附件1：

**贵州大学拟引进人才**

**科研计划书**

（自然科学类）

|  |  |
| --- | --- |
| 学 科 分 类 |  |
| 项 目 名 称 |  |
| 拟引进人才姓名 | 冯仁艳 |
| 拟 引 进 单 位 | 贵州大学人工智能研究院 |
| 填 表 日 期 | 2022/3/31 |

一、基本信息

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **申**  **请**  **人**  **信**  **息** | 姓 名 | 冯仁艳 | 性别 | 女 | 出生年月 | 1990.11 |
| 学历学位 | 研究生/博士 | | 职称 | | 无 |
| 联系电话 | 13595059698 | | | 电子邮箱 | renyan\_feng@163.com |
| 博士毕业学校及专业（方向） | | 贵州大学 | | | |
| 主要研究领域 | | 软件工程技术与人工智能 | | | |
| **项**  **目**  **基**  **本**  **信**  **息** | 项目名称 |  | | | | |
| 英文名称 |  | | | | |
| 研究期限 |  | | | | |
| 研究方向 |  | | | | |
| 申请直接费用 |  | | | | |
| **中文关键词** |  | | | | |
| **英文关键词** |  | | | | |

二、科学问题属性

1.“**鼓励探索，突出原创**”：科学问题源于科研人员的灵感和新思想，且具有鲜明的首创性特征，旨在通过自由探索产出从无到有的原创性成果。

2.“**聚焦前沿，独辟蹊径**”：科学问题源于世界科技前沿的热点、难点和新兴领域，且具有鲜明的引领性或开创性特征，旨在通过独辟蹊径取得开拓性成果，引领或拓展科学前沿。

3.“**需求牵引，突破瓶颈**”：科学问题源于国家重大需求和经济主战场，且具有鲜明的需求导向、问题导向和目标导向特征，旨在通过解决技术瓶颈背后的核心科学问题，促使基础研究成果走向应用。

4.“**共性导向，交叉融通**”：科学问题源于多学科领域交叉的共性难题，具有鲜明的学科交叉特征，旨在通过交叉研究产出重大科学突破，促进分科知识融通发展为知识体系。

|  |  |  |
| --- | --- | --- |
| 请阐明选择该科学问题属性的理由（800字以内）：软件和硬件系统的缺陷会导致严重的后果，如：美国爱国者导弹系统舍入错误引起严重的伤亡，温州7.23动车信号设备设计上的严重缺陷导致的动车脱节脱轨等。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。  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。形式化验证是一种广泛应用在硬件[1][2][3]和软件系统中[4][5]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。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)。  形式化验证的模型检测首先由Clarke提出。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，规范被表示成一个逻辑公式；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（）: 判断是否是的一个模型。在模型检测中，反应式系统通常用Kripke结构表示[6][7]。  当给定了反应式系统的Kripke结构模型和规范，就存在模型检测问题：。若不满足，使用什么信息对进行修正，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[8]，在人工智能（artificial intelligence，AI）中也称为最弱充分条件（weakest sufficient condition，WSC），与之对偶的另一个概念是最强必要条件（strongest necessary condition，SNC）[9][10]。  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”。  遗忘是一种知识抽取技术，其被应用于信息隐藏、冲突解决、多智能体中的知识融合、本体摘要提取和计算逻辑差等领域。本课题拟从遗忘的角度解决上述问题，架起知识表示与推理和形式化验证的桥梁。 | | |
| 中  文  摘  要 | 反应式系统（reactive system）是一种应用广泛的不终止系统。模型检测是保证系统正确性行之有效的形式化方法，此时反应式系统被表示为一个Kripke结构。规范（specification）信息提取和寻找使得系统满足给定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是最弱充分条件（the weakest sufficient condition, WSC）和最强必要条件（the strongest necessary condition, SNC）。遗忘是知识抽取的重要技术。本项目从遗忘的角度出发，研究反应式系统这种不终止类系统的WSC求解。特别地，本文以时序逻辑系统中的计算树逻辑（computation tree logic）和mu-演算（mu-calculus）为系统规范语言，研究这两种逻辑语言下的遗忘理论，并探讨这两种语言下遗忘与WSC的关系。 |
| 英  文  摘  要 | The reactive system is a kind of non-terminable system widely used in life. Model checking is an efficient technique to guarantee a system’s correctness, and a reactive system is represented as a Kripke structure in such an environment. Extracting information from a specification and finding the weakest information needed to make the system meet the specification are important problems for a long time. Forgetting is a significant technique to distill knowledge. This project is to compute the WSC of the non-terminable systems (e.g., reactive systems) in terms of forgetting. In particular, we take CTL and mu-calculus as the system specification languages to study the forgetting in these two logic systems, and explore the relationship between forgetting and WSC under these two logics. |

