北京大学

博士研究生学位论文

题目：遗忘及应用

姓　　名：	徐岱
学　　号：	10801830
院　　系：	数学科学学院
专　　业：	应用数学
研究方向：	人工智能
导师姓名：	林作铨 教授

二〇一一年四月

版权声明

摘　要

遗忘是与信念修正、知识更新和信念归并等并列的知识管理形式，它在简化推理、解决矛盾、逻辑程序、语义网和逻辑独立性中有重要作用。已有的变量遗忘和文字遗忘，只是过滤要遗忘的变量或文字，并不能真正模拟人类的遗忘，这是由于我们遗忘的知识用逻辑表示为公式或者语句。为此我们提出公式遗忘来捕捉这种更一般的知识变化，本文的主要工作是探索了公式遗忘的几个定义及在归并和逻辑独立性中的作用，主要体现在以下几个方面：

(1) 提出五个公设来规范公式遗忘应具有的行为，这些假设的合理性在于符合人们遗忘的直观。然后给出三种公式遗忘的定义：第一，由于公式是由变量或文字组成的，最朴素的想法是等价于遗忘这个公式中出现的符号。接着给出变量或文字最简公式，在最简公式的条件下，这种定义满足语法独立性，从而满足所有的公设。这种直接把公式遗忘转化为已有遗忘的方式尽管简单，但是不能很好地区分公式的形式。第二，从语义角度上，基于极小化准则，将模型对文字的转换推广到模型对公式的改变，而后者通常是模型的集合。这种遗忘包含文字遗忘作为其特例，并满足遗忘公设和析取分配律等。第三，基于遗忘是对知识认知不确定的过程，提出边缘公式和边缘模型的概念，进而选择部分模型作为遗忘的新模型。这种定义也满足遗忘公设。

(2) 通过考察已有遗忘和原公式本原蕴涵式集合的关系，研究推广这种关系到公式上来定义遗忘。既然本原蕴涵式是子句形式，所以先考虑子句遗忘。给出了子句距离定义来规定保留或删除哪些子句，定义了本原蕴涵式集合的消解闭包来执行操作，通过遗忘集就可以顺利定义子句遗忘。有了子句集遗忘就可以得到公式遗忘就是遗忘本原蕴涵式的集合。文中证明了公式遗忘满足所有公设，并且在特殊条件下，遗忘子公式可以将其看做一个整体。

(3) 在讨论遗忘应用时，一方面，考察了已有归并在矛盾推理过程中有时并不直观，为此提出两个新的归并性质。通过研究变量遗忘和已有归并算子的关系，适当修正提出两个新的算子，且都满足提出的性质。另一方面，通过遗忘定义了公式之间的强独立性、弱独立性，强相关性、弱相关性。最后指出遗忘和其他知识管理形式的异同。

关键词：遗忘，知识管理，信念修正，知识更新，信念归并

Forgetting and Its Application

Dai Xu (Applied Mathematics)
Directed by Professor Zuoquan Lin

Forgetting is one method of knowledge management paralleling with belief revision, knowledge update and belief merging, which plays an important role in simplifying reasoning, conflict resolution, logic programming, semantic networks and logical independence. The existing variables and literals forgetting only filter the variables or literals to be forgotten, and can not really simulate the human's forgetting, which is because the knowledge we lost is expressed as a logical formula or sentence. So we propose formula forgetting to capture this more general knowledge changes. This paper mainly explores several definitions of formula forgetting and investigates its application in merging and logical independence. The main contributions include the following:

(1) We propose five postulates of formula forgetting to regulate its behavior, and they are reasonable because of coincidence with the intuition. Then three definitions are given: First, the most simple idea is equivalent to forget the symbols appeared in the formula because the formula is composed by variables or literals. Then we give the variables-simplest or literals-simplest formula, under the condition of simplest formula, this definition is syntax-free and so meets with all postulates. Although the way directly translating into the existing operators is simple, but can not distinguish the forms of formulae. Second, from the semantic point of view, according to criteria based on minimization, the conversion of models to literal is promoted to change of model to formula but the latter is usually a collection of models. This forgetting includes literal forgetting as its special case and satisfies all postulates and distributive law of disjunction. Third, based on forgetting is the process of uncertainty of knowledge cognitive, we give the concepts of margin formula and models, and then select part of the models as new ones of forgetting. This definition also satisfies the postulates.

(2) By examining the relationship between forgetting and the prime implicates set of the original formula, we extend this relationship to define formula forgetting. Since the prime implicate is indeed a clause, so we first give the clause forgetting. We define the distance between clauses to decide which one to delete or keep, the resolution closure of prime implicates set to perform the operation, and forgetting set to give clause forgetting. We give clauses forgetting before considering

formula forgetting as forgetting the prime implicates set. The paper proves that the formula forgetting satisfies all postulates and in special case, a sub-formula can be treated as a whole in forgetting.

(3) In the discussion of applications, on the one hand, we investigate the existing merging operators are not always intuitive in contradictory reasoning, then propose two new properties for merging. By studying the relationship between variables forgetting and merging, we revise it to get two new operators satisfying the proposed properties. On the other hand, we propose by forgetting the strong and weak independence, the strong correlation and weak correlation. Finally, we compare forgetting with other methods of knowledge management.

Key Words: forgetting, knowledge management, belief revision, knowledge update, belief merging

目 录

插图

第一章 绪 论

1.1 背景

知识管理 [1, 2, 3, 4, 5]的任务涉及消除过期的知识和增加新的知识，有很多方法完成这个任务，包括信念改变（belief change） [6, 7]－信念修正（revision） [8, 9, 10]和收缩(contraction) [11, 12, 13, 14, 15]，知识更新（update）或擦除（erasure） [16, 17, 18, 19, 20]，信念归并（belief merging） [21, 22, 23]和遗忘（forgetting） [24, 25, 26, 27]。尽管他们都属于知识管理的范畴，但是不同的概念。信念改变、知识更新和归并这三类操作的对象是知识，而遗忘操作的对象是表达知识的语言。

知识更新和信念修正尽管看起来差不多，都是由于加入新知识导致与旧知识不协调，以及怎样消除这种矛盾以便在推理中能够采用经典逻辑方法。其实，它们有着本质的区别，知识更新描述的是动态世界的变化，通常应用于数据库与知识库，比如，增加张三5%的工资，这样数据库中张三原先的工资就应舍弃，发生了变化，涨了5%。再比如，由于机器人行动所带来的知识变化，机器人拿起一个苹果，机器人走到另一个房间等都描述了机器人的状态发生了变化，原先的知识随机器人的行动可能不再为真。知识更新是我们最经常碰到的情况，因为社会就是不断变化发展的。相反，信念修正描述的是静态世界，假设我们在诊断一个有问题的数字电路，它的毛病不知道出在哪里，但是确定在某处。随着我们不断进行测试，后面的测试结果可能和前面的矛盾，但是这个电路的状态并没有变化。再如，就目前情况我们认为月球上没有水，也许在不久将来，随着登月的次数增多，也许我们会发现水。尽管我们的知识发生变化，但是月球到底有没有水是确定的。从这个角度看，修正的是原先错误的信息，更新的是过时的知识。

我们的信息可能来自于多个信息源，通常综合这些信息源的知识是不协调的，因此我们不能简单把它们放到一起做推理，这就是知识归并要解决的问题。当然每一个信息源的知识看做一个知识库，本身是协调的。归并的问题就是解决冲突的过程，可以把要归并的知识库看作是同等的，也可以看做带权的或者分层的（ [28, 29, 30, 31, 32]），这依赖于不同的应用背景。带完整性约束的归并是附加一个完整性约束条件（通常也是一逻辑公式），归并结果必须要满

足此条件。归并的算子除了基于语法的（一般是取极大协调子集），主要是基于语义的，就是按照某种合理偏序，选择最小的模型作为我们的结果。

尽管遗忘最早可以追溯到布尔提出的排除中间项 [33]，但直到文献 [24]才引入到人工智能领域里。引入遗忘概念的动机是在控制自主机器人时，假设机器人有一个表示它最初状态的知识库，随着机器人执行动作，它必须忘记不再为真的知识以更新知识库，使它们不会影响到机器人以后的行为。遗忘的形式很多，最基本的是变量遗忘 [24]，即我们的遗忘结果不包含任何遗忘的变量。这种遗忘是过滤掉和给定查询（通常是由论域内变量子集建立的）无关的信息，它可以应用在推理、解决矛盾、逻辑程序、语义网和命题独立性（ [34, 35, 36, 37, 38, 25, 39, 40, 27]）。随后变量遗忘被推广到文字遗忘 [25]，形式上是遗忘的结果不包含任何遗忘的文字，但可能包含其负文字，也就是与遗忘文字相关的变量仍在结果中，所以这种遗忘不是消除变量，而是消除文字。文字遗忘提出的动机是考虑信息的一极是有意义的，即我们可能描述某个文字的信息但没有关于其否定的信息，比如人工智能领域的封闭世界和行动推理。这个概念还被推广到带可变命题符号的文字遗忘 [41]，它是在遗忘文字时附加一个可变的命题变元的集合，这种形式上的变化可以把求解限定的两步骤变成一步。上面说的遗忘都是在命题逻辑中，在模态逻辑中也给出了知识遗忘的概念，并类似信念修正给出了表示定理 [42, 26]。文献 [40, 43]给出了逻辑程序中的强遗忘和弱遗忘，但是由于这种遗忘是把逻辑程序转化为约简形式然后删除某些规则，所以它对于答集语义是语法相关的并且这种变换和语义的关系也不明确。文献 [39, 44]从答集语义的角度基于极小答集定义了遗忘，而且给出了与其他遗忘形式的关系。此外，遗忘还被推广应用到描述逻辑中（ [27, 45, 35]）。

1.2 问题提出

遗忘概念不管是在命题逻辑，还是在模态逻辑和逻辑程序等，不管是变量遗忘还是文字遗忘，他们都是对符号的遗忘，即遗忘的结果（公式或逻辑程序）中不能包含某些变量符号或者文字符号。正像人类的遗忘一样，日常生活中我们遗忘的通常是一件事情或者一条信息，而在知识表示中这通常会用逻辑公式表示，而不是单纯的一个原子。

