots a.(\cdots X \cdots)\cdots)$，但是 G 无法模拟

$$P \cdots \rightsquigarrow_{\simeq} \xrightarrow{a} \cdots P \cdots \rightsquigarrow_{\simeq} \xrightarrow{a} \cdots$$

综合以上两点，得出矛盾。因此 $P \not\simeq G$。 □

接下来，我们定义概率被守卫的。直观上，概率被守卫的和强被守卫的两者的区别是，进程 T 中，只要 T 不是以总概率为 1 做 τ 动作到 X ，那么 X 在 T 中是概率被守卫的。

定义 6.4 给定 $T \in S_{fs}$，X 是一个变量。我们有以下的递归定义：

- 如果 $T = X$，X 在 T 中不是概率被守卫的。
- 如果 $T = \mathbf{0}$，或者 $T = Y \neq X$，X 在 T 中是概率被守卫的。
- $T = \sum_{i\in I} \alpha_i.T_i$。如果对于任意的 $i \in I$，X 在 T_i 中是概率被守卫的，那么 X 在 T 中是概率被守卫的。
- $T = \bigoplus_{i\in I} p_i\tau.T_i$。如果存在 $i \in I$，X 在 T_i 中是概率被守卫的，那么 X 在 T 中是概率被守卫的。
- $T = \mu X.T'$，如果 X 在 T' 中是概率被守卫的，那么 X 在 T 中是概率被守卫的。

对于一个进程 E，如果 E 中所有出现的 $\mu X.T$ 中，X 在 T 中都是概率被守卫的，那么我们称进程 E 是概率被守卫的。我们用 $\mathcal{P}^{g}_{fs}$ 表示所有概率被守卫的进程集合。表6–1中列举了一些例子来说明概率被守卫的和其他被守卫的之间的区别。

虽然概率被守卫的其定义较为复杂，但是判定一个项是否是概率被守卫的并不会比强守卫的更难。只需要按照项的语法递归即可。

命题 6.5 可以在线性时间内判定一个项是否是概率被守卫的。

6.2.2 概率系统分支互模拟公理化

一个公理系统 $\mathcal{A}$ 是一个公理的集合。如果通过 $\mathcal{A}$ 中的公理可以推导出命题 P，我们记作 $\mathcal{A} \vdash P$。当我们希望强调 P 的证明源自 $\mathcal{A}$ 中某个公理 R 时，我们也会记作 $R \vdash P$。

我们给出一个有限状态随机通信系统演算分支互模拟上的公理系统 $\mathcal{A}$。

公理系统 $\mathcal{A}$

$E1$ $T = T$

$E2$ 如果 $S = T$，那么 $T = S$

$E3$ 如果 $S = T$，$T = R$，那么 $S = R$

$E4$ 如果对于 $i \in I$，$S_i = T_i$，那么 $\sum_{i\in I} \alpha_i.S_i = \sum_{i\in I} \alpha_i.T_i$

$E5$ 如果对于 $i \in I$，$S_i = T_i$，$\sum_{i\in I} p_i = 1$，那么 $\bigoplus_{i\in I} p_i\tau.S_i = \bigoplus_{i\in I} p_i\tau_i.T_i$

$E6$ 如果 $S = T$，那么 $\mu X.S = \mu X.T$

$A1$ $\bigoplus_{i\in I} p_i\tau.S_i \oplus p\tau.S \oplus q\tau.S = \bigoplus_{i\in I} p_i\tau.S_i \oplus (p+q)\tau.S,\ p+q<1$

$A2$ $p\tau.S \oplus q\tau.S = \tau.S$

$B1$ $\left(\sum_{i\in I'\subseteq I} \alpha_i.S_i\right) + \tau.\left(\sum_{i\in I} \alpha_i.S_i\right) = \sum_{i\in I} \alpha_i.S_i$

$B2$ 如果 $\dfrac{p_1}{q_1} = \cdots = \dfrac{p_i}{q_i} < 1$，$\sum_{i\in I} q_i = 1$，
那么 $\bigoplus_{i\in I} p_i\tau.S_i \oplus p\tau.\left(\bigoplus_{i\in I} q_i\tau.S_i\right) = \bigoplus_{i\in I} q_i\tau.S_i,\ p = 1 - \sum_{i\in I} p_i$

$R1$ $\mu X.T = T\{\mu X.T/X\}$

$R2$ 如果 $S = T\{S/X\}$，那么 $S = \mu X.T$；其中 X 在 T 中是概率被守卫的

$R3$ $\mu X.\left(\tau.X + \sum_{i\in I} \alpha_i.T_i\right) = \mu X\left(\sum_{i\in I} \alpha_i.T_i\right)$

$R4$ $\mu X.\left(\tau.(\tau.S + \sum_{i\in I} \alpha_i.T_i) + \sum_{j\in J} \beta_j.R_j\right) = \mu X\left(\tau.S + \sum_{i\in I} \alpha_i.T_i + \sum_{j\in J} \beta_j.R_j\right)$，
其中 X 在 S 中不是概率被守卫的

$R5$ $\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right) = \mu X.\left(\sum_{i\in I} \tau_i.S_i\right)$，
对于任意 $i \in I$，X 在 S_i 中不是概率被守卫的

其中，公理 $E1-4, 6, B1, R1$ 都是直接引自经典模型中的公理 [99, 100]（我们

重写了公理 $B1$ 来适应我们的语法)；此外我们提出公理 $E5, A1-2, B2, R2-5$ 来帮助实现概率扩展部分的公理化。

6.2.3 公理系统可靠性

公理 $E1-6$ 的可靠性可以由命题2.5直接得到。

公理 $A1$，$A2$，$B1$，$R3$ 的可靠性可以由分支互模拟的定义直接证明。

公理 $R1$ 的可靠性可以由以下性质得到

$$\mu X.T \xrightarrow{\alpha} F \iff T\{\mu X.T/X\} \xrightarrow{\alpha} F$$

本小节的剩余内容将证明其他公理的可靠性。

命题 6.6 ($B2$ 的可靠性) 如果 $\frac{p_1}{q_1} = \cdots = \frac{p_i}{q_i} < 1$，$\sum_{i\in I} q_i = 1$，那么 $\bigoplus_{i\in I} p_i\tau.E_i \oplus p\tau.(\bigoplus_{i\in I} q_i\tau.E_i) \simeq \bigoplus_{i\in I} q_i\tau.E_i, p = 1 - \sum_{i\in I} p_i$。

证明 令 $F_1 = \bigoplus_{i\in I} q_i\tau.E_i$，$F_2 = \bigoplus_{i\in I} p_i\tau.E_i \oplus p\tau.(\bigoplus_{i\in I} q_i\tau.E_i)$。我们考虑下列两种情况：

- $\forall i \in I, E_i \simeq F_1$。这种情况是比较显然的：

$$F_2 \simeq \bigoplus_{i\in I} p_i\tau.F_1 \oplus p\tau.(\bigoplus_{i\in I} q_i\tau.F_1) \simeq F_1$$

□

- $\exists i \in I, E_i \not\simeq F_1$。
  令 $I' \subseteq I$ 表示所有满足 $E_{i'} \not\simeq F_1$ 的下标 i' 的集合。令 $r_{i'} = \frac{q_{i'}}{\sum_{i'\in I'} q_{i'}}$，那么我们有 $F_1 \rightsquigarrow_{\simeq} \xrightarrow{r_{i'}} [E_{i'}]_{\simeq}$。可以验证，$F_2$ 到 $[E_{i'}]_{\simeq}$ 的条件概率为 $\frac{p_{i'}}{\sum_{i'\in I'} p_{i'}} = r_{i'}$，$F_2$ 和 F_1 有相同的 q-迁移，因此 $F_1 \simeq F_2$。

我们接下来介绍一个经典的证明互模拟关系的技术：

定义 6.5 如果一个等价关系 $\mathcal{R} \subseteq \mathcal{P}_{fs} \times \mathcal{P}_{fs}$ 满足：

- 如果 $E\mathcal{R}F$ 并且 $E \rightsquigarrow_{\simeq} \xrightarrow{\ell} \mathcal{B}_1$，$\ell \neq \tau \vee \mathcal{B}_1 \neq [E]_{\simeq}$，那么存在 E', F' 使得 $E' \in \mathcal{B}_1 \wedge F' \in \mathcal{B}_2 \wedge F \rightsquigarrow_{\simeq} \xrightarrow{\ell} \mathcal{B}_2 \wedge (E', F') \in \mathcal{R}$。
- 如果 $E\mathcal{R}F$ 并且 $E \rightsquigarrow_{\simeq} \xrightarrow{q} \mathcal{B}_1$，$\mathcal{B}_1 \neq [E]_{\simeq}$，那么存在 E', F' 使得 $E' \in \mathcal{B}_1 \wedge F' \in \mathcal{B}_2 \wedge F \rightsquigarrow_{\simeq} \xrightarrow{q} \mathcal{B}_2 \wedge (E', F') \in \mathcal{R}$。

我们称 $\mathcal{R}$ 是一个*关于 $\simeq$ 的分支互模拟*（branching bisimulation up to $\simeq$）。

下述命题说明了，如果我们可以证明一对进程满足一个关于 $\simeq$ 的分支互模拟，那么这对进程就满足分支互模拟。

命题 6.7 (Milner [11]) 如果 $\mathcal{R}$ 是一个关于 $\simeq$ 的分支互模拟，那么

$$E\mathcal{R}F \Rightarrow E \simeq F$$

命题 6.8 (公理 $R2$ 的可靠性) 如果 $F \simeq S\{F/X\}$，并且在 S 中，X 是概率被守卫的，那么 $F \simeq \mu X.S$。

证明 考虑下述关系

$$\mathcal{R} = \left\{ \left(T\{F/Y\}, T\{\mu X.S/Y\}\right) \ : \ \mathrm{fv}(T) = \{Y\} \right\}$$

我们有 $(F, \mu X.S) \in \mathcal{R}$（可以取 $T = Y$）。根据命题6.7，我们只需要证明 $\mathcal{R}$ 关于对称的闭包是一个关于 $\simeq$ 的分支互模拟。