三、正文

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **参照以下提纲撰写，要求内容翔实、清晰，层次分明，标题突出。**  **（一）立项依据与研究内容**（建议 8000 字以内）：  1.项目的立项依据（研究意义、国内外研究现状及发展动态分析，需结合科学研究发展趋势来论述科学意义；或贵州省“四新”“四化”建设及社会发展中迫切需要解决的关键科技问题来论述其应用前景。附主要参考文献目录）；  2.项目的研究内容、研究目标，以及拟解决的关键科学问题（此部分为重点阐述内容）；  3.拟采取的研究方案及可行性分析（包括研究方法、技术路线、实验手段、关键技术等说明）；  4.本项目的特色与创新之处；  5.年度研究计划及预期研究结果（包括拟组织的重要学术交流活动、国际合作与交流计划等）。  1.研究意义和国内外研究现状  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。  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（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(*s*)的方案：   1. 先生产足够的*se*，然后在再生产*sp*； 2. 先生产足够的*sp*，然后再生产*se*； 3. 同时生产*se*和*sp*。   D:\（博士）我的资料\Git\fry\毕业论文\终版\NnewCar.png这一过程可以由图 1中的Kripke结构（带标签的状态转换图）形式化地展现出来，其中:  图 1   * ={*d*,*s*,*se*,*sp*}为该工厂所需要考虑的原子命题集； * ={*s0*,*s1*,*s2*,*s3*,*s4*}为状态空间； * ={(*s0*,*s1*),(*s1*,*s2*),(*s1*,*s3*),(*s1*,*s4*),(*s2*,*s0*),(*s3*,*s0*),(*s4*, *s0*)}为状态转换关系集； * :S→2V为标签函数，具体地：(*s0*)={*d*}、(*s1*)={*s*}、(*s2*)={*se*}、(*s3*)={*sp*}和(*s4*)={*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。这意味着所有规范和Kripke结构都不再需要考虑*sp*的，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[11]。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，而保留与这些原子命题无关的性质是一个复杂的问题**。  1.1研究意义：  最强后件和最弱前件（WP）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[12]和制定验证条件[13]，还被应用于形式化验证过程中的负例生成[14]和系统精化（refinement）[15]。此外，还被用于概率程序中证明概率程序中给定的后件有确定的概率。  当模型不满足规范（）时，找到某个性质，使得若按照此性质进行修改后得到的新模型能满足是必要的，即是使得成立的充分条件（）。然而，现有方法不能直接应用于如下情形：计算当为不终止系统（如：反应式系统）时使得“”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“在下，使得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。如下问所说，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，架起了KR与形式化验证的桥梁。**  1.2国内外研究现状及发展动态分析：  （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。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）[[1]](#footnote-1)。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[7]。此外，其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[16]。反应式系统由来已久[17][18]，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[19]。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：模型检测方法[20][21][7]、目标驱动的运行时需求建模框架[16]、基于图模型的实时规则调度方法（graph-based real-time rule scheduling,简称GBRRS）[19]、SPARDL[22]和选择性测试方法[23]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）来对系统进行修改至关重要。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，本项目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。  （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[24]，在此之前的研究中多提到的是均匀插值[25][26]和SOQE[27]。  在命题逻辑中（propositional logic，PL），公式*ψ*遗忘一个原子命题p，通常记为Forget(*ψ*,{p})，得到的结果为*ψ*[p/false]∨*ψ*[p/true]，其中*ψ*[X/Y]为将*ψ*中X的全部出现替换为Y得到的结果。公式*ψ*中遗忘有限原子命题集P的定义如下：  Forget(*ψ*,ø) = *ψ*,  Forget(*ψ*, P ∪ {q}) = Forget(Forget(*ψ*, {q}), P).  在FOL中，遗忘通常被看作SOQE问题的一个实例：从FOL公式*ψ*中遗忘一个n-元谓词P，得到的结果为一个二阶公式∃R*ψ*[P/R][24]，其中R为n-元变量。从这个角度看来，遗忘就是找到一个与二阶公式∃R*ψ*[P/R]等价的一阶公式。然而，二阶逻辑的表达能力严格大于一阶逻辑，因而可以容易得出FOL下的遗忘不是封闭（存在）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[28]，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，其子类中的遗忘通常被用来抽取视图（review）[29][30][31][32][33]。这些子类包括：ALCOI（the basic ALC extended with nominals and inverse roles）和ALCOIH（ALCOI extended with role hierarchies, the universal role and role conjunction）。特别地，Sakr等人探索了遗忘名字（nominal）的方法，以捕获本体剩余词汇表上的语义内容[34]； Zhao等人提出了带有名字的描述逻辑计算遗忘的方法[33]。除了本体摘要抽取[35]，描述逻辑中的遗忘也被用与溯因推理（abduction reasoning)[36]。  现有计算一阶逻辑和描述逻辑遗忘的方法有基于归结和基于Ackermann引理的方法[37]。