我们从小到大经历了无数考试，在标准化考试中通常会有选择题，包括单选题、多选题及不定项选择题。单选题的意思是有且仅有一个正确答案，忘记这一点很有可能做错。假设每一题有4个选项，我们用命题a，b，c，d分别表

示A，B，C，D四个选项为正确答案。那么此题是单选可以形式化为

$$\phi = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge d)$$

在不仔细观察题意前提下，一些学生对于这种单选题通常会给出多个选项的答案。一方面是他们对其中两个或更多个选项把握不准，感觉都对；另一方面是没有注意此题是单选题，也就是满足我们上面给出的逻辑公式，所以这些学生实际上是把单选题当成了多项选择题。实质的原因是他们忘记了这个事实：不可能有两个选项同时为真，用逻辑公式表示为

$$\psi = \neg((a \wedge b) \vee (a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d) \vee (c \wedge d))$$

这种形式的遗忘过程，用$Forget(\phi, \psi)$表示，即我们现在不再是遗忘原子或文字（集合），而是遗忘一个公式。即在遗忘算子$Forget(\phi, \psi)$中，两个变元都是公式。尽管在 [46]中，作者主要分析了遗忘和知识更新（对于特定距离）的关系，并在文章结论部分提出了未来会考虑遗忘算子的第二个输入是一个语句（公式）的情况，但是目前还没有关于这种遗忘的定义。他们的出发点是既然遗忘和知识更新可以相互定义，且知识更新的输入一般就是一个语句，那么为什么不能考虑遗忘一个公式的情况呢？

本文正是基于这种考虑，探索公式（语句）遗忘的概念，从语义和语法上都定义了遗忘算子。对于上面这个例子，按照我们基于本原蕴涵式的遗忘定义，$Forget(\phi, \psi) \equiv a \vee b \vee c \vee d$。与变量遗忘的区别是，$ForgetVar(\phi, a)$代表忘记选项$A$的真假，$ForgetVar(\phi, b)$代表忘记选项$B$的真假，$ForgetVar(\phi, c)$代表忘记选项$C$的真假，$ForgetVar(\phi, d)$代表忘记选项$D$的真假，而$Forget(\phi, \psi)$表示遗忘掉这道题不可能有多个选项。

另一方面，假设我们现在有两个对象A和B，建立在变量集V_1和V_2上公式分别用来描述这两个对象。如果我们遗忘V_1，那么我们对A一无所知；如果遗忘V_2，那么对B一无所知。通常情况下，对象A和B是交互的，它们是相互作用的。而假设我们现在只关心对象B，那么变量遗忘显然不能满足要求，若遗忘V_1，则A和B交互的信息也丢失了，这显然不是我们期望的。

1.3 方法

在知识管理研究中，很大一部分工作都是以A.G.M信念修正公设 [47, 48]为出发点的。公设的意思是算子预期应该满足的性质，我们不能给出一个标准算

子，所以用公设来规范它们的行为。针对信念修正提出了6条公设，比如修正结果应该包含新来的知识，在什么情况下修正就是单纯的扩张，修正算子应满足语法独立性以及用一个合取公式去修正应该得到什么结果等；信念收缩有5个公设，规定收缩之后推理能力肯定比原先要弱（因为收缩就是要去掉某些信息），何时收缩算子不产生作用（与原先公式等价），收缩结果不蕴含要收缩掉的公式，收缩满足语法独立性及收缩的恢复性（收缩之后再添加，结果不变）等；知识更新满足8条公设，即蕴含新信息，如果原知识库蕴含新知识则更新不起作用，语法独立性，合取公式的更新结果如何，用两个公式分别去更新的相互作用，用一个公式去更新析取公式满足分配律等；知识擦除满足6个公设，即擦除是对原公式的弱化，擦除算子何时没有影响，擦除结果不蕴含被擦出的公式，语法独立性，擦除的恢复性及关于析取的分配律等；带完整性约束的归并满足9条公设，归并结果必须逻辑推出此完整性约束（即是必须被满足的），如各知识库整体并不与约束矛盾，那么归并是平凡的，语法独立性，归并不偏向于任何某个知识库，两个知识集并起来的归并与分开归并的关系，两个完整性约束情况下归并的行为如何。

从上面可以看出，所谓的公设无非是对算子行为的合理预期，必须满足人们的直观，既然还不能给出一个标准算子，那么就考虑在特定条件下各算子如何表现。另一方面，这些公设有相同或者假设。比如，他们都满足语法独立性，因为我们更多的关注语义，而不管用什么方式来表达；比较了算子作用前后知识库的变化（是否比原先更弱）；什么情况下相应的算子不产生作用等。我们将给出公式遗忘的公设，也循着这个框架。公式遗忘也应该满足语法独立性，遗忘之后应该比先前知识库要弱等，第三章将给出比较合理的遗忘公设。

对于信念修正、知识更新和归并，已经给出了各种算子，最重要的是基于语义的定义。语义实际上操作的是模型，就是模型选择和抛弃的过程。知识更新就是保留新知识且对原有知识库做尽可能小的改变，在新知识的模型中依次选择与每个旧知识库模型最近的作为结果。用距离来衡量远近，有了两个模型的距离（最著名的是Dalal距离）可以定义一个模型和一个知识库（用逻辑公式表示）的距离，进而还可以定义模型和知识集（知识库的集合，通常为公式的集合）的距离。知识擦除肯定比原先弱并且确实要删除信息，本着这两个原则，结果一方面肯定包含原先知识库的所有模型，另一方面在要擦除公式否定的模型中，依次选择与原先知识库每个模型最近的放在结果模型集中。信念修正与知识更新类似，但它选择的是与旧知识库最近的模型，而不是与它的每个

模型最近。信念收缩与知识擦除的区别与这类似。知识归并面对的是多个知识库，所以是在约束公式的模型中，选择与多个知识库整体距离（一般来说有3个这种距离）最小的模型。

对于公式遗忘，当然可以从模型角度来考虑。既然遗忘是对公式的弱化且被遗忘公式是未知的，那么一种方式类似于文字遗忘，首先把旧知识库的模型放到结果集中，其次因为遗忘了就不再被蕴含，所以对于每个旧知识的模型，寻找与其最近且使被遗忘公式为假的模型放到结果集中。另外一种考虑是遗忘使得被遗忘的知识处于最不稳定的状态，即我们并不知道其真假，而且对模型做比较少的改变就可以使真假值发生变化。因此我们定义公式边界的概念，这样遗忘就是选择旧知识模型和被遗忘公式边界的模型作为结果。

最后一种方法是，既然文字是最简单的公式，那么公式遗忘应该是文字遗忘的自然推广。考虑到变量或文字遗忘都可以转化为操纵其本原蕴涵式集合，即删除包含遗忘变量或文字的本原蕴涵式，剩余的就是我们的遗忘结果。由于遗忘和本原蕴涵式有着紧密的关系，而且本原蕴涵式本身是子句形式，在公式遗忘前先定义子句遗忘。子句遗忘就是去掉原公式与该子句相关的本原蕴涵式。因为每个命题公式都等价于其本原蕴涵式的合取，那么公式遗忘可以转化为子句集的遗忘。事实证明，这种遗忘概念具有很好的性质，符合人们的直观感觉。

1.4 主要结果和本文结构

本文把遗忘推广到一般的公式遗忘，首先给出五个公设表征遗忘算子应满足的性质，他们是：遗忘是对原知识库的弱化，遗忘不影响遗忘变元之外的其他符号，遗忘结果不蕴含被遗忘的公式，遗忘结果与公式的语法形式无关，多次遗忘同一公式与遗忘一次结果一致。接下来，我们给出遗忘的四类定义，第一种定义基于直接转化为变量或文字遗忘，后两种是从语义角度，最后一种基于本原蕴含式的遗忘是从语法角度。我们证明了给出的四类定义（或者稍加改进）都满足给定的遗忘公设，并且是已有文字（变量）遗忘的推广。在满足特殊条件下，基于本原蕴含式的公式遗忘可以把遗忘公式看做一个整体作遗忘，从而转化为文字遗忘。

由于遗忘是对公式的弱化，所以它在解决冲突时有重要作用。已有的归并算子在解决问题时有时并不直观，为此我们提出两个新的性质；部分知识库支

持某原子公式，其他知识库不出现此原子，那么该原子公式应该在归并结果中；一些知识库支持原子公式，另一些反对，那么对该公式保持中立。通过揭示遗忘和已有归并的关系，修改后得到两个新的归并算子，他们都满足新性质。

本文组织如下：第一章是绪论，介绍研究背景和方法。第二章介绍已有的遗忘概念，包括变量、文字遗忘、带可变命题符号的文字遗忘、模态逻辑中的知识遗忘、一阶逻辑中的谓词遗忘和答集逻辑程序中的遗忘。我们在第三章给出遗忘的五个公设和三类遗忘的定义，第四章给出基于本原蕴含式的遗忘，首先给出子句遗忘的概念。第五章探讨遗忘的应用，用遗忘定义新的归并算子和通过遗忘考察公式之间的独立性。第六章比较了遗忘与其他知识管理形式的异同，最后给出总结和展望。

第二章　遗忘的概念

本文主要讨论命题逻辑下的遗忘，它是最简单也是最经典的逻辑系统，约定用1或T表示真，0或F表示假，$\top$代表永真式，$\bot$代表永假式，用除T或F的英文字母表示命题。逻辑连接词$\neg, \vee, \wedge, \rightarrow, \leftrightarrow$作经典解释。因为$\{\neg, \vee\}$，$\{\neg, \wedge\}$都是功能完备集，即所有其他连接词都可以等价表示成他们，在随后的给出的命题公式，我们一般只包含$\neg, \vee, \wedge$这三个连接词。

通常用大写字母或希腊字母$\varphi, \psi, \Sigma, \phi$等（也可能带下标）表示公式。

使公式为真的一组变量真值指派称为公式的一个模型或解释，$Mod(\varphi)$表示公式φ的所有模型构成的集合。

原子命题公式或它的否定称为文字（literal），也成为原子公式或文字公式。有限个文字的析取式称为一个子句（clause），有限个文字的合取式称为一个项（term）。

我们用$Var(\varphi)$或者$Atoms(\varphi)$表示公式φ出现的命题变元的集合，$Lit(\varphi)$表示公式φ中出现文字的集合；对于任意集合V，$|V|$表示其包含元素的个数；对于两个集合V_1, V_2，集合差集$V_1 - V_2$代表属于集合V_1但不属于V_2的元素构成的集合。

