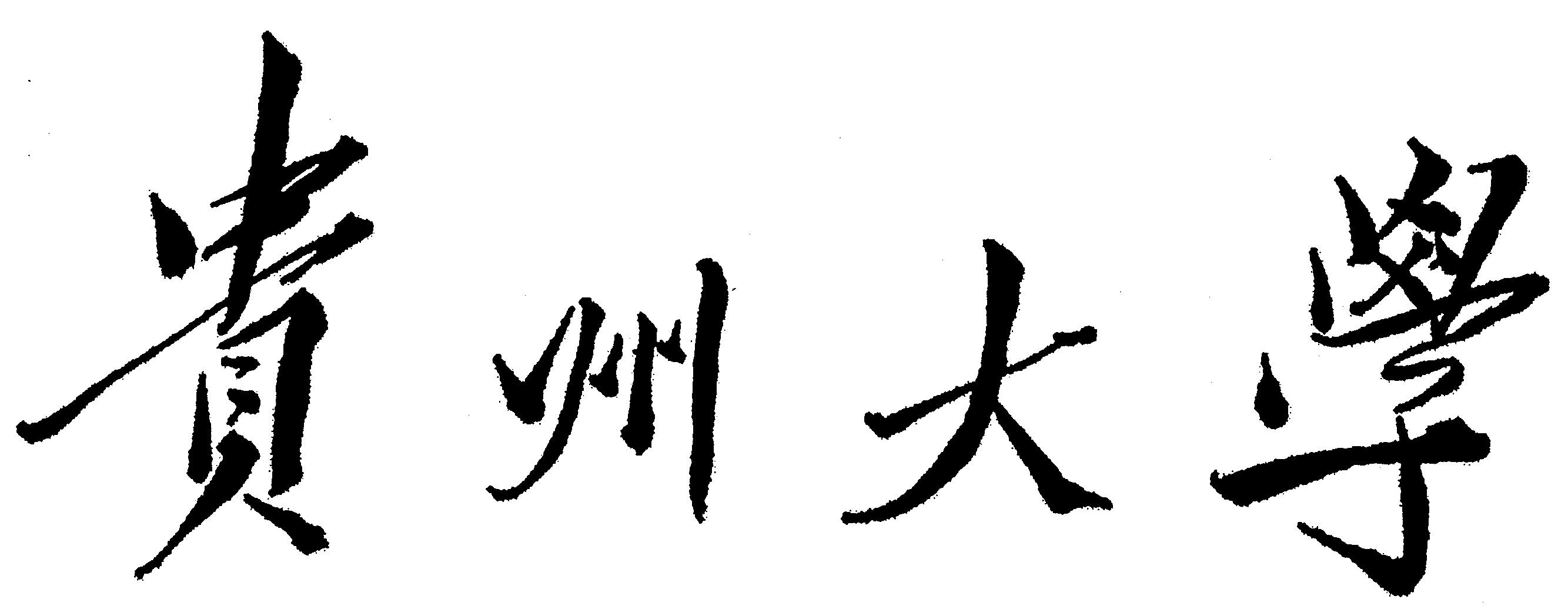
论文编号：



**2022**届博士学位论文

基于遗忘的反应式系统

最弱充分条件研究

学科专业：

软件工程

研究方向： 软件工程技术与人工智能

中国·贵州·贵阳

2022年 3月

# 摘 要

反应式系统是在对应用程序的即时响应（responsive）、 回弹性（resilient）、 弹性

（elastic）以及消息驱动（message driven）要求的基础上产生和发展起来的不终止系 统。 随着反应式系统越来越复杂，系统正确性、 系统及其系统规范（specification）之 间的一致性越来越难以得到保证。 模型检测是一个保证系统正确性行之有效的方法 之一，此时反应式系统被表示为一个Kripke结构。 然而在模型检测中，若系统不满足 给定的规范（即与规范不一致），寻找使得系统满足规范所需的最弱信息是长时间以 来的一个重要问题。 与这个问题密切相关的两个概念是最强必要条件（the strongest necessary condition, SNC）和最弱充分条件（the weakest sufficient condition, WSC），其 分别对应于形式化验证中的最强后件（the strongest post-condition, SP）和最弱前件

（the weakest precondition, WP）。 然而，现有的方法不能计算像反应式系统这样不终止 系统的SNC和WSC。 此外，随着系统的更新和演化，现有的规范不可避免地会与新的 知识相冲突。 此时，如何将之前融入的元素在不影响其它信息的情况下“移除”也是 个亟待解决的问题。

系统规范的描述语言以时序逻辑为主。 其中计算树逻辑（Computation tree logic， CTL）是一种重要的分支时间时序逻辑，其具有模型检测能多项式时间完成的特 性，因此被广泛用于系统规范描述中。 但是CTL具有表达能力不够强的缺陷，*µ*-演算

（*µ*-calculus）是一种比CTL表达能力更强但模型检测更加复杂的逻辑语言。 尽管如此， CTL和*µ*-演算的语义都是Kripke语义，这与反应式系统被表示为Kripke结构有着直接的 联系。 因而，本文以这两种逻辑语言为研究背景，探索这两种语言下反应式系统上述 问题的解决方法。

遗忘是一种知识抽取的技术，其被应用于信息隐藏、 冲突解决和计算逻辑差等领

域。 本文从遗忘的角度出发解决上述提到的问题，主要研究成果如下：

1. 研究了CTL下遗忘的概念及其相关性质。 首先，从模型在某个原子命题集合上 互模拟的角度给出了遗忘的定义；其次，探讨并证明了遗忘算子的代数属性，包括模 块性、交换性和同质性；第三，基于Zhang等人提出的四条公设，给出CTL遗忘的四条 公设，表示定理表明这四条公设对遗忘是充分且必要的，即：遗忘的结果满足四条公 设，且满足那四条公设的公式为遗忘的结果。

2. 研究了*µ*-演算遗忘。 *µ*-演算具有均匀插值（uniform interpolation）性质，本文 证明了*µ*-句子遗忘与均匀插值是等价的。 此外，提出一种新的互模拟，并证明*µ*-公式 对这种互模拟是不变的（invariant）。 最后，研究了*µ*-演算下遗忘的基本属性和复杂性， 为均匀插值的研究提供了新的思路。

贵州大学博士学位论文

3. 给出了遗忘与WSC（SNC）和知识更新（knowledge update）的关系。WSC对模 型的验证和修改具有重要作用，现有方法只能计算可终止模型的WSC，而不能计算 像反应式系统这类不可终止系统的WSC。 本文通过遗忘给出了计算WSC（SNC）的方 法，利用遗忘定义了CTL和*µ*-演算下的知识更新，并证明其满足Katsuno等人提出的知 识更新的八条公设。

4. 提出一种基于模型的CTL遗忘计算方法。 在这种方法中，规定了公式的长度为 整数*n*、公式所依赖的模型个数为有限个及构成公式的原子命题是有限的。 此时，公式 的模型可以用其特征公式——CTL公式来表示。 因此，遗忘结果可以由其所有模型在 给定原子命题集合上的特征公式的析取来表示。

5. 提出一种基于归结的方法计算CTL下的遗忘。 该方法使用Zhang等人提出的归

结系统*R*≻*,S* （clausal resolution calculus for CTL）。 在这种方法中，首先将CTL公式转

CTL

换为SNF*g*

CTL

子句（separated normal form with global clauses for CTL）的集合，最后再

将SNF*g*

CTL

子句转换为CTL公式。 在这一过程中，需要计算遗忘的公式总是和各个过程

的输出保持互模拟等价关系。

6. 利用Prolog实现了5中提到的基于归结的计算CTL遗忘的方法，并做了相应的实 验。 从标准数据集和随机产生的数据集里做了两组实验，分别为计算遗忘和SNC。 实 验表明公式越长或遗忘的原子个数越多，效率越低；此外，在随机产生的公式中，大 部分情况下能计算出SNC。

这些结果为时序逻辑下的遗忘理论的研究提供了框架，并为模型更新提供了辅助

工具——WSC。

关键词： 遗忘理论，最强必要条件，最弱充分条件，知识更新，人工智能

**Abstract**

With the software (hardware) systems of a computer becoming more and more complex, it gets hard to guarantee the correctness of systems and the consistency between systems and its specification. Model checking is a valid method to ensure the correctness of systems. However, it is an important problem in model checking to fix the system to make it consistent with its specification when the system does not satisfy the given specification. Moreover, in such a scenario, two logical notions introduced by E. Dijkstra are highly informative: the *strongest necessary condition* (SNC) and the *weakest sufficient condition* (WSC) of a given specification. These correspond to the *strongest post-condition* (SP) and the *weakest precondition* (WP) of such specification, respectively. Besides, the specification at hand is an unavoidable conflict with the new knowledge when the information of the system becomes clearer. In this case, another problem that needs to solve is to “*eliminate*”the containing elements without affecting the other information.

*Forgetting*, a technique to distill knowledge, was used to hide information, solve the con- flict, compute logic differences, and so on. This paper solves the above problems from the point of forgetting. The major contributions are as follows:

1. Given the definition and properties of forgetting in CTL. First, this paper defines forget- ting from the point of bisimulation over the given signature. Second, we explore the algebraic properties, including modularity, commutativity, and homogeneity, of forgetting. Third, the

expression theorem shows that there is an “if and only if”relation between forgetting and

the forgetting postulates proposed by Zhang et al., i.e., the result of forgetting satisfies the forgetting postulates, and the formula which satisfies the forgetting postulates is the result of forgetting.

1. Proposed a resolution-based method to compute the forgetting in CTL. This approach bases the resolution system proposed by Zhang et al., and the CTL formula is transformed into

a set of SNF*g*

CTL

clauses (separated normal form with global clauses for CTL) at first. At the

end of the approach, the SNF*g*

CTL

clauses are transformed into a CTL formula to obtain the

forgetting result. It is noteworthy that the output of each process is bisimilar equivalent of the input CTL formula.

1. We outline a situation in which the forgetting is closed. In this case, the length of formulas are limited to integer *n*, and the number of atoms for formulas and Kripke structures is finite. To prove that it is closed, we define the characterizing formula (i.e., a CTL formula)

贵州大学博士学位论文

of the (finite) initial K-structure and show that each CTL formula is equivalent to a disjunction of the characterizing formulas of its models. This fact means that the result of forgetting some atoms from a CTL formula always exists.

1. Explored the forgetting in *µ*-calculus. *µ*-calculus is a kind of logic which have uniform interpolation. This paper shows that the uniform interpolation and forgetting in *µ*-calculus are equivalent, this means that the forgetting in *µ*-calculus is closed which is the biggest differ- ence between *µ*-calculus and CTL. Moreover, the properties and complexity results related to forgetting are given, which proposes a new point for studying uniform interpolation.
2. Given the relation between forgetting and WSC (SNC or knowledge update). WSC is important to the verification and modification of the system. However，the existing methods

can only compute the WSC of a terminable system, and the WSC can not be obtained in non- terminable systems (e.g., the reactive system). This paper shows how to compute the WSC of the reactive system and define the knowledge update using forgetting to satisfy the postulates [proposed by Katsuno et al.](#_bookmark133).

1. Implemented the algorithm proposed in 2, and some experiments are shown. Two experiments, computing forgetting, and SNC, were performed for standard and randomly gen- erated datasets. Experiments show that the longer the formula is or the more atoms are forgot- ten, the lower the efficiency is. Moreover, SNC can be calculated in most cases of randomly generated formulas.

Its significance mainly provides a framework for the study of forgetting theory under temporal logic and provides an auxiliary tool for the model update.

**Keywords:** forgetting，the strongest necessary condition，the weakest sufficient condition， knowledge update

# 目 录

[摘要](#_bookmark1) **i**

## [Abstract](#_bookmark2) iii

[目录](#_bookmark3) **v**

|  |  |  |
| --- | --- | --- |
| [第一章](#_bookmark4) | [绪论](#_bookmark4) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **1** |
| [1.1](#_bookmark5) | [研究背景与意义](#_bookmark5) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 1 |
|  | [1.1.1 研究背景](#_bookmark6) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 1 |
|  | [1.1.2 研究意义](#_bookmark10) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 4 |
| [1.2](#_bookmark11) | [相关研究工作回顾](#_bookmark11) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 4 |
|  | [1.2.1 反应式系统](#_bookmark12) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 5 |
|  | [1.2.2 遗忘理论](#_bookmark13) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 5 |
|  | [1.2.3 SNC和WSC](#_bookmark14) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 8 |
| [1.3](#_bookmark15) | [研究目标及主要结果](#_bookmark15) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 9 |
| [1.4](#_bookmark16) | [论文组织结构](#_bookmark16) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 11 |
| [第二章](#_bookmark18) | [基础知识](#_bookmark18) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | **13** |
| [2.1](#_bookmark19) | [反应式系统](#_bookmark19) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 13 |
| [2.2](#_bookmark20) | [Kripke结构](#_bookmark20) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 13 |
|  | [2.2.1 真假赋值和K-解释](#_bookmark21)· · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 14 |
|  | [2.2.2 Kripke结构](#_bookmark24) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 16 |
| [2.3](#_bookmark25) | [时序逻辑](#_bookmark25) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 17 |
|  | [2.3.1 计算树逻辑（CTL）](#_bookmark26) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 17 |
|  | [2.3.2 CTL的标准形式](#_bookmark28) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 22 |
|  | [2.3.3 *µ*-演算](#_bookmark32) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 24 |
|  | [2.3.4 *µ*-公式的析取范式](#_bookmark33) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 28 |
| [2.4](#_bookmark34) | [CTL下的归结](#_bookmark34) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 29 |
| [2.5](#_bookmark36) | [遗忘理论和SNC（WSC）](#_bookmark36) · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · · | 32 |

[2.5.1 经典逻辑的遗忘](#_bookmark37) 32

[2.5.2 模态逻辑S5的遗忘](#_bookmark39) 34

[2.5.3 遗忘的计算方法](#_bookmark41) 36

[2.5.4 基于遗忘的SNC（WSC）计算](#_bookmark42) 38

[2.6 本章小结](#_bookmark44) 40

[第三章 CTL和µ-演算遗忘理论](#_bookmark45) 41

[3.1 CTL遗忘理论](#_bookmark46) 41

[3.1.1 互模拟](#_bookmark47) 41

[3.1.2 遗忘算子及其性质](#_bookmark57) 48

[3.2 *µ*-演算遗忘理论](#_bookmark60) 55

[3.2.1 Var-互模拟](#_bookmark61) 55

[3.2.2 遗忘算子及其性质](#_bookmark69) 63

[3.2.3 计算复杂性](#_bookmark79) 75

[3.3 本章小结](#_bookmark81) 77

[第四章 遗忘理论在反应式系统中的应用](#_bookmark82) **79**

[4.1 最弱充分条件](#_bookmark83) 79

[4.2 知识更新](#_bookmark88) 83

[4.3 本章小结](#_bookmark92) 86

[第五章 CTL遗忘计算方法](#_bookmark94) 88

[5.1 基于模型的有界CTL遗忘计算](#_bookmark95) 88

[5.1.1 描述初始结构](#_bookmark96) 89

[5.1.2 遗忘封闭性及复杂性](#_bookmark106) 101

[5.1.3 基于模型的遗忘算法](#_bookmark110) 103

[5.2 基于归结的遗忘计算方法](#_bookmark111) 105

[5.2.1 基于归结的算法CTL-forget](#_bookmark113) 106

[5.2.2 基于Prolog的CTL-forget算法实现](#_bookmark121) 115

[5.2.3 实验](#_bookmark122) 116

[5.3 本章小结](#_bookmark128) 119

[第六章 总结与展望](#_bookmark129) 121

[6.1 工作总结](#_bookmark130) 121

[6.2 研究展望](#_bookmark131) 122

[参考文献](#_bookmark132) 123

[攻读博士学位期间科研和论文情况](#_bookmark241) **136**

[致谢](#_bookmark242) 137

[1.1 由系统故障引起的重大事件概览](#_bookmark7) 1

[2.1 转换规则](#_bookmark29) 22

[2.2 CTL化简规则，其中*Q* ∈ {A*,* E}且*T* ∈ {X*,* G*,* F}。](#_bookmark30) 23

[2.3 归结规则](#_bookmark35) 30

[5.1 计算CTL-forget(*ϕ,V* )所使用的CPU时间（单位：秒(s)）](#_bookmark123) 117

[1.1 汽车制造企业模型](#_bookmark9) 4

[1.2 本文的章节内容组织结构图](#_bookmark17) 11

[3.1 K-结构之间的*V* -互模拟关系示意图](#_bookmark50) 42

[3.2 两个{*ch*}-互模拟的Kripke结构示意图](#_bookmark64) 57

[4.1 初始结构间的](#_bookmark90)≤[*M* 关系。](#_bookmark90) 84

[4.2 状态空间为{*s*0*, s*1}的六个Kripke结构示意图](#_bookmark93) 87

[5.1 初始结构*K*2（源于图](#_bookmark104)[3.1](#_bookmark50)[）及其计算树示意图](#_bookmark104) 96

[5.2 基于归结的遗忘的主要流程图](#_bookmark112) 105

[5.3 计算CTL-forget(*ϕ,V* )使用的时间和在“移除原子命题”步骤后SNF*g* 子](#_bookmark124)

[CTL](#_bookmark124)

[句的个数，其中*ϕi* = 12。](#_bookmark124) 117

[5.4 计算CTL-forget(*ϕ,V* )使用的时间和在“移除原子命题”步骤后SNF*g* 子](#_bookmark125)

[CTL](#_bookmark125)

[句的个数，其中*ϕi* = 16。](#_bookmark125) 118

[5.5 计算3-CNF公式SNC的CPU时间](#_bookmark126) 118

[5.6 计算CTLSNC的平均时间和存在SNC的公式占比](#_bookmark127) 119

# 第一章 绪论

首先，本章介绍研究背景，分析保障系统正确的重要性，并阐述研究意义。 其次， 综合分析遗忘理论、 最强必要（最弱充分）条件等关键技术的国内外研究动态，以及 遗忘理论在形式化验证中的应用研究趋势。 然后，围绕研究对象凝练出关键问题与目 标。 进一步，介绍本文的核心研究内容以及取得的主要成果。 最后，给出具体章节组 织安排。

**1.1** 研究背景与意义

**1.1.1** 研究背景

形式化验证是一种广泛应用在硬件[[2](#_bookmark134)-[4](#_bookmark135)]和软件系统中[[5](#_bookmark136)-[6](#_bookmark137)]有别于测试的、 采用数学 方法证明系统满足给定特性的验证（verification）技术。 软件和硬件的缺陷会导致严重 的后果，如表[1.1](#_bookmark7)中列出的几个重大事件。 近年来，为了减少系统（尤其是像火箭发射 系统和卫星发射系统等关键领域的系统）错误带来的损失，形式化方法的研究与应用 越来越受到人们的重视。 INTEL、 AMD、 IBM、 NVIDIA、 CADENCE、 Motorola、 西 门子和微软等大型公司纷纷引入了形式化验证方法。 与此同时，学术界也在形式化验 证领域取得了突破性的成果，比如：剑桥大学对ARM6处理器进行了验证[[7](#_bookmark138)]，为类似 于ARM这样的处理器提供了潜在的形式化验证方法，德国的Verisoft项目验证了一个一 万行的操作系统内核[[8](#_bookmark139)]。

表 1.1: 由系统故障引起的重大事件概览

时间 事故原因 损失

1991年 美国爱国者导弹系统舍入错误 28名士兵死亡、100人受伤等

1996年 阿丽亚娜5型运载火箭软件在不同飞

行条件下的代码重用

火箭与其它卫星毁灭

1999年 火星探测器用错度量单位 探测器坠毁并造成了3.27亿美元的损

失

2011年 温州7.23动车信号设备在设计上存在

严重的缺陷

动车脱节脱轨、多人失去生命

反应式系统是一种特殊的应用系统，它不终止且与环境有着持续不断的交互。 其 应用十分广泛：上至航空电子等安全攸关的领域，下至生活息息相关的汽车电子。 必 须在指定时间期限内完成对外部事件的检测和目标事件的响应，是安全攸关反应式系

统的核心要求。 因而，近年来包括形式化方法在内的多种方法用于确保反应式系的正

确性。

形式化验证有两种主要的验证方法：自动定理证明（Automated theorem proving） 和模型检测（Model checking)。 在自动定理证明中，系统模型和规范（specification） 被同一种形式化语言分别描述为*ϕimp*和*ϕspec*， 然后证明其推理的正确性， 即:证明 “*ϕimp* → *ϕspec*”或“*ϕimp* ↔ *ϕspec*”，即证明模型系统*ϕimp*是否满足给定的规范*ϕspec*。常用 的自动定理证明方式有归结（Resolution）[[9](#_bookmark140)]和常用于模态逻辑的基于表推理（tableau） [[10](#_bookmark141)]的方法。 计算机程序和系统验证（verification of computer programs and systems）是 自动定理证明的新领域，它使用基于规则的方法来验证程序的正确性。 然而，当程序 为循环语句时，验证所需要的不变式（invariant）获取是个困难的问题。 因此，为了避 免类似于在Hoare逻辑[[11](#_bookmark142)]、 动态逻辑[[12](#_bookmark143)]和分离逻辑[[13](#_bookmark144)]种为了形式化验证而寻找不变式 问题，Fangzhen Lin提出将一个程序（program）转换为一阶理论，然后再使用一阶理 论中的自动定理证明方法来验证[[14](#_bookmark145)]。

形式化验证的模型检测首先由Clarke提出，并用于解决并发系统验证问题[[15](#_bookmark146)]。 Clarke和Emerson指出，在有限状态的并发系统中，使用时态逻辑推演系统（deductive system）中的公理和推理规则进行构造性证明（proof construction）的方法来证明该系 统是否满足给定的规范是不必要的[[16](#_bookmark147)]。 因为在有限状态并发系统中，并发系统可以被 抽象为一个Kripke结构*M* ，规范被表示成一个逻辑公式*ϕ*；此时，该验证问题就变成 检验一个Kripke结构是否满足该公式，即模型检测（*M* |=? *ϕ*）: 判断*M* 是否是*ϕ*的一 个模型。

近年来，模型检测问题在知识表示与推理（KR）领域的推进下取得了丰富的科 研成果，例如：基于SAT的有界（bounded）模型检测[[17](#_bookmark148)]和基于OBDD的符号模型检 测[[18](#_bookmark149)]已经使得模型检测问题在时间和空间效率上取得了很大的进步，在一定程度上 缓解了其固有的状态空间爆炸问题。 此外，大量优质的模型验证器，如：NuSMV[1](#_bookmark0)、 SPIN[2](#_bookmark0)、 Uppaal[3](#_bookmark0) 等，也相继发展起来，并且大部分的验证器都可以用来验证多种时态 逻辑描述的公式。

时态逻辑作为一种描述系统规范的形式化语言，其研究状态随时间变化的系统的 逻辑特性。 由于软件和硬件系统运行的本质是状态变化的过程，所以时态逻辑在软件 和硬件系统验证中应用得相当广泛。 计算树逻辑（Computation Tree Logic, CTL）是分 支时态逻辑的一种，其模型检测是多项式时间可完成的。 然而，CTL表达系统性质的 能力不如*µ*-演算（*µ*-calculus），如：“某给定的系统中存在一条路径使得该路径上处于 偶数位置的状态满足特定的性质”这一规范是不能用CTL来表示的[[19](#_bookmark150)]。 充分考虑这两 种逻辑语言自身的特性，本文主要研究CTL和*µ*-演算。 因此，本文所说的公式指CTL

1<http://nusmv.fbk.eu/>

2<http://spinroot.com/spin/whatispin.html> 3<http://www.uppaal.org/>

（或*µ*-演算）公式，即用来描述一个规范（或性质）的公式是CTL（或*µ*-演算）公式。

在模型检测中，反应式系统通常用Kripe结构表示[[19](#_bookmark150)-[20](#_bookmark151)]。 当给定了反应式系统

的Kripke结构模型*M* 和规范*ϕ*，就存在模型检测问题*M* |=? *ϕ*：

( *Yes,* 若*M* |= *ϕ*；

系统输出=

*No*和负例*,* 否则。

此时， 使用什么信息对*M* 进行修正， 使得其满足给定的规范是一个重要的问 题。 这就是寻找最弱前件（weakest precondition， WP） 问题[[21](#_bookmark152)]， 在人工智能（artificial intelligence，AI）中也称为最弱充分条件（weakest sufficient condition，SNC），与之对 偶的另一个概念是最强必要条件（strongest necessary condition，SNC）[[22](#_bookmark153)-[23](#_bookmark154)]。

从知识抽取（或“消除”）的角度来看。 出于安全考虑，查看信息时需要将有的信 息隐藏而只抽取关注部分信息。 此外，随着时间推移，由于系统的部分信息会因某些 原因而过时，需要将这样的信息在不影响其它信息的情况下“消除”。 如下示例：

例 **1.1** (汽车制造企业模型)**.** 一个汽车制造企业能够生产两种汽车：小轿车(*se*)和跑 车(*sp*)。 每隔一段时间，该企业都会做一个生产决策(*d*)，即：合理的生产计划。 刚开 始的时候，该企业做出了具有三个选择的方案：

(1) 先生产足够的*se*，然后在再产*sp*；

(2) 先生产足够的*sp*，然后再生产*se*；

(3) 同时生产*se*和*sp*。

这一过程可以由图 [1.1](#_bookmark9)中的Kripke结构（带标签的状态转换图）*M* = (*S, R, L*)形式化地

展现出来，其中:

∙ *V* = {*d, s, se, sp*}为该工厂所需要考虑的原子命题集；

∙ *S* = {*s*0*, s*1*, s*2*, s*3*, s*4}为状态空间；

∙ *R* = {(*s*0*, s*1)*,* (*s*1*, s*2)*,* (*s*1*, s*3)*,* (*s*1*, s*4)*,* (*s*2*, s*0)*,* (*s*3*, s*0)*,* (*s*4*, s*0)}为状态转换关系集；

∙ *L* : *S* → 2*V* 为标签函数，具体地：*L*(*s*0) = {*d*}、*L*(*s*1) = {*s*}、*L*(*s*2) = {*se*}、*L*(*s*3) =

{*sp*}和*L*(*s*4) = {*se, sp*}。

假定，由于经济危机或者战略调整，导致该企业不能再生产跑车。 这意味着所有

关于*sp*的规范和Kripke结构都不再需要考虑，因此应该“移除”。

日常生活中也有很多上述例子中的场景，如：商业交易过程、软件开发过程等[[24](#_bookmark155)]。 但是对于给定原子命题集，从这些大型系统（或规范）中“移除”一些原子命题，但 保持与这些原子命题无关的性质是一个复杂的问题。

基于上述存在的问题，下面给出一些解决方案及其意义所在。

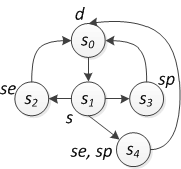


图 1.1: 汽车制造企业模型

**1.1.2** 研究意义

最强后件（the strongest post-condition，SP）和最弱前件（WP）（分别对应于上文 提到的SNC和WSC）是形式化验证中两个重要的概念，其不仅被用于汇编语言程序 推理[[25](#_bookmark156)]和制定验证条件[[26](#_bookmark157)]，还被应用于形式化验证过程中的负例生成[[27](#_bookmark158)]和系统精化

（refinement）[[28](#_bookmark159)]。当模型*M* 不满足规范*ϕ*（*M* |= *ϕ*）时，找到某个性质*ψ*，使得若*M* 按 照此性质进行修改后得到的新模型*M* ′能满足*ϕ*是必要的，即*ψ*是使得*M* |= *ϕ*成立的充 分条件（*M* |= *ψ* → *ϕ*）。 然而，现有方法不能直接应用于如下情形：计算当*M* 为不终 止系统（如：反应式系统（reactive system））时使得“*M* |= *ϕ*”为真的最强必要条件 和最弱充分条件（其详细原因将在下文指出）。 此时，探索如何计算“在*M* 下使得*ϕ*满 足的定义在某个符号集合上的*SNC*和*WSC*”将更进一步完善模型检测问题，同时也为 基于*WSC*的负例生成和精化提供了理论依据和新的计算方法。

本文将探索一种称为遗忘（forgetting）的方法来计算充分（必要）条件。 正如下 文将要说到的，遗忘理论作为知识表示与推理（KR）中的重要理论，具有较长的科研 历史，且在许多逻辑中都有了较为成熟的研究。 然而，在时序逻辑方面的研究目前还 不成熟。 因此，本文的研究将为时序逻辑下遗忘理论的研究提供一个理论框架。 与此 同时，借助遗忘计算上述形式化验证问题中的充分（必要）条件，这架起了*KR*与形式 化验证的桥梁。

**1.2** 相关研究工作回顾

遗忘是一种重要的知识抽取工具，它具有均匀插值（uniform interpolation）和二阶 量化消解（second-order quantifier elimination，SOQE）两种称谓。 在很长一段时间内， 遗忘被用于描述逻辑中本体（ontology）摘要的提取、 敏感信息的隐藏和软件工程中 计算两个文件的逻辑差（logical differences）。 此外，其也被用于包括信念更新（belief update）、修改（repair）、规划（planning）和知识独立性的其它领域。

反应式系统是一种不终止的、 与环境有着持续不断交互的系统，其应用广泛、 种

类繁多。本文探究如何使用遗忘计算反应式系统下的SNC（WSC），下面就与本文密切

相关的反应式系统、遗忘理论和SNC（WSC）进行回顾。

**1.2.1** 反应式系统

随着应用程序在各个领域的应用，对其需求已经发生了重大变化，继而引起模式 变化。 以前，一个大型应用程序的运行通常需要数十台服务器，这么高昂的代价换来 的是效果和维护成本不成正比：秒级的响应时间，需要数小时的维护时间。 当前，移 动和云端技术的成熟使应用程序不拘泥于以前的方式，它们可以被部署到移动设备和 具有多核心处理器的云端集群上。 但是用户对应用程序的响应时间要求从秒级变到毫 秒级，相应的成功率要求为100%，程序需要处理的数据量也从之前的GB级扩展到现 在的PB级。 原有的软件架构已经无法满足当前的需求。

在这种背景下，应用程序需要具备以下特质：即时响应性（Responsive）、 回弹 性（Resilient）、 弹性（Elastic）以及消息驱动（Message Driven）[4](#_bookmark0)。 反应式系统是一 种与环境有着持续不断交互的系统，具有不终止的性质[[19](#_bookmark150)]，且同时满足上述几种要 求，且从传统的静态模式逐步向动态、 开放、 自适应、 服务化的模式演化[[29](#_bookmark160)]。 反应 式系统由来已久[[30](#_bookmark161)-[31](#_bookmark162)]，且种类繁多，如微处理器（microprocessor）、 计算机操作系统

（computer operating systems），航空交通管制系统（air traffic control systems）、 车载电

子设备（on-board avionics）以及其它嵌入式系统。

此外，反应式系统的应用非常广泛：包括航空电子等安全攸关的领域和生活息息 相关的汽车电子等领域。 安全攸关反应式系统的核心要求是：必须在指定时间期限 内完成对外部事件的检测和目标事件的响应，否则会产生灾难性的后果[[32](#_bookmark163)]。 为了确 保反应式系统的正确性和实时性，研究者们提出了不同的解决方法，包括：模型检 查方法[[19](#_bookmark150),[33](#_bookmark164)-[34](#_bookmark165)]、 目标驱动的运行时需求建模框架[[29](#_bookmark160)]、 基于图模型的实时规则调度方 法（graph-based real-time rule scheduling,简称GBRRS） [[32](#_bookmark163)]、 SPARDL[[35](#_bookmark166)]和选择性测试 方法[[36](#_bookmark167)]等。

在上面的方法中，多数是验证（检测）给定的系统是否具有某种性质，而当系统 不满足该性质时，对如何寻找可靠的信息（称为充分条件）来对系统进行修改没有相 关研究。 反应式系统是一种不终止的系统，通常被看作一个Kripke结构。 基于此，本 文探索如何计算该充分条件，使得系统在该条件下满足给定的性质。

**1.2.2** 遗忘理论

遗忘这一词源于Lin等人关于一阶逻辑（first-order logic，FOL）的工作[[37](#_bookmark168)]，在此

之前的研究中多提到的是均匀插值[[38](#_bookmark169)-[39](#_bookmark170)]和SOQE[[40](#_bookmark171)]。

在命题逻辑中（propositional logic， PL）， 公式*ϕ*遗忘一个原子命题*p*， 通常记

为*Forget*(*ϕ,* {*p*})，得到的结果为*ϕ*[*p/*⊥]∨*ϕ*[*p/*⊤]（其与∃*pϕ*等价），其中*ϕ*[*X /Y* ]为将*ϕ*中*X* 的

4[https://www](http://www.reactivemanifesto.org/zh-CN).reactiv[emanifesto.org/zh-CN](http://www.reactivemanifesto.org/zh-CN)

全部出现替换为*Y* 得到的结果。 公式*ϕ*中遗忘有限的原子命题集*P*的定义如下：

*Forget*(*ϕ,* 0/ ) = *ϕ,*

*Forget*(*ϕ, P* ∪ {*q*}) = *Forget*(*Forget*(*ϕ,* {*q*})*, P*)*.*

在FOL中，遗忘通常被看为SOQE问题的一个实例：从FOL公式*ϕ*中遗忘掉一个*n*- 元谓词*P*，得到的结果为一个二阶公式∃*Rϕ*[*P/R*][[37](#_bookmark168)]。 从这个角度看来，遗忘就是找到 一个与二阶公式∃*Rϕ*[*P/R*]等价的一阶公式。 然而，二阶逻辑的表达能力是严格大于一 阶逻辑的，因而可以容易得出FOL下的遗忘不是封闭的结论，也就是从某些一阶公式 中遗忘某些谓词得到的结果不可以用一阶公式来表示。 作为FOL的一个子类，描述逻 辑公式的遗忘也不总是存在的[[41](#_bookmark172)]，甚至对最基本的描述逻辑ALC而言，遗忘的存在性 问题都是不可判定的。 尽管如此，描述逻辑作为一种在语义网领域很重要的语言，其 子类（包括ALCOHI和ALCOIH）中的遗忘通常被用来抽取视图（review）[[42](#_bookmark173)-[46](#_bookmark174)]。

现 有 的 计 算 一 阶 逻 辑 和 描 述 逻 辑 遗 忘 的 方 法 有 基 于 归 结（resolution） 和 基 于Ackermann引理的方法[[47](#_bookmark175)]。 其中基于归结的方法是一种基于子句归结的方法，其 基础是归结规则。 通常在这种方法中首先把公式转换为其子句形式，然后使用归结规 则，最后移除含有要遗忘的谓词（原子命题）的子句，得到的结果可能就为遗忘的结 果（在后文中会详细介绍与本文相关的归结规则和转换规则）。 基于Ackermman引理的 方法主要是直接或间接（扩展）使用下面的Ackermann引理得到的。

引理 **1.1** (引理6.1[[47](#_bookmark175)])**.** 给定关系变元*X* 和一阶公式*α*(*x, z*)和*β* (*X* )，其中*x*和*z*为普通变元

构成的多元组、*x*中变元的个数与*X* 的参数个数相同、且*α*中不包括*X*。

∙ 若*β* (*X* )关于*X* 是正的，即：*X* 在*β* (*X* )中的每次出现前面都有偶数个“¬”符号，

则：

∃*X* {∀*x*[*X* (*x*) → *α*(*x, z*)] ∧ *β* (*X* )} ≡ *β* (*X* )*X* (*x*) 。

*α*(*x,z*)

∙ 若*β* (*X* )关于*X* 是负的，即：*X* 在*β* (*X* )中的每次出现前面都有奇数个“¬”符号，

则：

其中，*β* (*X* )*X* (*x*)

*α*(*x,z*)

∃*X* {∀*x*[*α*(*x, z*) → *X* (*x*)] ∧ *β* (*X* )} ≡ *β* (*X* )*X* (*x*) 。

表示将*β* (*X* )中*X* (*x*)的全部出现用*α*(*x, z*)来替换得到的公式。

*α*(*x,z*)

知识遗忘（knowledge forgetting）在模态逻辑S5中首先被提出并被用于推理智能体 的知识状态（知识或者信念）[[48](#_bookmark176)]。 模态逻辑中的遗忘不同于经典逻辑下的遗忘，因为 模态逻辑系统中引入了模态词，此时就不能以简单的谓词（命题）替换的方式获取遗 忘的结果，如：

例 **1.2.** [[49](#_bookmark177)] 令S5公式*ϕ* = K *p* ∧ ¬K*q* ∧ ¬K¬*q*，则使用命题逻辑下的计算方法得到的结果 为*ϕ*[*q/*⊤] ∨ *ϕ*[*q/*⊥] ≡ ⊥。 这显然是不正确的，因为在遗忘*q*之后智能体的知识库不应该 变得不一致。

为此，新的计算方法和四个能精确描述知识遗忘的基本公设被给出，这几个公设 为：削弱（weaking）、 正支持（positive persistence）、 负支持（negative persistence）和 无关性（irrelevance）。 此外，模态谓词逻辑下信息不完备的知识遗忘[[50](#_bookmark178)]、 模态一阶逻 辑S5中的遗忘也得到了研究[[51](#_bookmark179)]，Fang等人讨论了关于多模态（multi-modal）*Kn*、 *Dn*、 *Tn*、 *K*45*n*、 *KD*45*n*情形下遗忘的存在性[[52](#_bookmark180)-[54](#_bookmark181)] ——这些逻辑里的遗忘总是存在的，其 中*n*为智能体的个数。

均匀插值作为遗忘的一个对偶概念，模态逻辑系统种的均匀插值也得到了一些研 究成果：S5、K和KD模态逻辑系统具有均匀插值性质[[55](#_bookmark182)]，而一些模态逻辑系统没有均 匀插值性质，如：量词模态逻辑S5（quantified modal logic S5）[[56](#_bookmark183)]和K4、 和S4及其扩 展都没有均匀插值性质[[57](#_bookmark184)]，因而其遗忘也不是封闭的。 因此，研究这些具有均匀插值 性质的模态逻辑下的遗忘时可以借鉴S5系统下的遗忘方法，也可以参考K系统下的基 于归结计算均匀插值的方法。 对于那些没有均匀插值的模态逻辑系统可以考虑模态逻 辑下的Ackermann引理[[47](#_bookmark175)]。

在非单调推理（non-monotonic reasoning）环境中，科研工作者们也从遗忘需要满 足的基本条件的视角研究了基于回答集语义的逻辑程序的遗忘，这些工作包括Zhang、 Wang等人发表在AI、AAAI和JAIR上的文章[[58](#_bookmark185)-[65](#_bookmark186)]，Eiter、Gonccalves等人的综述[[66](#_bookmark187)-[67](#_bookmark188)]。

遗忘有很多应用，下面列出几点：

∙ 计算后继状态公理：在规划问题中，根据最强必要条件和最弱充分条件有利于求

出后继状态公理[[23](#_bookmark154)]。

∙ 信息隐藏：在某些关键领域，为实现隐私保护，敏感信息必须被隐藏。 现有的方 法包括基于本体[[41](#_bookmark172)]和基于DataSecOps的个人信息保护[[68](#_bookmark189)]，要做到隐私保护，只 需要隐藏（遗忘）那些敏感的概念（concept）和角色（role）符号。 值得注意的 是个人信息保护涉及被遗忘权[[69](#_bookmark190)];

∙ 知识更新：在许多场景，知识不是一层不变的，随着时间或空间的推移，会有新 的知识加入，如何用新加入的信息更新原有知识而保证知识库的一致性是知识更 新需要解决的问题。 此外，知识更新也需要满足一些基本条件，在这些基本条件 中，Katsuno和Mendelzon提出的(*U* 1)-(*U* 8)较为常用；

∙ 提取本体的概要：当一个本体工程师想要快速了解并测试一个本体的内容时，能

事先快速地摒弃许多无关信息并提取出本体的概要是非常有用的；

∙ 知识归并：知识库通常来自于多个信息源，这些分布的信息大多会存在冲突

（不一致），因而不能简单地将他们放在一起。 归并考虑当这些信息矛盾时如何

将它们整合[[70](#_bookmark191)-[74](#_bookmark192)]，基于遗忘的归并可以尽可能少地遗忘原子以保持知识的一致

性[[75](#_bookmark193)]。

∙ 逻辑独立性：生活中许多场景都需要我们判断哪些信息是无关的（或相关的）， 即信息的独立性[[76](#_bookmark194)-[79](#_bookmark195)]，如：知识归并中能知道哪些信息是无关的，那么知识归 并将变得容易一些。 在智能推理中，智能体也需要判断哪些东西（如：命题或 文字）是无关的。 除了无关性，还有其它名字揭示了这一本质[**?** ]，如：独立性

（independence）、 非冗余的（irredundancy）、 非影响的（influenceability）、 分离的

（separability）以及交互性（interactivity）。 基于遗忘的独立性表明被遗忘的原子

（文字、 公式）与得到的结果无关，所以，当遗忘的结果与原公式等价时，该公

式与被遗忘的原子（文字、公式）无关[[75](#_bookmark193)]。

## 1.2.3 SNC和WSC

正如上文所说，WSC（SNC）对于软件工程中系统的形式化验证非常重要。 一般 说来，最强必要条件（SNC）是最一般的推论（the most general consequence），即：命 题成立时能推出的最强的后件（SP），SNC能够蕴涵所有的必要条件；最弱充分条件

（WSC）是最特殊的诱因（the most specific abduction），即：使得命题成立的最弱的前

提条件（WP），WSC能被所有的充分条件蕴涵。

给定一个程序（program）*S*和某一状态（state）的规范（specification）*Q*，若*S*关

于*Q*的WSC是一个能够描述*S*初始状态的规范，那么前提是*S*满足以下两个条件：

（i）*S*必须终止，

（ii）*S*执行完成后必须到达能满足*Q*的状态。

Dijkstra提出了四条规则来计算这样的规范，程序语言里的四种语句（即：赋值语句

（assignment statement）、 顺序语句（sequence statement）、 条件语句（conditional state- ment）和循环语句（loop statement））分别对应了这四种规则[[21](#_bookmark152)]。此外，SNC和WSC还 可用于系统精化[[28](#_bookmark159)]、 模型检测中的负例产生[[27](#_bookmark158)]、 汇编语言程序的推理[[25](#_bookmark156)]和制定验证 条件[[26](#_bookmark157)]。 在上述计算WSC的方法中，有一个必须满足的要求是“*S*必须终止”，这就是 上文中说到的当系统为反应式系统这类不终止系统时现有方法不能计算WSC的主要原 因。

在知识表示与推理中，SNC和WSC为因果理论中后继状态公理的计算提供了一种 方法[[23](#_bookmark154)]，且SNC和WSC都可以用遗忘来计算[[22](#_bookmark153),[80](#_bookmark196)]。随后，SNC和WSC被扩展到FOL下， 且用SOQE实现了SNC和WSC的计算[[80](#_bookmark196)]。

本文将在背景知识部分对SNC和WSC在命题逻辑和模态逻辑这两种情形进行详细

介绍（包括其定义和算法）。

第一章 绪论

**1.3** 研究目标及主要结果

相 关 研 究 工 作 表 明， 现 存 方 法 不 能 求 解 反 应 式 系 统 下 的WSC (SNC)。 然 而， WSC是一种进行系统修改的重要知识，寻求一种有效的求解方法有利于确保系统 的正确性。 在知识表示与推理中，遗忘技术可以用于计算给定理论（公式）的WSC (SNC)。但是，如上所述，时序逻辑下的遗忘理论尚处于不成熟阶段，没有一个统一的 理论框架。 此外，如何用遗忘来计算给定系统模型和性质的WSC (SNC) 也是一个重要 问题。

基于此，本文从遗忘理论的角度出发，研究反应式系统下*SNC*和*WSC*的计算方法， 从而为计算不终止类系统下定义在某个符号集上的*SNC*和*WSC*提供了新的方法，架起 形式化验证与*KR*之间的桥梁。 为了实现这一目标，本文主要研究内容及结果如下：

(1) *CTL*和*µ-*演算的遗忘理论

本 文 探 究 了CTL和*µ*-演 算 中 遗 忘 的 定 义 和 性 质， 特 别 是 其 遗 忘 结 果 的 存 在 性、 复杂性等， 为探索用遗忘计算SNC和WSC奠定论基 础。 具体说来， 遗忘具有 削弱(Weakening, (W))、 正维持(Positive Persistence, (PP))、 负维持(Negative Persistence, (NP))、 无关性(Irrelevance, (IR)) 等基本准则[[48](#_bookmark176)]。 本文探索CTL和*µ*-演算遗忘的以上 四个准则，并探讨其与存在性之间的关系。 此外，本文研究了CTL和*µ*-演算中计算 遗忘结果的方法， 探讨了CTL和*µ*-演算与遗忘相关问题的复杂性结果， 为研究计 算SNC和WSC的性质、算法以及基本准则等作铺垫。 具体说来，有以下两点：

* **CTL**的遗忘理论： CTL不具有Crig插值(Crig interpolation) 性质，因而不具有均匀 插值(uniform interpolation) 性质[[81](#_bookmark197)]，即：存在一个两个公式*ϕ*和*ψ*，若*ϕ* |= *ψ*，不 存在由*ϕ*和*ψ*的公共元素构成的公式*θ* ，使得*ϕ* |= *θ* 且*θ* |= *ψ*。 在这种情况下，本 文除了研究上述CTL中遗忘的性质，还针对CTL子类，特别是能保证其遗忘结果 仍然是CTL可表达的子类的遗忘进行了研究。 在这些子类中，一个特殊的子类 是约束CTL (bunded CTL)：每个公式具有有限个模型，且每一个模型都能用一 个CTL公式表达。 因此，其遗忘是封闭的。

∙ *µ***-**演算的遗忘理论： 与CTL不同，*µ*-演算虽然表达能力比CTL强，其可满足性问 题也比CTL的复杂，但是*µ*-演算具有均匀插值性质[[82](#_bookmark198)]。 这意味着，*µ*-句子“遗 忘”任意原子命题集得到的结果仍然是*µ*-句子。 本文给出了*µ*-演算下遗忘的主 要框架：包括上述遗忘的性质和计算遗忘的方法。 与CTL不同，*µ*-演算公式含 有变元，为此，本文提出一种新的互模拟，并证明*µ*-公式对这种互模拟是不变 的(invariant)。特别地，证明了*µ*-演算遗忘与均匀插值是一个对偶概念，这为研究 均匀插值提供了另一种途径。 此外，本文还证明了当*µ*-公式为析取*µ*-公式时，计 算遗忘可以在多项式时间内完成，这为*µ*-演算遗忘的计算提供了一种有效的方 法。

(2) 计算*CTL*遗忘的算法

基于上述研究结果，设计并使用Prolog实现了一个计算CTL遗忘结果的原型系统， 并从实验角度研究其计算代价。 此外，对于约束CTL的情形，本文也提出了一种基于 模型的算法。

(3) 遗忘理论在反应式系统的形式化验证和知识更新中的应用

反应式系统用Kripke结构表示，Kripke结构是公式的模型。 因而，基于上述研究， 给出了基于遗忘的计算“有限系统模型（Kripke结构）在给定条件下的SNC和WSC” 的方法。 如上所述，一个有限的系统模型能够被一个CTL公式描述，遗忘可以看作以 公式和原子命题为运算对象的函数，因而可以使用遗忘计算其SNC和WSC。 此外，知 识更新是一种使用新发现的性质更新已有理论的技术，本文探讨了如何使用遗忘更 新CTL和*µ*-演算表达的知识。 表明了使用遗忘定义的知识更新满足现有的知识更新的 八条准则。

针对上述几个内容，解决了以下3个关键问题：

(1) *CTL*的遗忘什么情形下存在？如何计算遗忘？

CTL是一种分支时序逻辑，其引入了时态算子，已有文献表明CTL不具有均匀插 值性质。 因为遗忘与均匀插值是一对对偶概念，研究CTL的遗忘不能像已有的经典命 题逻辑和模态逻辑S5那样。 为此，本文深度剖析现存的归结规则，提出了一种基于 归结的计算遗忘的方法：当所有在转换为CTL标准形式过程中引入的新原子命题都被 “消除”时，使用这一方法得到的结果即为遗忘结果。 尽管CTL遗忘不是封闭的，但本 文给出：当被归结的原子命题只同时出现在同一模态词下的命题公式里时，遗忘总是 存在的。

此外，针对约束CTL的遗忘，证明了其遗忘结果可以由有限个模型的特征公式

（一种CTL公式）的析取来表示，所以这种情形下的遗忘是封闭（存在）的。

(2) 遗忘理论与反应式系统的*SNC*和*WSC*的关系

在经典命题逻辑和一阶逻辑中，Lin 和Doherty 等人分别提出了遗忘与SNC (WSC) 的关系[[22](#_bookmark153),[80](#_bookmark196)]。 特别地，经典命题逻辑中的SNC和WSC被用于计算规划问题中的后继状 态公理。 本文给出，给定一个有限反应式系统(Kripke structure)，并将该系统表示为其 特征公式，就可使用上述的CTL和*µ*-演算遗忘计算SNC和WSC。

(3) *CTL*和*µ-*演算的遗忘在推理问题上的复杂性

计算复杂性理论致力于将可计算问题根据它们本身的复杂性分类。 研究表明，在 经典命题逻辑中：CNF (Conjunctive normal form) 公式遗忘的推理问题最难是Π*P*-完全 的，DNF (Disjunctive normal form) 公式遗忘的蕴涵问题是co-NP-完全的。 在命题模态 逻辑S5中，遗忘的模型检测问题是NP-完全的，对应的蕴涵问题是Π*P*-完全的。 基于

2

2

此，本文从现有复杂性结果和自动机理论研究CTL和*µ*-演算遗忘在推理问题上的复 杂性。 研究表明*µ*-演算下关于遗忘的模型检测是EXPTIME的，蕴涵问题的复杂性都 是EXPTIME-完全的。 而CTL下关于遗忘的模型检测：当只考虑特殊段CTLAF （公式中 只包括时态算子AF）时，其模型检测的复杂性为NP-完全的，而有的蕴涵问题的复杂 性在co-NP-完全的和ΠP-完全的之间。

2

**1.4** 论文组组织织织结结构

本文研究了计算树逻辑和*µ*-演算下的遗忘理论，并探讨如何使用遗忘技术来计 算SNC（WSC）和知识更新。 全文共分为九章，组织结构如图[1.2](#_bookmark17)所示，各章节内容的 具体安排如下：

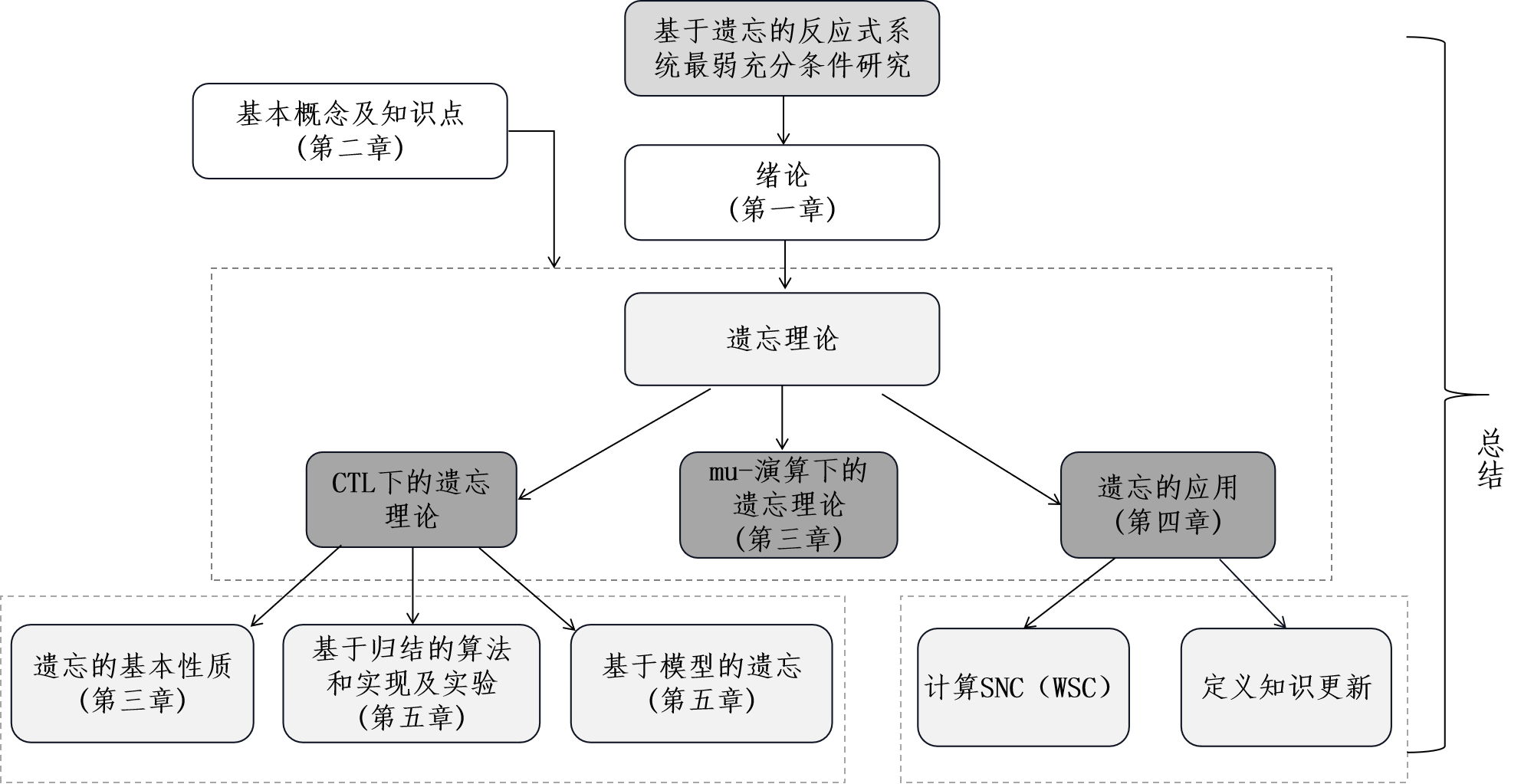


图 1.2: 本文的章节内容组织结构图

第[一](#_bookmark4)章为绪论，首先阐述了本文的研究背景及意义，并分析给出了存在的问题， 凝练出本文研究需要解决的关键问题。 基于上述分析，阐述了本文的研究内容和研究 取得的主要成果。 最后给出了本文的章节组织结构安排。

第[二](#_bookmark18)章为背景知识，介绍了本文研究所涉及逻辑语言的语法语义及相关技术。 首先，给出了经典命题逻辑下解释（赋值）的定义，以及相关语言的模型结构—— Kripke结构。然后，本章给出CTL和*µ*-演算的语法和语义。最后，除了最强必要条件和 最弱充分条件在经典逻辑和模态逻辑S5下的研究，本章还介绍了与上述逻辑语言密切 相关的两个技术：遗忘和CTL下的归结。

第[三](#_bookmark45)章给出*CTL*和*µ-*演算遗忘的定义及其基本性质。 首先，给出一种新的互模

拟定义，并介绍其基本属性；其次，使用互模拟来定义CTL和*µ*-演算下的遗忘，并

研究遗忘的属性，包括表达性属性（representation theory）和复杂性结果等。 本章指 出CTL遗忘是不封闭的，即存在某些公式的遗忘结果不能用CTL来表示。 此外，证明 了*µ*-句子遗忘的结果总是存在的，且不含有不动点操作的这类*µ*-公式遗忘的结果总在 这个类里面。

第[四](#_bookmark82)章为遗忘理论在反应式系统中的应用。 讲述如何将遗忘应用于计算反应式系 统在给定条件下的SNC（WSC）和定义知识更新。 此时，有限状态的反应式系统模型 的WSC（SNC）可以通过遗忘来计算，且通过遗忘定义的知识更新满足Katsuno等人提 出的八条基本准则。

第[五](#_bookmark94)章为*CTL*遗忘的计算方法。 讲述了两种计算CTL遗忘的方法：基于归结和基 于模型的方法。此外，给出了基于归结的算法Prolog实现描述和实验结果分析。在基于 模型的计算CTL遗忘中：第 [5.1](#_bookmark95)节探讨了约束CTL遗忘是封闭的情形；为此，提出了一 种约束的互模拟，并给出给定深度的计算树在给定原子命题集下的特征公式，继而给 出有限情况下Kripke结构在给定原子命题集下的特征公式；最后说明，在约束情形下， CTL公式的遗忘结果总是CTL公式可表达的。 在基于归结的计算CTL遗忘中：首先，

将CTL公式转换为归结规则需要的子句形式——SNF*g*

CTL

子句（后文详细介绍）；其次，

使用归结规则计算要遗忘的原子命题上所有可能的归结结果；随后，移除那些含有要 遗忘的原子命题的子句，并给出一种一般化的Ackermann引理消除一些新引入的原子 命题；最后，将得到的结果转换成CTL公式。 此外，根据上述过程，提出计算CTL下 的遗忘的算法，并分析该算法的时间和空间复杂性。

第[六](#_bookmark129)章为总结与展望。 首先总结了本文的研究工作，进一步，展望了未来研究工

作的方向和重点。

# 第二章 基础知识

本章主要介绍本文用到的符号、 术语以及逻辑理论基础，包括：反应式系统、 *Kripke*结构、 时序逻辑（尤其是计算树逻辑（*CTL*）和*µ-*演算）和遗忘理论等。 首先， 介绍解释时序逻辑语言所需的模型结构，即*Kripke*结构。 其次，介绍时序逻辑中本文 探讨的计算树逻辑和*µ-*演算的语法和语义、 *CTL*归结系统及这些规则的基本性质。 此 外，遗忘理论是本文的研究重点，其概念、 性质及密切相关的逻辑系统种的研究情况 将会被当作本章的重点详细介绍。

本文将命题变量（也叫原子命题）集记作*A* ，*V* ⊆ *A* 是*A* 的子 集。 此外， 规

定*V* 是*V* 在*A* 上的补，也即是*V* = *A* − *V*。

**2.1** 反应式系统

反应式系统是一种不终止的、 与环境有着持续不断交互的系统。 其种类繁多，如 微处理器（microprocessor）、 计算机操作系统（computer operating systems），航空交通 管制系统（air traffic control systems）、 车载电子设备（on-board avionics）以及其它嵌 入式系统。 保证这些系统的正确性至关重要，而系统设计正变成验证系统的同义词。

“*The design of modern information processing systems like digital circuits or protocols is becoming more and more difficult. A large part of the design more and more a synonym for verifying systems.*”

–Klaus Schneider

模型检测是理论计算机科学中最为成功的技术之一，也是反应式系统验证的一 种重要手段，时序逻辑是描述系统规范的语言[[31](#_bookmark162)]。 在模型检测中，反应式系统通常 用Kripe结构表示[[19](#_bookmark150)-[20](#_bookmark151)]。 本文主要探讨规范描述语言和Kripke结构之间的关系，如何将 反应式系统转换为Kripke结构的方法可以参见文章[[83](#_bookmark199)-[85](#_bookmark200)].