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Ackermann引理得到的。  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）[38]。因为模态逻辑系统中引入了模态词，模态逻辑中的遗忘不同于经典逻辑下的遗忘。所以，不能以简单的谓词（命题）替换的方式获取遗忘的结果。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[39]、模态一阶逻辑S5中的遗忘也得到了研究[40]，Fang等人讨论了关于多模态（multi-modal）Kn、 Dn、 Tn、K45n、KD45n情形下遗忘的存在性[41][42][42]——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。从形式上说，如果一个逻辑系统L中任意的公式α和*ψ*，若α⊨L*ψ*，则存在一个公式β使得α⊨Lβ、β⊨L *ψ*和Var(β)⊆Var(α)∩Var(*ψ*)，则L具有或Craig插值性质，若β与*ψ*无关，则L具有均匀插值性质。其在模态逻辑和时序逻辑系统中取得了一系列研究成果：LTL、CTL和CTL\*不具有均匀插值性质[43][44]，S5、K和KD模态逻辑系统具有均匀插值性质[45]，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）[46]、K4、和S4及其扩展都没有均匀插值性质[47]，因而其遗忘也不是封闭的。研究具有均匀插值性质的模态逻辑遗忘可以借鉴S5系统的遗忘方法，也可以参考K系统下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[48]。  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[49][50][51][52][53][54][55][56]，和Eiter、Gonccalves等人的综述[57][58]。  遗忘有很多应用，这里列出下面几点：   1. 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[10]。 2. 信息隐藏：在某些关键领域，为实现隐私保护，必须隐藏敏感信息。现有方法包括基于本体[28]和基于DataSecOps的个人信息保护[59]。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[60]; 3. 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(U1)-(U8)较为常用； 4. 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； 5. 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），因而不能简单地将它们放在一起。归并考虑当这些信息矛盾时如何将它们整合[61][62][63][64][65]，且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[66]。 6. 逻辑独立性：生活中许多场景都需要判断哪些信息是无关的（或相关的），即信息的独立性[67][68][69][70]，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[66]。   （3）最强必要条件（SNC）和最弱充分条件（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。此外，SNC和WSC还可用于系统精化[15]、模型检测中的负例产生[14]、汇编语言程序的推理[12]和制定验证条件[13]。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前提条件（WP），WSC能被所有的充分条件蕴涵。  最弱预期（Weakest preexpectation）是在WP的基础上发展起来的，是通过将谓词的域（即：从状态到布尔值的函数）扩展到随机变量的域（即：从状态到0到1区间的函数）上得到的[71][72][73]。最弱预期被用来证明概率程序中给定的后件有确定的概率，例：Agda中的比特币智能合约验证[74]和检查代数安全断言的有效性[75]。  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 statement）和循环语句（loop statement））分别对应了这四种规则[8]。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[10]，且SNC和WSC都可以用遗忘来计算[9][76]。随后，SNC和WSC被扩展到FOL下，且用SOQE实现了SNC和WSC的计算[76]。  形式化验证是一种广泛应用在硬件\cite{lam2007,lv2000,jani2007Verilog}和软件系统中\cite{yuan2008,gu2005}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。软件和硬件系统的缺陷会导致严重的后果，如表 1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\cite{DBLP:conf/tphol/Fox03}， 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\cite{DBLP:conf/sefm/DaumSS09}。  表 1   |  |  |  | | --- | --- | --- | | 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（reactive system）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$\varphi\_{imp}$和$\varphi\_{spec}$，然后证明其推理“$\varphi\_{imp}\rto \varphi\_{spec}$”或“$\varphi\_{imp}\lrto\varphi\_{spec}$”的正确性，即：证明模型系统$\varphi\_{imp}$是否满足给定的规范$\varphi\_{spec}$。常用的自动定理证明方式有归结（resolution）\cite{DBLP:journals/jacm/Robinson65}和常用于模态逻辑的基于表推理（tableau）\cite{hughes1996new}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\cite{Hoare1969}、动态逻辑\cite{harel1979first}和分离逻辑\cite{DBLP:conf/lics/Reynolds02}形式化验证过程中寻找不变式问题，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\cite{DBLP:journals/ai/Lin16}。  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\cite{DBLP:conf/spin/Clarke08}。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）的方法证明系统是否满足给定的规范是不必要的\cite{clarke1981design}。