$\vdash$表示语法推出，$\models$表示语义推出，对于公理演绎系统和自然推演系统，这两个推出是等价的。我们今后主要用$\models$表示逻辑蕴含或推出。

对于像“小王是人，人都是会死的，所以小王是会死的”这样的推理，命题逻辑无法表达，由此引出一阶逻辑。一阶语言的符号包括以下几类：

(1) 个体变元如x，y，z等
(2) 函数符号（函词）如f，g，h等，个体符号（表示论域中的个体）用a，b，c等表示，谓词（表示关系）用P，Q，R等表示，其中有一个二元谓词=，称为等词（表示恒同关系）。
(3) 命题联结词有$\neg, \vee, \wedge, \rightarrow$和存在量词$\exists$（存在某个个体）、全称量词$\forall$（对于任意个体）。

一阶语言的项归纳定义如下：

(1) 变元和个体符号是项。
(2) 若$t_1, t_2, \cdots, t_n$是项，f是一个n元函数符号，则$f(t_1, t_2, \cdots, t_n)$也是项。

一阶公式归纳定义为：

(1) 若$t_1, t_2, \cdots, t_n$是项，P是n元谓词符号，则$P(t_1, t_2, \cdots, t_n)$是公式，也称为原子公式。
(2) 若A是公式，则$\neg A$是公式；若A，B是公式，则$A \wedge B, A \vee B, A \rightarrow B$都是公式。
(3) 若A是公式，则$(\exists x)A(x)$和$(\forall x)A(x)$都是公式。

谓词演算作为一个形式系统，可以规定它的解释。给定一个论域，对于谓词演算中出现的个体符号、函数符号及谓词依次解释为论域中的个体及定义在此论域上的函数及关系。此论域及其对于谓词演算中形式符号的解释称为该演算的一个结构或模型。由对于个体符号和函数符号的解释可知，项可解释为复合函数，它指称个体。原子公式$P(t_1, t_2, \cdots, t_n)$解释为$t_1, t_2, \cdots, t_n$所指称的个体满足n元关系P。若公式$A(x)$表示关系，则$(\forall x)A(x)$解释为论域中所有个体满足关系A，$(\exists x)A(x)$解释为论域中存在某个体满足关系A。

“所有个体”、“存在个体”中，量词加在论域的个体上，称为一阶量词。在一阶逻辑中使用的量词仅限于一阶量词。“所有函数”、“存在函数”、“所有关系”和“存在关系”是二阶量词。此外还有更高阶的量词。相应地也有二阶逻辑、高阶逻辑等。

下面考虑命题模态逻辑$S5$，其递归定义如下：

$$\phi ::= \perp | p | \neg\phi | \phi supset \psi | K\phi$$

。这里p是原子，K是模态词，表示知道。$\top$，$\phi \vee \psi$，$\phi \wedge \psi$按标准方式定义。不包含模态词的公式称为对象公式（objective formulas）。

一个Kripke结构（Kripke structure）是三元组$S =< W, R, L >$，这里W是可能世界集合，每个可能世界是某些原子的集合，R是W上的等价关系，L是W中世界解释的集合。一个Kripke解释是二元组$M =< S, w >$，这里$w \in W$。$S5$ Kripke解释可以简化为$M =< W, w >$，w是实际世界，并称$M =< W, w >$为$k-$解释。

在$k-$解释和公式之间的满足关系$\models$递归定义如下：

- $\langle W, w\rangle \not\models \bot$；
- $\langle W, w\rangle \models p$ 当且仅当$p \in w$；
- $\langle W, w\rangle \models \neg\varphi$ 当且仅当$\langle W, w\rangle \not\models \varphi$；
- $\langle W, w\rangle \models \varphi \supset \psi$ 当且仅当$\langle W, w\rangle \not\models \varphi$ 或者$\langle W, w\rangle \models \psi$；
- $\langle W, w\rangle \models K\varphi$ 当且仅当$\forall w' \in W$，都有$\langle W, w'\rangle \models \varphi$。

M 是公式φ 的一个$k - model$ 当且仅当$M \models \varphi$。$Mod(\varphi)$ 表示公式φ 的所有k–模型。

假设w, w' 是两个世界，用$w \simeq_V w'$ 表示对于所有不在V 中的原子p，满足$p \in w$ 当且仅当$p \in w'$。

答集程序设计(answer set programming)的理念是用逻辑程序对问题进行编码，其中逻辑程序的答集对应于问题的解，然后通过答集求解器计算答集。两个可用的答集求解器是SMODELS [1] 和DLV [2]。关于逻辑程序特别是答集逻辑程序更详细的细节可参考 [49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]

我们简单回顾一下答集逻辑程序中的基本概念。一个析取逻辑程序是一个如下规则的有限集合：

$$a_1 \vee \cdots \vee a_s \leftarrow b_1, \cdots, b_m, notc_1, \cdots, notc_n (1)$$

其中$s, m, n \geq 0$并且a_i, b_j 和c_k 都是经典的命题文字。

给定一个具有(1)形式的规则r，我们有$head(r) = a_1 \vee \cdots \vee a_s$，$body(r) = body^+(r) \cup notbody^-(r)$，这里$body^+ = \{b_1, \cdots, b_m\}$ 和$body^- = \{c_1, \cdots, c_n\}$。

逻辑程序的答集语义定义如下。给定一个解释X，P 在X 上的规约被定义为$P^X = \{head(r) \leftarrow body^+(r) \mid body^-(r) \cap X = \emptyset\}$。如果$X$ 是P^X 的最小模型，那么X 是P 的答集。我们用$\mathcal{AS}(P)$ 表示P 的答集的集合。

例 2.1: 假设逻辑程序P 由下面3条规则组成：

$$a \vee b \leftarrow notc$$

$$d \leftarrow a$$

[1] http://www.tcs.hut.fi/Software/smodels/
[2] http://www.dbai.tuwien.ac.at/proj/dlv/

$$d \leftarrow b$$

那么P 有两个答集$X_1 = \{a, d\}$ 和$X_2 = \{b, d\}$，显然X_1, X_2 是不可比的。

2.1 变量遗忘

给定一个理论T 和一个基原子p，我们现在定义在T 中遗忘p。直觉上，结果理论应该比最初的理论更弱并且蕴含同样的与p 无关的句子。

为了给出遗忘的语义定义，先定义一阶语言$\mathcal{L}$ 上结构的等价关系。假设$P(\overrightarrow{t})$ 是$\mathcal{L}$ 的基原子，M_1 和M_2 是$\mathcal{L}$ 上两个一阶结构，定义$M_1 \sim_{P(\overrightarrow{t})} M_2$ 当且仅当除了$P(\overrightarrow{t})$ 的真值，M_1 和M_2 对其他解释都一样：

1. M_1 和M_2 定义域相同且对函词解释相同。

2. 对不同于P 的谓词符号Q，$M_1[Q] = M_2[Q]$。

3. 假设$\overrightarrow{u} = M_1[\overrightarrow{t}]$, 对于任何不同于$\overrightarrow{u}$ 的元组$\overrightarrow{d}$ 都有$\overrightarrow{d} \in M_1[P]$ 当且仅当$\overrightarrow{d} \in M_2[P]$。

定义 2.1: [24] 假设T 是一个理论，p 是一个基原子。理论T' 称为在T 中遗忘p 的结果，如果对于任意结构M，M 是T' 的模型当且仅当有一个T 的模型M' 满足$M \sim_p M'$。

显然，如果T' 和T'' 都是T 遗忘p 的结果，那么T' 和T'' 是逻辑等价的，说明在逻辑等价下遗忘的结果是唯一的。接下来，我们用$ForgetVar(T, p)$ 表示T 遗忘p 结果，下面是遗忘算子的几个重要性质。

命题 2.1: （ [24]）对于任意理论T和基原子p，有$T \models Forget(T, p)$。

上述性质反映了人类遗忘的本质，当遗忘某些知识后，他们所知道的东西比原先少。也就是遗忘后的结果理论比原先要弱，结果理论知道的东西，原先的理论也知道，但反之不一定成立。

一个重要的结论是，对于有限理论T 和基原子p，$ForgetVar(T, p)$总是存在的且可以只通过语法操作来得到。在给出这个结论前，需要作如下说明。

假设φ 是一个公式，$P(\overrightarrow{t})$ 是一个基原子。用$\varphi[P(\overrightarrow{t})]$ 表示在φ 中，用$[t = t' \wedge P(\overrightarrow{t})] \vee [t \neq t' \wedge P(\overrightarrow{t'})]$ 取代每一个$P(\overrightarrow{t'})$. 显然，φ 和$\varphi[P(\overrightarrow{t})]$ 是等价

的。用$\varphi^{+}_{P(\overrightarrow{t})}$表示在$\varphi[P(\overrightarrow{t})]$中用$true$代替$P(\overrightarrow{t})$的结果，类似地，$\varphi^{-}_{P(\overrightarrow{t})}$表示在$\varphi[P(\overrightarrow{t})]$中用$false$代替$P(\overrightarrow{t})$的结果。

比如，假设$\varphi = student(John) \vee student(Joe) \vee teacher(John)$，$P(\overrightarrow{t}) = student(John)$。那么，$\varphi[P(\overrightarrow{t})] \equiv [John = John \wedge student(John)] \vee [John \neq John \wedge student(John)] \vee [John = Joe \wedge student(John)] \vee [John \neq Joe \wedge student(Joe)] \vee teacher(John)$。

因此，$\varphi^{+}_{P(\overrightarrow{t})} \equiv [John = John \wedge true] \vee [John \neq John \wedge true] \vee [John = Joe \wedge true] \vee [John \neq Joe \wedge student(Joe)] \vee teacher(John) \equiv true$。

类似地，$\varphi^{-}_{P(\overrightarrow{t})} \equiv [John \neq Joe \wedge student(Joe)] \vee teacher(John)$。

对于任何协调的理论$T = \{\varphi_1, \cdots, \varphi_n\}$，每一个$\varphi_i$都是公式，那么$T \equiv \{\varphi_1 \wedge \cdots \wedge \varphi_n\}$。因此，我们可以只考虑单公式构成的理论。还可以用$\exists p.\varphi$表示在公式$\varphi$中遗忘原子$p$，同样地，$\exists V.\varphi$表示在公式$\varphi$中遗忘原子集合$V$。