下文介绍用于描述反应式系统的Kripke结构和描述反应式系统规范的时序逻辑语

言：CTL和*µ*-演算。

* 1. **Kripke**结构

Kripke结构作为一种描述转换系统（transition system）的数学模型，在理论计算机 科学领域有着广泛的应用，它是解释时序逻辑公式的模型结构，本节将对Kripke结构 做详细介绍。

**2.2.1** 真假赋值和**K-**解释

经典命题语言*L p*由以下三类符号构成：

∙ 命题符号：一般用小写拉丁字母*p*， *q*， *r*， *. . .* 来表示， 且这些命题符号来源

于*A* ；

∙ 联结符号：¬（否定），∧（合取），∨（析取），→（蕴涵），↔（等值于）；

∙ 标点符号：(（左括号），)（右括号）。

*L p*的原子公式集合是*Atom*(*L p*) = *A* ，公式集为*F* (*L p*)。 *ϕ* ∈ *F* (*L p*)当且仅当它

能由（有限次使用）以下的三条规则生成[[86](#_bookmark201)]：

* + - 如果*ϕ* ∈ *Atom*(*L p*)，则*ϕ* ∈ *F* (*L p*)。

∙ 如果*ϕ* ∈ *F* (*L p*)，则(¬*ϕ*) ∈ *F* (*L p*)。

∙ 如果*ϕ*、*ϕ*′ ∈ *F* (*L p*)，则(*ϕ* \* *ϕ*′) ∈ *F* (*L p*)。 其中，\* ∈ {∧*,* ∨*,* →*,* ↔}。

此外，“⊤”和“⊥”和也是原子公式，分别称作“真”和“假”。原子命题及其否定形 式称为文字（literal），有限个文字的析取称为子句（clause），有限个文字的合取称为 项（term）。

例 **2.1.** 下面几个字符串是*L p*的公式：

∙ (*q* ∨ *p*)；

∙ (((¬*p*) ↔ (*q* ∨ *r*)) → (*r* ∧ *p*))。

而字符串*p* ∧ ∨*q*不属于集合*F* (*L p*)。

在本文中，称*L p*的公式为命题公式（在不引起歧义的情况下也称之为公式）。 此 外，规定联结符号的优先级有助于简化公式，即：省略掉冗余的标点符号。 为此，规 定如下符号优先级，且每个左边的联结符号优先于右边的联结符号。

¬ ∧ ∨ → ↔

此时，例 [2.1](#_bookmark22)中的公式(((¬*p*) ↔ (*q* ∨ *r*)) → (*r* ∧ *p*))可写为(¬*p* ↔ *q* ∨ *r*) → *r* ∧ *p*。

定义 **2.1** (真假赋值)**.** 真假赋值是以所有命题符号的集为定义域，以真假值的集{0*,* 1}为

值域的函数*v* : *A* → {0*,* 1}。

后文中也用⊤代表1，⊥代表0（此时真假赋值为*v* : *A* → {⊥*,* ⊤}），且满足对任意真

假赋值*v*，都有 *v*

⊤

⊥

= ⊤和 *v*

= ⊥。 由定义 [2.1](#_bookmark23)可知，一个真假赋值要同时给*A* 中的所有

命题符号指派一个真假值，所以真假赋值的个数为2|*A* |。 真假赋值*v*给公式*ϕ*指派的值

记作*ϕv*，定义如下：

定义 **2.2** (公式的真假值)**.** 真假赋值*v*给公式指派的真假值递归定义如下：

∙ *pv* ∈ {⊥*,* ⊤}，其中*p* ∈ *A* 。

(

∙ (¬*ϕ*)*v* =

⊤*,* 如果*ϕv* = ⊤；

⊥*,* 否则。

(

∙ (*ϕ* ∧ *ψ*)*v* =

(

∙ (*ϕ* ∨ *ψ*)*v* =

⊤*,* 如果*ϕv* = *ψv* = ⊤；

⊥*,* 否则。

⊤*,* 如果*ϕv* = ⊤或*ψv* = ⊤；

⊥*,* 否则。

(

∙ (*ϕ* → *ψ*)*v* =

(

∙ (*ϕ* ↔ *ψ*)*v* =

⊤*,* 如果*ϕv* = ⊥或*ψv* = ⊤；

⊥*,* 否则。

⊤*,* 如果*ϕv* = *ψv*；

⊥*,* 否则。

对任意命题公式*ϕ*和真假赋值*v*，当*ϕv* = ⊤时，称*v*是公式*ϕ*的一个模型，记为*v* |= *ϕ*，读作“*v*满足*ϕ*”。 一般地，若存在一个真假赋值*v*使得*v* |= *ϕ*，则称公式*ϕ*是可满足 的。 如果¬*ϕ*是不可满足的，则称*ϕ*是有效的。 可满足问题（SAT）是一个NP-完全问 题，当公式可满足时，关注如何验证和求解极小模型[[87](#_bookmark202)-[89](#_bookmark203)]；而当公式不可满足时，极 大可满足和极小不可满足就成为研究的重点[[90](#_bookmark204)-[91](#_bookmark205)]。

值得注意的是，一个真假赋值（本文也称作解释）也可以看作是原子命题集*I*，其

表示：属于*I*的原子命题解释为真，反之为假。

模态逻辑在经典逻辑中引进“必然”和“可能”这两种模态词。 如上所述，命题 的真假值只有两种，命题是真的或是假的。 而在模态逻辑中，把命题区分为必然真的 命题和并非必然真的命题，把假命题区分为必然假的和并非必然假的命题。 对于任何 命题*ϕ*，可以有两种模态命题：“*ϕ*是必然的”和“*ϕ*是可能的”。 值得一提的是，时序 逻辑也是模态逻辑的一种[[47](#_bookmark175)]。 尽管如此，在本文中，模态逻辑通常指不带有时序操作 符（参见下文）的情况，时序逻辑特指带有时序操作符的情况。

本文所说的模态逻辑为命题单模态逻辑（propositional mono-modal logic）。 模态公

式集*FM* 是包含“⊤”和“⊥”的满足如下条件的最小集：

* + - *A* ⊆ *FM* ；

∙ 如果*ϕ* ∈ *FM* ，则(¬*ϕ*)，(K*ϕ*) ∈ *FM* ；

∙ 如果*ϕ*，*ψ* ∈ *FM* ，则(*ϕ* \* *ψ*) ∈ *FM* ，其中\* ∈ {∧*,* ∨*,* →*,* ↔}。

令B = ¬K¬，则B*ϕ* ∈ *FM*。其中，K和B叫做模态符号，分别表示“必然”和“可能”。

可能世界语义（或*Kripke*语义）是标准的命题模态逻辑语义[[92](#_bookmark206)]。 Kripke语义定义 在Kripke结构上，一个Kripke结构是一个三元组(*S, R, L*)（下一节中将详细介绍）。其中， *S*是状态的非空集合，*R* ⊆ *S* × *S*是可达性关系。 特别地，如果*R*是一个等价关系（模 态逻辑S5中），那么一个Kripke结构可以写成一个二元组⟨*W, w*⟩，其中*W* 是状态的非空 集合，*w*是*W* 中的元素，每个状态是原子命题集。 此时，称*M* = ⟨*W, w*⟩为一个K-解释

（K-interpretation）[[48](#_bookmark176)]。

定义 **2.3.** 给定一个K*-*解释*M* = ⟨*W, w*⟩，其与*S5*公式的可满足关系归纳定义如下：

∙ *M* |= ⊥，*M* |= ⊤；

∙ *M* |= *p*当且仅当*p* ∈ *w*，其中*p* ∈ *A* ；

∙ *M* |= ¬*ϕ*当且仅当*M* |= *ϕ*；

∙ *M* |= *ϕ* ⊃ *ψ*当且仅当*M* |= *ϕ*或*M* |= *ψ*；

∙ *M* |= K*ϕ*当且仅当∀*w*′ ∈ *W* 有⟨*W, w*′⟩ |= *ϕ*。

*M* = ⟨*W, w*⟩称为公式*ϕ*的K-模型（K-model），当且仅当*M* |= *ϕ*。 此外，如果存在 一个*M* = ⟨*W, w*⟩使得*M* |= *ϕ*， 则称公式*ϕ*是可满足 的。 如果对所有的*M* = ⟨*W, w*⟩， *M* |= *ϕ*都成立，则称*ϕ*是有效的。

## 2.2.2 Kripke结构

给定一个可数无限索引集Ind，一个初始Ind-Kripke结构是一个五元组*M* = (*S, R, L,* [ ]*, s*0)，

其中：

∙ *S*是状态的非空集合，*s*0是*M* 的初始状态（下面详细介绍）；

∙ *R* ⊆ *S* × *S*是状态转换函数，且对任意*s* ∈ *S*，存在*s*′ ∈ *S*使得(*s, s*′) ∈ *R*；

∙ *L* : *S* → 2*A* 是一个标签函数；

∙ [ ] : Ind → 2*S*×*S*是一个函数， 其使得对任意*ind* ∈ Ind， 若*s* ∈ *S*， 则存在唯一一 个*s* ∈ *S*使得(*s, s*′) ∈ [*ind*] ∩ *R*。

Kripke结构*M* = (*S, R, L*)上的路径是*M* 上的状态构成的无限序列*π* = (*s*0*, s*1*, s*2*, . . .* )， 且满足对任意 *j* ≥ 0，(*sj, s j*+1) ∈ *R*。 用*s*′ ∈ *π*表示*s*′是路径*π*上的一个状 态。 特别地， 用*πs*表示以*s*为起点的*M* 上的一条路径，即：*πs* = (*s* = *s*0*, s*1*, . . .* )。 如果对任意*s*′ ∈ *S*， 都存在路径*πs*使得*s*′ ∈ *πs*， 那么称*s*为初始状 态。 Ind-Kripke 结构*M* = (*S, R, L,* [ ])上 的一条索引路径*π*⟨*ind*⟩ （*ind* ∈ Ind）是一条路径(*s* (= *s*)*, s , s , . . .* )， 且对任意 *j* ≥ 0，

*s*

有(*sj, s j*+1) ∈ [*ind*]。

0 1 2

此外，可由初始Ind-Kripke结构定义其它几种特殊的Kripke结构：

∙ 初始Kripke结构*M* = (*S, R, L, s*0)：从初始Ind-Kripke结构*M* 中去掉[ ]元素得到；

∙ Ind-Kripke结构*M* = (*S, R, L,* [ ])：从初始Ind-Kripke结构*M* 中去掉初始状态*s*0得

到；

∙ Kripke结构*M* = (*S, R, L*)：从初始Ind-Kripke结构*M* 中同时去掉[ ]和*s*0得到。

通常一个转换系统（transition system）（包括反应式系统）能够被抽象为一个Kripke结 构[[24](#_bookmark155)]。 一个*(Ind-)*结构是一个二元组*K* = (*M , s*)，其中*M* 是一个初始(Ind-)Kripke结构， *s*是*M* 中的一个状态。 如果*s*是*M* 的初始状态，则称*K* 是初始*(Ind-)*结构。 在这些结构 中，（索引）路径这一概念可以类似地定义。

树是一种只有一个根节点（没有其它节点指向，且可达于其它任意节点的节点） 的无环图。 给定一个初始Kripke结构*M* = (*S, R, L, s*0)和一个状态*s* ∈ *S*，*M* 上以*s*为根节 点、深度为*n*（*n* ≥ 0）的计算树Tr*M* (*s*)递归定义如下[[93](#_bookmark207)]：

*n*

∙ Tr*M* (*s*) 是只有一个节点*s*（其标签为*L*(*s*)）的树。

0

*M*

* Tr

*n*+1

(*s*)是 以*s*为 根 节 点（标 签 为*L*(*s*)） 的 树， 并 且 若(*s, s*′) ∈ *R*， 则*s*有 一 棵 子

树Tr*M* (*s*′)。

*n*

**2.3** 时序逻辑

时序逻辑（也叫时态逻辑）是一种描述系统规范的形式化语言，它研究状态随时 间变化而变化的系统的逻辑特性。 由于软件和硬件系统运行的本质是状态变化的过程， 所以时序逻辑在程序验证和硬件验证中应用得相当广泛。 计算树逻辑（computation tree logic, CTL）是分支时序逻辑的一种，其模型检测是多项式时间可完成的。 然而， CTL表达系统性质的表达能力不如*µ*-演算（*µ*-calculus），如：“某给定的系统中存在一 条路径使得该路径上的第偶数个状态满足特定的性质”这一规范是不能用除了*µ*-演算 以外的其它时序逻辑表示的[[19](#_bookmark150)]。 本节主要介绍CTL和*µ*-演算这两种时序逻辑。

**2.3.1** 计算树逻辑（**CTL**）

计算树逻辑是由Clarke和Emerson等人于1986年提出的一种分支时间时序逻辑[[94](#_bookmark208)]， 它 能 很 好 的 描 述 并 发 系 统 的 一 些 性 质， 包 括： 互 斥 属 性 和 安 全 属 性 等。 此 外， Emerson和Halpern证明了CTL具有小模型属性：如果一个公式是可满足的，那么它在一 个小的有限模型下是可满足的[[95](#_bookmark209)]。 具体说来，对于给定的CTL公式*ϕ*，如果公式的长 度[1](#_bookmark0)为*n*（记为：|*ϕ*| = *n*），则存在一个状态数为*n*8*n*的初始结构(*M , s*0)使得(*M , s*0) |= *ϕ*。

这里给出带索引的CTL公式定义，CTL公式是这种公式的子类。 带索引的CTL的

语言*L* 由下面的几类符号构成：

1给定公式*ϕ*，出现在该公式里的符号的个数为公式的长度，记为|*ϕ*|。

∙ 原子命题集*A* ；

∙ 可数无限索引集合Ind；

* + - 命题常量**start**；

∙ 常量符号：⊤和⊥，分别表示“真”和“假”；

∙ 联结符号：∨和¬，分别表示“析取”和“否定”；

∙ 路径量词：A、E和E*ind* ，分别表示“所有”、“存在”和“存在索引为*ind* ∈ Ind”的

路径；

∙ 时 序 操 作 符： X、 F、 G、 U和W， 分 别 表 示“下 一 个 状 态”、“将 来 某 一 个 状

态”、“将来所有状态”、“直到”和“除非”；

∙ 标点符号：“(”和“)”。

带索引的CTL公式（在不引起歧义时也叫做公式）的时序算子与CTL公式的时序算子 相同，是路径量词和时序操作符的组合（路径量词在前，时序操作符在后），如：AX、 EX、 E*ind* X、 AF等。 此时，语言*L* 的存在范式*(existential normal form, ENF)*可以用巴科 斯范式递归定义如下：

*φ* ::= **start** | ⊥ | *p* | ¬*φ* | *φ* ∨ *φ* | EX*φ* | EG*φ* | E(*φ* U *φ* ) | E⟨*ind*⟩ X*φ* | E⟨*ind*⟩ G*φ* | E⟨*ind*⟩(*φ* U*φ* )

其中，*p* ∈ *A* ，*ind* ∈ Ind。 *L* 中其它形式的公式可以通过如下定义（使用上述定义 中的形式）得到：

*ϕ* ∧ *ψ ϕ* → *ψ*

A(*ϕ* U*ψ*)

A(*ϕ* W*ψ*)

E(*ϕ* W*ψ*)

AF*ϕ* EF*ϕ* AX*ϕ* AG*ϕ*

*de f*

= ¬(¬*ϕ* ∨ ¬*ψ*) (2.1)

*de f*

= ¬*ϕ* ∨ *ψ* (2.2)

*de f*

= ¬E(¬*ψ* U(¬*ϕ* ∧ ¬*ψ*)) ∧ ¬EG¬*ψ* (2.3)

*de f*

= ¬E((*ϕ* ∧ ¬*ψ*)U(¬*ϕ* ∧ ¬*ψ*)) (2.4)

*de f*

= ¬A((*ϕ* ∧ ¬*ψ*)U(¬*ϕ* ∧ ¬*ψ*)) (2.5)

*de f*

= A(⊤U*ψ*) (2.6)

*de f*

= E(⊤U*ψ*) (2.7)

*de f*

= ¬EX¬*ϕ* (2.8)

*de f*

= ¬EF¬*ϕ* (2.9)

没有索引和**start**的公式称为CTL公式。此外，对于给定的公式*ϕ*，其否定范式（negation normal form, NNF）是将否定联结词“¬”的出现通过上述定义变化到只出现在原子命 题之前的形式。

与经典命题逻辑一样，规定联结符号的优先级有助于减少过多括号使用带来的符 号冗余。 带索引的CTL中的联结符号的优先级如下，且左边的联结符号优先于右边的 联结符号：

¬*,* EX*,* EF*,* EG*,* AX*,* AF*,* AG*,* E⟨*ind*⟩ X*,* E⟨*ind*⟩ F*,* E⟨*ind*⟩ G*,* ∧*,* ∨*,* EU*,* AU*,* EW*,* AW*,* E⟨*ind*⟩ U*,* E⟨*ind*⟩ W*,* → *.*

给定一个不包含“→”的公式*ϕ*和原子命题*p*，若在*ϕ*中，*p* 的前面有偶数个否定¬， 则称*p* 在*ϕ* 中的出现为正出现，否则为负出现。 若*ϕ*中所有*p*的出现都为正出现（或负 出现），则称*ϕ*关于*p*是正的（或负的）。 此外，对于给定的公式集，如果该集合中的所 有公式关于*p*都是正的（或负的），则说该集合关于*p*是正的（或负的）。

带索引的CTL的语义定义在Kripke结构上，其递归定义如下。

定义 **2.4** (带索引的CTL的语义)**.** 给定公式*ϕ*，初始*Ind-Kripke*结构*M* = (*S, R, L,* [ ]*, s*0) 和

状态*s* ∈ *S*。(*M , s*)与*ϕ*之间的可满足关系(*M , s*) |= *ϕ*定义如下：

∙ (*M , s*) |= ***start*** 当且仅当*s* = *s*0；

∙ (*M , s*) |= ⊥；

∙ (*M , s*) |= *p* 当且仅当*p* ∈ *L*(*s*)；

∙ (*M , s*) |= *ϕ*1 ∨ *ϕ*2当且仅当(*M , s*) |= *ϕ*1或(*M , s*) |= *ϕ*2；

∙ (*M , s*) |= ¬*ϕ*当且仅当(*M , s*) |= *ϕ*；

∙ (*M , s*) |= EX*ϕ*当且仅当存在*S*中的一个状态*s*1，使得(*s, s*1) ∈ *R*且(*M , s*1) |= *ϕ*；

∙ (*M , s*) |= EG*ϕ*当且仅当存在*M* 上的一条路径*πs* = (*s*1 = *s, s*2*, . . .* )， 使得对每一 个*i* ≥ 1都有(*M , si*) |= *ϕ*；

∙ (*M , s*) |= E(*ϕ* U*ψ*)当且仅当存在*M* 上的一条路径*πs* = (*s*1 = *s, s*2*, . . .* )，使得对某一 个*i* ≥ 1有(*M , si*) |= *ψ*，同时对任意1 ≤ *j < i*，有(*M , s j* ) |= *ϕ*；

*ind*⟩

∙ (*M , s*) |= E⟨*ind*⟩ X*ψ* 当且仅当对索引路劲*π*⟨

*s*

∙ (*M , s*) |= E⟨*ind*⟩ G*ψ* 当且仅当对任意*s*′ ∈ *πs*

⟨*ind*⟩

*ind*⟩

，(*M , s*′) |= *ψ* 且(*s, s*′) ∈ [*ind*]*;*

*,* (*M , s*′) |= *ψ;*

∙ (*M , s*) |= E⟨*ind*⟩(*ψ*1 U*ψ*2) 当且仅当存在*π*⟨

= (*s* = *s , s , . . .* )中的*s*

*(*1 ≤ *j)*使得(*M , s* ) |=

*ind*⟩

*s* 1 2 *j j*

*ψ*2且对任意*sk* ∈ *π*⟨

，若1 ≤ *k < j*，则(*M , s* ) |= *ψ* 。

*s k* 1

与Browne和Bolotov等 人 的 工 作 类 似， 本 文 只 将 初 始Ind-结 构 作 为 模 型 的 候 选 项[[93](#_bookmark207),[96](#_bookmark210)]，即：对于给定的Ind-结构(*M , s*)和带索引的CTL公式*ϕ*，如果(*M , s*) |= *ϕ*且*s* = *s*0，则称(*M , s*)为公式*ϕ*的一个模型。、

令*ϕ*、*ϕ*1和*ϕ*2为公式，这里列出文中出现的一些记号及其含义。

∙ *Mod*(*ϕ*)：公式*ϕ*的所有模型构成的集合；

∙ 可满足：如果*Mod*(*ϕ*) ̸= 0/ ，则称*ϕ*是可满足的；

∙ 逻辑蕴涵：若*Mod*(*ϕ*1) ⊆ *Mod*(*ϕ*2)，则称*ϕ*1逻辑地蕴涵*ϕ*2，记为*ϕ*1 |= *ϕ*2；

∙ 逻辑等值：当*ϕ*1 |= *ϕ*2且*ϕ*2 |= *ϕ*1时，即*Mod*(*ϕ*1) = *Mod*(*ϕ*2)，则称*ϕ*1和*ϕ*2为逻辑等

值公式（简称为等值公式），记作*ϕ*1 ≡ *ϕ*2；

∙ *Var*(*ϕ*)：出现在*ϕ*中的原子命题集。

上述的记号也适用于讨论的对象为公式集的情形。 此外，给定一个公式集Π和一 个初始Ind-结构*K* ，若对于Π中的任意一个公式*ϕ*，都有*K* |= *ϕ*，则*K* |= Π。若Π为公

式的有限集合，则用V Π（八 Π）表示Π中公式的析取（合取），在本文中Π也表示八 Π。

此外，给定公式*ϕ*和原子命题集*V* ，如果存在一个公式*ψ*使得*Var*(*ψ*) ∩ *V* = 0/ 且*ϕ* ≡ *ψ*，那么说*ϕ*与*V* 中的原子命题无关，简称为*V-*无关（*V-irrelevant*），写作IR(*ϕ,V* )。 一 种特殊的形式是*Var*(*ϕ*) ⊆ *V* ，此时称*ϕ*为集合*V* 上的公式。 可以类似定义公式集与原子

命题集的无关性，即：对于公式集合Π中的任意一个公式*ϕ*，如果IR(*ϕ,V* )均成立，那

么Π与*V* 中的原子命题无关，记为IR(Π*,V* )。

CTL公式与结构之间的可满足性与定义 [2.4](#_bookmark27)类似，只是将Ind-结构替换为结构。 除

此之外，相应的符号（记号）也与上述情况类似。

根据上面的定义，以下结论是显然的。

引理 **2.1.** 给定两个公式*ϕ* 和*ψ*，则下列结论成立：

1. AG(*ϕ* ∧ *ψ*) ≡ (AG*ϕ*) ∧ (AG*ψ*)；
2. AGAG(*ϕ*) ≡ AG(*ϕ*)；
3. AG(***start*** → *ϕ*) ≡ *ϕ*。

证明**.** (i) (⇒) 对AG(*ϕ* ∧ *ψ*)的任意模型(*M , s*)

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= *ϕ* ∧ *ψ*

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= *ϕ* 和(*M , si*) |= *ψ*

⇒ (*M , s*) |= (AG*ϕ*) ∧ (AG*ψ*).

(⇐) 对(AG*ϕ*) ∧ (AG*ψ*)的任意模型(*M , s*)

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= *ϕ* 和(*M , si*) |= *ψ*

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= *ϕ* ∧ *ψ*

⇒ (*M , s*) |= AG(*ϕ* ∧ *ψ*)。

(ii) (⇒) 对AGAG(*ϕ*)的任意模型(*M , s*)

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= AG(*ϕ*)

⇒ (*M , s*) |= AG(*ϕ*)。

(⇐) 对AG(*ϕ*)的任意模型(*M , s*)

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= *ϕ*

⇒ ∀*π*′ = (*si, si*+1*, . . .* )，对任意*i* ≥ 0 和 *j* ≥ *i*，有(*M , s j* ) |= *ϕ*

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= AG(*ϕ*)

⇒ (*M , s*) |= AGAG(*ϕ*)。

* 1. (⇒) 对AG(**start** → *ϕ*)的任意模型(*M , s*)

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，有对任意*i* ≥ 0，有(*M , si*) |= **start** → *ϕ*

⇒ (*M , s*) |= *ϕ* （因为(*M , s*) |= **start**）

(⇐) 对*ϕ*的任意模型(*M , s*)

⇒ (*M , s*) |= **start** → *ϕ* （因为(*M , s*) |= **start**）

⇒ (*M , s*′) |= **start** → *ϕ* （因为对任意*s*′ ̸= *s*，(*M , s*′) |= **start**）

⇒ ∀*π* = (*s* = *s*0*, s*1*, . . .* )，对任意*i* ≥ 0，有(*M , si*) |= **start** → *ϕ*

⇒ (*M , s*′) |= AG(**start** → *ϕ*)。

给定两个带索引的CTL公式*ϕ* 和*ψ*，若*ϕ*是可满足的当且仅当*ψ*是可满足的，则

称*ϕ* 和*ψ* 是等价可满足的（*equi-satisfiable*）。 由此可见，相互等价（等值）的公式是等

价可满足的，但是反过来却不成立。 例：公式E 们不是等价的。 所以，可以得到下面的引理。

引理 **2.2.** 令*ϕ*为带索引的*CTL*公式

⟨1⟩

X *p*和E

⟨2⟩

X *p*是等价可满足的，但是它

*(i) ϕ* 和*ϕ*′ 是等价可满足的，其中*ϕ*′ 是由*ϕ*通过用新的索引命名现有的一些索引得到

的公式。

*(ii) ϕ* |= *ϕ*′′，其中*ϕ*′′ 是移除掉*ϕ*中的索引得到的公式。

证明**.** (i) 令*ϕ*′ = *ϕ*[*i/ j*] 为用新的索引 *j* 替换*ϕ*中的索引得到的公式。

(⇒) 对*ϕ*的任意模型(*M , s*)，其中*M* = (*S, R, L,* [ ]*, s*)

⇒ (*M* ′*, s*) ↔0/ (*M , s*) 和(*M* ′*, s*) |= *ϕ*′，其中*M* ′ = (*S, R, L,* [ ]′*, s*) 且对任意*x* ∈ Ind

( [*i*]*,* 若*x* = *j*;

[*x*]′ =

[*x*]*,* 否则。

可以类似地证明对*ϕ*′的任意模型(*M , s*)，存在(*M* ′*, s*) 使得(*M* ′*, s*) ↔0/ (*M , s*) 和(*M* ′*, s*) |=

*ϕ*。

(ii) 这可从带索引的CTL的语义得出。

值得注意的是，引理中(ii)的逆命题不成立。 例如：令*ϕ* = E⟨1⟩ X *p* ∧ E⟨1⟩ X¬*p*，显

然*ϕ*′′ = EX *p* ∧ EX¬*p*是可满足的，但是*ϕ*不可满足。 因此，*ϕ*′′ |= *ϕ*。

**2.3.2 CTL**的标准形形式式

已有结果表明，任意CTL公式能够在多项式时间内被转换为CTL全局子句分离范

式（separated normal form with global clauses for CTL，SNF*g*

CTL

子句）[[97](#_bookmark211)-[98](#_bookmark212)]。SNF*g* 子句

是具有下面几种形式的公式：

CTL

AG(**start** → V*k*

*j*=1

*mj* ) (初始句，initial clause)

AG(⊤ → V*k*

*j*=1

*mj* ) (全局子句，global clause)

八*n*

AG(

*i*=1

*li* → AX V*k*

*mj* ) (A-步子句，A-step clause)

八*n*

AG(

*i*=1

*li* → E

⟨*ind*⟩

V*k*

*j*=1

*j*=1

X

*mj* ) (E-步子句，E-step clause)

八*n*

AG(

*i*=1

*li* → AF*l*) (A-某时子句，A-sometime clause)

八*n*

AG(

*i*=1

*li* → E

⟨*ind*⟩

F*l*) (E-某时子句，E-sometime clause)

其中*k*和*n*都是大于0的常量，**start**是命题常量符号，*li*（1 ≤ *i* ≤ *n*）、*m j*（1 ≤ *j* ≤ *k*）和*l*都

是文字，且*ind* ∈ Ind。从上述标准形式中，可以看到每个SNF*g*

CTL

子句都是AG(*P* → *G*)形

式。 因此在不引起歧义的情况下，下文中使用*P* → *G*指代这些子句。 此外，除了额外

说明，本文通常将SNF*g*

CTL

子句和子句统称为子句。

一个CTL公式*ϕ*可以通过表 [2.1](#_bookmark29)中的规则转换为一个SNF*g*

CTL

子句集，记为*Tϕ*。

表 2.1: 转换规则

*q* → E*T ϕ* ; **Trans(2)** *q* → E(*ϕ*1 U*ϕ*2 ) ; **Trans(3)** *q* → *ϕ*1 ∧ *ϕ*2 ;

**Trans(1)** *q* → E *Tϕ*

*q* → E

(*ϕ* U*ϕ* )

*q* → *ϕ , q* → *ϕ*

⟨*ind*⟩

⟨*ind*⟩ 1 2 1 2

*q* → *ϕ*1 ∨ *ϕ*2 (如果*ϕ*2 不是子句) ; **Trans(5)** *q* → *D* ; *q* →⊥

;

*q* → ⊤

**Trans(4)**

*q* → *ϕ*1

∨ *p, p* → *ϕ*2

⊤ → ¬*q* ∨ *D*

⊤ → ¬*q* {}

**Trans(6)**

*q* → *Q*X*ϕ* (如果*ϕ* 不是子句)

*q* → *Q*X *p, p* → *ϕ*

; **Trans(7)**

*q* → *Q*F*ϕ* (如果*ϕ* 不是文字) ;

*q* → *Q*F *p, p* → *ϕ*

*q* → *Q*(*ϕ*1 U*ϕ*2 ) (如果*ϕ*2 不是文字) ; **Trans(10)** *q* → *Q*G*ϕ* ;

**Trans(8)**

*q* → *Q*(*ϕ* U *p*)*, p* → *ϕ*

*q* → *p, p* → *ϕ, p* → *Q*X *p*

1 2

**Trans(9)**

*q* → *Q*(*ϕ*1 W*ϕ*2 ) (如果*ϕ*2 不是文字) ;

*q* → *Q*(*ϕ* W *p*)*, p* → *ϕ*

1 2

*q* → *Q*(*ϕ* U*l* ) ; **Trans(12)** *q* → *Q*(*ϕ* W*l* )

**Trans(11)** *q* → *l* ∨ *p, p* → *ϕ, p* → *Q*X(*l* ∨ *p*)*, q* → *Q*F*l*

*.*

*q* → *l* ∨ *p, p* → *ϕ, p* → *Q*X(*l* ∨ *p*)

在表 [2.1](#_bookmark29)中，*T* ∈ {X*,* G*,* F}，*ind*是规则中引入的新索引且*Q* ∈ {A*,* E⟨*ind*⟩}；*q*是一个原 子命题, *l*是一个文字, *D*是文字的析取（即子句）, *p*是新的原子命题；*ϕ*，*ϕ*1，和*ϕ*2都 是CTL公式。

规则**Trans(1)**和规则**Trans(2)**为每一个存在路径量词E引入一个新的索引*ind*；规

则**Trans(3)**到规则**Trans(5)**通过引入新的原子命题替换复杂的公式；规则**Trans(6)**到规

则**Trans(12)**用于移除那些不能出现在SNF*g*

CTL

中的时序算子[[99](#_bookmark213)]。

给定一个CTL公式*ϕ*，将其转换为一个SNF*g*

CTL

字句集合的主要步骤如下：

* + 1. 将公式CTL转换为其NNF（negation normal form）形式，记为*nn f* (*ϕ*)；

(2) 使用表 [2.2](#_bookmark30)中的等值公式化简*nn f* (*ϕ*)，得到*simp*(*nn f* (*ϕ*))；

(3) 使用表 [2.1](#_bookmark29)中的规则将{AG(**start** → *z*)*,* AG(*z* → *simp*(*nn f* (*ϕ*)))}化简为SNF*g* 子

CTL

句集合*Tϕ* ，其中*Tϕ* 由导出（*derivation*）序列生成：

*T*0 = {AG(**start** → *p*)*,* AG(*p* → **simp**(**nnf**(*ϕ*)))}*, T*1*, . . . , Tn* = *Tϕ*

使得

**–** *p*是一个新的原子命题，即：*p* ∈*/* {**start**} ∪ *Var*(*ϕ*)；

**–** *Tt*+1 = (*Tt* − {*ψ*}) ∪ *Rt* (*t* ≥ 0)，其中*ψ*为*Tt* 中的非SNF*g*

CTL

子句，且*Rt* 是使用一

条匹配的归则作用到*ψ*上得到的结果集；

**–** *Tn*中的每个公式都是SNF*g*

CTL

子句形式。

表 2.2: CTL化简规则，其中*Q* ∈ {A*,* E}且*T* ∈ {X*,* G*,* F}。

|  |  |  |  |
| --- | --- | --- | --- |
| (*ϕ* ∧ ⊤) → *ϕ*; | (*ϕ* ∧ ⊥) → ⊥; | (*ϕ* ∨ ⊤) → ⊤; | (*ϕ* ∨ ⊥) → *ϕ*; |
| ¬⊤ → ⊥; | ¬⊥ → ⊤; | *QT* ⊥ → ⊥; | *QT* ⊤ → ⊤; |
| *Q*(*ϕ* U⊥) → ⊥; | *Q*(*ϕ* U⊤) → ⊤; | *Q*(⊥U*ϕ*) → *ϕ*; | *Q*(⊤U*ϕ*) → *Q*F*ϕ*; |

*Q*(*ϕ* W⊥) → *Q*G*ϕ* ; *Q*(*ϕ* W⊤) → ⊤; *Q*(⊥W*ϕ* ) → *ϕ* ; *Q*(⊤W*ϕ* ) → ⊤*.*

下文也将SNF*g*

CTL

(*ϕ*)记为由*ϕ*通过转换规则获得的SNF*g*

子句集。

引理 **2.3.** 给定*CTL*公式*ϕ*，

CTL

*(i)* 对*ϕ*中 的 每 一 个 路 径 量 词E， 有 且 仅 有 一 个 新 的 索 引 在 转 换 过 程 中 被 引 入，

即SNF*g*

CTL

(*ϕ*) 中对*ϕ*中的每一个E都有唯一一个带索引的存在路径量词。

1. SNF*g*

CTL

(*ϕ*)中不存在两个E*-*某时子句有相同的索引， 即若*Pi* → E

⟨ *ji*⟩

F*li*

(*i* = 1*,* 2)

在SNF*g*

CTL

(*ϕ*) 中，则 *j*1 ̸= *j*2。

1. 原子命题*p* ∈ *Var*(*ϕ*)不会出现在SNF*g*

CTL

(*ϕ*) 中的子句蕴含式的左手边。

证明**.** (i) 显然规则**Trans(1)** 和**Trans(2)**为每一个E路径量词引入一个新的索引。 此外，

一旦路径量词E被索引标记了，它就不会被其它索引标记。

(ii) 由(i)可知，每个E被唯一的索引标记。 此外在转换过程中不会产生新的E-某时

子句。 因此结论成立。

(iii) 从转换规则容易看出*ϕ*中的原子命题不会出现在SNF*g*

CTL

子句的左边。

值得注意的是，每条转换规则的前件*ϕ*和结果*ψ*分别都是形如AG*ϕ* 和AG*ψ*的公式。

此外，由规则**Trans(11)**可知在SNF*g*

CTL

(*ϕ*)中，E-某时子句和E-步子句可能有相同的索

引。

下面通过一个简单的例子[[98](#_bookmark212)]来展示上述转换步骤：

例 **2.2.** 令*ϕ* = ¬AF *p* ∧ AF(*p* ∧ ⊤)，下面给出将*ϕ*转换为SNF*g*

CTL

的详细步骤。

(1) 将公式*ϕ*转换为其NNF形式：EG¬*p* ∧ AF(*p* ∧ ⊤)；

(2) 化简(1)中的公式为：EG¬*p* ∧ AF *p*；

(3) 使用表 [2.1](#_bookmark29)中的规则转换{AG(**start** → *z*)*,* AG(*z* → (EG¬*p* ∧ AF *p*))}，详细步骤如

下：

* 1. **start** → *z*
  2. *z* → EG¬*p* ∧ AF *p*
  3. *z* → EG¬*p* (2*,* **Trans(3)**)
  4. *z* → AF *p* (2*,* **Trans(3)**)

5*. z* → E⟨1⟩ G¬*p* (3*,* **Trans(1)**)

6*. z* → *x* (5*,* **Trans(10)**)

7*. x* → ¬*l* (5*,* **Trans(10)**)

8*. x* → E⟨1⟩ G*x* (5*,* **Trans(10)**)

9*.* ⊤ → ¬*z* ∨ *x* (6*,* **Trans(5)**)

10*.* ⊤ → ¬*x* ∨ ¬*p* (7*,* **Trans(5)**)

因此，得到的*ϕ*对应的SNF*g*

CTL

子句为：

1*.* **start** → *z* 2*. z* → AF *p* 3*. x* → E⟨1⟩ G*x* 4*.* ⊤ → ¬*z* ∨ *x* 5*.* ⊤ → ¬*x* ∨ ¬*p.*

**2.3.3** *µ***-**演算

*µ*-演算是一种表达能力与S2S[2](#_bookmark0)相同的逻辑语言，线性时序逻辑（linear temporal logic，LTL）、CTL和CTL\*能表达的性质都能用*µ*-演算来表达。 *µ*-演算是模态逻辑的扩 展，构成*µ*-演算语言的符号有：

∙ 原子命题符号集：*A* ；

∙ 变元符号的可数集：*V* ；

∙ 常量符号：⊥和⊤；

∙ 布尔联结符号：∨，∧，→和¬;

2无限完全二叉树下的一元二阶理论（monadic second order theory of the infinite complete binary tree），简称为S2S。

∙ 路径量词符号：A和E;

∙ 时序操作符号：X 用于用于表示“下一个状态”;

∙ 不动点符号：*µ*和*ν*，分别表示“最小不动点”和“最大不动点”。

通常认为AX和EX的优先级比布尔连接符高[[100](#_bookmark214)]，本文规定各类符号之间的优先级

如下（从左到右优先级逐渐变低）：

¬ EX AX ∧ ∨ *µ ν.*

此时，可如下定义*µ*-演算公式（简称为*µ*-公式或公式）：

*ϕ* ::= *p* | *X* | ¬*ϕ* | *ϕ* ∨ *ϕ* | AX*ϕ* | *νX.ϕ*

其中*p* ∈ *A* 且*X* ∈ *V* 。 此外，公式*νX.ϕ*中的*X* 总是正出现在*ϕ*中，即：*ϕ*中*X* 的每一次 出现之前都有偶数个否定符号¬。 AX和EX为时序算子。 称出现在*µX.ϕ*和*νX.ϕ*中的变 元*X* 是受约束的（bound），不受约束的变元称为自由变元。 原子命题和变元符号及其 各自的否定称为文字，出现在公式*ϕ*中的原子命题和变元的集合记为*Var*(*ϕ*)。

其它公式定义如下：

∙ ⊥ =*de f p* ∧ ¬*p*, ⊤ =*de f* ¬⊥,

∙ *ϕ* ∧ *ψ* =*de f* ¬(¬*ϕ* ∨ ¬*ψ*),

* + - *ϕ* → *ψ* =*de f* ¬*ϕ* ∨ *ψ*,
    - EX*ϕ* =*de f* ¬AX¬*ϕ*,

∙ *µX.ϕ*(*X* ) =*de f* ¬*νX.*¬*ϕ*(¬*X* )

其中*p* ∈ *A* ，*ϕ*(*X* )表示*X* 出现在公式*ϕ*中，*ϕ*(¬*X* )是将*ϕ*中*X* 的所有出现同时用¬*X* 替换

得到的公式。

注意：*µ*-演算公式的定义通常考虑动作集*Act*和一组与集合*Act*中的动作*a*相关的模 态词“⟨*a*⟩”[[101](#_bookmark215)-[103](#_bookmark217)]。 为了方便，本文考虑公式里只有一个动作的情形，但是本文的结 论可以扩展到一般的情形。 此时，模态词中的动作*a*可以省略，且公式EX*ϕ* （或AX*ϕ*） 与公式⟨*a*⟩*ϕ*（或[*a*]*ϕ*）[[103](#_bookmark217)]相同。

对于给定的公式*ϕ*，若出现在其中的自由变元与受约束变元不同，且每个变元最

多被约束一次，则称公式*ϕ*是取名恰当的（well-named）。 此外，若公式*δ X.ϕ*(*X* )（*δ* ∈

{*µ, ν*}）中变元*X* 的每次出现都是在EX或AX的辖域[3](#_bookmark0)内，则称变元在公式*δ X.ϕ*(*X* )中是

受保护的（guarded）。一个没有自由变元出现的公式称为*µ-*句子（sentence）。在本文中

3给定公式\**ϕ* （\* ∈ {¬*,* EX*,* AX*,µX, νX* }），则称*ϕ*为\*在公式\**ϕ*中的辖域。 对于公式*ϕ* \* *ψ* （\* ∈ {∨*,* ∧}），则分别

称*ϕ*和*ψ*为它们之间的\*在*ϕ* \* *ψ*中的左辖域和右辖域。

所谈到的公式指的是取名恰当的、受保护的*µ-*句子。

与CTL公式类似，*µ*-演算公式的语义定义在Kripke结构上。 但是，与CTL不同的 是，这里不要求*M* = (*S, R, L, r*)中的*r*为初始状态，且这里的*r*称为根（root），*R*为*S*上的 任意二元关系，但仍然称*M* = (*S, R, L, r*)为初始Kripke结构。

注意：虽然这里的Kripke结构不要求其二元关系是完全的，但是这里的情况更加

一般化，其结论也能推广到二元关系是完全的情形。

定义 **2.5.** 给定*µ-*演算公式*ϕ*、 初始*Kripke*结构*M* 和一个从*V* 中的变量到*M* 中状态的赋 值函数*v* : *V* → 2*S*。 公式在*M* 和*v*上的解释是*S*的一个子集‖*ϕ M* （如果在上下文中*M* 是 明确的，则可以省去上标）：

‖*v*

‖*p*‖*v* = {*s* | *p* ∈ *L*(*s*)}*,*

‖*X* ‖*v* = *v*(*X* )*,*

‖*ϕ*1 ∨ *ϕ*2‖*v* = ‖*ϕ*1‖*v* ∪ ‖*ϕ*2‖*v ,*

‖AX*ϕ*‖*v* = {*s* | ∀*s*′*.*(*s, s*′) ∈ *R* ⇒ *s*′ ∈ ‖*ϕ*‖*v*}*,*

‖*νX.ϕ*‖*v* = ｜

{

*S*′ ⊆ *S* | *S*′ ⊆ ‖*ϕ*‖*v*[*X* :=*S*′]}*.*

其中，*v*[*X* := *S*′]是一个赋值函数，它除了*v*[*X* := *S*′](*X* ) = *S*′之外，和*v*完全相同。 也即

是，对任意*Y* ∈ *V* ：

*v*[*X* := *S*′](*Y* ) =

( *S*′， 若*Y* = *X* ；

*v*(*Y* )， 否则。

直观地，‖*ϕ*‖*M* 为在*M* 和*v*上*ϕ*为真的状态集 合。 称由*M* ， 其赋值*v*和*M* 上的状 态*s*构成的三元组(*M , s, v*)为赋值（当*s*为*M* 的根时，(*M , s, v*)简写为(*M , v*)，也称其为 一个赋值）。在下文中，若*s* ∈ ‖*ϕ*‖*v*，则称*s*“满足”*ϕ*，记为(*M , s, v*) |= *ϕ*。用(*M , v*) |= *ϕ*代替(*M , r, v*) |= *ϕ*。 当公式*ϕ*为*µ*-句子时，可以将赋值函数*v*省略。

*v*

引理 **2.4.** 令*M* = (*S, r, R, L*)、*Ms* = (*S, s, R, L*)、*v* : *V* → 2*S*、*s* ∈ *S*且*ϕ*为*µ-*公式，则：

*M Ms*

*(i)* 对任意*s*1 ∈ *S*，*s*1 ∈ ‖*ϕ*‖*v* 当且仅当*s*1 ∈ ‖*ϕ*‖*v* ；

*(ii)* (*M , s, v*) |= *ϕ*当且仅当(*Ms, v*) |= *ϕ*。

证明**.** (i) 基始**.** (a) *ϕ* = *p*，其中*p* ∈ *A* 。

*s*1 ∈ ‖*ϕ*‖*v*

*M*

⇔ *s*1 ∈ {*s*′ | *p* ∈ *L*(*s*′)}

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

(b) *ϕ* = *X*，其中*X* ∈ *V* 。

*s*1 ∈ ‖*ϕ*‖*M*

*v*

⇔ *s*1 ∈ *v*(*X* )

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

归纳步**.** (a) *ϕ* = ¬*ϕ*1。

*s*1 ∈ ‖*ϕ*‖*M*

*v*

⇔ *s*1 ̸∈ ‖*ϕ*1‖*M*

*v*

⇔ *s*1 ̸∈ ‖*ϕ*1‖*Ms*

*v*

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

1. *ϕ* = *ϕ*1 ∨ *ϕ*2。

*s*1 ∈ ‖*ϕ*‖*M*

*v*

⇔ *s*1 ∈ ‖*ϕ*1‖*M* 或*s*1 ∈ ‖*ϕ*2‖*M*

(归纳假设)

*v* *v*

⇔ *s*1 ∈ ‖*ϕ*1‖*Ms* 或*s*1 ∈ ‖*ϕ*2‖*Ms*

(归纳假设)

*v* *v*

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

1. *ϕ* = AX*ϕ*1。

*s*1 ∈ ‖*ϕ*‖*M*

*v*

⇔ *s*1 ∈ {*s*′ | ∀*s*′′*.*(*s*′*, s*′′) ∈ *R* ⇒ *s*′′ ∈ ‖*ϕ*1‖*M* }

*v*

⇔ *s*1 ∈ {*s*′ | ∀*s*′′*.*(*s*′*, s*′′) ∈ *R* ⇒ *s*′′ ∈ ‖*ϕ*1‖*Ms* } (归纳假设)

*v*

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

1. *ϕ* = *νX.ϕ*1。 *s*1 ∈ ‖*ϕ*‖*M*

*v*

⇔ *s*1 ∈ J{*S*′ ⊆ *S* | *S*′ ⊆ ‖*ϕ*1‖*M* }

*v*[*X* :=*S*′]

*M*

⇔ 存在*S*′ ⊆ *S*，使得*s*1 ∈ *S*′和*S*′ ⊆ ‖*ϕ*1‖*v*[*X* :=*S*′]

⇔ 存在*S*′ ⊆ *S*，使得*s*1 ∈ *S*′和*S*′ ⊆ ‖*ϕ*1‖*Ms*

*v*[*X* :=*S*′]

(归纳假设)

⇔ *s*1 ∈ J{*S* ⊆ *S* | *S* ⊆ ‖*ϕ*1‖*v*[*X* :=*S*′]}

′ ′ *Ms*

⇔ *s*1 ∈ ‖*ϕ*‖*v* 。

*Ms*

(ii) (*M , s, v*) |= *ϕ*

⇔ *s* ∈ ‖*ϕ*‖*M*

*v*

⇔ *s* ∈ ‖*ϕ*‖*Ms*

*v*

⇔ (*Ms, v*) |= *ϕ*。

(i)

因此，只将(*M , v*)作为公式的模型，记*Mod*(*ϕ*)为*ϕ*的模型的集合，即*Mod*(*ϕ*) =

{(*M , v*) | (*M , r, v*) |= *ϕ*}（当*ϕ*为*µ*-句子时，也可简写为*Mod*(*ϕ*) = {*M* | (*M , r, v*) |= *ϕ*}）。

其它记号与CTL情形类似，这里不再赘述。

**2.3.4** *µ***-**公式的析取范式

Janin等人首先提出了*µ*-演算的析取范式[[104](#_bookmark218)]， 后来被逐步完善， 本文使用文

章[[82](#_bookmark198)]中的析取*µ*-公式的定义。

在给出该定义之前， 事先给出*µ*-公式的另一种定义， 称为覆盖*-*语法（cover- syntax）。 该定义是用覆盖操作（cover operator） 的集合替换上述*µ*-公式的定义中 的EX得到。 在覆盖-语法中，

∙ *Cover*(0/ )是公式；

∙ 对任意*n* ≥ 1，若*ϕ*1*,..., ϕn*是公式，则*Cover*(*ϕ*1*,..., ϕn*)是公式。

对于给定的初始结构*M* = (*S, R, L, r*)和赋值函数*v*：

∙ (*M , r, v*) |= *Cover*(0/ )当且仅当*r*没有任何的后继状态；

∙ (*M , s, v*) |= *Cover*(*ϕ*1*,..., ϕn*)当且仅当

**–** 对任意*i* = 1*, ..., n*，存在(*s,t*) ∈ *R*使得(*M ,t, v*) |= *ϕi*;

**–** 对任意(*s,t*) ∈ *R*，存在*i* ∈ {1*,..., n*}使得(*M ,t, v*) |= *ϕi*。

尽管覆盖-语法在形式上与上一小节中*µ*-公式的定义有所不同，但已证明这两种定

义是等价的[[82](#_bookmark198)]，主要因为*Cover*公式与EX公式之间可以通过下面的等式转换：

*Cover*(*ϕ*1*,..., ϕn*) ⇔ EX*ϕ*1 ∧ · · · ∧ EX*ϕn* ∧ AX(*ϕ* ∨ · · · ∨ *ϕn*)*,*

反之，

EX*ϕ* ⇔ *Cover*(*ϕ,* ⊤)*.*

基于此，可以给出析取*µ*-公式的形式定义如下：

定义 **2.6** (析取*µ*-公式[[82](#_bookmark198)])**.** 析取*µ-*公式集*Fd* 是包含⊤、 ⊥和不矛盾的文字的合取且封闭

于下面几条规则的最小集合*:*

*(1)* 析取式（*disjunctions*）：若*α, β* ∈ *Fd* ，则*α* ∨ *β* ∈ *Fd* ；

*(2)* 特殊合取式（*special conjunctions*）：若*ϕ*1*, . . . , ϕn* ∈ *Fd* 且*δ* 为不矛盾的文字的合取，

则*δ* ∧ *Cover*(*ϕ*1*,..., ϕn*) ∈ *Fd* ；

*(3)* 不动点操作（*fixpoint operators*）：若*ϕ* ∈ *Fd* ，且对任意公式*ψ*，*ϕ*不含有形如*X* ∧

*ψ*的子公式，则*µX.ϕ*和*νX.ϕ*都在*Fd* 中。

例 **2.3.** 容 易 检 查*νX.*( *j* ∧ *ch*) ∧ EX(¬ *j* ∧ ¬*ch*) ∧ EXEX*X* 和*νX.*( *j* ∧ *ch*) ∧ AX(¬ *j* ∧ ¬*ch*) ∧

AXAX*X* 都不是析取*µ*-公式。而 *j* ∧ *ch* ∧ EX(¬ *j* ∧ ¬*ch*)、 *µX.*( *j* ∧ *ch*) ∧ EX*X* 和*νX.*( *j* ∧ *ch*) ∧

EXEX*X* 都为析取*µ*公式，因为：

*j* ∧ *ch* ∧ EX(¬ *j* ∧ ¬*ch*) ≡ *j* ∧ *ch* ∧ *Cover*(¬ *j* ∧ ¬*ch,* ⊤)*,*

*µX.*( *j* ∧ *ch*) ∧ EX*X* ≡ *µX.*( *j* ∧ *ch*) ∧ *Cover*(*X,* ⊤)*,*

且

*νX.*( *j* ∧ *ch*) ∧ EXEX*X* ≡ *νX.*( *j* ∧ *ch*) ∧ *Cover*(*Cover*(*X,* ⊤)*,* ⊤)*.*

均匀插值是一个重要的逻辑概念，其有以下含义：给定两个具有*ϕ* |= *ψ*关系的

公式*ϕ*和*ψ*，如果存在公式*θ* ，使得*ϕ* |= *θ*、 *θ* |= *ψ*且*Var*(*θ* ) ⊆ *Var*(*ϕ*) ∩ *Var*(*ψ*)，则称 公式*θ* 是*ϕ*和*ψ*的Craig插值。 若*θ* 与*ψ*无关，而只与*Var*(*ϕ*) ∩ *Var*(*ψ*)有关，则称*θ* 为*ϕ*关 于*Var*(*ϕ*) ∩ *Var*(*ψ*)的均匀插值。 均匀插值的定义[[82](#_bookmark198)]如下：

定义 **2.7.** 给定一个*µ-*句子*ϕ*和集合*V* ⊆ *Var*(*ϕ*)，*ϕ*关于*V* 的均匀插值是满足下列条件

的*µ-*句子*θ* ：

∙ *ϕ* |= *θ* ；

∙ 对任意公式*ψ*，若*Var*(*ϕ*) ∩ *Var*(*ψ*) ⊆ *V* 且*ϕ* |= *ψ*，则*θ* |= *ψ*；

* + - *Var*(*θ* ) ⊆ *V*。

直观上，*ϕ*关于*Var*(*ϕ*) ∩ *Var*(*ψ*)的均匀插值是从*ϕ*中“移除”不在*Var*(*ϕ*) ∩ *Var*(*ψ*)中 的元素，而保留其在*Var*(*ϕ*) ∩ *Var*(*ψ*)上的结论得到的结果。这与遗忘有着密切的关系。

已有结论表明析取*µ*-公式的均匀插值是容易计算的[[82](#_bookmark198)]，即：

定理 **2.1.** 析取*µ-*公式*ϕ*的均匀插值∃*pϕ*与*µ-*公式*ϕ*[*p/*⊤*,* ¬*p/*⊥]等价，其中*ϕ*[*p/*⊤*,* ¬*p/*⊥]是

同时将*p*及其否定¬*p*分别用⊤和⊥替换得到。

**2.4 CTL**下的归结

归结是一种用于判定给定的命题公式（或一阶公式）是否可满足的技术，该技术 可以追溯到1960年Davis等的工作[[105](#_bookmark219)]，之后被Robinson加以完善[[9](#_bookmark140)]。 对于给定的公式， 归结给出一个反驳定理证明过程。

在看见了归结在在命题逻辑和一阶逻辑中取得成就之后，科研工作者们开始将精 力致力于其它非经典逻辑中，并取得了相当显著的理论成果，如：模态逻辑（K系统， Q系统，T系统，S4和S5系统）中的归结[[106](#_bookmark220)]和时序逻辑（尤其是线性时序逻辑（LTL） 和CTL）中的归结[[96](#_bookmark210),[107](#_bookmark221)]。

这里主要介绍与本文直接相关的CTL归结。 CTL归结起源于BolotovF的研究[[96](#_bookmark210)]，

之后被Zhang等人完善[[98](#_bookmark212)]。 不论是在BolotovF的工作还是在Zhang等人的工作中，其关

贵州大学博士学位论文

表 2.3: 归结规则

*P* → A X(*C* ∨ *l* )*, Q* → A X(*D* ∨ ¬*l* ) ; **(SRES2)** *P* → E⟨*ind* ⟩ X(*C* ∨ *l* )*, Q* → A X(*D* ∨ ¬*l* ) ;

**(SRES1)**

*P* ∧ *Q* → AX(*C* ∨ *D*)

*P* ∧ *Q* → E

⟨*ind*⟩

X(*C* ∨ *D*)

*P* → E⟨*ind* ⟩ X(*C* ∨ *l* )*, Q* → E⟨*ind* ⟩ X(*D* ∨ ¬*l* ) ; **(SRES4) start** → *C* ∨ *l ,* **start** → *D* ∨ ¬*l* ;

**(SRES3)**

*P* ∧ *Q* → E

⟨*ind*⟩

X(*C* ∨ *D*)

**start** → *C* ∨ *D*

**(SRES5)** ⊤ → *C* ∨ *l ,* **start** → *D* ∨ ¬*l* ; **(SRES6)** ⊤ → *C* ∨ *l , Q* → A X(*D* ∨ ¬*l* ) ;

**start** → *C* ∨ *D*

*Q* → AX(*C* ∨ *D*)

**(SRES7)** ⊤ → *C* ∨ *l , Q* → E⟨*ind* ⟩ X(*D* ∨ ¬*l* ) ; **(SRES8)** ⊤ → *C* ∨ *l ,* ⊤ → *D* ∨ ¬*l* ;

*Q* → E⟨*ind*⟩ X(*C* ∨ *D*)

八*n*

⊤ → *C* ∨ *D*

八*n*

**(RW1)**

*i*=1 *mi* → AX ⊥; **(RW2)**

*i*=1 *mi* → E⟨*ind*⟩ X ⊥;

V*n*

*m*

*i*=1 ¬

⊤ →

V*n*

*i*=1 ¬

⊤ →

*m*

Λ → E X E G *l Q* → A F¬*l* ; **(ERES2)** Λ → E⟨*ind* ⟩ X E⟨*ind* ⟩ G*l , Q* → E⟨*ind* ⟩ F¬*l*

**(ERES1)**

*,*

*Q* → A(¬ΛW¬*l*)

*Q* → E

⟨*ind*⟩

*.*

(¬ΛW¬*l*)

键点都是将CTL公式转换为一个SNF*g*

CTL

子句集合。 本文使用Zhang等人提出的归结系

统*R*≻*,S* [[98](#_bookmark212)]，如表 [2.3](#_bookmark35)所示。

CTL

在表 [2.3](#_bookmark35)中*P*和*Q*是文字的合取，*C*和*D*是文字的析取，*l*是一个文字，称每条规则横

线下面的公式为横线上面的公式关于文字*l*的归结结果。 此外，Λ = V*n*

八*mi*

*Pi*、 *Pi*是

文字的析取，其中1 ≤ *i* ≤ *n*和1 ≤ *j* ≤ *m*。

*i*=1

*i*=1 *j j*

规则**SRES1-8**称为步*-*归结规则（step resolution rule）、**RW1-2**称为重写规则（rewrite rule）、 **ERES1-2**称为可能归结规则（eventuality resolution rule）。 值得注意的是，规 则**ERES1** 的前提“Λ → EXEG*l*”表示如下子句集ΛEG：

*P*1 1 *n n*

...

1 → \**C*1 *, P*1 → \**C*1 *,*

...