- 如果 $T\{F/Y\} \leadsto_{\simeq} \xrightarrow{\ell} \mathcal{B}_1$，$\ell \neq \tau \vee \mathcal{B}_1 \neq [T\{F/Y\}]_{\simeq}$。
  考虑该 ℓ-迁移中的 ϵ-树 $\mathcal{T}_{\simeq}^{T\{F/Y\}}$，因为 $F \simeq S\{F/X\}$，我们可以递归的将 $\mathcal{T}_{\simeq}^{T\{F/Y\}}$ 中从节点 F 开始的子树替换为 ϵ-树 $\mathcal{T}_{\simeq}^{S\{F/X\}}$，这样就构造了一棵 ϵ-树 $\mathcal{T}_{\simeq}^{T\{\mu X.S/Y\}}$。我们记 $T\{\mu X.S/Y\} \leadsto_{\simeq} \xrightarrow{\ell} \mathcal{B}_2$。接下来，我们需要证明 $\mathcal{B}_1$ 和 $\mathcal{B}_2$ 中各存在一个节点满足 $\mathcal{R}$ 关系。

  如果 $\mathcal{T}_{\simeq}^{T\{F/Y\}}$ 中存在某个叶子结点形如 $E' = T'\{F/Y\} \in \mathcal{B}_1$，即没有进入进程 F。那么 $\mathcal{T}_{\simeq}^{T\{\mu X.S/Y\}}$ 中存在一个叶子节点 $F' = T'\{\mu X.S/Y\} \in \mathcal{B}_2$ 且 $E'\mathcal{R}F'$。

  否则，$\mathcal{T}_{\simeq}^{T\{F/Y\}}$ 的所有分支都会进入 F。对应的 $\mathcal{T}_{\simeq}^{T\{\mu X.S/Y\}}$ 的所有分支都会进入 $\mu X.S$。固定 $\mathcal{T}_{\simeq}^{T\{F/Y\}}$ 中某个叶子结点 $E' \in \mathcal{B}_1$，E' 出现在从 F 开始增长的一棵 ϵ-树中。因为 $F \simeq S\{F/X\}$，并且 X 在 S 中是概率被守卫的，所以在一棵从 $S\{F/X\}$ 开始增长的 ϵ-树中，一定存在一个分支不会进入 F（或者该分支中本身就不包含 F）。因此存在一个可以从 S 到达的进程 E''，满足 $E''\{F/X\} \simeq E'$。那么 $\mathcal{T}_{\simeq}^{T\{\mu X.S/Y\}}$ 中存在叶子节点 $F' = E''\{\mu X.S/X\} \in \mathcal{B}_2$。所以我们有

  $$(E''\{F/X\}, F') = (E''\{Y/X\}\{F/Y\}, E''\{Y/X\}\{\mu X.S/Y\}) \in \mathcal{R}$$

- 如果 $T\{F/Y\} \leadsto_{\simeq} \xrightarrow{q} \mathcal{B}_1$ 使得 $\mathcal{B}_1 \neq [T\{F/Y\}]_{\simeq}$.
  和 ℓ-迁移的情况类似，不再赘述。 □

命题 6.9 (公理 $R4$ 的可靠性) $\mu X.(\tau.(\tau.S + \sum_{i\in I}\alpha_i.T_i) + \sum_{j\in J}\beta_j.R_j) \simeq \mu X(\tau.S + \sum_{i\in I}\alpha_i.T_i + \sum_{j\in J}\beta_j.R_j)$, 其中，$X$ 在 E 中不是概率被守卫的。

证明 定义

$$A_1 \overset{\text{def}}{=} \mu X.(\tau.(\tau.S + \sum_{i\in I}\alpha_i.T_i) + \sum_{j\in J}\beta_j.R_j)$$

$$A_2 \overset{\text{def}}{=} \tau.S\{A_1/X\} + \sum_{i\in I}\alpha_i.T_i\{A_1/X\}$$

$$B_1 \overset{\text{def}}{=} \mu X(\tau.S + \sum_{i\in I}\alpha_i.T_i + \sum_{j\in J}\beta_j.R_j)$$

首先，我们证明 $A_1 \simeq A_2$。对于 A_2 的任何动作，A_1 可以模拟，并且得到的两个进程是相等的。另一个方向，因为 S 中，X 不是概率被守卫的，可以证明路径 $A_2 \xrightarrow{\tau} S\{A_1/X\} \xrightarrow{\lambda_1} S_1\{A_1/X\} \xrightarrow{\lambda_2} \cdots \xrightarrow{\lambda_m} A_1$ 中的动作 $\lambda_i \in \{\tau\} \cup \{p\tau \mid 0 < p < 1\}$ 都不改变状态。我们构建一棵 ϵ-树 $\mathcal{T}_{\simeq}^{A_2}$，它的每一个分支都会进入 A_1。因此对于所有的 $\mathcal{T}_{\simeq}^{A_1}$，都存在一棵有相同叶子结点集合的 ϵ-树 $\mathcal{T}_{\simeq}^{A_2}$。

接下来，我们证明 $A_1 \simeq B_1$。因为 $A_1 \xrightarrow{\tau} A_2$ 中的内部动作不改变状态，A_1 可以模拟 B_1 的动作，并且得到的两个进程是相等的。另一个方向，给定一棵 ϵ-树 $\mathcal{T}_{\simeq}^{A_1}$，我们可以构造一棵 ϵ-树 $\mathcal{T}_{\simeq}^{B_1}$。如果 $\mathcal{T}_{\simeq}^{A_1}$ 中有一条边 $A_1 \xrightarrow{\tau} A_2$，那么 $\mathcal{T}_{\simeq}^{B_1}$ 中什么都不做；在其他情况下，$\mathcal{T}_{\simeq}^{B_1}$ 会模仿 $\mathcal{T}_{\simeq}^{A_1}$ 的行为。可以看到 $\mathcal{T}_{\simeq}^{A_1}$ 和 $\mathcal{T}_{\simeq}^{B_1}$ 的叶子结点集合是相同的。

□

最后我们证明公理 $R5$ 的可靠性。首先，下面的引理可以说明，如果 $\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right)$ 中的变量 X 在任意 S_i 中都不是概率被守卫的，那么迁移 $\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right) \xrightarrow{p_i\tau} S_i\left\{\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right) / X\right\}$ 中的概率 τ 动作都是不改变状态的。

引理 6.10 令 $A = \mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right)$, $B_i = S_i\left\{\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right) / X\right\}$，如果 X 在 A 中不是概率被守卫的，那么对于任意 $i \in I$，$A \simeq B_i$。

证明 考虑下述关系：

$$\mathcal{R} = \left\{\left(A, B_i\right) : i \in I\right\} \cup \simeq$$

我们可以证明 $\mathcal{R}$ 是一个分支互模拟关系。

- 给定一棵正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$。因为 X 在 A 中不是概率被守卫的，那么首先我们一定能够找到一棵从 B_i 出发的正则的 ϵ-树，其叶子结点全部为 A，之后

我们将所有叶子结点 A 替换为 $\mathcal{T}_{\mathcal{R}}^{A}$，那么得到的正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{B_i}$ 中的叶子结点集合与 $\mathcal{T}_{\mathcal{R}}^{A}$ 的叶子结点集合一样，因此 B_i 可以模拟 A 的所有 ℓ-迁移和 q-迁移。

- 给定一棵正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{B_i}$。那么从结点 A 出发我们可以构造如下的 ϵ-树：如果遇到节点 B_i，我们将其替换为 $\mathcal{T}_{\mathcal{R}}^{B_i}$；如果遇到节点 $B_j, j \neq i$，那么因为 X 在 B_j 中不是概率被守卫的，我们一定能够找到一棵从 B_i 出发的正则的 ϵ-树，其叶子结点全部为 A。我们继续从结点 A 开始不断重复该操作，即可构造出正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$，使得其叶子结点集合和 $\mathcal{T}_{\mathcal{R}}^{B_i}$ 的叶子结点集合一样。因此 A 可以模拟 B_i 的所有 ℓ-迁移和 q-迁移。

综合以上两点，可以说明 $\mathcal{R}$ 是一个分支互模拟关系。因此 $A \simeq B_i$。 □

有了引理6.10，我们可以证明公理 $R5$ 的可靠性。

命题 6.11 (公理 $R5$ 的可靠性) 如果对于 $i \in I$，X 在 S_i 中不是概率被守卫的，那么 $\mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right) \simeq \mu X.\left(\sum_{i\in I} \tau_i.S_i\right)$。

证明 令进程 $A = \mu X.\left(\bigoplus_{i\in I} p_i\tau.S_i\right)$，进程 $B = \mu X.\left(\sum_{i\in I} \tau_i.S_i\right)$。考虑下述关系：

$$\mathcal{R} = \{(A, B)\ \} \cup \simeq$$

我们可以证明 $\mathcal{R}$ 是一个分支互模拟关系。

- 给定一棵正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$。
  - 如果 $\mathcal{T}_{\mathcal{R}}^{A}$ 的叶子节点中包括 A，那么根据引理6.10，该 ϵ-树上不存在 ℓ-迁移或者 q-迁移。
  - 否则的话，可以从进程 B 构建一棵 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{B}$，其叶子结点可以做的动作集合是 $\mathcal{T}_{\mathcal{R}}^{A}$ 的叶子节点的动作集合的子集。因此 B 可以模拟 A 的所有 ℓ-迁移和 q-迁移。
- 给定一棵正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{B}$。
  - 如果 $\mathcal{T}_{\mathcal{R}}^{B}$ 的叶子节点中包括 B，那么根据引理6.10，该 ϵ-树上不存在 ℓ-迁移或者 q-迁移。
  - 否则的话，假设 $\mathcal{T}_{\mathcal{R}}^{B}$ 中从根节点的第一步出发的进程是 S_i。我们可以从结点 A 出发可以构造如下的 ϵ-树：如果遇到节点 S_i，我们按照 $\mathcal{T}_{\mathcal{R}}^{B}$ 中的方法构造；如果遇到节点 $S_j, j \neq i$，那么因为 X 在 S_j 中不是概率被守卫的，我们一定能够找到一棵从 S_i 出发的正则的 ϵ-树，其叶子结点全部为 A。我们继续从结点 A 开始不断重复该操作，即可构造出正则的 ϵ-树 $\mathcal{T}_{\mathcal{R}}^{A}$，使得其叶子结点可以做的动作集合和 $\mathcal{T}_{\mathcal{R}}^{B}$ 的叶子结点可以做的动作集合一样。因此 A 可以模拟 B 的所有 ℓ-迁移和 q-迁移。