命题 2.2: ([24])

对于任何理论$T = \{\varphi\}$，p是一个基原子，那么

$$ForgetVar(T, p) = \{\varphi^{+}_{p} \vee \varphi^{-}_{p}\}$$

上面这个性质告诉我们，变量遗忘肯定是存在的，并且在逻辑等价性下结果是唯一的。

例 2.2: 假设$T_1 = \{student(John) \vee student(Joe) \vee teacher(John)\}$，那么

$Forget(T_1, student(John)) = \{\top\}$。

假设$T_2 = \{\varphi\} = \{(\exists x).student(x)\}$，既然$\varphi(student(John)) = (\exists x).(x = John \wedge student(John)) \vee (x \neq John \wedge student(x))$。那么$\varphi^{+}_{(student(John))}$等价于$true$，从而$ForgetVar(T_2, student(John)) = \{true\}$。

假设$T_3 = \{\varphi\} = \{(\forall x).student(x)\}$, $\varphi(student(John)) = (\forall x).(x = John \wedge student(John)) \vee (x \neq John \wedge student(x))$。所以$\varphi^{+}_{(student(John))} \equiv \varphi^{-}_{(student(John))} \equiv (\forall x).x \neq John \wedge student(x)$，因此$ForgetVar(T_3, student(John)) = (\forall x).x \neq John \wedge student(x)$。

对于命题公式里的遗忘，情况会更简单。因为$\varphi[p] = \varphi$，所以$ForgetVar(\varphi, p) = \varphi(p/true) \vee \varphi(p/false)$。其中$\varphi(p/true)$和$\varphi(p/false)$分

别表示在公式φ中用$true$或者$false$取代每一个p所得结果。并且既然理论可以看做其公式集合的合取，所以我们有时用一个公式表示一个理论。

例 2.3: 如果公式$\varphi_1 = p \wedge q$，那么$ForgetVar(\varphi_1, p) = \varphi_1(p/true) \vee \varphi_1(p/false) = (true \wedge q) \vee (false \wedge q) = q$；如果公式$\varphi_2 = p \vee q$，那么$ForgetVar(\varphi_2, p) = \varphi_2(p/true) \vee \varphi_2(p/false) = (true \vee q) \vee (false \vee q) = true = \top$。

到目前为止，我们只给出了单个原子的遗忘，接下来我们定义原子集合上的遗忘。假设$p_1, \cdots, p_n$是基原子序列，那么$Forget(\varphi, \{p_1, \cdots, p_n\}) = ForgetVar(ForgetVar(\varphi, \{p_1, \cdots, p_{n-1}\}), p_n)$。很容易得出，遗忘与基原子序列的顺序无关，我们有下面的性质。

命题 2.3: ([24]) 对于任意理论T和任意基原子p_1, p_2，有$ForgetVar(ForgetVar(T, p_1), p_2)$和$ForgetVar(ForgetVar(T, p_2), p_1)$是逻辑等价的。

正是由于上述性质，我们才能定义原子集合上的遗忘。

既然遗忘后的结果比原先的理论更弱，实际上，在不考虑遗忘变量集的条件下，它是原先理论的最强的必要条件。

命题 2.4: 假设理论T和变量集V，$Var(T)$表示T中出现的原子变量的集合。对于任何一个由$Var(T) - V$中变量构造的公式φ，如果$T \models \varphi$，那么$ForgetVar(T, V) \models \varphi$。

下面是几个推论，第一个是说遗忘的越多，知道的越少。

推论 2.1: 假设φ是协调的, $V_1 \subseteq V_2 \subseteq Var(\varphi)$, 那么$ForgetVar(\varphi, V_1) \models Forget(\varphi, V_2)$。

下一个性质描述了一种极端情况，即遗忘掉所有出现的变量之后得到逻辑常量。

推论 2.2: $V = Var(\varphi)$. 如果φ协调，那么$ForgetVar(\varphi, V) \equiv \top$；否则，$ForgetVar(\varphi, V) \equiv \bot$。

如果遗忘未出现的原子变量，结果不会发生变化。

推论 2.3: 假设$p \notin Var(\varphi)$, 那么$ForgetVar(\varphi, p) \equiv \varphi$。

从遗忘的定义可以看出，遗忘是对两个真值指派代入做析取，所以遗忘满足析取分配律。

命题 2.5: （ [25]）假设φ, ψ是两个命题公式，V是变量集，那么$ForgetVar(\varphi \vee \psi, V) \equiv ForgetVar(\varphi, V) \vee ForgetVar(\psi, V)$。

遗忘是对公式的弱化，并且这种弱化保持单调性。

命题 2.6: 如果$\varphi \models \varphi'$ 那么$ForgetVar(\varphi, V) \models ForgetVar(\varphi', V)$。

下面这个性质是一个直接推论，如果某个公式独立于遗忘变量，那么遗忘变量的结果不变，因此保持我们的推理关系。

推论 2.4: （ [25]）当$Var(\varphi') \cap V = \emptyset$ 时，那么$ForgetVar(\varphi, V) \models \varphi'$。

下一个性质说明变量遗忘的主要特征，我们通过遗忘最终得到一个不包含任何遗忘变量的公式。

命题 2.7: ([25]) 对于任意公式φ和变量集合V，有$ForgetVar(\varphi, L)$ 是φ能推出的且变量独立于V 的最强推论。

一个公式φ变量独立于V指的是存在φ的等价公式φ'，使得φ'不包含V中的变量，我们会在第五章讨论遗忘和独立性的关系。

和遗忘相对的概念是记住（remembering），记住某些知识就相当于遗忘其他的信息。假设T是一理论，$p_1, p_2, \cdots, p_k$是变量序列，$p_{k+1}, \cdots, p_n$是T中剩余的变量。我们定义在T中记住$p_1, \cdots, p_k$的就是遗忘掉其余变量的结果，即$Remember(T, p_1, \cdots, p_k) \equiv ForgetVar(T, p_{k+1}, \cdots, p_n)$。

下面我们给出比变量遗忘更广的概念：文字遗忘。

2.2 文字遗忘

2.2.1 文字遗忘

在信息的一极是有意义的情况下，文字遗忘 [25]对推理是有帮助的，它被用来区分公式传达关于某个文字的信息而没有关于它的反面的信息，诸如人工智能领域的封闭世界推理和行为推理都存在这种情况。

定义 2.2: （ [25]）假设φ 是一命题公式，L 是文字集合，$ForgetLit(\varphi, L)$ 表示在公式φ 遗忘文字集L，如下递归定义：

- $ForgetLit(\varphi, \emptyset) = \varphi$
- $ForgetLit(\varphi, \{l\}) = \varphi_{l\leftarrow 1} \vee (\neg l \wedge \varphi)$
- $ForgetLit(\varphi, \{l\} \cup L) = ForgetLit(ForgetLit(\varphi, L), l)$

上述对文字集合遗忘的定义是合理的因为遗忘与文字的顺序无关。从定义很容易得到

$$ForgetLit(\varphi, \{l\}) = \varphi_{l\leftarrow 1} \vee (\neg l \wedge \varphi_{l\leftarrow 0})$$

。

上面给出的是语法定义，为了给出文字遗忘的语义，令$Force(\omega, l)$ 表示使l 为真且与ω 距离最近的模型。

命题 2.8: （ [25]）$ForgetLit(\varphi, \{l\})$ 的模型集合可以表示为：
$Mod(ForgetLit(\varphi, \{l\})) = Mod(\varphi) \cup \{Force(\omega, \neg l) \mid \omega \models \varphi\} = Mod(\varphi) \cup \{\omega \mid Force(\omega, l) \models \varphi\}$。

文字遗忘有许多和变量遗忘相似的性质。

命题 2.9: （ [25]）$ForgetLit(\varphi, L)$ 的模型集合可以表示为：
$Mod(ForgetLit(\varphi, L)) = \{\omega \mid Force(\omega, L_1) \models \varphi, L_1 \subseteq L\}$。

例 2.4: 假设$\varphi = (\neg a \vee b) \wedge (a \vee c)$，我们有
$ForgetLit(\varphi, \{\neg a\}) \equiv \varphi_{\neg a\leftarrow 1} \vee (a \wedge \varphi) \equiv [(true \vee b) \wedge (false \vee c)] \vee (a \wedge b) \equiv (a \vee c) \wedge (b \vee c)$。

下面是文字遗忘的一个推论，它说明文字遗忘对文字集合的反单调性及保持推理的单调性。

推论 2.5: （ [25]）假设φ, ψ 是命题公式，L_1, L_2 是文字集合。

- $\varphi \models ForgetLit(\varphi, L_1)$ 成立。
- 如果$\varphi \models \psi$，那么$ForgetLit(\varphi, L) \models ForgetLit(\psi, L)$成立。
- 如果$L_1 \subseteq L_2$，那么$ForgetLit(\varphi, L_1) \models ForgetLit(\varphi, L_2)$成立。

上面第一条指的是文字遗忘也是弱化公式，第二条指出文字遗忘满足公式的单调性，最后一条指的是遗忘的越多，知道的越少，是关于遗忘集的反单调性。

下一命题是文字遗忘的主要性质，遗忘和独立性有着紧密联系。

命题 2.10: ([25]) 对于任意公式φ和文字集合L，有$ForgetLit(\varphi, L)$是φ能推出的且文字独立于L 的最强推论。

一个公式φ文字独立于L指的是存在φ的等价公式φ'，使得φ'不包含L中的文字，但是可能包含L中文字的否文字。

例 2.5: 再来看前面那个例子，我们有$ForgetLit((\neg a \vee b) \wedge (a \vee c), \neg a) \equiv (a \vee c) \wedge (b \vee c)$。尽管我们遗忘的是$\neg a$，但是结果公式中仍然包含变量$a$，即$\neg a$的否定文字。而且不存在等价于$(a \vee c) \wedge (b \vee c)$的公式且是不包含$a$的。

推论 2.6: ([25]) φ 文字独立于L 当且仅当$\varphi \equiv ForgetLit(\varphi, L)$。

推论 2.7: ([25]) 如果φ 文字独立于L，那么$\psi \models \varphi$ 当且仅当$ForgetLit(\psi, L) \models \varphi$。