*P*1 1 *n n*

*m*1 → \**Cm*1 *, . . . Pmn* → \**Cmn ,*

其中，对任意*i* (1 ≤ *i* ≤ *n*)，

∙ 存在一个索引*ind* ∈ *I* 使得\* 为空符号或者为{AX*,* E⟨*ind*⟩ X}中的一个，

* + - (八*mi* *i*

*j*=1 *Cj* ） → *l*成立，

* + - (八*mi* *i*

*i*

*n mi* *i*

*j*=1 *Cj* ） → (V =1 八 *j*=1 *Pj* ）成立。

上面的最后两个条件确保了子句集合ΛEG能够蕴涵Λ → EXEG*l*。 规则**ERES2**的第一 个前提与**ERES1**的类似。 **ERES1**的结果能通过表 [2.1](#_bookmark29)中的转换规则转换成等价可满足 的全局和A-步子句集：

*mi*

A → AX(¬*l* ∨ ／ ¬*Pi*) | 1 ≤ *i* ≤ *n*}*,*

{*w*

*j*

¬*l*

*j*=1

第二章 基础知识

*mi*

{⊤ → ¬*Q* ∨ ¬*l* ∨ ／ ¬*Pi* | 1 ≤ *i* ≤ *n*}*,*

*j*

*j*=1

{⊤ → ¬*Q* ∨ ¬*l* ∨ *w*A *, w*A

→ AX(¬*l* ∨ *w*A )}*.*

¬*l* ¬*l* ¬*l*

**ERES2**的结果则通过表 [2.1](#_bookmark29)中的规则转换成等价可满足的全局和E-步子句集：

¬*l* → E⟨*ind*⟩ X(¬*l* ∨

{*wind*

*mi*

*mi*

／

*i*

*j*=1

¬*Pj* ) | 1 ≤ *i* ≤ *n*}*,*

{⊤ → ¬*Q* ∨ ¬*l* ∨ ／ ¬*Pi* | 1 ≤ *i* ≤ *n*}*,*

*j*

*j*=1

{⊤ → ¬*Q* ∨ ¬*l* ∨ *w*A *, wind* → E

X(¬*l* ∨ *wind* )}*.*

¬*l* ¬*l*

⟨*ind*⟩ ¬*l*

注意，在转换**ERES1-2**的结果为子句集合的过程中会引入一个新的原子命题，即*w*A 和*wind* [[98](#_bookmark212)]。

因而，每个归结规则的前件和结果都是子句形式。

¬*l* ¬*l*

对于给定的CTL公式，使用上述的归结规则可以导出一个子句集。 形式化地说，

源于SNF*g*

CTL

子句集合*S*的一个推导（derivation）是一个满足如下条件的SNF*g*

子句集

合的序列*S*0*, S*1*, S*2*, . . . ,*：

CTL

∙ *S*0 = *S*，且

∙ *Si*+1 = *Si* ∪ {*α*} (*i* ≥ 0)，其中*α* ∈*/ Si*是对*Si*的某些子句使用一条归结规则得到的结

果。

*g*

SNF

CTL

子句集合*S*的一个反驳是一个源于*S*的推导*S*0*, S*1*, S*2*, . . . , Si*，且*Si* （*i* ≥ 0）中

包含一个矛盾：公式⊤ → ⊥ 或**start** → ⊥。

为了判定CTL公式*ϕ*的可满足性，基于归结的判定过程用于检查*Tϕ* 是否有反驳存 在。 定理5.6、5.30和6.1[[98](#_bookmark212)]已经证明这一过程是可靠和完备的。 因而，下面的推论显然 成立。

推论 **2.1.** 给定两个*CTL*公式*ϕ*和*ψ*。 则*ϕ* |= *ψ* 当且仅当*Tϕ*∧¬*ψ* 有一个反驳。

例 **2.4** (例 [2.2](#_bookmark31)的扩展)**.** 对例 [2.2](#_bookmark31)中的子句使用表 [2.3](#_bookmark35)中的归结规则，得到如下子句：

1. **start** → *x* (1*,* 4*,* **SRES5**)

(2) *w* → AX(*p* ∨ ¬*x*) (2*,* 3*,* 5*,* **ERES1**)

(3) ⊤ → ¬*z* ∨ *p* ∨ ¬*x* (2*,* 3*,* 5*,* **ERES1**)

(4) ⊤ → ¬*z* ∨ *l* ∨ *w* (2*,* 3*,* 5*,* **ERES1**)

(5) *w* → AX(*x* ∨ *w*) (2*,* 3*,* 5*,* **ERES1**)

(6) ⊤ → ¬*z* ∨ ¬*x* (5*,* (3)*,* **SRES8**)

(7) **start** → ¬*x* (1*,* (6)*,* **SRES5**)

(8) **start** → ⊥ ((1)*,* (7)*,* **SRES4**)*.*

由于在这一推导中有一个子句集合包含一个矛盾，即：**start** → ⊥，所以*Tϕ* 存在一个反

驳。 因此，*ϕ*是不可满足的。

**2.5** 遗忘理论和**SNC**（**WSC**）

这部分主要介绍遗忘理在经典逻辑和模态逻辑S5下的定义，以及基于遗忘的SNC

（WSC）计算方法。

**2.5.1** 经典逻辑的遗忘

遗忘一词起源于经典逻辑（包括命题逻辑和一阶逻辑） [[37](#_bookmark168)]，给定一个命题公 式*ϕ*和一个原子命题*p*，下面介绍如何从*ϕ*中遗忘（forget）掉*p*。 在第[1.2.2](#_bookmark13)节中说过， 从*ϕ*中遗忘掉*p* 得到的结果为*Forget*(*ϕ,* {*p*}) ≡ *ϕ*[*p/*⊤] ∨ *ϕ*[*p/*⊥]。

例 **2.5.** 某学校有*a*和*b*两个食堂，学生只能去*a*或去*b*食堂吃饭，如果想吃烤鱼（fish， *f* ） 就去*a*食堂吃饭，如果想吃炒饭（rice，*r*）就去*b*食堂吃饭。 这一知识可表示为命题公 式*ϕ* = (*a* ∨ *b*) ∧ ( *f* → *a*) ∧ (*r* → *b*)。 如果此时不考虑鱼，即：由于某种原因*a*食堂就不再 卖烤鱼了，此时就应该“遗忘”烤鱼（ *f* ）。 这一计算过程表示如下：

*Forget*(*ϕ,* { *f* }) ≡ *ϕ*[ *f /*⊤] ∨ *ϕ*[ *f /*⊥]

≡ [(*a* ∨ *b*) ∧ (⊤ → *a*) ∧ (*r* → *b*)] ∨ [(*a* ∨ *b*) ∧ (⊥ → *a*) ∧ (*r* → *b*)]

≡ [(*a* ∨ *b*) ∧ *a* ∧ (*r* → *b*)] ∨ [(*a* ∨ *b*) ∧ (*r* → *b*)]

≡ (*a* ∨ *b*) ∧ (*r* → *b*)*.*

直观上来看，这个结果应该比原始公式*ϕ*还要弱，但是能够蕴含同样的任何不包 含 *f* 的句子（sentence），也就是说遗忘只能影响与 *f* 相关的语义。 这一性质可由互模拟 这一词来表示。 解释之间的互模拟为：给定原子命题*p*，如果对任意*q* ∈ *A* − {*p*}有*q* ∈ *I*1当且仅当*q* ∈ *I*2，则称解释*I*1与*I*2是*p*互模拟的，记为：*I*1 ∼*p I*2。

在一阶逻辑中，一阶逻辑语言*Lf* 的解释有两种：*Lf* 和结构有联系或没有联系。 这 里考虑和结构有联系的情形，一个一阶结构由论域（domain）、 指定的个体、 关系和 函数构成。 此时，*Lf* 中的个体符合、 *n*-元关系符号和*m*-元函数符号分别被解释为这个 结构中指定论域中的个体、 论域上的*n*-元关系和*m*-元全函数（即处处有定义的函数）。

对于给定的一阶结构*M*和*X* ∈ {个体符号，元组，关系符号，函数符号}，*M*[*X* ]表示结

构*M*对*X* 的解释，且*M*[(*a*1*, a*2*, . . . , ai*)] = (*M*[*a*1]*, M*[*a*2]*, . . . , M*[*ai*])。

给定实例化（ground atom）原子*P*(*\_t*)（*\_t*是一个*n*元组）、*M*1和*M*2为一阶结构，则*M*1 ∼*P*(*\_t*)

*M*2，当且仅当除了*P*(*\_t*)的真值，*M*1和*M*2相同，即：

(i) *M*1和*M*2有相同的论域，且每个函数符号被解释成相同的函数；

(ii) 对于和*P*不同的任意关系符号*Q*，*M*1[*Q*] = *M*2[*Q*];

(iii) 令*\_u* = *M*1[*\_t*]， 则对于该论域中任意 与*\_u*不 同的元组*d\_*， *d\_* ∈ *M*1[*P*] 当且仅当*d\_* ∈

*M*2[*P*]。

一阶逻辑中遗忘实例化原子的形式化定义[[37](#_bookmark168)]为：

定义 **2.8.** 给定一个句子（*sentence*）*ϕ*和实例化原子*p*，*ϕ*′是从*ϕ*中遗忘掉*p*的结果当且 仅当对任意结构*M*，*M*是*ϕ*′的模型当且仅当存在一个*ϕ*的模型*M*′，使得*M* ∼*p M*′。

从句子*ϕ*中遗忘实例化原子*P*(*\_t*)比命题逻辑下的遗忘多了一步，即事先将*ϕ*中的所

有*P*(*\_t*′)的出现用(*\_t* =*\_t*′ ∧ *P*(*\_t*)) ∨ (*\_t* ̸=*\_t*′ ∧ *P*(*\_t*′))来替换，并且将这一结果记为*ϕ*[*P*(*\_t*)]。

例 **2.6.** 令*ϕ* = *J*(*mo*) ∨ *J*( *f a*) ∨ *B*(*sm*)、 *p* = *J*(*mo*)，则：

*ϕ*[*p*] ≡ (*mo* = *mo* ∧ *J*(*mo*)) ∨ (*mo* ̸= *mo* ∧ *J*(*mo*))∨

(*mo* = *f a* ∧ *J*(*mo*)) ∨ (*mo* ̸= *f a* ∧ *J*( *f a*)) ∨ *B*(*sm*)*.*

*Forget*(*ϕ, p*) ≡ *ϕ*[*p*][*p/*⊤] ∨ *ϕ*[*p*][*p/*⊥]

≡ (*mo* = *mo* ∧ ⊤) ∨ (*mo* ̸= *mo* ∧ ⊤) ∨ (*mo* = *f a* ∧ ⊤) ∨ (*mo* ̸= *f a* ∧ *J*( *f a*)) ∨ *B*(*sm*)

∨ (*mo* = *mo* ∧ ⊥) ∨ (*mo* ̸= *mo* ∧ ⊥) ∨ (*mo* = *f a* ∧ ⊥) ∨ (*mo* ̸= *f a* ∧ *J*( *f a*)) ∨ *B*(*sm*)

≡ (*mo* = *mo*) ∨ (*mo* ̸= *mo*) ∨ (*mo* = *f a*) ∨ (*mo* ̸= *f a* ∧ *J*( *f a*)) ∨ *B*(*sm*)*.*

然而，遗忘一整个关系（谓词）“*P*”而不是其实例得到的结果是一个二阶公式， 且结构间在谓词上的互模拟与上述在实例上的有所不同：对于谓词*P*和结构*M*1、 *M*2， *M*1 ∼*P M*2 当且仅当：

(i) *M*1和*M*2有相同的论域，且每个函数符号被解释成相同的函数；

(ii) 对于和*P*不同的任意关系符号*Q*，*M*1[*Q*] = *M*2[*Q*]。

也即是排除了实例情形下的第三个条件，因为此时考虑的是整个谓词。 而遗忘谓词的

定义与遗忘实例的定义类似，只是将*M* ∼*p M*′变为*M* ∼*P M*′。

研究表明，从句子*ϕ*中遗忘谓词*P*的结果为*Forget*(*ϕ, P*) = (∃*R*)*ϕ*[*P/R*][[37](#_bookmark168)]，其中*P*是*n*- 元谓词，*R*是*n*-元谓词变量。 正如前文所说，一阶逻辑遗忘不是封闭的，此时不一定能 找到一个与(∃*R*)*ϕ*[*P/R*]等价的一阶公式。

本文采用基于归结的方法来计算CTL中的遗忘，因此，这里给出命题逻辑下基于

归结的遗忘定义[[108](#_bookmark222)]。

定义 **2.9.** 给定命题公式*ϕ*和原子命题*p*，

*Forget*(*ϕ, p*) = {*C* ∈ *CNF*(*ϕ*) | *p* 不出现在*C*中} ∪ *Res*(*CNF*(*ϕ*)*, p*)

其中*CNF*(*ϕ*)表示形成*ϕ*的合取范式的子句构成的集合，*Res*(*S, p*) = {*C*1 ∨*C*2 | *C*1 ∨ *p,C*2 ∨

¬*p* ∈ *S*}。

从定义[2.9](#_bookmark38)不难看出计算从*ϕ*中遗忘*p*的结果可以分为三个步骤：

(1) 计算*ϕ*的合取范式，并得到*CNF*(*ϕ*)；

1. 计算*Res*(*CNF*(*ϕ*)*, p*)；

(3) 去除*CNF*(*ϕ*)包含*p*的子句。

遗忘的定义种类很多，本文的定义采用的是上述所说的互模拟方式，因此这里不再赘

述其它定义，感兴趣的读者可以参考Eiter的文章[[66](#_bookmark187)]。

在描述逻辑中，如果遗忘的结果可以用当前讨论的描述逻辑来表示，则该结果就 是一个均匀插值。 而判定均匀插值是否存在通常是很复杂的，如：在*A L C* 和*E L* 中 是双指数时间的。 因此，描述逻辑遗忘通常也是很困难的。 尽管如此，也有很多方法 克服这些问题，其中扩展描述语言（如：从*A L C* 到*A L C v* [[109](#_bookmark223)]）或引入新的辅助符 号[[110](#_bookmark224)]是常用的方法。 一些计算遗忘的工具是：基于skolem化和SOQE的SCAN [4](#_bookmark0)、 基于 归结的Lethe[[111](#_bookmark225)]和基于Ackermann引理的FAME[[112](#_bookmark226)]。

**2.5.2** 模态逻辑**S5**的遗忘

由于时序逻辑是模态逻辑的一种，其语义是Kripke语义，这里介绍与其密切相关

且基础的模态逻辑S5遗忘。 与经典逻辑遗忘相似，S5遗忘也用互模拟来定义。

原子命题集*w*1和*w*2是*V* -互模拟的，当且仅当*w*1 − *V* = *w*2 − *V* ，记为*w*1 ∼*V w*2，其 中*w*1*, w*2*,V* ⊆ *A* 。 给定原子命题集*V* ⊆ *A* 、 两个K-解释*M* = ⟨*W, w*⟩和*M* ′ = ⟨*W, s*⟩，则 称*M* 和*M* ′是*V* -互模拟的（记为*M* ↔*V M* ′） [[49](#_bookmark177)]， 当且仅当存在一个二元关系*σ* ⊆ *W* × *W* ′使得(*w, w*′) ∈ *σ* ，且：

4<http://www.mettel-prover.org/scan/index.html>

(i) ∀*w*1 ∈ *W* ，∃*w*2 ∈ *W* ′使得(*w*1*, w*2) ∈ *σ* ；

(ii) ∀*w*2 ∈ *W* ′，∃*w*1 ∈ *W* ′使得(*w*1*, w*2) ∈ *σ* ；

(iii) 若(*w*1*, w*2) ∈ *σ* ，则*w*1 ∼*V w*2。

条件(i)和(ii)分别称为前向条件（forth condition）和后向条件（back condition）。 值 得注意的是，即使*M*和*M*′有*V* -互模拟关系，*M*和*M*′也可能有不同数量的世界个数。 除 此之外，从定义中不难看出，如果*M* ↔*V M* ′，则有*Atom*(*W* ) − *V* = *Atom*(*W* ′) − *V* ，其 中*Atom*(*X* )（*X* 是可能世界的集合）是由出现在*X* 中的世界中的原子构成的集合。 从定 义中还可得出↔*V* 是一个等价关系。

S5关于*V* -互模拟是不变的：如果两个K-解释*M*和*M*′有*V* -互模拟关系，那么对于任

何不包含*V* 中任何原子的公式*ϕ*，*M*和*M*′同时满足或不满足公式*ϕ*。 S5中的知识遗忘

（knowledge forgetting）定义如下[[49](#_bookmark177)]：

定义 **2.10** (knowledge forgetting)**.** 给定模态*S5*公式*ϕ*和*V* ⊆ *A* 。如果下面的等式成立，则

称知识集*KForget*(*ϕ,V* )是从*ϕ*遗忘*V* 得到的结果：

*Mod*(*KForget*(*ϕ,V* )) = {*M* ′ | ∃*M* ∈ *Mod*(*ϕ*)*, M* ↔*V M* ′}*.*

Zhang等人还提出了能精确描述知识遗忘的四个基本条件（公设），给定两个公

式*ϕ*和*ϕ*′ = *KForget*(*ϕ,V* )，*V* ⊆ *A* 是原子命题集。 知识遗忘满足以下性质：

(**W**) 削弱（Weaking）: *ϕ* |= *ϕ*′；

(**NP**) 正支持（Positive Persistence）：如果IR(*φ ,V* )并且*ϕ* |= *φ* ，则*ϕ*′ |= *φ* ； (**PP**) 负支持（Negative Persistence）：如果IR(*φ ,V* )并且*ϕ* |= *φ* ，则*ϕ*′ |= *φ* ； (**IR**) 无关性（Irrelevance）：IR(*ϕ*′*,V* )。

直观上，(**W**)和(**IR**)表明“遗忘”削弱了公式*ϕ*且得到的结果与*V* 无关，(**PP**)和(**NP**)表 明对任意与*V* 无关的公式*φ* ，*ϕ* |= *φ* 当且仅当*ϕ*′ |= *φ*。 总而言之，遗忘得到的结果能推 出所有与*V* 无关且能被*ϕ*推出的结果，但不能推出所有与*V* 无关且不能被*ϕ*推出的结果。 从数据库和安全的层面讲，遗忘相当于从已有的关系表中构建出一个视图，达到了隐 私保护的作用。

这四个性质与知识遗忘的关系如下所述[[49](#_bookmark177)]：

定理 **2.2.** 给定公式*ϕ*和*ϕ*′，*V* ⊆ *A* 为原子命题集。 下面的陈述是等价的：

* 1. *ϕ*′ ≡ *KForget*(*ϕ,V* )*,*
  2. *ϕ*′ ≡ {*φ* | *ϕ* |= *φ* 且*IR*(*φ,V* )}*,*

*(iii)* 若*ϕ*、*ϕ*′和*V* 为*(i)*和*(ii)*中提到的符号，则公设*(****W****)*、*(****PP****)*、*(****NP****)*和*(****IR****)*成立*.*

在本文中也将说明CTL和*µ*-演算的遗忘也有上述性质。

任意S5公式都能转换为与之等价的模态合取范式（MCNF）[[113](#_bookmark227)]，其模态子句形式

为：

*C*0 ∨ K*C*1 ∨ · · · ∨ K*Cn*−1 ∨ B*Cn,*

或具有如下形式的公式的析取——模态析取范式（MDNF）[[49](#_bookmark177),[51](#_bookmark179)]:

*ϕ*0 ∧ K*ϕ*1 ∧ B*ϕ*2 ∧ · · · ∧ B*ϕn* (2.10)

其中*ϕi* （0 ≤ *i* ≤ *n*）为命题逻辑公式，*Ci* （0 ≤ *i* ≤ *n* − 1）为经典子句，*Cn*为CNF公式， 且任意*ϕi*和*Ci*都可能缺失。 从子句[2.10](#_bookmark40)遗忘掉原子命题*p*可以转换成命题逻辑中的遗忘， 即：

*KForget*(*ϕ*0 ∧ K*ϕ*1 ∧ B*ϕ*2 ∧ · · · ∧ B*ϕn*)

*n*

≡ *Forget*(*ϕ*0*,* {*p*}) ∧ K(*Forget*(*ϕ*1*,* {*p*})) ∧ 八 B(*Forget*(*ϕ*1 ∧ *ϕi,* {*p*}))*.*

*i*=1

由此可以得出任意S5公式遗忘都能转换为命题逻辑公式的遗忘，而命题逻辑遗忘已有

算法和实现，这将在计算SNC和WSC部分给出。

**2.5.3** 遗忘的计算方法

在第 [2.5.1](#_bookmark37)和[2.5.2](#_bookmark39)小节中详细介绍了经典逻辑和模态逻辑S5下遗忘的定义和一些直 接的计算方法。 总的来说，这些方法分为两类：“代替”的方法和归结的方法。 其中 “代替”法是将要遗忘的原子命题在公式里用“⊤”或“⊥”代替，归结的方法主要使 用归结规则来“消除”需要遗忘的原子命题。 然而，上文中并没有对归结方法进行详 细的介绍，所以，这部分给出命题情形下归结方法的详细描述。

归结方法取决于子句的形式，子句的形式决定了归结规则的复杂性。 经典命题逻 辑中的子句形式比较单一，就只有一种——文字的析取，因此归结规则就比较简单， 即：

*C*1 ∨ *p* *C*2 ∨ ¬*p,*

*C*1 ∨ *C*2

其中*C*1和*C*2是子句， *p*是原子命题。 在这种情况下，基于归结的方法就如定义 [2.9](#_bookmark38)那样 简单。 在一阶逻辑中，将公式转换为子句形式的过程比较复杂，而归结规则也相对复 杂一些。 但是在一阶情形下的归结系统R[[47](#_bookmark175)]是可靠的且归结反驳是完备的。

上一节已经说明任意S5公式能够转化成模态子句*C*0 ∨ K*C*1 ∨ · · · ∨ K*Cn*−1 ∨ B*Cn*的合

取，因此，S5的归结系统RS5[[106](#_bookmark220)]如下：

*C* ∨ K(*l* ∨ *D*) *C*′ ∨ B(¬*l* ∨ *D*′*, E*)

*C* ∨ K⊥

(KB)

*C C*′

′ ′ ; (K⊥) ;

∨ ∨ B(*D* ∨ *D ,* ¬*l* ∨ *D , E*) *C*

*C* ∨ K(*l* ∨ *D*) *C*′ ∨ K(¬*l* ∨ *D*′)

*C* ∨ B(⊥*, E* )

(KK)

*C* ∨ *C*′

∨ K(*D* ∨ *D*′)

; (B⊥) *C* ;

*C* ∨ K(*l* ∨ *D*) *C*′ ∨ ¬*l*

*C* ∨ *l C*′ ∨ ¬*l*

(K) *C*

∨ *C*′ ∨ *D*

; (*Clas*)

;

*C* ∨ *C*′

*C* ∨ B(*l* ∨ *D,* ¬ ∨ *D*′*, E*)

*E* [*D* ∨ *D* ∨ *C*])

(B)

*C*

∨

B(*D* ∨

*D*′*, l*

∨ *D,* ¬ ∨

; (*Fact*)

*D*′*, E*)

*E*[*D*

*.*

∨ *C*]

其中*l*为文字，*C*、*C*′、*D*、*D*′为子句，*E*为子句集；对于子句集*S*，B(*S*)表示B(八 *S*)；

*E*[*ψ*]表示*ψ*是*E*的子公式。

基于上述归结系统RS5，提出计算遗忘的算法。 为了更清楚地描述该算法，这里

还需要介绍两个概念：模态子句的包蕴（subsume）和清除（suppressing）。给定两个模

态子句*C* = *C*0 ∨ K*C*1 ∨ · · · ∨ K*Cn*−1 ∨ B*Cn*和*C*′ = *C*′ ∨ K*C*′ ∨ · · · ∨ K*C*′

∨ B*C*′ ，如果满足下

面三个条件，则说*C*包蕴*C*′：

0 1 *m*−1 *m*

∙ *C*0包蕴*C*′ ，即*Lit*(*C*) ⊆ *Lit*(*C*′)；

0

∙ ∀*Ci*（1 ≤ *i* ≤ *n* − 1），∃*C*′ （1 ≤ *j* ≤ *m* − 1）使得*Ci*包蕴*Cj* ；

*j*

∙ 对*C*′ 中的任意合取项*e*′，存在*Cn*中的一个合取项*e*使得*e*包蕴*e*′。

*m*

其中*Lit*(*X* )为出现在*X* 中文字的集合。

“清除”操作主要是用于移除那些包含要遗忘的原子命题的公式。具体地，令*φ* 为

子句，*V* ⊆ *A* 为原子命题集，*φ* 在*V* 上的清除操作记为*Supp*(*V, φ* )，且：

(

*Supp*(*V, φ* ) =

⊤*,* 若存在*V* 中的元素*p*使得*p* ∈ *Var*(*φ* )；

*φ,* 否则。

令*φ* = *C*0 ∨K*C*1 ∨···∨K*Cn*−1 ∨B*Cn*为模态子句，*φ* 在*V* 上的清除操作也记为*Supp*(*V, φ* )，

且：

／

*Supp*(*V,C*0) ∨ ／

＼

K*Supp*(*V,Ci*)



∨ B  八



*Supp*(*V, α*) *.*

1≤*i*≤*n*−1

*α* is a conjunct of *Cn*

模态S5下基于归结的算法如算法 [2.1](#_bookmark0)所示。 在该算法中，第7-9行用于移除具有形 式*p* ∨ *D*或¬*p* ∨ *D*（*p* ∈ *V* ）的子句，以免产生无用的结果，因为这些结果在第11行也会 被移除。

算法 **2.1** S5下基于归结的遗忘计算

输入**:**

Γ*,V* ： S5公式，原子命题集

输出**:**

*KForget*(Γ*,V* )：从Γ中遗忘掉*V* 中原子的结果

1: 将Γ转换为模态子句集Γ′；

2: Γ2 = {*C* | *C* ∈ Γ′*, Var*(*C*) ∩ *V* = 0/ }，Γ1 = Γ′ − Γ2

3: **if** *V* = 0/ **then**

4: 跳转到11;

5: **end if**

6: 从*V* 中随机选择一个原子*p*，且令*V* = *V* − {*p*};

7: 化简Γ1（*C*1*,C*′ ∈ Γ）：

8: 若*C*′包蕴*C*1，则从Γ1中删除*C*1；

9: 若*C*1形如*p* ∨ *D*或¬*p* ∨ *D*，则从Γ1中删除*C*1（*D*为模态子句）

10: 跳转到2;

11: Γ3 = {*Supp*(*V, φ* ) | *φ* ∈ Γ1};

12: **return** Γ ∪ Γ3 .

**2.5.4** 基于遗忘的**SNC**（**WSC**）计算

SNC和WSC的定义最先由Lin提出[[22](#_bookmark153)]，这部分给出其在命题逻辑和一阶逻辑下的

形式化定义和计算方法。

定义 **2.11.** 令*ϕ*是一个命题公式，*V* ⊆ *ϕ*，*q*是一个出现在*ϕ*中，但是不出现在*V* 中的命 题。 对于*V* 上的公式*φ* ，若*ϕ* |= *q* → *φ* （*ϕ* |= *φ* → *q*），则称公式*φ* 是*q*在*V* 和*ϕ*上的必要条 件（充分条件）。 如果对于任意*q*在*V* 和*ϕ*上的必要条件（充分条件）*φ* ′都有*ϕ* |= *φ* → *φ* ′

（*ϕ* |= *φ* ′ → *φ* ），则称*φ* 是*q*在*V* 和*ϕ*上的最强必要条件（最弱充分条件）。

SNC和WSC具有对偶关系，且任意公式的SNC（WSC）都能转换成原子命题的形 式计算[[22](#_bookmark153)]。 因此，这里只讨论原子命题情形下SNC（WSC）的定义及其计算。

定理 **2.3.** 给定命题公式*ϕ*、原子命题集*V* ⊆ *Var*(*ϕ*)和原子命题*q* ∈ (*Var*(*ϕ*) − *V* )。 令*V* ′ =

*Var*(*ϕ*) − (*V* ∪ {*q*})，则

* *q*在*V* 和*ϕ*上的*SNC*是*Forget*(*ϕ*[*q/*⊤]*,V* ′)；
* *q*在*V* 和*ϕ*上的*WSC*是¬*Forget*(*ϕ*[*q/*⊥]*,V* ′)。

定理 [2.3](#_bookmark43)表明可以用遗忘计算SNC和WSC。基于遗忘的计算SNC（WSC）的详细算 法如算法 [2.2](#_bookmark0)所示，其中一个子句集合的极小集（minimal set of clauses）为满足下面性 质的集合：

∙ 所有的单元子句都被替换为⊤；

∙ 没有一个子句被集合中的另一个子句包蕴。

此外，对于公式集合*S*，*S*[*X/Y* ]为将*S*中每个公式中*X* 的出现全都替换成*Y* ，即*S*[*X/Y* ] =

{*ϕ*[*X/Y* ] | *ϕ* ∈ *S*}。

算法 **2.2** 命题逻辑下基于遗忘的SNC计算

输入**:**

Γ*,V, q*：子句集合，原子命题集，出现在Γ且不出现在*V* 中的原子命题

输出**:**

*φ* ： *V* 上的公式（*φ* 是*q*在集合*V* 和Γ上的最强必要条件）

1: *T*1 = {*C* | *C* ∈ Γ是*V* 上的一个子句}，*T*2 = Γ − *T*1

2: 将出现在*T*2中的*q*用⊤代替，并将得到的结果和*T*1分别转换成为子句集合的极小集*T*3和*T*0.

3: 令*V* ′ = *Var*(*T*3) − *V* ;

4: **if** *V* ′ = 0/ **then**

5: 跳转到post-processing;

6: **end if**

7: 从*V* ′中随机选择一个原子*p*，且令*V* ′ = *V* ′ − {*p*};

8: 将*T*3[*p/*⊤] ∪ *T*3[*p/*⊥]转换为极小集得到结果*T*3，跳转到4；

9: post-processing：根据下面步骤化简*T*3：

10: 移除*T*3中被*T*0包蕴的子句；

11: 对*T*3中的每个子句*α*，将(*T*3 − {*α*}) ∪ *T*0转换为极小子句集*Tα* ；

12: 如果*α*被*Tα* 中的某个子句包蕴，则将*α*从*T*3中删除；

13: **return** *T*3中子句的合取

在一阶逻辑中，SNC和WSC的定义和命题逻辑下相似，也可用遗忘来计算[[80](#_bookmark196)]。 不 同的是，一阶逻辑中的遗忘计算比较复杂，且遗忘的结果不一定能用一阶语言表示出 来。

定理 **2.4.** 对任意一阶公式*α*、关系符号的集合*P* 和句子*Th:*

* *α*在*P*和*Th*上的*SNC*是∃Φ*.*[*Th* ∧ *α*]，
* *α*在*P*和*Th*上的*WSC*是∀Φ*.*[*Th* → *α*]*,*

其中Φ是出现在*Th* ∧ *α*且不出现在*P*中的关系符号的集合。

正如前面所说，一阶逻辑下的遗忘主要使用归结和SOQE的方法来计算，但由于本

文不涉及相关知识，所以这里就不详细介绍一阶逻辑下遗忘的计算。

在模态逻辑S5中，SNC和WSC也可以通过遗忘来计算：

定理 **2.5.** 给定*S5*公式Γ和原子命题集*V* ⊆ *A* ，*q* ∈ *Var*(Γ) − *V* ，则：

1. *q*在*V* 和Γ上的*SNC*为*KForget*(Γ ∧ *q, Var*(Γ) − *V* )；
2. *q*在*V* 和Γ上的*WSC*为¬*KForget*(Γ ∧ ¬*q, Var*(Γ) − *V* )。

此时，由算法 [2.1](#_bookmark0)不难得出计算SNC和WSC的算法，这里就不再赘述。

**2.6** 本章小结

围绕本文的研究工作，本章首先介绍了最基本的真假赋值概念，给出了命题逻辑 公式的解释，随后给出了时序逻辑依赖的Kripke结构的定义。 其次，本章详细介绍了 带索引的CTL和*µ*-演算公式的语法和语义。CTL公式是带索引的CTL公式的子类，基于

此，介绍了CTL公式的标准形式——SNF*g*

CTL

子句，并详细介绍了如何将CTL公式转换

为SNF*g*

CTL

子句集。 随后对CTL中的归结系统和归结过程做了详细介绍，为下文使用基

于归结的方法计算CTL遗忘做铺垫。 最后，本章详细介绍了本文密切相关的经典逻辑 和模态逻辑下遗忘的定义、基本公设及相关算法，并给出这些逻辑系统下SNC（WSC） 的定义和使用遗忘计算SNC（WSC）的详细算法。 本章介绍的内容为后续章节提供了 基本模型与定义，是开展后续研究工作的理论出发点。

第三章 **CTL**和*µ***-**演算遗忘理论

从一个公式中“遗忘”一些原子命题得到的结果应该不“违背”定义在其它原子 命题集合上的公式，即：对于其它原子命题集合上的公式，原始公式能够逻辑蕴涵它 当且仅当遗忘结果能逻辑蕴涵它。 从模型的角度来讲，遗忘结果的模型与原始公式的 模型在“除去”被遗忘的那些原子命题之后是相互模拟的。 互模拟描述的是两个在行 为上能够相互替代的转换系统[*[24*](#_bookmark155)*]*。 在本文中，转换系统（包括反应式系统）被描述成 为*Kripke*结构。 因此，为了定义*CTL*遗忘，这部分给出在给定原子命题集合上*Ind-*结构 之间互模拟的定义及其性质。 此外，*µ-*演算是一种表达能力比*CTL*强的语言，且其公 式中包含自由变元。 因而，本章将扩展上述提到的互模拟到变元上。

根据上述两种互模拟，给出了*CTL*和*µ-*演算遗忘的定义。 与后面章节将要讲述的

约束*CTL*下的遗忘相对应，这部分探索没有约束的遗忘的一般性质。

本章其余部分组织如下：第[*3.1*](#_bookmark46)节首先定义一种*V-*互模拟的概念及其性质，并定 义互模拟等价；其次，定义*CTL*下的遗忘，并探索遗忘的基本性质和公设。 第[*3.2*](#_bookmark60)节定 义*Var-V -*互模拟，并证明其对*µ-*公式具有不变性；然后给出*µ-*公式遗忘的定义及相关 性质。 最后，进行本章工作总结。

**3.1 CTL**遗忘理论

**3.1.1** 互模拟

这部分给出定义在给定原子命题集合*V* 上的互模拟概念，本文称之为*V* -互模拟。 尽管zhang等人在S5中给出了相似的概念[[48](#_bookmark176)]，但是如在基础知识部分所述，*S*5的语义 是定义在一种特殊的Kripke结构（K-解释）下的，其不具有一般性。 因此这里研究一 种更加一般的*V* -互模拟。

定义 **3.1** (*V* -互模拟)**.** 给定原子命题集合*V* ⊆ *A* 、 索引集合*I* ⊆ *Ind*和初始*Ind-*结构*Mi* = (*Si, Ri, Li,* [ ]*i, si* ) (*i* = 1*,* 2)。 对关系*BV* ⊆ *S*1 × *S*2和任意*s*1 ∈ *S*1和*s*2 ∈ *S*2，若(*s*1*, s*2) ∈ *BV* 蕴 涵下列条件，则称*BV* 是*M*1 和*M*2之间的一个*V -*互模拟关系：

0

*(i) L*1(*s*1) − *V* = *L*2(*s*2) − *V* ；

*(ii)* ∀*r*1 ∈ *S*1，若(*s*1*, r*1) ∈ *R*1，则∃*r*2 ∈ *S*2 使得(*s*2*, r*2) ∈ *R*2 和(*r*1*, r*2) ∈ *BV* ；

*(iii)* ∀*r*2 ∈ *S*2，若(*s*2*, r*2) ∈ *R*2，则∃*r*1 ∈ *S*1 使得(*s*1*, r*1) ∈ *R*1 和(*r*1*, r*2) ∈ *BV*。

若*M*1和*M*2之间存在一个*V* -互模拟关系*BV* 使得(*s*1*, s*2) ∈ *BV* ，则称两个Ind-结构*K*1

= (*M*1*, s*1) 和*K*2 = (*M*2*, s*2) 是*V* -互模拟的（也称*K*1和*K*2关于*V* 是互模拟的），记为*K*1 ↔*V*

*K*2。 令*i* ∈ {1*,* 2}，*πi* = (*si,*1*, si,*2*, . . .*) 为*Mi* 上的路径，若对任意 *j* ≥ 1都有*K*1*, j* ↔*V K*2*, j*，

则称这两条路径是*V* -互模拟的，记为*π*1 ↔*V π*2，其中*Ki, j* = (*Mi, si, j* )。

直观上，若两个状态在不考虑*V* ⊆ *A* 中的元素时，其行为是相同的，则称这两个 状态在*V* 上是“互模拟”的。 当*V* = 0/ ，*V* -互模拟的三个条件即为CTL中的互模拟要满 足的条件。 下文中，若能从上下文中明确初始Ind-Kripke结构，则将*K*1 ↔*V K*2 简写

为*s*1 ↔*V s*2。

例 **3.1.** 令*K*1，*K*2和*K*3为三个Ind-结构（其索引对应的后继函数与*V* -互模拟无关，所以 在图里没给出），分别如图中的*K*1，*K*2和*K*3所示。 它们之间的互模拟关系如图中虚线 所示，即*K*1 ↔{*sp*} *K*2，*K*2 ↔{*se*} *K*3和*K*1 ↔{*sp,se*} *K*3。 此外，可以看出*K*1，*K*2和*K*3之 间是互不互模拟[[24](#_bookmark155)]的，即不0/ -互模拟。

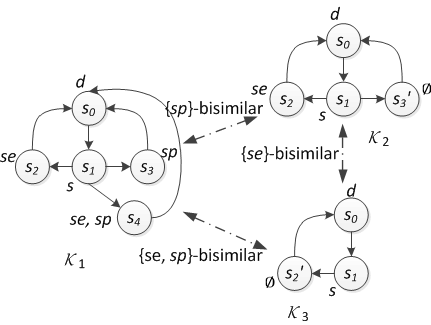


图 3.1: K-结构之间的*V* -互模拟关系示意图

*V* -互模拟给出了两个结构之间相互模仿的行为关系，下述命题给出了这种关系的

一些关键性质。

命题 **3.1.** 给定集合*Vi* ⊆ *A* 、 状态*s*′、 路径*π*′和*Ind-*结构*Kj* = (*Mj, s j* )，其中*i* = 1*,* 2， *j* =

*i* *i*

1*,* 2*,* 3。 如果*K*1 ↔*V*1 *K*2且*K*2 ↔*V*2 *K*3，则：

*(i) K*1 ↔*V*1∪*V*2 *K*3*;*

*(ii)* 若*V*1 ⊆ *V*2，则*K*1 ↔*V*2 *K*2；

1. *s*′ ↔*V s*′

(*i* = 1*,* 2) 蕴涵*s*′ ↔*V* ∪*V*

*s*′ ；

1 *i* 2

1 1 2 2

*(iv) π*′ ↔*V π*′ (*i* = 1*,* 2) 蕴涵*π*′ ↔*V* ∪*V*

*π*′ ；

1 *i* 2

1 1 2 2

*(v)* 对*M*1上的每条路径*πs*1 ，存在*M*2上的一条路径*πs*2 使得*πs*1 ↔*V*1 *πs*2 ，反之也成立。

证明**.** (i) 令*Mj* = (*Sj, Rj, Lj,* [ ] *j, s j* ) ( *j* = 1*,* 2*,* 3)，*B*为*s*1和*s*2之间的*V*1-互模拟关系，即*s*1

0

↔*V*1 *s*2 通过*B*形成*V*1-互模拟关系，*s*2 ↔*V*2 *s*3通过*B*′′形成*V*2-互模拟关系。 令*B*′ = {(*w*1*,*

*w*3) | (*w*1*, w*2) ∈ *B* 和(*w*2*, w*3) ∈ *B*′′，其中*wi* ∈ *Si* (*i* = 1*,* 2*,* 3)}。 为了证明*K*1 ↔*V*1∪*V*2 *K*3， 下证*B*′ 是一个包含(*s*1*, s*3) 的*V*1 ∪ *V*2-互模拟关系。 由于(*s*1*, s*2) ∈ *B* 和(*s*2*, s*3) ∈ *B*′′，所 以(*s*1*, s*3) ∈ *B*′。 对于所有(*w*1*, w*3) ∈ *B*′：

(a) 存在一个*w*2 ∈ *S*2使得(*w*1*, w*2) ∈ *B* 且(*w*2*, w*3) ∈ *B*′′，因此由*w*1 ↔*V*1 *w*2可知，*L*1(*w*1)

−*V*1 = *L*2(*w*2) − *V*1，且由*w*2 ↔*V*2 *w*3可知*L*2(*w*2) − *V*2 = *L*3(*w*3) − *V*2。 所以有*L*1(*w*1) −

(*V*1 ∪ *V*2) = *L*3(*w*3) − (*V*1 ∪ *V*2)。

(b) ∀*u*1 ∈ *S*1，若(*w*1*, u*1) ∈ *R*1，则∃*u*2 ∈ *S*2 使得(*w*2*, u*2) ∈ *R*2 和(*u*1*, u*2) ∈ *B* （由*B*′的定义 可知(*w*1*, w*2) ∈ *B* 且(*w*2*, w*3) ∈ *B*′′）；因而∃*u*3 ∈ *S*3 使得(*w*3*, u*3) ∈ *R*3 且(*u*2*, u*3) ∈ *B*′′， 所以由*B*′的定义可知(*u*1*, u*3) ∈ *B*′。

(c) ∀*u*3 ∈ *S*3，若(*w*3*, u*3) ∈ *R*3，则∃*u*2 ∈ *S*2 使得(*w*2*, u*2) ∈ *R*2 和(*u*2*, u*3) ∈ *B*2；因此∃*u*1 ∈

*S*1 使得(*w*1*, u*1) ∈ *R*1 且(*u*1*, u*2) ∈ *B*，所以由*B*′的定义可知(*u*1*, u*3) ∈ *B*′。

(ii) 假定*BV*1 是*M*1 和*M*2之间的一个*V*1-互模拟关系，且(*s*1*, s*2) ∈ *BV*1。 下证*BV*1 也

是*M*1 和*M*2之间的一个*V*2-互模拟关系。 对任意(*w*1*, w*2) ∈ *BV*1 ，有：

∙ 因为*L*1(*w*1) − *V*1 = *L*2(*w*2) − *V*1 和*V*1 ⊆ *V*2，所以*L*1(*w*1) − *V*2 = *L*2(*w*2) − *V*2；

∙ ∀*r*1 ∈ *S*1，若(*w*1*, r*1) ∈ *R*1，因为*BV*1 是*M*1 和*M*2之间的一个*V*1-互模拟关系，则∃*r*2 ∈

*S*2 使得(*w*2*, r*2) ∈ *R*2 且(*r*1*, r*2) ∈ *BV*1 ；

∙ ∀*r*2 ∈ *S*2，若(*w*2*, r*2) ∈ *R*2，因为*BV*1 是*M*1 和*M*2之间的一个*V*1-互模拟关系，则∃*r*1 ∈

*S*1使得(*w*1*, r*1) ∈ *R*1 且(*r*1*, r*2) ∈ *BV*1。

由于*Vi* ⊆ (*V*1 ∪ *V*2)（*i* = 1*,* 2），则(iii)是(ii)的一种特殊情况，因而(iv)从(iii)可以容易

得到。

(v) 可以从*V* -互模拟的定义容易得到，因为*s*1 ↔*V*1 *s*2 （即：*K*1 ↔*V*1 *K*2）。

在命题[3.1](#_bookmark51)中，性质(iii)-(v)的含义比较直观。 性质(i)表示如果一个结构分别与另 外的两个结构*K*1和*K*3具有*V*1和*V*2-互模拟关系，则*K*1和*K*3 是*V*1 ∪ *V*2-互模拟的（如图 [3.1](#_bookmark50)所示）。 如后文所示，这一性质对遗忘性质的证明至关重要。 性质(ii)表示若两个结 构关于某一集合是互模拟的，则这两个结构关于该集合的超集是互模拟的。

从互模拟的定义来看，如果两个结构是*V* -互模拟的，那么对于与*V* 中原子命题无

关的公式*ϕ*来说，这两个结构同时满足或不满足*ϕ*。 这一性质可以形式化地描述如下：

定理 **3.1.** 令*V* ⊆ *A* 是原子命题集，*Ki* (*i* = 1*,* 2)是两个具有*V-*互模拟的*Ind-*结构，即：

*K*1 ↔*V K*2，Φ是一个*CTL*公式且*IR*(Φ*,V* )。 则有*K*1 |= Φ当且仅当*K*2 |= Φ。

证明**.** 下面通过归纳CTL公式的结构证明这一结论。假设*Var*(Φ)∩*V* = 0/ ，*K*1 = (*M , s*)和*K*1 = (*M* ′*, s*′)。

基始**.** Φ = *p* （*p* ∈ *A* − *V* ）。

(*M , s*) |= Φ当且仅当*p* ∈ *L*(*s*) （可满足关系的定义）

⇔ *p* ∈ *L*′(*s*′) （*s* ↔*v s*′）

⇔ (*M* ′*, s*′) |= Φ。

归纳步**.** (a) Φ = ¬*ψ*。

(*M , s*) |= Φ当且仅当(*M , s*) |= *ψ*

⇔ (*M* ′*, s*′) |= *ψ* （归纳假设）

⇔ (*M* ′*, s*′) |= Φ。

1. Φ = *ψ*1 ∨ *ψ*2。

(*M , s*) |= Φ

⇔ (*M , s*) |= *ψ*1或(*M , s*) |= *ψ*2

⇔ (*M* ′*, s*′) |= *ψ*1或(*M* ′*, s*′) |= *ψ*2 （归纳假设）

⇔ (*M* ′*, s*′) |= Φ。

1. Φ = EX*ψ*。

(*M , s*) |= Φ

⇔ 存在一条路径*π* = (*s, s*1*, . . .* )使得(*M , s*1) |= *ψ*

⇔ 存在一条路径*π*′ = (*s*′*, s*′ *, . . .* )使得*π* ↔*V π*′ (*s* ↔*V s*′，命题 [3.1](#_bookmark51))

1

⇔ *s*1 ↔*V s*′

1

(*π* ↔*V π*′)

⇔ (*M* ′*, s*′ ) |= *ψ* （归纳假设）

1

⇔ (*M* ′*, s*′) |= Φ。

1. Φ = EG*ψ*。

(*M , s*) |= Φ

⇔ 存在一条路径*π* = (*s* = *s*0*, s*1*, . . .* )，使得对于任意*i* ≥ 0，都有(*M , si*) |= *ψ*

⇔ 存在一条路径*π*′ = (*s*′ = *s*′ *, s*′ *, . . .* )，使得*π* ↔*V π*′ (*s* ↔*V s*′，命题 [3.1](#_bookmark51))

0 1

⇔ 对于任意*i* ≥ 0，都有*si* ↔*V s*′

*i*

(*π* ↔*V π*′)

⇔ 对于任意*i* ≥ 0，都有(*M , s*′) |= *ψ* （归纳假设）

*i*

⇔ (*M* ′*, s*′) |= Φ。

1. Φ = E(*ψ*1 U*ψ*2)。

(*M , s*) |= Φ

⇔ 存在一条路径*π* = (*s* = *s*0*, s*1*, . . .* )和*i* ≥ 0，使得(*M , si*) |= *ψ*2，且对所有的0 ≤ *j < i*都

有(*M , s j* ) |= *ψ*1

⇔ 存在一条路径*π*′ = (*s*′ = *s*′ *, s*′ *, . . .* )，使得*π* ↔*V π*′ (*s* ↔*V s*′，命题 [3.1](#_bookmark51))

0 1

⇔ (*M* ′*, s*′) |= *ψ*2，且对于所有的0 ≤ *j < i*，都有(*M* ′*, s*′ ) |= *ψ*1 （归纳假设）

*i j*

⇔ (*M* ′*, s*′) |= Φ。

上述定理中，公式Φ不包含索引，否则，Ind-结构中的索引函数可能会影响公式 的可满足性。 例：令*φ* = E⟨1⟩ X *p*、 *K* = (*M , s*) 和*M* = (*S, R, L,* [ ]*, s*0)，其中*S* = {*s*0*, s*1}, *L*(*s*0) = 0/、*L*(*s*1) = {*p*}、*R* = {(*s*0*, s*1)*,* (*s*0*, s*0)*,* (*s*1*, s*1)*,* (*s*1*, s*0)} 和[1] = {(*s*0*, s*1)*,* (*s*1*, s*1)}。显

然，*K* |= *φ*。 令*K* ′ = (*M* ′*, s*) 和*M* ′ = (*S, R, L,* [ ]′*, s*0)，其中[1]′ = {(*s*0*, s*0)*,* (*s*1*, s*1)}。 显

然，*K* ↔{*q*} *K* ′，IR(*φ,* {*q*})。 但是，*K* ′ |= *φ*。

例 **3.2.** 令*ϕ*1 = *d* ∧ EF*se* ∧ AG(*se* → AX*d*)和*ϕ*2 = *d* ∧ AX*se*是两个CTL公式，且IR(*ϕ*1*,* {*sp*})

和IR(*ϕ*2*,* {*sp*})。 因此，可以验证图[3.1](#_bookmark50)中的*K*1和*K*2都满足*ϕ*1，但是都不满足*ϕ*2。

定义 **3.2** (互模拟等价，bisimilar equivalence)**.** 给定原子命题集合*V* ⊆ *A* ，公式*ϕ* 和*ψ*。 若对任意*K* |= *ϕ*，都存在一个*K* ′ |= *ψ*，使得*K* ↔*V K* ′；且对任意*K* ′ |= *ψ*，都存在一 个*K* |= *ϕ*，使得*K* ↔*V K* ′，则称公式*ϕ*和*ψ*是*V* -互模拟等价的(bisimilar equivalence)， 记为*ϕ* ≡*V ψ*。

由定义[3.1](#_bookmark48)和[3.2](#_bookmark53)，和命题[3.1](#_bookmark51)可容易得出下列引理。

引理 **3.1.** 对任意*V* ⊆ *A* ，↔*V* 和≡*V* 为等价关系。

证明**.** 由命题[3.1](#_bookmark51)(i)可知↔*V* 是传递的。 显然也是自反和对称的。 因此是等价关系。

关系≡*V* 显然是自反和对称的。 假设*ϕ* ≡*V ψ*和*ψ* ≡*V ξ*。 则对任意*K* |= *ϕ*，由*ϕ* ≡*V ψ*可知， 存在一个*K* ′ |= *ψ*使得*K* ′ ↔*V K* ； 且由*ψ* ≡*V ξ* 可知， 存在一个*K* ′′ |= *ξ* 使 得*K* ′ ↔*V K* ′′。又因为↔*V* 是一个等价关系，因此，*K* ↔*V K* ′′。类似地，对任意*K* ′′ |= *ξ* ，存在*K* |= *ϕ* 使得*K* ′′ ↔*V K* 。 这表明含≡*V* 是传递的。 因此，≡*V* 是等价关系。

此外，由互模拟等价的定义和上述结论可得以下推论。

推论 **3.1.** 令*V*、*V*1、*V*2 为*A* 的子集，*ϕ*和*ψ*为公式。

*(i)* 若*ϕ* ≡ *ψ*，则*ϕ* ≡*V ψ*。

*(ii)* 若*ϕ* 和*ψ*不包括索引，且*ϕ* ≡0/ *ψ*，则*ϕ* ≡ *ψ*。

1. 若*ϕ* ≡*Vi ψ* (*i* = 1*,* 2)，则*ϕ* ≡*V*1∪*V*2 *ψ*。
2. 若*ϕ* ≡*V*1 *ψ* 和*V*1 ⊆ *V*2，则*ϕ* ≡*V*2 *ψ*。

证明**.** (i) 对任意*ϕ*（或*ψ*）的模型*K* 和*V* ⊆ *A* ，存在*K* ↔*V K* 。 因此，*ϕ* ≡*V ψ*。

(ii) 对任意*ϕ*的模型*K* ，存在*ψ*的一个模型*K* ′使得*K* ↔0/ *K* ′。 显然IR(*ψ,* 0/ )，由定 理 [3.1](#_bookmark52)可知*K* |= *ψ*。类似地，对任意*K* ′ |= *ψ*，存在*K* |= *ϕ*使得*K* ↔0/ *K* ′，因此*K* ′ |= *ϕ*。

(iii) 对任意*K* |= *ϕ*，存在*K* ′ |= *ψ*，使得*K* ↔*Vi K* ′ (*i* = 1*,* 2)。因此，由命题[3.1](#_bookmark51)(i)可 知*K* ↔*V*1∪*V*2 *K* ′。 类似地，对任意*K* ′ |= *ψ*，存在*K* |= *ϕ*，使得*K* ↔*V*1∪*V*2 *K* ′。 因此， *ϕ* ≡*V*1∪*V*2 *ψ*。

同理可证(iv)。

请注意，在上述结论(ii)中“*ϕ*和*ψ*不包含索引”是必要的。 否则，令*ϕ* = E⟨1⟩ X *p*

和*ψ* = E⟨2⟩ X *p*，可以证明，*ϕ* ≡0/ *ψ*，但是*ϕ* ̸≡ *ψ*。

命题 **3.2.** 令*ϕ*为一个*CTL*公式。则*ϕ* ≡*U Tϕ* ，其中*Tϕ* = SNF*g*

CTL

(*ϕ*)和*U* = *Var*(*Tϕ* )−*Var*(*ϕ*)。

证明**.** 令*T*0*, T*1*, . . . , Tn* = *Tϕ* 为转换过程产生的公式集合序列，其中*p*是不出现在*ϕ*中的原 子命题，*T*0 = {AG(**start** → *p*)*,* AG(*p* → **simp**(**nnf**(*ϕ*)))}且对任意*i*（0 ≤ *i < n*），有*Ti*+1 = (*Ti* − {*ψ*}) ∪ *Ri*（*Trans*(*ψ*)返回的结果为*Ri*）。 此外，在这一过程中，所有的公式都是否 定范式的形式。

为了证明命题中的结论，只需证明，对任意*i*（0 ≤ *i < n*），*Ti* ≡*V* ′ *Ti*+1成立。 由 于*Ti*+1是 由*Ti*通 过 表[2.1](#_bookmark29)中 的 规 则 作 用 于*Ti*中 的 某 一 个 公 式 得 到， 因 此 证 明 过 程 分 为两个部分：（1） 从*ϕ*到*T*0部分；（2） 对表[2.1](#_bookmark29)中的规则做归纳的部 分。 假设*M*1 = (*S*1*, R*1*, L*1*,* [ ]*, s*1)和*M*2 = (*S*2*, R*2*, L*2*,* [ ]2*, s*2)。

（1）下证*ϕ* ≡{*p*} *T*0。

(⇒) ∀(*M*1*, s*1) ∈ *Mod*(*ϕ*)， 可以构造一个Ind-Kripke结构*M*2 = (*S*2*, R*2*, L*2*,* [ ]*, s*2)使 得*M*2除了*L*2(*s*2) = *L*1(*s*1) ∪ {*p*}（默认不出现在*ϕ*中的原子命题都不出现在状态的标签 中），其它元素都与*M*1中元素相同。 显然，(*M*2*, s*2) |= *T*0且(*M*1*, s*1) ↔{*p*} (*M*2*, s*2)。

(⇐) ∀(*M*1*, s*1) ∈ *Mod*(*T*0)，由**start**的语义可知(*M*1*, s*1) |= *ϕ*。

（2） 下证对任意*i* （0 ≤ *i < n*）， 有*Ti* ≡*V* ′ *Ti*+1成立， 其中*Ti*+1 = (*Ti* − {*ψ*}) ∪ *Ri*。 这里，用*ψ* →*t Ri*表示*Ri*是使用规则*t*在公式*ψ*上得到的结果，且*Ti* = *X* ∪ {*ψ*}（显然， *Ti*+1 = *X* ∪ *Ri*）。 下面证明规则*t* ∈ {**Trans(1), Trans(4), Trans(6)**}的情形，其它情形可以 类似地证明。

（a）*t* = **Trans(1)**。

(⇒) (*M*1*, s*1) ∈ *Mod*(*Ti*)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → EX*ϕ*)

⇒ (*M*1*, s*1) |= *X*，且对任意路径*π*上的状态*s*1*, j* ( *j* ≥ 1)，有(*M , s*1*, j* ) |= ¬*q*，或存在一个

状态*s*1*, j*+1使得(*s*1*, j, s*1*, j*+1) ∈ *R*1且(*M , s*1*, j*+1) |= *ϕ*。

由此构造一个初始Ind-Kripke结构*M*2，使得*M*2与*M*1相同，除了：对使用规则**Trans(1)**在

公式AG(*q* → EX*ϕ*)上而引入的新索引*ind*，有[*ind*]2 = J

*s*∈*S*

*Rs* ∪ *Ry*。 其中：

∙ *Rs*1*, j* = {(*s*1*, j, s*1*, j*+1)*,* (*s*1*, j*+1*, s*1*, j*+2)*, . . .* } ( *j* ≥ 1)，其满足“若(*M*1*, s*1*, j* ) |= *q*，则(*M*1*, s*1*, j*+1)

|= *ϕ*”且“对于任意*i* ≥ *j*，若(*s*1*,i, s*′) ∈ *Rs* (*s* ̸= *s*1*, j* )，则*s*′ = *s*1*,i*+1”；

∙ *Ry* = {(*sx, sy*) | *sx* ∈ *S*，若对任意(*s*′ *, s*′ ) ∈ J

*Rs*，有*s*′

̸= *sx*，则找一个状态*sy* ∈ *S*2，

使得(*sx, sy*) ∈ *R*2}。

1 2 *s*∈*S* 1

显然，(*M*1*, s*1) ↔⟨0/ *,*{*ind*}⟩ (*M*2*, s*2)

⇒ 对任意从*s*2开始的路径*π* = (*s*2 = *s*2*,*1*, s*2*,*2*, . . .* )，如果*s*2*, j* ∈ *π*，则(*M*2*, s*2*, j* ) |= ¬*q*或(*M*2*, s*2*, j* ) |=

E⟨*ind*⟩ X*ϕ*

⇒ (*M*2*, s*2) |= AG(*q* → E⟨*ind*⟩ X*ϕ*)

⇒ (*M*2*, s*1) |= *X* ∧ AG(*q* → E⟨*ind*⟩ X*ϕ*)

(⇐) ∀(*M*1*, s*1) ∈ *Mod*(*Ti*+1)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → E⟨*ind*⟩ X*ϕ*)

⇒ (*M*1*, s*1) |= *X* 且(*M*1*, s*1) |= AG(*q* → E⟨*ind*⟩ X*ϕ*)

⇒ 对任意以*s*1为起点的路径上的任意状态*s*1*, j*，(*M*1*, s*1*, j* ) |= ¬*q*或(*M*1*, s*1*, j* ) |= EX*ϕ*

⇒ (*M*1*, s*1) |= AG(*q* → EX*ϕ*)

⇒ (*M*1*, s*1) |= *X* ∧ AG(*q* → EX*ϕ*)。

（b）*t* = **Trans(4)**。

(⇒) (*M*1*, s*1) ∈ *Mod*(*Ti*)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → *ϕ*1 ∨ *ϕ*2)

⇒ (*M*1*, s*1) |= *X*，且∀*s*′ ∈ *S*1，(*M*1*, s*′ ) |= *q* → *ϕ*1 ∨ *ϕ*2

1 1

⇒ (*M*1*, s*′ ) |= ¬*q*或(*M*1*, s*′ ) |= *ϕ*1 ∨ *ϕ*2。

1 1

如下构造初始Ind-Kripke结构*M*2 = (*S*2*, R*2*, L*2*,* [ ]2*, s*2)：

∙ *S*2 = *S*1，*R*2 = *R*1，[ ]2与[ ]1相同，且*s*2 = *s*1；

∙ *L*2与*L*1类似， 除了： 对任意*s*′

∈ *S*2， 若(*M*1*, s*′ ) |= ¬*q*， 则*L*2(*s*′ ) = *L*1(*s*′ )， 否则

1 1 1 1

“若(*M*1*, s*′ ) |= *ϕ*1，则*L*2(*s*′ ) = *L*1(*s*′ )，否则*L*2(*s*′ ) = *L*1(*s*′ ) ∪ {*p*}”。

1 1 1 1 1

显然，(*M*2*, s*′ ) |= (*q* → *ϕ*1 ∨ *p*) ∧ (*p* → *ϕ*2)且(*M*1*, s*1) ↔{*p*} (*M*2*, s*2)，所以，(*M*2*, s*1) |=

1

*Ti*+1。

(⇐) ∀(*M*1*, s*1) ∈ *Mod*(*Ti*+1)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → *ϕ*1 ∨ *p*) ∧ AG(*p* → *ϕ*2)。 显然，

(*M*1*, s*1) |= *Ti*。

（c）*t*=**Trans(6)**。

这里证明E

⟨*ind*⟩

X的情形，AX的情形可以类似地证明。

(⇒) (*M*1*, s*1) ∈ *Mod*(*Ti*)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → E⟨*ind*⟩ X*ϕ*)

⇒ (*M*1*, s*1) |= *X*，且对任意*s*′ ∈ *S*，(*M*1*, s*′ ) |= *q* → E⟨*ind*⟩ X*ϕ*

1 1

⇒ (*M*1*, s*′ ) |= ¬*q*，或者存在一个状态*s*′，使得(*s*′ *, s*′) ∈ [*ind*]且(*M*1*, s*′) |= *ϕ*。

1 1

如下构造初始Ind-Kripke结构*M*2 = (*S*2*, R*2*, L*2*,* [ ]2*, s*2)：

∙ *S*2 = *S*1，*R*2 = *R*1，[ ]2与[ ]1一样且*s*2 = *s*1；

∙ *L*2与*L*1类似， 除了： 对任意*s*′

∈ *S*2， 若(*M*1*, s*′ ) |= ¬*q*， 则*L*2(*s*′ ) = *L*1(*s*′ )， 否则

1 1 1 1

“若(*M*1*, s*′ ) |= *q*，则*L*2(*s*′) = *L*1(*s*′) ∪ {*p*} ((*s*′ *, s*′) ∈ *R*2)”。

1 1

显然，(*M*2*, s*2) |= AG(*q* → E⟨*ind*⟩ X *p*) ∧ AG(*p* → *ϕ*), (*M*2*, s*2) |= *Ti*+1且(*M*1*, s*1) ↔{*p*} (*M*2*, s*2)

(*s*2 = *s*1)。

(⇐) (*M*1*, s*1) ∈ *Mod*(*Ti*+1)，即(*M*1*, s*1) |= *X* ∧ AG(*q* → E⟨*ind*⟩ X *p*) ∧ AG(*p* → *ϕ*)。 显然，

(*M*1*, s*1) |= *Ti*。

例 **3.3.** 令*ϕ* = A((*p* ∧ *q*)U( *f* ∨ *m*)) ∧ *r*。 *Tϕ* 是下面子句构成的集合：

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : **start** → *z,* | 2 : ⊤ → ¬*z* ∨ *r,* | 3 : ⊤ → ¬*x* ∨ *f* ∨ *m,* | 4 : ⊤ → ¬*z* ∨ *x* ∨ *y,* |
| 5 : ⊤ → ¬*y* ∨ *p,* | 6 : ⊤ → ¬*y* ∨ *q,* | 7 : *z* → AF*x,* | 8 : *y* → AX(*x* ∨ *y*) |

其中*x, y, z* 为新引入的原子命题。

**3.1.2** 遗忘算子及其性质

这部分给出CTL遗忘的定义及其相关性质。

定义 **3.3** (遗忘，forgetting)**.** 令*V* 是*A* 的子集，Φ是公式。 如果公式*ψ*满足下面条件，则

称*ψ*为从Φ中遗忘*V* 后得到的结果：

∙ *ψ*与*V* 中的原子命题无关（即：*IR*(*ψ,V* )）；

∙ *Mod*(*ψ*) = {*K* | *K* 是一个初始*Ind-*结构*,* ∃*K* ′ ∈ *Mod*(*φ* ) 使得 *K* ′ ↔*V K* }

从 定 义[3.3](#_bookmark58)可 以 看 出， 如 果 两 个 公 式*ψ*和*ψ*′都 是Φ遗 忘*V* 中 元 素 后 得 到 的 结 果， 则*Mod*(*ψ*) ≡ *Mod*(*ψ*′)。 从这个角度来看，Φ遗忘*V* 中元素后得到的所有结果是语义 等价的。 将遗忘结果记为FCTL(*φ,V* )，不做其它说明的情况下，这表示*φ* 遗忘*V* 是CTL可

表示的。此外，当*V* 中只包含一个元素的时候，可以省略集合符号，即：FCTL (Φ*,* {*p*}) ≡

FCTL(Φ*, p*)。

遗忘的定义说明了，如果公式*ψ*的任意一个模型*K* 都能找到*ϕ*的一个模型*K* ′使 得*K* ↔*V K* ′，则称*ψ*为*ϕ*遗忘*V* 中原子命题后得到的结果。 为刻画S5逻辑下遗忘的直 观含义，Zhang等人提出了遗忘的特性——这些特性被称为遗忘理论公设（forgetting postulates）[[48](#_bookmark176)]。类似地，给定CTL公式*ϕ*、*ϕ*′ = FCTL(*ϕ,V* )和原子命题集合*V* ⊆ *A* 和*ϕ*′ =

FCTL(*ϕ,V* )，CTL下遗忘理论公设如下：

(**W**) 削弱：*ϕ* |= *ϕ*′；

(**PP**) 正支持：对任意与*V* 无关的公式*η*，若*ϕ* |= *η*则*ϕ*′ |= *η*； (**NP**) 负支持：对任意与*V* 无关的公式*η*，若*ϕ* |= *η*则*ϕ*′ |= *η*； (**IR**) 无关性：IR(*ϕ*′*,V* )。

其中(**W**)和(**IR**)表明，“遗忘”削弱了公式*ϕ*且得到的结果与*V* 无关；(**PP**)和(**NP**)表明，

对任意与*V* 无关的公式*η*，*ϕ* |= *η*当且仅当*ϕ*′ |= *η*。 总而言之，遗忘结果，能蕴涵所有

与*V* 无关且能被*ϕ*蕴涵的结果，不能蕴涵所有与*V* 无关且不能被*ϕ*蕴涵的结果。 此外， 这些公设不都是独立的（如：NP由**W**和**PP**可以得出），但是这里把它们都列出来以表 达更加直观的含义。 从数据库和安全的层面讲，遗忘相当于从已有的关系表中构建一 个视图，达到隐私保护的作用。 下面的定理表明上述公设对CTL的遗忘是充分且必要 的。

定理 **3.2** (表达性定理，Representation Theorem)**.** 给定*CTL*公式*ϕ*和*ϕ*′，*V* ⊆ *A* 为原子命

题集合。 下面的陈述是等价的：

1. *ϕ*′ ≡ FCTL(*ϕ,V* )*,*
2. *ϕ*′ ≡ {*φ* | *ϕ* |= *φ* 和*IR*(*φ,V* )}*,*

*(iii)* 若*ϕ*、*ϕ*′和*V* 为*(i)*和*(ii)*中提到的符号，则公设*(****W****)*、*(****PP****)*、*(****NP****)*和*(****IR****)*成立。

证明**.** (*i*) ⇔ (*ii*). 通过证明如下等式证明该结论。

*Mod*(FCTL(*ϕ,V* )) = *Mod*({*φ* | *ϕ* |= *φ,* IR(*φ,V* )})*.* (⇒) *K* ′ |= FCTL(*ϕ,V* )

⇒ ∃*K* 使得*K* |= *ϕ*且*K* ↔*V K* ′ (定义 [3.3](#_bookmark58))

⇒ ∀*φ* ，若*ϕ* |= *φ* 且IR(*φ,V* )，则*K* ′ |= *φ* (定理 [3.1](#_bookmark52))

⇒ *K* ′ |= {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}

(⇐) 因为IR(FCTL(*ϕ,V* )*,V* )且*ϕ* |= FCTL(*ϕ,V* )，由定义 [3.3](#_bookmark58)可知{*φ* | *ϕ* |= *φ,* IR(*φ,V* )} |=

FCTL(*ϕ,V* )。

(*ii*) ⇒ (*iii*).令*A* = {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}。首先，由于对任意*φ* ′ ∈ *A*，都有*ϕ* |= *φ* ′，所 以，*ϕ* |= *ϕ*′。其次，对任意公式*φ* ，若IR(*φ,V* )且*ϕ* |= *φ* ，则*φ* ∈ *A*，因此，*ϕ*′ |= *φ*。第三， 对任意公式*φ* ，若IR(*φ,V* )且*ϕ* |= *φ* ，则*φ* ̸∈ *A*。因此，*ϕ*′ |= *φ*。最后，因为对任意*φ* ′ ∈ *A*， 都有IR(*ϕ*′*,V* ) ，所以，IR(*φ* ′*,V* )。

(*iii*) ⇒ (*ii*).一方面，由(**PP**) 和(**NP**)可知，对任意公式*φ* ′且IR(*φ* ′*,V* )，*ϕ* |= *φ* ′当且仅 当*ϕ*′ |= *φ* ′。 所以，对任意*φ* ′ ∈ {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}，都有*ϕ*′ |= *φ* ′，所以，*ϕ*′ |= {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}。 另一方面，由(**W**) 和(**IR**)可知，{*φ* | *ϕ* |= *φ,* IR(*φ,V* )} |= *ϕ*′。 因此， *ϕ*′ ≡ {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}。

定理 [3.2](#_bookmark59)并不表明遗忘的结果一定存在。 事实上，存在一个CTL公式和原子命题集 合，使得从该公式里遗忘该集合得到的结果不可用CTL公式表达。 例：令*p*和*x*为两个 不同的原子命题，*ϕ*(*p, x*)[1](#_bookmark0)为下面公式合取[[81](#_bookmark197)]：

AG(¬*x* ∧ ¬AG *p* → ¬AX¬*x*)*,* AG(¬AX¬*x* → AX*x*)*,*

1*ϕ*(*p, x*) 表示具有原子命题集合*Var*(*ϕ*) = {*p, x*}的公式。

AG(AX*x* → ¬*x* ∧ ¬AG *p*)*,* AG(*x* → ¬AG *p*)*,* AG(AFAG *p*)*.*

[Maksimova](#_bookmark197)证明*ϕ*(*p, x*)∧*ϕ*(*p, y*) |= *x* ↔ *y*，且不存在CTL公式*ψ*使得*Var*(*ψ*) = {*p*}且*ϕ*(*p, x*) |= *x* ↔ *ψ*，即CTL不具有Beth性质。 这一结论蕴涵如下命题：