综合以上两点，可以说明 $\mathcal{R}$ 是一个分支互模拟关系。因此 $A \simeq B$。 □

推论 6.12 ($\mathcal{A}$ 的可靠性) 对于 $E,F \in \mathcal{P}_{fs}$，如果 $\mathcal{A} \vdash E = F$，那么 $E \simeq F$。

6.2.4 公理系统完备性

对于公理系统的完备性证明。我们会采用 Milner 的经典的证明完备性的思路[100]。首先我们会证明如果两个进程都是概率被守卫的，并且它们互模拟，那么它们可以被证明满足同一个递归规范。而根据 Milner 的唯一解命题，任意两个满足同一个递归规范的进程都可以被公理系统证明相等。最后，对于任意一个进程，它都可以被证明和一个概率被守卫的进程相等，从而我们可以证明公理系统对所有进程的完备性。

通过对 $\mathcal{S}_{fs}$ 中的项结构归纳，我们可以证明如下引理：

引理 6.13 对于一个项 $T \in \mathcal{S}_{fs}$,

- 如果 $T \xrightarrow{X} 0$，那么 $\vdash T = X$；
- 如果 $T \in \mathcal{S}_n$，那么 $\vdash T = \sum_{i\in I}\{\alpha_i.T_i | T \xrightarrow{\alpha_i} T_i\}$；
- 如果 $T \in \mathcal{S}_r$，那么 $\vdash T = \bigoplus_{i\in I}\{p_i\tau.T_i | T \xrightarrow{p_i\tau} T_i\}$。

定义 6.6 (递归规范) 一个*递归规范*（recursive specification）$\mathbb{S}$ 是一个公式的集合：

$$\{X = S_X | X \in V_{\mathbb{S}}\}$$

其中，$V_{\mathbb{S}}$ 是一个变量集合。

给定一个进程 E 和一个变量 X_0，如果对于每一个变量 $X \in V_{\mathbb{S}}$，存在进程 E_X，其中，$E = E_{X_0}$，使得对于 $X \in V_{\mathbb{S}}$，满足

$$\mathcal{A} \vdash E_X = S_X\{E_Y/Y\}_{Y\in V_{\mathbb{S}}}$$

进程 E 作为变量 $X_0 \in V_{\mathbb{S}}$ 的解，*可在 $\mathcal{A}$ 中被证明满足递归规范 $\mathbb{S}$*（$\mathcal{A}$-provably satisfies the recursive specification $\mathbb{S}$）

给定一个递归规范 $\mathbb{S}$，两个变量 $X,Y \in V_{\mathbb{S}}$，如果 Y 在 E_X 中是自由的并且不是概率被守卫的，我们记作 $X >_u Y$。如果关系 $>_u$ 是 $V_{\mathbb{S}}$ 上的一个良基关系（well-founded relation），那么我们称 $\mathbb{S}$ 是概率被守卫的。

Milner 按照 $\mathbb{S}$ 中的公式的数量归纳，证明了唯一解命题[100]。这个命题在我们的模型中依然成立。

命题 6.14 如果 $\mathbb{S}$ 是一个有限的概率被守卫的递归规范，$X_0 \in V_{\mathbb{S}}$, 那么一定存在进程 E 作为 X_0 的解，可在 $R1$ 中被证明满足递归规范 $\mathbb{S}$。如果有两个满足条件的进程 E 和 F，那么 $R2 \vdash E = F$。

引理 6.15 给定 $G \in \mathcal{P}^g_{fs}$，如果对于所有的 $i \in I$，$G \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$，那么 $\mathcal{A} \vdash G = \bigoplus_{i\in I} q_i\tau.E_i$。

证明 定义一个自然数的二元组上的字典序：$(m,n) < (m',n')$ 表示 (1) $m < m'$ 或者 (2) $m = m' \wedge n < n'$。我们定义 $(m_1,n_1) + (m_2,n_2) = (m_1+m_2, n_1+n_2)$。定义 $(m,n)_1 = m, (m,n)_2 = n$。

递归的定义一个秩函数 $r : \mathcal{P}^g_{fs} \to \mathbb{N}\times\mathbb{N}$:

$$
\begin{aligned}
r(\mathbf{0}) &= (0,0)\\
r(\bigoplus_{i\in I} p_i\tau.E_i) &= (0,1) + \max_{i\in I}\{r(E_i)\}\\
r(\sum_{i\in I} \alpha_i.E_i) &= \max\left\{\left\{(0,1) + r(E_i) | \alpha_i = \tau\right\} \cup \left\{(0,1) | \alpha_i \neq \tau\right\}\right\}\\
r(\mu X.T) &= (1 + r(T\{\mathbf{0}/X\})_1, 0)
\end{aligned}
$$

我们会根据 $r(G)$ 来归纳证明该引理:

- 如果 $r(G) = (0,1)$，并且对于 $i \in I$，$G \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$。
  那么 G 一定可以写成 $\bigoplus_{i\in I}\left\{\bigoplus_{j\in J_i} p_j\tau.E_i | \sum_{j\in J_i} p_j = q_i\right\}$。那么 $A1 \vdash G = \bigoplus_{i\in I} q_i\tau.E_i$.
- 如果 $r(G) = (m,n) > (0,1)$，并且对于 $i \in I$，$G \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$。
  - G 的形式为 $\sum_{j\in J} \alpha_j.G_j$。
    那么对于任意 $j \in J$，我们一定有 $\alpha_j = \tau$ 并且 $G \simeq G_j \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$，$r(G_j) < r(G)$。根据归纳假设，对于任意的 $j \in J$，$\mathcal{A} \vdash G_j = \bigoplus_{i\in I} q_i\tau.E_i$。我们可以得到结论：$\mathcal{A} \vdash G = \bigoplus_{i\in I} q_i\tau.E_i$。
  - G 的形式为 $\bigoplus_{j\in J} p_j\tau.G_j$。
    第一种情况，如果存在 $j \in J$，使得 $G_j \not\simeq G$。根据引理6.13，

$$
\begin{aligned}
\mathcal{A} \vdash G = &\bigoplus_{i\in I}\left\{p_i\tau.E_i | \frac{p_i}{q_i} = c < 1, G \xrightarrow{p_i\tau} E_i \not\simeq G\right\} \oplus\\
&\bigoplus_{j\in J-I}\left\{p_j\tau.G_j : G \xrightarrow{p_j\tau} G_j \simeq G\right\}
\end{aligned}
$$

对于任意一个 $j \in J - I$，$r(G_j) < r(G)$。根据归纳假设，$\mathcal{A} \vdash G_j = \bigoplus_{i\in I} q_i\tau.E_i$，因此 $\mathcal{A} \vdash G = \bigoplus_{i\in I} q_i\tau.E_i$。

第二种情况，如果对于任意 $j \in J$, $G_j \simeq G$。那么对于所有的 $j \in J$，$G_j \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$ 并且 $r(G_j) < r(G)$。根据归纳假设，$\mathcal{A} \vdash G_j = \bigoplus_{i\in I} q_i\tau.E_i$。因此 $\mathcal{A} \vdash G = \bigoplus_{i\in I} q_i\tau.E_i$。

– G 的形式为 $\mu X.T$。

应用语义规则我们有

$$T\{\mu X.T/X\} \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i \tag{6–1}$$

对于进程 $T\{\bigoplus_{i\in I} q_i\tau.E_i/X\}$，它可以构建和进程 $T\{\mu X.T/X\}$ 一样的 ϵ-树。因此我们也有

$$T\{\bigoplus_{i\in I} q_i\tau.E_i/X\} \rightsquigarrow_{\simeq} \xrightarrow{q_i} E_i$$

根据秩函数的定义，$r(\mu X.T)_1 = 1 + r\left(T\left\{\bigoplus_{i\in I} q_i\tau.E_i/X\right\}\right)_1$，因此 $r(\mu X.T) > r(T\{\bigoplus_{i\in I} q_i\tau.E_i/X\})$。

根据归纳假设，$\mathcal{A} \vdash T\{\bigoplus_{i\in I} q_i\tau.E_i/X\} = \bigoplus_{i\in I} q_i\tau.E_i$，应用公理 $R2$，$\mathcal{A} \vdash \mu X.T = \bigoplus_{i\in I} q_i\tau.E_i$。

综合以上情况，我们完成了该引理证明。 □

命题 6.16 给定 $E_0, F_0 \in \mathcal{P}^g_{fs}$，如果 $E_0 \simeq F_0$，那么存在一个有限的概率被守卫的递归规范 $\mathbb{S}$，$X_0 \in V_{\mathbb{S}}$，使得 E_0 和 F_0 都可以作为 X_0 的解，可在 $\mathcal{A}$ 中被证明满足递归规范 $\mathbb{S}$。

证明 我们定义变量

$$V_{\mathbb{S}} = \left\{ X_{EF} | E \in \mathcal{P}_{E_0}, F \in \mathcal{P}_{F_0}, E \simeq F \right\}$$

其中，$X_0 = X_{E_0F_0}$。对于每一个变量 $X_{EF} \in V_{\mathbb{S}}$，$\mathbb{S}$ 中会按照如下规则加入公式：