现在来考虑文字遗忘的计算，因为遗忘就是析取形式，所以对于析取公式来说，文字遗忘就相当于在每个析取项中作文字遗忘。

命题 2.11: ([25]) $ForgetLit(\varphi \vee \psi, L) \equiv ForgetLit(\varphi, L) vee ForgetLit(\psi, V)$。

另一方面，在一个项中遗忘文字相当于在此项中去除文字。

命题 2.12: ([25]) 假设γ 是命题逻辑中的协调项，L是文字集，那么$ForgetLit(\gamma, L) \equiv \wedge_{l \in \gamma L} l$。

从上面两个命题看以看出，在析取范式中求解文字遗忘是多项式时间的，从每个析取项中删除L中的文字就可以了。而对于像$\Sigma \wedge \Phi$这样的合取公式情况就复杂多了，没有类似上面的结论。因为$ForgetLit(\Sigma, L) \wedge ForgetLit(\Phi, L)$是$ForgetLit(\Sigma \wedge \Phi, L)$的逻辑推论，但是反之一般不成立。

例 2.6: 假设$\Sigma = a \vee b$，$\Phi = \neg a$，那么$ForgetLit(\Sigma, \neg a) \equiv (false \vee b) \vee a \equiv a \vee b$，并且$ForgetLit(\Phi, \neg a) \equiv \top$，因此有$ForgetLit(\Sigma, \neg a) \wedge ForgetLit(\Phi, \neg a) \equiv a \vee b$。但是$ForgetLit(\Sigma \wedge \Phi, \neg a) = ForgetLit(\neg a \wedge b, \neg a) \equiv b$。

最后给出变量遗忘和文字遗忘的关系，原子对应于同时考虑其正负文字。假设

命题 2.13: φ 是一命题公式，p 是一命题原子，那么
$Forget(\varphi, p) \equiv ForgetLit(\varphi, p) \vee ForgetLit(\varphi, \neg p)$。

2.2.2 带可变命题符号的文字遗忘

带可变命题符号的遗忘 [41]是文字遗忘的一个自然的扩展，它引入的主要目的是自动计算各种机制的新方式，尤其可以用来计算限定，并把 [25]中两步步骤变成一步。使用这种机制可以把求限定的两个步骤变成一个。首先给出语义定义如下：

定义 2.3: （ [41]）假设φ 是一个公式，V 是一个命题符号集，L 是一个协调的文字集并且V 中的符号都不出现在L 中，$ForgetLitVar(\varphi, L, V)$ 表示在V 是可变符号下在φ 中遗忘L，那么$Mod(ForgetLitVar(\varphi, L, V)) = \{\omega \mid Force(\omega, L_1 \cup L_2) \models \varphi$ 这里$L_1 \subseteq L, L_2 \subseteq V^{\pm}, L_2$ 是协调的，并且$(\omega \not\models L_1$ 或者$L_2 = \emptyset)\}$。

从定义很容易看出，$ForgetLitVar(\varphi, L, V)$ 是比$ForgetLit(\varphi, L)$更弱的结果，并且有$\varphi \models ForgetLitVar(\varphi, L, V) \models ForgetLit(\varphi, L \cup V^{\pm})$。通常来说，反之是不成立的，来看下面的例子。

例 2.7: 假设$\varphi \equiv (a \vee b) \wedge (\neg a)$，那么考虑$ForgetLitVar(\varphi, \{\neg a\}, \{b\}) \equiv a \vee b$，而有$ForgetLit(\varphi, \neg a) \equiv b$，$ForgetLit(\varphi, \{\neg a\} \cup \{b\}^{\pm}) \equiv \top$。因此$\varphi \models ForgetLitVar(\varphi, L, V) \models ForgetLit(\varphi, L \cup V^{\pm})$，并且他们三者都是不相同的。

引入$ForgetLitVar$的动机是，我们想要遗忘文字集L，甚至可能要付出修改$V^{\pm}$中文字的代价。这就解释了如果我们遗忘L中至少一个文字，那么我们允许对$V^{\pm}$中文字的任何修改。但是我们并不能徒劳的只是修改$V^{\pm}$中的文字，我们的目标是尽可能遗忘L中的文字，这解释了定义中的条件$(\omega \not\models L_1$ 或者$L_2 = \emptyset)\}$。

例 2.8: 假设$P = \{a, b\}$，$V = \{c\}$，$Q = \{d\}$ 并且$\varphi = (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c \wedge d)$。通过计算很容易得到如下结果：

- $Forget(\varphi, V) \equiv (\neg a \wedge b) \vee (a \wedge \neg b \wedge d)$
- $ForgetLit(\varphi, P^- \cup V^{\pm}) \equiv b \vee (a \wedge \neg d)$
- $ForgetLitVar(\varphi, P^-, V) \equiv (a \wedge b) \vee (a \wedge \neg c \wedge \neg d) \vee (b \wedge c)$

带可变命题符号的文字遗忘有下述性质。

命题 2.14: 假设φ是命题公式，P, Q, V是两两不相交的命题符号集合，且$P \cup Q \cup V = PS$。

- $ForgetLit(\varphi, P^- \cup V^{\pm})$等价于集合$Th(\varphi) \cap X$，这里$X$是下列项析取的公式集合：$(\bigwedge P_1) \wedge (\bigwedge Q_l)$ 满足$P_1 \subseteq P$和$Q_l \subseteq Q^{\pm}$。
- $ForgetLitVar(\varphi, P^-, V)$等价于集合$Th(\varphi) \cap X$，这里$X$是下列项析取的公式集合$(\bigwedge P_1) \wedge (\bigwedge Q_l) \wedge \bigwedge_{l \in V_l}(l \vee (\vee(P - P_1)))$，这里$P_1 \subseteq P$，$V_l$和$Q_l$分别是$V^{\pm}$和$Q^{\pm}$的协调文字集。

2.3 其它遗忘

认知推理涉及在动态环境中对智能体的认知状态进行推理，但是在各种理论和方法中，总是假设智能体能够记住先前的知识，然而，总是有一些场景，我们想形式化被忽略的特定信息，例如，智能体也许并没有足够的内存来记住每一件事情（ [60]）。因此，知识遗忘是特定环境下智能体的重要行为。

已有的遗忘概念不能直接应用到模态逻辑，比如，在命题公式$T \equiv (p \rightarrow q) \wedge ((q \wedge r) \rightarrow s)$中遗忘原子$q$，得$ForgetVar(T, q) \equiv T[q/\top] \vee T[q/\bot] = (r \rightarrow s) \vee \neg p$，这种方法却不能直接用在模态逻辑中。考虑模态公式$T' \equiv \neg Kq \wedge \neg K\neg q$，如果遗忘原子$q$，用上述方法我们得到，$T'[q/\top] \vee T'[q/\bot] \equiv \bot$，结果显然不对，因为遗忘之后公式不可能是永假式。

知识遗忘的定义可以参考文献（ [61]， [62]， [26]），其语义定义可以参考文献（ [26]， [63]， [64]）。

2.3.1 知识遗忘

类似于一阶逻辑中的原子遗忘，要在模态逻辑中定义遗忘，首先要给出两个Kripke 解释关于变量集合的等价关系。假设$M = \langle W, w \rangle$ 和$M' = \langle W', w' \rangle$是两个$k$–解释，$V$ 是一原子集合。M 和M' 称为V 互模拟的（ [63]），表示为$M \leftrightarrow_V M'$，当且仅当下面条件成立：

- $w \simeq_V w'$
- $\forall w* \in W$，$\exists w*' \in W'$，满足$w* \simeq_V w*'$
- $\forall w*' \in W'$，$\exists w* \in W$，满足$w* \simeq_V w*'$

即使$M \leftrightarrow_V M'$，M 和M' 也可能有不同数目的可能世界。有了上述概念，我们现在定义知识遗忘。

定义 2.4: （ [26]）假设Γ 是一个知识集，V 是原子集合。$KForget(\Gamma, V)$ 称为在Γ 中遗忘V 的结果，如果它满足下面条件：$Mod(KForget(\Gamma, V)) = \{M' \mid \exists M \in Mod(\Gamma)$ 和$M \leftrightarrow_V M'\}$。

例 2.9: 考虑三个知识集$K(p \vee q)$，$Kp \vee Kq$ 和$K(p \wedge q)$。从上面定义很容易得出$KForget(K(p \vee q), \{p\}) \equiv \top$，$KForget(Kp \vee Kq, \{p\}) \equiv \top$ 和$KForget(K(p \wedge q), \{p\}) \equiv Kq$。

接下来给出知识遗忘的语义特征，下面的表示定理论证了上述定义的合理性。在进一步论述前，我们需要无关性的概念。

假设Γ 是一个知识集，V 是一个原子集。我们说Γ 无关于V，用$IR(\Gamma, V)$ 表示，如果存在一个知识集Γ' 满足$\Gamma \equiv \Gamma'$ 并且$Var(\Gamma') \cap V = \emptyset$。

现在给出下面4个假设：

- (W) 弱化：$\Gamma \models \Gamma'$。
- (PP)正保持性：如果$IR(\varphi, V)$ 并且$\Gamma \models \varphi$，那么$\Gamma' \models \varphi$。
- (NP)负保持性：如果$IR(\varphi, V)$ 并且$\Gamma \not\models \varphi$，那么$\Gamma' \not\models \varphi$。
- (IR) 无关性：$IR(\Gamma', V)$。

(W)是知识遗忘的实质：遗忘之后，结果知识集更弱，遗忘是对公式的弱化。$(PP), (NP)$表达了知识遗忘对于独立于遗忘变量的积极信息或者消极信息无影响。(IR)表明遗忘之后的公式与遗忘变量是无关的。实际上，上面第三个假设(NP)是多余的，因为我们有其余三个可以推导出它。假设$\Gamma' \models \varphi$，因为$\Gamma \models \Gamma'$，所以$\Gamma \models \varphi$，这与$\Gamma \not\models \varphi$矛盾，因此$(NP)$成立。

实际上这4条公设唯一表征知识遗忘算子，下面的表示定理阐述了这一点。

命题 2.15: ([26]) 假设Γ, Γ' 是两个知识集，V 是一个变量集。下列三条陈述等价：

- $\Gamma' \equiv KForget(\Gamma, V)$；
- $\Gamma' \equiv \{\varphi \mid \Gamma \models \varphi, IR(\varphi, V)\}$
- (W)，(PP)，(NP)和(IR) 四个公设成立。