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$\Hm$，规范被表示成一个逻辑公式$\varphi$；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（$\Hm\models^? \varphi$）: 判断$\Hm$是否是$\varphi$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\cite{DBLP:journals/ac/BiereCCSZ03}和基于OBDD的符号模型检测\cite{burch1992symbolic}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。与此同时，大量优质的模型验证器，如：NuSMV\footnote{http://nusmv.fbk.eu/}、SPIN\footnote{http://spinroot.com/spin/whatispin.html}、Uppaal\footnote{http://www.uppaal.org/} 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。  时序逻辑（也叫时态逻辑）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$\mu$-演算（$\mu$-calculus），如：“某给定的系统中存在一条路径，且这条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\cite{DBLP:series/txtcs/Schneider04}。充分考虑这两种逻辑语言自身的特性，本文主要研究$\CTL$和$\mu$-演算。因此，本文所说的公式指$\CTL$（或$\mu$-演算）公式，即用来描述一个规范（或性质）的公式是$\CTL$（或$\mu$-演算）公式。  在模型检测中，反应式系统通常用Kripe结构表示~\cite{DBLP:journals/fcsc/ZhuWXX10,DBLP:series/txtcs/Schneider04}。当给定了反应式系统的Kripke结构模型$\Hm$和规范$\varphi$，就存在模型检测问题$\Hm\models^? \varphi$：  \[\hbox{系统输出=}  \left\{  \begin{array}{ll}  Yes, \ \ \qquad \qquad \hbox{若$\Hm\models \varphi$；} \\  No\hbox{和负例}, \ \ \ \quad \ \ \hbox{否则。}  \end{array}  \right.  \]  当$\Hm$不满足$\varphi$时，\emph{使用什么信息对$\Hm$进行修正，使得其满足给定的规范$\varphi$是一个重要的问题}。这就是寻找{\em 最弱前件}（weakest precondition，WP）问题\cite{DBLP:journals/cacm/Dijkstra75}，在人工智能（artificial intelligence，AI）中也称为{\em 最弱充分条件}（weakest sufficient condition，WSC），与之对偶的另一个概念是最强必要条件（strongest necessary condition，SNC）\cite{DBLP:journals/ai/Lin01,DBLP:journals/jair/Lin03}。  最弱预期（Weakest preexpectation）是在WP的基础上发展起来的，是通过将谓词的域（即：从状态到布尔值的函数）扩展到随机变量的域（即：从状态到0到1区间的函数）上得到的\cite{DBLP:journals/toplas/MorganMS96,DBLP:phd/dnb/Kaminski19,fesefeldt2021weakest}。最弱预期被用来证明概率程序中给定的后件有确定的概率，例：Agda中的比特币智能合约验证\cite{DBLP:journals/corr/abs-2203-03054}和检查代数安全断言的有效性\cite{DBLP:journals/scp/Boreale20}。  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（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(*s*)的方案：   1. 先生产足够的*se*，然后在再生产*sp*； 2. 先生产足够的*sp*，然后再生产*se*； 3. 同时生产*se*和*sp*。   D:\（博士）我的资料\Git\fry\毕业论文\终版\NnewCar.png这一过程可以由图 1中的Kripke结构（带标签的状态转换图）形式化地展现出来，其中:  图 1   * ={*d*,*s*,*se*,*sp*}为该工厂所需要考虑的原子命题集； * ={*s0*,*s1*,*s2*,*s3*,*s4*}为状态空间； * ={(*s0*,*s1*),(*s1*,*s2*),(*s1*,*s3*),(*s1*,*s4*),(*s2*,*s0*),(*s3*,*s0*),(*s4*, *s0*)}为状态转换关系集； * :S→2V为标签函数，具体地：(*s0*)={*d*}、(*s1*)={*s*}、(*s2*)={*se*}、(*s3*)={*sp*}和(*s4*)={*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。这意味着所有规范和Kripke结构都不再需要考虑*sp*的，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  ~\cite{Baier:PMC:2008}。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，而保留与这些原子命题无关的性质是一个复杂的问题**。  2. 项目的研究内容、研究目标，以及拟解决的关键科学问题（此部分为重点阐述内容）。  系统规范的描述语言以时序逻辑为主。其中计算树逻辑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技术，其被应用于信息隐藏、冲突解决、多智能体中的知识融合、本体摘要提取和计算逻辑差等领域。特别地，在情景演算中，遗忘被用于求解WSC和SNC，从而计算智能规划中的后继状态公理。本文从遗忘的角度出发解决上述提到的问题，**主要研究内容**如下：  1. 研究CTL和μ-演算下遗忘的概念及其相关性质。首先，探索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和μ-演算遗忘的四条公设，并研究遗忘与这四条公设之间的关系。  特别地，CTL不具有均匀插值（Uniform interpolation）性质，而μ-演算的句子具有这一性质。此外，CTL公式对互模拟是不变的，即任意两个互模拟的模型同时满足或不满足同一个CTL公式。μ-演算公式含有变元，虽然μ-演算句子对互模拟是不变的，但μ-公式对互模拟不是不变的。因此，CTL遗忘理论注定与μ-演算遗忘理论具有本质的区别。本文从这些性质中抽丝剥茧，分别探索CTL和μ-演算的遗忘理论。  2. 研究CTL和μ-演算下遗忘是否存在的可判定性，即判定CTL和μ-演算下遗忘的结果是否分别可用CTL和μ-演算公式来表示。已有结果表明，μ-演算的句子具有均匀插值性质，即：μ-演算的句子的遗忘结果可以用μ-演算的句子来表示。但是当μ-演算公式中含有自由变元时，公式的遗忘结果是否总是μ-演算公式可表达的？此外，CTL不具有均匀插值性质，那么CTL公式的遗忘结果是否总是CTL公式可表达的？  3. 计算CTL和μ-演算遗忘。