1. 如果 $E \in \mathcal{S}_r$，并且对于任意的 E' 使得 $E \xrightarrow{p\tau} E'$，都有 $E' \simeq E$，那么

$$X_{EF} = \bigoplus \left\{ p\tau.X_{E'F} | E \xrightarrow{p\tau} E' \right\}$$

2. 如果条件1 不满足，$F \in \mathcal{S}_r$，并且对于任意的 F' 使得 $F \xrightarrow{p\tau} F'$，都有 $F' \simeq F$，那么

$$X_{EF} = \bigoplus \left\{ p\tau.X_{EF'} | F \xrightarrow{p\tau} F' \right\}$$

3. 如果条件1和2都不满足，并且 $E \in \mathcal{S}_r, F \in \mathcal{S}_r$，我们用下标集合 I 表示所有满足 $E \rightsquigarrow_{\simeq} \xrightarrow{q} \mathcal{B}_i$，$F \rightsquigarrow_{\simeq} \xrightarrow{q} \mathcal{B}_i$，$q > 0$ 的等价类 $\mathcal{B}_i$。我们有：

$$X_{EF} = \bigoplus_{i \in I} \left\{ q\tau.X_{E_iF_i} \right\}$$

4. 如果 $E \in \mathcal{S}_n, F \in \mathcal{S}_n$ 那么

$$X_{EF} = \sum \left\{ \alpha.X_{E'F'} | E \xrightarrow{\alpha} E', F \xrightarrow{\alpha} F', E' \simeq F' \right\} + \sum \left\{ \tau.X_{E'F} | E \xrightarrow{\tau} E', E' \simeq F \right\} + \sum \left\{ \tau.X_{EF'} | F \xrightarrow{\tau} F', E \simeq F' \right\}$$

5. 以上条件都不满足的话，

$$X_{EF} = \sum \left\{ \tau.X_{E'F} | E \xrightarrow{\tau} E', E' \simeq F \right\} + \sum \left\{ \tau.X_{EF'} | F \xrightarrow{\tau} F', E \simeq F' \right\}$$

接下来我们要说明下面的性质（记作 $\star\star$）：

如果变量 X_{EF} 被赋值为进程 E，我们可以证明：

$$\mathcal{A} \vdash E = S_{X_{EF}}\{E'/X_{E'F'}\}_{X_{E'F'} \in V_S}$$

性质 $\star\star$ 说明 E_0 可以作为 X_0 的解，可在 $\mathcal{A}$ 中被证明满足递归规范 $\mathbb{S}$。同理，F_0 也可以作为 X_0 的解，有相同的性质。因此命题得证。

接下来我们只需要证明性质 $\star\star$。情况1, 2, 5都可以直接由引理6.13证明；情况3可以由引理6.15得到；

在情况4中，$E, F \in \mathcal{S}_n$。我们需要证明：

$$\mathcal{A} \vdash E = \sum \left\{ \alpha.E' | E \xrightarrow{\alpha} E', F \xrightarrow{\alpha} F', E' \simeq F' \right\} + \sum \left\{ \tau.E' | E \xrightarrow{\tau} E', E' \simeq F \right\} + \sum \left\{ \tau.E | F \xrightarrow{\tau} F', E \simeq F' \right\} \tag{6–2}$$

根据引理6.13，$\vdash E = \sum_{i \in I}\{\alpha_i.E_i : E \xrightarrow{\alpha_i} E_i\}$，那么

$$B1 \vdash E = \sum \{\alpha.E' | E \xrightarrow{\alpha} E', F \xrightarrow{\alpha} F', E' \simeq F'\} + \sum \{\tau.E' | E \xrightarrow{\tau} E', E' \simeq F\} + \tau.E \tag{6–3}$$

如果存在进程 F' 使得 $F \xrightarrow{\tau} F' \simeq E$，(6–2) 和 (6–3) 直接可以相等。如果不存在这样的进程 F'，那么 E 的每一个动作需要被 F 直接模拟，也就意味着集合 $\left\{ \alpha_i.E_i : E \xrightarrow{\alpha_i} E_i \right\}$ 和集合 $\left\{ \alpha.E' | E \xrightarrow{\alpha} E', F \xrightarrow{\alpha} F', E' \simeq F' \right\} \cup \left\{ \tau.E' | E \xrightarrow{\tau} E', E' \simeq F \right\}$ 相等。证明完成。

□

推论 6.17 ($\mathcal{A}$ 对于概率被守卫的进程的完备性) 给的 $E, F \in \mathcal{P}_{fs}^{g}$，如果 $E \simeq F$ 那么 $\mathcal{A} \vdash E = F$。

最后我们说明 $\mathcal{A}$ 对于任意进程的完备性。

命题 6.18 对于进程 $E \in \mathcal{P}_{fs}$，存在一个概率被守卫的进程 E'，使得 $\mathcal{A} \vdash E = E'$。

证明 该证明与文章 [99] 的做法类似，都是对进程 E 中的 $\mu X.T$ 的嵌套层数归纳。我们只需要证明 $E = \mu X.F$ 的情况。假定对于所有的不动点算子嵌套层数少于 E 的进程 μYG，都存在一个概率被守卫的进程 $\mu YG'$，使得 $\mathcal{A} \vdash \mu YG = \mu YG'$。我们可以先把所有 F 中出现的最靠近语法树根节点的不动点项 $\mu Y.G$ 替换为 $G'\{\mu Y.G' / Y\}$。经过替换后 F 记作 F_1。我们最后要做的就是去掉 F_1 中的所有不是概率被守卫的变量 X。

如果一个变量不是概率被守卫的，我们可以先通过公理 $R5$ 把概率 τ 动作转化为非确定的 τ 动作。而对于非确定的 τ 动作，我们可以不断应用公理 $R4$ 逐渐减少不是概率被守卫的变量 X 前的 τ 动作的个数，直到最后应用公理 $R3$ 可以去掉最后一个 τ 动作。因此命题得证。 □

推论 6.19 ($\mathcal{A}$ 对于所有进程的完备性) 给定 E,$F \in \mathcal{P}_{fs}$，如果 $E \simeq F$，那么 $\mathcal{A} \vdash E = F$。

6.3 本章小结

在本章中，我们研究了随机通信系统演算上的分支互模拟等价。本章的工作分为两个方面。其一是我们给出了一个 $\mathcal{O}(n^4)$ 时间复杂度的等价性判定算法。其二是我们给出了一个有限公理系统，并证明了其可靠性和完备性。这两方面的工作都可以推广到用 Fu [22] 的模型无关的方法定义的其他概率进程演算模型中。

以下是本研究中的一些尚未解决的问题：

- 首先，如果等价关系中考虑了发散性（divergence）[113]，需要提出一些新的公理来完善我们的公理系统；
- 随机通信系统演算的分支互模拟的逻辑刻画（logical characterization）问题的研究 [114]；
- Fu 为了定义分支互模拟提出的 ϵ-树的技术是否可以到外部动作有概率的模型中。

以上问题都值得我们继续攻克。

第七章　全文总结及展望

7.1　全文总结

本文的主要研究内容是互模拟等价的验证问题。本文的主要工作大致可以分为两个部分：

1. 下推自动机上的强互模拟等价问题。

2. 随机通信系统演算上的分支互模拟等价问题。

对于 PDA 的强互模拟等价问题，我们证明了一个 ACKERMANN-难的下界（定理3.1）。这个结论于 Jančar 和 Schmitz 给出的 ACKERMANN 的算法一起，说明了该问题是一个 ACKERMANN-完备问题。我们在表格7–1[1]总结 PDA 以及部分相关模型互模拟问题目前的研究现状。

表 7–1 下推自动机以及相关模型上的互模拟等价研究现状总结

	确定下推自动机	下推自动机	一阶文法
上界	TOWER [38, 39]	ACKERMANN [71]	ACKERMANN [71]
下界	P-难 [59]	ACKERMANN-难	ACKERMANN-难 [39]

从最近对 PDA 的强互模拟问题的研究结果中，可以看到控制状态的数量是决定该问题复杂性的一个重要参数 [71]。本文研究了固定状态数 PDA 的强互模拟问题。给出了三个计算复杂性的结论：

- 当（赋范）PDA 的控制状态数量 $d \geq 4$ 时，有$\mathbf{F}_{d-1}$-难的下界（定理3.1）；
- 当赋范 PDA 的控制状态数量 $d = 2$ 时，有 EXPTIME-难的下界（定理4.1）；
- 当赋范 PDA 的控制状态数量为 d 时，有$\mathbf{F}_{d+3}$ 的上界（定理5.1）。

我们在表格7–2中总结了固定状态数 PDA 的强互模拟问题的研究现状。

随机通信系统演算是 Fu[22] 提出的一个概率进程演算模型。我们研究了这个模型上的分支互模拟问题。首先，我们设计了一个时间复杂度为 $\mathcal{O}(n^4)$ 的判定算法；另一方面，我们给出了一个公理系统并证明了其可靠性和完备性。

我们在随机通信系统演算的算法部分的工作中，提出了一个和 ϵ-树等效的概念：ϵ-图。从而保证了算法的终止性。在公理化部分的工作中，由于传统定义的被守卫的与随机通信系统演算无法很好的契合，我们提出了一个新的概念称作概率

[1]在本章的表格中，未标注引用的结论即为本文中的结论或本文结论的推论。

表 7–2 状态数为 d 的下推自动机互模拟参数复杂性总结

	$d=1$	$d=2$	$d=3$	$d\geq 4$
赋范的	P [37] P-难 [59]	$\mathbf{F}_5$ EXPTIME-难	$\mathbf{F}_6$ EXPTIME-难	$\mathbf{F}_{d+3}$ $\mathbf{F}_{d-1}$-难
非赋范的	2EXPTIME [52] EXPTIME-难 [54]	$\mathbf{F}_6$ [71] EXPTIME-难 [54]	$\mathbf{F}_7$ [71] EXPTIME-难 [54]	$\mathbf{F}_{d+4}$ [71] $\mathbf{F}_{d-1}$-难