下面的性质是一个与变量遗忘和文字遗忘相同的直接的推论（ [26]）。

如果$IR(\varphi, V)$，那么$KForget(\Gamma, V) \models \varphi$ 当且仅当$\Gamma \models \varphi$。

下面的命题（ [26]）说明知识遗忘是变量遗忘的自然推广，并且在某些特殊情况下可以转化为普通的变量。

命题 2.16: 假设φ 是一对象公式（不含模态词），V 是变量集。我们有 $KForget(\varphi, V) \equiv Forget(\varphi, V)$，$KForget(K\varphi, V) \equiv K(Forget(\varphi, V))$。

这个性质也说明了知识遗忘定义的合理性，变量遗忘是其一种特殊情况，下面还有一些直观的结论([26])。

命题 2.17: 假设$\Gamma, \Gamma_1, \Gamma_2$ 是3个知识集，φ_1 和φ_2 是两个公式，V 是变量集。下面结论成立：

- $KForget(\Gamma, V)$ 是可满足的当且仅当Γ 是可满足的。
- 如果$\Gamma_1 \equiv \Gamma_2$，那么$KForget(\Gamma_1, V) \equiv KForget(\Gamma_1, V)$。
- 如果$\Gamma_1 \models \Gamma_2$，那么$KForget(\Gamma_1, V) \models KForget(\Gamma_1, V)$。
- $KForget(\varphi_1 \vee \varphi_2, V) \equiv KForget(\varphi_1, V) \vee KForget(\varphi_2, V)$。
- $KForget(\varphi_1 \wedge \varphi_2, V) \models KForget(\varphi_1, V) \wedge KForget(\varphi_2, V)$。

2.3.2 遗忘一个关系

一阶逻辑是有函词、谓词和量词等构成，接下来我们考虑谓词遗忘。首先我们可以类似定义在谓词P 上的一个等价关系$\sim_P$：对于任何结构M_1, M_2，$M_1 \sim_P M_2$ 当且仅当M_1 和M_2 除了P 解释任何东西都一样。

定义 2.5: 假设T 是一个理论，P 是一个谓词。T' 称为在T 中遗忘P 的结果，如果对于任何结构M，$M \models T'$ 当且仅当有T 的一个模型M' 满足$M \sim_P M'$。

下面的结论表明求谓词遗忘并不难，但需要在二阶逻辑上考虑。

假设$T = \{\varphi\}$，P 是一个n 元谓词。那么$Forget(T, P) = \{(\exists R)\varphi(P/R)\}$。这里$R$ 是一个n 元的二阶谓词变量，$\varphi(P/Q)$ 是在φ 中用P 取代每一个Q。

例 2.10: 考虑理论$T_1 = \{student(John) \vee student(Joe) \vee teacher(John)\}$，根据上面定理，$forget(T; student) = \{(\exists R).R(John) \vee R(Joe) \vee teacher(John)\} = true$因为$(\exists R)R(John)$在二阶逻辑中是有效的。

再考虑理论$T_2 = \{(student(John) \vee student(joe)) \wedge teacher(John)\}$，我们得到$Forget(T, student) = \{(\exists R).(R(John) \vee R(joe)) \wedge teacher(John)\} = teacher(John)$。

尽管上面两个例子碰巧遗忘的结果是一阶可定义的，但是一般情况下对于一个有限的一阶理论和谓词，结果可能不是一阶可表达的。许多以前的性质同样成立。

假设T是一个理论，P和P'是两个谓词。

- $T \models Forget(T, P)$。
- 对于任意P不出现的公式φ，$T \models \varphi$当且仅当$Forget(T, P) \models \varphi$。
- $Forget(Forget(T, P), P')$和$Forget(Forget(T, P'), P)$等价。

所以我们可以定义谓词集合上的遗忘，并且遗忘的结果是唯一的，与谓词顺序无关。

2.3.3 在答集逻辑程序中遗忘

首先在逻辑程序中应用遗忘概念见文献 [40]、[43]，文章提出了强遗忘（strong forgetting）和弱遗忘（weak forgetting），即通过把逻辑程序转化为简化形式再删除某些规则。

定义 2.6: ([40]， [43]) 假设Π是一个程序，p是一原子，定义Π关于p的简化为$Reduct(\Pi, \{p\})$，如下获得：

- 对于任意满足$head(r) = \{p\}$的规则r，满足$p \in body^+(r')$的规则r'，用r''取代r'：$head(r') \leftarrow (body^+(r') - \{p\}), body^+(r), not(body^- r \cup body^-(r'))$；
- 从剩余的逻辑程序中删除规则r；

变量集的逻辑程序简化可以递归定义，假设P是变量集，则

- $Reduct(\Pi, \emptyset) = \Pi$
- $Reduct(\Pi, P \cup \{p\}) = Reduct(Reduct(\Pi, \{p\}), P)$

例 2.11: 假设$\Pi = \{a \leftarrow b, b \leftarrow a, d \leftarrow note\}$，则有

$$Reduct(Reduct(\Pi, \{a\}), \{b\}) = \{b \leftarrow b, d \leftarrow note\}$$

$$Reduct(Reduct(\Pi, \{b\}), \{a\}) = \{a \leftarrow a, d \leftarrow note\}$$

下面给出强遗忘和弱遗忘的概念。

定义 2.7: ([40]， [43]) 假设Π是一逻辑程序，p是原子，用$SForgetLP(\Pi, \{p\})$表示在Π中强遗忘p，它通过如下变化得到：

- $\Pi' = Reduct(\Pi, \{p\})$
- $\Pi' = \Pi' - \{r | r$是有效规则$\}$
- $\Pi' = \Pi' - \{r | head(r) = \{p\}\}$
- $\Pi' = \Pi' - \{r | p \in body^+(r)\}$
- $\Pi' = \Pi' - \{r | p \in body^-(r)\}$
- $SForgetLP(\Pi, \{p\}) = \Pi'$

强遗忘也可以推广到原子集合上。

- $SForgetLP(\Pi, \emptyset) = \Pi$
- $SForgetLP(\Pi, P \cup \{p\}) = SForgetLP(SForgetLP(\Pi, \{p\}), P)$

下面是弱遗忘定义。

定义 2.8: ([40]， [43]) 假设Π是一逻辑程序，p是原子，用$WForgetLP(\Pi, \{p\})$表示在Π中弱遗忘p，它通过如下变化得到：

- $\Pi' = Reduct(\Pi, \{p\})$
- $\Pi' = \Pi' - \{r | r$是有效规则$\}$
- $\Pi' = \Pi' - \{r | head(r) = \{p\}\}$
- $\Pi' = \Pi' - \{r | p \in body^+(r)\}$
- $\Pi' = \Pi' - \Pi^* \cup \Pi^\dagger$，这里$\Pi* = \{r | p \in body^-(r)\}$并且$\Pi^\dagger = \{r' | r' : head(r) \leftarrow body^+(r), not(body^-(r) - \{p\}), r \in \Pi^*\}$
- $WForgetLP(\Pi, \{p\}) = \Pi'$

弱遗忘也可以推广到原子集合上。

- $WForgetLP(\Pi, \emptyset) = \Pi$
- $WForgetLP(\Pi, P \cup \{p\}) = WForgetLP(WForgetLP(\Pi, \{p\}), P)$

例 2.12: 假设$\Pi_1 = \{a \leftarrow notb, b \leftarrow nota\}$，并且$\Pi_2 = \{a \leftarrow b, notc, a \leftarrow note, d \leftarrow a, e, e \leftarrow nota\}$，那么

$$SForgetLP(\Pi_1, \{a\}) = \emptyset$$

$$SForgetLP(\Pi_1, \{a, b\}) = \emptyset$$

$$WForgetLP(\Pi_1, \{a\}) = b \leftarrow$$

$$WForgetLP(\Pi_1, \{a, b\}) = \emptyset$$

$$SForgetLP(\Pi_2, \{a\}) = \{d \leftarrow b, e, notc\}$$

$$WForgetLP(\Pi_1, \{a\}) = \{d \leftarrow b, e, notc, e \leftarrow\}$$

这两种遗忘有连个主要缺点，一是语义不是完全清楚。更具体点就是逻辑程序的答集语义和有强、弱遗忘执行的语法变换之间的关系不清楚。第二个是对逻辑程序的非单调推理并没有给出预期的属性，尤其是，传统遗忘和这种逻辑程序中删除规则的方法之间有什么不同。而且这两种遗忘都是语法相关的。逻辑程序$P = \{p \leftarrow .q \leftarrow notp\}$和$Q = \{p \leftarrow\}$在答集语义下等价，但是$WForgetLP(P, p) = \{q \leftarrow\}$，$WForgetLP(Q, p) = \{\}$

经典逻辑的遗忘不能直接推广到逻辑程序上的遗忘，因为没有两个逻辑程序的析取，不过我们可以从逻辑程序的答集角度来定义。当我们遗忘某原子p时，显然不能直接在答集中删除p取剩下的部分。例如考虑$P = \{a \leftarrow, p \vee q \leftarrow\}$，那么$\mathcal{AS}(P) = \{\{a, p\}, \{a, q\}\}$，所以$\mathcal{AS}(P)\backslash p = \{\{a\}, \{a, q\}\}$。但是$\{\{a\} \subset \{a, q\}$，所以$\mathcal{AS}(P)\backslash p$ 不可能是一个逻辑程序的答集集合因为答集是不可比的。我们称集合X' 是X 的一个l-子集，用$X' \subseteq_l X$，如果$X'\backslash\{l\} \subseteq X\backslash\{l\}$。

一种解决方案是选择合适的极小答集，从而遗忘可以在这些极小集上产生出合理结果。

定义 2.9: ([39]， [44]) 假设P 是一个协调的逻辑程序，l 是一个文字，X 是文字的集合。

- 对于任何一个文字集合的集合$\mathcal{S}$，$X \in \mathcal{S}$ 是l-极小如果不存在$X' \in \mathcal{S}$ 满足$X' \subset_l X$。
- 逻辑程序P 的一个答集X 是l-答集，如果在集合$\mathcal{AS}(P)$ 中X 是l-极小。用$\mathcal{AS}_l(P)$ 表示P 中所有l-答集的集合。