命题 **3.3.** FCTL(*x* ∧ *ϕ*(*p, x*)*,* {*x*}) 在*CTL*中是不可表示的。

证明**.** 令*ψ*(*p*) = FCTL(*x* ∧ *ϕ*(*p, x*)*,* {*x*})为一个CTL公式。 有

*ϕ*(*p, x*) ∧ *ϕ*(*p, y*) ⊢ *x* ↔ *y*

⇒ *ϕ*(*p, x*) ∧ *x* ⊢ *ϕ*(*p, y*) → *y*

⇒ *ϕ*(*p, x*) ∧ *x* ⊢ *ψ*(*p*) 和*ψ*(*p*) ⊢ *ϕ*(*p, y*) → *p*(*y*) (定理 [3.2](#_bookmark59))

⇒ *ϕ*(*p, x*) ⊢ *x* → *ψ*(*p*)，和*ϕ*(*p, y*) ⊢ *ψ*(*p*) → *p*(*y*)，这表明*ϕ*(*p, x*) ⊢ *ψ*(*p*) → *p*(*x*)

⇒ *ϕ*(*p, x*) ⊢ *x* ↔ *ψ*(*p*)，这是一个矛盾。

本段和下一段在related work的国内外研究现状 中。 事实上， 遗忘结果的存在 性（可表达性）与均匀插值（或Craig插值）性质密切相关。 从形式上说，如果一个 逻辑系统*L* 中任意的公式*ϕ*和*ψ*，若*ϕ* |=*L ψ*，则存在一个公式*ξ* 使得*ϕ* ⊢*L ξ*、 *ξ* ⊢*L ψ*和*Var*(*ξ* ) ⊆ *Var*(*ϕ*) ∩ *Var*(*ψ*)，则*L* 具有或Craig插值性质，若*ξ* 与*ψ*无关，则*L* 具有均 匀插值性质。 研究表明，LTL、CTL和CTL\*不具有均匀插值性质[[81](#_bookmark197),[114](#_bookmark228)]。

从命题公式*ϕ*中遗忘原子命题*p*为：*Forget*(*ϕ,* {*p*}) ≡ *ϕ*[*p/*⊥] ∨ *ϕ*[*p/*⊤]。本文遗忘的 定义与Lin等人于1994提出命题逻辑下的遗忘一致，即：本文的CTL遗忘扩展了命题逻 辑遗忘。

定理 **3.3.** 给定一个命题公式*ϕ*和原子命题集*V* ⊆ *A* ，则下面逻辑等式成立。

FCTL(*ϕ,V* ) ≡ *Forget*(*ϕ,V* )*.*

证明**.** 一方面，对于FCTL(*ϕ,V* )的任意一个模型(*M , s*)，由遗忘的定义可知存在一个*ϕ*的 模型(*M* ′*, s*′)使得(*M , s*) ↔*V* (*M* ′*, s*′)。 因而有*L*(*s*) − *V* = *L*′(*s*′) − *V* （其中*L* ∈ *M* ，*L*′ ∈ *M* ′），这表明(*M , s*) |= *Forget*(*ϕ,V* )。

另一方面，对于*Forget*(*ϕ,V* )的任意一个模型(*M , s*)（*M* = (*S, R, L,* [ ]*, s*)），存在*ϕ*的 一个模型(*M* ′*, s*′)（*M* ′ = (*S*′*, R*′*, L*′*,* [ ]′*, s*′)），使得*L*(*s*) − *V* = *L*′(*s*′) − *V*。 如下构建一个初 始Ind-结构(*M*1*, s*1)，使得*M*1 = (*S*1*, R*1*, L*1*,* [ ]*, s*1)，其中：

∙ *S*1 = (*S* − {*s*}) ∪ {*s*1}；

∙ *R*1由将*R*出现的*s*替换为*s*1得到；

∙ 对于*S*1中任意一个状态*s*\*：

*L*1(*s*\*) =

( *L*′(*s*′)*,* 如果*s*\* = *s*1； *L*(*s*\*)*,* 否则。

显然，(*M*1*, s*1)是*ϕ*的一个模型且*s*1 ↔*V s*。 因此，(*M , s*)是FCTL(*ϕ,V* )的一个模型。

遗忘的另一个重要性质与*V* -无关性密切相关。 对于给定的公式*ψ* = *ϕ* ∧ (*q* ↔ *α*)，

如果IR(*ϕ* ∧ *α,* {*q*})，那么*ψ*遗忘*q*后得到的结果为*ϕ*。 这一性质与后文将要介绍的*SNC*

（*WSC*）的计算密切相关。 但是，由于其也是遗忘的性质，因而本文将其放在此处来探

讨。

引理 **3.2.** 给定两个公式*ϕ*和*α*，原子命题*q* ̸∈ (*Var*(*ϕ*) ∪ *Var*(*α*))，则FCTL (*ϕ* ∧(*q* ↔ *α*)*, q*) ≡

*ϕ*。

证明**.** (⇒) 令*ϕ*′ = *ϕ* ∧(*q* ↔ *α*)。对于FCTL (*ϕ*′*, q*)的任意模型(*M , s*)，由遗忘的定义可知存 在一个初始Ind-结构(*M* ′*, s*′)，使得(*M , s*) ↔{*q*} (*M* ′*, s*′)且(*M* ′*, s*′) |= *ϕ*′。(*M* ′*, s*′) |= *ϕ*显 然成立。 此外，由于IR(*ϕ,* {*q*})且(*M , s*) ↔{*q*} (*M* ′*, s*′)，由定理 [3.1](#_bookmark52)可知(*M , s*) |= *ϕ*。

(⇐) 令*M* = (*S, R, L,* [ ]*, s*)且(*M , s*) ∈ *Mod*(*ϕ*)。 下面构造初始Ind-结构(*M* ′*, s*)，使

得*M* ′ = (*S, R, L*′*,* [ ]*, s*)，其中：

*L*′ : *S* → *A* 且 ∀*s*\* ∈ *S,* 若 (*M , s*\*) |= *α,* 则*L*′(*s*\*) = *L*(*s*\*) − {*q*} 否则 *L*′(*s*\*) = *L*(*s*\*) ∪ {*q*}*,*

若 (*M , s*) |= *α,* 则*L*′(*s*) = *L*(*s*) ∪ {*q*}*,* 否则 *L*′(*s*) = *L*(*s*) − {*q*}*.*

可以得出(*M* ′*, s*) |= *ϕ*、 (*M* ′*, s*) |= *q* ↔ *α*且(*M* ′*, s*) ↔{*q*} (*M , s*)。 因此，(*M* ′*, s*) |=

*ϕ* ∧ (*q* ↔ *α*)。 所以，由(*M* ′*, s*) ↔{*q*} (*M , s*)可知，(*M , s*) |= FCTL(*ϕ* ∧ (*q* ↔ *α*)*, q*)。

根据遗忘的定义可以看出，从一个公式里遗忘某个原子命题集合中的元素是将该 集合看作一个整体来遗忘的。 下面的结论说明，遗忘原子命题集合可分解为遗忘单个 原子命题。

命题 **3.4** (分解性，Decomposition)**.** 对于给定的公式*ϕ*，原子命题集合*V* ，和原子命题*p*

（*p* ̸∈ *V* ），下面的结论成立：

FCTL(*ϕ,* {*p*} ∪ *V* ) ≡ FCTL(FCTL(*ϕ, p*)*,V* )*.*

证明**.** 一方面，令*M*1 = (*S*1*, R*1*, L*1*,* [ ]*, s*1)是一个初始Ind-Kripke结构，且(*M*1*, s*1)是FCTL(*ϕ,*

{*p*}∪*V* )的一个模型。由遗忘的定义可知，存在*ϕ*的一个模型(*M , s*)（*M* = (*S, R, L,* [ ]*, s*)）

使得(*M*1*, s*1) ↔{*p*}∪*V* (*M , s*)。 如下构建一个初始Ind-结构(*M*2*, s*2)，使得*M*2 = (*S*2*, R*2*, L*2*,*

[ ]*, s*2)且：

(1) 对于*s*2情形：令*s*2是满足下面条件的状态，

**–** *p* ∈ *L*2(*s*2)当且仅当*p* ∈ *L*1(*s*1)，

**–** 对任意*q* ∈ *V* ，*q* ∈ *L*2(*s*2)当且仅当*q* ∈ *L*(*s*)，

**–** 对其它原子命题*q*′，*q*′ ∈ *L*2(*s*2)，当且仅当*q*′ ∈ *L*1(*s*1)，当且仅当*q*′ ∈ *L*(*s*)。

(2) 其它情形：

**–** 对所有状态对(*w, w*1)，若*w* ∈ *S*，*w*1 ∈ *S*1且*w* ↔{*p*}∪*V w*1，如下构造*w*2 ∈ *S*2：

\* *p* ∈ *L*2(*w*2)当且仅当*p* ∈ *L*1(*w*1)，

\* 对任意*q* ∈ *V* ，*q* ∈ *L*2(*w*2)当且仅当*q* ∈ *L*(*w*)，

\* 对其它原子命题*q*′，*q*′ ∈ *L*2(*w*2)，当且仅当*q*′ ∈ *L*1(*w*1)，当且仅当*q*′ ∈

*L*(*w*)。

**–** 对于(*w*′ *, w*1) ∈ *R*1，若*w*2由*w*1构造而成，且*w*′ 由*w*′ 构造而成，则令(*w*′ *, w*2) ∈

1

*R*2。

2 1 2

(3) 删除*S*2和*R*2中重复的元素。

由此可知，(*M , s*) ↔{*p*} (*M*2*, s*2)且(*M*2*, s*2) ↔*V* (*M*1*, s*1)。 所以，(*M*2*, s*2) |= FCTL(*ϕ, p*)。

因此，(*M*1*, s*1) |= FCTL(FCTL(*ϕ, p*)*,V* )。

另一方面， 若(*M*1*, s*1)是FCTL(FCTL(*ϕ, p*)*,V* )的一个模型， 则存在一个初始-Ind结

构(*M*2*, s*2)，使得(*M*2*, s*2) |= FCTL(*ϕ, p*)且(*M*2*, s*2) ↔*V* (*M*1*, s*1)；且存在(*M , s*)，使得(*M , s*) |=

*ϕ*且(*M , s*) ↔{*p*} (*M*2*, s*2)。 因此，由命题 [3.1](#_bookmark51)(i)可知(*M , s*) ↔{*p*}∪*V* (*M*1*, s*1)，所以，(*M*1*, s*1) |= FCTL(*ϕ,* {*p*} ∪ *V* )。

由上面的命题不难看出，从公式中遗忘原子命题集合中的元素，可以将该集合拆 成两个集合遗忘。

推论 **3.2.** 对于给定的公式*ϕ*，原子命题集合*V*1和*V*2，有如下结论：

FCTL(*ϕ,V*1 ∪ *V*2) ≡ FCTL(FCTL(*ϕ,V*1)*,V*2)*.*

类似于被遗忘的原子命题集合能被拆成两个集合的遗忘，下面介绍在某些情况下， 从带时序算子的公式中遗忘一些原子命题，可以将这些时序算子提到遗忘操作的前 面。

命题 **3.5.** 令*V* ⊆ *A* 为原子命题集合，*φ* 为*CTL*公式，则下面等式成立：

1. FCTL(AX*φ,V* ) ≡ AXFCTL(*φ,V* )；
2. FCTL(EX*φ,V* ) ≡ EXFCTL(*φ,V* )；
3. FCTL(AF*φ,V* ) ≡ AFFCTL(*φ,V* )；
4. FCTL(EF*φ,V* ) ≡ EFFCTL(*φ,V* )；
5. FCTL(AG*φ,V* ) ≡ AGFCTL(*φ,V* )；
6. FCTL(EG*φ,V* ) ≡ EGFCTL(*φ,V* )。

证明**.** (i) (⇒) (*M , s*) |= FCTL(AX*φ,V* )

⇒ 存在(*M , s*) ↔*V* (*M* ′*, s*′)且(*M* ′*, s*′) |= AX*φ*

⇒ (*M , s*) ↔*V* (*M* ′*, s*′)，且对任意(*s*′*, s*′′) ∈ *R*′，有(*M* ′*, s*′′) |= *φ* (*R*′ ∈ *M* ′)

⇒ 对任意(*s, s*1) ∈ *R*，存在(*s*′*, s*′ ) ∈ *R*′，使得(*M , s*1) ↔*V* (*M* ′*, s*′ )，且对任意(*s*′*, s*′′) ∈ *R*′，

1 1

有(*M* ′*, s*′′) |= *φ*

⇒ 对任意(*s, s*1) ∈ *R*，有(*M , s*1) ↔*V* (*M* ′*, s*′ )且(*M* ′*, s*′ ) |= *φ*

1 1

⇒ 对任意(*s, s*1) ∈ *R*，(*M , s*1) |= FCTL(*φ,V* )

⇒ (*M , s*) |= AXFCTL(*φ,V* )。

(⇐) (*M , s*) |= AXFCTL(*φ,V* )

⇒ 对任意(*s, s*′) ∈ *R*，有(*M , s*′) |= FCTL(*φ,V* ) (*R* ∈ *M* )

⇒ 对任意(*s, s*′) ∈ *R*，有(*M , s*′) ↔*V* (*M* ′*, s*′′)且(*M* ′*, s*′′) |= *φ*

⇒ 对任意*i* ≥ 0，有(*M , s*′) ↔*V* (*M* ′*, s*′′)且(*M* ′*, s*′′) |= *φ* ，其中{*s*′ *, s*′ *, . . .*} = {*s*′ | (*s, s*′) ∈

*i i i i i* 0 1

*R*}且*M* ′ = (*S*′*, R*′*, L*′*,* [ ]′*, s*′′) (当*i* ̸= *j*时，假定*S*′ ∩ *S*′ = 0/ )

*i i i i i i i j*

⇒ (*M* \**, s*) ↔*V* (*M , s*) 且(*M* \**, s*) |= AX*φ* ，其中*M* \* = (*S*\**, R*\**, L*\**,* [ ]*, s*)且

∙ *S*\* = {*s*} ∪ J

*i*≥0

*S*′，

∙ *R*\* = {(*s, s*′′) | *i* ≥ 0} ∪ J

*i*

*R*′，

*i i*≥0 *i*

\* J

* *L* =

*i*

*i*≥0

*L*′ 和*L*\*(*s*) = *L*(*s*)，其中*L* ∈ *M*。

⇒ (*M , s*) |= FCTL(AX*φ,V* )。

(ii) (⇒) (*M , s*) |= FCTL(EX*φ,V* )

⇒ 存在(*M , s*) ↔*V* (*M* ′*, s*′) 且(*M* ′*, s*′) |= EX*φ*

⇒ 存在(*M , s*) ↔*V* (*M* ′*, s*′)，且对一些(*s*′*, s*′′) ∈ *R*′，有(*M* ′*, s*′′) |= *φ* (*R*′ ∈ *M* ′)

⇒ 存在(*s, s*1) ∈ *R*和(*s*′*, s*′ ) ∈ *R*′，使得(*M , s*1) ↔*V* (*M* ′*, s*′ )，且存在(*s*′*, s*′′) ∈ *R*′，使得(*M* ′*, s*′′) |=

1 1

*φ* （*R* ∈ *M* ）

⇒ 存在(*s, s*1) ∈ *R*和(*M , s*1) ↔*V* (*M* ′*, s*′ )，使得(*M* ′*, s*′ ) |= *φ*

1 1

⇒ 存在(*s, s*1) ∈ *R*，使得(*M , s*1) |= FCTL(*φ,V* )

⇒ (*M , s*) |= EXFCTL(*φ,V* )。

(⇐) (*M , s*) |= EXFCTL(*φ,V* )

⇒ 对一些(*s, s*′) ∈ *R*，(*M , s*′) |= FCTL(*φ,V* ) (*R* ∈ *M* )

⇒ 对一些(*s, s*′) ∈ *R*，有(*M , s*′) ↔*V* (*M* ′*, s*′′) 和(*M* ′*, s*′′) |= *φ* ，其中*M* ′ = (*S*′*, R*′*, L*′*,* [ ]′*, s*′′)

⇒ (*M* \**, s*) ↔*V* (*M , s*) 和(*M* \**, s*) |= EX*φ* ，其中*M* \* = (*S*\**, R*\**, L*\**,* [ ]*, s*)，

∙ *S*\* = *S* ∪ *S*′，

∙ *R*\* = {(*s, s*′′)} ∪ *R* ∪ *R*′，

∙ *L*\* = *L* ∪ *L*′ 且*L*\*(*s*) = *L*(*s*)，其中*L* ∈ *M*。

⇒ (*M , s*) |= FCTL(EX*φ,V* )。

(iii)和(iv)可以分别类似(i)和(ii)来证明。

(v) (⇒) (*M , s*) |= FCTL(AG*φ,V* )

⇒ 存在(*M* ′*, s*′) |= AG*φ* 且(*M , s*) ↔*V* (*M* ′*, s*′)

⇒ 对*M* 上的每一条路径*π* = (*s* = *s*1*, s*2*, . . .* )，存在*M* ′上的一条路径*π*′ = (*s*′ = *s*1*, s*′ *, . . .* )，

2

使得*π* ↔*V π*′，反之也成立，且对任意*i* ≥ 1，有(*M* ′*, s*′) |= *ϕ*

*i*

⇒ 对*π*上的任意*s*′（*i* ≥ 1），有(*M* ′*, s*′) |= FCTL(*φ,V* )

*i* *i*

⇒ 对*π*′上的任意*si*（*i* ≥ 1），有(*M , si*) |= FCTL(*φ,V* ) (IR(FCTL(*φ,V* )*,V* ))

⇒ 对任意*t* ∈ *S*，有(*M ,t*) |= FCTL(*φ,V* ) (*S* ∈ *M* )

⇒ (*M , s*) |= AGFCTL(*φ,V* )。

(⇐) (*M , s*) |= AGFCTL(*φ,V* )

⇒ 对*M* 上的每一条路径*π* = (*s* = *s*0*, s*1*, . . .* )，和任意*i* ≥ 0，有(*M , si*) |= FCTL(*φ,V* )

⇒ ∀*t* ∈ *S*，(*M ,t*) |= FCTL(*φ,V* ) (*S* ∈ *M* )

⇒ ∀*t* ∈ *S*，有(*M ,t*) ↔*V* (*M* ′*,t*′)且(*M* ′*,t*′) |= *φ*

⇒ ∀*si* ∈ *S* （*i* ≥ 0），有(*M , si*) ↔*V* (*M* ′*, s*′)且(*M* ′*, s*′) |= *φ* ，其中*M* ′ = (*S*′*, R*′*, L*′*,* [ ]′*, s*′)

*i i i i*

*i i i i i i*

⇒ (*M* \**, s*\*) ↔*V* (*M , s*)和(*M* \**, s*\*) |= AG*φ* ，其中*M* \* = (*S*\**, R*\**, L*\**,* [ ]*, s*\*)，

∙ *S*\* = J *S*′，

*i*

∙ *s*\* = *s*′ ，

0

∙ *R*\* = {(*s*′ *, s*′ ) | (*sx, sy*) ∈ *R, x, y* ≥ 0} ∪ J

*R*′，其中*R* ∈ *M* ，

*x y i*≥0 *i*

∙ 对任意*ti* ∈ *S*′，*L*\*(*ti*) = *L*′(*ti*)。

*i* *i*

⇒ (*M , s*) |= FCTL(AG*φ,V* )。

(vi) 可类似(v)来证明。

推论 **3.3.** 给定命题集合*V* ⊆ *A* ，命题公式*ϕ*和*CTL*公式AG*ψ*，则

FCTL(*ϕ* ∧ AG*ψ*) ≡ FCTL(*ϕ* ∧ *ψ,V* ) ∧ AGFCTL(*ψ,V* )*.*

证明**.** (*M , s*0) |= FCTL(*ϕ* ∧ AG*ψ*)

⇔ 存在(*M* ′*, s*′ ) ↔*V* (*M , s*0)，使得(*M* ′*, s*′ ) |= *ϕ* ∧ AG*ψ*

0 0

⇔ 存在(*M* ′*, s*′ ) ↔*V* (*M , s*0)，使得(*M* ′*, s*′ ) |= *ϕ* ∧ *ψ*且(*M* ′*, s*′ ) |= AG*ψ*

0 0 0

⇔ 存在(*M* ′*, s*′ ) ↔*V* (*M , s*0)，使得(*M* ′*, s*′ ) |= FCTL(*ϕ* ∧ *ψ,V* )且(*M* ′*, s*′ ) |= FCTL(AG*ψ,V* )

0 0 0

⇔ 存在(*M* ′*, s*′ ) ↔*V* (*M , s*0)，使得(*M* ′*, s*′ ) |= FCTL(*ϕ* ∧ *ψ,V* )且(*M* ′*, s*′ ) |= AGFCTL(*ψ,V* )

0 0 0

⇔ (*M , s*0) |= FCTL(*ϕ* ∧ *ψ,V* ) ∧ AGFCTL(*ψ,V* )。

**3.2** *µ***-**演算遗忘理论

*µ*-演算是一种表达能力较强的逻辑语言，它能表达CTL不能表达的一些性质，例 如：Kripke结构中有一条路径，在这条路径上，基数位置的状态满足公式¬*q* ∧ ¬*p*，但 是偶数位置的状态满足*q* ∧ *p*。 这一性质不能用CTL公式来表达，但是可以用*µ*-演算公 式表达如下：

*ϕ* = *νX.*(*p* ∧ *q*) ∧ EX(¬*p* ∧ ¬*q*) ∧ EXEX*X.*

这种情形在日常生活中是很常见的，如：偏序关系(N*,* ≤)（自然数集上的小于等于 关系）构成的Kripke结构，其基数节点为基数、偶数节点为偶数。事实上，CTL不能表 达具有有规律的性质[[115](#_bookmark229)]，其主要原因是

对 于 给 定 的 原 子 命 题*p*， 任 意 包 含*n*个“X” 时 序 算 子 的 命 题 时 序 公 式

（*proposinal temporal formula*，*PTL*），对 序列“*pi*(¬*p*)*pw*”有相同的真值，

其中*i > n*。

因而得出如下结论：

对任意*m* ≥ 2，性质“*p*在所有状态*si*上为真（*i* = *k* \* *m*，整数*k* ≥ 0）”不能

用*PTL*中的公式来表示。

本小节给出*µ*-演算遗忘的定义，并说明本文所定义的*µ*-演算遗忘与文章[[82](#_bookmark198)]中定义 的均匀插值具有对偶关系。 此时，本文给出的遗忘的性质无疑也是均匀插值所具有的 性质，这为*µ*-演算均匀插值的探索提供了另一种思路。 此外，借助于均匀插值的计算 方法，本文给出计算遗忘的方法。 形成了遗忘和均匀插值之间相辅相成的作用。

本小节的组织结构如下。 首先，给出Var-*V* -互模拟的定义及证明Var-*V* -互模拟对*µ*- 公式是不变的；根据Var-*V* -互模拟定义*µ*-演算遗忘；然后，探讨遗忘的性质，并给出 其与均匀插值的关系；最后给出与遗忘相关问题的复杂性。

**3.2.1 Var-**互模拟

与CTL情形下的遗忘相似，这里先给出*V* -互模拟的定义。令*Mi* = (*Si, Ri, Li, ri*)，*i*为

自然数集N中的元素。

定义 **3.4** (*V* -互模拟)**.** 给定原子命题集合*V* ⊆ *A* 和两个*Kripke*结构*M*1和*M*2。若下面几个

条件满足，则称*B* ⊆ *S*1 × *S*2是*M*1和*M*2的*V-*互模拟关系：

* *r*1*Br*2，

∙ 对任意*s* ∈ *S*1和*t* ∈ *S*2， 若*sBt*， 则对任意*p* ∈ *A* − *V* ， 有*p* ∈ *L*1(*s*)当且仅当*p* ∈

*L*2(*t*)，

∙ 若(*s, s*′) ∈ *R*1和*sBt*，则存在一个*t*′，使得*s*′*Bt*′和(*t,t*′) ∈ *R*2，且

∙ 若*sBt*和(*t,t*′) ∈ *R*2，则存在一个*s*′，使得(*s, s*′) ∈ *R*1和*t*′*Bs*′。

一方面，与CTL下的*V* -互模拟不同的是，这里要求*r*1*Br*2 （即：(*r*1*, r*2) ∈ *B*）。 如 果*M*1和*M*2之间存在一个*V* -互模拟关系*B*则称这两个Kripke结构*M*1和*M*2及由这两 个Kripke结构构成的结构(*M*1*, r*1)和(*M*2*, r*2)是*V* -互模拟的，分别记为*M*1 ↔*V M*2和(*M*1*, r*1) ↔*V* (*M*2*, r*2)。 显然，Kripke结构之间的*V* -互模拟与结构之间的*V* -互模拟是等价的： *M*1 ↔*V M*2当且仅当(*M*1*, r*1) ↔*V* (*M*2*, r*2)，因此在下文中只讨论Kripke结构之间*V* -互模 拟的性质，而结构之间的互模拟的性质与之相同。 不难看出初始结构之间的*V* -互模拟 是结构之间的*V* -互模拟的一个特例。

另一方面，*V* -互模拟与*L* -互模拟[2](#_bookmark0) [[102](#_bookmark216)]类似， 不同的是*V* -互模拟只考虑原子命 题，且当*L* 为原子命题集合时是*L* -互模拟补命题（这里默认除了原子命题之外的符 号都是相同的，因此只考虑原子命题）。 此外，已有结果表明*L* -句子（符号只出现 在*L* 中的*µ*-句子）关于*L* -互模拟是不变的，即若*M* 和*M* ′是*L* -互模拟的，则对于*L* - 句子*ϕ*有*M* |= *ϕ*当且仅当*M* ′ |= *ϕ* [[100](#_bookmark214),[102](#_bookmark216)]。 因此，若IR(*ϕ,V* )且*M* ↔*V M* ′，则*M* |= *ϕ*当 且仅当*M* ′ |= *ϕ*。 本文称这一性质为*V* -不变性。

例 **3.4.** 令*ϕ* = *νX.*(*ch* ∧ *j* ∧ EXEX*X* )，易证下面两个赋值(*M , v*)和(*M* ′*, v*′)（图 [3.2](#_bookmark64)中的两 个Kripke结构）是*ϕ*的模型，其中

∙ *M* = (*S, r, R, L*)，其中*S* = {*s*0*, s*1*, s*2}，*r* = *s*0，*L*(*s*0) = {*ch, j*}，*L*(*s*1) = {*ch*}，*L*(*s*2) =

0/ ，*R* = {(*s*0*, s*1)*,* (*s*1*, s*0)*,* (*s*0*, s*2)*,* (*s*2*, s*0)}，*v*(*X* ) = {*s*0}，

∙ *M* ′ = (*S*′*, r*′*, R*′*, L*′)，其中*S*′ = {*t*0*,t*1}*, r*′ = *t*0，*L*′(*t*0) = { *j*}，*L*′(*t*1) = 0/ ，*R*′ = {(*t*0*,t*1)*,* (*t*1*, t*0)}且*v*′(*X* ) = {*t*0}。

由于*M* 和*M* ′之间存在一个二元{*ch*}-互模拟*B* = {(*s*0*,t*0)*,* (*s*1*,t*1)*,* (*s*2*,t*1)}， 所以，

*M* ↔{*ch*} *M* ′。

*µ*-句子互模拟是不变的，*µ*-公式则不是[[116](#_bookmark230)]。这里定义一种对*µ*-公式是不变的互模

拟。

2*L* -互模拟是一种将定义[3.4](#_bookmark62)中的*V* 替换为*L* 的二元关系。

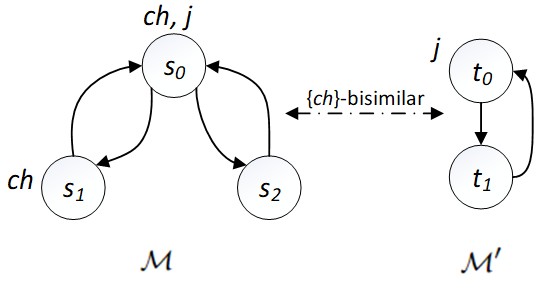


图 3.2: 两个{*ch*}-互模拟的Kripke结构示意图

定义 **3.5** (变元-命题-互模拟)**.** 给定*V* ⊆ *A* 、*Mi* = (*Si, ri, Ri, Li*)为*Kripke*结构、 *si* ∈ *Si*且*vi* :

*V* → 2*Si* ，其中*i* ∈ {1*,* 2}。 若关系*B* ⊆ *S*1 × *S*2满足：

∙ (*s*1*, s*2) ∈ *B*，

∙ *B*是*M*1 和*M*2之间的*V-*互模拟，且

∙ 对对任意(*t*1*,t*2) ∈ *B*和*X* ∈ *V* − *V*1，*t*2 ∈ *v*2(*X* )当且仅当*t*1 ∈ *v*1(*X* )，

则称*B*是(*M*1*, s*1*, v*1) 和(*M*2*, s*2*, v*2)之间的一个⟨*V*1*,V* ⟩-互模拟。

若(*M , s, v*)和(*M* ′*, s*′*, v*′)之间存在一个⟨*V*1*,V* ⟩-互模拟关系*B*，则称(*M , s, v*)和(*M* ′*, s*′*,*

*v*′)是⟨*V*1*,V* ⟩-互模拟的，记为(*M , s, v*) ↔⟨*V*1*,V* ⟩ (*M* ′*, s*′*, v*′)。 若(*M , s, v*)和(*M* ′*, s*′*, v*′)之间存 在一个⟨0/ *,V*1⟩-互模拟，则称(*M , s, v*)和(*M* ′*, s*′*, v*′)是*Var-V*1*-*互模拟的，记为(*M , s, v*) ↔*V*1 (*M* ′*, s*′*, v*′)；称⟨0/ *,V*1⟩-互模拟为Var-*V*1-互模拟。此外，若*s* = *r* 且*s*′ = *r*′，则(*M , s, v*) ↔⟨*V*1*,V* ⟩ (*M* ′*, s*′*, v*′)简写为(*M , v*) ↔⟨*V*1*,V* ⟩ (*M* ′*, v*′)。

*M* ↔*V M* ′但(*M , v*) ̸↔*V* (*M* ′*, v*′)有时是成立的。 例：例 [3.4](#_bookmark63)中的两个Kripke结构*M*

和*M* ′ 是0/ -互模拟的。然而，由于*r*1 ∈ *v*(*X* )和(*r*1*, r*′ ) ∈ *B*，但是*r*′ ̸∈ *v*′(*X* )，所以(*M , v*) ̸↔0/

2 2

(*M* ′*, v*′)。 对*S*1 ⊆ *S*和二元关系*B* ⊆ *S* × *S*′，记*B*(*S*1) = {*s*′ | (*s, s*′) ∈ *B, s* ∈ *S*1}。

显然，若(*M , s, v*) ↔*V* (*M* ′*, s*′*, v*′)，则*M* ↔*V M* ′。 再者，令*v*(*X* ) = *S*且*v*′(*X* ) = *S*′， Kripke结构*M* 和*M* ′之间的任意*V* -互模拟可以容易地扩展为(*M , r, v*)和(*M* ′*, r*′*, v*′)之间 的Var-*V* -互模拟，其中*S*（*S*′）为*M* （*M* ′）的状态集。

例 **3.5** (例 [3.4](#_bookmark63)的延续)**.** 令*M* 和*M* ′为图 [3.2](#_bookmark64)中的Kripke结构，*v* : *V* → 2*S*和*v*′ : *V* → 2*S*′ 为

将*V* 中的变元分别赋值到*M* 和*M* ′的状态集上的赋值函数。可以检查下面的结论成立：

∙ 若对任意*X* ∈ *V* ，*v*(*X* ) = {*s*0*, s*1*, s*2} 且*v*′(*X* ) = {*t*0*,t*1}，则(*M , v*) ↔{*ch*} (*M* ′*, v*′)；

∙ 若对任意*X* ∈ *V* − {*X*1}，*v*(*X*1) = {*s*0}、 *v*′(*X*1) = {*t*1}、 *v*(*X* ) = {*s*0*, s*1*, s*2}且*v*′(*X* ) =

{*t*0*,t*1}，则(*M , v*) ̸↔{*ch*} (*M* ′*, v*′)；这是因为(*s*0*,t*0) ∈ *B*且*s*0 ∈ *v*(*X*1)，但是*t*0 ∈*/ v*′(*X*1)。

显然，对任意集合*V* ⊆ *A* ，每个Var-*V* -互模拟与一个*V* -互模拟对应，即：(*M , v*) ↔*V*

(*M* ′*, v*′) 蕴 涵*M* ↔*V M* ′。 再 者， 对 任 意*X* ∈ *V* ， 令*v*(*X* ) = *S*且*v*′(*X* ) = *S*′， Kripke结

构*M* 和*M* ′之间的任意*V* -互模拟可以容易地扩展为(*M , r, v*)和(*M* ′*, r*′*, v*′)之间的Var-*V* -

互模拟，其中*S*和*S*′分别为*M* 和*M* ′中的状态集合。

可以容易证明↔*V* 有如下性质。

命题 **3.6.** 令*V*1*, V*2 ⊆ *V* 、*V,V*1 ⊆ *A* 且*Mi*为*Kripke*结构（*i* = 1*,* 2*,* 3），若*vi* : *V* → 2*Si* ，则：

*(i)* ↔⟨*V*1*,V* ⟩为赋值间的等价关系。

*(ii)* 若(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*2*, s*2*, v*2)、*V*1 ⊆ *V*2且*V* ⊆ *V*1，

则(*M*1*, s*1*, v*1) ↔⟨*V*2*,V*1⟩ (*M*2*, s*2*, v*2)；

*(iii)* 若(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*2*, s*2*, v*2)且(*M*2*, s*2*, v*2) ↔⟨*V*2*,V*1⟩ (*M*3*, s*3*, v*3)，

则(*M*1*, s*1*, v*1) ↔⟨*V*1∪*V*2*,V* ∪*V*1⟩ (*M*3*, s*3*, v*3)。

证明**.** (i) 这里从自反性、对称性和传递性来证明该关系是一个等价关系。

(1) ↔*V* 是自反的。 显然，对任意的赋值(*M , s, v*)都有(*M , s, v*) ↔⟨*V*1*,V* ⟩ (*M , s, v*)。

(b) 对称性：下证对任意(*M*1*, s*1*, v*1)和(*M*2*, s*2*, v*2)，若(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*2*, s*2*, v*2)， 则(*M*2*, s*2*, v*2) ↔⟨*V*1*,V* ⟩ (*M*1*, s*1*, v*1)。假定(*M*1*, s*1*, v*1) 和(*M*2*, s*2*, v*2) 有⟨*V*1*,V* ⟩-互模拟关系*B*， 如下构造关系*B*1：*B*1 = {(*s,t*) | (*t, s*) ∈ *B* with *t* ∈ *S*1 and *s* ∈ *S*2}。 下证*B*1是*M*2*M*1之间 的*V* -互模拟关系。

∙ 因为(*r*1*, r*2) ∈ *B*，所以，(*r*2*, r*1) ∈ *B*1；

∙ 对任意*s* ∈ *S*2 和*t* ∈ *S*1，若(*s,t*) ∈ *B*1，则(*t, s*) ∈ *B*。 因此，对任意*p* ∈ *A* − *V* ， *p* ∈ *L*1(*t*)当且仅当*p* ∈ *L*2(*s*)；

∙ 因为*B*是*M*1和*M*2之间的*V* -互模拟关系，所以*V* -互模拟的第三和第四个点很容易

能够证明。

此外，若(*s,t*) ∈ *B*，则对任意*X* ∈ *V* − *V*1，*s* ∈ *v*1(*X* ) 当且仅当*t* ∈ *v*2(*X* )。因此，若(*t, s*) ∈ *B*1， 则对任意*X* ∈ *V* − *V*1，*t* ∈ *v*2(*X* ) 当且仅当*s* ∈ *v*1(*X* )。 所以，(*M*2*, s*2*, v*2) ↔⟨*V*1*,V* ⟩ (*M*1*, s*1*, v*1)。

(c) 传递性：下证若(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*2*, s*2*, v*2)和(*M*2*, s*2*, v*2) ↔⟨*V*1*,V* ⟩ (*M*3*, s*3*, v*3)，

则(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*3*, s*3*, v*3)。(*M*1*, s*1*, v*1) (*M*2*, s*2*, v*2)有⟨*V*1*,V* ⟩-互模拟关系*B*1，(*M*2*, s*2*, v*2) (*M*3*, s*3*, v*3)有⟨*V*1*,V* ⟩-互模拟关系*B*2，如下构造*B*：*B* = {(*s, z*) | (*s,t*) ∈ *B*1 and (*t, z*) ∈ *B*2 with *s* ∈ *S*1*,t* ∈ *S*2*,* and *z* ∈ *S*3}。 可以类似(b)证明*B*是*M*1和*M*3之间的*V* -互模拟关系。 因此，*M*1 ↔⟨*V*1*,V* ⟩ *M*3。

此外，若(*s,t*) ∈ *B*，则对任意*X* ∈ *V* − *V*1，*s* ∈ *v*1(*X* ) 当且仅当*t* ∈ *v*2(*X* )。 因此， 若(*t, s*) ∈ *B*1，则对任意*X* ∈ *V* − *V*1，*t* ∈ *v*2(*X* ) 当且仅当*s* ∈ *v*1(*X* )。 所以，(*M*2*, v*2) ↔*V* (*M*1*, v*1)。

(ii) 假定*BV* 是(*M*1*, s*1*, v*1)和(*M*2*, s*2*, v*2)之间的⟨*V*1*,V* ⟩-互模拟关系。下证*BV* 也是(*M*1*, s*1*, v*1)和(*M*2*, s*2*, v*2)之间的⟨*V*2*,V*1⟩-互模拟关系。 显然：

∙ (*r*1*, r*2) ∈ *BV* ；

∙ 对任意*w*1 ∈ *S*1和*w*2 ∈ *S*2，若(*w*1*, w*2) ∈ *BV* ，则对任意*p* ∈ *A* − *V*1， *p* ∈ *L*1(*w*1)当且 仅*p* ∈ *L*2(*w*2)（因为*V* ⊆ *V*1且对任意*p* ∈ *A* − *V* ，*p* ∈ *L*1(*w*1)当且仅当*p* ∈ *L*2(*w*2)）；

∙ 因为*BV* 是*M*1 and *M*2之间的*V* -互模拟，所以*V* -互模拟的第三和第四个点很容易

能够证明。

因此，*BV* 是*M*1 and *M*2之间的*V*1-互模拟。

此外，若(*s,t*) ∈ *BV* ，则对任意*X* ∈ *V* − *V*2，*s* ∈ *v*1(*X* ) 当且仅当*t* ∈ *v*2(*X* )。 因此，

(*M*1*, s*1*, v*1) ↔⟨*V*2*,V*1⟩ (*M*2*, s*2*, v*2)。

(iii) 由(ii)可知，若(*M*1*, s*1*, v*1) ↔⟨*V*1*,V* ⟩ (*M*2*, s*2*, v*2)，则(*M*1*, s*1*, v*1) ↔⟨*V*1∪*V*2*,V* ∪*V*1⟩ (*M*2*, s*2*, v*2)。 因此，由(i)可知(*M*1*, s*1*, v*1) ↔⟨*V*1∪*V*2*,V* ∪*V*1⟩ (*M*3*, s*3*, v*3)。

不难看出，命题 [3.6](#_bookmark65)对于Kripke结构之间的↔*V* 关系也是成立的。 直观地说，(i)表 示↔⟨*V*1*,V* ⟩是赋值之间的自反、 对称和传递关 系。 (ii)指明若两个赋值是⟨*V*1*,V* ⟩-互模 拟的，则对于*V* （*V*1）的任意一个超级*V*1 （*V*2），这两个赋值都是⟨*V*2*,V*1⟩-互模拟的。 (iii)表示如果一个Kripke结构和其它两个Kripke结构互相⟨*V*1*,V* ⟩和⟨*V*2*,V*1⟩互模拟，则这 两个Kripke结构⟨*V*1 ∪ *V*2*,V* ∪ *V*1⟩-互模拟。

引理 **3.3.** 令*M* = (*S, r, R, L*)、*ϕ*为*µ-*公式且*S*1 ⊆ *S*2 ⊆ *S*。若*X* 正出现在*ϕ*中，则‖*ϕ*‖*M* ⊆

*v*[*X* :=*S*1]

*v*[*X* :=*S*2]*.*

‖*ϕ*‖

*M*

证明**.** 通过归纳*ϕ*的结构来证明。

基始**.** (a) *ϕ* = *p*，其中*p* ∈ *A* 。

‖*ϕ*‖*v*[*X* :=*S*1] = {*p* | *p* ∈ *L*(*s*)}

*M*

*v*[*X* :=*S*2] *.*

= ‖*ϕ*‖

*M*

(b) *ϕ* = *Y* ，其中*Y* ∈ *V* − {*X* }。

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

= *v*[*X* := *S*1](*Y* )

= *v*[*X* := *S*2](*Y* )

*v*[*X* :=*S*2] *.*

= ‖*ϕ*‖

*M*

(c) *ϕ* = *X*。

‖*ϕ*‖*v*[*X* :=*S*1] = *v*[*X* := *S*1](*X* )

*M*

= *S*1

⊆ *S*2

= *v*[*X* = *S*2](*X* )

*v*[*X* :=*S*2] *.*

= ‖*ϕ*‖

*M*

归纳步**.** (a) *ϕ* = *ϕ*1 \* *ϕ*2，其中\* ∈ {∨*,* ∧}且*X* 正出现在*ϕi*中（*i* = 1*,* 2）。

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

*M*

= ‖*ϕ*1

‖*v*[*X* :=*S*1]

*M*

\* ‖*ϕ*2‖

*M*

*v*[*X* :=*S*1]

*M*

⊆ ‖*ϕ*1‖*v*[*X* :=*S*2] \* ‖*ϕ*2‖*v*[*X* :=*S*2] (归纳假设)

*v*[*X* :=*S*2] *.*

= ‖*ϕ*‖

*M*

(b) *ϕ* = AX*ϕ*1，且*X* 正出现在*ϕ*1中。

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

= {*s* | ∀*s*′*.*(*s, s*′) ∈ *R* ⇒ *s*′ ∈ ‖*ϕ*1 *M* }

*M*

‖*v*[*X* :=*S*1]

⊆ {*s* | ∀*s*′*.*(*s, s*′) ∈ *R* ⇒ *s*′ ∈ ‖*ϕ*1‖*v*[*X* :=*S* ]} (归纳假设)

2

*M*

= ‖*ϕ .*

‖*v*[*X* :=*S*2]

(c) *ϕ* = EX*ϕ*1，且*X* 正出现在*ϕ*1中。

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

= {*s* | ∃*s*′*.*(*s, s*′) ∈ *R* ∧ *s*′ ∈ ‖*ϕ*1 *M* }

*M*

‖*v*[*X* :=*S*1]

⊆ {*s* | ∃*s*′*.*(*s, s*′) ∈ *R* ∧ *s*′ ∈ ‖*ϕ*1‖*v*[*X* :=*S* ]} (归纳假设)

2

*v*[*X* :=*S*2] *.*

= ‖*ϕ*‖

*M*

(d) *ϕ* = *νY.ϕ*1，且*X* 正出现在*ϕ*1中。

(d.1) *Y* = *X* .

*M*

‖*ϕ*‖*v*[*X* :=*S*1] =

*M*

｜

{*S*′

⊆ *S* | *S*′

⊆ ‖*ϕ*1‖*v*[*X* :=*S*1][*X* :=*S*′]}

第三章 CTL和*µ*-演算遗忘理论

= ｜ *M*

{*S*′ ⊆ *S* | *S*′ ⊆ ‖*ϕ*1‖*v*[*X* :=*S*′]}

= ｜ *v*[*X* :=*S* ][*X* :=*S* ]

{*S*′ ⊆ *S* | *S*′ ⊆ ‖*ϕ*1‖*M* }

2

′

*M*

= ‖*ϕ .*

‖*v*[*X* :=*S*2]

(d.2) *Y* ̸= *X* .

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

｜ ′ ′ *M*

=

*S* ⊆ *S* | *S* ⊆ ‖*ϕ*1‖

{ *v*[*X* :=*S*1][*Y* :=*S*′]}

｜

′ ′ *M*

{*S* ⊆ *S* | *S* ⊆ ‖*ϕ*1‖

⊆ *v*[*X* :=*S*2][*Y* :=*S*′]

} (归纳假设)

*M*

= ‖*ϕ .*

‖*v*[*X* :=*S*2]

(e) *ϕ* = *µX.ϕ*1，且*X* 正出现在*ϕ*1中。

(e.1) *Y* = *X* .

*M*

‖*ϕ*‖

*v*[*X* :=*S*1]

飞 *M*

{ 1

=

*S*′ ⊆ *S* | ‖*ϕ*1‖*v*[*X* :=*S* ][*X* :=*S*′]

= 飞 *M*

⊆ *S*′}

{*S*′ ⊆ *S* | ‖*ϕ*1‖*v*[*X* :=*S*′] ⊆ *S*′}

= 飞 *M*

{*S*′ ⊆ *S* | ‖*ϕ*1‖*v*[*X* :=*S* ][*X* :=*S*′] ⊆ *S*′}

2

*M*

= ‖*ϕ .*

‖*v*[*X* :=*S*2]

(e.2) *Y* ̸= *X* . 注意，对任意*µ*-公式*ψ*，‖*ψ*‖*M*

*v*[*X* :=*S*]

⊆ *S*。

‖*ϕ*‖*M*

= 飞 *S*′ ⊆ *S* | ‖*ϕ*1‖

⊆ *S*′}

*v*[*X* :=*S*1] {

*M*

飞

*v*[*X* :=*S*1][*Y* :=*S*′]

*M*

⊆ {*S*′ ⊆ *S* | ‖*ϕ*1‖*v*[*X* :=*S* ][*Y* :=*S*′] ⊆ *S*′} (归纳假设)

2

*M*

= ‖*ϕ .*

‖*v*[*X* :=*S*2]

命题 **3.7** (不变性)**.** 令*ϕ*为*µ-*公式、*V*1 ⊆ *V* 且*V* ⊆ *A* 。若(*M , s, v*) ↔⟨*V*1*,V* ⟩ (*M* ′*, s*′*, v*′)且*IR*(*ϕ,V* ∪

*V*1)，则(*M , s, v*) |= *ϕ*当且仅当(*M* ′*, s*′*, v*′) |= *ϕ*。

证明**.** 令*M* = (*S, r, R, L*) 且*M* ′ = (*S*′*, r*′*, R*′*, L*′)。假设*Var*(*ϕ*)∩(*V* ∪*V*1) = 0/ 且(*M , s, v*)和(*M* ′*, s*′*, v*′)之

间的⟨*V*1*,V* ⟩-互模拟关系为*B*。 基始**.** (a) *ϕ* = *p*，其中*p* ̸∈ *V*。 (*M , s, v*) |= *ϕ*

⇔ *p* ∈ *L*(*s*)

贵州大学博士学位论文

⇔ *p* ∈ *L*(*s*′) ((*M , s, v*) ↔⟨*V*1*,V* ⟩ (*M* ′*, s*′*, v*′), *p* ̸∈ *V* )

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

(b) *ϕ* = *X*，其中*X* ̸∈ *V*1。

(*M , s, v*) |= *ϕ*

⇔ *s* ∈ *v*(*X* )

⇔ *s*′ ∈ *v*′(*X* ) ((*s, s*′) ∈ *B*, *X* ̸∈ *V*1)

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。 归纳步**.** (a) *ϕ* = ¬*ϕ*1。 (*M , s, v*) |= *ϕ*

⇔ (*M , s, v*) |= *ϕ*1

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*1 (归纳假设)

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

1. *ϕ* = *ϕ*1 ∨ *ϕ*2。

(*M , s, v*) |= *ϕ*

⇔ (*M , s, v*) |= *ϕ*1或(*M , s, v*) |= *ϕ*2

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*1或(*M* ′*, s*′*, v*′) |= *ϕ*2 (归纳假设)

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

1. *ϕ* = AX*ϕ*1。

(*M , s, v*) |= *ϕ*

⇔ 对任意(*s, s*1) ∈ *R*，由(*M , s*1*, v*) |= *ϕ*1

⇔ 对任意(*s*′*, s*′ ) ∈ *R*′，存在(*s, s*1) ∈ *R*，使得(*s*1*, s*′ ) ∈ *B* ((*M , s, v*) ↔⟨*V ,V* ⟩ (*M* ′*, s*′*, v*′))

1 1 1

⇔ 对任意(*s*′*, s*′ ) ∈ *R*′，有(*M* ′*, s*′ *, v*′) |= *ϕ*1 (归纳假设)

1 1

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

1. *ϕ* = *νX.ϕ*1。

(*M , s, v*) |= *ϕ*

⇒ *s* ∈ J{*S*1 ⊆ *S* | *S*1 ⊆ ‖*ϕ*1‖*M* }

*v*[*X* :=*S*1]

⇒ 存在*S*1 ⊆ *S*，使得*S*1 ⊆ ‖*ϕ*1‖*v*[*X* :=*S*1] 和*s* ∈ *S*1

*M*

⇒ 对任意*w* ∈ *S*2 = *S*1 ∪ {*s*2 | (*t, si*) ∈ *B, s*2 ∈ *S*和*s*1 ∈ *S*1，其中*i* = 1*,* 2}，存在*w*′ ∈ *S*1使

得(*M , w, v*) ↔⟨*V*1*,V* ⟩ (*M , w*′*, v*) (命题 [3.6](#_bookmark65))

⇒ 对任意*w* ∈ *S*2 = *S*1 ∪ {*s*2 | (*t, si*) ∈ *B, s*2 ∈ *S*和*s*1 ∈ *S*1，其中*i* = 1*,* 2}，存在*w*′ ∈ *S*1使

得(*M , w, v*[*X* := *S*2]) ↔⟨*V*1*,V* ⟩ (*M , w*′*, v*[*X* := *S*2])

⇒ 对任意*w* ∈ *S*2，有*w* ∈ ‖*ϕ*1‖*M*

*v*[*X* :=*S*2]

⇒ 存在*S*2 ⊆ *S*，使得*S*2 ⊆ ‖*ϕ*1‖*v*[*X* :=*S*2]

*M*

*M*

*v*[*X* :=*S*1]

(*S*1 ⊆ ‖*ϕ*1‖

, 归纳假设, 引理 [3.3](#_bookmark66))

⇒ 对任意*u*′ ∈ *B*(*S*2)，存在*u* ∈ *S*2使得(*M* ′*, u*′*, v*′[*X* := *B*(*S*2)]) ↔⟨*V*1*,V* ⟩ (*M , u, v*[*X* := *S*2])

第三章 CTL和*µ*-演算遗忘理论

⇒ 对任意*u*′ ∈ *B*(*S*2)，有*u*′ ∈ ‖*ϕ*1‖*v*′[*X* :=*B*(*S*2)] (归纳假设)

*M* ′

⇒ 存在*B*(*S*2) ⊆ *S*′，使得*B*(*S*2) ⊆ ‖*ϕ*1‖*v*′[*X* :=*B*(*S*2)] 且*s*′ ∈ *B*(*S*2) (*s* ∈ *S*2)

*M* ′

*M* ′

⇒ *s*′ ∈ J{*S*′ ⊆ *S*′ | *S*′ ⊆ ‖*ϕ*1‖ }

1 1 *v*′[*X* :=*S*′ ]

1

⇒ (*M* ′*, s*′*, v*′) |= *ϕ*。

另一个方向可类似地证明。

命题 [3.7](#_bookmark67)表明，若两个赋值是Var-*V* -互模拟的，则这两个赋值同时满足（或不满 足）同一个与*V* 无关的公 式。 例如，(*Mi, vi*) *i* = 1*,* 2为两个赋值， *ϕ* = *X*，*V* = {*p*}； 若(*M*1*, v*1) ↔*V* (*M*2*, v*2)，则可以证明(*M*1*, v*1) |= *ϕ* 当且仅当(*M*2*, v*2) |= *ϕ*。

此时，*µ*-演算下的遗忘如下定义：

定义 **3.6** (*µ*-演算下的遗忘)**.** 令*V* ⊆ *A* 和*ϕ*为*µ-*公式。 若*Var*(*ψ*) ∩ *V* = 0/ 且下面等式成立，

则称*ψ*是从*ϕ*中遗忘*V* 后得到的结果：

*Mod*(*ψ*) = {(*M , v*) | ∃(*M* ′*, v*′) ∈ *Mod*(*ϕ*) 且(*M* ′*, v*′) ↔*V* (*M , v*)}。

显然，当讨论的公式为*µ*-句子的时候，定义 [3.6](#_bookmark68)中的模型变成*M* 和*M* ′，即去掉赋

值函数*v*和*v*′。

定义 [3.6](#_bookmark68)表明如果*ψ*和*ψ*′都是从*ϕ*中遗忘*V* 中的原子命题得到的结果，则*Mod*(*ψ*)

= *Mod*(*ψ*′)，也就是说遗忘的结果之间是语义等价的（即有相同的模型）。 此时，将遗 忘的结果记为F*µ* (*ϕ,V* )，除非另有说明，否则F*µ* (*ϕ,V* )表示从*ϕ*中遗忘*V* 的结果是*µ*-公 式。

**3.2.2** 遗忘算子及其性质

这部分展示*µ*-演算下遗忘的语义性质。 特别地，这里将证明上述*µ*-演算下的遗忘 的定义与遗忘的那几条规则具有“当且仅当的关系”，且从任意*µ*-句子中遗忘任意原 子命题集的结果总是一个*µ*-句子。 此外，也研究了遗忘算子的代数性质，包括分解性

（decomposition）、切片性（slice）和同质性（homogeneity）。

定理 **3.4.** 给定原子命题*q* ∈ *A* 和*µ-*句子*ϕ*， 则存在一个*µ-*句子*ψ*使得*Var*(*ψ*) ∩ {*q*} =

0/ 且*ψ* ≡ F*µ* (*ϕ,* {*q*})。

证明**.** 已有结果表明，对任意*µ*-句子*ϕ*和原子命题*p*，存在一个{*p*}-无关的*µ*-句子*ϕ*′

（即IR(*ϕ*′*,* {*p*})）使得[[102](#_bookmark216)]：

*M* |= *ϕ*′ 当且仅当∃*M* ′ ∈ *Mod*(*ϕ*) 使得*M* ↔{*p*} *M* ′*.*

这与本文遗忘的定义一致，因此上述结论成立。

与模态S5和CTL情形类似，下面给出*µ*-演算下遗忘的基本公设：

(**W**) 削弱：*ϕ* |= *ϕ*′;

(**PP**) 正支持：对任意*µ*-句子*η*，若IR(*η,V* )和*ϕ* |= *η*则*ϕ*′ |= *η*； (**NP**) 负支持：对任意*µ*-句子*η*，若IR(*η,V* )和*ϕ* |= *η*则*ϕ*′ |= *η*； (**IR**) 无关性：IR(*ϕ*′*,V* )。

其中*V* ⊆ *A* 、*ϕ*为*µ*-公式、*ϕ*′是从*ϕ*中遗忘*V* 后得到的结果。

定理 **3.5** (表示性定理)**.** 给定*µ-*公式*ϕ*、 *ϕ*′和*φ* ，*V* ⊆ *A* 为原子命题集。 下面的几个陈述

是等价的：

*(i) ϕ*′ ≡ F*µ* (*ϕ,V* )*,*

*(ii) ϕ*′ ≡ {*φ* | *ϕ* |= *φ* 且*IR*(*φ,V* )}*,*

*(iii)* 若*ϕ*、 *ϕ*′及*V* 和*(i)*、 *(ii)*中的符号表示相同公式和原子命题集，则*(****W****)*、 *(****PP****)*、 *(****NP****)*

和*(****IR****)* 成立。

证明**.** (*i*) ⇔ (*ii*). 为了证明这一结论成立，只需证明：

*Mod*(F*µ* (*ϕ,V* )) = *Mod*({*φ* | *ϕ* |= *φ,* IR(*φ,V* )})*.* (⇒) 对F*µ* (*ϕ,V* )的任意模型(*M* ′*, v*′)

⇒ ∃(*M , v*)使得(*M , v*) |= *ϕ*和(*M , v*) ↔*V* (*M* ′*, v*′) (定义[3.6](#_bookmark68))

⇒ 对于任意与*V* -无关且*ϕ* |= *φ* 的*φ* 都有(*M* ′*, v*′) |= *φ*

⇒ (*M* ′*, v*′) |= {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}

(⇐) 由于IR(F*µ* (*ϕ,V* )*,V* )和*ϕ* |= F*µ* (*ϕ,V* )， 由定义[3.6](#_bookmark68)可知{*φ* | *ϕ* |= *φ,* IR(*φ,V* )} |=

F*µ* (*ϕ,V* )。

(*ii*) ⇒ (*iii*). 为了方便，令*A* = {*φ* | *ϕ* |= *φ,* IR(*φ,V* )}。 首先，对任意*A*中的公式*φ* ′都 有IR(*φ* ′*,V* )，所以有IR(*A,V* )。 因此，IR(*ϕ*′*,V* )。 其次，对任意*φ* ′ ∈ *A*，都有*ϕ* |= *φ* ′，所 以*ϕ* |= *ϕ*′。 第三，∀*φ* 且IR(*φ,V* )，若*ϕ* |= *φ* 则*φ* ∈ *A*，因而*ϕ*′ |= *φ*。 最后，∀*φ* 且IR(*φ,V* )， 若*ϕ* |= *φ* 则*φ* ̸∈ *A*。 因此，由定义[3.6](#_bookmark68)和*V* -无关性可知*ϕ*′ |= *φ*。

|  |  |  |
| --- | --- | --- |
| (*iii*) ⇒ (*ii*). (1) *ϕ*′ |= {*φ* | *ϕ* |= *φ,* IR(*φ,V* )} |  | ((**PP**)) |
| (2) {*φ* | *ϕ* |= *φ,* IR(*φ,V* )} |= *ϕ*′ |  | ((**W**)，(**IR**)) |
| ⇒ *ϕ*′ ≡ {*φ* | *ϕ* |= *φ,* IR(*φ,V* )} | ((1)，(2)). |  |

定理[3.5](#_bookmark71) 表明基本公设对*µ*-演算下的遗忘是充分且必要的：基本公设能描述遗忘的

结果，遗忘的结果具有基本公设里的性质。 这与S5和CTL下的情形相同。

D’Agostino 研究了*µ*-演算下的均匀插值，并指出*µ*-算具有均匀插值性质[[82](#_bookmark198),[102](#_bookmark216)-[103](#_bookmark217)]。

换 句 话 说， 这 意 味 着 对 任 意*µ*-句 子*ϕ*和 有 限 的 原 子 命 题 集*V* ⊆ *Var*(*ϕ*)， 都 存 在 一 个*V* -无关且与*ϕ*最接近的*µ*-句子～∃*V ϕ*。 值得注意的是， 对于*µ*-句子*ϕ*， 上述定义的

遗忘F*µ* (*ϕ,V* )与～∃*V ϕ* [[82](#_bookmark198)] 语义等价。

推论 **3.4.** 令*V* ⊆ *A* 且*ϕ* 为*µ-*句子，则～∃*V ϕ* ≡ F*µ* (*ϕ,V* )。

证明**.** 显然*ϕ* |= F*µ* (*ϕ,V* ) 且F*µ* (*ϕ,V* ) ∩ *V* = 0/。 此外，对任意*φ* ，若*ϕ* |= *φ* 且IR(*φ,V* )，由

定理 [3.5](#_bookmark71)可知F*µ* (*ϕ,V* ) |= *φ*。

因此，由均匀插值的定义可知F*µ* (*ϕ,V* )是*ϕ* 关于*Var*(*ϕ*) − *V* 的均匀插值。

除了上述的表示性定理，后文将说明公设(**IR**)对计算SNC 和WSC是重要的。 对 于*µ*-句子*ψ* = *ϕ* ∧ (*q* ↔ *α*)，*ϕ* ∧ *α*是{*q*}-无关的，则从*ψ*中遗忘*q*得到的结果是*ϕ*。 正如 将在第[四](#_bookmark82)章中展示，这一性质有助于将任意公式的SNC（WSC）转换为命题下的SNC

（WSC）。 这一性质可形式化如下：

引理 **3.4.** 令*ϕ*和*α*为两个*µ-*公式，*q*为原子命题且*q* ̸∈ *Var*(*ϕ*) ∪ *Var*(*α*)。 则F*µ* (*ϕ* ∧ (*q* ↔

*α*)*, q*) ≡ *ϕ*。

证明**.** 令*ϕ*′ = *ϕ* ∧(*q* ↔ *α*)。对F*µ* (*ϕ*′*, q*)的任意一个模型(*M , v*)，存在一个赋值(*M* ′*, v*′) 使 得(*M , v*) ↔{*q*} (*M* ′*, v*′) 且(*M* ′*, v*′) |= *ϕ*′。显然，(*M* ′*, v*′) |= *ϕ*，又因为IR(*ϕ,* {*q*}) 且(*M , v*) ↔{*q*} (*M* ′*, v*′)，所以(*M , v*) |= *ϕ*。

令(*M , v*) ∈ *Mod*(*ϕ*)，其中*M* = (*S, s, R, L*)。如下构造(*M* ′*, v*′)，其中*M* ′ = (*S, s, R, L*′)且：

*L*′ : *S* → 2*A* 且∀*s*\* ∈ *S,* 若(*M , s*\*) |= *α*，则*L*′(*s*\*) = *L*(*s*\*) − {*q*}，

否则*L*′(*s*\*) = *L*(*s*\*) ∪ {*q*}，

*L*′(*s*) = *L*(*s*) ∪ {*q*}若(*M , s*) |= *α*，否则*L*′(*s*) = *L*(*s*) − {*q*}。

显然，*M* ′ ↔{*q*} *M* ，所以*M* 和*M* ′之间存在一个{*q*}-互模拟关系*B*。 再者，对任意*X* ∈

*V* ，令*v*′(*X* ) = *B*(*v*(*X* ))。

所以，(*M* ′*, v*′) |= *ϕ*、 (*M* ′*, v*′) |= *q* ↔ *α*且(*M , v*) ↔{*q*} (*M* ′*, v*′)。 因此，(*M* ′*, v*′) |=

*ϕ* ∧ (*q* ↔ *α*)。 又因为(*M , v*) ↔{*q*} (*M* ′*, v*′) 且IR(F*µ* (*ϕ* ∧ (*q* ↔ *α*)*, q*)*,* {*q*})，从而(*M , v*) |=