被守卫的，从而很好的继承了传统的被守卫的在公理化过程中扮演的角色。此为本文在技术方面的贡献。

7.2 工作展望

这里总结一些本文暂时还没有解决，并且值得考虑的问题：

1. 首先是 PDA 互模拟问题的参数复杂性。在表格7–2中可以看到，目前绝大部分都还不是完备结果，值得进一步探索。
2. 我们目前考虑的随机通信系统演算的分支互模拟等价没有考虑发散性。考虑了发散性以后的分支互模拟等价的算法和公理化是值得继续研究的。此外，还有分支互模拟的逻辑刻画，以及 ϵ-树技术向外部动作有概率的模型中的推广，都是我们未来可以考虑的问题。

此外，领域中还有一些很有挑战性的公开问题：

1. DPDA 的语言等价问题的计算复杂性。如表格7–1中所示，目前最好的上界是 TOWER，而下界只是 P-难。
2. BPA 的强互模拟问题的计算复杂性。如表格7–2中所示，目前最好的上界是 2EXPTIME，而下界是 EXPTIME-难。

附录 A　ϵ-树举例

这里介绍一些例子来帮助读者理解 ϵ-树，ℓ-迁移和 q-迁移的定义。

例 A.1　这个例子来自于文章 [22] 的例 7。令进程 $P = \mu X(\frac{1}{2}\tau.a \oplus \frac{1}{2}\tau.X)$，假设等价关系 $\mathcal{R}$ 满足 $P\mathcal{R}a$。图A–1列举了三棵 P 关于 $\mathcal{R}$ 的 ϵ-树。

首先最左边的单个节点 P 也可以作为一个 ϵ-树，因为进程 P 不能做非确定动作，所以这棵树上没有对应的 ℓ-迁移；此外，从 P 出发的所有概率动作的结果都还在等价类 $[P]_{\mathcal{R}}$ 中，因此也不满足 q-迁移中 $q > 0$ 的条件。因此从这棵树上我们找不到对应的 ℓ-迁移或 q-迁移。

在中间的 ϵ-树中，叶子结点包括 a 和 P，进程 a 只能做非确定动作 a，进程 P 只能做概率动作 $\frac{1}{2}\tau$。按照定义，从这棵树上我们找不到对应的 ℓ-迁移或 q-迁移。

在右边的 ϵ-树中，叶子结点只有 a，所有的叶子结点都可以做 a 动作到进程 $\mathbf{0}$。按照定义，这棵树上有 a-迁移。我们可以记作 $P \rightsquigarrow_{\mathcal{R}} \xrightarrow{a} [\mathbf{0}]_{\mathcal{R}}$。

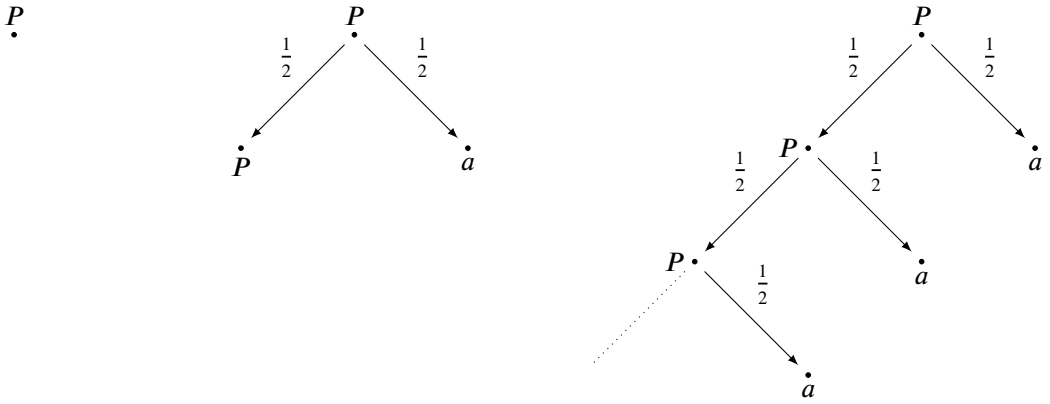


图 A–1　ϵ-树的例子A.1

通过例子A.1我们可以看到，

- 给定了进程 P 和关系 $\mathcal{R}$，可能存在许多 ϵ-树。这些 ϵ-树可以是有限的，也可以是无限的。进程单个节点也可以作为一棵 ϵ-树。
- 在 ℓ-迁移和 q-迁移的定义中，我们只要求存在一棵满足条件的 ϵ-树。在 ℓ-迁移中，我们会要求这棵 ϵ-树的所有节点都可以做相同的动作到达同一个等价类中。

例 A.2 令进程

$$Q = \frac{1}{2}\tau.\left(\mu X(\frac{1}{2}\tau.X \oplus \frac{1}{2}\tau.X)\right) \oplus \frac{1}{2}\tau.a$$

我们定义进程 $Q_1 = \mu X(\frac{1}{2}\tau.X \oplus \frac{1}{2}\tau.X)$。假设等价关系 $\mathcal{R}$ 满足 $Q\mathcal{R}Q_1\mathcal{R}a$。需要说明这里的关系 $\mathcal{R} \nsubseteq \simeq$。那么图A–2中是一棵 Q 关于 $\mathcal{R}$ 的 ϵ-树。记作 $\mathcal{T}_{\mathcal{R}}^{Q}$。根据(6–1)中的定义，$P^f(\mathcal{T}_{\mathcal{R}}^{Q}) = \frac{1}{2}$。因此该 ϵ-树不是正则的，从这棵树上我们找不到对应的 ℓ-迁移或 q-迁移。

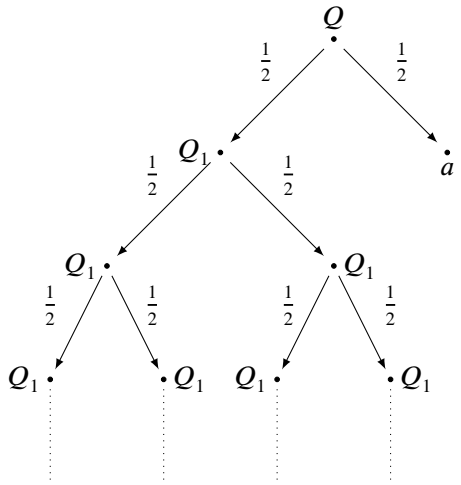


图 A–2 ϵ-树的例子A.2

通过例子A.2我们可以看到，在定义 ℓ-迁移和 q-迁移时，ϵ-树的正则性是很必要的。显然，此例子中的进程 Q 不应该和例子A.1的进程 P 互模拟。

例 A.3 令进程

$$G = \mu X\left(\frac{1}{2}\tau.X \oplus \frac{1}{2}\tau.\left(\frac{1}{4}\tau.X \oplus \frac{1}{4}\tau.S_1 \oplus \frac{1}{4}\tau.S_2 \oplus \frac{1}{4}\tau.T\right)\right)$$

我们定义进程 $G_1 = \frac{1}{4}\tau.G \oplus \frac{1}{4}\tau.S_1 \oplus \frac{1}{4}\tau.S_2 \oplus \frac{1}{4}\tau.T$。如果等价关系 $\mathcal{R}$ 满足 $G\mathcal{R}G_1$，$S_1\mathcal{R}S_2$。图A–3中是一棵 G 关于 $\mathcal{R}$ 的 ϵ-树。其叶子结点只有 G_1。根据定义：

$$P_{\mathcal{R}}\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [S_1]_{\mathcal{R}}\right) = \frac{P\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [S_1]_{\mathcal{R}}\right)}{1 - P\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [G_1]_{\mathcal{R}}\right)} = \frac{2}{3}$$

$$P_{\mathcal{R}}\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [T]_{\mathcal{R}}\right) = \frac{P\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [T]_{\mathcal{R}}\right)}{1 - P\left(G_1 \xrightarrow{\coprod_{i\in[1,4]} p_i\tau} [G_1]_{\mathcal{R}}\right)} = \frac{1}{3}$$

这棵树有两个对应的 q-迁移：记作 $G_1 \rightsquigarrow_{\mathcal{R}} \xrightarrow{\frac{2}{3}} [S_1]_{\mathcal{R}}$，$G_1 \rightsquigarrow_{\mathcal{R}} \xrightarrow{\frac{1}{3}} [T]_{\mathcal{R}}$。

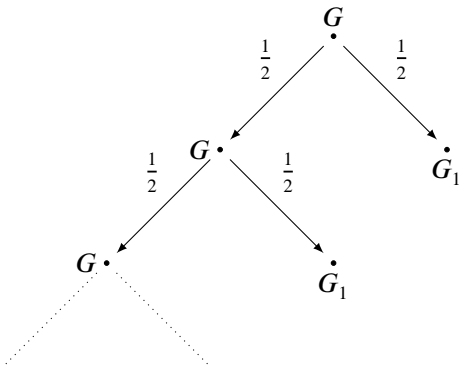


图 A–3 ϵ-树的例子A.3

通过例子A.3我们可以看到，在定义 q-迁移时考虑的是到一个等价类的条件概率。在这个定义下，我们可以该例子A.3中的进程 G 和进程 $\frac{2}{3}\tau.S_1 \oplus \frac{1}{3}\tau.T$ 互模拟。

例 A.4 这个例子来自于文章 [22] 的例 9。令进程

$$H = \mu X\left(\frac{1}{2}\tau.(a+\tau.X) \oplus \frac{1}{2}\tau.(b+\tau.X)\right)$$

假设等价关系 $\mathcal{R}$ 满足 $H\ \mathcal{R}\ (a+\tau.H)\ \mathcal{R}\ (b+\tau.H)$。图A–4中列举了两棵 P 关于 $\mathcal{R}$ 的 ϵ-树。

在左边的 ϵ-树中，叶子结点都是 $b+\tau.H$，所有的叶子结点都可以做 b 动作到进程 $\mathbf{0}$。按照定义，这棵树上有 b-迁移。我们可以记作 $H \rightsquigarrow_{\mathcal{R}} \xrightarrow{b} [\mathbf{0}]_{\mathcal{R}}$。

在右边的 ϵ-树中，叶子结点都是 $a+\tau.H$，所有的叶子结点都可以做 a 动作到进程 $\mathbf{0}$。按照定义，这棵树上有 a-迁移。我们可以记作 $H \rightsquigarrow_{\mathcal{R}} \xrightarrow{a} [\mathbf{0}]_{\mathcal{R}}$。