有了极小的概念，现在准备定义答集逻辑程序的遗忘。

定义 2.10: ([39]) 假设P 是协调逻辑程序，l 是文字，逻辑程序P' 称为在P 中遗忘l 的结果，如果它满足：

- $Lit_{P'} \subseteq Lit_P \backslash \{l\}$，即$l$ 不出现在P'
- $\mathcal{AS}(P') = \mathcal{AS}_l(P)$

我们用$Forget(P, l)$ 表示在程序P 中遗忘l 。

在答集语义下的逻辑程序中，遗忘的结果也是唯一的。假设逻辑程序P'和P'' 都是在程序P 中遗忘文字l 的结果，那么他们等价。

下面的结果更一般，遗忘在答集语义下是语法无关的。

命题 2.18: ([39]) 假设逻辑程序P 和P' 是两个协调逻辑程序，l 是一个文字，那么$Forget(P, l)$ 和$Forget(P, l)$ 是等价的。

一个逻辑程序遗忘之后还是一个逻辑程序，这两个逻辑程序的答集是怎样的，下面的结论揭示了他们之间的关系。

命题 2.19: 对于每一个逻辑程序P和文字l，有

- $\mathcal{AS}(Forget(P, l)) = \{X \backslash \{l\} \mid X \in \mathcal{AS}_l(P)\}$
- 如果$X \in \mathcal{AS}_l(P)$ 且$l \notin X$，那么$X \in \mathcal{AS}(forget(P, l))$
- 任意$X \in \mathcal{AS}(P)$ 满足$l \in X$，那么$X \backslash \{l\} \in \mathcal{AS}(Forget(P, l))$
- 任意$X' \in \mathcal{AS}(Forget(P, l))$，那么$X'$ 或者$X' \cup \{l\}$ 属于$\mathcal{AS}(P)$。
- 对于每一个$X \in \mathcal{AS}(P)$，都存在$X' \in \mathcal{AS}(Forget(P, l))$ 满足$X' \subseteq X$。
- 如果l 不出现在程序P 中，那么$Forget(P, l) \equiv P$。

逻辑程序下一个性质是说答集的数目不会随着遗忘而增加。

假设P 是一个协调逻辑程序，l 是一个文字，那么$|\mathcal{AS}(Forget(P, l))| \leq |\mathcal{AS}(P)|$。

同样我们可以给出在逻辑程序中变量集合的遗忘，因为结果与遗忘顺序无关。其结果是通过逐个遗忘掉每一个文字得到的。

假设P是以协调逻辑程序，$V = \{l_1, \cdots, l_n\}$是文字集合，那么$Forget(P, V) \equiv Forget(Forget(Forget(P, l_1), l_2), \cdots, l_n)$。

来看下面的例子。

例 2.13: 假设P是下面的逻辑程序：

$$flies(Tweety) \leftarrow pigeon(Tweety)$$

$$\neg flies(Tweety) \leftarrow penguin(Tweety)$$

$$pigeon(Tweety) \vee penguin(Tweety)$$

这个程序有两个答集：$\{pigeon(Tweety), flies(Tweety)\}$和$penguin(Tweety), \neg flies(Tweety)$。

表示$Forget(P, flies(tweety))$一个程序是：

$$\neg flies(Tweety) \leftarrow penguin(Tweety)$$

$$pigeon(Tweety) \vee penguin(Tweety) \leftarrow .$$

表示$Forget(Forget(P, flies(Tweety)), \neg flies(Tweety))$的逻辑程序是：

$$pigeon(Tweety) \vee penguin(Tweety) \leftarrow .$$

最后给出答集逻辑程序遗忘和经典遗忘的关系，方法是找到与逻辑程序相关的经典公式集合。对于每个逻辑程序，它的完备集表示为$Comp(P)$，有如下公式组成：

- $body(r) \rightarrow head(r)$对于$P$中每一个规则$r$，
- 公式$a \rightarrow \bigvee_{r \in P, a \in head(r)} (body(r) \wedge \bigwedge_{p \in head(r) \{a\}} \neg p)$对于每一个原子$a$。

对于一个逻辑程序P，它的正依赖图G_P是一个有向图，使得节点是P中出现的原子，从p到q有边当且仅当存在一个规则r满足$p \in head(r)$且$q \in$

$body^+(r)$。非空变量集$L = \{p_1, \cdots, p_k\}$是P的环（loop）当且仅当对于L中任意不同的原子p_i, p_j，存在从p_i到p_j的非零路径使得路径上所有点都属于L。L的环公式如下：

$$CLF(L) = (p_1 \wedge \cdots \wedge p_k) \rightarrow (\bigvee_{\phi \in R(L)} \phi)$$

这里$R(L)$ 是公式$body(r) \wedge \bigwedge_{p \in head(r) L} \neg p$ 的集合，并且此规则r满足$r \in P$，有$head(r) \cap L = \emptyset$ 和$body^+(r) \cap L = \emptyset$。

我们令$lcomp(P) = comp(P) \cup CLF(P)$（$CLF(P)$是所有环公式），一个基本结果是$lcomp(P)$表征了答集语义。

命题 2.20: （[65]）假设P是析取逻辑程序，X是P中出现部分变量的集合，那么X是P的答集当且仅当X是$lcomp(P)$的一个模型。

下一个命题说明了逻辑程序遗忘和变量遗忘的关系。

命题 2.21: （[39]）假设P是一协调的析取逻辑程序，p是其中出现的一个原子，那么变量集X是$Forget(P, p)$ 的答集当且仅当X是$ForgetVar(lcomp(P), p)$的最小模型，也就是说

$$\mathcal{AS}(Forget(P, p)) = MMod(ForgetVar(lcomp(P), p))$$

这里$MMod(T)$代表理论T在经典逻辑下关于包含关系最小的模型。

第三章　公式遗忘的公设和语义

3.1 信念修正与知识更新

知识是动态变化的，最简单的形式是扩张（expansion），即当一个新的命题与一个给定理论协调时被添加到这个理论中，那么我们只需要像集合一样增加元素，这种扩张在逻辑推理下是封闭的。第二种理论改变称为收缩，即要添加的新命题与给定理论中某个命题x冲突，那么x必须被消弱或者排除，核心的问题决定哪些命题应该被拒绝以使收缩后的理论协调并且逻辑封闭的。第三种改变称为信念修正，新命题与原理论冲突，怎么样调整使得它包含新命题协调。后两种情形不像第一种那么简单，基本的形式化方法是给出描述这两种改变的特性。A.G.M.已经给出了各自的公设。

先给出信念修正的公设，用*表示修正算子。

(R1) $\psi * \phi \vdash \phi$。

(R2) 如果$\psi \wedge \phi$ 是可满足的，那么$\psi * \phi \leftrightarrow \psi \wedge \phi$。

(R3)如果ϕ是可满足的，那么$\psi * \phi$ 也可满足。

(R4) 如果$\psi_1 \leftrightarrow \psi_2$和$\phi_1 \leftrightarrow \phi_2$成立，那么$\psi_1 * \phi_1 \leftrightarrow \psi_2 * \phi_2$。

(R5) $(\psi * \phi) \wedge \mu \vdash \psi * (\phi \wedge \mu)$。

(R6)如果$(\psi * \phi) \wedge \mu$可满足那么$\psi * (\phi \wedge \mu) \vdash (\psi * \phi) \wedge \mu$。

(R1)说明修正之后的理论或知识库应该包含新命题，这符合我们的直观，新知识应该比旧知识更具可靠性，当发生冲突时，首先保留新知识，然后删除旧知识以达到整个知识库的协调性。(R2)指出当新命题与给定理论不矛盾时，信念修正就变成了扩张，直接将新命题添加到理论中，即取它们的合取作为结果。(R3)因为新知识肯定是要接受的，所以它如果可满足，那我们的修正结果也是。(R4)表明修正算子是语法无关的，对于语义等价的理论或命题，修正结果是相同的。(R5)和(R6)一起说明了如果修正结果与某命题合取是协调的，那么它与把此命题加入修正过程得到的结果是逻辑等价的。

下面是信念收缩的公设，用–表示收缩。

(C1) $\psi \vdash \psi - \phi$。

(C2) 如果$\psi \nvdash \phi$，那么$\psi - \phi \leftrightarrow \psi$。

(C3) 如果$\nvdash \phi$，那么$\psi - \phi \nvdash \phi$。

(C4) 如果$\psi_1 \leftrightarrow \psi_2$ 并且$\phi_1 \leftrightarrow \phi_2$，那么$\psi_1 - \phi_1 \leftrightarrow \psi_2 - \phi_2$。

(C5) $(\psi - \phi) \wedge \phi \vdash \psi$。

(C1)表示信念收缩由于在原先理论中删除某些命题，所以肯定比原理论要弱。(C2)是说我们在原理论中去掉一个本不成立的命题，那么收缩结果不会发生变化。(C3)如果被收缩的命题不是重言式，那么收缩就是正常的，并且不会推出此命题。(C4)表明收缩算子也应该满足语法无关性，与公式的表示形式无关，语义等价的公式收缩结果页等价。(C5)是说收缩具有恢复性，当收缩结果再添加被删除的命题时能够蕴含原理论。

另一种知识管理的形式是知识更新，它满足如下公设：

(U1) $\psi \diamondsuit \phi \vdash \phi$。

(U2) 如果$\psi \nvdash \phi$，那么$(\psi \diamondsuit \phi) \leftrightarrow \psi$。

(U3) 如果ψ 并且ϕ 是可满足的，那么$\psi \diamondsuit \phi$也是。

(U4) 如果$\psi_1 \leftrightarrow \psi_2$ 和$\phi_1 \leftrightarrow \phi_2$ 那么$(\psi_1 \diamondsuit \phi_1) \leftrightarrow (\psi_2 \diamondsuit \phi_2)$

(U5) $(\psi \diamondsuit \mu) \wedge \phi \vdash \psi \diamondsuit (\mu \wedge \phi)$。

(U6)如果$\psi \diamondsuit \phi_1 \vdash \phi_2$且$\psi \diamondsuit \phi_2 \vdash \phi_1$，那么$(\psi \diamondsuit \phi_1) \leftrightarrow (\psi \diamondsuit \phi_2)$。