F*µ* (*ϕ* ∧ (*q* ↔ *α*)*,* {*q*})。

正如在第[一](#_bookmark4)章中所说的，遗忘在经典命题逻辑中首先被提出，并应用于各种领域。

首先， 回顾一下从命题公式*ϕ*中遗忘原子命题*p*得到的结果为*Forget*(*ϕ,* {*p*}) ≡ *ϕ*[*p/*⊥] ∨ *ϕ*[*p/*⊤]，且*Forget*(*ϕ,V* ∪ {*p*})被递归地定义为：*Forget*(*Forget*(*ϕ,* {*p*})*,V* )，其 中*Forget*(*ϕ,* 0/ ) = *ϕ*。 此外，对于给定的Kripke结构*M* = (*S, R, L, r*)和命题公式*ψ*，*M* |= *ψ*当且仅当*L*(*r*) |= *ψ*。 经典命题逻辑与*µ*-演算下的遗忘之间的联系如下：

定理 **3.6.** 令*ϕ*为命题公式，*V* ⊆ *A* 为原子命题集，则

F*µ* (*ϕ,V* ) ≡ *Forget*(*ϕ,V* )*.*

证明**.** 令*M* = (*S, R, L, r*)和*M* ′ = (*S*′*, R*′*, L*′*, r*′)为Kripke结构。

(⇒) 对任意*M* ∈ *Mod*(F*µ* (*ϕ,V* ))

⇒ 由定义[3.3](#_bookmark58)可知存在*M* ′ ∈ *Mod*(*ϕ*)使得*M* ↔*V M* ′，且*M* 和*M* ′之间的*V* -互模拟关系

为*B*

⇒ *rBr*′

⇒ *M* |= *Forget*(*ϕ,V* ) (IR(*Forget*(*ϕ,V* )*,V* ), *V* -无关性)

(⇐) 对任意*M* ∈ *Mod*(*Forget*(*ϕ,V* ))

⇒ ∃*M* ′ ∈ *Mod*(*ϕ*)使得∀*p* ∈ *A* − *V* ，*p* ∈ *L*(*r*)当且仅当*p* ∈ *L*′(*r*′) (*Forget*的定义)

如下构造Kripke 结构*M*1 = (*S*1*, R*1*, L*1*, r*1)：

\* *S*1 = (*S* − {*r*}) ∪ {*r*1},

\* *R*1与*R*相同，除了*r*被*r*1替换，且

\* *L*1与*L*相同，除了*L*1(*r*1) = *L*′(*r*′)。

⇒ *M*1 |= *ϕ*且*M*1 ↔*V M*

⇒ *M* |= F*µ* (*ϕ,V* ) (IR(F*µ* (*ϕ,V* )*,V* ), *V* -无关性）

定理[3.6](#_bookmark73)表明*µ*-演算下的遗忘是命题逻辑下遗忘的扩展，这提示我们是否命题情形 下遗忘拥有的性质*µ*-演算下的遗忘也具有。 下面的性质在命题逻辑、S5[[48](#_bookmark176)] 和CTL中都 成立，这里证明其在*µ*-演算中也成立。

命题 **3.8.** 给定*µ-*公式*ϕ*、*ϕi*和*ψi*（*i* = 1*,* 2），*V* ⊆ *A* 为原子命题集。 则：

*(i)* F*µ* (*ϕ,V* )是可满足的当且仅当*ϕ*是可满足的；

*(ii)* 若*ϕ*1 ≡ *ϕ*2，则F*µ* (*ϕ*1*,V* ) ≡ F*µ* (*ϕ*2*,V* )； *(iii)* 若*ϕ*1 |= *ϕ*2，则F*µ* (*ϕ*1*,V* ) |= F*µ* (*ϕ*2*,V* )； *(iv)* F*µ* (*ψ*1 ∨ *ψ*2*,V* ) ≡ F*µ* (*ψ*1*,V* ) ∨ F*µ* (*ψ*2*,V* )；

*(v)* F*µ* (*ψ*1 ∧ *ψ*2*,V* ) |= F*µ* (*ψ*1*,V* ) ∧ F*µ* (*ψ*2*,V* )；

*(vi)* F*µ* (*ϕ*1 ∧ *ϕ*2*,V* ) ≡ F*µ* (*ϕ*1*,V* ) ∧ *ϕ*2 若*IR*(*ϕ*2*,V* )。

证明**.** (i) (*M , v*) ∈ *Mod*(F*µ* (*ϕ,V* ))

⇔ ∃(*M* ′*, v*′) ∈ *Mod*(*ϕ*) 使得(*M , v*) ↔*V* (*M* ′*, v*′)。

因为*ϕ* |= F*µ* (*ϕ,V* )，所以(*M , v*) ∈ *Mod*(*ϕ*) 蕴涵(*M , v*) |= F*µ* (*ϕ,V* )。

(ii) 和(iii) 可以类似地证明。

(iv) (*M , v*) ∈ *Mod*(F*µ* (*ϕ*1 ∨ *ϕ*2*,V* ))

⇔ 存在(*M* ′*, v*′) ∈ *Mod*(*ϕ*1 ∨ *ϕ*2)使得(*M , v*) ↔*V* (*M* ′*, v*′) 和(*M* ′*, v*′) |= *ϕ*1 或(*M* ′*, v*′) |= *ϕ*2

⇔ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 使得(*M* ′*, v*′) |= *ϕ*1 或(*M* ′*, v*′) |= *ϕ*2，则存在(*M*1*, v*1) ∈ *Mod*(F*µ* (*ϕ*1*,V* ))

使得(*M* ′*, v*′) ↔*V* (*M*1*, v*1)，或∃(*M*2*, v*2) ∈ *Mod*(F*µ* (*ϕ*2*,V* )) 使得(*M* ′*, v*′) ↔*V* (*M*2*, v*2)

⇔ (*M , v*) |= F*µ* (*ϕ*1*,V* )或(*M , v*) |= F*µ* (*ϕ*2*,V* )

⇔ (*M , v*) |= F*µ* (*ϕ*1*,V* ) ∨ F*µ* (*ϕ*2*,V* )。

(v)可以像(iv)一样证明。

(vi) (*M , v*) |= F*µ* (*ϕ*1 ∧ *ϕ*2*,V* )

⇔ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 使得(*M* ′*, v*′) |= *ϕ*1 ∧ *ϕ*2

⇔ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 使得(*M* ′*, v*′) |= *ϕ*1 和(*M* ′*, v*′) |= *ϕ*2

⇔ (*M , v*) |= F*µ* (*ϕ*1*,V* ) 且(*M , v*) |= *ϕ*2 (IR(*ϕ*2*,V* ))

⇔ (*M , v*) |= F*µ* (*ϕ*1*,V* ) ∧ *ϕ*2。

命题[3.8](#_bookmark74)(i) 表明从一个*µ*-句子中遗忘一些原子命题不影响该句子的可满足性； 从(ii)可以看出，如果两个句子是等价的，则他们遗忘相同原子命题得到的结果是等价 的；(iv)指出析取公式*ϕ*1 ∨ *ϕ*2的遗忘可以由分开计算遗忘后在析取而得到；而正如(v) 中指出的那样，合取公式*ψ*1 ∧ *ψ*2的遗忘不能分别计算再合取，这甚至对于命题公式 都是不成立。 例：令*ϕ* = *p* ∧ (*q* ∨ ¬*p*)，从*ϕ*中遗忘*p*的结果为*q*，但是*Forget*(*p,* {*p*}) ∧ *Forget*(*q* ∨ ¬*p,* {*p*}) ≡ ⊤。 显然二者不等价。

下面是关于*µ*-演算下遗忘算子的其它性质。

命题 **3.9** (分解性)**.** 给定*µ-*句子*ϕ*、原子命题集*V* 和原子命题*p*且*p* ∈*/ V* ，则：

F*µ* (*ϕ,* {*p*} ∪ *V* ) ≡ F*µ* (F*µ* (*ϕ,* {*p*})*,V* )*.*

证明**.** 令(*M*1*, v*1)（*M*1 = (*S*1*, R*1*, L*1*, s*1)）为F*µ* (*ϕ,* {*p*} ∪ *V* )的模型。 由遗忘的定义可知， 存在*ϕ*的一个模型(*M , v*)（*M* = (*S, R, L, s*)）使得(*M*1*, v*1) ↔{*p*}∪*V* (*M , v*)。如下构造Kripke 结构*M*2 = (*S*2*, R*2*, L*2*, s*2)：

(1) 对于*s*2，令*s*2为满足下列条件的状态：

**–** *p* ∈ *L*2(*s*2)当且仅当*p* ∈ *L*1(*s*1)，

**–** 对任意*q* ∈ *V* ，*q* ∈ *L*2(*s*2)当且仅当*q* ∈ *L*(*s*)，

**–** 对于其它的原子命题*q*′，*q*′ ∈ *L*2(*s*2)当且仅当*q*′ ∈ *L*1(*s*1)当且仅当*q*′ ∈ *L*(*s*)。

(2) 其它情形：假定*M*1和*M* 有{*p*} ∪ *V* -互模拟关系*B*。 (i) 对任意*w* ∈ *S*和*w*1 ∈ *S*1且(*w, w*1) ∈ *B*，令*w*2 ∈ *S*2和

\* *p* ∈ *L*2(*w*2)当且仅当*p* ∈ *L*1(*w*1)，

\* 对任意*q* ∈ *V* ，*q* ∈ *L*2(*w*2)当且仅当*q* ∈ *L*(*w*)，

\* 对其它原子命题*q*′，*q*′ ∈ *L*2(*w*2)当且仅当*q*′ ∈ *L*1(*w*1)当且仅当*q*′ ∈ *L*(*w*)。

(ii) 若(*w*′ *, w*1) ∈ *R*1，且*w*2是由*w*1构造，*w*′ ∈ *S*2由*w*′ 构造，则(*w*′ *, w*2) ∈ *R*2。

1

(3) 删除*S*2和*R*2中重复的元素。

2 1 2

显然，*M* ↔{*p*} *M*2和*M*2 ↔*V M*1。此外，对所有的*X* ∈ *V* ，令*v*2(*X* ) = *B*1(*v*(*X* ))。因此，

(*M , v*) ↔{*p*} (*M*2*, v*2) 且(*M*2*, v*2) ↔*V* (*M*1*, v*1)，从而(*M*2*, s*2) |= F*µ* (*ϕ, p*)。所以，(*M*1*, v*1) |=

F*µ* (F*µ* (*ϕ, p*)*,V* )。

另一方面，假设(*M*1*, v*1)是F*µ* (F*µ* (*ϕ, p*)*,V* )的一个模型

⇒ 存在(*M*2*, v*2)使得(*M*2*, v*2) |= F*µ* (*ϕ, p*) 且(*M*2*, v*2) ↔*V* (*M*1*, v*1) (定义[3.6](#_bookmark68))

⇒ 存在(*M , v*) 使得(*M , v*) |= *ϕ* 且(*M , v*) ↔{*p*} (*M*2*, v*2) (定义[3.6](#_bookmark68))

因此，由命题[3.6](#_bookmark65) (iii)可知(*M , v*) ↔{*p*}∪*V* (*M*1*, v*1)，所以(*M*1*, v*1) |= F*µ* (*ϕ,* {*p*} ∪ *V* )。

下面这一性质为上述命题的推论。

推论 **3.5** (切片性)**.** 给定*µ-*句子*ϕ*和原子命题集*Vi* ⊆ *A* (*i* = 1*,* 2)，有：

F*µ* (*ϕ,V*1 ∪ *V*2) ≡ F*µ* (F*µ* (*ϕ,V*1)*,V*2)*.*

F*µ* 的另一个性质是关于AX和EX时序算子的：形如AX*ϕ*或EX*ϕ*的*µ*-句子的遗忘可 以提到AX和EX后面计算。 而对于*µX.ϕ*和*νX.ϕ*就没有这样的性质，因为*ϕ*显然不是一 个*µ*-句子。

引理 **3.5.** 令(*M , s, v*) ↔⟨{*X* }*,V* ⟩ (*M* ′*, s*′*, v*′)且*ϕ*是一个*µ-*公式。若*IR*(*ϕ,V* )且*X* 正出现在*ϕ*中，

则(*M , s, v*) |= *ϕ* 当且仅当(*M* ′*, s*′*, v*′) |= *ϕ*。

证明**.** 假设(*M , s, v*)和(*M* ′*, s*′*, v*′) 之间的⟨{*X* }*,V* ⟩-互模拟关系为*BX*。

基始**.** (a) *ϕ* = *p*，其中*p* ∈ *A* 。

(*M , s, v*) |= *ϕ*

⇔ *p* ∈ *L*(*s*)

⇔ *p* ∈ *L*′(*s*) ((*M , s, v*) ↔⟨{*X* }*,V* ⟩ (*M* ′*, s*′*, v*′))

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

(b) *ϕ* = *Y* ，其中*Y* ∈ *V* − {*X* }。

(*M , s, v*) |= *ϕ*

⇔ *s* ∈ *v*(*Y* )

⇔ *s*′ ∈ *v*′(*Y* ) ((*M , s, v*) ↔⟨{*X* }*,V* ⟩ (*M* ′*, s*′*, v*′))

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。 归纳步**.** (a) *ϕ* = ¬*ϕ*1。 (*M , s, v*) |= *ϕ*

⇔ (*M , s, v*) |= *ϕ*1

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*1

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

(b) *ϕ* = *ϕ*1 ∨ *ϕ*2容易证明。

1. *ϕ* = AX*ϕ*1。

(*M , s, v*) |= *ϕ*

⇔ 对任意(*s, s*1) ∈ *R*，(*M , s*1*, v*) |= *ϕ*1

⇔ 对任意(*s*′*, s*′ ) ∈ *R*′，存在(*s, s*1) ∈ *R*使得(*M , s*1*, v*) ↔⟨{*X* }*,V* ⟩ (*M* ′*, s*′ *, v*′)

1 1

⇔ 对任意(*s*′*, s*′ ) ∈ *R*′，(*M* ′*, s*′ *, v*′) |= *ϕ*1 (归纳假设)

1 1

⇔ (*M* ′*, s*′*, v*′) |= *ϕ*。

1. *ϕ* = *νY.ϕ*1。

(d.1) *Y* = *X*。(*M , s, v*) |= *νX.ϕ*1

⇔ *s* ∈ ‖*ϕ*‖*M*

*v*

⇔ *s* ∈ J{*S*1 ⊆ *S* | *S*1 ⊆ ‖*ϕ*1‖*M* }

*v*[*X* :=*S*1]

*M*

⇔ 存在*S*1 ⊆ *S*，使得*S*1 ⊆ ‖*ϕ*1‖*v*[*X* :=*S*1]和*s* ∈ *S*1

⇔ 存在*S*2 = *S*1 ∪ {*s*2 | (*si,t*) ∈ *BX , s*2 ∈ *S,* and *s*1 ∈ *S*1 with *i* = 1*,* 2}，使得*S*2 ⊆ ‖*ϕ*1‖*M*

*v*[*X* :=*S*2]

且*s* ∈ *S*2 (看命题 [3.7](#_bookmark67)的证明)

⇔ 存在*BX* (*S*2) ⊆ *S*′，使得*BX* (*S*2) ⊆ ‖*ϕ*1‖*M*

′

、 *s*′

*B* (*S* )且(*M , v*[*X* := *S* ])

*v*′[*X* :=*BX* (*S*2)]

∈ *X* 2

2 ↔*V*

(*M* ′*, v*′[*X* := *BX* (*S*2)]) ((*s, s*′) ∈ *BX* , 不变性)

*M* ′

⇔ *s*′ ∈ J{*S*′ ⊆ *S*′ | *S*′ ⊆ ‖*ϕ*1‖ }

1 1 *v*′[*X* :=*S*′ ]

1

⇔ (*M* ′*, s*′*, v*′) |= *νX.ϕ*1。

(d.2) *Y* ̸= *X*。(*M , s, v*) |= *νY.ϕ*1

⇔ *s* ∈ ‖*ϕ*‖*M*

*v*

⇔ *s* ∈ J{*S*1 ⊆ *S* | *S*1 ⊆ ‖*ϕ*1‖*M* }

*v*[*Y* :=*S*1]

*M*

⇔ 存在*S*1 ⊆ *S*，使得*S*1 ⊆ ‖*ϕ*1‖*v*[*Y* :=*S*1]且*s* ∈ *S*1

*M*

⇔ 存在*S*2 = *S*1 ∪ {*s*2 | (*si,t*) ∈ *BX , s*2 ∈ *S,* and *s*1 ∈ *S*1 with *i* = 1*,* 2}，使得*S*2 ⊆ ‖*ϕ*1‖*v*[*Y* :=*S*2]

且*s* ∈ *S*2 (看命题 [3.7](#_bookmark67)的证明)

⇔ 存在*BX* (*S*2) ⊆ *S*′，使得*BX* (*S*2) ⊆ ‖*ϕ*1‖*M*

′

、*s*′

*B* (*S* )且(*M , v*[*Y* := *S* ])

*v*′[*Y* :=*BX* (*S*2)]

∈ *X* 2

2 ↔⟨{*X* }*,V* ⟩

(*M* ′*, v*′[*Y* := *BX* (*S*2)]) ((*s, s*′) ∈ *BX* , 归纳假设)

*M* ′

⇔ *s*′ ∈ J{*S*′ ⊆ *S*′ | *S*′ ⊆ ‖*ϕ*1‖ }

1 1 *v*′[*Y* :=*S*′ ]

1

⇔ (*M* ′*, s*′*, v*′) |= *νX.ϕ*1。

命题 **3.10** (同质性)**.** 给定原子命题集合*V* ⊆ *A* 和*µ-*公式*ϕ*，则*:*

1. F*µ* (AX*ϕ,V* ) ≡ AXF*µ* (*ϕ,V* )；
2. F*µ* (EX*ϕ,V* ) ≡ EXF*µ* (*ϕ,V* )；
3. F*µ* (*νX.ϕ,V* ) ≡ *νX.*F*µ* (*ϕ,V* )；

*(iv)* F*µ* (*µX.ϕ,V* ) ≡ *µX.*F*µ* (*ϕ,V* )。

证明**.** 令(*M , v*)（*M* = (*S, R, L, r*)）、(*Mi, vi*)（*Mi* = (*Si, Ri, Li, ri*)）和(*M* ′*, v*′)（*M* ′ = (*S*′*, R*′*,*

*L*′*, r*′)）为赋值，其中*i* ∈ N。

(i) (⇒) (*M , v*) |= F*µ* (AX*ϕ,V* )

⇒ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 且(*M* ′*, v*′) |= AX*ϕ*

⇒ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 且对任意(*r*′*, r*′′) ∈ *R*′，(*M* ′*, r*′′*, v*′) |= *ϕ*

⇒ 对任意(*r, r*1) ∈ *R*，存在(*r*′*, r*′ ) ∈ *R*′ 使得(*M , r*1*, v*) ↔*V* (*M* ′*, r*′ *, v*′) 且对任意(*r*′*, r*′′) ∈ *R*′，

1 1

(*M* ′*, r*′′*, v*′) |= *ϕ*

⇒ 对任意(*r, r*1) ∈ *R*，存在(*M , r*1*, v*) ↔*V* (*M* ′*, r*′ *, v*′) 且(*M* ′*, r*′ *, v*′) |= *ϕ*

1 1

⇒ 对任意(*r, r*1) ∈ *R*，(*M , r*1*, v*) |= F*µ* (*ϕ,V* ) (IR(F*µ* (*ϕ,V* )*,V* ))

⇒ (*M , v*) |= AX(F*µ* (*ϕ,V* ))。

(⇐) (*M , v*) |= AXF*µ* (*ϕ,V* )

⇒ 对任意(*r, r*1) ∈ *R*，(*M , r*1*, v*) |= F*µ* (*ϕ,V* )

⇒ 对任意(*r, r*1) ∈ *R*，(*Mr*1 *, r*1*, v*) |= F*µ* (*ϕ,V* )

⇒ 对任意(*r, r*1) ∈ *R*，存在(*Mr*1 *, r*1*, v*) ↔*V* (*M* ′*, r*′*, v*′) 且(*M* ′*, r*′*, v*′) |= *ϕ*

⇒ 对任意*i* ≥ 0，存在(*Mri , ri, v*) ↔*V* (*M* ′*, r*′*, v*′) 且(*M* ′*, r*′*, v*′) |= *ϕ*，其中{*r*0*, r*1*, . . .* } = {*s* |

*i i i*

*i i i*

(*r, s*) ∈ *R*}，*M* ′ = (*S*′*, R*′*, L*′*, r*′)（假定当*i* ̸= *j*时，*S*′ ∩*S*′ = 0/ ），*Bi*是(*Mr , ri, v*) 和(*M* ′*, r*′*, v*′)之

*i i i i i*

*i j* *i*

*i i i*

间的Var-*V* -互模拟关系

⇒ (*M* \**, v*\*) ↔*V* (*M , v*) 且(*M* \**, v*\*) |= AX*ϕ*，其中*M* \* = (*S*\**, R*\**, L*\**, r*)且

∙ *S*\* = {*r*} ∪ J

*i*≥0

*S*′,

∙ *R*\* = {(*r, r*′) | *i* ≥ 0} ∪ J

*i*

*R*′,

*i i*≥0 *i*

∙ 对任意*s* ∈ *S*\*，*L*\*(*s*) = J

*i*≥0

*i*

*L*′(*s*)，且*L*\*(*r*) = *L*(*r*)，

∙ 对任意*X* ∈ *V* ，

*v*\*(*X* ) =

*i*≥0 *vi*(*X* )*,* 若*r* ̸∈ *v*(*X* )；

( J

*i*≥0 *vi*(*X* ) ∪ {*r*}*,* 否则。

J

⇒ (*M , v*) |= F*µ* (AX*ϕ,V* )。

(ii)可类似于(i)证明。

(iii) (⇒) (*M , v*) |= F*µ* (*νX.ϕ,V* )

⇒ 存在(*M* ′*, v*′) ↔*V* (*M , v*) 且(*M* ′*, v*′) |= *νX.ϕ*

⇒ *r*′ ∈ J{*S*1 ⊆ *S*′ | *S*1 ⊆ ‖*ϕ*‖*M*

′

*v*′[*X* :=*S*1]}

*ϕ*

⇒ 存在*S*1 ⊆ *S*′ 使得*S*1 ⊆ ‖*ϕ*‖*M*

′

、*r*′

*S* 和*r*′ *M* ′

*v*′[*X* :=*S*1]

*M* ′

∈ 1 ∈ ‖

*M* ′

‖*v*′[*X* :=*S*1]

⇒ *r*′ ∈ ‖F*µ* (*ϕ,V* )‖*v*′[*X* :=*S* ] 且*S*1 ⊆ ‖F*µ* (*ϕ,V* )‖*v*′[*X* :=*S* ] (*ϕ* |= F*µ* (*ϕ,V* ))

1 1

*M* ′

⇒ *r*′ ∈ J{*S*1 ⊆ *S*′ | *S*1 ⊆ ‖F*µ* (*ϕ,V* )‖*v*′[*X* :=*S* ]}

1

⇒ (*M* ′*, v*′) |= *νX.*F*µ* (*ϕ,V* )

⇒ (*M , v*) |= *νX.*F*µ* (*ϕ,V* ) (IR(*νX.*F*µ* (*ϕ,V* )*,V* )) (⇐) (*M , v*) |= *νX.*F*µ* (*ϕ,V* )

⇒ *r* ∈ J{*S*1 ⊆ *S* | *S*1 ⊆ ‖F*µ* (*ϕ,V* )‖*M* }

*v*[*X* :=*S*1]

*M*

⇒ 存在*S*1 ⊆ *S*，使得*S*1 ⊆ ‖F*µ* (*ϕ,V* )‖*v*[*X* :=*S*1] 且*r* ∈ *S*1

⇒ 存在*S*1 ⊆ *S*2 ⊆ *S*且(*M* ′*, v*′) ↔*V* (*M , v*[*X* := *S*2])（有Var-*V* -互模拟关系*B*），使得(*M* ′*, v*′) |=

*ϕ*且*S*2 ⊆ ‖F*µ* (*ϕ,V* )‖*M*

*v*[*X* :=*S*2]

⇒ 存在(*M* ′*, v*′) ↔*V* (*M , v*[*X* := *S*2])，使得*B*(*S*2) ⊆ ‖*ϕ*‖*M*

⇒ 存在*B*(*S*2) ⊆ *S*′，使得*B*(*S*2) ⊆ ‖*ϕ*‖*M*

′

且*r*′

′

*v*′[*X* :=*B*(*S*2)]

*B*(*S* ) ((*r, r*′)

*B*, *r S* )

⇒ *r*′ ∈ J{*S*′ ⊆ *S*′ | *S*′ ⊆ ‖*ϕ*‖*M* }

′

*v*′[*X* :=*B*(*S*2)] ∈ 2

∈ ∈ 2

1 1

*v*′

′

⇒ (*M* ′*, v*′) |= *νX.ϕ*

[*X* :=*S*1]

⇒ (*M* ′*, v*′) |= F*µ* (*νX.ϕ,V* )且(*M* ′*, v*′) ↔⟨{*X* }*,V* ⟩ (*M , v*)

⇒ (*M , v*) |= F*µ* (*νX.ϕ,V* ) (引理 [3.5](#_bookmark75))

(iv) 可类似地证明。

AX（或EX）在F*µ* 上的同质性表明，在从AX*ϕ* （或EX*ϕ*）遗忘*V* 中的原子命题等价

于将F*µ* 提到AX和EX后面计算的结果。 (iii)和(iv)，遗忘原子命题与不动点操作是无关 的。 特别地，当命题[3.10](#_bookmark76)中的公式*φ* 为命题公式时，从AX*φ* (EX*φ* ) 或*µX.ϕ* 中遗忘原子 命题可以使用命题逻辑的遗忘计算方法来计算。

前文证明了*µ*-句子遗忘任意原子命题集合的结果总是*µ*-句子。 这里讨论一种*µ*-公 式的子类：不含有不定点操作的*µ*-公式集，记为X-类。 通过等值式：AX*ϕ*1 ∧ AX*ϕ*2 ≡ AX(*ϕ*1 ∧ *ϕ*2) 和EX*ϕ*1 ∨ EX*ϕ*2 ≡ EX(*ϕ*1 ∨ *ϕ*2)，可以将X-类中的任意公式转换为具有下面形 式的公式的析取：

*ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn*， (3.1)

其中*ϕ*0是X-类中不含有模态词的公式，*ϕi* (1 ≤ *i* ≤ *n*)为X-类中的公式，且任意*ϕi* (0 ≤ *i* ≤ *n*) 都有可能缺失。

引理 **3.6.** 令*V* ⊆ *A* 为原子命题集，*ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn*为具有形式[*(3.1*](#_bookmark77)*)*的可满

足公式，则

F*µ* (*ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn,V* )

≡ F*µ* (*ϕ*0*,V* ) ∧ F*µ* (AX*ϕ*1*,V* ) ∧ 八

2≤*i*≤*n*

≡ F*µ* (*ϕ*0*,V* ) ∧ AXF*µ* (*ϕ*1*,V* ) ∧ 八

2≤*i*≤*n*

F*µ* (EX(*ϕi* ∧ *ϕ*1)*,V* )

EXF*µ* (*ϕi* ∧ *ϕ*1*,V* )*.*

证明**.** 由命题 [3.10](#_bookmark76)的(i)和(ii)可知，

F*µ* (*ϕ*0*,V* ) ∧ F*µ* (AX*ϕ*1*,V* ) ∧ 八

2≤*i*≤*n*

≡ F*µ* (*ϕ*0*,V* ) ∧ AXF*µ* (*ϕ*1*,V* ) ∧ 八

F*µ* (EX(*ϕi* ∧ *ϕ*1)*,V* )

EXF*µ* (*ϕi* ∧ *ϕ*1*,V* )*.*

2≤*i*≤*n*

(⇒) (*M , v*) |= F*µ* (*ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn,V* )

⇒ 存在(*M , v*) ↔*V* (*M* ′*, v*′) 使得(*M* ′*, v*′) |= *ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn*

⇒ (*M* ′*, v*′) |= *ϕ*0、(*M* ′*, v*′) |= AX*ϕ*1和(*M* ′*, v*′) |= EX*ϕi*，其中2 ≤ *i* ≤ *n*

⇒ (*M* ′*, v*′) |= *ϕ*0、(*M* ′*, v*′) |= AX*ϕ*1和(*M* ′*, v*′) |= EX(*ϕi* ∧ *ϕ*1)，其中2 ≤ *i* ≤ *n* ((*M , v*) |=

AX*ϕ*1)

⇒ (*M , v*) |= F*µ* (*ϕ*0*,V* )、 (*M , v*) |= F*µ* (AX*ϕ*1*,V* )和(*M , v*) |= F*µ* (EX(*ϕi* ∧ *ϕ*1)*,V* )，其中2 ≤

*i* ≤ *n* ((*M , v*) ↔*V* (*M* ′*, v*′), IR(F*µ* (*ϕ*0*,V* )*,V* ), IR(F*µ* (AX*ϕ*1*,V* )*,V* ), IR(F*µ* (EX(*ϕi* ∧*ϕ*1)*,V* )*,V* ))

⇒ (*M , v*) |= F*µ* (*ϕ*0*,V* ) ∧ F*µ* (AX*ϕ*1*,V* ) ∧ 八

2≤*i*≤*n*

F*µ* (EX(*ϕi* ∧ *ϕ*1)*,V* )

(⇐) (*M , v*) |= F*µ* (*ϕ*0*,V* )∧F*µ* (AX*ϕ*1*,V* )∧八

2≤*i*≤*n*

F*µ* (EX(*ϕi* ∧*ϕ*1)*,V* )，其中*M* = (*S, r, R, L*)

⇒ (*M , v*) |= F*µ* (*ϕ*0*,V* ) ∧ AXF*µ* (*ϕ*1*,V* ) ∧ 八

2≤*i*≤*n*

EXF*µ* (*ϕi* ∧ *ϕ*1*,V* )

⇒ (*M , r, v*) |= F*µ* (*ϕ*0*,V* )，对任意(*r, r*′) ∈ *R*，(*M , r*′*, v*) |= F*µ* (*ϕ*1*,V* )，且对任意2 ≤ *i* ≤ *n*，

存在(*r, r*′′) ∈ *R* 使得(*M , r*′′*, v*) |= F*µ* (*ϕi* ∧ *ϕ*1*,V* )

⇒ 存 在(*M* ′*, v*) |= *ϕ*0 且(*M* ′*, v*) ↔*V* (*M , v*)， 其 中*M* ′ = (*S, r, R, L*′)； 对 任 意(*r, r j* ) ∈ *R*，

存在(*M* ′*, r*′ *, v*′ ) 使得(*M* ′*, r*′ *, v*′ ) ↔*V* (*Mr , r j, v*) 且(*M* ′*, r*′ *, v*′ ) |= *ϕ*1；且对任意(*r, r j* ) ∈ *R*

*j j j*

*j j j j*

*j j j*

且(*Mrj , r j, v*) |= F*µ* (*ϕi* ∧*ϕ*1*,V* ) (2 ≤ *i* ≤ *n*)，存在(*M* ′*, r*′ *, v*′ ) ↔*V* (*Mr , r j, v*) 使得(*M* ′*, r*′ *, v*′ ) |=

*j j j j*

*j j j*

*ϕi* ∧ *ϕ*1 ( *j* ≥ 0，当*x* ̸= *y*时，假设*Sx* ∩ *Sy* = 0/ )，其中{*r*0*, r*1*, . . .* } = {*r*′ | (*r, r*′) ∈ *R*}，*M* ′ =

*j*

(*S*′ *, R*′ *, L*′ *, r*′ )

*j j j j*

⇒ 存在(*M* \**, v*\*) ↔*V* (*M , v*)，使得(*M* \**, v*\*) |= *ϕ*0 ∧ AX*ϕ*1 ∧ 八

2≤*i*≤*n*

EX(*ϕi* ∧ *ϕ*1)，其中*M* \* =

(*S*\**, R*\**, L*\**, r*)且

∙ *S*\* = {*r*} ∪ J *j*≥0 *S*′ ，

*j*

∙ *R*\* = {(*r, r*′ ) | *j* ≥ 0} ∪ J

*j*

*j*≥0

*R*′ ，

*j*

∙ *L*\*(*r*) = *L*′(*r*)，且对任意*s* ∈ *S*′ ，*L*\*(*s*) = *L*′ (*s*)，

∙ 对任意*X* ∈ *V* ，

*j j*

( J ′

*v*\*(*X* ) =

*j*≥0 *v j* (*X* )*,* 若*r* ̸∈ *v*(*X* )；

J *j*≥0 *v j* (*X* ) ∪ {*r*}*,* 否则。

′

⇒ (*M , v*) |= F*µ* (*ϕ*0 ∧ AX*ϕ*1 ∧ EX*ϕ*2 ∧ · · · ∧ EX*ϕn,V* )。

给定X-类中的公式*ϕ*，公式*ϕ*的度（记为*degree*(*ϕ*)）定义如下：

*degree*(*X* ) = *degree*(*p*) = *degree*(¬*p*) = 0*, degree*(AX*ψ*) = *degree*(*ψ*) + 1*, degree*(EX*ψ*) = *degree*(*ψ*) + 1*,*

*degree*(*ψ*1 \* *ψ*2) = max{*degree*(*ψ*1)*, degree*(*ψ*2)}*,*

其中*X* ∈ *V* 、 *p* ∈ *A* 和\* ∈ {∨*,* ∧}。

命题 **3.11.** 令*V* ⊆ *A* 为原子命题集、 *ϕ*为X*-*类中的公式。 存在X*-*类中的公式*ψ*使得*ψ* ≡

F*µ* (*ϕ,V* )。

证明**.** 假定*degree*(*ϕ*) = *n*；通过归纳公式的度来证明这一结论。

基始**.** *n* = 0. 令*ϕ*是一个可能含有自由变元的度为0的*µ*-公式。 由定理 [3.6](#_bookmark73)和命题 [3.8](#_bookmark74) (vi)可知F*µ* (*ϕ,V* ) = *Forget*(*ϕ,V* )。

归纳步**.** 假设对所有*degree*(*ϕ*1) ≤ *k*的X-类中公式*ϕ*1，F*µ* (*ϕ,V* )在X-类中。 证明对任

意公式*ϕ*2且*degree*(*ϕ*2) = *k* + 1，F*µ* (*ϕ*2*,V* ) 在X-类中。

*ϕ*2可以被转换为形如公式[(3.1](#_bookmark77))的公式的析取*ϕ*′，且*degree*(*ϕ*′) ≤ *k* + 1（因为在转换

过程中不会引入新的嵌套模态词）。

不失一般性地，假设

*ϕ*′ = ／

1≤*i*≤*x*

*ϕi,*0 ∧ AX*ϕi,*1 ∧ EX*ϕi,*2 ∧ · · · ∧ EX*ϕi,ni ,*

且*ϕi, j* (1 ≤ *i* ≤ *x,* 1 ≤ *j* ≤ *ni*)是*degree*(*ϕi, j* ) ≤ *k*的公式。

由引理 [3.6](#_bookmark78)和命题 [3.8](#_bookmark74)(iv)可知：

F*µ* (*ϕ*′*,V* ) ≡ ／

1≤*i*≤*x*

／

F*µ* (*ϕi,*0 ∧ AX*ϕi,*1 ∧ EX*ϕi,*2 ∧ · · · ∧ EX*ϕi,ni ,V* )

八

≡

1≤*i*≤*x*

(F*µ* (*ϕi,*0*,V* ) ∧ AXF*µ* (*ϕi,*1*,V* ) ∧

2≤ *j*≤*ni*

EXF*µ* (*ϕi,*1 ∧ *ϕi, j,V* ))。

因此，有归纳假设可知F*µ* (*ϕ*′*,V* )在X-类中。

例 **3.6.** 令*ϕ*1 = *X* ∧ *p*、 *ϕ*2 = AX(*c* ∧ EX*d*) ∧ AX*e*、 *ϕ*3 = EX¬*d* ∧ (EX¬*p* ∨ EX *p*)、 *ϕ* = *ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3且*V* = {*e, d*}，其中*X* ∈ *V* 且*p, c, d, e*为原子命题。

如下计算公式*ϕ*的度：

*degree*(*ϕ*) = max{*degree*(*ϕ*1)*, degree*(*ϕ*2 ∧ *ϕ*3)}

= max{0*,* max{*degree*(*ϕ*2)*, degree*(*ϕ*3)}

= 2*,*

*degree*(*ϕ*1) = 0*,*

*degree*(*ϕ*2) = max{*degree*(AX(*c* ∧ EX*d*))*, degree*(AX*e*)}

= max{max{0*,* 1} + 1*,* 1}

= 2*,*

*degree*(*ϕ*3) = max{*degree*(EX¬*d*)*, degree*(EX¬*p* ∨ EX *p*)}

= max{1*,* max{1*,* 1}}

= 1*.*

此外，公式*ϕ*可如下转换为具有形式[(3.1](#_bookmark77))的公式的析取：

*ϕ* = *ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3

≡ *X* ∧ *p* ∧ AX(*c* ∧ *e* ∧ EX*d*) ∧ EX¬*d* ∧ (EX¬*p* ∨ EX *p*)

≡ (*X* ∧ *p* ∧ AX(*c* ∧ *e* ∧ EX*d*) ∧ EX¬*d* ∧ EX¬*p*)∨

(*X* ∧ *p* ∧ AX(*c* ∧ *e* ∧ EX*d*) ∧ EX¬*d* ∧ EX *p*)*.*

则从*ϕ*中遗忘*V* 的结果为：

F*µ* (*ϕ,V* ) ≡ F*µ* (*X* ∧ *p* ∧ AX(*c* ∧ *e* ∧ EX*d*) ∧ EX¬*d* ∧ EX¬*p,V* )∨

F*µ* (*X* ∧ *p* ∧ AX(*c* ∧ *e* ∧ EX*d*) ∧ EX¬*d* ∧ EX *p,V* )

≡ (*X* ∧ *p* ∧ AXF*µ* (*c* ∧ *e* ∧ EX*d,V* )∧

EXF*µ* (¬*d* ∧ *c* ∧ *e* ∧ EX*d,V* ) ∧ EXF*µ* (¬*p* ∧ *c* ∧ *e* ∧ EX*d,V* ))∨ (*X* ∧ *p* ∧ AXF*µ* (*c* ∧ *e* ∧ EX*d,V* )∧

EXF*µ* (¬*d* ∧ *c* ∧ *e* ∧ EX*d,V* ) ∧ EXF*µ* (*p* ∧ *c* ∧ *e* ∧ EX*d,V* ))

≡ (*X* ∧ *p* ∧ AX*c* ∧ EX*c* ∧ EX(¬*p* ∧ *c*)) ∨ (*X* ∧ *p* ∧ AX*c* ∧ EX*c* ∧ EX(*p* ∧ *c*))

≡ *X* ∧ *p* ∧ AX*c* ∧ EX*c* ∧ (EX(¬*p* ∧ *c*) ∨ EX(*p* ∧ *c*))*.*

**3.2.3** 计算复杂性

析取*µ*-公式*ϕ*的均匀插值为～∃*pϕ*（*p* ∈ *A* ），且与*ϕ*[*p/*⊤*,* ¬*p/*⊤]等价[[82](#_bookmark198)]，其中*ϕ*[*p/*⊤*,* ¬*p/*⊤]表

示将*ϕ*中的*p*和¬*p*同时用⊤替换。 正如之前说过的F*µ* (*ϕ,V* )与均匀插值～∃*V ϕ*等价[[82](#_bookmark198)]。 因

此，下面的命题容易证明。

推论 **3.6.** 给定*µ-*句子*ϕ*和原子命题*p* ∈ *A* 。 若*ϕ*是一个析取*µ-*句子，F*µ* (*ϕ,* {*p*})能在线

性时间内计算。

证明**.** 当*ϕ*为吸取公式时，F*µ* (*ϕ,* {*p*})可通过将*ϕ*中的文字*p*和¬*p*同时替换为⊤得到。 这

只需扫描一遍公式*ϕ*，因此只需要*ϕ*大小的线性时间。

这种情况下，可以首先将一个*µ*-句子转化为析取*µ*-公式，再去计算遗忘。 下面的

例子给出如何计算从析取*µ*-公式中遗忘“*ch*”。

例 **3.7.** 令*ϕ*1 = *j* ∧ *ch* ∧*Cover*(¬ *j* ∧¬*ch,* ⊤)、*ϕ*2 = *µX.*( *j* ∧ *ch*) ∧*Cover*(*X,* ⊤)、*ϕ*3 = *νX.*( *j* ∧ *ch*) ∧ *Cover*(*Cover*(*X,* ⊤)*,* ⊤)。 令*V* = {*ch*}，我们能容易地计算从这些公式里遗忘*V*。

(1) F*µ* (*ϕ*1*,V* ) ≡ *j* ∧ *Cover*(¬ *j,* ⊤) ≡ *j* ∧ EX(¬ *j*)；

(2) F*µ* (*ϕ*2*,V* ) ≡ *µX. j* ∧ *Cover*(*X,* ⊤) ≡ *µX. j* ∧ EX*X* ；

1. F*µ* (*ϕ*2*,V* ) ≡ *νX. j* ∧ *Cover*(*Cover*(*X,* ⊤)*,* ⊤) ≡ *νX. j* ∧ EX(EX*X* )。

尽管如此，关于遗忘的模型检测（即：检查一个结构是否为从*µ*-句子中遗忘某个

原子命题集的模型）也是不容易的。

命题 **3.12** (模型检测)**.** 给定一个有限的*Kripke* 结构*M*、 一个*µ-*句子*ϕ*和原子命题集*V* ⊆

*A* 。 有：

*(i)* 判定*M* |=? F*µ* (*ϕ,V* )在EXPTIME中；

*(ii)* 若*ϕ*是一个析取*µ-*公式，则判定*M* |=? F*µ* (*ϕ,V* )在NP∩*co-*NP中。

证明**.** 对于一个*µ*-公式*ϕ*，如果该公式为一个析取*µ*-公式可在多项式时间内构造一 个*µ*-自动机（也叫模态自动机[[100](#_bookmark214)]）*Aϕ* ，否则需要指数时间构造其对应的*µ*-自动机[3](#_bookmark0). 这 里证明(ii)，(i)可以类似地证明。

令*Aϕ* 为一个*µ*-自动机且满足对任意Kripke *N* ，*Aϕ* 接受*N* 当且仅当*N* |= *ϕ*，其 中*Aϕ* = (*Q,* Σ*p,* Σ*r, q*0*, δ ,* Ω)、 Σ*p* = *Var*(*ϕ*)。 不是一般性地，假定*V* ⊆ *Var*(*ϕ*)和*V* = {*p*}。 因此，构造一个*µ*-自动机*B* = (*Q,* Σ*p* − *V,* Σ*r, q*0*, δ* ′*,* Ω)使得对任意*q* ∈ *Q*和*L* ⊆ Σ*p* − *V* ，

*δ* ′(*q, L*) = *δ* (*q, L*) ∪ *δ* (*q, L* ∪ {*p*})*.*

已有结论表明，对任意Kripke结构*N* ，*B*接受*N* 当且仅当存在*ϕ*的一个模型*N* ′使

[[102](#_bookmark216)]

得*N* ↔{*p*} *N* ′

，即*B*对应于和F*µ* (*ϕ,V* )等价的*µ*-句子。

在这种情况下， 判定*M* |=? F*µ* (*ϕ,V* )问题被归约到判定是否*B*接受*M* 的问 题。 而*B*从根*r*接受一个Kripke结构*M* = (*S, R, L, r*)当且仅当Eve在参数游戏（parity game） *G* (*M , A* )上有一个从(*r, q*0)开始的赢的策略，这一问题在NP∩co-NP[[100](#_bookmark214)]中。

给定*µ*-句子*ϕ*和*ψ*，*V* 为原子命题集。 从知识进化的角度来看，以下推理问题（在

命题逻辑里也有研究[[117](#_bookmark231)]）是值得探索的：

(i) [Var-weak] *ϕ*在*ψ*的原子命题上的约束至多有*ψ*强，即*ψ* |= F*µ* (*ϕ,V* )；

(ii) [Var-strong] *ϕ*在*ψ*的原子命题上的约束至少有*ψ*强，即F*µ* (*ϕ,V* ) |= *ψ*；

1. [Var-entailment] *ϕ*在*Var*(*ϕ*) ∩ *Var*(*ψ*)上 的约束比*ψ*在*Var*(*ϕ*) ∩ *Var*(*ψ*)上 的约束强，

即F*µ* (*ϕ,V* ) |= F*µ* (*ψ,V* ),

定理 **3.7** (Entailment)**.** 给定*µ-*句子*ϕ*和*ψ*，*V* 为原子命题集，则下面的判定问题为EXPTIME*-*

完全的。

*(i)* 判定F*µ* (*ϕ,V* ) |=? *ψ*，

*(ii)* 判定*ψ* |=? F*µ* (*ϕ,V* )，

*(iii)* 判定F*µ* (*ϕ,V* ) |=? F*µ* (*ψ,V* )。

证明**.** 在EXPTIME中. 令*Aϕ* 和*Aψ* 分别为*ϕ*和*ψ*的*µ*-自动机，由命题[3.12](#_bookmark80)的证明可从*Aϕ* 构 造F*µ* (*ϕ,V* )的*µ*-自动机*B*。 由命题7.3.2[[118](#_bookmark232)]可知，可以在线性时间内构造*Aψ* 的补自动 机*C*，因此可以在线性时间内构造*C*和*B*的交自动机*AC*∩*B*。 此时，判定问题F*µ* (*ϕ,V* ) |=

?

3个人通信: Giovanna D’Agostino, 2020.

*ψ*被规约称判定*AC*∩*B* 接受的语言是否为空，这一问题时EXPTIME-完全的(定理7.5.1

[[118](#_bookmark232)])。

因此，判定是否F*µ* (*ϕ,V* ) |=? *ψ*是EXPTIME的。

EXPTIME-难. F*µ* (*ϕ,V* ) |= *ψ* 当且仅当¬F*µ* (*ϕ,V* )∨*ψ*是有效的。令*ψ* ≡ ⊥，F*µ* (*ϕ,V* ) |=

⊥ 当且仅当*ϕ* |= ⊥，判定*ϕ* |=? ⊥是EXPTIME-完全的[[100](#_bookmark214)]。 因此原问题是EXPTIME-完全

的。

(ii) 在EXPTIME中. 可类似于(i)来证明。

EXPTIME-难. *ψ* |= F*µ* (*ϕ,V* ) 当 且 仅 当¬*ψ* ∨ F*µ* (*ϕ,V* )是 有 效 的。 令*ψ* ≡ ⊤， ⊤ |= F*µ* (*ϕ,V* ) 当且仅当F*µ* (*ϕ,V* )是有效的， 当且仅当*ϕ*是有效的（或*V* = *Var*(*ϕ*)且*ϕ*是 可 满足的）。 判定*V* 是否等于*Var*(*ϕ*)能在*ϕ*大小的线性时间内完成。 此外，判定*ϕ*是有效的

（或可满足的）是EXPTIME-完全的。 因此原问题是EXPTIME-完全的。

(iii) 因为F*µ* (*ϕ,V* ) |= F*µ* (*ψ,V* )当且仅当*ϕ* |= F*µ* (*ψ,V* )， 所以这可类似(ii)来证 明。

与上面的几个推论问题类似，下面考虑这几个等价问题：Lang等人提出的“var- independence”和“var-equivalence”问题[[119](#_bookmark233)]，及“Var-match”问题:

* 1. [Var-independence] 公式*ϕ*是否独立于原子命题集*V* ，即F*µ* (*ϕ,V* ) ≡ *ϕ*，

(ii) [Var-match] *ϕ*在*ψ*的原子命题上的约束与*ψ*等价，即F*µ* (*ϕ,V* ) ≡ *ψ*，

1. [Var-equivalence] *ϕ*和*ψ*在原子命题*V* 上的约束是否等价，即F*µ* (*ϕ,V* ) ≡ F*µ* (*ψ,V* )。

推论 **3.7.** 给定*µ-*句子*ϕ*和*ψ*，*V* 为原子命题集。 则下面的判定问题为EXPTIME*-*完全的。

*(i)* 判定*ψ* ≡? F*µ* (*ϕ,V* )，

?

1. 判定F*µ* (*ϕ,V* ) ≡
2. 判定F*µ* (*ϕ,V* ) ?

≡

*ϕ*，

F*µ* (*ψ,V* )。

**3.3** 本章小结

本章基于现有不同环境下的互模拟，为了定义CTL公式的遗忘给出了Ind-结构下 的*V* -互模拟的定义。 Ind-结构间的*V* -互模拟描述的是两个Ind-结构除了*V* 中的元素之外， 它们的状态转换行为是能够互相模拟的。 这与遗忘所描述的“遗忘不想考虑的原子命 题应不影响剩余原子命题上的结论”一致。 因此，使用*V* -互模拟刻画原始公式与遗忘 结果的模型之间的关系，从而定义CTL遗忘。 遗忘作为本文主要探讨的对象，本章通 过研究*V* -互模拟的一些基本性质，探索遗忘应有的一般性质，这些性质包括：分解性、 切片性、 同质性及命题逻辑也满足的一些性质。 除了这些基本性质，还说明了本文所

贵州大学博士学位论文

给出的遗忘定义是命题逻辑下遗忘定义的扩展。 这些都为后文探索如何使用遗忘计算

最强必要条件和最弱充分条件奠定了坚实的基础。

与CTL不同，*µ*-演算是一种带变元的比CTL表达能力强的逻辑系统。 本章给出了 包含变元的互模拟定义Var-*V* -互模拟，并证明了Var-*V* -互模拟对*µ*-公式的不变性，即： 若两个赋值是Var-*V* -互模拟的，则对任意*V* -无关的公式这两个赋值同时满足（或不满 足）该公式。 证明了遗忘对*µ*-句子和不含有不动点操作的*µ*-公式总是存在的，且遗忘 与均匀插值对*µ*-句子等价的。 此外，研究了*µ*-演算下的代数性质，表明*µ*-演算遗忘也 具有分解性、 切片性和同质性。 表示性定理给出了遗忘与四条基本公设之间当且仅当 的关系。 最后，*µ*-析取公式遗忘的模型检测在NP ∩ *co*-NP中，*µ*-句子遗忘的模型检测 在EXPTIME中，而*µ*-句子遗忘的蕴涵问题是EXPTIME完全的。

# 第四章 遗忘理论在反应式系统中的应用

遗忘可以用于从本体中抽取摘要、 隐蔽敏感信息、 计算逻辑差等；*WSC*和*SNC*在 智能规划和形式化验证里有重要作用，例：在*2003*年，*Lin*使用*WSC*和*SNC*计算规划问 题中的后继状态公理。 本章针对第[一](#_bookmark4)章提出的问题*(2)*，研究*CTL*和*µ-*演算中如何使用 遗忘计算反应式系统的*SNC*（*WSC*），及使用遗忘定义知识更新。 即：

∙ 对于给定的公式*ϕ*、 原子命题*q*和原子命题集合*V* ⊆ *Var*(*ϕ*)，若满足*q* ∈ *Var*(*ϕ*) − *V* ，则*q*在*ϕ*和*V* 上的*SNC*等价于公式*ϕ* ∧ *q*遗忘不在*V* 中的原子命题后得到结果； *WSC*做为*SNC*的一个对偶概念，有相似的结论。 而不终止的系统（包括反应是系 统你）在本章看作是一个有限的初始结构，任意有限的初始结构（*K* ）都能用其 特征公式来表示（给定有限初始结构，如何计算其特征公式将在第[五](#_bookmark94)章中的介 绍，本章只需认为其是一个***CTL***公式），因而可以使用遗忘的方法来计算其*SNC*

（*WSC*）。

∙ 基于遗忘的知识更新通过最小的改变现有知识的模型来使其适应新信息，从这个 角度上可以看作是基于模型的更新，且使用这种方法的更新满足*katsuno*等人提出 的八条基本准则[*[1*](#_bookmark133)*]*。

本章其余部分组织如下：首先，第[*4.1*](#_bookmark83)节给出最强必要条件和最弱充分条件的定 义，然后探索这两个概念之间及原子命题和任意公式之间这两个概念的联系，最后给 出基于遗忘的计算方法；其次，第[*4.2*](#_bookmark88)节提出基于遗忘的知识更新方法，并证明其与基 于偏序关系的方法等价；最后，总结本章的研究工作。

**4.1** 最弱充分条件

这部分介绍如何使用遗忘理论计算最强必要条件和最弱充分条件。 直观地说，最 强必要条件指最一般的结果（the most general consequence），最弱充分条件指最特殊的 诱因（the most specific abduction）。 下面给出其形式化定义，本章所说的公式指CTL公 式，且这章所有的定义和结论都可以平凡地移到*µ*-句子下。

定义 **4.1** (充分和必要条件)**.** 给定两个公式*ϕ*和*ψ*，*V* ⊆ *Var*(*ϕ*)，*q* ∈ *Var*(*ϕ*)−*V* 和*Var*(*ψ*) ⊆

*V*。

* + 若*ϕ* |= *q* → *ψ*，则称*ψ*是*q*在*V* 和*ϕ*上的必要条件（necessary condition，NC）；
  + 若*ϕ* |= *ψ* → *q*，则称*ψ*是*q*在*V* 和*ϕ*上的充分条件（sufficient condition，SC）；

∙ 若*ψ*是*q*在*V* 和*ϕ*上的必要条件， 且对于任意*q*在*V* 和*ϕ*上的必要条件*ψ*′都有*ϕ* |= *ψ* → *ψ*′， 则 称*ψ*是*q*在*V* 和*ϕ*上 的 最 强 必 要 条 件 （strongest necessary condition， SNC）；

∙ 若*ψ*是*q*在*V* 和*ϕ*上的充分条件， 且对于任意*q*在*V* 和*ϕ*上的充分条件*ψ*′都有*ϕ* |=

*ψ*′ → *ψ*，则称*ψ*是*q*在*V* 和*ϕ*上的最弱充分条件（weakest sufficient condition，WSC）；

从上述定义可以看出，SNC（WSC）是*q*在*V* 和*ϕ*上的NC（SC）中最强（最弱）的 一个。 此外，如果公式*ψ*和*ψ*′都是*q*在*V* 和*ϕ*上的SNC（WSC），则*ψ* ≡ *ψ*′。 下面的命题 表明SNC和WSC是一对对偶概念。

命 题 **4.1** (对 偶 性)**.** 令*V*、 *q*、 *ϕ*和*ψ*为 定 义[*4.1*](#_bookmark84)出 现 的 符 号。 则*ψ*是*q*在*V* 和*ϕ*上 的*SNC*

（*WSC*）当且仅当¬*ψ*是¬*q*在*V* 和*ϕ*上的*WSC*（*SNC*）。

证明**.** (i) 假定*ψ*是*q*在*V* 和*ϕ*上的SNC，*ψ*′是¬*q*在*V* 和*ϕ*上SC。

*ϕ* |= *q* → *ψ*

⇒ *ϕ* |= ¬*ψ* → ¬*q*。

这表明¬*ψ*是¬*q*在*V* 和*ϕ*的SC。

*ϕ* |= *ψ*′ → ¬*q*

⇒ *ϕ* |= *q* → ¬*ψ*′，即：¬*ψ*′是*q*在*V* 和*ϕ*上的NC

⇒ *ϕ* |= *ψ* → ¬*ψ*′ (假设)

⇒ *ϕ* |= *ψ*′ → ¬*ψ*。

另一个方向可以类似地证明。

(ii) 可以类似SNC的情形证明。

在定义[4.1](#_bookmark84)中，用公式*α*替换*q*，*V* ⊆ *Var*(*α*) ∪ *Var*(*φ* )，则为公式的最强必要条件和

最弱充分条件的定义。 下面的命题表明了原子命题的充分（必要）条件与公式的充分

（必要）条件之间的关系：通过计算原子命题的充分（必要）条件来计算公式的充分

（必要）条件。

命题 **4.2.** 给定公式Γ和*α*，*V* ⊆ *Var*(*α*) ∪ *Var*(Γ)， *q*是不出现在Γ和*α*中的原子命 题。

*ϕ*是集合*V* 上的公式，则*ϕ*是*α*在*V* 和Γ上的*SNC*（*WSC*）当且仅当*ϕ*是*q*在*V* 和Γ′上的*SNC*

（*WSC*），其中Γ′ = Γ ∪ {*q* ↔ *α*}。

证明**.** 这里证明SNC的情形，WSC的情形可以类似地证明。记*SNC*(*ϕ, β,V,* Γ)为*ϕ*是*β* 在*V* 和Γ上

的SNC，*NC*(*ϕ, β,V,* Γ)为*ϕ*是*β* 在*V* 和Γ上的NC。

(⇒) 下证*SNC*(*ϕ, α,V,* Γ) ⇒ *SNC*(*ϕ, q,V,* Γ′)。 假设*ϕ*′是*q*在*V* 和Γ′上的NC。

Γ |= *α* → *ϕ* (已知)

⇒ Γ ∧ (*q* ↔ *α*) |= *α* → *ϕ*

⇒ Γ′ |= *q* → *ϕ* (*q* ≡ *α*)

Γ′ |= *q* → *ϕ*′ (假设)

⇒ Γ′ |= *α* → *ϕ*′ (*q* ≡ *α*)

⇒ F*µ* (Γ′*,* {*q*}) |= *α* → *ϕ*′ (IR(*α* → *ϕ*′*,* {*q*}), (**PP**))

⇒ Γ |= *α* → *ϕ*′ (引理 [3.4](#_bookmark72))

⇒ Γ |= *ϕ* → *ϕ*′ (已知: *SNC*(*ϕ, α,V,* Γ))

⇒ Γ′ |= *ϕ* → *ϕ*′.