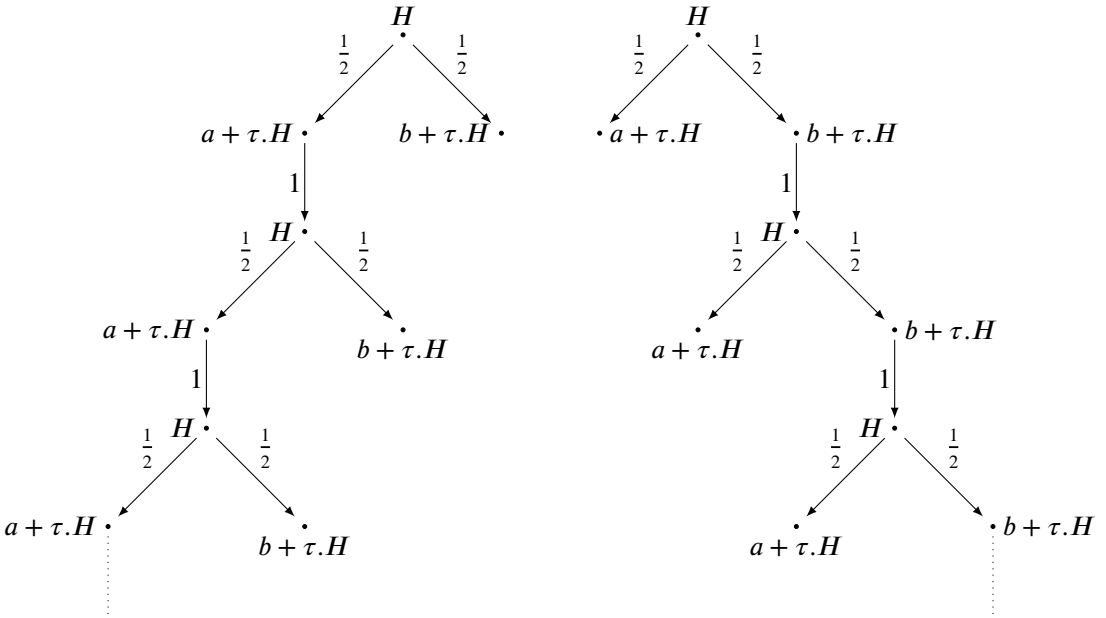


图 A–4 ϵ-树的例子A.4

在前面的例子中，所有的 ϵ-树的边的标记都在 $(0,1)$ 中。所以我们补充了例子A.4。它说明当 ϵ-树的边标记为 1 时，其对应的就是一个非确定的内部动作。

参考文献

[1] EMERSON E A, CLARKE E M. Characterizing correctness properties of parallel programs using fixpoints[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 1980: 169-181.

[2] CLARKE E M, EMERSON E A. Design and synthesis of synchronization skeletons using branching time temporal logic[C]//Workshop on Logic of Programs. [S.l. : s.n.], 1981: 52-71.

[3] CLARKE E M, EMERSON E A, SISTLA A P. Automatic verification of finite-state concurrent systems using temporal logic specifications[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1986, 8(2): 244-263.

[4] KUČERA A, JANČAR P. Equivalence-Checking on Infinite-State Systems: Techniques and Results[J]. Theory and Practice of Logic Programming, 2006, 6(201).

[5] BRYANT R E. Graph-based algorithms for boolean function manipulation[J]. Computers, IEEE Transactions on, 1986, 100(8): 677-691.

[6] BURCH J R, CLARKE E M, MCMILLAN K L, et al. Symbolic model checking: 1020 states and beyond[J]. Information and computation, 1992, 98(2): 142-170.

[7] PELED D. All from one, one for all: on model checking using representatives[C]//International Conference on Computer Aided Verification. [S.l. : s.n.], 1993: 409-423.

[8] GODEFROID P, van LEEUWEN J, HARTMANIS J, et al. Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem[M]. [S.l.]: Springer Heidelberg, 1996.

[9] CLARKE JR E M, GRUMBERG O, KROENING D, et al. Model checking[M]. [S.l.]: MIT press, 2018.

[10] PARK D. Concurrency and automata on infinite sequences[G]//Theoretical Computer Science. [S.l.]: Springer, 1981: 167-183.

[11] MILNER R. Communication and concurrency[M]. [S.l.]: Prentice hall New York etc., 1989.

[12] VAN GLABBEEK R J, WEIJLAND W P. Branching time and abstraction in bisimulation semantics[J]. Journal of the ACM (JACM), 1996, 43(3): 555-600.

[13] GLABBEEK R V. The linear time-branching time spectrum[G]//CONCUR'90 Theories of Concurrency: Unification and Extension. [S.l. : s.n.], 1990.

[14] PETERSON J L. Petri nets[J]. ACM Computing Surveys (CSUR), 1977, 9(3): 223-252.

[15] MAYR R. Process rewrite systems[J]. Information and Computation, 2000, 156(1-2): 264-286.

[16] SRBA J. Roadmap of infinite results[G]//Current Trends in Theoretical Computer Science: The Challenge of the New Century Vol 1: Algorithms and Complexity Vol 2: Formal Models and Semantics. [S.l.]: World Scientific, 2004: 337-350.

[17] GIACALONE A, JOU C C, SMOLKA S A. Algebraic reasoning for probabilistic concurrent systems[C]//Proc. IFIP TC2 Working Conference on Programming Concepts and Methods. [S.l. : s.n.], 1990.

[18] HANSSON H, JONSSON B. A calculus for communicating systems with time and probabilities[C]//[1990] Proceedings 11th Real-Time Systems Symposium. [S.l. : s.n.], 1990: 278-287.

[19] LOWE G. Probabilities and priorities in timed CSP[J]., 1993.

[20] ANDOVA S. Process algebra with probabilistic choice[C]//International AMAST Workshop on Aspects of Real-Time Systems and Concurrent and Distributed Software. [S.l. : s.n.], 1999: 111-129.

[21] HERESCU O M, PALAMIDESSI C. Probabilistic asynchronous π-calculus[C]//International Conference on Foundations of Software Science and Computation Structures. [S.l. : s.n.], 2000: 146-160.

[22] FU Y. A Uniform Approach to Random Process Model[J/OL]., 2019. https://arxiv.org/pdf/1906.09541.pdf.

[23] SCHMITZ S. Complexity hierarchies beyond elementary[J]. ACM Transactions on Computation Theory (TOCT), 2016, 8(1): 3.

[24] MOORE E F. Gedanken-experiments on sequential machines[J]. Automata studies, 1956, 34: 129-153.

[25] BAR-HILLEL Y, PERLES M, SHAMIR E. On formal properties of simple phrase structure grammars[J]. Sprachtypologie und Universalienforschung, 1961, 14: 143-172.

[26] CHOMSKY N. On certain formal properties of grammars[J]. Information and control, 1959, 2(2): 137-167.

[27] GINSBURG S, GREIBACH S. Deterministic context free languages[J]. Information and Control, 1966, 9(6): 620-648.

[28] KORENJAK A J, HOPCROFT J E. Simple deterministic languages[C]//7th Annual Symposium on Switching and Automata Theory (swat 1966). [S.l. : s.n.], 1966: 36-46.

[29] VALIANT L G. The equivalence problem for deterministic finite-turn pushdown automata[J]. Information and Control, 1974, 25(2): 123-133.

[30] VALIANT L G, PATERSON M S. Deterministic one-counter automata[J]. Journal of Computer and System Sciences, 1975, 10(3): 340-350.

[31] ROMANOVSKII V Y. The equivalence problem for real-time deterministic pushdown automata[J]. Cybernetics, 1986, 22(2): 162-175.

[32] OYAMAGUCHI M. The equivalence problem for real-time DPDAs[J]. Journal of the ACM (JACM), 1987, 34(3): 731-760.

[33] SÉNIZERGUES G. The equivalence problem for deterministic pushdown automata is decidable[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 1997: 671-681.

[34] SÉNIZERGUES G. L(A)=L(B)? decidability results from complete formal systems[J]. Theoretical Computer Science, 2001, 251(1-2): 1-166.

[35] STOCKMEYER L J, MEYER A R. Word problems requiring exponential time (preliminary report)[C]//Proceedings of the fifth annual ACM symposium on Theory of computing. [S.l. : s.n.], 1973: 1-9.

[36] CHO S, HUYNH D T. The parallel complexity of finite-state automata problems[J]. Information and Computation, 1992, 97(1): 1-22.

[37] HIRSHFELD Y, JERRUM M, MOLLER F. A polynomial algorithm for deciding bisimilarity of normed context-free processes[J]. Theoretical Computer Science, 1996, 158(1-2): 143-159.

[38] STIRLING C. Deciding DPDA equivalence is primitive recursive[C]// International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2002: 821-832.

[39] JANČAR P. Equivalences of pushdown systems are hard[C]//International Conference on Foundations of Software Science and Computation Structures. [S.l. : s.n.], 2014: 1-28.

[40] PAIGE R, TARJAN R E. Three partition refinement algorithms[J]. SIAM Journal on Computing, 1987, 16(6): 973-989.

[41] KANELLAKIS P C, SMOLKA S A. CCS expressions, finite state processes, and three problems of equivalence[J]. Information and Computation, 1990, 86(1): 43-68.

[42] BAETEN J C, BERGSTRA J A, KLOP J W. Decidability of bisimulation equivalence for process generating context-free languages[J]. Journal of the ACM, 1993, 40(3): 653-682.

[43] CAUCAL D. Graphes canoniques de graphes algébriques[J]. RAIRO-Theoretical Informatics and Applications, 1990, 24(4): 339-352.

[44] HÜTTEL H, STIRLING C. Actions speak louder than words: Proving bisimilarity for context-free processes[J]. Journal of Logic and Computation, 1998, 8(4): 485-509.