如果CTL和μ-演算公式的遗忘结果是否存在是可判定的，则如何设计一个可靠且完备的算法计算遗忘结果是一个重要的问题。如果CTL和μ-演算公式的遗忘结果是否存在是不可判定的，则探究遗忘存在的CTL和μ-演算子类就尤为重要。  4. 给出遗忘与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，并证明其与Katsuno等人提出的八条公设之间的关系。  针对以上几个内容，有以下几个**拟解决的关键科学问题**：  （1）具有不动点算子和变元的公式如何定义遗忘？如何遗忘变元？  （2）遗忘什么情形下存在？如何计算遗忘？  （3）CTL和μ-演算下，遗忘和WSC（SNC）是否存在这一问题的可判定性。  （4）遗忘与反应式系统的WSC和SNC的关系如何？即：如何使用遗忘计算WSC（SNC）？是否可以SNC和WSC计算遗忘？  3.拟采取的研究方案及可行性分析（包括研究方法、技术路线、实验手段、关键技术等说明）；  （1）**研究方案**  本文的目的是从遗忘理论的角度出发，拟研究反应式系统的SNC和WSC的计算，为计算不终止类系统下的定义在某个符号集上的SNC和WSC提供新的方法，从而架起形式化验证与KRR之间的桥梁。总体研究方案如图 2所示。    图 2 总体研究方案  本文拟从探索遗忘的基本属性出发首先研究CTL和-演算的遗忘理论，包括其算法与复杂性等，最终形成CTL和-演算的遗忘理论框架。为研究使用遗忘理论计算反应式系统的SNC和WSC奠定理论基础，从而促进系统精化和负例生成的研究。  给定一个Kripke结构和性质（公式），如果是的一个模型（也即是满足属性），则什么都不用做；如果不是的一个模型，如何使用遗忘理论的方法计算一个使得满足的充分条件是本文要解决的一个关键问题。其技术路线如图 3所示。    图 3 关键技术  当不满足时，本文拟将表示成一个CTL公式，即：的特征公式。在这种情况下，模型检测问题就变成了一个逻辑推论问题。如果，那么，需要找到一个使得。此时，探讨使用遗忘理论计算就变得相当自然，从而找到使得F在下满足的充分条件。  （2）**可行性分析**：理论上，在前期工作中对已有的不同逻辑语言下的遗忘理论进行了深入学习，同时CTL、μ-演算原子命题遗忘已经取得了初步进展（发表了一篇CCF推荐的B类会议：KR 2020，两篇SCI论文在审）。此外，已经熟悉了现有比较优越的模型检测工具，为后期的工作开展打下了一定的基础。  实践上，可信软件是当前比较热门的研究领域，其涉及到的模型检测是证明系统是否满足给定性质的重要技术。探讨反应式系统的WSC的计算方法为负例生成和系统精化奠定了基础。此外，本论文工作依托贵州大学人工智能与软件新技术实验室，完整地配备了高性能工作站及工作平台，为论文的顺利完成提供了充分的保障。因此，本课题从理论上和实践上都是可行的。  4.本项目的特色与创新之处；  本文从知识表示与推理（KRR）的角度研究形式化方法中的反应式系统的WSC的计算方法，搭建起了KRR与形式化方法的桥梁。特别地，本文从已经很成熟的遗忘理论出发，研究形式化中表达能力较强的CTL和μ-演算在反应式系统下的WSC的计算方法。其创新之处主要体现在如下两个方面：  (1)CTL和μ-演算的遗忘理论  据我所知，目前还没有人尝试研究这种带有时序连接词的逻辑语  言的遗忘理论。从这种逻辑语言中直接遗忘原子命题的遗忘理论为这类逻辑语言遗忘理论的研究提供了一个理论框架，且为计算反应式系统的WSC奠定了理论基础，从而进一步完善这类系统在模型检测领域的研究。  (2)从KRR的角度探讨形式化方法中的重要概念  与以前的形式化方法中使用Hoare三元组计算可终止代码段的WSC不同，本文探讨反应式系统这种不终止系统的WSC的计算，因而不能使用传统的方法进行计算。本文拟研究将系统对应的Kripke结构用特征公式描述，然后借助遗忘理论来计算其所对应的WSC，这架起了KRR与形式化验证的桥梁。  此外，本文立于上述所说的理论之上，从实际出发探索了上述方法在形式化方法实例中的应用。    5.年度研究计划及预期研究结果（包括拟组织的重要学术交流活动、国际合作与交流计划等）。  **参考文献**  [1] 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.  [2] 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.  [3] 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.  [4] 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.  [5] 古天龙. 软件开发的形式化方法[M]. 北京: 高等教育出版社, 2005.  [6] ZHU J, WANG H, XU Z等. A new model for model checking: cycle-weighted Kripke structure[J]. Frontiers Comput. Sci. China, 2010, 4(1): 78–88.  [7] SCHNEIDER K. Verification of Reactive Systems - Formal Methods and Algorithms[M]. Springer, 2004.  [8] DIJKSTRA E W. Guarded Commands, {N}ondeterminacy and {F}ormal {D}erivation of {P}rograms[J]. Commun. {ACM}, 1975, 18(8): 453–457.  [9] LIN F. On strongest necessary and weakest sufficient conditions[J]. Artif. Intell., 2001, 128(1–2): 143–159.  [10] LIN F. Compiling Causal Theories to Successor State Axioms and STRIPS-Like Systems[J]. Journal of Artificial Intelligence Research, 2003, 19: 279–314.  [11] BAIER C, KATOEN J. Principles of Model Checking[M]. The MIT Press, 2008.  [12] LEGATO W J. A weakest precondition model for assembly language programs[J]. April, 2002.  [13] LEINO K R M. Efficient weakest preconditions[J]. Information Processing Letters, 2005, 93(6): 281–288.  [14] DAILLER S, HAUZAR D, MARCHÉ C等. Instrumenting a weakest precondition calculus for counterexample generation[J]. Journal of logical and algebraic methods in programming, Elsevier, 2018, 99: 97–113.  [15] WOODCOCK J C P, MORGAN C. Refinement of state-based concurrent systems[C]//International Symposium of VDM Europe. Springer, Berlin, Heidelberg, 1990: 340–351.  [16] 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.  [17] HAREL D, PNUELI A. On the Development of Reactive Systems[C]//APT K R. Logics and Models of Concurrent Systems - Conference proceedings, Colle-sur-Loup (near Nice), France, 8-19 October 1984. Springer, 1984, 13: 477–498.  [18] PNUELI A. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends[G]//Current trends in Concurrency. Springer, 1986: 510–584.  [19] 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).  [20] CLARKE E M, GRUMBERG O, PELED D A. Model checking[M]. {MIT} Press, 2001.  [21] CLARKE E, GRUMBERG O, LONG D. Model checking[C]//NATO ASI DPD. 1996: 305–349.  [22] 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.  [23] 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.  [24] LIN F, REITER R. Forget it[C]//Working Notes of AAAI Fall Symposium on Relevance. 1994: 154–159.  [25] VISSER A. Uniform Interpolation and Layered Bisimulation[C]//Gödel’96. 1996: 139–164.  [26] KONEV B, WALTHER D, WOLTER F. Forgetting and uniform interpolation in large-scale description logic terminologies[C]//Twenty-First International Joint Conference on Artificial Intelligence. 2009.  [27] ACKERMANN W. Untersuchungen über das Eliminationsproblem der mathematischen Logik[J]. Mathematische Annalen, Springer, 1935, 110(1): 390–413.  [28] KONEV B, LUTZ C, WALTHER D等. Model-theoretic inseparability and modularity of description logic ontologies[J]. Artificial Intelligence, 2013, 203: 66–103.  [29] WANG Z, WANG K, TOPOR R W等. Forgetting for knowledge bases in \textrm{DL-Lite}[J]. Annuals of Mathematics and Artificial Intelligence, 2010, 58(1–2): 117–151.  [30] LUTZ C, WOLTER F. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics[C]//IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. Barcelona, Catalonia, Spain: IJCAI/AAAI, 2011: 989–995.  [31] KONEV B, LUDWIG M, WALTHER D等. The Logical Difference for the Lightweight Description Logic $\mathcal{EL}$[J]. Journal of Artificial Intelligence Research, 2012, 44: 633–708.  [32] ZHAO Y, SCHMIDT R A. Role forgetting for ALCOQH ($Δ$)-ontologies using an ackermann-based approach[C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence. 2017: 1354–1361.  [33] ZHAO Y, SCHMIDT R A, WANG Y等. A Practical Approach to Forgetting in Description Logics with Nominals[C]//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. {AAAI} Press, 2020: 3073–3079.  [34] SAKR M, SCHMIDT R A. Semantic Forgetting in Expressive Description Logics[C]//KONEV B, REGER G. Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings. Springer, 2021, 12941: 118–136.  [35] CHEN J, ALGHAMDI G, SCHMIDT R A等. Ontology Extraction for Large Ontologies via Modularity and Forgetting[C]//KEJRIWAL M, SZEKELY P A, TRONCY R. Proceedings of the 10th International Conference on Knowledge Capture, {K-CAP} 2019, Marina Del Rey, CA, USA, November 19-21, 2019. ACM, 2019: 45–52.  [36] FORTIN M, KONEV B, WOLTER F. Interpolants and Explicit Definitions in Horn Description Logics (Extended Abstract)[C]//HOMOLA M, RYZHIKOV V, SCHMIDT R A. Proceedings of the 34th International Workshop on Description Logics {(DL} 2021) part of Bratislava Knowledge September {(BAKS} 2021), Bratislava, Slovakia, September 19th to 22nd, 2021. CEUR-WS.org, 2021, 2954.  [37] GABBAY D M, OHLBACH H J. Quantifier Elimination in Second-Order Predicate Logic[C]//Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR’92). Cambridge, MA, USA, October 25-29, 1992. 