总之，*ϕ*是*q*在*V* 和Γ′上的NC，且对任意*q*在*V* 和Γ′上的NC*ϕ*′，有Γ′ |= *ϕ* → *ϕ*′。 因

此，*SNC*(*ϕ, q,V,* Γ′)。

(⇐) 下证*SNC*(*ϕ, q,V,* Γ′) ⇒ *SNC*(*ϕ, α,V,* Γ)。 假定*ϕ*′是*α*在*V* 和Γ上的NC。

Γ′ |= *q* → *ϕ* (已知)

⇒ Γ′ |= *α* → *ϕ* (*q* ≡ *α*)

⇒ F*µ* (Γ′*,* {*q*}) |= *α* → *ϕ* (IR(*α* → *ϕ,* {*q*}), (**PP**))

⇒ Γ |= *α* → *ϕ* (引理 [3.4](#_bookmark72))

Γ |= *α* → *ϕ*′ (假设)

⇒ Γ′ |= *α* → *ϕ*′

⇒ Γ′ |= *q* → *ϕ*′ (*q* ≡ *α*)

⇒ Γ′ |= *ϕ* → *ϕ*′ (已知：*SNC*(*ϕ, q,V,* Γ′))

⇒ F*µ* (Γ′*,* {*q*}) |= *ϕ* → *ϕ*′ (IR(*ϕ* → *ϕ*′*,* {*q*}), (**PP**))

⇒ Γ |= *ϕ* → *ϕ*′ (引理 [3.4](#_bookmark72))

总之，*ϕ*是*α*在*V* 和Γ上的NC，且对任意*α*在*V* 和Γ上的NC*ϕ*′，有Γ |= *ϕ* → *ϕ*′。因此，

*SNC*(*ϕ, α,V,* Γ)。

例 **4.1** (例[3.1](#_bookmark49)的延续)**.** 本例来源于图[3.1](#_bookmark49)中的初始结构*K*2。令*ψ* = EX(*s* ∧(EX*se* ∨ EX¬*d*))、

*ϕ* = EX(*s* ∧ EX¬*d*)、*A* = {*d, s, se*}和*V* = {*s, d*}。 下面证明*ϕ*是*ψ*在*V* 和*K*2上的WSC：

(i) 由已知有*ϕ* |= *ψ*和*Var*(*ϕ*) ⊆ *V*。 此外，(*M , s*0) |= *ϕ* ∧ *ψ*，因此*K*2 |= *ϕ* → *ψ*，即： *ϕ*是*ψ*在*V* 和*K*2上的SC;

(ii) 下证“对任意*ψ*在*V* 和*K*2上的SC *ϕ*′都有*K*2 |= *ϕ*′ → *ϕ*”。易知若*K*2 |= *ϕ*′，则*K*2 |= *ϕ*′ → *ϕ*。 假设*K*2 |= *ϕ*′。 由*ϕ*′是*ψ*在*V* 和*K*2上的SC可知*ϕ*′ |= EX(*s* ∧ *φ* )，其中*φ* 是 使得*φ* |= EX*se* ∨ EX¬*d*成立的公 式。 又IR(*ϕ*′*,V* )， 所以*φ* |= EX¬*d*。 因此， *ϕ*′ |= *ϕ*且*K*2 |= *ϕ*′ → *ϕ*。

如何使用遗忘计算SNC（WSC）是本章讨论的关键问题。 下面首先给出其理论基

础，然后再做直观的解释。

定理 **4.1.** 给定公式*ϕ*、原子命题集*V* ⊆ *Var*(*ϕ*)和原子命题*q* ∈ *Var*(*ϕ*) − *V*。

1. FCTL(*ϕ* ∧ *q,* (*Var*(*ϕ*) ∪ {*q*}) − *V* )是*q*在*V* 和*ϕ*上的*SNC*；
2. ¬FCTL(*ϕ* ∧ ¬*q,* (*Var*(*ϕ*) ∪ {*q*}) − *V* )是*q*在*V* 和*ϕ*上的*WSC*。

证明**.** (i) 令*F* = FCTL(*ϕ* ∧ *q,* (*Var*(*ϕ*) ∪ {*q*}) − *V* )。

“NC”部分：由遗忘的定义可知*ϕ* ∧*q* |= *F*。因此，*ϕ* |= *q* → *F* ，即：*F* 是*q*在*V* 和*ϕ*上

的NC。

“SNC” 部 分： 假 设*ψ*′为*q*在*V* 和*ϕ*上 的 任 意NC， 即： *ϕ* |= *q* → *ψ*′。 由 定 理[3.1](#_bookmark52) 和*IR*(*ψ*′*,* (*Var*(*ϕ*) ∪ {*q*}) − *V* )可知，若*ϕ* ∧ *q* |= *ψ*′，则*F* |= *ψ*′。 由假设可知*ϕ* |= *q* → *ψ*′， 所以*ϕ* ∧ *F* |= *ψ*′，因此*ϕ* |= *F* → *ψ*′。

由上面两部分可知，*F* 是*q*在*V* 和*ϕ*上的SNC。

(ii) 令*F* = ¬FCTL(*ϕ* ∧¬*q,* (*Var*(*ϕ*)∪{*q*})−*V* )。由命题[4.1](#_bookmark85)可知，对任意命题*q*′，FCTL (*ϕ* ∧

*q*′*,* (*Var*(*ϕ*) ∪ {*q*′}) − *V* )是*q*′在*V* 和*ϕ*上 的SNC， 当 且 仅 当¬FCTL(*ϕ* ∧ *q*′*,* (*Var*(*ϕ*) ∪ {*q*′}) −

*V* )是¬*q*′在*V* 和*ϕ*上的WSC。由(i)可知FCTL (*ϕ* ∧*q*′*,* (*Var*(*ϕ*)∪{*q*′})−*V* )是*q*′在*V* 和*ϕ*上的SNC， 所以¬FCTL(*ϕ* ∧*q*′*,* (*Var*(*ϕ*)∪{*q*′})−*V* )是¬*q*′在*V* 和*ϕ*上的WSC。令*q* = ¬*q*′，可得*F* 是*q*在*V* 和*ϕ*上 的WSC。

令*F* = FCTL(*ϕ* ∧ *q,* (*Var*(*ϕ*) ∪{*q*}) −*V* )。上面的定理可以直观地解释如下：由遗忘理 论的定义可知*ϕ* ∧ *q* |= *F* ，这说明*F* 是*q*在*V* 和*ϕ*上的NC；对任意与(*Var*(*ϕ*) ∪ {*q*}) − *V* 无 关的公式*ψ*，若*ϕ* ∧ *q* |= *ψ*，则由表达性定理可知*F* |= *ψ*。

由第[五](#_bookmark94)章可知，任意有限的初始结构都能由一个CTL公式表示，所以由上面的定

理自然地就能得到给定有限初始结构下的SNC和WSC。

推论 **4.1.** 令*K* = (*M , s*0)为初始结构，其中*M* = (*S, R, L, s*0)为有限原子命题集合*A* 上的

初始*-Kripke*结构，*V* ⊆ *A* 且*q* ∈ *V* ′ = *A* − *V*。 则：

1. FCTL(*FA* (*K* ) ∧ *q,V* ′)是*q*′在*V* 和*K* 上的*SNC*；
2. ¬FCTL(*FA* (*K* ) ∧ ¬*q,V* ′)是*q*′在*V* 和*K* 上的*WSC*。

例 **4.2** (例 [1.1](#_bookmark8)的延续)**.** 令*A* = {*d, se, sp, s*}和*V* = {*d, se*}，求*s*在*V* 和初始结构*K* = (*M , s*0)上

的WSC，其中*M* 为例 [1.1](#_bookmark8)中初始状态为*s*0的汽车制造企业模型结构。

由定理 [4.1](#_bookmark86)可知，*s*在*V* 和初始结构*K* = (*M , s*0)上的WSC为¬FCTL(*FA* (*K* )∧¬*s,* {*s*}∪

{*sp*})。

由于涉及到后文中遗忘的计算方法，本例的详细计算过程放到第 [5.1](#_bookmark95)。

第四章 遗忘理论在反应式系统中的应用

**4.2** 知识更新

本小节介绍遗忘的另一个应用：知识更新（Knowledge update）。具体说来，本节将 使用遗忘定义知识更新，使得用这种方式定义的知识更新满足下面由Katsuno和Mendelzon提 出的基本条件[[1](#_bookmark133)]：

∙ (U1) Γ ◇ *ϕ* |= *ϕ*；

∙ (U2) 若Γ |= *ϕ*，则Γ ◇ *ϕ* ≡ Γ；

∙ (U3) 若Γ和*ϕ*都是可满足的，则Γ ◇ *ϕ*是可满足的；

∙ (U4) 若Γ1 ≡ Γ2且*ϕ*1 ≡ *ϕ*2，则Γ1 ◇ *ϕ*1 ≡ Γ2 ◇ *ϕ*2；

∙ (U5) (Γ ◇ *ϕ*) ∧ *ψ* |= Γ ◇ (*ϕ* ∧ *ψ*)；

∙ (U6) 若Γ ◇ *ϕ* |= *ψ*且Γ ◇ *ψ* |= *ϕ*，则Γ ◇ *ϕ* ≡ Γ ◇ *ψ*；

∙ (U7) 若Γ有唯一一个模型，则(Γ ◇ *ϕ*) ∧ (Γ ◇ *ψ*) |= Γ ◇ (*ϕ* ∨ *ψ*)；

∙ (U8) (Γ1 ∨ Γ2) ◇ *ϕ* ≡ (Γ1 ◇ *ϕ*) ∨ (Γ2 ◇ *ϕ*)。

其中，◇为知识更新操作，*ϕ* ◇ *ψ*表示用*ψ*更新*ϕ*得到的结果。

本小节假设所有初始结构都是有限的，即：状态来源于有限状态空间且*A* 为有限

原子命题集。

由第[五](#_bookmark94)章可知，任意*A* 上的有限初始结构*M*（为了简化符号，本节用初始Kripke结 构*M* 代替初始结构(*M , s*0)）都能用一个CTL公式——特征公式*FA* (*M* )来表示。 给定 公式*ϕ*和*ψ*，*Vmin* ⊆ *A* 为使得FCTL(*ϕ,Vmin*) ∧ *ψ*可满足的极小子集。 此外，记

｜

*Vmin*⊆*A*

*Mod*(FCTL(*FA* (*M* )*,Vmin*) ∧ *ψ*)

为所有FCTL(*FA* (*M* )*,Vmin*) ∧ *ψ*的模型集合的并集。 此时，可如下定义*µ*-演算下的知识

更新操作：

定义 **4.2.** 给定公式Γ和*ϕ*。 知识更新操作◇*CTL*定义如下：

*Mod*(Γ ◇*CTL ϕ*) = ｜

｜ *Mod*(FCTL(*FA* (*M* )*,Vmin*) ∧ *ϕ*)*,*

*M* ∈*Mod*(Γ) *Vmin*⊆*A*

其中，*FA* (*M* )是*M* 在*A* 上的特征公式，*Vmin* ⊆ *A* 是使得FCTL(*FA* (*M* )*,Vmin*)可满足的

极小子集。

直观上，Γ ◇CTL *ϕ*表示：通过极小化改变Γ的模型，使其与*ϕ*的模型一致来更新Γ。

所以，这样定义的知识更新是一种基于模型的知识更新方法。

此外，CTL知识更新也可以用类似于命题逻辑里基于偏序关系的更新来定义：

令*I*，*J*1和*J*2为三个解释，即：原子命题集；则*J*1比*J*2更接近*I*（记为：*J*1 ≤*I,pma J*2）当且

仅当Diff(*I, J*1) ⊆ Diff(*I, J*2)，其中Diff(*X,Y* ) = {*p* ∈ *A* | *X* (*p*) ̸= *Y* (*p*)}。 那么命题逻辑知 识更新——用*ψ*更新Γ，即为*ψ*的关于偏序关系≤*I,pma*的所有极小模型的集合（*I*是Γ的模 型），即：

*Mod*(Γ ◇*pma ψ*) = ｜

*I*∈*Mod*(Γ)

*Min*(*Mod*(*ψ*)*,* ≤*I,pma*)*.*

其中，*Min*(*Mod*(*ψ*)*,* ≤*I,pma*)是*ψ*的关于偏序关系≤*I,pma*的极小模型集。

类似地，这里定理有限初始结构之间关于另一个初始结构的偏序关系。

定义 **4.3.** 给定三个有限初始结构*M*、 *M*1和*M*2， *M*1比*M*2更接近*M* （记为*M*1 ≤*M M*2），当且仅当对任意*V*2 ⊆ *A* ，若*M*2 ↔*V*2 *M* ，则存在*V*1 ⊆ *V*2使得*M*1 ↔*V*1 *M*。*M*1 *<M M*2当且仅当*M*1 ≤*M M*2且*M*2 ̸≤*M M*1。

在这种情形下，若*M*1 ≤*M M*2，则存在一个原子命题的极小集*Vmin*使得*M*1 ↔*Vmin*

*M* ，且对任意*V* ⊆ *A* ，若*M*2 ↔*V M* 则*Vmin* ⊆ *V* 且*M*1 ↔*V M*。

例 **4.3.** 令*M* = (*S, R, L, r*)、*M*1 = (*S*1*, R*1*, L*1*, r*1)、*M*2 = (*S*2*, R*2*, L*2*, r*2)为三个初始结构（如 图 [4.1](#_bookmark90)），其中*S* = *S*1 = *S*2 = {*s*0*, s*1}，*r* = *r*1 = *r*2 = *s*0，*R* = *R*1 = *R*2 = {(*s*0*, s*1)*,* (*s*1*, s*1)}， *L*(*s*0) = {*ch, j*}，*L*1(*s*0) = *L*2(*s*0) = {*ch*}，*L*(*s*1) = *L*1(*s*1) = 0/ ，*L*2(*s*1) = { *j*}。

可以检查*M* ↔{ *j*} *M*1，*M* ↔{ *j,ch*} *M*2，{ *j*} ⊆ { *j, ch*}，且对任意原子命题集*V* ′ ⊂ { *j*}

（或*V* ′ ⊂ { *j, ch*}），有*M* ̸↔*V* ′ *M*1（或*M* ̸↔*V* ′ *M*2）。 因此，*M*1 ≤*M M*2。

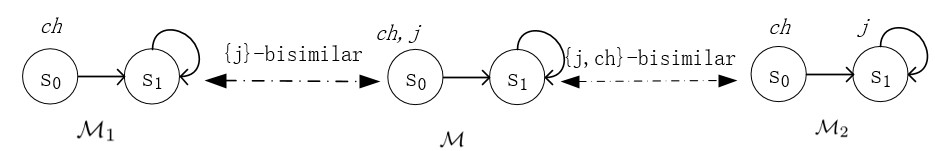


图 4.1: 初始结构间的≤*M* 关系。

给定有限初始结构集*M*和有限初始结构*M* ，用*Min*(*M,* ≤*M* )表示*M*中关于偏序关

系≤*M* 的极小有限初始结构集。 则≤*M* 与知识更新操作◇CTL有如下关系。

定理 **4.2.** 给定*µ-*句子Γ和*ϕ*，则：

*Mod*(Γ ◇*CTL ϕ*) = ｜

*M* ∈*Mod*(Γ)

*Min*(*Mod*(*ϕ*)*,* ≤*M* )*.*

证明**.** *M* ′ ∈ *Mod*(Γ ◇CTL *ϕ*)

⇔ 存在*M* |= Γ和极小子集*Vmin* ⊆ *A* ，使得*M* ′ ∈ *Mod*(F*µ* (*FA* (*M* ), *Vmin*) ∧ *ϕ*)

⇔ 存 在*M* |= Γ和*Vmin* ⊆ *A* ， 使 得*M* ↔*Vmin M* ′和*M* ′ |= *ϕ*， 且*Vmin*是 使 得F*µ* (*FA* (*M* ),

*Vmin*) ∧ *ϕ*可满足的*A* 极小子集

⇔ 存在*M* |= Γ，使得对任意*M* ′′ |= *ϕ*，*M* ′ ≤*M M* ′′

⇔ *M* ′ ∈ *Min*(*Mod*(*ϕ*)*,* ≤*M* )。

从定理[4.2](#_bookmark91)可以看出，通过遗忘定义的知识更新操作与通过有限初始结构间的偏序 关系定义的知识更新一致，且通过遗忘定义的知识更新操作满足Katsuno和Mendelzon提 出的八条基本条件。

定理 **4.3.** 知识更新操作◇*CTL*满足*Katsuno*和*Mendelzon*提出的基本条件*(U1)-(U8)*。

证明**.** (U1). 由定理 [4.2](#_bookmark91)可知，*Mod*(Γ ◇*µ ϕ*) ⊆ *Mod*(*ϕ*)，因此Γ ◇*µ ϕ* |= *ϕ*。

(U2). *M* ′ ∈ *Mod*(Γ ◇*µ ϕ*)

⇔ 存在*M* |= Γ，使得对任意*M* ′′ |= *ϕ* 和*V* ⊆ *A* ，*M* ′′ ↔*V M* 蕴涵*M* ′ ↔*V M*

⇔ 存在*M* |= Γ和*Vmin* = 0/ ，使得*M* ′ ↔*Vmin M* (Γ |= *ϕ*)

⇔ *M* ′ |= Γ。

容易证明◇*µ* 满足(U3)和(U4)。

(U5). *M* ′ |= (Γ ◇*µ ϕ*) ∧ *ψ*

⇒ 存在*M* |= Γ，使得对任意*M* ′′ |= *ϕ* ∧ *ψ* 和*V* ⊆ *A* ，*M* ′′ ↔*V M* 蕴涵*M* ′ ↔*V M*

⇒ 存在*M* |= Γ和*Vmin* ⊆ *A* ，使得*M* ′ ↔*Vmin M* 和*M* ′ |= *ϕ* ∧ *ψ*，其中*Vmin*是使得*M* ′ ↔*Vmin*

*M* 的*A* 的极小子集

⇒ *M* ′ |= Γ ◇*µ* (*ϕ* ∧ *ψ*)。

(U6). *M* ′ |= Γ ◇CTL *ϕ*

⇔ 存在*M* |= Γ，使得对任意*M* ′′ |= *ϕ* 和*V* ⊆ *A* ，*M* ′′ ↔*V M* 蕴涵*M* ′ ↔*V M*

⇔ 存在*M* |= Γ和*Vmin* ⊆ *A* ，使得*M* ′ ↔*Vmin M* ，其中*Vmin*是使得F*µ* (*FA* (*M*1)*,Vmin*) ∧ *ϕ* 和F*µ* (*FA* (*M*1)*,Vmin*) ∧ *ψ*是一致的 (Γ ◇CTL *ϕ* |= *ψ*, Γ ◇CTL *ψ* |= *ϕ*) %(*Otherwise, suppose that V* ⊂ *Vmin s.t.* F*µ* (*FA* (*M*1)*,V* ) ∧ *ψ is consistent as well. Then,* F*µ* (*FA* (*M*1)*,V* )

∧*ϕ should also be consistent by* Γ ◇*CTL ϕ* |= *ψ, which contradicts to the fact that Vmin is the*

*minimal set of atoms s.t.* F*µ* (*FA* (*M*1)*,Vmin*) ∧ *ϕ is consistent.*)

⇔ 存在*M* |= Γ，使得对任意*M* ′′ |= *ψ* 和*V* ⊆ *A* ，*M* ′′ ↔*V M* 蕴涵*M* ′ ↔*V M*

⇔ *M* ′ |= Γ ◇CTL *ψ*。

(U7). *M* ′ |= (Γ ◇CTL *ϕ*) ∧ (Γ ◇CTL *ψ*)，且设*M* 是Γ的唯一模型

⇒ 存在两个极小子集*V*1*,V*2 ⊆ *A* ，使得*M* ↔*V*1 *M* ′ 和*M* ↔*V*2 *M* ′

⇒ 存在两个极小子集*V*1*,V*2 ⊆ *A* ，使得*M* ′ |= F*µ* (*FA* (*M* )*,V*1) ∧ *ϕ* 和*M* ′ |= F*µ* (*FA* (*M* )*,*

*V*2) ∧ *ψ*是一致的

⇒ *M* ′ |= F*µ* (*FA* (*M* )*,V*1 ∩ *V*2)，*M* ′ ↔*V*1∩*V*2 *M* ，*V*1 = *V*2

⇒ *V*1是使得F*µ* (*FA* (*M* )*,V*1) ∧ (*ϕ* ∨ *ψ*)可满足的极小子集 %(*Otherwise, suppose that V*3 ⊂ *V*1 *s.t.* F*µ* (*FA* (*M* )*, V*3)∧(*ϕ* ∨*ψ*) *is satisfiable. Then* F*µ* (*FA* (*M* )*, V*3)∧*ϕ or* F*µ* (*FA* (*M* )*,*

*V*3) ∧ *ψ is satisfiable. Without loss of generality, suppose that* F*µ* (*FA* (*M* )*,V*3) ∧ *ϕ is satisfi- able, V*1 *is not the minimal set, a contradiction.*)

⇒ *M* ′ |= Γ ◇CTL (*ϕ* ∨ *ψ*)。

(U8). *M* |= (Γ1 ∨ Γ2) ◇CTL *ϕ*

⇔ 存在*M*1 |= Γ1 (或*M*1 |= Γ2) 和一个极小子集*Vmin*，使得*M* ↔*Vmin M*1

⇔ *M* |= (Γ1 ◇CTL *ϕ*) ∨ (Γ2 ◇CTL *ϕ*)。

例 **4.4.** 令*A* = {*ch, j*}、 *ϕ* = *νX. j* ∧ *ch* ∧ EXEX*X*、 *ψ* = *νX.*¬ *j* ∧ *ch* ∧ EXEX*X* 且Kripke结构

的状态空间为{*s*0*, s*1}，则用*ψ*更新*ϕ*计算如下：

*Mod*(*ϕ*) ={((1)*, r* = *s*0*, L*(*s*0) = {*ch, j*}*, L*(*s*1) = {*ch, j*})*,*

((2)*, r* = *s*1*, L*(*s*1) = {*ch, j*}*, L*(*s*0) = {*ch, j*})*,*

((3)*, r* = *s*0*, L*(*s*0) = {*ch, j*}*, L*(*s*1) = *C* )*,*

((4)*, r* = *s*1*, L*(*s*1) = {*ch, j*}*, L*(*s*0) = *C* )*,*

((5)*, r* = *s*0*, L*(*s*0) = {*ch, j*}*, L*(*s*1) = *C* )*,*

((6)*, r* = *s*1*, L*(*s*1) = {*ch, j*}*, L*(*s*0) = *C* )*, . . .* }

*Mod*(*ψ*) ={((1)*, r* = *s*0*, L*(*s*0) = {*ch*}*, L*(*s*1) = {*ch*})*,*

((2)*, r* = *s*1*, L*(*s*1) = {*ch*}*, L*(*s*0) = {*ch*})*,*

((3)*, r* = *s*0*, L*(*s*0) = {*ch*}*, L*(*s*1) = *C* )*,*

((4)*, r* = *s*1*, L*(*s*1) = {*ch*}*, L*(*s*0) = *C* )*,*

((5)*, r* = *s*0*, L*(*s*0) = {*ch*}*, L*(*s*1) = *C* )*,*

((6)*, r* = *s*1*, L*(*s*1) = {*ch*}*, L*(*s*0) = *C* )*, . . .* }

其中，四元组((*i*)*, r* = *sk, L*(*s*0) = *V*1*, L*(*s*1) = *V*1)表示Kripke结构(*S, r, R, L*)，其中*S* = {*s*0*, s*1}、 *r* = *sk* (*r* ∈ {0*,* 1})、 转换关系如图 [4.2](#_bookmark93)中的(i) (*i* ∈ {1*,* 2*,* 3*,* 4*,* 5*,* 6}[1](#_bookmark0))、 *s*0 和*s*1分别被*V*1 ⊆

{*ch, j*} 和*V*2 ⊆ {*ch, j*}标记且*C* ∈ {0/ *,* { *j*}*,* {*ch*}*,* { *j, ch*}}。

*Mod*(*ϕ* ◇*µ ψ*) = J

*M* ∈*Mod*(*ϕ*)

*Min*(*Mod*(*ψ*)*,* ≤*M* )，根据定义 [4.3](#_bookmark89)容易检查*Mod*(*ϕ* ◇*µ ψ*) =

*Mod*(*ψ*)。 直观地说，由于在*ψ*中 *j*在偶数状态不再为真、 *ch*保持为真且*ψ*和*ϕ*都不知道

模型偶数状态的信息，因而用*ψ*更新*ϕ*得到的结果为*ψ*自身。

**4.3** 本章小结

本章讨论了如何使用遗忘计算最强必要条件（最弱充分条件）和定义知识更新。

特别地，本章首先给出了SNC（WSC）的定义，表明SNC和WSC是一对对偶概念，因

1这里只列出部分转换关系，其余转换关系可以容易地枚举出来。

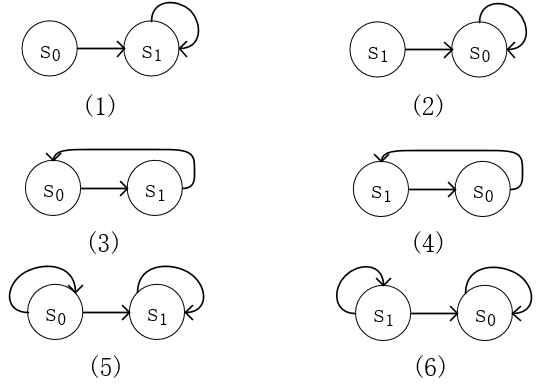


图 4.2: 状态空间为{*s*0*, s*1}的六个Kripke结构示意图

而只要知道其一就能知道另一个。 其次，结论表明任意公式的SNC（WSC）可以转 换成原子命题的SNC（WSC）来计算，并给出使用遗忘计算原子命题在给定条件下 得SNC（WSC）的方法。 最后，分别给出使用遗忘和偏序关系的方法定义知识更新， 并证明了这两种定义等价且满足Katsuno等人提出的八条准则。

第五章 **CTL**遗忘计算方法

第 [三](#_bookmark45)章探索了*CTL*遗忘的定义及其基本性质，本章给出两种计算*CTL*遗忘的算法：

基于模型的算法和基于归结的算法*CTL-forget*。 基于模型的计算方法考虑固定有限状态空间下的初始结构。 本章证明这样的初始

结构可由一个*CTL*公式来表示，因而其遗忘结果可以由*CTL*公式表示。

基于归结的计算方法首先将*CTL*公式转化为SNF*g*

CTL

子句，然后使用归结系统*R*≻*,S* 中

的规则对要遗忘的原子命题归结，最后将得到的结果等价转化为*CTL*公式。 基于此，

*CTL*

本章详细介绍了基于*Prolog*的*CTL-forget*算法实现，并给出部分实验结果及结果分析。 最后总结本章的主要工作。

**5.1** 基于模型的有界**CTL**遗忘计算

遗忘与均匀插值是一对对偶概念。 已有研究表明CTL不具有均匀插值性质[[81](#_bookmark197)]，这 表明CTL遗忘不是封闭的[1](#_bookmark0)。 此时，探索CTL下遗忘封闭的情形对深入应用遗忘有重要 意义。 为此，本章首先提出有限初始结构的特征公式；其次，表明CTL公式的遗忘结 果在此情形下可以表示成其模型的特征公式的析取；最后，提出一种基于模型的方法 计算遗忘。

现实情况下能处理的系统都是有限的，且在某一固定环境下所涉及到的原子命题

是有限的。 因此，在这部分讨论一种约束的CTL，即：

∙ （1）出现在CTL公式中的原子命题的个数是有限的（即*A* 是有限的）；

∙ （2）初始结构的状态空间*S*是一个有限的固定状态空间*S* = {*b*1*,..., bm*}的子集

（即*S* ⊆ *S* ），且使得对于任意约束长度的CTL公式*ϕ*，若*ϕ*是可满足的，则存在一 个初始结构(*M , s*0)，使得(*M , s*0) |= *ϕ*且其状态空间是*S* 的子集。

由此可见，在这种情形下只有有限个初始结构会被考虑。 下文表明在这一约束条件 下CTL中的遗忘是封闭的。

本节其余部分组织如下：首先，在第[5.1.1](#_bookmark96)节定义一种有界*V* -互模拟的概念，然后 给出初始结构的特征公式；其次，在第[5.1.2](#_bookmark106)节描述约束CTL下的遗忘是封闭的，并给 出计算方法；最后，在第[5.1.3](#_bookmark110)节给出基于模型的计算遗忘的算法，并分析算法的复杂 性。

1对于给定的逻辑语言*L* 和该语言上的操作*O*，若*O*作用到*L* 中的元素后得到的结果仍然在*L* 中，则称*O*在*L* 下

是封闭的。

**5.1.1** 描述初始结构

本小节介绍与一个初始结构相关的CTL公式——特征公式。 对于一个给定的初始 结构，其特征公式与其计算树的特征公式密切相关。 因此，本节首先介绍计算树之间 的*V* -互模拟关系；然后，给出计算树的特征公式的定义；最后，给出初始结构的特征 公式。

**5.1.1.1** 计算树*V* **-**互模拟

在第[三](#_bookmark45)章中的定义[3.1](#_bookmark48)给出了Ind-结构之间的*V* -互模拟定义及其相关性质，结构之 间的*V* -互模拟有相同的定义，即：Ind-结构之间*V* -互模拟具有的性质都可以平凡的迁 移到结构之间*V* -互模拟。 这里给出有限结构间的有界*V* -互模拟（与深度*n*关联的*V* -互模 拟）的定义，并证明*V* -互模拟与有界*V* -互模拟在有限结构下等价。 因此，第[三](#_bookmark45)章遗忘 有的性质在本章约束情形下基本也有[[120](#_bookmark234)]，这里不再赘述。

首先给出能够描述一定深度*n* ∈ N的计算树之间的*V* -互模拟关系，记为*BV*。 令*V* ⊆

*n*

*A* 是原子命题集，*i* ∈ {1*,* 2}，*Mi* = (*Si, Ri, Li, si* )是初始Kripke结构，*Ki* = (*Mi, si*)是结

0

构。 *BV* 递归定义如下：

*n*

∙ 若*L*1(*s*1) − *V* = *L*2(*s*2)，则(*K*1*, K*2) ∈ *BV* ；

0

∙ 对任意*n* ≥ 0，若满足下面几个条件，则(*K*1*, K*2) ∈ *BV*

*n*+1

成立：

**–** (*K*1*, K*2) ∈ *BV* ；

0

**–** 对任意(*s*1*, s*′ ) ∈ *R*1，存在(*s*2*, s*′ ) ∈ *R*2，使得(*K* ′*, K* ′) ∈ *BV* ；

1 2 1 2 *n*

**–** 对任意(*s*2*, s*′ ) ∈ *R*2，存在(*s*1*, s*′ ) ∈ *R*1，使得(*K* ′*, K* ′) ∈ *BV*。

2

其中*Ki* ′ = (*Mi, s*′)。

*i*

1 1 2 *n*

当所谈及的原子命题集*V* 很显然的时候，上述*BV* 中的*V* 可以省略，记为*Bn*。 此

*n*

外，当讨论的*Mi* (*i* = 1*,* 2)是显然的时候，可以使用(*s*1*, s*2) ∈ *Bn*代替((*M*1*, s*1)*,* (*M*2*, s*2))

∈ *Bn*。 有界*V* -互模拟关系就可以定义如下：

定义 **5.1** (有界*V* -互模拟)**.** 令*V* 是*A* 的一个子集，*i* ∈ {1*,* 2}，*K*1和*K*2是结构。

∙ *K*1和*K*2是有界*V-*互模拟的，当且仅当对所有*n* ≥ 0，都有(*K*1*, K*2) ∈ *Bn*。若*K*1和*K*2是

有界*V-*互模拟的，则记为*K B K* 。

1 ↔*V* 2

[2](#_bookmark0) *B B*

∙ 对*Mi*上的路径*πi* = (*si,*1*, si,*2*, . . .* )，若对于任意 *j* ∈ N≥1 ，都有*K*1*, j* ↔*V K*2*, j*，则*π*1 ↔*V*

*π*2。 其中*Ki, j* = (*Mi, si, j* )。

值得注意的是，满足有界*V* -互模拟关系的结构之间有且仅有一个有界*V* -互模拟关

系，即*Bk*+1 = *Bk*（*k* ∈ N）。

2N为整数集，N

≥1

是大于等于1的整数集。

引理 **5.1.** 对于有限初始*Kripke*结构，有界*V-*互模拟是一个*V -*互模拟。

证明**.** 令*V* ⊆ *A* 、 *Ki* = (*Mi, si*) (*i* ∈ {1*,* 2})和*Mi* = (*Si, Ri, Li, si* )为有限的初始-Kripke结 构。因为*Si* (*i* = 1*,* 2)是有限的，所以*Bn* ⊆ *S*1 × *S*2 (*n* ≥ 0)都是有限的。由于对任意*n* ≥ 0， 都有*Bn*+1 ⊆ *Bn*。 因此，存在一个最小数*k*，使得*Bk*+1 = *Bk*（用*B*表示）。

0

下证对任意*r*1 ∈ *S*1和*r*2 ∈ *S*2，有(*r*1*, r*2) ∈ *B*蕴涵

(a) *L*1(*r*1) − *V* = *L*2(*r*2) − *V* ；

(b) ∀*w*1 ∈ *S*1，若(*r*1*, w*1) ∈ *R*1，则∃*w*2 ∈ *S*2，使得(*r*2*, w*2) ∈ *R*2和(*w*1*, w*2) ∈ *B*；

(c) ∀*w*2 ∈ *S*2，若(*r*2*, w*2) ∈ *R*2，则∃*w*1 ∈ *S*1，使得(*r*1*, w*1) ∈ *R*1和(*w*1*, w*2) ∈ *B*。

首先，由对任意*n* ≥ 0，都有(*r*1*, r*2) ∈ *Bn*，可知(*r*1*, r*2) ∈ *B*0，这表明*L*1(*r*1) − *V* =

*L*2(*r*2) − *V*。 因此(a)成立。

其次，令*w*1 ∈ *S*1和(*r*1*, w*1) ∈ *R*1。 有

(*r*1*, r*2) ∈ *B*、*w*1 ∈ *S*1和(*r*1*, w*1) ∈ *R*1

⇒ 由*Bk*+1 = *B*可知(*r*1*, r*2) ∈ *Bk*+1、*w*1 ∈ *S*1和(*r*1*, w*1) ∈ *R*1

⇒ 由定义可知∃*w*2 ∈ *S*2，使得(*r*2*, w*2) ∈ *R*2和(*w*1*, w*2) ∈ *Bk*

⇒ 由*B* = *Bk*可知∃*w*2 ∈ *S*2，使得(*r*2*, w*2) ∈ *R*2和(*w*1*, w*2) ∈ *B*

⇒ (b)成立。

可以类似地证明(c)也成立。 因此，*B*是*M*1和*M*2之间的一个*V* -互模拟关系。

由引理[5.1](#_bookmark97)可知，*B*既是一个有界*V* -互模拟关系也是一个*V* -互模拟关系。 因此，下

面的推论显然成立。

推论 **5.1.** 令*V* ⊆ *A* 和*Ki* = (*Mi, si*)（*i* ∈ {1*,* 2}）。若*Mi* = (*Si, Ri, Li, si* )是有限的初始*Kripke*结

0

构，则*s*1和*s*2是有界*V-*互模拟的，当且仅当*s*1 ↔*V s*2。

本章只涉及有限的初始Kripke结构，因此，本章用↔ （*V* -互模拟）表示 *B* （有

*V* ↔*V*

界*V* -互模拟）。

给定原子命题集*V* ⊆ *A* 和初始Kripke结构*Mi*（*i* = 1*,* 2）。 如果下面条件同时满足：

∙ *L*1(*s*1) − *V* = *L*2(*s*2) − *V* ,

∙ 对Tr*n*(*s*1)的任意子树Tr*n*−1(*s*′ )，都存在Tr*n*(*s*2)的子树Tr*n*−1(*s*′ )，使得Tr*n*−1(*s*′ ) ↔*V*

1 2 1

Tr*n*−1(*s*′ )，且

2

∙ 对任意Tr*n*(*s*2)的子树Tr*n*−1(*s*′ )，都存在Tr*n*(*s*1)的子树Tr*n*−1(*s*′ )，使得Tr*n*−1(*s*′ ) ↔*V*

2 1 1

Tr*n*−1(*s*′ )；

2

则 称*M*1的 计 算 树Tr*n*(*s*1)和*M*2的 计 算 树Tr*n*(*s*2)是*V* -互 模 拟 的（记 为(*M*1*,* Tr*n*(*s*1)) ↔*V*

(*M*2*,* Tr*n*(*s*2))，简写为Tr*n*(*s*1) ↔*V* Tr*n*(*s*2)）

特别地，当*n* = 0时，只需考虑第一个条件即可。

命题 **5.1.** 给定原子命题集*V* ⊆ *A* 和结构(*Mi, si*)（*i* = 1*,* 2）。 那么：

(*s*1*, s*2) ∈ *Bn*当且仅当对任意0 ≤ *j* ≤ *n*，有*Trj* (*s*1) ↔*V Trj* (*s*2)。

证明**.** 下面从两个方面来证明这一结论。

(⇒) 下证“如果(*s*1*, s*2) ∈ *Bn*，则对于任意0 ≤ *j* ≤ *n*，有*Trj* (*s*1) ↔*V Trj* (*s*2)”成立。 (*s*1*, s*2) ∈ *Bn*表明*Trn*(*s*1)和*Trn*(*s*2)的根有同样的标签（除了*V* 里的元素之外）。 此外，对 任意(*s*1*, s*1*,*1) ∈ *R*1，存在一个(*s*2*, s*2*,*1) ∈ *R*2，使得(*s*1*,*1*, s*2*,*1) ∈ *Bn*−1；且对任意(*s*2*, s*2*,*1) ∈ *R*2， 存在一个(*s*1*, s*1*,*1) ∈ *R*1，使得(*s*1*,*1*, s*2*,*1) ∈ *Bn*−1。 因此，由定义可知*Tr*1(*s*1) ↔*V Tr*1(*s*2)。 递归地使用上述方法可得对任意0 ≤ *j* ≤ *n*，都有*Trj* (*s*1) ↔*V Trj* (*s*2)。

(⇐)下证“如果对于任意0 ≤ *j* ≤ *n*有*Trj* (*s*1) ↔*V Trj* (*s*2)，则*Trj* (*s*1) ↔*V Trj* (*s*2)”成 立。 由*Tr*0(*s*1) ↔*V Tr*0(*s*2)可知*L*(*s*1) − *V* = *L*′(*s*2) − *V* ，因而(*s*1*, s*2) ∈ *B*0。 由*Tr*1(*s*1) ↔*V Tr*1(*s*2)可知*L*(*s*1) − *V* = *L*′(*s*2) − *V* ，且对于一棵树根的任意后继状态*s*，都能找到另一棵 树根的一个后继状态*s*′，使得(*s, s*′) ∈ *B*0。 因此，(*s*1*, s*2) ∈ *B*1。 同理可证(*s*1*, s*2) ∈ *B*2,

. . . , (*s*1*, s*2) ∈ *Bn*。

命题[5.1](#_bookmark98)表明如果任意两个初始结构中的两个状态*s*1和*s*2能够在*V* 上相互模拟对方 直到*n*步，当且仅当分别以*s*1和*s*2为根的计算树能在*V* 上相互模拟直到深度为*n*。 由此可 知，如果同一初始结构的两个状态*s*和*s*′不是*V* -互模拟的，则存在一个数*k* ∈ N，使得分 别以*s*和*s*′为根的计算树Tr*k*(*s*)和Tr*k*(*s*′)不是*V* -互模拟的。

命题 **5.2.** 给定原子命题集*V* ⊆ *A* 、 初始*Kripke*结构*M* 和两个状态*s, s*′ ∈ *S*。 若*s* ̸↔*V s*′，

则存在一个最小整数*k*，使得*Trk*(*s*)和*Trk*(*s*′)不是*V -*互模拟的。

证明**.** 若*s* ̸↔*V s*′，则存在一个最小的数*c*，使得(*si, s j* ) ∈*/ Bc*。 因此，由命题[5.1](#_bookmark98)可知，存 在一个最小整数*m*（*m* ≤ *c*），使得Tr*m*(*si*)和Tr*m*(*s j* )不是*V* -互模拟的。 令*k* = *m*可得上述 结论。

**5.1.1.2** 计算树的特征公式

由上面小节的讨论可知，*V* -互模拟可以将计算树区别开[3](#_bookmark0)。 下面讨论如何使 用CTL公式描述计算树，且表明具有（或没有）*V* -互模拟关系之间的计算树的特征公 式之间的关系。 为此，首先给出计算树特征公式的定义。

3相似的方法在Mycielski等人的文章中已被使用[[121](#_bookmark235)]， 在这篇文章中一元公式的结构通过等价类≡*k* 被描述

为Hintikka公式[[122](#_bookmark236)]。 另一个类似的工作是Yankov-Fine构造 [[123](#_bookmark237)]。

定义 **5.2.** 给定原子命题集*V* ⊆ *A* 、 初始*Kripke*结构*M* = (*S, R, L, s*0)和状态*s* ∈ *S*。 定义

在*V* 上的计算树*Trn*(*s*)的特征公式（记为*FV* (*Trn*(*s*))，*n* ≥ 0）递归定义如下：

*FV* (*Tr*0(*s*)) = 八

*p*∈*V* ∩*L*(*s*)

*p* ∧ 八

*q*∈*V* −*L*(*s*)

¬*q,*

／ ／ ＼

*FV* (*Trk*+1(*s*)) = 八

(*s,s*′)∈*R*

EX*FV* (*Trk*(*s*′)) ∧ AX

(*s,s*′)∈*R*

*FV* (*Trk*(*s*′))

∧ *FV* (*Tr*0(*s*)) （*k* ≥ 0）。

由定义[5.2](#_bookmark100)可知，计算树的特征公式从三个方面展示了计算树的信息：

(1) 只考虑*V* 中的原子命题；

(2) 突出了树节点的内容，即：对于任意原子命题*p* ∈ *V* ，若*p*在节点的标签中，则其

正出现在特征公式中，否则负出现在特征公式中；

(3) 公式中的时序算子表示了状态之间的转换关系。

通俗一点， *FV* (Tr0(*s*))表明了节点*s*的在*V* 上的内容； EX的合取部分和AX部分保证，

以*s*的每个直接后继状态*s*′为根、深度为*k*的计算树都有一个CTL公式来描述。 下面的结论表明，若两个计算树是*V* -互模拟的，则它们在*V* 上的特征公式是逻辑

等价的。

引理 **5.2.** 给定原子命题集*V* ⊆ *A* 、初始*Kripke*结构*M* = (*S, R, L, s*0)和*M* ′ = (*S*′*, R*′*, L*′*, s*′ )、

0

*s* ∈ *S*、*s*′ ∈ *S*′且*n* ≥ 0。 若*Trn*(*s*) ↔*V Trn*(*s*′)，则*FV* (*Trn*(*s*)) ≡ *FV* (*Trn*(*s*′))。

证明**.** 通过归纳计算树的深度*n*来证明。

基始（*n* = 0）： 对任意*sx* ∈ *S*和*s*′ ∈ *S*′，若Tr0(*sx*) ↔

Tr0(*s*′ )，则由*L*(*sx*) −*V* = *L*′(*s*′ ) −

*x V x* *x*

*V* 可知*FV* (Tr0(*sx*)) ≡ *FV* (Tr0(*s*′ ))。

*x*

归纳步（*n >* 0）： 假设对任意0 ≤ *m* ≤ *n*，若Tr*m*(*s*) ↔*V* Tr*m*(*s*′)，则*FV* (Tr*m*(*s*)) ≡

*FV* (Tr*m*(*s*′))。 下证若Tr*n*+1(*s*) ↔*V* Tr*n*+1(*s*′)，则*FV* (Tr*n*+1(*s*)) ≡ *FV* (Tr*n*+1(*s*′))。

由归纳假设可知，对任意*k* = *m*、 *sk* ∈ *S*和*s*′ ∈ *S*′，若Tr*n*−*k*(*sk*) ↔

Tr*n*−*k*(*s*′ )，则*FV* (

*k V k*

Tr*n*−*k*(*sk*)) ≡ *FV* (Tr*n*−*k*(*s*′ ))。 因 此， 要 证 原 结 论 成 立， 只 需 要 证 明： 若Tr*n*−*k*+1(*sk*−1)

*k*

↔*V* Tr*n*−*k*+1(*s*′ −1)，则*FV* (Tr*n*−*k*+1(*sk*−1)) ≡ *FV* (Tr*n*−*k*+1(*s*′

))。其中，(*sk*−1*, sk*) ∈ *R*且(*s*′ *,*

*k*

*k*) ∈ *R* 。 显然，由计算树的特征公式可知：

*k*−1

*k*−1

*s*′ ′

*FV* (Tr*n*−*k*+1(*sk*−1)) =

／

八

(*sk*−1*,sk* )∈*R*

＼

EX*FV* (Tr*n*−*k*(*sk*)) ∧

／

AX ／

*FV* (Tr*n*

＼

*k*(*s* )) ∧ *F*

(Tr (*s* ))

− *k*

(*sk*−1*,sk* )∈*R*

*V* 0 *k*−1

且

*FV* (Tr*n*−*k*+1(*s*′

*k*−1



)) = 

(*s*′

八

*,s*′ )∈*R*



EX*FV* (Tr*n*−*k*(*s*′ ))∧

*k*

*k*−1 *k*

 

AX  ／

*FV* (Tr*n*

*k*(*s*′ )) ∧ *F*

(Tr (*s*′

))*.*

*s*′

(

*k*−1

− *k*

*,s*′ )∈*R*

*k*

*V* 0 *k*−1

又因为Tr*n*−*k*+1(*sk*−1) ↔*V* Tr*n*−*k*+1(*s*′ −1)，所以，对任意(*sk*−1*, sk*) ∈ *R*，存在(*s*′

*, s*′ ) ∈

*k k*−1 *k*

*R*′，使得Tr*n*−*k*(*sk*) ↔*V* Tr*n*−*k*(*s*′ )，且对任意(*s*′

*, s*′ ) ∈ *R*′，存在(*sk*−1*, sk*) ∈ *R*，使得Tr*n*−*k*(*sk*) ↔

*k k*−1 *k* *V*

Tr*n*−*k*(*s*′ )。 因此，由归纳假设可知*FV* (Tr*n*−*k*+1(*sk*−1)) ≡ *FV* (Tr*n*−*k*+1(*s*′

))。

*k k*−1

此外，对于初始Kripke结构*M* 上的状态*s*和*s*′，若(*M , s*)是定义在*V* 上、 根为*s*′、 深 度为*n*的计算树特征公式的模型，则*s*和*s*′至少属于*Bn*，即：*s*和*s*′能想互模拟至少到 第*n*层深度。

引理 **5.3.** 令*V* ⊆ *A* 、*M* = (*S, R, L, s*0)、*M* ′ = (*S*′*, R*′*, L*′*, s*′ )、*s* ∈ *S*、*s*′ ∈ *S*′且*n* ≥ 0，则：

0

1. (*M , s*) |= *FV* (*Trn*(*s*))；
2. 若(*M , s*) |= *FV* (*Trn*(*s*′))，则*Trn*(*s*) ↔*V Trn*(*s*′)。

证明**.** (i) 基始（*n* = 0）：从树的特征公式定义可知*L*(*s*) |= *FV* (Tr0(*s*))，所以，(*M , s*) |=

*FV* (Tr*n*(*s*))。

归纳步（*n >* 0）： 假设对任意*k* ≥ 0，(*M , s*) |= *FV* (Tr*k*(*s*))， 下面证明(*M , s*) |=

*FV* (Tr*k*+1(*s*))，即：

(*M , s*) |=

／

八

(*s,s*′)∈*R*

＼

EX*T* (*s*′)

∧ AX

／

／

(*s,s*′)∈*R*

＼

*T* (*s*′)

∧ *FV* (Tr0(*s*))*.*

其中*T* (*s*′) = *FV* (Tr*k*(*s*′))。 由基始可知(*M , s*) |= *FV* (Tr0(*s*))。 由归纳假设可知， 对任 意(*s, s*′) ∈ *R*，有(*M , s*′) |= *FV* (Tr*k*(*s*′))。 因此，(*M , s*) |= EX*FV* (Tr*k*(*s*′)，从而(*M , s*) |= (*s,s*′)∈*R* EX*FV* (Tr*k*(*s*′))。

八

同理，对任意(*s, s*′) ∈ *R*，都有(*M , s*′) |= V ′

*FV* (Tr*k*(*s*′))。 因此，

(*s,s* )∈*R*

(*M , s*) |= AX

／

／

(*s,s*′)∈*R*

＼

*FV* (Tr*k*(*s*′))

所以，对任意*n* ≥ 0，(*M , s*) |= *FV* (Tr*n*(*s*))。

(ii) 基始（*n* = 0）：若(*M , s*) |= *FV* (Tr0(*s*′))，则*L*(*s*)−*V* = *L*′(*s*′)−*V*。因此，Tr0(*s*) ↔*V*

Tr0(*s*′)。

归纳步（*n >* 0）： 假定若(*M , s*) |= *FV* (Tr*n*−1(*s*′))，则Tr*n*−1(*s*) ↔*V* Tr*n*−1(*s*′)。 下证

若(*M , s*) |= *FV* (Tr*n*(*s*′))，则Tr*n*(*s*) ↔*V* Tr*n*(*s*′)。

(a) 由基始知*L*(*s*) − *V* = *L*′(*s*′) − *V* ；

(b) 因为(*M , s*) |= *FV* (Tr*n*(*s*′))， 所以，(*M , s*) |= AX (V

1

(*s*′*,s*′ )∈*R*

1

*FV* (Tr*n*−

1(*s*′ ))飞。 所以，

对于任意(*s, s*1) ∈ *R*，存在(*s*′*, s*′ ) ∈ *R*′，使得(*M , s*1) |= *FV* (Tr*n*−1(*s*′ ))。 由归纳假

1 1

设可知Tr*n*−1(*s*1) ↔*V* Tr*n*−1(*s*′ )。 即：∀(*s, s*1) ∈ *R*，∃(*s*′*, s*′ ) ∈ *R*′，使得Tr*n*−1(*s*1) ↔

1 1 *V*

Tr*n*−1(*s*′ )。

1

(c) 因为(*M , s*) |= *FV* (Tr*n*(*s*′))，所以，(*M , s*) |= 八 ′ ′

′ EX*FV* (Tr*n*−1(*s*′ ))。 对任意(*s*′*,*

(*s ,s*1)∈*R* 1

1) ∈ *R* ，存在(*s, s*1) ∈ *R*，使得(*M , s*1) |= *FV* (Tr*n*−1(*s*1)。 由归纳假设可知Tr*n*−1(*s*1)

*s*′ ′ ′

↔*V* Tr*n*−1(*s*′ )，即：∀(*s*′*, s*′ ) ∈ *R*′，∃(*s, s*1) ∈ *R*，使得Tr*n*−1(*s*1) ↔

Tr*n*−1(*s*′ )。

1 1 *V* 1

**5.1.1.3** 初始结构的特征公式

由*V* -互模拟的定义和命题[5.2](#_bookmark99)可以自然地得到一个*V* -互模拟的补概念——*V* -可区 分。特别地，在命题[5.2](#_bookmark99)中，若初始Kripke结构*M* 的两个状态*s*和*s*′不是*V* -互模拟的（即： *s* ̸↔*V s*′），则称*s*和*s*′是*V -*可区分的。 用dis*V* (*M , s, s*′*, k*)表示状态*s*和*s*′在命题[5.2](#_bookmark99)中所说的 最小数*k*下是*V* -可区分的。

此 外， 对 于 给 定 的 初 始Kripke结 构*M* 和 原 子 命 题 集*V* ， 若 在*M* 中 存 在 两 个 状 态*s*和*s*′是*V* -可区分的，则称*M* 是*V* -可区分的。 而对于一个*V* -可区分的初始Kripke结 构*M* ，存在一个最小数*k*，使得对于该结构上的任意两个状态*s*和*s*′，若*s*和*s*′是可区分 的，则(*s, s*′) ̸∈ *Bk*。 本文称这样的数为*M* 关于*V* 的特征数，记为*ch*(*M ,V* )，其定义如 下：

*ch*(*M ,V* ) =

( max{*k* | *s, s*′ ∈ *S* 且dis*V* (*M , s, s*′*, k*)}*, M* 是*V* -可区分的；

min{*k* | *Bk* = *Bk*+1*, k* ≥ 0}*,* 否则。

由*ch*(*M ,V* )定义可知，对于任意*M* 和*V* ，*ch*(*M ,V* )总是存在的，这体现在两个方

面：

(1) 若*M* 是*V* -可区分的，则存在两个状态*s*和*s*′是*V* -可区分的，由命题[5.2](#_bookmark99)可知，存在

一个数*k*，使得dis*V* (*M , s, s*′*, k*)成立；

(2) 若对于任意*k* ≥ 0和*M* 上的两个状态*s*和*s*′，都有(*s, s*′) ∈ *Bk*且*Bk* = *Bk*+1，则*ch*(*M ,V* ) =

0。

直观上，特征数*c* = *ch*(*M ,V* )将*M* 上的状态分为两大类：第一类中的任意两个状

态*s*和*s*′是*V* -可区分的，且(*s, s*′) ̸∈ *Bc*；第二类中的状态都是*V* -不可区分的。

引理 **5.4.** 令*V* ⊆ *A* 、*M* = (*S, R, L, s*0)、*k* = *ch*(*M ,V* )且*s* ∈ *S*，则

*(i)* (*M , s*) |= *FV* (*Trk*(*s*))；

*(ii)* 对任意*s*′ ∈ *S*，(*M , s*) ↔*V* (*M , s*′)当且仅当(*M , s*′) |= *FV* (*Trk*(*s*))。

证明**.** (i) 这由引理[5.3](#_bookmark102)易知。

(ii) 令*φ* = *FV* (Tr*k*(*s*))（*k*为*M* 关于*V* 的特征数）。 由(i)可知(*M , s*) |= *φ* ，从而对任

意*s*′ ∈ *S*，若*s* ↔*V s*′，由定理[3.1](#_bookmark52)和IR(*φ, A* − *V* )可知(*M , s*′) |= *φ*。

假定(*M , s*′) |= *φ*。 若*s* 的*V s*′，则Tr*k*(*s*) ̸↔*V* Tr*k*(*s*′)，因而由引理[5.3](#_bookmark102)可知(*M , s*′) |= *φ* ，

这与假定矛盾。

由此，可如下定义初始结构的特征公式。

定义 **5.3** (特征公式)**.** 给定原子命题集*V* ⊆ *A* 和初始结构*K* = (*M , s*0)，其中*c* = *ch*(*M ,V* )。

对任意*M* 上的状态*s*′ ∈ *S*，记*T* (*s*′) = *FV* (*Trc*(*s*′))。 则*K* 关于*V* 的特征公式 *FV* (*K* )为：

*T* (*s*0) ∧ 八 AG

*s*∈*S*

／

*T* (*s*) → 八

(*s,s*′)∈*R*

EX*T* (*s*′) ∧ AX

／ ／

(*s,s*′)∈*R*

＼＼

*T* (*s*′) *.*

有时为了凸显出初始Kripke结构及其初始状态，也把特征公式写为*FV* (*M , s*0)。 显 然，IR(*FV* (*M , s*0)*,V* )。此外，在特征公式的定义中，使用了深度为*c*（即：特征数）的 计算树的特征公式：意在表明对任意*M* 上的两个状态*s*和*s*′，*s*和*s*′是*V* -可区分的当且仅 当*FV* (Tr*c*(*s*)) ̸≡ *FV* (Tr*c*(*s*′))。 特别地，*T* (*s*)确保了状态*s* ∈ *S*被CTL公式描述，而其余部 分表明了结构*M* 上状态之间的转换关系。 下面的例子给出了计算特征公式的一般步 骤。

例 **5.1** (例[3.1](#_bookmark49)的延续)**.** 考虑图[5.1](#_bookmark104)中左边的初始结构*K*2 = (*M , s*0)（其最初出现在图[3.1](#_bookmark50)中）。 左边的为*M* 上的四棵计算树：从左到右表示以*s*0为根、 深度分别为0、 1、 2和3的计算 树（为简化图， 计算树的标签没有给出， 但是每个树节点的标签可从*K*2找到）。 令*V* = {*d*}，则*V* = {*s, se*}。

因为*L*(*s*1) − *V* = *L*(*s*2) − *V* ，所以有Tr0(*s*1) ↔*V* Tr0(*s*2)。 由于存在(*s*1*, s*2) ∈ *R*，使得 对任意(*s*2*, s*′) ∈ *R*， 都有*L*(*s*2) − *V* ̸= *L*(*s*′) − *V* ， 所以， Tr1(*s*1) ̸↔*V* Tr1(*s*2)。 由此可知， *s*1和*s*2是*V* -可区分的，且dis*V* (*M , s*1*, s*2*,* 1)。

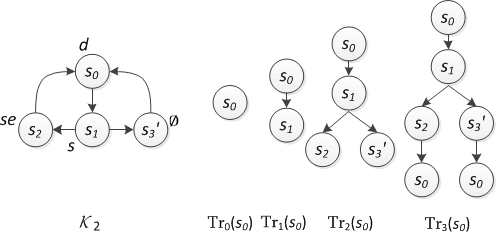


图 5.1: 初始结构*K*2（源于图[3.1](#_bookmark50)）及其计算树示意图

同理可得：dis*V* (*M , s*0*, s*1*,* 0)、dis*V* (*M , s*1*, s*′ *,* 1)、dis*V* (*M , s*0*, s*2*,* 0)和dis*V* (*M , s*0*, s*′ *,* 0)。

3 3

此外，*s*2 ↔*V s*′ 。 因此，可以计算*M* 关于*V* 的特征数为：

3

*ch*(*M ,V* ) = max{*k* | *s, s*′ ∈ *S* 且dis*V* (*M , s, s*′*, k*)} = 1*.*

所以，可以由以下步骤计算*K*2关于*V* 的特征公式：

*FV* (Tr0(*s*0)) = *d, FV* (Tr0(*s*1)) = ¬*d,*

*FV* (Tr0(*s*2)) = ¬*d, FV* (Tr0(*s*′ )) = ¬*d,*

3

*FV* (Tr1(*s*0)) = EX¬*d* ∧ AX¬*d* ∧ *d* ≡ AX¬*d* ∧ *d,*

*FV* (Tr1(*s*1)) = EX¬*d* ∧ EX¬*d* ∧ AX(¬*d* ∨ ¬*d*) ∧ ¬*d* ≡ AX¬*d* ∧ ¬*d,*

*FV* (Tr1(*s*2)) = EX*d* ∧ AX*d* ∧ ¬*d* ≡ AX*d* ∧ ¬*d,*

*FV* (Tr1(*s*′ )) ≡ *FV* (Tr1(*s*2))*,*

3

*FV* (*M , s*0) ≡ AX¬*d* ∧ *d*∧

AG(AX¬*d* ∧ *d* → AX(AX¬*d* ∧ ¬*d*))∧ AG(AX¬*d* ∧ ¬*d* → AX(AX*d* ∧ ¬*d*))∧ AG(AX*d* ∧ ¬*d* → AX(AX¬*d* ∧ *d*))*.*

下面的定理表示，特征公式确实描述了一个初始结构。 此时，对系统结构的操作 就可转换为对其特征公式的操作，如：下文中的给定系统下的最弱充分条件计算。 直 观上，特征公式保持了给定初始结构在原子命题集*V* 上的所有特性，即：具有*V* -互模 拟的两个初始结构关于*V* 的特征公式逻辑等价。

定理 **5.1.** 令*V* ⊆ *A* 、*M* = (*S, R, L, s*0)且*M* ′ = (*S*′*, R*′*, L*′*, s*′ )，则：

0

*(i)* (*M* ′*, s*′ ) |= *FV* (*M , s*0) 当且仅当(*M , s*0) ↔

(*M* ′*, s*′ )；

0 *V* 0

*(ii)* 若*s*0 ↔*V s*′ 则*FV* (*M , s*0) ≡ *FV* (*M* ′*, s*′ )。

0 0

证明**.** (i) 令*FV* (*M , s*0)为(*M , s*0)关于*V* 的特征公式。显然，IR(*FV* (*M , s*0)*,V* )。为了证明

上述结论成立，首先证明(*M , s*0) |= *FV* (*M , s*0)。

令*c* = *ch*(*M ,V* )，由引理[5.3](#_bookmark102)可知(*M , s*0) |= *FV* (Tr*c*(*s*0))。 下证特征公式里的另一部

分，即：(*M , s*0) |= 八

*s*∈*S*

AG *G*(*M , s*)，其中

／ ＼ ／ ＼

*G*(*M , s*) = *FV* (Tr*c*(*s*)) →

八

(*s,s*1)∈*R*

EX*FV* (Tr*c*(*s*1))

∧ AX

／

(*s,s*1)∈*R*

*FV* (Tr*c*(*s*1)) *.*

为此，下面证明(*M , s*0) |= AG *G*(*M , s*)。 考虑下面两种情况：

∙ 若(*M , s*0) |= *FV* (Tr*c*(*s*))，显然(*M , s*0) |= *G*(*M , s*)；

∙ 若(*M , s*0) |= *FV* (Tr*c*(*s*))：

(*M , s*0) |= *FV* (Tr*c*(*s*))

⇒ *s*0 ↔*V s* （引理[5.4](#_bookmark103)）

∀(*s, s*1) ∈ *R*：

(*M , s*1) |= *FV* (Tr*c*(*s*1)) （*s*1 ↔*V s*1）

八

⇒ (*M , s*) |=

(*s,s*1)∈*R*

⇒ (*M , s*0) |= 八

EX*FV* (Tr*c*(*s*1))

EX*FV* (Tr*c*(*s*1)) （IR(八

EX*FV* (Tr*c*(*s*1))*,V* ), *s*0 ↔ *s*）

∀(*s, s*1) ∈ *R*：

(*M , s*1) |= V

(*s,s*1)∈*R*

*FV* (Tr*c*(*s*2))

(*s,s*1)∈*R* *V*

(*s,s*2)∈*R*

⇒ (*M , s*) |= AX (V

(*s,s*2)∈*R*

⇒ (*M , s*0) |= AX (V

*FV* (Tr*c*(*s*2))）

*FV* (Tr*c*(*s*2))） （IR(AX (V

*FV* (Tr*c*(*s*2))） *,V* ), *s*0 ↔ *s*）

(*s,s*2)∈*R*

⇒ (*M , s*0) |= *G*(*M , s*)。

(*s,s*2)∈*R* *V*

对任意其它能从*s*0可达的状态*s*′，都可以类似地证明(*M , s*′) |= *G*(*M , s*)。 因此，对

任意*s* ∈ *S*，(*M , s*0) |= AG *G*(*M , s*)，从而(*M , s*0) |= *FV* (*M , s*0)。

下面从两个方面证明(i)成立：

(⇐) 证明：若*s*0 ↔*V s*′ ，则(*M* ′*, s*′ ) |= *FV* (*M , s*0)。因为(*M , s*0) |= *FV* ( *M , s*0)且IR(*FV*

0 0

(*M , s*0)*,V* )，由定理[3.1](#_bookmark52)可知(*M* ′*, s*′ ) |= *FV* (*M , s*0)。

0

(⇒) 证明：若(*M* ′*, s*′ ) |= *FV* (*M , s*0)，则*s*0 ↔

*s*′ 。 为此，下面证明对任意*n* ≥ 0，

0 *V* 0

*Trn*(*s*0) ↔*V Trn*(*s*′ )。

0

基始（*n* = 0）： 由特征公式的定义，显然*Tr*0(*s*0) ≡ *Tr*0(*s*′ )成立。

0

归纳步骤（*n >* 0）： 假定对任意*k >* 0，都有Tr*k*(*s*0) ↔*V* Tr*k*(*s*′ )，下面证明Tr*k*+1(*s*0)

0

↔*V* Tr*k*+1(*s*′ )。令(*s*0*, s*1)*,* (*s*1*, s*2), *. . .* , (*sk*−1*, sk*) ∈ *R*且(*s*′ *, s*′ )*,* (*s*′ *, s*′ )*, . . . ,* (*s*′

*, s*′ ) ∈ *R*′，即对

0 0 1 1 2

*k*−1 *k*

于任意0 ≤ *i* ≤ *k* − 1，*si*+1（*s*′

*i*+1

）是*si*（*s*′）的直接后继状态。 由归纳假设可知，只需证

明Tr1(*sk*) ↔*V* Tr1(*s*′ )。

*i*

*k*

(a) 由归纳假设可知*L*(*sk*) − *V* = *L*′(*s*′ ) − *V*。

*k*

在讨论其它点时，首先考虑下面事实（**fact**）：

(*M* ′*, s*′ ) |= *FV* (*M , s*0)

0

⇒ ∀*s*′ ∈ *S*′，∀*s* ∈ *S*，

(*M* ′*, s*′) |= *FV* (Tr*c*(*s*)) → (八

(*s,s*1)∈*R*

(I) (*M* ′*, s*′ ) |= *FV* (Tr*c*(*s*0)) → (八

EX*FV* (Tr*c*(*s*1))） ∧ AX (V

EX*FV* (Tr*c*(*s*1))） ∧ AX (V

(*s,s*1)∈*R*

*FV* (Tr*c*(*s*1))）

*FV* (Tr*c*(*s*1))）

0 (*s*0*,s*1)∈*R*

(*s*0*,s*1)∈*R*

(II) (*M* ′*, s*′ ) |= *FV* (Tr*c*(*s*0))) (已知)

0

(III) (*M* ′*, s*′ ) |= (八

EX*FV* (Tr*c*(*s*1))） ∧ AX (V

*FV* (Tr*c*(*s*1))） ((I),(II))

0 (*s*0*,s*1)∈*R*

(*s*0*,s*1)∈*R*

(b) 下证∀(*sk, sk*+1) ∈ *R*，存在(*s*′ *, s*′

) ∈ *R*′，使得*L*(*sk*+1) − *V* = *L*′(*s*′

) − *V*。

*k k*+1

*k*+1

(1) (*M* ′*, s*′ ) |= 八

EX*FV* (Tr*c*(*s*1)) (III)

0 (*s*0*,s*1)∈*R*

(2) ∀(*s*0*, s*1) ∈ *R*，∃(*s*′ *, s*′ ) ∈ *R*′，使得(*M* ′*, s*′ ) |= *FV* (Tr*c*(*s*1)) (1)

0 1 1

(3) Tr*c*(*s*1) ↔*V* Tr*c*(*s*′ ) ((2), 引理[5.3](#_bookmark102))

1

(4) *L*(*s*1) − *V* = *L*′(*s*′ ) − *V* ((3), *c* ≥ 0)

1

(5) (*M* ′*, s*′ ) |= *FV* (Tr*c*(*s*1)) →

1

(八

(*s*1*,s*2)∈*R*

EX*FV* (Tr*c*(*s*2))）

∧ AX

(V

(*s*1*,s*2)∈*R*

*FV* (Tr*c*(*s*2))）

## (fact)

(6) (*M* ′*, s*′ ) |= (八

EX*FV* (Tr*c*(*s*2))） ∧ AX (V

*FV* (Tr*c*(*s*2))） ((2), (5))

1

(7) (*M* ′*, s*′ ) |= (八

(*s*1*,s*2)∈*R*

(*s*1*,s*2)∈*R*

EX*FV* (Tr*c*(*sk*+1))） ∧ AX (V

*FV* (Tr*c*(*sk*+1))） (与(6)类似)

*k* (*sk,sk*+1)∈*R*

(*sk,sk*+1)∈*R*

(8) ∀(*sk, sk*+1) ∈ *R*，∃(*s*′ *, s*′

) ∈ *R*′，使得(*M* ′*, s*′

) |= *FV* (Tr*c*(*sk*+1)) (7)

*k k*+1

*k*+1

(9) Tr*c*(*sk*+1) ↔*V* Tr*c*(*s*′

*k*+1

(10) *L*(*sk*+1) − *V* = *L*′(*s*′

) ((8), 引理[5.3](#_bookmark102))

) − *V* ((9), *c* ≥ 0)