[45] GROOTE J F. A short proof of the decidability of bisimulation for normed BPA-processes[J]. Information Processing Letters, 1992, 42(3): 167-171.

[46] CHRISTENSEN S. Decidability and decomposition in process algebras[J]., 1993.

[47] BAETEN J C, WEIJLAND W P. Process Algebra, volume 18 of[J]. Cambridge tracts in theoretical computer science, 1990: 141-147.

[48] HUYNH D T, TIAN L. Deciding bisimilarity of normed context-free processes is in Σp2[J]. Theoretical Computer Science, 1994, 123(2): 183-197.

[49] CZERWINSKI W, LASOTA S. Fast equivalence-checking for normed context-free processes[C]//IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010). [S.l. : s.n.], 2010.

[50] CHRISTENSEN S, HUTTEL H, STIRLING C. Bisimulation equivalence is decidable for all context-free processes[J]. Information and Computation, 1995, 121(2): 143-148.

[51] BURKART O, CAUCAL D, STEFFEN B. An elementary bisimulation decision procedure for arbitrary context-free processes[C]//International Symposium on Mathematical Foundations of Computer Science. [S.l. : s.n.], 1995: 423-433.

[52] JANCAR P. Bisimilarity on basic process algebra is in 2-ExpTime (an explicit proof)[J]. ArXiv preprint arXiv:1207.2479, 2012.

[53] SRBA J. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2002: 716-727.

[54] KIEFER S. BPA bisimilarity is EXPTIME-hard[J]. Information Processing Letters, 2013, 113(4): 101-106.

[55] FU Y. Checking equality and regularity for normed BPA with silent moves[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2013: 238-249.

[56] HE C, HUANG M. Branching bisimilarity on normed BPA is EXPTIME-complete[C]//2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science. [S.l. : s.n.], 2015: 180-191.

[57] HUANG M, YIN Q. Two lower bounds for BPA[C]//28th International Conference on Concurrency Theory (CONCUR 2017). [S.l. : s.n.], 2017.

[58] MAYR R. Weak bisimilarity and regularity of BPA is EXPTIME-hard[C]//In Proccedings of the 10th International Workshop on Expressiveness in Concurrency (EXPRESS'03. [S.l. : s.n.], 2002.

[59] BALCÁZAR J, GABARRO J, SANTHA M. Deciding bisimilarity isP-complete[J]. Formal aspects of computing, 1992, 4(1): 638-648.

[60] CHRISTENSEN S, HIRSHFELD Y, MOLLER F. Bisimulation equivalence is decidable for basic parallel processes[C]//International Conference on Concurrency Theory. [S.l. : s.n.], 1993: 143-157.

[61] JANCAR P. Strong bisimilarity on basic parallel processes in PSPACE-complete[C]//18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings. [S.l. : s.n.], 2003: 218-227.

[62] SRBA J. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard[C]//Annual Symposium on Theoretical Aspects of Computer Science. [S.l. : s.n.], 2002: 535-546.

[63] HIRSHFELD Y, JERRUM M, MOLLER F. A polynomial-time algorithm for deciding bisimilation equivalence of normed Basic Parallel Processes[J]. SICS Research Report, 1994.

[64] JANCAR P, KOT M. Bisimilarity on normed Basic Parallel Processes can be decided in time O (n3)[C]//Proceedings of the Third International Workshop on Automated Verification of Infinite-State Systems–AVIS:vol. 2004. [S.l. : s.n.], 2004: 9.

[65] CZERWIŃSKI W, HOFMAN P, LASOTA S. Decidability of branching bisimulation on normed commutative context-free processes[J]. Theory of Computing Systems, 2014, 55(1): 136-169.

[66] SRBA J. Complexity of weak bisimilarity and regularity for BPA and BPP[J]. Electronic Notes in Theoretical Computer Science, 2003, 39(1): 79-93.

[67] 尹强. 无限状态系统上分支互模拟等价验证[D]. 上海交通大学, 2016.

[68] STIRLING C. Decidability of bisimulation equivalence for normed pushdown processes[C]//International Conference on Concurrency Theory. [S.l. : s.n.], 1996: 217-232.

[69] SÉNIZERGUES G. Decidability of bisimulation equivalence for equational graphs of finite out-degree[C]//Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280). [S.l. : s.n.], 1998: 120-129.

[70] STIRLING C. Decidability of bisimulation equivalence for pushdown processes[R]. [S.l. : s.n.], 2000.

[71] JANČAR P, SCHMITZ S. Bisimulation Equivalence of First-Order Grammars is ACKERMANN-Complete[C]//2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). [S.l. : s.n.], 2019: 1-12.

[72] BENEDIKT M, GÖLLER S, KIEFER S, et al. Bisimilarity of pushdown automata is nonelementary[C]//2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science. [S.l. : s.n.], 2013: 488-498.

[73] SRBA J. Undecidability of weak bisimilarity for pushdown processes[C]// International Conference on Concurrency Theory. [S.l. : s.n.], 2002: 579-594.

[74] YIN Q, FU Y, HE C, et al. Branching bisimilarity checking for PRS[C]// International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2014: 363-374.

[75] SÉNIZERGUES G. The bisimulation problem for equational graphs of finite outdegree[J]. SIAM Journal on Computing, 2005, 34(5): 1025-1106.

[76] FU Y, YIN Q. Decidability of Epsilon Pushing PDA[J]., 2018.

[77] HIRSHFELD Y, JERRUM M. Bisimulation equivalence is decidable for normed process algebra[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 1999: 412-421.

[78] JANČAR P. Undecidability of bisimilarity for Petri nets and some related problems[J]. Theoretical Computer Science, 1995, 148(2): 281-301.

[79] LARSEN K G, SKOU A. Bisimulation through probabilistic testing[J]. Information and computation, 1991, 94(1): 1-28.

[80] Van GLABBEEK R J, SMOLKA S A, STEFFEN B. Reactive, generative, and stratified models of probabilistic processes[J]. Information and Computation, 1995, 121(1): 59-80.

[81] HENNESSY M. Exploring probabilistic bisimulations, part I[J]. Formal Aspects of Computing, 2012, 24(4-6): 749-768.

[82] FENG Y, ZHANG L. When equivalence and bisimulation join forces in probabilistic automata[C]//International Symposium on Formal Methods. [S.l. : s.n.], 2014: 247-262.

[83] CRAFA S, RANZATO F. A spectrum of behavioral relations over LTSs on probability distributions[C]//International Conference on Concurrency Theory. [S.l. : s.n.], 2011: 124-139.

[84] BAIER C, ENGELEN B, MAJSTER-CEDERBAUM M. Deciding bisimilarity and similarity for probabilistic processes[J]. Journal of Computer and System Sciences, 2000, 60(1): 187-231.

[85] ZHANG L, HERMANNS H, EISENBRAND F, et al. Flow faster: Efficient decision algorithms for probabilistic simulations[C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. [S.l. : s.n.], 2007: 155-169.

[86] BAIER C, HERMANNS H. Weak bisimulation for fully probabilistic processes[C]//International Conference on Computer Aided Verification. [S.l. : s.n.], 1997: 119-130.

[87] PHILIPPOU A, LEE I, SOKOLSKY O. Weak bisimulation for probabilistic systems[C]//International Conference on Concurrency Theory. [S.l. : s.n.], 2000: 334-349.

[88] BAETEN J C M, BERGSTRA J A, SMOLKA S A. Axiomatizing probabilistic processes: ACP with generative probabilities[J]. Information and Computation, 1995, 121(2): 234-255.

[89] BANDINI E, SEGALA R. Axiomatizations for probabilistic bisimulation[C]// International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2001: 370-381.

[90] DENG Y, PALAMIDESSI C. Axiomatizations for probabilistic finite-state behaviors[C]//International Conference on Foundations of Software Science and Computation Structures. [S.l. : s.n.], 2005: 110-124.

[91] HANSSON H, JONSSON B. A framework for reasoning about time and reliability[C]//[1989] Proceedings. Real-Time Systems Symposium. [S.l. : s.n.], 1989: 102-111.

[92] JOU C, SMOLKA S. Equivalences and complete axiomatizations for probabilistic processes."[J]. To appear, 1990.

[93] STARK E W, SMOLKA S A. A complete axiom system for finite-state probabilistic processes.[C]//Proof, language, and interaction. [S.l. : s.n.], 2000: 571-596.

[94] DENG Y. Semantics of probabilistic processes: an operational approach[M]. [S.l.]: Springer, 2015.

[95] JANČAR P. Bisimulation Equivalence of First-Order Grammars[C]//ESPARZA J, FRAIGNIAUD P, HUSFELDT T, et al. Automata, Languages, and Programming. Berlin, Heidelberg: Springer, 2014: 232-243. DOI: 10.1007/978-3-662-43951-7_20.

[96] JANCAR P. Equivalence of pushdown automata via first-order grammars[J]. ArXiv preprint arXiv:1812.03518, 2018.

[97] HOPCROFT J, ULLMAN J. Introduction to Automata Theory, Languages and Computation.[M]. [S.l.]: Addison-Wesley Publishing Company, 1979.

[98] CAUCAL D. On the regular structure of prefix rewriting[J]. Theoretical Computer Science, 1992, 106(1): 61-86.

[99] Van GLABBEEK R J. A complete axiomatization for branching bisimulation congruence of finite-state behaviours[C]//International Symposium on Mathematical Foundations of Computer Science. [S.l. : s.n.], 1993: 473-484.

[100] MILNER R. A complete axiomatisation for observational congruence of finite-state behaviours[J]. Information and Computation, 1989, 81(2): 227-247.

[101] JANČAR P, SRBA J. Undecidability of bisimilarity by defender's forcing[J]. Journal of the ACM (JACM), 2008, 55(1): 5.

[102] SCHNOEBELEN P. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets[C]//International Symposium on Mathematical Foundations of Computer Science. [S.l. : s.n.], 2010: 616-628.

[103] FIGUEIRA D, FIGUEIRA S, SCHMITZ S, et al. Ackermannian and primitive-recursive bounds with Dickson's Lemma[C]//2011 IEEE 26th Annual Symposium on Logic in Computer Science. [S.l. : s.n.], 2011: 269-278.