1992: 425–435.  [38] ZHANG Y, ZHOU Y. Knowledge forgetting: Properties and applications[J]. Artificial Intelligence, 2009, 173(16–17): 1525–1537.  [39] 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.  [40] 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.  [41] FANG L, LIU Y, VAN DITMARSCH H. Forgetting in multi-agent modal logics[J]. Artif. Intell., 2019, 266: 51–80.  [42] 文习明, 方良达, 余泉等. 多智能体模态逻辑系统$K\_n$中的知识遗忘[J]. 逻辑学研究, 2019, 02(12): 43–62.  [43] MAKSIMOVA L. Temporal logics of ``the next’’ do not have the beth property[J]. Journal of Applied Non-Classical Logics, 1991, 1: 73–76.  [44] D’AGOSTINO G. Interpolation in non-classical logics[J]. Synthese, 2008, 164(3): 421–435.  [45] IEMHOFF R. Uniform interpolation and sequent calculi in modal logic[J]. Archive for Mathematical Logic, 2019, 58(1–2): 155–181.  [46] FINE K. Failures of the Interpolation Lemma in Quantified Modal Logic[J]. The Journal of Symbolic Logic, 1979, 44(2): 201–206.  [47] SCHUMM G F. Some failures of interpolation in modal logic[J]. Notre Dame journal of formal logic, 1986, 27(1): 108–110.  [48] Second Order Quantifier Elimination Foundations, Computational Aspects and Applications.pdf[J]. .  [49] ZHANG Y, FOO N Y. Solving logic program conflict through strong and weak forgettings[J]. Artificial Intelligence, 2006, 170(8–9): 739–778.  [50] EITER T, WANG K. Semantic forgetting in answer set programming[M]. Elsevier Science Publishers Ltd., 2008.  [51] WONG K-S. Forgetting in Logic Programs[D]. The University of New South Wales, 2009.  [52] WANG Y, ZHANG Y, ZHOU Y等. Knowledge Forgetting in Answer Set Programming[J]. Journal of Artificial Intelligence Research, 2014, 50: 31–70.  [53] WANG Y, WANG K, ZHANG M. Forgetting for Answer Set Programs Revisited[C]//IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence. Beijing, China: IJCAI/AAAI, 2013: 1162–1168.  [54] WANG Y, WANG K, WANG Z等. Knowledge Forgetting in Circumscription: {A} Preliminary Report[C]//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.  [55] DELGRANDE J P. A Knowledge Level Account of Forgetting[J]. J. Artif. Intell. Res., 2017, 60: 1165–1213.  [56] GONÇALVES R, KNORR M, LEITE J等. On the limits of forgetting in Answer Set Programming[J]. Artif. Intell., 2020, 286: 103307.  [57] EITER T, KERN-ISBERNER G. A Brief Survey on Forgetting from a Knowledge Representation and Reasoning Perspective[J]. KI-Künstliche Intelligenz, Springer, 2019, 33(1): 9–33.  [58] GONÇALVES R, KNORR M, LEITE J. Forgetting in Answer Set Programming--A Survey[J]. arXiv preprint arXiv:2107.07016, 2021.  [59] 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.  [60] 朱嘉玮. 从 “被遗忘权” 看个人信息保护[J]. 现代商贸工业, 2020, 8.  [61] LIBERATORE P, SCHAERF M. Arbitration (or How to Merge Knowledge Bases)[J]. {IEEE} Trans. Knowl. Data Eng., 1998, 10(1): 76–90.  [62] KONIECZNY S, PÉREZ R P. Merging Information Under Constraints: {A} Logical Framework[J]. J. Log. Comput., 2002, 12(5): 773–808.  [63] MAYNARD-ZHANG P, LEHMANN D. Representing and Aggregating Conflicting Beliefs[J]. J. Artif. Intell. Res., 2003, 19: 155–203.  [64] KONIECZNY S, LANG J, MARQUIS P. DA2 merging operators[J]. Artificial Intelligence, Elsevier, 2004, 157(1–2): 49–79.  [65] EVERAERE P, KONIECZNY S, MARQUIS P. Disjunctive merging: Quota and Gmin merging operators[J]. Artif. Intell., 2010, 174(12–13): 824–849.  [66] 徐岱. 遗忘及应用[D]. 北京: 北京大学, 2011.  [67] SANDU G. On the logic of informational independence and its applications[J]. J. Philos. Log., 1993, 22(1): 29–60.  [68] SANDU G. The Logic of Informational Independence and Finite Models[J]. Log. J. {IGPL}, 1997, 5(1): 79–95.  [69] VÄÄNÄNEN J A. On the semantics of informational independence[J]. Log. J. {IGPL}, 2002, 10(3): 339–352.  [70] SEVENSTER M. On The Computational Consequences of Independence in Propositional Logic[J]. Synth., 2006, 149(2): 257–283.  [71] MORGAN C, MCIVER A, SEIDEL K. Probabilistic Predicate Transformers[J]. {ACM} Trans. Program. Lang. Syst., 1996, 18(3): 325–353.  [72] KAMINSKI B L. Advanced weakest precondition calculi for probabilistic programs[D]. {RWTH} Aachen University, Germany, 2019.  [73] FESEFELDT I. Weakest Preexpectation Reasoning for Probabilistic Concurrent Pointer Programs[C]//Tagungsband zum 21. Kolloquium Programmiersprachen und Grundlagen der Programmierung. 2021: 34.  [74] ALHABARDI F F, BECKMANN A, LAZAR B等. Verification of Bitcoin’s Smart Contracts in Agda using Weakest Preconditions for Access Control[J]. CoRR, 2022, abs/2203.0.  [75] BOREALE M. Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial odes[J]. Sci. Comput. Program., 2020, 193: 102441.  [76] DOHERTY P, LUKASZEWICZ W, SZALAS A. Computing Strongest Necessary and Weakest Sufficient Conditions of First-Order Formulas[C]//NEBEL B. IJCAI. Morgan Kaufmann, 2001: 145–154.  **（二）研究基础与工作条件**  1.研究基础（与本项目相关的研究工作积累 和已取得的研究工作成绩）；  与本项目相关的研究工作积累：首先，本人硕士时研究了基于归结的模态逻辑S5遗忘的计算。时序逻辑是模态逻辑的一种，因此，S5遗忘计算的研究有利于时序逻辑遗忘研究的开展。其次，本人博士期间阅读了大量相关文献，对这方面相关的知识和研究动向有了较为深刻的理解。此外，研究了CTL和μ-演算遗忘的定义和基本性质。这为以后深度研究CTL和μ-演算遗忘打下了坚实的基础。最后，博士期间深刻的科研经历为后续研究提供了有利的指导。  已取得的研究工作成绩：   1. 提出了时序逻辑下遗忘理论的基本框架。博士期间的工作从定义CTL和μ-演算遗忘开始，逐步深入，探索了遗忘的基本性质和计算方法，为时序逻辑遗忘理论提供基本研究框架。 2. 给出了使用遗忘计算反应式系统WSC（SNC）的方法。反应式系统是一种不终止的系统，在模型检测中，其被表示为Kripke结构。在博士研究工作中，探索了将有限Kripke结构表示为CTL公式的方法，从而使用遗忘计算其WSC（SNC）。 3. 提出了变元-命题-互模拟并探索其性质。已有结果表明μ-演算公式对互模拟不具有不变性。在博士论文中，提出了变元-命题-互模拟，并证明了其对μ-演算公式是不变的。这就是说，如果两个模型结构是<Var,V>-互模拟的，则对任意与Var和V无关的μ-公式，这两个模型结构同时满足或不满足这个公式。   2.工作条件（包括已具备的实验条件，尚缺少的实验条件和拟解决的途径，包括利用国家实验室、国家重点实验室和部门重点实验室等研究基地的计划与落实情况）；  本项目依托贵州大学人工智能与软件新技术实验室，完整地配备了高性能工作站及工作平台：包括一台高性能的电脑和一个服务器。这些为项目的顺利完成提供了充分的保障。  3.正在承担的与本项目相关的科研项目情况（申请人正在承担的与本项目相关的科研项目情况，包括国家自然科学基金的项目和国家其他科技计划项目，要注明项目的名称和编号、经费来源、起止年月、与本项目的关系及负责的内容等）；  4.完成国家自然科学基金项目情况（对申请人负责的前一个已结题科学基金项目（项目名称及批准号）完成情况、后续研究进展及与本申请项目的关系加以详细说明。另附该已结题项目研究工作总结摘要（限500字）和相关成果的详细目录）。  **（三）其他需要说明的问题**  1.本项目与已承担项目或博士论文的关系（凡以各级各类项目或博士学位论文、博士后出站报告为基础申报的课题，须阐明已承担项目、学位论文或报告与本课题的联系和区别）。  本项目以本人博士论文为基础，从以下几个方面对博士论文做了延申：   1. 研究μ-演算下如何遗忘变元。μ-演算中包括两种变元：自由变元和约束变元。之前的研究工作中，只探索了如何遗忘原子命题，并未涉及变元的遗忘。本项目将探索如何遗忘公式中的变元及相关性质。 2. 探索更多遗忘存在的子类。在博士论文中，探索了部分特殊的、遗忘存在的子类，本项目将探索更一般的、遗忘存在的子类。公式越特殊，其表达能力越弱。因此，探索更一般的子类有利于表达能力的提升，使之使用更多的场景。 3. CTL和μ-演算下，遗忘和WSC（SNC）是否存在这一问题的可判定性。这一问题目前博士论文还未涉及到，探究该问题有助于遗忘的计算。 4. 遗忘与反应式系统的WSC和SNC的关系如何？博士论文中探讨了遗忘可以计算WSC（SNC），即：当遗忘存在时，其就是一个WSC。但是，没有探索WSC（SNC）的存在反过来也可以计算遗忘。   2.其他。 |

四、项目资金预算表

|  |  |  |
| --- | --- | --- |
| **序号** | **科目名称** | **金额** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |
| **6** |  |  |
| **7** |  |  |
| **8** |  |  |
| **9** |  |  |
| **10** |  |  |
| **11** |  |  |
| **12** |  |  |
| **13** |  |  |
| **14** |  |  |
| **15** |  |  |
| **16** |  |  |
| **项目直接费用合计** | |  |

1. https://www.reactivemanifesto.org/zh-CN [↑](#footnote-ref-1)