*k*+1

(c) 下证∀(*s*′ *, s*′

) ∈ *R*′，存在(*sk, sk*+1) ∈ *R*，使得*L*(*sk*+1) − *V* = *L*′(*s*′

) − *V*。

*k k*+1

*k*+1

(1) (*M* ′*, s*′ ) |= AX (V

*FV* (Tr*c*(*sk*+1))） (上面的(7))

*k*

(2) ∀(*s*′ *, s*′

(*sk,sk*+1)∈*R*

) ∈ *R*′，∃(*sk, sk*+1) ∈ *R*，使得(*M* ′*, s*′

) |= *FV* (Tr*c*(*s*′

)) (1)

*k k*+1

*k*+1

*k*+1

(3) Tr*c*(*sk*+1) ↔*V* Tr*c*(*s*′

*k*+1

) ((2), 引理[5.3](#_bookmark102))

(4) *L*(*sk*+1) − *V* = *L*′(*s*′

*k*+1

) − *V* ((3), *c* ≥ 0)

(ii) 由引理[5.2](#_bookmark101)和[5.4](#_bookmark103)易得。

上文描述了初始结构特征公式的计算，下面给出例[4.2](#_bookmark87)的详细计算过程。

例 **5.2** (例 [4.2](#_bookmark87)的延续)**.** 令*A* = {*d, se, sp, s*}和*V* = {*d, se*}，求*s*在*V* 和初始结构*K* = (*M , s*0)上

的WSC，其中*M* 为例 [1.1](#_bookmark8)中初始状态为*s*0的汽车制造企业模型结构。

由定理 [4.1](#_bookmark86)可知，*s*在*V* 和初始结构*K* = (*M , s*0)上的WSC为¬FCTL(*FA* (*K* )∧¬*s,* {*s*}∪

{*sp*})。

FCTL(*FA* (*K* ) ∧ ¬*s,* {*s*} ∪ {*sp*})

≡ FCTL(FCTL(*FA* (*K* ) ∧ ¬*s,* {*sp*})*,* {*s*})

≡ FCTL(FCTL(*FA* (*K* )*,* {*sp*}) ∧ ¬*s,* {*s*})*.*

*K*1 |= FCTL(*FA* (*K* )*,* {*sp*})

⇔ 存在*K* ′ ↔{*sp*} *K*1，使得*K* ′ |= *FA* (*K* )和

1 1

⇔ 存在*K* ′ ↔{*sp*} *K*1，使得*K* ′ ↔0/ *K*

1 1

⇔ 存在*K* ′ ↔{*sp*} *K* 和*K* ′ |= *FV* ∪{*s*}(*K* ) (定理 [5.1](#_bookmark105))

1 1

⇔ *K*1 |= *FV* ∪{*s*}(*K* )。

由此可知，FCTL(*FA* (*K* )*,* {*sp*}) ≡ *FV* ∪{*s*}(*K* )。所以要计算¬FCTL (*FA* (*K* )∧¬*s,* {*s*}∪

{*sp*})，只需计算¬FCTL(*FV* ∪{*s*}(*K* ) ∧¬*s,* {*s*})。令*V* ′ = *V* ∪{*s*} = {*d, se, s*}，则*V* ′ = {*sp*}。

易知，除了*s*2和*s*4是*V* ′-互模拟的以外，其余都是*V* ′-可区分的，且dis*V* ′ (*M , si, s j,* 0)，

其中*i* ̸= *j*且*i*和 *j*不能同时分别取值2和4。 因此可以计算*M* 关于*V* ′的特征数为： *ch*(*M ,V* ′) = max{*k* | *s, s*′ ∈ *S* 且dis*V* ′ (*M , s, s*′*, k*)} = 0。

所以，可以由以下步骤计算*K*2关于*V* 的特征公式：

*FV* ′ (Tr0(*s*0)) = *d* ∧ ¬*se* ∧ ¬*s, FV* ′ (Tr0(*s*1)) = ¬*d* ∧ ¬*se* ∧ *s, FV* ′ (Tr0(*s*2)) = *FV* ′ (Tr0(*s*4)) = ¬*d* ∧ *se* ∧ ¬*s, FV* ′ (Tr0(*s*3) = ¬*d* ∧ ¬*se* ∧ ¬*s, FV* ′ (*M , s*0) ≡ *d* ∧ ¬*se* ∧ ¬*s*∧

AG(*d* ∧ ¬*se* ∧ ¬*s* → AX(¬*d* ∧ ¬*se* ∧ *s*))∧

AG(¬*d* ∧ ¬*se* ∧ *s* → [AX((¬*d* ∧ *se* ∧ ¬*s*) ∨ (¬*d* ∧ ¬*se* ∧ ¬*s*))∧

EX(¬*d* ∧ *se* ∧ ¬*s*) ∧ EX(¬*d* ∧ ¬*se* ∧ ¬*s*)])∧ AG(¬*d* ∧ ¬*se* ∧ ¬*s* → AX(*d* ∧ ¬*se* ∧ ¬*s*))∧ AG(¬*d* ∧ *se* ∧ ¬*s* → AX(*d* ∧ ¬*se* ∧ ¬*s*))*.*

令：

*ϕ*1 = *d* ∧ ¬*se* ∧ ¬*s* → AX(¬*d* ∧ ¬*se* ∧ *s*) ≡ ¬*d* ∨ *se* ∨ *s* ∨ AX(¬*d* ∧ ¬*se* ∧ *s*)*,*

*ϕ*2 = ¬*d* ∧ ¬*se* ∧ *s* → AX(¬*d* ∧ *se* ∧ ¬*s*) ≡ *d* ∨ *se* ∨ ¬*s*∨

[AX((¬*d* ∧ *se* ∧ ¬*s*) ∨ (¬*d* ∧ ¬*se* ∧ ¬*s*)) ∧ EX(¬*d* ∧ *se* ∧ ¬*s*) ∧ EX(¬*d* ∧ ¬*se* ∧ ¬*s*)]*,*

*ϕ*3 = ¬*d* ∧ ¬*se* ∧ ¬*s* → AX(*d* ∧ ¬*se* ∧ ¬*s*) ≡ *d* ∨ *se* ∨ *s* ∨ AX(*d* ∧ ¬*se* ∧ ¬*s*)*,*

*ϕ*4 = ¬*d* ∧ *se* ∧ ¬*s* → AX(*d* ∧ ¬*se* ∧ ¬*s*) ≡ *d* ∨ ¬*se* ∨ *s* ∨ AX(*d* ∧ ¬*se* ∧ ¬*s*)*.*

此外，令

*ψ*1 = ¬*d* ∨ *se* ∨ *s ψ*2 = *d* ∨ *se* ∨ ¬*s*

*ψ*3 = *d* ∨ *se* ∨ *s ψ*4 = *d* ∨ ¬*se* ∨ *s*

*α*1 = AX(¬*d* ∧ ¬*se* ∧ *s*) *α*2 = AX((¬*d* ∧ *se* ∧ ¬*s*) ∨ (¬*d* ∧ ¬*se* ∧ ¬*s*))

*α*3 = EX(¬*d* ∧ *se* ∧ ¬*s*) ∧ EX(¬*d* ∧ ¬*se* ∧ ¬*s*) *α*4 = AX(*d* ∧ ¬*se* ∧ ¬*s*)

*α*5 = AX(*d* ∧ ¬*se* ∧ ¬*s*) *ψ* = *d* ∧ ¬*se* ∧ ¬*s.*

则，

*ψ* ∧ *ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3 ∧ *ϕ*4 ≡ (*ψ* ∧ *ψ*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ* ∧ *ψ*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *α*5)∨

(*ψ* ∧ *ψ*1 ∧ *ψ*2 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ* ∧ *ψ*1 ∧ *ψ*2 ∧ *α*4 ∧ *α*5)∨

(*ψ* ∧ *ψ*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ* ∧ *ψ*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *α*4)∨ (*ψ* ∧ *ψ*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ* ∧ *ψ*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *α*5)∨ (*ψ* ∧ *α*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ* ∧ *α*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *α*5)∨

(*ψ* ∧ *α*1 ∧ *ψ*2 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ* ∧ *α*1 ∧ *ψ*2 ∧ *α*4 ∧ *α*5)∨

(*ψ* ∧ *α*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ* ∧ *α*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *α*5)∨

(*ψ* ∧ *α*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ* ∧ *α*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *α*5)*.*

*ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3 ∧ *ϕ*4 ≡ (*ψ*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *α*5)∨

(*ψ*1 ∧ *ψ*2 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ*1 ∧ *ψ*2 ∧ *α*4 ∧ *α*5)∨

(*ψ*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *ψ*4) ∨ (*ψ*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *α*4)∨ (*ψ*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *ψ*4) ∨ (*ψ*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *α*5)∨ (*α*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *ψ*4) ∨ (*α*1 ∧ *ψ*2 ∧ *ψ*3 ∧ *α*5)∨

(*α*1 ∧ *ψ*2 ∧ *α*4 ∧ *ψ*4) ∨ (*α*1 ∧ *ψ*2 ∧ *α*4 ∧ *α*5)∨

(*α*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *ψ*4) ∨ (*α*1 ∧ *α*2 ∧ *α*3 ∧ *ψ*3 ∧ *α*5)∨

(*α*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *ψ*4) ∨ (*α*1 ∧ *α*2 ∧ *α*3 ∧ *α*4 ∧ *α*5)*.*

所以，

¬FCTL(*FV* ∪{*s*}(*K* ) ∧ ¬*s,* {*s*})

≡ ¬FCTL(*FV* ∪{*s*}(*K* )*,* {*s*})

≡ ¬FCTL(*ψ* ∧ AG(*ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3 ∧ *ϕ*4)*,* {*s*})

≡ ¬(FCTL(*ψ* ∧ *ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3 ∧ *ϕ*4*,* {*s*}) ∧ AGFCTL(*ϕ*1 ∧ *ϕ*2 ∧ *ϕ*3 ∧ *ϕ*4*,* {*s*}))

／(

≡ ¬ *d* ∨ ((*d* ∨ *se*) ∧ AX(*d* ∧ ¬*se*)） ∨ ((*d* ∨ *se* ∨ AX¬*d*) ∧ EX(¬*d* ∧ *se*) ∧ EX(¬*d* ∧ ¬*se*)）

∨ (*d* ∧ AX(¬*d* ∧ ¬*se*)) ∨ ((*d* ∨ ¬*se*) ∧ EX¬*d*）飞∧

(( ） (

AG (*d* ∨ *se*) ∧ AX(*d* ∧ ¬*se*) ∨

AX¬*d* ∧ EX(¬*d* ∧ *se*)∧

＼

EX(¬*d* ∧ ¬*se*)） ∨ ((*d* ∨ *se*) ∧ AX(¬*d* ∧ ¬*se*)） ∨ EX¬*d*飞 。

**5.1.2** 遗忘封闭性及复杂性

当CTL公式的长度（符号的个数）为*n*时，由小模型理论可知，定义在状态个数 为*k* = *n*8*n*的状态空间*S* = {*s*1*, s*2*, . . . , sk*}上的初始Kripke结构能保证公式的可满足性[[95](#_bookmark209)]。 对于其它拥有同样大小的状态空间上的任意初始结构，都能在状态空间*S* 上找到一个 初始结构与之互模拟，且由定理[5.1](#_bookmark105)可知它们有相同的特征公式。 因此，只有有限个初 始结构作为该公式的候选模型。 从而下面结论成立。

引理 **5.5.** 给定*CTL*公式*ϕ*，下面等式成立：

*ϕ* ≡ ／

(*M ,s*0)∈*Mod*(*ϕ*)

*FA* (*M , s*0)*.*

证明**.** 令(*M* ′*, s*′ )为*ϕ*的模型。 由定理[5.1](#_bookmark105)可知(*M* ′*, s*′ ) |= *FA* (*M* ′*, s*′ )，则：

0

(*M* ′*, s*′ ) |= ／

0

0 0

*FA* (*M , s*0)*.*

(*M ,s*0)∈*Mod*(*ϕ*)

另一方面，假定(*M* ′*, s*′ )为V

*FA* (*M , s*0)的模型。则存在(*M , s*0) ∈ *Mod*(*ϕ*)

0 (*M ,s*0)∈*Mod*(*ϕ*)

，使得(*M* ′*, s*′ ) |= *FA* (*M , s*0)。 由定理[5.1](#_bookmark105)可知(*M , s*0) ↔0/ (*M* ′*, s*′ )，从而由定理[3.1](#_bookmark52)可

0 0

知(*M , s*0)是*ϕ*的一个模型。

该结论表明：任意CTL公式都与其模型的特征公式的析取逻辑等价。 这对遗忘理 论的封闭性提供了重要的理论支撑，即：从公式里遗忘原子命题集*V* 中的元素只需找 到与给定公式的模型*V* -互模拟的那些模型就能确定遗忘的结果。 形式化地，对于给定

的公式*ϕ*和原子命题集*V* ，从*ϕ*中遗忘*V* 中的元素得到的结果为：

／

*K* ∈{*K* ′|∃*K* ′′∈*Mod*(*φ* )*, K* ′′↔*V K* ′}

*FV* (*K* )*.*

下面分析遗忘在CTL段CTLAF下推理问题的复杂性结果，其中CTLAF表示CTL公式 只包含时序算子AF的子类。 这一类公式在并发系统中的互斥和等待等属性描述中有重 要作用[[24](#_bookmark155)]。 尽管这类公式是相对简单的，但是下面将要说明判定一个模型是否为此类 公式遗忘结果的模型是NP-完全的。

命题 **5.3** (模型检测)**.** 给定一个结构(*M , s*0)、 原子命题集*V* ⊆ *A* 和公式*ϕ* ∈ *CTL*AF，判

定(*M , s*0) 是否为FCTL(*ϕ,V* )的模型是NP*-*完全的。

证明**.** 在NP中. 假 定(*M , s*0) |= FCTL(*ϕ,V* )， 则 存 在 一 个 初 始 结 构(*M* ′*, s*′ )， 使 得 (a)

0

(*M* ′*, s*′ ) |= *ϕ*且 (b) (*M , s*0) ↔*V* (*M* ′*, s*′ )。 已有结果表明：(a)能在*M* ′和*ϕ*的大小的多

0 0

项式时间内完成[[33](#_bookmark164)]；条件(b)也可用Baier等人的推论7.45[[24](#_bookmark155)]的方法证明能在多项式时间

内完成。 又因为猜(*M , s*0)多项式大小的初始结构(*M* ′*, s*′ )能在多项式时间内完成。 因

0

此，这一问题在NP中。

NP-难. 已经证明命题逻辑下遗忘的模型检测是NP-难的，由于命题逻辑下的遗忘

是CTL的一种[[49](#_bookmark177)]。 因此，上述问题是NP-难的。

关于遗忘的逻辑蕴涵问题也是值得考虑的。 下面讨论CTLAF段下关于遗忘的逻辑

蕴涵问题的复杂性。 下文用记号*α* |=? *β* 表示“*α*是否逻辑蕴涵*β* ”。

定理 **5.2** (Entailment)**.** 令*ϕ*和*ψ*为*CTL*AF中的两个公式，*V* 为原子命题集。 则：

*(i)* 判定FCTL(*ϕ,V* ) |=? *ψ*是*co-*NP*-*完全的，

*(ii)* 判定*ψ* |=? FCTL(*ϕ,V* )是ΠP *-*完全的，

2

*(iii)* 判定FCTL(*ϕ,V* ) |=? FCTL(*ψ,V* )是ΠP *-*完全的。

2

证明**.** (i) co-NP-难. 已有结论表明， 判定公式*ϕ*是否是可满足的是NP-完全的[[124](#_bookmark238)]。 令*ψ* ≡ ⊥，只需证明*ϕ*是不可满足的，该问题是co-NP-难的。 所以，原问题是co-NP-难 的。

在co-NP中. 定理[3.4](#_bookmark70)表明FCTL(*ϕ,V* ) |= *ψ*当且仅当*ϕ* |= *ψ*且IR(*ψ,V* )。 在CTLAF中， 判定*ϕ* |= *ψ*是co-NP的[[124](#_bookmark238)]。 这里证明IR(*ψ,V* )是否成立是co-NP的。 不失一般性地，假 定*ψ*是可满足的，则*ψ*有一个大小为|*ψ*|的多项式的模型。这里讨论该问题的补问题：判 定*ψ*是*V* -有关的，即¬IR(*ψ,V* )。 显然，¬IR(*ψ,V* )当且仅当存在*ψ*的一个模型(*M , s*0)和 大小为|*ψ*|的多项式的初始结构(*M* ′*, s*′ )，使得(*M , s*0) ↔*V* (*M* ′*, s*′ )且(*M* ′*, s*′ ) |= *ψ*。 因

0 0 0

此，判定¬IR(*ψ,V* )是否成立有如下两个步骤：(1)猜两个大小为|*ψ*|的多项式的初始结

构(*M , s*0)和(*M* ′*, s*′ )，使得(*M , s*0) |= *ψ*且(*M* ′*, s*′ ) |= *ψ*，和(2)检查(*M , s*0) ↔*V* (*M* ′*, s*′ )。

0 0 0

显然，(1)和(2)都是能在多项式时间内完成的。

(ii) 在ΠP中. 考虑这一问题的补问题。猜一个大小为|*ψ*|的多项式的*ψ*的模型(*M , s*0)且

2

检查是否(*M , s*0) |= FCTL(*ϕ,V* )。由命题[5.3](#_bookmark108) 可知这一问题在ΣP 中。因此，原问题在ΠP 中。

2 2

2 -难. 令*ψ* ≡ ⊤。 则这一问题被规约为判定FCTL(*ϕ,V* )的有效性问题。 由于命题遗

ΠP

忘是CTL遗忘的特殊情形，因此该问题的困难属性直接来源于文献[[119](#_bookmark233)]。

(iii) 在ΠP中. 假定FCTL(*ϕ,V* ) |= FCTL(*ψ,V* )。则存在一个初始结构(*M , s*) |= FCTL(*ϕ,V* ) 且(*M , s*) |= FCTL(*ψ,V* )，即存在一个初始结构(*M*1*, s*1) ↔*V* (*M , s*)，使得(*M*1*, s*1) |= *ϕ* 但 是对其它(*M , s*) ↔*V* (*M*2*, s*2)，使得(*M*2*, s*2) |= *ψ*。由命题[5.3](#_bookmark108)的证明可知，(*M , s*)和(*M*1*, s*1) 能在|*ϕ*|、 |*ψ*|和|*V* |的多项式时间内完成。 显然，猜使得(*M*1*, s*1) ↔*V* (*M , s*) 成立的大 小为|*ϕ*|多项式的(*M , s*)和(*M*1*, s*1) 可以在多项式时间内完成， 且对任意(*M , s*) ↔*V* (*M*2*, s*2)， 检 查(*M*2*, s*2) |= *ψ*可 以 在|*ψ*|和|*M*2|的 多 项 式 时 间 内 完 成。 因 此， 该 问 题

2

是ΠP 的。

2

2 -难. 由于IR(FCTL(*ψ,V* )*,V* )，因此FCTL(*ϕ,V* ) |= FCTL(*ψ,V* )当且仅当*ϕ* |= FCTL(*ψ,*

ΠP

*V* )。 所以由(ii)不难证明该问题。

定理[5.2](#_bookmark109)蕴涵下列结论。

推论 **5.2.** 令*ϕ*和*ψ*为*CTL*AF中的两个公式，*V* 原子公式集。 则

*(i)* 判定*ψ* ≡? FCTL(*ϕ,V* )是ΠP *-*完全的，

2

1. 判定FCTL(*ϕ,V* ) ≡? *ϕ*是*co-*NP*-*完全的，
2. 判定FCTL(*ϕ,V* ) ≡? FCTL(*ψ,V* )是ΠP *-*完全的。

2

**5.1.3** 基于模型的遗忘算法

这里基于模型的遗忘算法，该算法与第[5.2.1.3](#_bookmark119)节中基于语法的算法不同。 本小节 的算法通过计算所有可能的模型来计算遗忘，且其正确性可由引理[5.5](#_bookmark107)和定理[5.1](#_bookmark105)保证。 下面的例子来源于研究背景和意义部分。 这里给出如何计算从规范中遗忘一些原

子命题。

例 **5.3.** 对于图[1.1](#_bookmark9)中的*K*1，假定一个已知的规范（性质）为*α* = EF(*se* ∧ *sp*)。 显然*K*1满 足*α*。 若想移除掉*sp*，即从*α*中遗忘*sp*，则FCTL(*α,* {*sp*}) ≡ EF*se*。 此时，该汽车制造公 司可以用新的规范EF*se*（这保证驾车最终会被生产）引导未来汽车的生产。

如下所述，通过计算所有可能的模型是非常低效的。 但是，这为从理论的角度探

索遗忘有重要作用。

算法 **5.1** 基于模型的CTL遗忘过程

**Input**: CTL公式*ϕ*和原子命题集*V*

**Output**: FCTL(*ϕ,V* )

**1** *ψ* ← ⊥ **foreach** *A* 和*S* 上的初始结构*K* **do**

**2 if** *K* |= *ϕ* **then continue**

**foreach** 满足*K* ↔*V K* ′的初始结构*K* ′ **do**

**3** *ψ* ← *ψ* ∨ *FV* (*K* ′)

**4 end**

**5 end**

**6 return** *ψ*

命题 **5.4.** 令*ϕ*为*CTL*公式，*V* ⊆ *A* 为原子命题集，状态空间大小为|*S* | = *m*，|*A* | = *n*，

|*V* | = *x*。则使用算法[*5.1*](#_bookmark0)计算从*ϕ*中遗忘*V* 中原子的空间复杂度为*O*((*n*−*x*)*m*2(*m*+2)2*nm* log *m*)，

且时间复杂性至少与空间复杂性相同。

证明**.** 假 定 每 个 状 态 或 原 子 命 题 占 据log *m*的 空 间 且*n* ≤ *m*， 则 一 个 状 态 对(*s, s*′)占 据2 × log *m*位（bit）。 对任意*B* ⊆ *S* 和*s*0 ∈ *B*，构造如下初始结构(*M , s*0)，其中*M* = (*B, R, L, s*0)， 则*R*中 至 多 有|*B*| 个 状 态 对 且*L*中 至 多|*B*| × *n*个(*s, A*)对（*A* ⊆ *A* ）。 因 此，

2

2

(*M , s*0)至多占据(|*B*| + |*B* 2

|

+ |*B*| × *n*) × log *m*位。 此外，对于状态集*B*，初始状态有|*B*|种

选择，有|*B*||*B*|种关系*R*，(2*n*)|*B*|种标签函数*L*。

在最坏的情形下（即|*B*| = *m*），可构成*m* × (*mm* × 2*nm* × *m*)个初始结构。 因此，最多

有*mm*+2 × 2*nm*个初始结构，且最多花费(*mm*+2 × 2*nm* × (*m* + *m*2 + *nm*)) × log *m*位。

令*k* = *n* − *x*，对任意有*i* ≥ 1个状态的初始结构*K* = (*M , s*0)，其中*M* = (*B, R, L, s*0)。 在最坏的情形下（即*ch*(*M ,V* ) = *i*时），需要*N*(*i*) = *Pi*(*s*0) + *i* × (*Pi*(*s*) + *i* × *Pi*(*s*′))位来存 储*K* 在*V* 上的特征公式。其中*s*′*, s* ∈ *B*，*Pi*(*y*)是存储*FV* (Tr*i*(*y*))（*y* ∈ *B*）所用的空间（这 里假定EX和AX使用了相同的存储单位）。

*FV* (Tr*i*(*y*))（0 ≤ *n* ≤ *i*）被递归地定义如下：

(1)*n* = 0*, P*0(*y*) = *k*

(2)*n* = 1*, P*1(*y*) = *k* + *i* × *k* = *k* + *i* × *P*0(*y*)

(3)*n* = 2*, P*2(*y*) = *k* + *i* × (*k* + *i* × *k*) = *k* + *i* × *P*1(*y*)

*. . . . . .*

(*i* + 1)*n* = *i, Pi*(*y*) = *k* + *i* × *Pi*−1(*y*)*.*

因此，有

*Pi*(*y*) = *k* + *i* × *k* + *i*2 × *k* + · · · + *ii* × *k* =

*ii* 1

−

*k,*

*i* − 1

*N*(*i*) = *Pi*(*s*0) + *i* × (*Pi*(*s*) + *i* × *Pi*(*s*′))

= (*i*2 + *i* + 1)*Pi*(*y*)

= (*i*2 + *i* + 1) *i* − 1 *k.*

*i*

*i* − 1

在最坏的情形下（即具有*m*个状态的初始结构）有*mm*+2 × 2*nm*个初始结构， 需

要(*mm*+2 × 2*nm* × *N*(*m*)) × log *m*位来存储遗忘的结果。

因此，空间复杂性为*O*((*n* − *x*)*m*2(*m*+2)2*nm* × log *m*)。

**5.2** 基于归结的遗忘计算方法

已有结果显示，任意CTL公式可以转换为SNF*g*

CTL

子句集。 归结是一种以子句为计

算对象的判断可满足性的方法。 本章提出一种基于归结的计算遗忘的方法，其主要思

想是：首先，将给定的CTL公式转换为SNF*g*

CTL

子句集（这一部分在第[二](#_bookmark18)章已经给出了

详细的介绍；其次，在相应的原子命题集*V* 上使用归结规则得到归结结果；最后，“消 除”之前引入的索引和**start**，最终得到遗忘的结果。 其主要流程图如图[5.2](#_bookmark112)所示。 正如 本章所要说明的那样，CTL不具有均匀插值这一属性，基于归结的方法在有的情况下 是不能计算出遗忘结果的。 然而，在有些CTL子类下，本章提出的方法能够计算出其 遗忘结果。

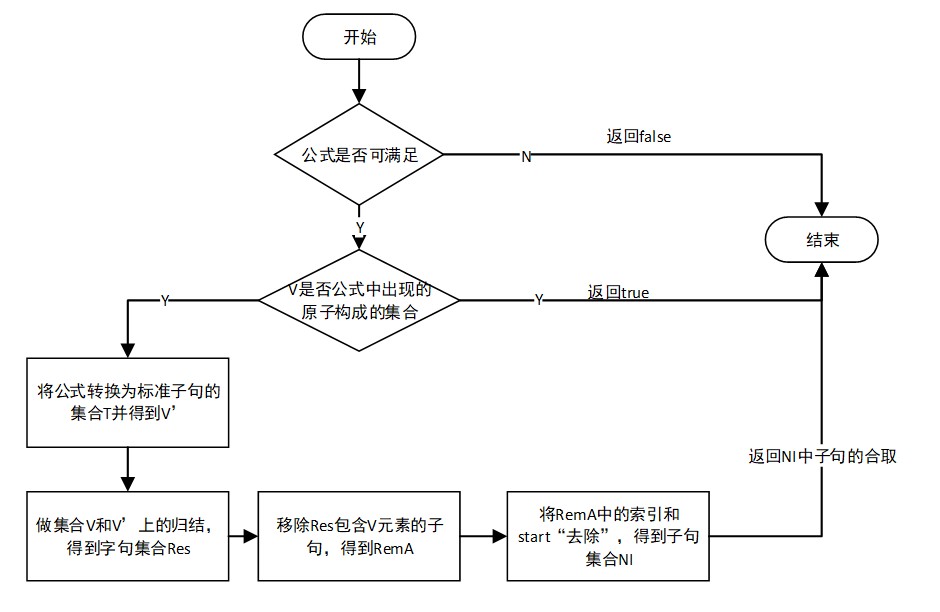


图 5.2: 基于归结的遗忘的主要流程图

本章展示如何使用第[2.4](#_bookmark34)节中表[2.3](#_bookmark35)中的归结规则来计算CTL下的遗忘。 为了计算

从CTL公式*ϕ*中遗忘集合*V* 中的原子命题，需要解决如下两个主要问题：

(1) 如何表示CTL公式和带索引的CTL公式之间的关系？如在第[二](#_bookmark18)章中所说，将一

个CTL公式转换为SNF*g*

CTL

子句集会引入新的原子命题和索引。 虽然已有的研究

说明，CTL公式可以转换为带索引的公式集并保证其可满足性，然而并没有表 明这两种形式的公式之间的模型具有怎样的联系。 本章从互模拟等价的角度探 讨CTL公式和归结等过程得到的（带索引）公式之间的联系，为计算遗忘提供理 论基础。

(2) 如何“移除”无关的原子命题（包括需要遗忘的原子命题和转换过程中引入的新 的原子命题），以及如何“消除”索引？为此，本章给出“移除”原子命题的一 般操作，并提出一种一般化的Ackermann引理。 为了“消除”索引，探索几个逻 辑等价关系。

本章其余部分组织如下：首先，在第[5.2.1](#_bookmark113)节详细介绍CTL-forget各个过程及算法复 杂性；其次，在第节介绍基于Prolog的CTL-算法的实现；最后，在第[5.2.3](#_bookmark122)节给出并分 析实验结果。

* + 1. 基于归结的算法**CTL-forget**

通过上述分析，本节主要描述如何使用基础知识部分说的*R*≻*,S* 归结系统对要遗忘

CTL

的原子命题做归结，然后如何将归结后得到的结果等价转换为CTL公式。

* + - 1. **CTL**归结**UF**

这部分给出如何使用归结规则（表[2.3](#_bookmark35)）来计算CTL中的遗忘。 这个过程的主要思

想是：将转换过程中获得的SNF*g*

CTL

子句在表[2.3](#_bookmark35)中归结规则上做穷尽的归结，即直到不

产生新的归结结果。 值得注意的是，该归结过程涉及的归结原子命题包括：需要遗忘

的原子命题和转换过程中引入的新的原子命题。

令*T* 为SNF*g*

CTL

子句集， *p*为原子命题。 *T* 在*p*上的展开（记为UF(*T, p*)）是集合*T* 和

如下集合的并集：

{*α* | *α* 是*T* 中的公式关于文字*l* ∈ {*p,* ¬*p*} 的归结结果}*.*

对于原子命题集*V* ，定义UF(*T,* 0/ ) = *T* 且UF(*T,* {*p*}∪*V* ) = UF(UF(*T, p*)*,V* )。直观上，*T* 在*p*上

的展开是对*p*做穷尽的归结，即：直到对*p*使用规则**SRES1-8**不再产生新的SNF*g* 子句

CTL

为止（即使使用了重写规则和可能规则之后也不会产生新的子句）。

下面的命题展示了对任意CTL公式*ϕ*，从UF(*Tϕ ,V* ) 中移除掉含有*V* 中元素的子句 得到的结果在不考虑*V* 中元素和新引入的元素的情况下是互模拟等价 的。 下文中 记*ERes*(*ϕ,V* ) = {*α* ∈ UF(*Tϕ ,V* ) | *Var*(*α*) ∩ *V* = 0/ }。

命题 **5.5.** 令*ϕ* 为一个*CTL*公式，*V* ⊆ *A* 为原子命题集。 则*Tϕ* ≡*U* ERes(*ϕ,V* )，其中*U* =

*Var*(UF(*Tϕ ,V* )) − (*Var*(*ϕ*) − *V* )。

证明**.** 从两个方面来证明这一结论：**(F1)** *Tϕ* ≡*U* UF(*Tϕ ,V* )，**(F2)** UF(*Tϕ ,V* ) ≡*U ERes*(*ϕ,V* )。

为此，定义如下由SNF*g*

CTL

子句集构成的序列：*T*0 = *Tϕ , T*1*, T*2*, . . . , Tn* = UF(*Tϕ ,V* )，其

中*Ti*+1 = *Ti* ∪ *Ri* (0 ≤ *i < n*)、 *Ri* 是对Π ⊆ *Ti*使用一条匹配的规则*r*且该规则归结的原子命

题为*p* ∈ *V* ，这一过程记为Π →*r Ri*。

**(F1)** 为了证明*Tϕ* ≡*U* UF(*Tϕ ,V* )，只需证明对任意0 ≤ *i < n*，有*Ti* ≡*U Ti*+1。

(1) 若*r* ∈ {**(SRES1)***, . . . ,* **(SRES8)***,* (**RW1**)*,* (**RW2**)}，则*Ti* ≡{*p*} *Ti*+1，其中。

一方面，显然Π |= *Ri*，所以*Ti* |= *Ti*+1。 另一方面*Ti* ⊆ *Ti*+1，所以*Ti*+1 |= *Ti*。

(2) 下证若Π →*r Ri*且*r* =**(ERES1)**，则*Ti* ≡{*l,w*A *T*

，其中*l* = *p*或*l* = ¬*p*，*w*A

是与子

¬*l* }

*i*+1 ¬*l*

句*Q* → AF¬*l*相关的新的原子命题，*l*是文字（即：*p*或者¬*p*）。

已有结果表明Π |= *Ri* [[125](#_bookmark239)]，因此，*Ti*+1 = *Ti* ∪ ΛA ，其中ΛA 是通过使用表[2.1](#_bookmark29)中的

¬*l* ¬*l*

转 换 规 则 作 用 到*Ri*上 得 到 的SNF*g*

CTL

子 句 集[[126](#_bookmark240)]。 显 然， 对 所 有(*M*1*, s*1) ∈ *Mod*(*Ti* =

*X* ∪ Π)都 存 在 一 个(*M*2*, s*2) ∈ *Mod*(*Ti*+1 = *Ti* ∪ ΛA )， 使 得(*M*1*, s*1) ↔

(*M , s* )，

¬*l* ⟨{*p,w*A }*,*Ø⟩ 2 2

¬*l*

且 对 任 意(*M*2*, s*2) ∈ *Mod*(*Ti*+1 = *Ti* ∪ ΛA )， 存 在 一 个(*M*1*, s*1) ∈ *Mod*(*Ti* = *X* ∪ Π)， 使

¬*l*

得(*M*1*, s*1) ↔{*p,w*A

(*M , s* )。 又{*p, w*¬*l* } ⊆ *U*，由命题[3.2](#_bookmark55)可知*T* ≡ *T* 。

¬*l* } 2 2

A

*i U i*+1

当规则为**(ERES2)**时可以类似地证明。

总 之，*V* ⊆ *U* .因 此， 由 推 论[3.1](#_bookmark54)(iii)可 知， 对 任 意0 ≤ *i < n*， 有*Ti* ≡*U Ti*+1。 又 因

为≡*U* 为等价关系，所以*Tϕ* ≡*U* UF(*Tϕ ,V* )。

**(F2)** 假设*V* = {*p*}，*Ci* (*i* = 1*,* 2)为经典子句， *l* = *p* 或*l* = ¬*p*。 显然UF(*Tϕ ,V* ) |= *ERes*(*ϕ,V* )，这里证明对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))（其中*M* = (*S, R, L,* [ ]*, s*)）， 存在一个初始Ind-结构*K* ′ = (*M* ′*, s*′)，使得*K* ↔*U K* ′ 且*K* ′ |= UF(*Tϕ ,V* )。

因为*p*只出现在SNF*g*

CTL

子句的右手边，从以下几点证明上述结论成立。

(1) 假定UF(*Tϕ ,V* )有全局子句，则对于任意*C* = ⊤ → *C*1 ∨ *l* ∈ UF(*Tϕ ,V* )：

(a) 如果不存在*C*′ ∈ UF(*Tϕ ,V* )，使得*C* 和*C*′ 在*p*上是可归结的，则UF(*Tϕ ,V* )中不存

在除了*Pt*-某时子句之外的子句*C*′ 包含文字¬*l*，其中*Pt* ∈ {A*,* E}。

若对任意其它的子句*C*′，*p* ̸∈ *Var*(*C*′)，则对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))，如

下构造(*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ]*, s*) （即*s*′ = *s*），其中*L*′ 与*L* 一样，除了对任意*s*1 ∈ *S*，

若(*M , s*1) |= *C*1 ∨ *l*则“若*l* = *p*令*L*′(*s*1) = *L*(*s*1) ∪ {*p*}，否则令*L*′(*s*1) = *L*(*s*1) − {*p*}”。 显

然(*M , s*) ↔{*p*} (*M* ′*, s*′) 和(*M* ′*, s*′) |= *C*′ ∧ *C*。

若*C*′ = *Q* → *Pt* F¬*l*，不失一般性地，令*l* = *p*。 对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ, V* ))，如下构造(*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ]*, s*)，其中*L*′与*L* 相同，除了*L*′(*s*) = *L*(*s*)∪{*p*}， 且对任意具有(*s, s*′) ∈ *R*关系的状态*s*′ ∈ *S*′，令*L*′(*s*) = *L*(*s*) − {*p, q*}，其中*q* ∈ (*Var*(UF(*Tϕ , V* )) − *Var*(*ϕ*)) 是负出现在*C*1 中的原子命题，且对任意其它的非初始状态*s*′′ ∈ *S*，*L*′(*s*′′) = (*L*(*s*′′) − {*Q*}) ∪ {*p*} （*Q*在*Pt*-某时子句中是一个原子）。 显然，(*M , s*) ↔{*p,q*} (*M* ′*, s*′) 和(*M* ′*, s*′) |= *C*′ ∧ *C*。

(b) 若存在子句*C*′ ∈ UF(*Tϕ ,V* )，使得*C*和*C*′在*p*上是可归结的：

(i) 若*C*′ = *Q* → *Pt* X(*C*2 ∨ ¬*l*) （令*Pt* = A，*Pt* = E可类似地证明），则有*Q* → AX(*C*1 ∨ *C*2) ∈ UF(*Tϕ ,V* )。 因此，对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))，如下构造*K* ′ = (*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ] *, s*) （即*s*′ = *s*），其中*L*′ 与*L*一样，除了对任意*s*1 ∈ *S*， 若(*M , s*1) |= *Q*，则对任意(*s*1*, s*2) ∈ *R*，若(*M , s*2) |= *C*1 则“若*l* = *p*，则令*L*′(*s*2) = *L*(*s*2) ∪{*p*}，否则令*L*′(*s*2) = *L*(*s*2) −{*p*}”，否则，若(*M , s*2) |= *C*1 ∧¬*C*2，则“若*l* = *p*， 则令*L*′(*s*2) = *L*(*s*2) − {*p*}， 否则令*L*′(*s*2) = *L*(*s*2) ∪ {*p*}”；否 则， 若(*M , s*2) |=

¬*C*1 ∧ *C*2，则“若*l* = *p*，则令*L*′(*s*2) = *L*(*s*2) ∪ {*p*}，否则*L*′(*s*2) = *L*(*s*2) − {*p*}”。 显

然，*K* ↔{*p*} *K* ′ 且*K* ′ |= *C*′ ∧ *C*。

(ii) 若*C*′ = *Q* → *Pt* F¬*l*。 不失一般性地，假设*l* = *p*。*C*和*C*′在*p*上是可归结的，则一定

存在子句集{*P*1 → \**C*1, . . . , *P*1

→ \**C*1

, *Pn* → \**Cn*, . . . , *P*1

→ \**C*1

}，使得\*要么要

1 1 *m*1

*m*1 1 1

*mn mn*

么为空字符串，要么为{AX*,* E⟨*ind*⟩ X}中的一个（¬*C*1 → *l*为子句集中的一个），使

得V*n*

八*mi* *i*

*i*=1

*j*=1 *Pj* → EXEG*l*。 因此，通过使用规则ERES1 （ERES2规则类似）可以得

到一个子句*C*′′ = ⊤ → ¬*Q* ∨ ¬*p* ∨ *C*1。 所以，在使用规则SRES8在*C*和*C*′′上后得到

子句⊤ → ¬*Q* ∨ *C*1。 在这种情况下，对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))，如下 构造*K* ′ = (*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ]*, s*) （即*s*′ = *s*），其中*L*′与*L*一样，除了对任 意*s*1 ∈ *S*，若(*M , s*1) |= *Q*，则令*L*′(*s*1) = *L*(*s*1) − {*p*}，否则令*L*′(*s*1) = *L*(*s*1) ∪ {*p*}。 可以检查*K* ↔{*p*} *K* ′和*K* ′ |= *C*′ ∧ *C*。

(ii) 可以类似地证明其它类型的子句，且得到*K* ↔*U K* ′ 和*K* ′ |= UF(*Tϕ ,V* )。

(2) 考虑*Pt*-步子句的情形。令*C* ∈ UF(*Tϕ ,V* )为*Q* → AX(*C*1 ∨ *l*)。不失一般性地，假设

存在某些子句*C*′ ∈ UF(*Tϕ ,V* )，使得*C*和*C*′在*p*上是可归结的（*l* = *p*）。

若*C*′ = *Q*1 → *Pt* X(*C*2 ∨ ¬*l*) （令*Pt* = E*ind* ，*Pt* = A的情形可以类似地证明），因此， *Q* ∧ *Q*1 → E*ind* X(*C*1 ∨ *C*2) ∈ UF(*Tϕ ,V* )。 所以，对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))，如 下构造*K* ′ = (*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ]*, s*)（即*s*′ = *s*）， 其中*L*′与*L*一样， 除了对任 意*s*1 ∈ *S*

第五章 CTL遗忘计算方法

(i) 若(*M , s* ) |= *Q* ∧ *Q* 则“若(*M , s* ) |= ¬*Q* ∧ *Q* 则（若对(*s , s*′ ) ∈ *π*⟨*ind*⟩

有(*M , s*′ ) |= *C*2

2 *s* 2

1 1 1 1 1

则令*L*′(*s*′ ) = *L*(*s*′ ) − {*p*}否则令*L*′(*s*′ ) = *L*(*s*′ )），否则若(*M , s*1) |= *Q* ∧ ¬*Q*1则对任

2 2 2 2

意(*s*1*, s*2) ∈ *R*，（若(*M , s*2) |= *C*1则令*L*′(*s*2) = *L*(*s*2) ∪{*p*}，否则令*L*′(*s*′ ) = *L*(*s*′ )），否

2 2

则令*L*′(*s*′ ) = *L*(*s*′ )”。

2 2

*ind*⟩

(ii) 否则若(*M , s*1) |= *Q* ∧ *Q*1，则对(*s*1*, s*′ ) ∈ *π*⟨

有(*M , s*′ ) |= *C*1 ∨*C*2。因此，若(*M , s*′ )

2 *s* 2 2

|= *C*1 ∧ ¬*C*2则*L*′(*s*′ ) = *L*(*s*′ ) − {*p*}，否则若(*M , s*′ ) |= ¬*C*1 ∧ *C*2则令*L*′(*s*′ ) = *L*(*s*′ ) ∪

2 2 2 2 2

{*p*}，否则令*L*′(*s*′ ) = *L*(*s*′ )。对其它满足(*s*1*, s*2) ∈ *R*和*s*2 ̸= *s*′ 的状态*s*2，若(*M , s*1) |=

2 2 2

*Q*和(*M , s*2) |= ¬*C*1，则令*L*′(*s*2) = *L*(*s*2) ∪ {*p*}，否则令*L*′(*s*′ ) = *L*(*s*′ )。

2 2

显然，*K* ↔{*p*} *K* ′和*K* ′ |= *C*′ ∧ *C*，其中*K* ′ = (*M* ′*, s*′)。

若*C*′ = *Q*1 → *Pt* F¬*l* （令*Pt* = A，*Pt* = E的情形可类似地证明）。 若*C*和*C*′在*p*上是

可归结的，则必须存在一个包含子句¬*C*1 → *l*的SNF*g*

子句集{*P*1 → \**C*1, . . . , *P*1 →

CTL

1 1 *m*1

\**C*1

, *Pn* → \**Cn*, . . . , *P*1

→ \**C*1

}， 使 得\*为 空 字 符 串 或 集 合{AX*,* E

X}中 的 一 个，

*m*1

且V*n*

1 1

八*mi* *i*

*mn mn*

⟨*ind*⟩

*i*=1

*j*=1 *Pj* → EXEG*l*。 在这种情况下，对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))，如下

构造(*M* ′*, s*′)：令*M* ′ = (*S, R, L*′*,* [ ]*, s*) （即*s*′ = *s*），其中*L*′和*L*一样，除了对任意状态*s*′ ∈

*S*′，若*L*′(*s*′) |= *Q*和(*s*′*, s*′′) ∈ *R*，则令*L*′(*s*′′) = *L*(*s*′′) ∪ {*p*}，若(*M , s*) |= *Q*1，则令*L*′(*s*) =

*L*(*s*) − {*p*}。 显然，(*M , s*) ↔{*p*} (*M* ′*, s*′)和(*M* ′*, s*′) |= *C*′ ∧ *C*。

因此，对任意*K* = (*M , s*) ∈ *Mod*(*ERes*(*ϕ,V* ))（*M* = (*S, R, L,* [ ]*, s*)），存在一个初

始Ind-结构*K* ′ = (*M* ′*, s*′)，使得*K* ↔*U K* ′和*K* ′ |= UF(*Tϕ ,V* )。

例 **5.4** (例[3.3](#_bookmark56)的延续)**.** 令*V* = {*p, r*}，则UF(*Tϕ ,V* ∪ {*x, y, z*}) 除了例[3.3](#_bookmark56)中的子句，还包含

如下子句：

|  |  |  |  |
| --- | --- | --- | --- |
| (1) **start** → *r* | (1*,* 2*,* **SRES5**) | (2) **start** → *x* ∨ *y* | (1*,* 4*,* **SRES5**) |
| (3) ⊤ → ¬*z* ∨ *y* ∨ *f* ∨ *m* | (3*,* 4*,* **SRES8**) | (4) *y* → AX( *f* ∨ *m* ∨ *y*) | (3*,* 8*,* **SRES6**) |
| (5) ⊤ → ¬*z* ∨ *x* ∨ *p* | (4*,* 5*,* **SRES8**) | (6) ⊤ → ¬*z* ∨ *x* ∨ *q* | (4*,* 6*,* **SRES8**) |
| (7) *y* → AX(*x* ∨ *p*) | (5*,* 8*,* **SRES6**) | (8) *y* → AX(*x* ∨ *q*) | (6*,* 8*,* **SRES6**) |
| (9) **start** → *f* ∨ *m* ∨ *y* | (3*,* (2)*,* **SRES5**) | (10) **start** → *x* ∨ *p* | (5*,* (2)*,* **SRES5**) |
| (11) **start** → *x* ∨ *q* | (6*,* (2)*,* **SRES5**) | (12) ⊤ → *p* ∨ ¬*z* ∨ *f* ∨ *m* | (5*,* (3)*,* **SRES8**) |
| (13) ⊤ → *q* ∨ ¬*z* ∨ *f* ∨ *m* | (6*,* (3)*,* **SRES8**) | (14) *y* → AX(*p* ∨ *f* ∨ *m*) | (5*,* (4)*,* **SRES6**) |
| (15) *y* → AX(*q* ∨ *f* ∨ *m*) | (6*,* (4)*,* **SRES6**) | (16) **start** → *f* ∨ *m* ∨ *p* | (5*,* (9)*,* **SRES5**) |
| (17) **start** → *f* ∨ *m* ∨ *q* | (6*,* (9)*,* **SRES5**) |  |  |

在从UF(*Tϕ ,V* ∪ {*x, y, z*})中移除掉包含*V* 中元素的子句后，得到*ERes*(*ϕ,V* )，其包含

如下子句：

**start** → *z,* **start** → *f* ∨ *m* ∨ *q,* **start** → *x* ∨ *y,* **start** → *q* ∨ *x,* **start** → *f* ∨ *m* ∨ *y,*

⊤ → *f* ∨ *m* ∨ ¬*x,* ⊤ → *q* ∨ *f* ∨ *m* ∨ ¬*z,* ⊤ → *f* ∨ *m* ∨ ¬*z* ∨ *y,*

⊤ → *q* ∨ *x* ∨ ¬*z,* ⊤ → *x* ∨ *y* ∨ ¬*z,* ⊤ → *q* ∨ ¬*y, z* → AF*x,*

*y* → AX(*q* ∨ *f* ∨ *m*)*, y* → AX(*x* ∨ *q*)*, y* → AX(*x* ∨ *y*)*, y* → AX( *f* ∨ *m* ∨ *y*)*.*

可以看出，尽管*ERes*(*ϕ,V* )中不包含具有索引的公式，但是有的子句包含出现

在*Tϕ* 中的新原子命题。

* + - 1. 转换**SNF***g*

**CTL**

子句为**CTL**公式

在上一节中描述了归结过程，但是正如归结规则和转换规则所示，在这两个过程

中会引入索引和新的原子命题。 本节介绍如何移除掉这些新引入的元素。

下面的引理表明，可以移除掉SNF*g*

CTL

子句集中的索引而保持互模拟等价。

引理 **5.6.** 令 *j* ∈ *I* ，*ψi, ϕi* (1 ≤ *i* ≤ *n*)为*CTL*公式。 有：

*(i)* {*ψi* → E⟨ *j*⟩ X*ϕi* | 1 ≤ *i* ≤ *n*} ≡ {(八

*i*∈*S*

*ψi*) → E⟨ *j*⟩ X (八

*ϕi*) | *S* ⊆ {1*,..., n*}}，

*(ii)* {*ψi* → E⟨ *j*⟩ X*ϕi* | 1 ≤ *i* ≤ *n*} ≡0/ {(八

*i*∈*S*

*i*∈*S*

*ψi*) → EX (八

*ϕi*) | *S* ⊆ {1*,..., n*}}，

*(iii)* {(*ψ*1 → E⟨ *j*⟩ F*ϕ*1)*,* (*ψ*2 → E⟨ *j*⟩ X*ϕ*2)} ≡0/

*i*∈*S*

(*ψ*1 → *ϕ*1 ∨ EXEF*ϕ*1) ∧ (*ψ*2 → EX*ϕ*2) ∧ (*ψ*1 ∧ *ψ*2 → ((*ϕ*1 ∧ EX*ϕ*2) ∨ EX(*ϕ*2 ∧ EF*ϕ*1)))*.*

证明**.** (i) (⇒) 对等式左边公式的任意模型(*M , s*0)，若(*M , s*0) |= 八*m*

*i*=1

*Pji* （ *ji* ∈ {1*,..., n*}，

1 ≤ *m* ≤ *n*），则存在*s*0的下一个状态*s*1，使得(*s*0*, s*1) ∈ [ *j*]和(*M , s*1) |= 八*m*

*i*=1

*ϕji*。 由[ *j*]的定

义可知(*s*0*, s*1) ∈ *R*，因此(*M , s*0) |= 八*m*

*i*=1

*Pji*

→ E⟨ *j*⟩

八*m*

*i*=1

X(

*ϕji* )。

(⇐) 显然等式左边集合为右边的子集， 因此：{(八

*i*∈*S*

*ψi*) → E⟨ *j*⟩ X (八

*ϕi*) | *S* ⊆

{1*,..., n*}} |= {*ψi* → E⟨ *j*⟩ X*ϕi* | 1 ≤ *i* ≤ *n*}。

*i*∈*S*

(ii) (⇒) 对等式左边公式的任意模型(*M , s*0)，若(*M , s*0) |= 八*m*

*i*=1

*Pji* （ *ji* ∈ {1*,..., n*}，

1 ≤ *m* ≤ *n*），则存在*s*0的下一个状态*s*1，使得(*s*0*, s*1) ∈ [ *j*]且(*M , s*1) |= 八*m*

*i*=1

*ϕji*。 由[ *j*]的定

义可知(*s*0*, s*1) ∈ *R*，因此(*M , s*0) |= 八*m*

*i*=1

*Pji*

→ EX(八*m*

*ϕji* )。

(⇐) 对等式右边公式的任意模型(*M , s*0)， 若(*M , s*0) |= 八*m*

*i*=1

*i*=1

*Pji* （ *ji* ∈ {1*,..., n*}，

1 ≤ *m* ≤ *n*），则存在*s*0的下一个状态*s*1，使得(*M , s*1) |= 八*m*

*i*=1

*ϕji*。 容易构造一个初始Ind-

结构(*M* ′*, s*0)（*M* ′ = (*S, R, L,* [ ]′*, s*0)）， 使得(*M* ′*, s*0)与(*M , s*0)相同， 除了(*s*0*, s*1) ∈ [ *j*]′，

即(*M , s*0) ↔0/ (*M* ′*, s*0)且(*M* ′*, s*0) |= {*ψi* → E⟨ *j*⟩ X*ϕi* | 1 ≤ *i* ≤ *n*}。

(iii)的证明与(ii)的证明类似。

算法 **5.2** RM-index(Σ)

**Input**: A finite set Σ of SNF*g*

CTL

**Output**: A set of CTL formulas

clauses

**7 foreach** Σ中拥有相同索引⟨*i*⟩的E*-*子句构成的极大子集∆ **do**

**8 if** 存在索引为⟨*i*⟩的E*-*某时子句*α* ∈ Σ **then**

**9 foreach** *β* ∈ *rei*(∆) **do** Σ ← Σ ∪ *r f i*(*α, β* ) Σ ← Σ − {*α*}

**10 end**

**11** Σ ← Σ − ∆ ∪ *rxi*(∆)

**12 end**

**13 return** Σ

本文将引理[5.6](#_bookmark116)中(i)、(ii)、(iii)等号≡\*（\* ∈ {空字符串，0/ }）的右边分别用*rei*({*αi* | 1 ≤ *i* ≤ *n*})、*rxi*({*αi* | 1 ≤ *i* ≤ *n*})、*r f i*({*β*1*, α*2})来表示，其中*αi* = *ψi* → E⟨ *j*⟩ X*ϕi* (1 ≤ *i* ≤ *n*) 和*β*1 = *ψ*1 → E⟨ *j*⟩ F*ϕ*1。

因为EX*ϕ*1 ∧ EX*ϕ*2 |= EX(*ϕ*1 ∧ *ϕ*2)，引理[5.6](#_bookmark116)(i)的目的是为了保证若*ψ*1和*ψ*2在当前状 态满足，则*ϕ*1和*ϕ*2被同一条路径满足，这可以推广到多个*ψi* (1 ≤ *i* ≤ *n*)的情形。 (iii)表 示可以将每一个E-某时子句和一个E-步子句结合得到新的满足互模拟等价的CTL公式。 (ii)表明拥有相同索引的E-步子句可以结合成新的满足互模拟等价的CTL公式。 这一过 程可由算法[5.2](#_bookmark0)实现。 下面的推论是上述引理的一个简单应用，它表明当移除索引之

后Σ和RM-index(Σ)是互模拟等价的。

推论 **5.3.** 令*ϕ* 为一个*CTL*公式、*V* ⊆ *A* 为原子命题集、 Σ = ERes(UF(*ϕ,V* ∪ *U* )*,V* )，其

中*U* = *Var*(*Tϕ* ) − *Var*(*ϕ*)。 则*RM-index*(Σ) ≡0/ Σ。

通过下面的定理，可以移除掉一些新引入的原子命题而保持互模拟等价。

引理 **5.7** (一般化的Ackermann引理，Generalised Ackermann’s Lemma)**.** 令*x*为一个原子 命题、∆ = {AG(⊤ → ¬*x* ∨*C*1)*, . . . ,* AG(⊤ → ¬*x* ∨*Cn*)*,* AG(*x* → *B*1)*,...,* AG(*x* → *Bm*)}为只包 含一个*x*的*CTL*公式集（*n, m* ≥ 1）、Γ为*x*正出现在其中的有限个*CTL*公式集。 则有：

I

Γ ∪ ∆ ≡{*x*} Γ *x/*

八 ({*Ci* | 1 ≤ *i* ≤ *n*} ∪ {*Bi* | 1 ≤ *i* ≤ *m*})

*.* (5.1)

证明**.** 令*ϕ* = 八 ({*Ci* | 1 ≤ *i* ≤ *n*} ∪ {*Bi* | 1 ≤ *i* ≤ *m*})。 不失一般性地， 令Γ为一个CTL公

式，*M* = (*S, R, L,* [ ]*, s*0)，*ψi*(*x*) (*i* = {1*,* 2})为*x*正出现在其中的CTL公式。

(⇒) 对公式Γ∧八 ∆的任意模型(*M , s*0)，这里证明通过归纳公式Γ的结构证明(*M , s*0)

|= Γ[*x/ϕ*]。

基始. 令Γ = *x*，因为(*M , s*0) |= *x*，显然(*M , s*0) |= *ϕ*成立。

归纳步. (1) 令Γ = *ψ*1(*x*) ∧ *ψ*2(*x*)。 由归纳假设可知(*M , s*0) |= Γ[*x/ϕ*]。

(2) 令Γ = EX*ψ*1(*x*)。

(*M , s*0) |= Γ ∧ 八 ∆

⇒ 存在(*s*0*, s*1) ∈ *R*，使得(*M , s*1) |= *ψ*1(*x*)和(*M , s*1) |= 八 ∆成立

⇒ 由归纳假设可知(*M , s*1) |= *ψ*1(*x*)[*x/ϕ*]

⇒ (*M , s*0) |= EX*ψ*1(*x*)[*x/ϕ*]

⇒ (*M , s*0) |= (EX*ψ*1(*x*))[*x/ϕ*]。

(3) 令Γ = AX*ψ*1(*x*)。

(*M , s*0) |= Γ ∧ 八 ∆

⇒ 对任意(*s*0*, s*1) ∈ *R*，有(*M , s*1) |= *ψ*1(*x*)和(*M , s*1) |= 八 ∆

⇒ 对任意(*s*0*, s*1) ∈ *R*，由归纳假设可知(*M , s*1) |= *ψ*1(*x*)[*x/ϕ*]

⇒ (*M , s*0) |= AX*ψ*1(*x*)[*x/ϕ*]

⇒ (*M , s*0) |= (AX*ψ*1(*x*))[*x/ϕ*]。

(4) 令Γ = E(*ψ*1(*x*)U*ψ*2(*x*))。

(*M , s*0) |= Γ ∧ 八 ∆

⇒ 存在一条路径*π* = (*s*0*, s*1*, . . .* ) ∈ *R*，使得对某个 *j* ≥ 0，有(*M , s j* ) |= *ψ*2(*x*)，对任意0 ≤

*i < j*，有(*M , si*) |= *ψ*1(*x*)，且对所有*x* ≥ 0，有(*M , sx*) |= 八 ∆

⇒ 由归纳假设可知，存在一条路径*π* = (*s*0*, s*1*, . . .* ) ∈ *R*，使得对某个 *j* ≥ 0，有(*M , s j* ) |=

*ψ*2(*x*)[*x/ϕ*]，且对任意0 ≤ *i < j*，有(*M , si*) |= *ψ*1(*x*)[*x/ϕ*]

⇒ (*M , s*0) |= E((*ψ*1(*x*)[*x/ϕ*]) U (*ψ*2(*x*)[*x/ϕ*]))

⇒ (*M , s*0) |= (E(*ψ*1(*x*)U*ψ*2(*x*)))[*x/ϕ*])。

可以类似证明其它情况。

(⇐) 对Γ[*x/ϕ*]的任意模型(*M , s*0)（*M* = (*S, R, L,* [ ]*, s*0)），构造一个初始Ind-Kripke 结构*M* ′ = (*S, R, L*′*,* [ ]*, s*0)，其中*L*′与*L*一样，除了：对任意*s*′ ∈ *S*′，若(*M* ′*, s*′) |= *ϕ*，则 令*L*′(*s*′) = *L*(*s*) ∪ {*x*}，否则令*L*′(*s*′) = *L*(*s*) − {*x*}（即(*M* ′*, s*0) |= *x* ↔ *ϕ*）。

显然易证，(*M , s*0) ↔{*x*} (*M* ′*, s*′ )且(*M* ′*, s*′ ) |= Γ ∪ ∆。

0 0

在这种情形下，记GAL(Γ ∪ ∆*,* {*x*}) = Γ[*x/* 八 ({*Ci* | 1 ≤ *i* ≤ *n*} ∪ {*Bi* | 1 ≤ *i* ≤ *m*})]。 对

于CTL公式集Σ，用GAL(Σ*,* {*x*})表示GAL(Γ ∪ ∆*,* {*x*})，其中∆ ⊆ Σ为与引理[5.7](#_bookmark117)中∆有相同

性质的出现在Σ中有唯一负出现*x*的公式集，Γ ⊆ Σ是*x*正出现在其中的公式集。 对于原

子命题集*V* ，定义

GAL(Σ*,V* ∪ {*x*}) = GAL(GAL(Σ*,* {*x*})*,V* )*.*

例 **5.5** (例[5.4](#_bookmark115)的延续)**.** 首先考虑原子命题*x*、∆ = {⊤ → *f* ∨ *m* ∨ ¬*x*}和Γ = *ERes*(*ϕ,V* ) − ∆。

Γ中包含*x*的公式关于*x*都为正的，因此Γ[*x/*( *f* ∨ *m*)]包含如下公式：

**start** → *z,* **start** → *f* ∨ *m* ∨ *q,* **start** → *f* ∨ *m* ∨ *y,*

算法 **5.3** CTL-forget(*ϕ,V* )

**Input**: A CTL formula *ϕ* and a set *V* of atoms

**Output**: A conjunction of formulas

**14 if** *ϕ* ≡ ⊥ **then return** ⊥ ; // 若公式不可满足，则遗忘结果为⊥

**15 if** *V* = *Var*(*ϕ*) **then return** ⊤ ; // 若遗忘所有原子命题，则结果为⊤

**16** *Tϕ* ← SNF*g*

CTL

(*ϕ*) ; // 将*ϕ*转换为SNF*g* 子句

**17** Σ ← UF(*Tϕ ,V* ∪ *U* ) where *U* = *Var*(*Tϕ* ) − *Var*(*ϕ*); // 展开 **18** Σ ← *ERes*(Σ*,V* ) ; // 移除包含*V* 中元素的子句 **19** Σ ← RM-index(Σ) ; // 从Σ移除索引

C T L

**20** Σ ← GAL(Σ*, Var*(Σ) − *Var*(*ϕ*)) ; // 移除留存的新的原子命题

**21** Replacing each initial clause “AG(**start** → *ϕ*)” in Σ by AG*ϕ* ; // 去除**start**

**22 return** Σ

|  |  |  |  |
| --- | --- | --- | --- |
| ⊤ → *q* ∨ *f* ∨ *m* ∨ ¬*z,* | ⊤ → *f* ∨ *m* ∨ *y* ∨ ¬*z,* | ⊤ → *q* ∨ ¬*y,* | *z* → AF( *f* ∨ *m*)*,* |
| *y* → AX(*q* ∨ *f* ∨ *m*)*,* | *y* → AX( *f* ∨ *m* ∨ *y*)*.* |  |  |

第二步考虑原子命题*z*、∆′ = {⊤ → *q* ∨ *f* ∨ *m* ∨¬*z,* ⊤ → *f* ∨ *m* ∨ *y* ∨¬*z, z* → AF( *f* ∨ *m*)} 和Γ′ = Γ[*x/*( *f* ∨ *m*)] −∆′，其中*z*正出现在Γ′中。因此，Γ′′ = Γ′[*z/*(*q* ∨ *f* ∨ *m*) ∧( *f* ∨ *m* ∨ *y*) ∧ AF( *f* ∨ *m*)]包含如下公式：