[104] SCHMITZ S. Algorithmic Complexity of Well-Quasi-Orders[D]. 2017.

[105] SCHMITZ S. The Parametric Complexity of Lossy Counter Machines[C]//. [S.l. : s.n.], 2019.

[106] JANČAR P. Decidability of DPDA Language Equivalence via First-Order Grammars[C]//2012 27th Annual IEEE Symposium on Logic in Computer Science. [S.l.]: IEEE, 2012: 415-424. DOI: 10.1109/LICS.2012.51.

[107] CAUCAL D. Bisimulation of context-free grammars and of pushdown automata[J]. Modal Logic and Process Algebra, 1995, 53: 85-106.

[108] MINSKY M L. Computation[M]. [S.l.]: Prentice-Hall Englewood Cliffs, 1967.

[109] JANCAR P, OSICKA P, SAWA Z. EXPSPACE-hardness of behavioural equivalences of succinct one-counter nets[J]. ArXiv preprint arXiv:1801.01073, 2018.

[110] STIRLING C. Decidability of Bisimulation Equivalence for Normed Pushdown Processes[J]. Theoretical Computer Science, 1998, 195(2): 113-131.

[111] STIRLING C. Decidability of DPDA equivalence[J]. Theor. Comput. Sci., 2001, 255(1-2): 1-31.

[112] SCHMITZ S. Complexity bounds for ordinal-based termination[C]// International Workshop on Reachability Problems. [S.l. : s.n.], 2014: 1-19.

[113] LOHREY M, D'ARGENIO P R, HERMANNS H. Axiomatising divergence[C]//International Colloquium on Automata, Languages, and Programming. [S.l. : s.n.], 2002: 585-596.

[114] SEGALA R, LYNCH N. Probabilistic simulations for probabilistic processes[J]. Nordic Journal of Computing, 1995, 2(2): 250-273.

致　谢

在博士研究的过程中，我得到了很多人的帮助。在这里，我要向他们表达衷心的感谢！

首先，特别感谢我的导师傅育熙教授。这些年来，傅老师在我的学习和科研中给了我莫大的帮助和支持。傅老师学术功底深厚，对很多学术领域都有深刻的见解。傅老师讲授的《计算复杂性》课程令我大开眼界，傅老师引导我们探索真理的场景令我至今难忘。傅老师每年会组织我们参加实验室的暑期学校，我从中了解到了很多其他的研究方向，扩展了我的视野。傅老师为实验室的学生创造了一个轻松自由的研究环境，尤其对于理论研究者来说，在这个环境下做研究是相当幸福的事情。在我的研究过程中，每当我遇到瓶颈时，傅老师总能为我指明方向。在本文的写作过程中，傅老师也提出了很多宝贵的建议。傅老师正直宽厚的人格魅力，勤奋积极的工作作风，严谨求实的治学态度都是我终生学习的榜样。

我要感谢龙环老师。龙老师非常热心，她时刻关注我们的研究状态，并且关心学生们的生活。在我状态不好时，龙老师会及时提醒和鼓励我。在我面临选择时，龙老师会给我很多经验和建议。龙老师这些年的帮助对我意义重大，我将始终铭记。感谢徐贤老师。从徐老师那里我学到了很多高阶演算的知识。和徐老师一起合作时，他会仔细验证每一个细节，他严谨的治学态度令我十分敬佩。感谢陈翌佳老师，陈老师每次的学术报告都非常精彩，希望未来有更多的机会向他学习。感谢邓玉欣老师，邓老师随和而儒雅，本文的概率进程演算部分得到了他的一些有价值的建议。感谢董笑菊老师，董老师讲授的《并行理论》课程为我的研究打下了很好的基础。感谢李国强老师，李老师是我进入理论计算机方向最早的引路人，并且给过我很多很好的建议。感谢蔡小娟老师，蔡老师讲授的《程序语言理论》课程令我受益匪浅。感谢张驰豪老师和符鸿飞老师，他们都曾经为我的研究提供过很多建议。

我要感谢尹强博士。和尹师兄的讨论是非常高效的，他总能很快理清我们讨论的思路和重点，抓住问题的关键，提出可行的办法。尹师兄不仅思路清晰，其写作也是简洁明了。和他在研究中的合作令我受益匪浅。感谢陶秀挺博士，他乐观豁达，热爱生活，和他一起度过了一段愉快的时光。感谢黄明璋博士。我们在许多公开问题上都做过讨论和尝试，遗憾最终没能解决。他思考的速度和深度给我留下了深刻的印象。我也要感谢何超栋博士，薛建新博士，汪洋博士，以及杨启哲，王培新，吴昊等师兄弟们，和他们一起学习，一起聚会是我博士期间最难

忘的经历之一。感谢曾经在 BASICS 实验室共同学习的所有同学，包括杨非，温韵清，方冰冰，刘立，雷素华，黄炫圭，王立超，李彦龙，靳阳，邓立聪，李春淼，王若愚，熊浩，崔毅以及其他兄弟姐妹们，和他们一起经历了许多值得一生珍藏的回忆。此外，感谢肖涛博士在我毕业过程中不厌其烦地解答我的问题，并给我很多建议。感谢室友王帅博士多年的友谊，一同经历了一段共同奋斗，相互鼓励的时光。

对于读到此文的每一个人，感谢你们的兴趣。

最后，也是最重要的，我要特别感谢我的家人。感谢我的父母，他们一直以来默默的支持和鼓励着我。感谢我的爱人和儿子，他们是我最大的精神来源。

上海交通大学 博士 学位论文答辩决议书

0140379002

姓 名	张文博	学 号	0140379002	所在学科	软件工程
指导教师	傅育熙	答辩日期	2020-9-10	答辩地点	上海交通大学徐汇校区工程馆338室
论文题目	互模拟等价性验证问题研究				

投票表决结果：5 / 5 / 5 （同意票数/实到委员数/应到委员数） 答辩结论：☑通过 ☐未通过

评语和决议：

张文博的博士论文研究了下推自动机和随机CCS两个模型的互模拟等价验证问题，选题具有重要的理论意义。论文研究内容及主要成果如下：

1、证明了下推自动机的强互模拟是Ackermann-难的，从复杂性的角度彻底解决了该问题；

2、从参数复杂性的角度证明了上述问题的一些新的上下界；

3、给出了随机CCS的一个可靠完备的公理系统。给出了随机CCS的有穷互模拟等价的一个判定算法。

论文所叙述的研究工作扎实，内容丰富，层次分明，全文有很好的一致性，对于文中的技术证明给出了很好的直观思路。论文解决了验证领域内的一个重要公开问题。论文提出的方法具有很好的创新性。论文工作表明，张文博具有本学科坚实宽广的理论基础和系统深入的专门知识。论文结构完整，写作规范，是一篇优秀的博士论文。答辩过程中回答清晰，经答辩委员会无记名投票，一致通过张文博的博士论文答辩，建议授予工学博士学位。

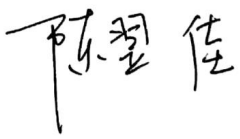
2020 年 9 月 10 日

答辩委员会成员	职务	姓名	职称	单位	签名
	主席	陈翌佳	教授	复旦大学	陈翌佳
	委员	徐贤	副教授	华东理工大学	徐贤
	委员	朱其立	教授	上海交通大学	
	委员	邓玉欣	教授	华东师范大学	
	委员	陈仪香	教授	华东师范大学	
	秘书	龙宇	副教授	上海交通大学	龙宇

张文博的博士论文研究互模拟等价的验证算法和复杂性。论文重点研究了下推自动机和随机 CCS 两个模型的互模拟等价验证问题。选题具有重要的理论意义。论文研究内容及主要成果如下：

1. 证明了下推自动机的强互模拟是 Ackermann-难的，从复杂性的角度彻底解决了该问题。
2. 从参数复杂性的角度证明了上述问题的一些新的上下界。
3. 给出了随机 CCS 的一个可靠完备公理系统。给出了随机 CCS 上的分支互模拟等价的一个判定算法。

论文所叙述的研究扎实，内容丰富，层次分明，全文有着很好的一致性，对于文中的技术性证明给出了很好的直观思路。论文解决了验证领域内的一个重要公开问题，且文中提出的方法具有很好的创新性。论文工作表明，张文博具有本学科坚实宽广的理论基础和系统深入的专门知识，独立从事学术研究工作能力强。论文结构完整，写作规范，是一篇优秀的博士论文。答辩过程中陈述清晰，回答问题正确。经答辩委员会无记名投票，一致通过张文博的博士论文答辩，建议授予张文博同学工学博士学位。



攻读博士学位期间已发表或录用的论文

[1] Wenbo Zhang, Xian Xu, Qiang Yin, Huan Long. On the Interactive Power of Higher-order Processes Extended with Parameterization. Submitted.

[2] Wenbo Zhang. The Parameterized Complexity of Bisimulation Equivalence for Normed Pushdown Automata. Submitted.

[3] Wenbo Zhang, Qiang Yin, Huan Long, Xian Xu. Bisimulation Equivalence of Pushdown Automata is Ackermann-Complete. In Proceedings of the 47th International Colloquium on Automata, Languages and Programming (ICALP'20), 141:1-14, 2020.

[4] Wenbo Zhang, Huan Long, Xian Xu. Uniform Random Process Model Revisited. In Proceedings of the 17th Asian Symposium on Programming Languages and Systems (APLAS'19), 388-404, 2019.

[5] 张文博, 龙环. 向量加法系统验证问题研究综述. 软件学报. 29(6), 1566-1581, 2018.

攻读博士学位期间参与的项目

[1] 国家自然科学基金，面上项目，61872142，带参数化操作的高阶进程研究，2019/1-2022/12。

[2] 国家自然科学基金，面上项目，61772336，无穷状态系统等价性验证，2018/01-2021/12。

[3] 国家自然科学基金，面上项目，61472239，进程理论中的否定结果研究，2015/01-2018/12。