**start** → (*q* ∨ *f* ∨ *m*) ∧ ( *f* ∨ *m* ∨ *y*) ∧ AF( *f* ∨ *m*)*,* **start** → *f* ∨ *m* ∨ *q,* **start** → *f* ∨ *m* ∨ *y,*

⊤ → *q* ∨ ¬*y, y* → AX(*q* ∨ *f* ∨ *m*)*, y* → AX( *f* ∨ *m* ∨ *y*)*.*

不难证明*ERes*(*ϕ,V* ) ≡{*x,z*} Γ′′。 因为Γ′′包含一个公式其关于*y*既不是正的也不是负

的，因此，这里不能对Γ′′和*y*使用上述过程。

**5.2.1.3** 算法及其复杂性分析

现在可以给出计算CTL下遗忘的算法——算法[5.3](#_bookmark0)。 该算法的输入为一个CTL公

式*ϕ*和一个原子命题集，输出为一个与*ϕ*互模拟等价的CTL公式。

定理 **5.3.** 令*ϕ*为一个*CTL*公式、*V* ⊆ *A* 、Σ =*CTL-forget*(*ϕ,V* )且*U* = *Var*(Σ) − *Var*(*ϕ*)，则

1. Σ ≡*V* ∪*U ϕ*，
2. 若*U* = 0/ ，则Σ ≡ FCTL(*ϕ,V* )。

证明**.** (i) 这一结论直接来源于命题[3.2](#_bookmark55)和[5.5](#_bookmark114)，引理[5.6](#_bookmark116)和[5.7](#_bookmark117)。

(ii) 若*U* = 0/ ，则由(i)可知Σ ≡*V ϕ*。 又由于Σ 和FCTL(*ϕ,V* )都是*V* -无关的且都不包含

索引，因此，由FCTL(*ϕ,V* ) ≡*V ϕ*可知Σ ≡ FCTL(*ϕ,V* )。

例 **5.6** (例[5.5](#_bookmark118)的延续)**.** 容易看出CTL-forget(*ϕ,* {*p, r*})包含下面的公式

(*q* ∨ *f* ∨ *m*) ∧ ( *f* ∨ *m* ∨ *y*) ∧ AF( *f* ∨ *m*)*,* AG(⊤ → *q* ∨ ¬*y*)*,*

AG(*y* → AX(*q* ∨ *f* ∨ *m*))*,* AG(*y* → AX( *f* ∨ *m* ∨ *y*))*.*

尽管如此，有的CTL公式的遗忘结果总是存在的，如下面的结论所示。

命题 **5.6.** 给定*CTL*公式*ϕ*， 若*ϕ*满足下面约束， 则ERes(*ϕ,V* ) ≡ FCTL(*ϕ,V* )：（*1*）*ϕ*中 不包括操作符*PtT* （其中*Pt* ∈ {A*,* E}且*T* ∈ {U*,* G}）；（*2*）对 于任意原子命题*p* ∈ *V* ， 若*p*和¬*p*出现在同一时序算子的范围内。

证明**.** 不失一般性地， 假设*V* = {*p*}。 对任意上述所说形式的CTL公式*ϕ*， 假定*ϕ* = *ϕ*1 ∧ AXEF*ϕ*2， 其中*p* ̸∈ *Var*(*ϕ*1)且*ϕ*2是 一个包含子句*C*1 = ¬*p* ∨ *ψ*1和*C*2 = *p* ∨ *ψ*2的CNF (conjunctive normal form)公 式。 *ϕ*可以被转换为包含集合Π = {⊤ → ¬*x* ∨ *p* ∨ *ψ*1*,* ⊤ →

¬*x* ∨ ¬*p* ∨ *ψ*2}的子句集Σ，其中*x*为新引入的原子命题，*ψi*（*i* = 1*,* 2）为经典子句。除此

之外，Σ中不包含其它含有*p*的公式。

由归结过程可产生子句⊤ → ¬ ∨ *ψ*1 ∨ *ψ*2，由定理[5.7](#_bookmark117)可知，*x*可以被*ψ*1 ∨ *ψ*2替换。又 因为公式*ϕ*中不包含*PtT* 时序算子，因而不会引入嵌套原子命题（同时出现在→两边 的原子命题），此时对新引入的其余的原子命题都可使用定理[5.7](#_bookmark117)。 因此，由定理[5.3](#_bookmark120)可 知CTL-forget(*ϕ,V* ) ≡ FCTL(*ϕ,V* )。

已有结果表明，转换过程和归结过程会终止[[98](#_bookmark212)]。 此外，移除原子命题、 移除索 引、 替换*V* ′中的原子命题和转换到CTL过程都会终止，因此算法[5.3](#_bookmark0)会终止。 其具体的 时间和空间复杂性如下面的结论所示。

命题 **5.7.** 给定*CTL*公式*ϕ*和原子命题集*V* ⊆ *A* 。 算法[*5.3*](#_bookmark0)的时间和空间复杂性为*O*((*m* + 1)24(*n*+*n*′))，其中*n* = |*Var*(*ϕ*)|、 *n*′ = |*V* ′|为新引入的原子命题的个数、 *m*为引入的索引个 数。

证明**.** 由于转换过程在多项式时间内完成，移除原子命题、 移除索引、 转换到CTL过 程和替换*V* ′中的原子命题，最多都只需要扫描归结过程得到的子句集就能完成。 因此， 算法的复杂性主要依赖于归结过程。

对于给定的公式*ϕ*和*V* ，归结过程产生的子句个数为(*m* + 1)24(*n*+*n*′) + (*m* \* (*n* + *n*′) +

*n* + *n*′ + 1)22(*n*+*n*′)+1)。

在上述结论中值得注意的是：*m*的大小不会大于公式*ϕ*中出现的时序算子的个数。 因此，可以得出算法[5.3](#_bookmark0)的计算复杂性仅与出现在*ϕ*中的原子命题和时序算子的个数相 关。

* + 1. 基于**Prolog**的**CTL-forget**算法实现

基于Prolog的CTL-forget算法实现系统以CTL公式和原子命题集为输入，CTL公式

为输出。 其所识别的CTL公式的符号与第[二](#_bookmark18)章中CTL的语言符号对应关系如下：

∙ *xi*和其余小写字母开头的字符串构成原子命题集，其中*i* ≥ 0为自然数，且*xi*和*z*被

设定为只能是在如下描述的转换过程中引入的原子命题；

∙ “ *f alse*”和“*true*”分别与常量符号“⊥”和“⊤”对应；

∙ “*start*”与命题常量“**start**”对应；

∙ “&”、“∖*/*”、“−”和“− *>*”分别与联结符号“∧”、“∨”、“¬”和“→”对应；

∙ “˜”和“ˆ”分别与路径量词“A”和“E”对应；

∙ “@”、“\*”、“?”和“$”分别与时序操作符“G”、“X”、“F”和“U”对应。

例 **5.7.** 字符串(˜\*((−*y*1∖*/* − *y*2∖*/* − *y*4)&(−*y*1∖*/y*2∖*/y*4)&(*y*1∖*/y*2∖*/* − *y*3)&(*y*1∖*/y*3∖*/* − *y*4)&(−*y*1∖*/y*2∖*/* − *y*3)))为CTL 公式。

此系统主要包括五个模块：转换模块（transCTL2SNF/6）、 归结模块（两个过程： step resolution/3和temp resolution/8）、“移除”原子命题模块（removeAtom/3），“移除” 索引（pro6/3）和“移除”新引入的原子命题（ackerM/3）。 下面就这几个模块做详细 的介绍。

转换模块：用二元谓词transCTL2SNF/6来实现将CTL公式转换为SNF*g*

CTL

子句的过

程。 在使用该谓词前，谓词ctl2NNF/2将输入的CTL公式*ϕ*转换其NNF形式并使用表

[2.2](#_bookmark30)中的等式化简公式等到公式*simp*(*nn f* (*ϕ*))，并产生如下公式列表（list）：

[*start*− *> z, z*− *> simp*(*nn f* (*ϕ*))]*.*

transCTL2SNF/6谓词通过迭代表 [2.1](#_bookmark29)中的每条规则将CTL公式转换为SNF*g*

CTL

子句集，每

一个子句用一个四元谓词snf clause(Parm1,Parm2,Parm3,Parm4)表示，其中：

∙ Parm1表示子句类型，即第 [2.3.2](#_bookmark28)节中的六种字句类型，用“start”、“true”、“ef”、 “af”、“ex”和“ax”分别表示初始子句、 全局子句、 A-某时子句、 E-某时子句、 A-步子句和E-步子句；

∙ Parm2为原子命题构成的列表，表示SNF*g*

CTL

∙ Parm3为原子命题构成的列表，表示SNF*g*

CTL

子句的头； 子句的尾；

∙ Parm4表示索引号，且如果Parm4=*nil*，则该子句不是带索引的子句。

在transCTL2SNF(Parm1,Parm2,Parm3,Parm4,Parm5,Parm6)中：参数Parm1为公式形

成的列表，即：[*start*− *> z, z*− *> simp*(*nn f* (*ϕ*))]；Parm2和Parm3初始化为0，分别表示

索引号和新引入的原子命题下；Parm4和Parm5分别表示转换过程结束后返回的索引号

和原子命题下标的最大值；Parm6为经过转换Parm1后得到的SNF*g*

CTL

子句集。

归结模块：主要有两个过程step resolution/3和temp resolution/8。step resolution(List1, V, List2)从List1中选择能使用表 [2.3](#_bookmark35)中的归结规则对*V* 中的元素做规约公式，并将得到

结果存在列表List2中。 这一过程循环执行， 直到不再产生新的SNF*g*

CTL

子句为 止。

temp resolution/8过程主要用于寻找归结规则**ERES1**和**ERES2**的前提（一组SNF*g* 子句

CTL

集，参照第 [2.4](#_bookmark34)节）。

“移除”原子命题模块：主要用于将含有要遗忘的原子命题的SNF*g*

CTL

子句去掉，

这一过程比较简单，这里不详细介绍。

“移除”索引模块过程使用引理 [5.6](#_bookmark116)中的互模拟等价公式将SNF*g*

CTL

子句中的索引去

掉，因为CTL公式中不包含索引，所以这一过程是必须的。

“移除”新引入的原子命题使用一般化的Ackermann引理（引理 [5.7](#_bookmark117)）将转换模块

中引入的新原子命题去除。

**5.2.3** 实验

本节给出所提出的基于归结的遗忘计算的实验结果，并分析实验结果。 第[四](#_bookmark82)章 提出的算法[5.3](#_bookmark0)用Prolog语言实现， 并在Linux服务器上进行了实验， 该服务器是具 有8个Intel核和32GB内存的i7CPU，其锁频和主频分别为4770 K和3.50 GHz。 每次计算 的时间限制在1200秒以内。 实验分为两个部分：(1) 在随机数据集和标准数据集上的遗 忘；(2) 在随机数据集上命题逻辑公式和CTL公式的SNC计算。 所有实验数据和实验结 果都可以从网上获取[4](#_bookmark0)。

此外，在这部分3-CNF公式*ϕ*的长度（记为|*ϕ*|）表示*ϕ*中子句的个数。

**5.2.3.1** 遗忘实验分析

这部分的实验数据分为两组：一组来源于标准数据集，令一组是随机生成的数据。 标准数据集来源于CTL-RP [5](#_bookmark0)。 但是由于数据集里的大部分公式是不可满足的，这种情 形下遗忘的结果总是为⊥。 因此，这里对数据集进行了简单的处理：从标准数据集里 抽取了“sample01”文件中的s001.ctl、 s002.ctl、 s003.ctl和s004.ctl文件，并从这些公式 里取前面的两个子公式的合取构成新的公式，分别称为s001、 s002、 s003和s004。 此 外，从s001.ctl中取前三个子公式的合取构成新的公式s001-3。

计算CTL-forget(*ϕ,V* )所使用的CPU时间（单位：秒(s)，不指出时也默认为秒）如 表[5.1](#_bookmark123)所示，其中*ϕ* ∈ {*s*001*, s*002*, s*003*, s*004*, s*001-3}，|*V* | ∈ {1*,* 2*,* 3*,* 4}。 从中可以看出公 式长度越长、被遗忘的原子命题个数越多，则计算所需要的时间越长。

4<https://github.com/fengrenyan/forgetting-in-CTL/tree/main/Appendix> 5<https://sourceforge.net/projects/ctlrp/>

第五章 CTL遗忘计算方法

表 5.1: 计算CTL-forget(*ϕ,V* )所使用的CPU时间（单位：秒(s)）

|*V* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ϕ* | 1 | 2 | 3 | 4 |
| s001 | 0.0505 | 0.1053 | 0.2259 | 0.3680 |
| s002 | 0.3645 | 1.0416 | 5.6372 | 10.0184 |
| s003 | 97.5341 | 71.5396 | 190.1157 | 423.5793 |
| s004 | 77.5086 | 77.4246 | 101.1284 | 118.7461 |
| s001-3 | 681.2883 | 613.1859 | 1617.047 | 2356.949 |

除了上述标准数据集中的公式，我们也做了具有以下形式的公式的遗忘实验:

*ϕ* = *ϕ*1 ∧ AX*ϕ*2 ∧ EX*ϕ*3

其中*ϕi*（*i* = 1*,* 2*,* 3）是随机产生的定义在原子命题集*A* 上的3-CNF公式，且|*ϕ*1| = |*ϕ*2| =

|*ϕ*3|、|*A* | = 4。这里做了六组计算FCTL (*ϕ,V* )的实验，即|*ϕi*| ∈ {12*,* 16}和|*V* | ∈ {1*,* 2*,* 3}的

组合，每一组有二十个公式。

|*ϕi*| = 12时的实验结果如图[5.3](#_bookmark124)所示， 图[5.3](#_bookmark124)(a) 展示了计算遗忘所需要的时间，

图[5.3](#_bookmark124)(b)展示了在计算过程“移除原子命题”后SNF*g*

CTL

子句的个数。 其中*x*轴表示第

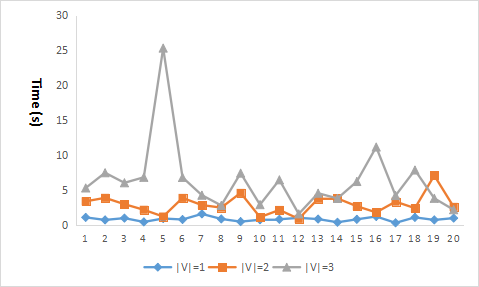
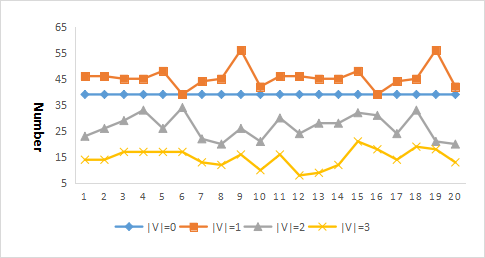
几个公式，*y*轴分别表示时间和数量。 从图[5.3](#_bookmark124)里面可以看出，需要遗忘的原子命题

个数越多，所用时间越长且在“移除原子命题”后剩余的SNF*g*

CTL

子句的个数越少。

当|*ϕi*| = 16时的实验结果如图[5.4](#_bookmark125)所示，其与|*ϕi*| = 12时有相似的结果。

(a) 计算遗忘需要的CUP时间 (b) SNF*g*

CTL

子句的个数

图 5.3: 计算CTL-forget(*ϕ,V* )使用的时间和在“移除原子命题”步骤后SNF*g*

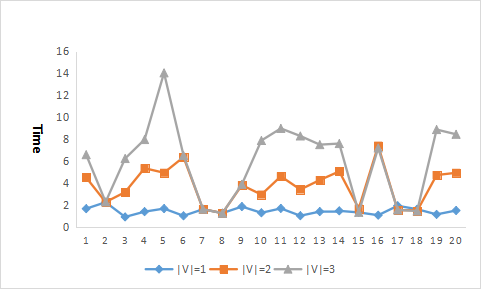
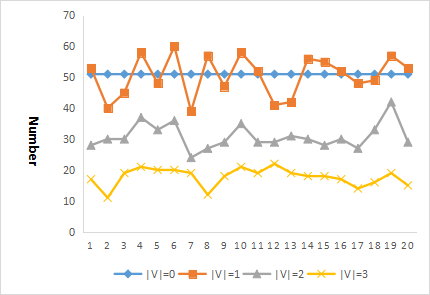
CTL

子句的个

数，其中*ϕi* = 12。

**5.2.3.2 SNC**计算结果分析

这部分实验分析使用基于遗忘的方法计算CTL公式的SNC，分为两组实验：计算 经典命题公式和CTL公式的SNC，即：计算*q*在*V* 和*ϕ* ∧ *q*上的SNC（FCTL(*ϕ* ∧ *q, Var*(*ϕ*) − *V* ∪ {*q*})），其中*V* ⊆ *Var*(*ϕ*)、 *q* ∈ *Var*(*ϕ* ∧ *q*) − *V*。 这些公式*ϕ*都是随机生成的定义在原

(a) 计算遗忘需要的CUP时间 (b) SNF*g*

CTL

子句的个数

图 5.4: 计算CTL-forget(*ϕ,V* )使用的时间和在“移除原子命题”步骤后SNF*g*

CTL

子句的个

数，其中*ϕi* = 16。

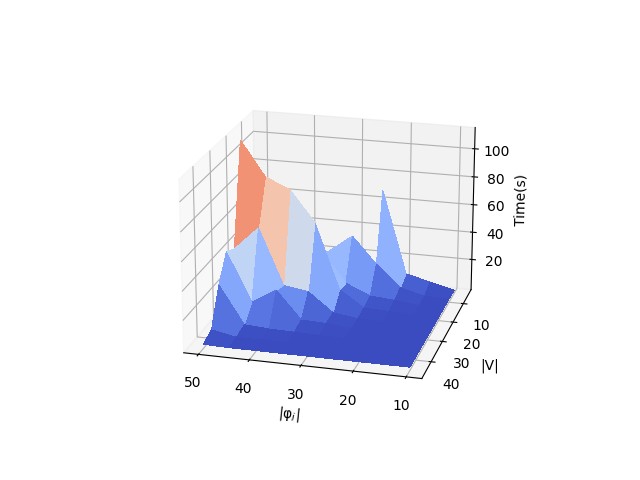
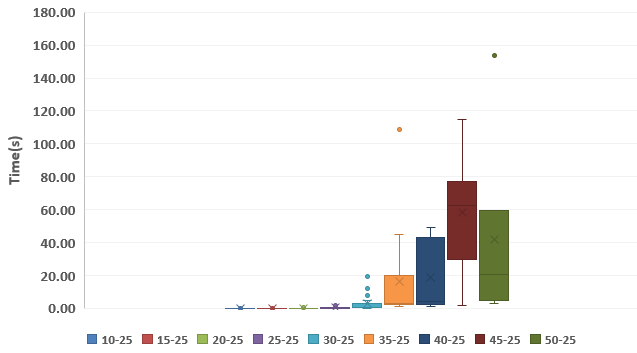
子命题集*A* 上的公式、*V* 也是在计算过程中随机生成的、 *q* ̸∈ *Var*(*ϕ*)是一个固定的原子

命题且|*A* | = 50。

首先测试随机3-CNF命题公式。 令|*V* |的取值范围为{5*,* 10*,...,* 40*,* 45}，3-CNF公式

的子句个数*nc*范围为{10*,* 15*,...,* 45*,* 50}。 在每种情形当中，计算20个随机实例(*ϕ, q,V* )：

*ϕ*为*A* 上的公式，且*V* ⊆ *Var*(*ϕ*)。 计算SNC的平均CPU时间如图[5.5](#_bookmark126)所示。

(a) 平均CPU时间(s) (b) |*V* | = 25时所使用CPU时间箱线图

图 5.5: 计算3-CNF公式SNC的CPU时间

从图[5.5](#_bookmark126)(a)可看出，随着|*ϕ*|增大或|*V* |的减小时间消耗越大。 直观上，若|*ϕ*|越大 或者|*V* |越小，则FCTL(*ϕ,V* )越难计算。 这与上一小节中的结论相符合。 图[5.5](#_bookmark126)(b)展示 了当|*V* | = 25、 *nc* ∈ {10*,* 15*,...,* 45*,* 50}时20个随机实例的箱线图。 这同样证明了*nc*越 大SNC越难计算。

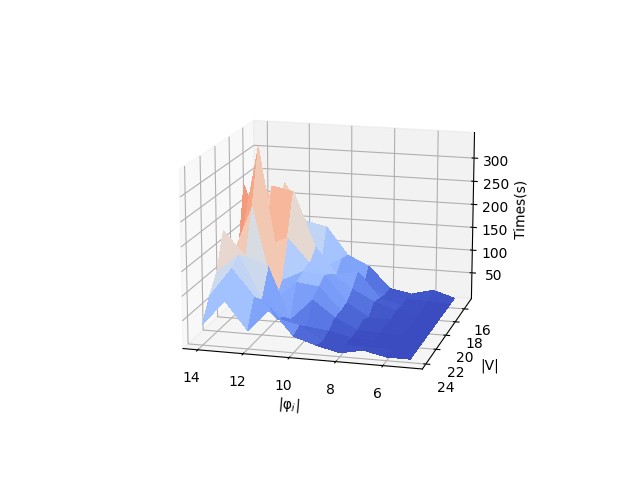
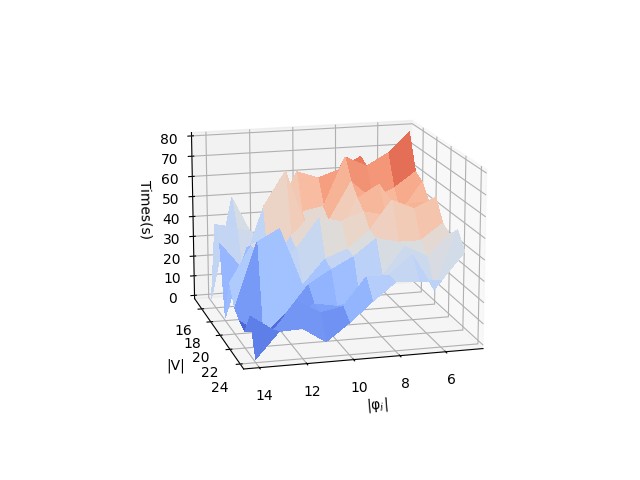
其次，测试具有如下形式的CTL公式的SNC的计算：

*ϕ*1 ∧ AX*ϕ*2 ∧ EX*ϕ*3

其 中*ϕi* （*i* = 1*,* 2*,* 3） 为 随 机 产 生 的 定 义 在*A* 上 的3-CNF公 式， 且 满 足|*ϕ*1| = |*ϕ*2| =

|*ϕ*3|。 在这种情形下， 每个实例(*ϕ, q,V* )是随机产生的， 其中*ϕ* = *ϕ*1 ∧ AX*ϕ*2 ∧ EX*ϕ*3、 *V* ⊆ *Var*(*ϕ*)、 |*ϕ*| ∈ {5*,* 6*, . . . ,* 13*,* 14}、 且|*V* | ∈ {15*,* 16*,...,* 23*,* 24}。 值 得 注 意 的 是 在 实 例(*ϕ, q,V* )中，*q*可能没有在*V* 和*ϕ* ∧ *q*上的SNC。

图[5.6](#_bookmark127)(a)展示了每种情形计算40个实例的SNC的平均CPU时间。 与命题公式的情形 相似，若|*ϕ*|越大或者|*V* |越小，则FCTL(*ϕ,V* )越难计算。 此外，图[5.6](#_bookmark127)(b)展示了每种情形 下40个实例中SNC存在的占比，即：|*ϕ*|越小或者|*V* |越小则SNC存在的占比越大。 特别 地，当|*ϕi*| = 5且|*V* | = 16时，SNC存在的占比为80%。

(a) 平均CPU时间(s) (b) 存在SNC的公式占比(%)

图 5.6: 计算CTLSNC的平均时间和存在SNC的公式占比.

综上所述，算法[5.3](#_bookmark0)大多数情况下能计算出SNC（WSC），且当需要遗忘的原子个

数很少或公式长度较小的时候计算效率很高。

**5.3** 本章小结

本章第 [5.1](#_bookmark95)讨论了一种基于模型的计算约束CTL下遗忘的计算。 为此，本章第一节 首先提出了一种有界*V* -互模拟概念，并证明了该有界*V* -互模拟与*V* -互模拟在有限结构 下是等价的。 此外，定义了给定深度的计算树在给定原子命题集上的特征公式，由此 定义了有限结构的特征公式。 结论表明初始结构能够满足给定的特征公式当且仅当该 初始结构与特征公式对应的初始结构在给定原子命题集上互模拟。 基于此，得出了任 意公式语义等价于其所有模型的特征公式的析取，因而可以计算遗忘的结果。 最后， 本章给出了基于模型的计算遗忘算法，并分析该算法关于公式的大小是指数空间的。

本章第 [5.2](#_bookmark111)探索了如何使用Zhang等人提出的归结系统计算CTL下的遗忘。 为此，

首先使用Zhang提出的转换规则将CTL公式转换成SNF*g*

CTL

子句集，这一步骤在第[二](#_bookmark18)章讲

述CTL的标准形式时已经给出。 基于此，使用归结规则在要遗忘的原子命题集上做穷 尽的归结。 然后给出几个互模拟等价的等式消除索引，并提出一般化的Ackermann引 理用于移除一些新引入的原子命题。 最后，给出计算遗忘的算法，并分析该算法的复 杂性，分析表明使用该算法计算遗忘的时间和空间复杂性关于公式和要遗忘的原子命 题的个数是指数时间的。

随后，第 [5.2.2](#_bookmark121)节给出了基于Proplog的算法CTL-forget实现介绍。 第 [5.2.3](#_bookmark122)节从两个 角度来评估算法CTL-forget的实现系统。 实验结果表明，从标准数据集中获取的公式和 随机产生的公式下遗忘的计算对需要遗忘的原子命题个数和公式长度都很敏感：要遗 忘的原子命题越多或公式越长，计算效率越低。 此外，关于计算SNC的实验表明，本 文提出的算法在用于实验的数据集中大部分情况下是能计算出SNC的。 三维图和箱线 图都表示计算SNC体现了跟计算遗忘一样的规律，这与SNC是由遗忘来计算是一致的。

# 第六章 总结与展望

本章首先总结文中针对*CTL*和*µ-*演算下的遗忘理论做出的研究工作，概括文中使 用的方法及取得的研究成果。 其次，讨论分析本文工作中存在的不足之处，并基于此 对本文的后续研究内容进行了展望。

**6.1** 工作总结

随着计算机系统日益变得复杂，描述系统规范的语言也变得越来越复杂。 随着信 息的更新，系统的规范也随着改变，因而急需有效的方法来提取相关原子命题下的知 识。 遗忘是一种知识提取的方法，本文从语法和语义的角度探索了广泛应用于并行系 统的CTL下的遗忘。 此外，也探索了表达能力更强的*µ*-演算下的遗忘。

本文以使用遗忘计算反应式系统在给定条件下的WSC（SNC）和知识更新为主线，

解决了第一章提出的问题，并取得了以下成果：

(1) 本文通过互模拟定义了CTL遗忘，并给出CTL遗忘具有分解性、 切片性和同质

性等基本属性。 此外，提出了一种基于归结的算法计算CTL遗忘。 为此，提出一般化

的ACkermann引理和消除索引的方法，用于将归结得到的SNF*g*

CTL

子句转换为CTL公式。

最后，使用Prolog实现了该算法，并做了相关实验评估了该系统。

特别地，本文证明了CTL遗忘不总是存在的。 因此，探索了约束CTL遗忘。 在这 种情形中，构成CTL公式原子命题是有限的，公式长度为*n*且公式模型是有限的。 提出 一种有限互模拟，并证明这种有限互模拟与上述的互模拟在有限结构下是等价的。 本 文证明，有限结构可以用一个CTL公式表达。 从而证明任意公式的遗忘是存在的，即： 与该公式的模型互模拟的所有结构特征公式的吸取。

(2) *µ*-演算是一种表达能力强的逻辑语言。 本文给出了*µ*-演算遗忘的定义，并证 明了其满足基本的遗忘性质。 为此，本文提出了一种新的互模拟，证明了*µ*-公式对这 一互模拟是不变的，即：若两个结构是*V* -互模拟的（*V* 是原子命题集），则对于任意 与*V* 无关的公式，这两个结构同时满足（或不满足）该公式。

此外，本文探索了两个遗忘存在的子类：*µ*-句子和X-类。

(3) 使用遗忘计算SNC（WSC）和定义知识更新。 对于给定的公式和原子命题集，

若遗忘除这些原子命题之外的原子命题结果可用CTL公式表示，则该结果一定是SNC

（WSC），即：CTL下可以用遗忘来计算SNC（WSC）。在约束情形下SNC（WSC）一定 可以用遗忘方法来计算，因为这种情形下遗忘结果总是存在的。 此外，当给定的反应 式系统为有限系统的情形下，可以将该系统表示成特征公式，然后再使用遗忘来计算。

贵州大学博士学位论文

最后，提出了两种定义知识更新的方法：基于遗忘的定义和基于模型间偏序关系的定 义，并证明这两种方法定义的知识更新是等价的且满足Katsuno等人提出的八条基本准 则。

(4) 实现了基于归结的算法，并做了实验，得到“遗忘的原子命题个数越少，计算

效率越高”的结论。

**6.2** 研究展望

本文探讨了对系统设计至关重要的抽取信息的方法——遗忘，并使用该方法计 算SNC（WSC）和定义知识更新。 文中指出CTL下的遗忘不是封闭的，并提出了基于 归结的计算遗忘的方法。 下面问题值得将来继续研究：

(1) 探索CTL中遗忘封闭的子类。 在系统规范描述中，有时候用到的公式不一定 很复杂，也不一定需要用到所有的时序词。 为此，探索简单并足够表达某些性质 的CTL公式的子类，在这些子类下遗忘是容易计算的。

(2) 探索如何使用计算出来的WSC（SNC）更新（修改）系统模型。 特别地，当一 个系统*M* 不满足规范*φ* 时，可以计算在某个命题集合*V* 上的最弱充分条件*ψ*使得*M* 满 足，即*M* |= *ψ* → *φ* 且*ψ*是*V* 上的公式。 这时，如何使用获得的WSC来更新系统得到新 系统*M* ′使其满足*φ* 也是重要的。

(3) 尽管CTLAF段下遗忘的模型检测和推理判定问题的复杂性已经给出，但是更多 关于遗忘的问题的复杂性应该被探索：整个CTL下的遗忘的模型检测和推理判定问题 的复杂性。

(4) 研究算法 [5.3](#_bookmark0)的完备性。 尽管在第[五](#_bookmark94)给出了算法正确性的分析，但是并没有讨 论算法的完备性。 在将来的工作中可以研究算法 [5.3](#_bookmark0)的完备性，并探索其他可靠且完备 的算法。

# 参考文献

[1] KATSUNO H, MENDELZON A O. On the difference between updating a knowledge base and revising it[C]//ALLEN J F, FIKES R, SANDEWALL E. Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR’91). Cambridge, MA, USA, April 22-25, 1991. [S.l.]: Morgan Kaufmann, 1991: 387-394.

[2] LAM W K. 硬件设计验证–基于模拟与形式的方法[M]. 北京: 机械工业出版社, 2007.

[3] 吕毅. 时序逻辑电路的形式验证方法研究[D]. 北京: 中国科学院研究生院(计算技

术研究所), 2000.

[4] JANICK B夏宇闻. System Verilog验证方法学[M]. 北京: 北京航空航天大学出版社

学, 2007.

[5] 袁志斌. 软件开发的形式化工程方法[M]. 北京: 清华大学出版社, 2008.

[6] 古天龙. 软件开发的形式化方法[M]. 北京: 高等教育出版社, 2005.

[7] FOX A C J. Formal specification and verification of ARM6[C/OL]//BASIN D A, WOLFF B. Lecture Notes in Computer Science: volume 2758 Theorem Proving in Higher Order Logics, 16th International Conference, TPHOLs 2003, Rom, Italy, September 8-12, 2003, Proceedings. Springer, 2003: 25-40. [https://doi.org/10.1007/](https://doi.org/10.1007/10930755_2)

[10930755 2](https://doi.org/10.1007/10930755_2).

[8] DAUM M, SCHIRMER N, SCHMIDT M. Implementation correctness of a real-time operating system[C/OL]//HUNG D V, KRISHNAN P. Seventh IEEE International Con- ference on Software Engineering and Formal Methods, SEFM 2009, Hanoi, Vietnam, 23-27 November 2009. IEEE Computer Society, 2009: 23-32. [https://doi.org/10.1109/](https://doi.org/10.1109/SEFM.2009.14) [SEFM.2009.14](https://doi.org/10.1109/SEFM.2009.14).

[9] ROBINSON J A. A machine-oriented logic based on the resolution principle[J/OL]. Journal of the ACM (JACM), 1965, 12(1):23-41. [http://doi.acm.org/10.1145/321250.3](http://doi.acm.org/10.1145/321250.321253) [21253](http://doi.acm.org/10.1145/321250.321253).

[10] HUGHES G E, CRESSWELL M J, CRESSWELL M M. A new introduction to modal logic[M]. [S.l.]: Psychology Press, 1996.

[11] HOARE C A R. An axiomatic basis for computer programming[J/OL]. Commun. ACM, 1969, 12(10):576-580. <https://doi.org/10.1145/363235.363259>.

[12] HAREL D. Lecture notes in computer science: volume 68 first-order dynamic logic [M/OL]. Springer, 1979. <https://doi.org/10.1007/3-540-09237-4>.

[13] REYNOLDS J C. Separation logic: A logic for shared mutable data structures[C/OL]// 17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings. IEEE Computer Society, 2002: 55-74. [https:](https://doi.org/10.1109/LICS.2002.1029817)

[//doi.org/10.1109/LICS.2002.1029817](https://doi.org/10.1109/LICS.2002.1029817).

[14] LIN F. A formalization of programs in first-order logic with a discrete linear order [J/OL]. Artificial Intelligence, 2016, 235:1-25. [https://doi.org/10.1016/j.artint.2016.01](https://doi.org/10.1016/j.artint.2016.01.014)

[.014](https://doi.org/10.1016/j.artint.2016.01.014).

[15] CLARKE E M. The birth of model checking[C/OL]//GRUMBERG O, VEITH H. Lec- ture Notes in Computer Science: volume 5000 25 Years of Model Checking - History, Achievements, Perspectives. Springer, 2008: 1-26. [https://doi.org/10.1007/978-3-540-](https://doi.org/10.1007/978-3-540-69850-0_1)

[69850-0 1](https://doi.org/10.1007/978-3-540-69850-0_1).

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

[17] BIERE A, CIMATTI A, CLARKE E M, et al. Bounded model checking[J/OL]. Adv.

Comput., 2003, 58:117-148. <https://doi.org/10.1016/S0065-2458(03)58003-2>.

[18] 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.

[19] SCHNEIDER K. Texts in theoretical computer science. an EATCS series: Verification of reactive systems - formal methods and algorithms[M/OL]. Springer, 2004. [https:](https://doi.org/10.1007/978-3-662-10778-2)

[//doi.org/10.1007/978-3-662-10778-2](https://doi.org/10.1007/978-3-662-10778-2).

[20] ZHU J, WANG H, XU Z, et al. A new model for model checking: cycle-weighted kripke structure[J/OL]. Frontiers Comput. Sci. China, 2010, 4(1):78-88. [https://doi.org/10.100](https://doi.org/10.1007/s11704-009-0066-7) [7/s11704-009-0066-7](https://doi.org/10.1007/s11704-009-0066-7).

[21] DIJKSTRA E W. Guarded commands, Nondeterminacy and Formal Derivation of Pro- grams[J/OL]. Commun. ACM, 1975, 18(8):453-457. [https://doi.org/10.1145/360933.3](https://doi.org/10.1145/360933.360975)

[60975](https://doi.org/10.1145/360933.360975).

[22] LIN F. On strongest necessary and weakest sufficient conditions[J/OL]. Artificial Intel- ligence, 2001, 128(1-2):143-159. <https://doi.org/10.1016/S0004-3702(01)00070-4>.

[23] LIN F. Compiling causal theories to successor state axioms and strips-like systems [J/OL]. Journal of Artificial Intelligence Research, 2003, 19:279-314. [https://doi.org/](https://doi.org/10.1613/jair.1135) [10.1613/jair.1135](https://doi.org/10.1613/jair.1135).

[24] BAIER C, KATOEN J. Principles of model checking[M]. [S.l.]: MIT Press, 2008.

[25] LEGATO W J. A weakest precondition model for assembly language programs[M]. [S.l.]: April, 2002.

[26] LEINO K R M. Efficient weakest preconditions[J/OL]. Information Processing Letters, 2005, 93(6):281-288. <https://doi.org/10.1016/j.ipl.2004.10.015>.

[27] DAILLER S, HAUZAR D, MARCHE´

C, et al. Instrumenting a weakest precondition

calculus for counterexample generation[J/OL]. J. Log. Algebraic Methods Program., 2018, 99:97-113. <https://doi.org/10.1016/j.jlamp.2018.05.003>.

[28] WOODCOCK J, MORGAN C. Refinement of state-based concurrent systems[C/OL]// BJØRNER D, HOARE C A R, LANGMAACK H. Lecture Notes in Computer Sci- ence: volume 428 VDM ’90, VDM and Z - Formal Methods in Software Develop- ment, Third International Symposium of VDM Europe, Kiel, FRG, April 17-21, 1990, Proceedings. 1990: 340-351. [https://doi.org/10.1007/3-540-52513-0 18](https://doi.org/10.1007/3-540-52513-0_18).

[29] 简云松. 一个目标驱动的运行时需求管理框架的研究与实现[D]. 北京: 清华大学, 2012.

[30] HAREL D, PNUELI A. On the development of reactive systems[C/OL]//APT K R. NATO ASI Series: volume 13 Logics and Models of Concurrent Systems - Conference proceedings, Colle-sur-Loup (near Nice), France, 8-19 October 1984. Springer, 1984: 477-498. [https://doi.org/10.1007/978-3-642-82453-1 17](https://doi.org/10.1007/978-3-642-82453-1_17).

[31] PNUELI A. Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends[M/OL]//DE BAKKER J W, DE ROEV- ER W P, ROZENBERG G. Lecture Notes in Computer Science: volume 224 Cur-

rent Trends in Concurrency, Overviews and Tutorials. Springer, 1986: 510-584. <https://doi.org/10.1007/BFb0027047>.

[32] 王娟娟, 乔颖, 熊金泉, 等. 多核环境下基于图模型的实时规则调度方法[J]. Journal of Software, 2019, 30(2).

[33] CLARKE E M, GRUMBERG O, PELED D A. Model checking[M/OL]. MIT Press,

2001. <http://books.google.de/books?id=Nmc4wEaLXFEC>.

[34] CLARKE E, GRUMBERG O, LONG D. Model checking[C]//NATO ASI DPD. [S.l.: s.n.], 1996: 305-349.

[35] 王政. 嵌入式周期控制系统的建模与分析[D]. 上海: 华东师范大学, 2012.

[36] 李书浩, 王戟, 董威, 等. 反应式系统面向性质测试的方法框架[J]. 电子学报, 2004, 32(S1):226-230.

[37] LIN F, REITER R. Forget it[C]//Working Notes of AAAI Fall Symposium on Relevance. [S.l.: s.n.], 1994: 154-159.

[38] VISSER A. Uniform interpolation and layered bisimulation[M]//Go¨del’96: Logical foundations of mathematics, computer science and physics—Kurt Go¨del’s legacy, Brno, Czech Republic, August 1996, proceedings. [S.l.]: Association for Symbolic Logic, 1996: 139-164.

[39] KONEV B, WALTHER D, WOLTER F. Forgetting and uniform interpolation in large- scale description logic terminologies[C/OL]//BOUTILIER C. IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, Califor- nia, USA, July 11-17, 2009. 2009: 830-835. [http://ijcai.org/Proceedings/09/Papers/14](http://ijcai.org/Proceedings/09/Papers/142.pdf) [2.pdf](http://ijcai.org/Proceedings/09/Papers/142.pdf).

[40] ACKERMANN W. Untersuchungen u¨ber das eliminationsproblem der mathematischen logik[J]. Mathematische Annalen, 1935, 110(1):390-413.

[41] KONEV B, LUTZ C, WALTHER D, et al. Model-theoretic inseparability and modu- larity of description logic ontologies[J/OL]. Artificial Intelligence, 2013, 203:66-103. <https://doi.org/10.1016/j.artint.2013.07.004>.

[42] WANG Z, WANG K, TOPOR R W, et al. Forgetting for knowledge bases in DL-Lite[J].

Annuals of Mathematics and Artificial Intelligence, 2010, 58(1-2):117-151.

[43] LUTZ C, WOLTER F. Foundations for uniform interpolation and forgetting in expres- sive description logics[C/OL]//WALSH T. IJCAI 2011, Proceedings of the 22nd Inter- national Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. IJCAI/AAAI, 2011: 989-995. [https://doi.org/10.5591/978-1-57735-516-](https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170)

[8/IJCAI11-170](https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-170).

[44] KONEV B, LUDWIG M, WALTHER D, et al. The logical difference for the lightweight description logic *E L* [J]. Journal of Artificial Intelligence Research, 2012, 44:633-708.

[45] ZHAO Y, SCHMIDT R A. Role forgetting for ALCOQH(*δ* )-ontologies using an ackermann-based approach[C/OL]//SIERRA C. Proceedings of the Twenty-Sixth Inter- national Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017. ijcai.org, 2017: 1354-1361. [https://doi.org/10.24963/ijcai.2017/18](https://doi.org/10.24963/ijcai.2017/188)

[8](https://doi.org/10.24963/ijcai.2017/188).

[46] ZHAO Y, SCHMIDT R A, WANG Y, et al. A practical approach to forgetting in de- scription logics with nominals[C/OL]//The Thirty-Fourth AAAI Conference on Artifi- cial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Ad- vances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. AAAI Press, 2020: 3073-3079. <https://aaai.org/ojs/index.php/AAAI/article/view/5702>.

[47] GABBAY D M, SCHMIDT R A, SZALAS A. Studies in logic : Mathematical log- ic and foundations: volume 12 second-order quantifier elimination - foundations, computational aspects and applications[M/OL]. College Publications, 2008. [http:](http://collegepublications.co.uk/logic/mlf/?00009)

[//collegepublications.co.uk/logic/mlf/?00009](http://collegepublications.co.uk/logic/mlf/?00009).

[48] ZHANG Y, ZHOU Y. Knowledge forgetting: Properties and applications[J]. Artificial Intelligence, 2009, 173(16-17):1525-1537.

[49] ZHANG Y, ZHOU Y. Properties of knowledge forgetting[C]//PAGNUCCO M, THIELSCHER M. Proceedings of NMR 2008. Sydney, Australia: [s.n.], 2008: 68- 75.

[50] 文习明. 信息不完备下的知识遗忘[J]. 现代计算机(专业版), 2019, 11:8-13.

[51] LIU Y, WEN X. On the progression of knowledge in the situation calculus[C]//IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. Barcelona, Catalonia, Spain: IJCAI/AAAI, 2011: 976-982.

[52] FANG L, LIU Y, VAN DITMARSCH H. Forgetting in multi-agent modal logics[J/OL].

Artificial Intelligence, 2019, 266:51-80. <https://doi.org/10.1016/j.artint.2018.08.003>.

[53] 文习明, 方良达, 余泉, 等. 多智能体模态逻辑系统*KD*45*n* 中的知识遗忘[J]. 计算机

科学, 2019, 46(7):195-205.

[54] 文习明, 方良达, 余泉, 等. 多智能体模态逻辑系统*Kn*中的知识遗忘[J]. 逻辑学研究, 2019, 02(12):43-62.

[55] IEMHOFF R. Uniform interpolation and sequent calculi in modal logic[J/OL]. Archive for Mathematical Logic, 2019, 58(1-2):155-181. [https://doi.org/10.1007/s00153-018-0](https://doi.org/10.1007/s00153-018-0629-0)

[629-0](https://doi.org/10.1007/s00153-018-0629-0).

[56] FINE K. Failures of the interpolation lemma in quantified modal logic[J/OL]. The Journal of Symbolic Logic, 1979, 44(2):201-206. <https://doi.org/10.2307/2273727>.

[57] SCHUMM G F. Some failures of interpolation in modal logic[J/OL]. Notre Dame journal of formal logic, 1986, 27(1):108-110. <https://doi.org/10.1305/ndjfl/1093636529>.

[58] ZHANG Y, FOO N Y. Solving logic program conflict through strong and weak forget- tings[J]. Artificial Intelligence, 2006, 170(8-9):739-778.

[59] EITER T, WANG K. Semantic forgetting in answer set programming[J/OL]. Artificial Intelligence, 2008, 172(14):1644-1672. <https://doi.org/10.1016/j.artint.2008.05.002>.

[60] WONG K S. Forgetting in logic programs[D]. [S.l.]: The University of New South Wales, 2009.

[61] WANG Y, ZHANG Y, ZHOU Y, et al. Knowledge forgetting in answer set programming [J/OL]. Journal of Artificial Intelligence Research, 2014, 50:31-70. [https://doi.org/10.1](https://doi.org/10.1613/jair.4297) [613/jair.4297](https://doi.org/10.1613/jair.4297).

[62] WANG Y, WANG K, ZHANG M. Forgetting for answer set programs revisited [C/OL]//ROSSI F. IJCAI 2013, Proceedings of the 23rd International Joint Confer- ence on Artificial Intelligence, Beijing, China, August 3-9, 2013. 2013: 1162-1168. <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6807>.

[63] WANG Y, WANG K, WANG Z, et al. Knowledge forgetting in circumscription: A preliminary report[C/OL]//BONET B, KOENIG S. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.

AAAI Press, 2015: 1649-1655. [http://www.aaai.org/ocs/index.php/AAAI/AAAI15/pap](http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9866) [er/view/9866](http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9866).

[64] DELGRANDE J P. A knowledge level account of forgetting[J/OL]. Journal of Artificial Intelligence Research, 2017, 60:1165-1213. <https://doi.org/10.1613/jair.5530>.

[65] GONC¸ ALVES R, KNORR M, LEITE J, et al. On the limits of forgetting in answer set programming[J]. Artificial Intelligence, 2020, 286(0):103307.

[66] EITER T, KERN-ISBERNER G. A brief survey on forgetting from a knowledge repre- sentation and reasoning perspective[J]. KI-Ku¨nstliche Intelligenz, 2019, 33(1):9-33.

[67] GONC¸ ALVES R, KNORR M, LEITE J. Forgetting in answer set programming–a survey [J]. arXiv preprint arXiv:2107.07016, 2021.

[68] 王文宇. 基于DataSecOps 的个人信息遗忘[J]. 信息安全研究, 2021, 7(9):856-860.

[69] 朱嘉玮. 从“被遗忘权”看个人信息保护[J]. 现代商贸工业, 2020, 8.

[70] LIBERATORE P, SCHAERF M. Arbitration (or how to merge knowledge bases)[J/OL]. IEEE Trans. Knowl. Data Eng., 1998, 10(1):76-90. <https://doi.org/10.1109/69.667090>.

[71] KONIECZNY S, PE´ REZ R P. Merging information under constraints: A logical frame- work[J/OL]. J. Log. Comput., 2002, 12(5):773-808. [https://doi.org/10.1093/logcom/1](https://doi.org/10.1093/logcom/12.5.773) [2.5.773](https://doi.org/10.1093/logcom/12.5.773).

[72] MAYNARD-ZHANG P, LEHMANN D. Representing and aggregating conflicting be- liefs[J/OL]. J. Artif. Intell. Res., 2003, 19:155-203. <https://doi.org/10.1613/jair.1206>.

[73] KONIECZNY S, LANG J, MARQUIS P. Da2 merging operators[J]. Artificial Intelli- gence, 2004, 157(1-2):49-79.

[74] EVERAERE P, KONIECZNY S, MARQUIS P. Disjunctive merging: Quota and gmin merging operators[J/OL]. Artif. Intell., 2010, 174(12-13):824-849. [https://doi.org/10.1](https://doi.org/10.1016/j.artint.2010.05.001) [016/j.artint.2010.05.001](https://doi.org/10.1016/j.artint.2010.05.001).

[75] 徐岱. 遗忘及应用[D]. 北京: 北京大学, 2011.

[76] SANDU G. On the logic of informational independence and its applications[J/OL]. J. Philos. Log., 1993, 22(1):29-60. <https://doi.org/10.1007/BF01049180>.

[77] SANDU G. The logic of informational independence and finite models[J/OL]. Log. J. IGPL, 1997, 5(1):79-95. <https://doi.org/10.1093/jigpal/5.1.79>.

贵州大学博士学位论文

[78] VA¨ A¨ NA¨ NEN J A. On the semantics of informational independence[J/OL]. Log. J. IGPL, 2002, 10(3):339-352. <https://doi.org/10.1093/jigpal/10.3.339>.

[79] SEVENSTER M. On the computational consequences of independence in propositional logic[J/OL]. Synth., 2006, 149(2):257-283. [https://doi.org/10.1007/s11229-005-3878-](https://doi.org/10.1007/s11229-005-3878-5)

[5](https://doi.org/10.1007/s11229-005-3878-5).

[80] DOHERTY P, LUKASZEWICZ W, SZALAS A. Computing strongest necessary and weakest sufficient conditions of first-order formulas[C]//NEBEL B. Proceedings of IJ- CAI’01. [S.l.]: Morgan Kaufmann, 2001: 145-154.

[81] MAKSIMOVA L. Temporal logics of “the next” do not have the beth property[J]. Jour- nal of Applied Non-Classical Logics, 1991, 1:73-76.

[82] D’AGOSTINO G, LENZI G. On modal *µ*-calculus with explicit interpolants[J/OL]. Journal of Applied Logic, 2006, 4(3):256-278. [https://doi.org/10.1016/j.jal.2005.06.00](https://doi.org/10.1016/j.jal.2005.06.008) [8](https://doi.org/10.1016/j.jal.2005.06.008).

[83] SCHNEIDER K. A verified hardware synthesis of esterel programs[C]//IFIP Working Conference on Distributed and Parallel Embedded Systems. [S.l.]: Springer, 2000: 205- 214.

[84] SCHNEIDER K. Embedding imperative synchronous languages in interactive theorem provers[C/OL]//2nd International Conference on Application of Concurrency to System Design (ACSD 2001), 25-30 June 2001, Newcastle upon Tyne, UK. IEEE Computer Society, 2001: 143. <https://doi.org/10.1109/CSD.2001.981772>.

[85] SCHNEIDER K. Proving the equivalence of microstep and macrostep semantics [C/OL]//CARREN˜ O V, MUN˜ OZ C A, TAHAR S. Lecture Notes in Computer Science: volume 2410 Theorem Proving in Higher Order Logics, 15th International Confer- ence, TPHOLs 2002, Hampton, VA, USA, August 20-23, 2002, Proceedings. Springer, 2002: 314-331. [https://doi.org/10.1007/3-540-45685-6 21](https://doi.org/10.1007/3-540-45685-6_21).

[86] 陆钟万. 面向计算机科学的数理逻辑[M]. 北京: 北京大学出版社, 1989.

[87] BEN-ELIYAHU-ZOHARY R, ANGIULLI F, FASSETTI F, et al. Decomposing min- imal models[C/OL]//BARTA´ K R, MCCLUSKEY T L, PONTELLI E. CEUR Work-

shop Proceedings: volume 1648 Proceedings of the Workshop on Knowledge-based Techniques for Problem Solving and Reasoning co-located with 25th International Joint

Conference on Artificial Intelligence (IJCAI 2016), New York City, USA, July 10, 2016. CEUR-WS.org, 2016. <http://ceur-ws.org/Vol-1648/paper1.pdf>.

[88] BEN-ELIYAHU-ZOHARY R, ANGIULLI F, FASSETTI F, et al. Modular construc-

tion of minimal models[C/OL]//BALDUCCINI M, JANHUNEN T. Lecture Notes in Computer Science: volume 10377 Logic Programming and Nonmonotonic Reason- ing - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings. Springer, 2017: 43-48. [https://doi.org/10.1007/978-3-319-61660-5 5](https://doi.org/10.1007/978-3-319-61660-5_5).

[89] 张丽, 王以松, 谢仲涛, 等. 基于MiniSAT 的命题极小模型计算方法[J]. 计算机研究

与发展, 2021, 58(11):2515.

[90] LEI Z, CAI S. Solving (weighted) partial maxsat by dynamic local search for SAT [C/OL]//LANG J. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. ijcai.org, 2018: 1346-1352. <https://doi.org/10.24963/ijcai.2018/187>.

[91] LEI Z, CAI S. Solving set cover and dominating set via maximum satisfiability[C/OL]// The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty- Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. AAAI Press, 2020: 1569-1576.

<https://aaai.org/ojs/index.php/AAAI/article/view/5517>.

[92] KRIPKE S A. Semantical analysis of modal logic i normal modal propositional calculi [J]. Mathematical Logic Quarterly, 1963, 9(5-6):67-96.

[93] BROWNE M C, CLARKE E M, GRU¨ MBERG O. Characterizing finite Kripke struc- tures in propositional temporal logic[J]. Theoretical Computer Science, 1988, 59(1-2): 115-131.

[94] CLARKE E M, EMERSON E A, SISTLA A P. Automatic verification of finite-state concurrent systems using temporal logic specifications[J/OL]. ACM Trans. Program. Lang. Syst., 1986, 8(2):244-263. <https://doi.org/10.1145/5397.5399>.

[95] EMERSON E A, HALPERN J Y. Decision procedures and expressiveness in the tem- poral logic of branching time[J/OL]. Journal of computer and system sciences, 1985, 30(1):1-24. <https://doi.org/10.1016/0022-0000(85)90001-7>.

[96] BOLOTOV A, FISHER M. A clausal resolution method for CTL branching-time tem- poral logic[J/OL]. Journal of Experimental & Theoretical Artificial Intelligence, 1999, 11(1):77-93. <https://doi.org/10.1080/095281399146625>.

[97] ZHANG L, HUSTADT U, DIXON C. First-order resolution for CTL[R]. [S.l.]: Tech- nical Report ULCS-08-010, Department of Computer Science, University of Liverpool, 2008.

[98] ZHANG L, HUSTADT U, DIXON C. A resolution calculus for the branching-time temporal logic CTL[J]. ACM Transactions on Computational Logic (TOCL), 2014, 15 (1):1-38.

[99] ZHANG L, HUSTADT U, DIXON C. CTL-RP: A computation tree logic resolution prover[J/OL]. AI Commun., 2010, 23(2-3):111-136. [https://doi.org/10.3233/AIC-2010](https://doi.org/10.3233/AIC-2010-0463)

[-0463](https://doi.org/10.3233/AIC-2010-0463).

[100] BRADFIELD J C, WALUKIEWICZ I. The *µ*-calculus and model checking[M/OL]// CLARKE E M, HENZINGER T A, VEITH H, et al. Handbook of Model Checking. 2018: 871-919. [https://doi.org/10.1007/978-3-319-10575-8 26](https://doi.org/10.1007/978-3-319-10575-8_26).

[101] KOZEN D. Results on the propositional *µ*-calculus[J/OL]. Theoretical Computer Sci- ence, 1983, 27:333-354. <https://doi.org/10.1016/0304-3975(82)90125-6>.

[102] D’AGOSTINO G, HOLLENBERG M. Uniform interpolation, automata and the modal

*µ*-calculus[J]. Logic Group Preprint Series, 1996, 165.

[103] D’AGOSTINO G, HOLLENBERG M. Logical questions concerning the *µ*-calculus: Interpolation, lyndon and los-tarski[J/OL]. The Journal of Symbolic Logic, 2000, 65 (1):310-332. <https://doi.org/10.2307/2586539>.

[104] JANIN D, WALUKIEWICZ I. Automata for the modal *µ*-calculus and related result- s[C/OL]//WIEDERMANN J, HA´ JEK P. Lecture Notes in Computer Science: volume 969 Mathematical Foundations of Computer Science 1995, 20th International Sympo- sium, MFCS’95, Prague, Czech Republic, August 28 - September 1, 1995, Proceedings. 1995: 552-562. [https://doi.org/10.1007/3-540-60246-1 160](https://doi.org/10.1007/3-540-60246-1_160).

[105] DAVIS M, PUTNAM H. A computing procedure for quantification theory[J/OL]. Jour- nal of the ACM (JACM), 1960, 7(3):201-215. [http://doi.acm.org/10.1145/321033.321](http://doi.acm.org/10.1145/321033.321034) [034](http://doi.acm.org/10.1145/321033.321034).

[106] ENJALBERT P, DEL CERRO L F. Modal resolution in clausal form[J/OL]. Theoretical Computer Science, 1989, 65(1):1-33. <https://doi.org/10.1016/0304-3975(89)90137-0>.

[107] CAVALLI A R, DEL CERRO L F. A decision method for linear temporal logic[C/OL]// SHOSTAK R E. Lecture Notes in Computer Science: volume 170 7th International Conference on Automated Deduction, Napa, California, USA, May 14-16, 1984, Pro- ceedings. Springer, 1984: 113-127. [https://doi.org/10.1007/978-0-387-34768-4 7](https://doi.org/10.1007/978-0-387-34768-4_7).

[108] DELGRANDE J P. Towards a knowledge level analysis of forgetting[C/OL]//BARAL C, GIACOMO G D, EITER T. Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014. AAAI Press, 2014. [http://www.aaai.org/ocs/index.php/KR/KR14/paper/v](http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7979) [iew/7979](http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7979).

[109] KOOPMANN P, SCHMIDT R A. Uniform interpolation of -ontologies using fixpoints [C/OL]//FONTAINE P, RINGEISSEN C, SCHMIDT R A. Lecture Notes in Computer Science: volume 8152 Frontiers of Combining Systems - 9th International Sympo- sium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings. Springer, 2013: 87-102. [https://doi.org/10.1007/978-3-642-40885-4 7](https://doi.org/10.1007/978-3-642-40885-4_7).

[110] ZHAO Y. Automated semantic forgetting for expressive description logics[D/OL]. The University of Manchester (United Kingdom), 2018. [http://ethos.bl.uk/OrderDetails.d](http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.740373) [o?uin=uk.bl.ethos.740373](http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.740373).

[111] KOOPMANN P. Practical uniform interpolation for expressive description logics [D/OL]. University of Manchester, UK, 2015. [http://ethos.bl.uk/OrderDetails.do?uin](http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.674705)

[=uk.bl.ethos.674705](http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.674705).

[112] ZHAO Y, SCHMIDT R A. FAME: an automated tool for semantic forgetting in ex- pressive description logics[C/OL]//GALMICHE D, SCHULZ S, SEBASTIANI R. Lec- ture Notes in Computer Science: volume 10900 Automated Reasoning - 9th Inter- national Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Confer- ence, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings. Springer, 2018: 19-27.

[https://doi.org/10.1007/978-3-319-94205-6 2](https://doi.org/10.1007/978-3-319-94205-6_2).

[113] BIENVENU M. Prime implicates and prime implicants in modal logic[C/OL]// Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Ju- ly 22-26, 2007, Vancouver, British Columbia, Canada. AAAI Press, 2007: 379-384. <http://www.aaai.org/Library/AAAI/2007/aaai07-059.php>.

[114] D’AGOSTINO G. Interpolation in non-classical logics[J]. Synthese, 2008, 164(3): 421-435.

[115] WOLPER P. Temporal logic can be more expressive[J/OL]. Information and control, 1983, 56(1/2):72-99. <https://doi.org/10.1016/S0019-9958(83)80051-5>.

[116] JANIN D, WALUKIEWICZ I. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic[C]//International Conference on Concurrency Theory. [S.l.]: Springer, 1996: 263-277.

[117] WANG Y. On forgetting in tractable propositional fragments[J/OL]. CoRR, 2015, ab- s/1502.02799. <http://arxiv.org/abs/1502.02799>.

[118] COMON H. Tree automata techniques and applications[J]. [http://www.](http://www/) grappa. univ- lille3. fr/tata, 1997.

[119] LANG J, LIBERATORE P, MARQUIS P. Propositional independence: Formula- variable independence and forgetting[J/OL]. Journal of Artificial Intelligence Research, 2003, 18:391-443. <https://doi.org/10.1613/jair.1113>.

[120] FENG R, ACAR E, SCHLOBACH S, et al. On sufficient and necessary conditions in bounded CTL: A forgetting approach[C]//Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning: volume 17. [S.l.: s.n.], 2020: 361-370.

[121] MYCIELSKI J, ROZENBERG G, SALOMAA A. Lecture notes in computer science: volume 1261 structures in logic and computer science, A selection of essays in honor of andrzej ehrenfeucht[C/OL]. Springer, 1997. <https://doi.org/10.1007/3-540-63246-8>.

[122] HINTIKKA J. Distributive normal forms in the calculus of predicates[J]. Cambridge University Press, 1953, 20(2).

[123] YANKOV V A. Three sequences of formulas with two variables in the positive propo- sitional logic[J]. Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya, 1968, 32(4):880-883.

[124] MEIER A, THOMAS M, VOLLMER H, et al. The complexity of satisfiability for fragments of CTL and CTL\*[J]. International Journal of Foundations of Computer Science, 2009, 20(05):901-918.

[125] BOLOTOV A. Clausal resolution for branching-time temporal logic.[D]. [S.l.]: Manch- ester Metropolitan University, 2000.

[126] ZHANG L, HUSTADT U, DIXON C. A refined resolution calculus for CTL[C]// International Conference on Automated Deduction. [S.l.]: Springer, 2009: 245-260.

# 攻读博士学位期间科研和论文情况

参与国家自然科学基金三项，发表CCF B类会议论文一篇，SCI论文两篇在审。

# 致 谢

如果世界上有幸运儿，我想我应该是其中一个。 而这些幸运都是由我身边的老师、

亲人和朋友一点点累积的。

在论文即将完成之际，谨向我的导师致以最诚挚的谢意和最崇高的敬意。 导师严 谨的科研态度让我获益良多，能在知识表示与推理这个要求严谨的方向有点点成果， 导师功不可没。 感谢导师多年来给予的辛勤培养、 无私教育和悉心关怀。 导师在本文 选题、 研究、 以及撰写过程中自始至终给予的悉心指导，使我能够在宽松的学术氛围 中及时跟踪国际最前沿的学术动态并且全心全力地投入到感兴趣的科学研究中。 导师 崇高的师德风范、 渊博的知识储备、 严谨的治学态度、 敏锐的学术直觉、 忘我的工作 精神、积极的生活态度使我深受教益。

感谢参与该博士学位论文的各位评审，感谢您们为提高我的博士学位论文质量所

提出的宝贵修改意见和建议。

感谢我的家人付出的一切。 感谢他们在生活上的悉心照顾和学业上的鼓励支持，

博士论文的顺利完成同时凝结着他们的心血、汗水、理解和牺牲。

最后，向所有曾经给予我关心和帮助的人们致以最真诚的谢意和祝福。