(U7) 如果ψ 是完全的，则$(\psi \diamondsuit \phi_1) \wedge (\psi \wedge \phi_2) \vdash (\psi \diamondsuit (\phi_1 \vee \phi_2)$。

(U8) $(\psi_1 \vee \psi_2) \diamondsuit \phi \leftrightarrow (\psi_1 \diamondsuit \phi) \vee (\psi_2 \diamondsuit \phi)$。

正像信念修正和收缩是相对的概念，知识更新和知识擦除也是一对概念。我们用$\varphi \blacklozenge \mu$表示在知识库$\varphi$ 中擦除μ，满足如下公设：

(E1)φ 蕴含$\varphi \blacklozenge \mu$

(E2) 如果$\varphi \models \neg \mu$，那么$\varphi \blacklozenge \mu \equiv \varphi$

(E3)如果φ是可满足的且μ不是重言式，那么$\varphi \blacklozenge \mu \not\models \mu$

(E4)如果$\varphi_1 \equiv \varphi_2$ 且$\mu_1 \equiv \mu_2$，那么$\varphi_1 \blacklozenge \mu_1 \equiv \varphi_2 \blacklozenge \mu_2$

(E5)$(\varphi \blacklozenge \mu) \wedge \mu$蕴含$\varphi$

(E6) $(\varphi_1 \vee \varphi_2) \blacklozenge \mu \equiv (\varphi_1 \blacklozenge \mu) \vee (\varphi_2 \blacklozenge \mu)$

归并是要从几个相互独立通常又是不协调的多个信息源中集成一个协调的知识库，这样归并一般不是平凡的（即单纯把他们放到一起）。$\triangle$ 是归并算子，μ是约束条件，下面我们给出带约束条件的归并公设。

(IC0) $\triangle_{\mu}(\Phi) \vdash \mu$。

(IC1) 如果μ 是协调的，那么$\triangle_{\mu}(\Phi)$ 也是协调的。

(IC2) 如果$\wedge\Phi$ 与μ 是协调的，那么$\triangle_{\mu}(\Phi) = \wedge\Phi \wedge \mu$。

(IC3) 如果$\Phi_1 \equiv \Phi_2$，$\mu_1 \equiv \mu_2$，那么$\triangle_{\mu_1}(\Phi_1) \equiv \triangle_{\mu_2}(\Phi_2)$。

(IC4) 如果$\varphi \models \mu$ 并且$\varphi' \models \mu$，那么$\triangle_{\mu}(\varphi \sqcup \varphi') \wedge \varphi \not\models \bot$ 推出$\triangle_{\mu}(\varphi \sqcup \varphi') \wedge \varphi' \not\models \bot$。

(IC5) $\triangle_{\mu}(\Phi_1) \wedge \triangle_{\mu}(\Phi_2) \vdash \triangle_{\mu}(\Phi_1 \sqcup \Phi_2)$。

(IC6) 如果$\triangle_{\mu}(\Phi_1) \wedge \triangle_{\mu}(\Phi_2)$ 协调，那么$\triangle_{\mu}(\Phi_1 \sqcup \Phi_2) \vdash \triangle_{\mu}(\Phi_1) \wedge \triangle_{\mu}(\Phi_2)$。

(IC7) $\triangle_{\mu_1}(\Phi) \wedge \mu_2 \vdash \triangle_{\mu_1 \wedge \mu_2}(\Phi)$。

(IC8) 如果$\triangle_{\mu_1}(\Phi) \wedge \mu_2$ 是协调的，则$\triangle_{\mu_1 \wedge \mu_2}(\Phi) \vdash \triangle_{\mu_1}(\Phi) \wedge \mu_2$。

直观上来说，$\triangle_{\mu}(\Phi)$是满足约束μ距离Φ最近的知识库。(IC0)确保归并结果满足完整性约束条件。(IC1)表明如果约束条件是协调的，那么归并结果也是协调的。(IC2)表明如果可能的话，归并结果所有知识库与约束条件的合取，只要此合取式是协调的。(IC3)是语法独立性，即如果两个知识集等价且两个约束条件等价，那么两个归并结果逻辑等价。(IC4)是公平假设，两个知识库归并，结果不会偏向任何一方，同等对待这两个知识库。(IC5)和(IC6)两项说明，如果两个子群组至少一个共同选择，那么整个群组的归并结果就是这些共同选择。(IC7)和(IC8)是相应信念修正的一般化，他们说明在后者是协调情况下，两个约束的归并和一个归并再合取另一个是等价的。

知识管理包括信念改变（信念修正、信念收缩）、知识更新、知识归并（后面会提到）和遗忘，前面三种操作的是知识或者信念，目前还没有一个标准算子，针对前面三种提出了公设用以规范他们所应满足的规则，因此我们首先考虑给出遗忘的公设，暂时用$Forget(\varphi, \psi)$表示在公式φ中遗忘ψ。

- (F1) $\varphi \models Forget(\varphi, \psi)$。

- (F2)如果$S = Atoms(\varphi) - Atoms(\psi) \neq \emptyset$，那么$\varphi$ 和$Forget(\varphi, \psi)$在S上推理结果一样。

- (F3) 如果$\not\models \psi$ 那么$Forget(\varphi, \psi) \not\models \psi$。

- (F4) 如果$\varphi_1 \leftrightarrow \varphi_2$ and $\psi_1 \leftrightarrow \psi_2$ 那么$Forget(\varphi_1, \psi_1) = forget(\varphi_2, \psi_2)$。

- (F5) $Forget(Forget(\varphi, \psi), \psi) \equiv Forget(\varphi, \psi)$。

(F1)表明遗忘是知识减少的过程，公式变弱，即原知识肯定蕴含遗忘之后的结果，无论变量遗忘还是文字遗忘都满足这条性质，这是遗忘的本质。(F2)说的是遗忘对遗忘之外的变元没有影响。因为$a \wedge b \models b$并且$a \vee c$不含原子b，那么$Forget(a \wedge b, a \vee c) \models b$。由此我们还能推出，如果遗忘公式$\varphi$与$\psi$没有公共原子变量，那么遗忘结果还是$\varphi$。比如，$Forget(a \vee b, c \wedge d) \equiv a \vee b$。(F3)表明遗忘掉一个非重言式的公式，结果不会再蕴含此公式，遗忘了就不知道了。(F4)表达了等价公式遗忘结果相同的，即遗忘结果与语法形式无关，这条性质在知识管理的各类方法都要求是成立的。(F5)遗忘是稳定的过程，多次遗忘同一个公式与遗忘一次结果是等价的。

最简单的遗忘定义是转化为变量遗忘，思想是遗忘一个公式等价于遗忘公式中出现的每个变元，这就把遗忘一个公式转化成了遗忘一个变量集。

定义 3.1: 对于任意公式φ, ψ，我们的第一个遗忘算子表示为$Forget_0(\varphi, \psi)$，如下定义

$$Forget_V(\varphi, \psi) \equiv ForgetVar(\varphi, Atoms(\psi))$$

这种定义直观、简单，根据变量遗忘的性质，一个最直接的推论是，如果$Atoms(\varphi) \cap Atoms(\psi) = \emptyset$，那么$Forget_V(\varphi, \psi) \equiv \varphi$。而且它满足我们关于公式遗忘其中四个公设。

命题 3.1: 算子$Forget_V(\varphi, \psi)$满足(F1)-(F3)，(F5)。

证明: 由变量遗忘的性质得，遗忘一个变量集肯定比原先公式要弱，所以$\varphi \models ForgetVar(\varphi, Atoms(\psi))$，(F1)成立。

由变量遗忘性质，遗忘对遗忘变量以外的变量没有影响。遗忘公式ψ只是相当于遗忘ψ中出现的每个变量，对其他变量没有影响。如果$S = Atoms(\varphi) - Atoms(\psi) \neq \emptyset$，那么对于任意建立在$S$上的公式$\varphi'$，必定有$\varphi \models \varphi'$当且仅当$Forget(\varphi, \psi) \models \varphi'$，所以(F2)成立。

根据定义，$Forget_V(\varphi, \psi)$ 不包含ψ 中出现的任何变元，所以显然$Forget_V(\varphi, \psi) \not\models \psi$，(F3)成立。同样的原因，继续遗忘$\psi$，结果不会发生变化。

我们看到$Forget_V(\varphi,\psi)$并不满足(F4)。例如，$\varphi \equiv a \wedge b \wedge c$，$\psi_1 \equiv a \vee b$和$\psi_2 \equiv (a \vee b) \wedge (c \vee \neg c)$，显然有$\psi_1 \equiv \psi_2$。所以$Forget_V(\varphi,\psi_1) \equiv c$，但是$Forget_V(\varphi,\psi_2) \equiv \top$。所以这种遗忘定义与公式的语法形式相关。通过这个例子，我们看到在公式ψ_2中，c是可以去掉的。只要我们适当限定公式的形式，上面关于公式遗忘的概念就可以满足语法独立性。下面最简公式的概念就是为这目的而提出来的，最简公式中的符号都是相关的，如果某个公式去掉某个变量符号之后还等价于原先的公式，那么就不是最简的。既然要考虑公式中出现的符号是不是都相关这个问题，我们完全可以用遗忘来定义。

定义 3.2: 对任意公式φ，如果不存在变量集$V \subseteq Var(\varphi)$，使得$\varphi \equiv ForgetVar(\varphi,V)$，那么$\varphi$就是变量最简公式。

例 3.1: 公式$\varphi_1 = (a \wedge b) \vee (\neg b \wedge c)$是变量最简公式；

公式$\varphi_2 = (a \wedge b) \vee (\neg b \wedge c) \vee (b \wedge b)$也是变量最简公式，尽管最后面部分$(b \wedge b)$是可以去掉的，我们变量最简公式的概念只是考虑有没有多余的变量，而不管公式的长度；

公式$\varphi_3 = (a \wedge b) \wedge (a \vee c)$不是变量最简的，因为$ForgetVar(\varphi,c) \equiv a \wedge b$，而$\varphi_3 \equiv a \wedge b$，所以$ForgetVar(\varphi,c) \equiv \varphi_3$。

不过按照变量最简公式的定义，判断某个公式是不是最简的，我们需要考察集合$Var(\varphi)$的所有子集，总共有$2^{|Var(\varphi)|}$。我们需要计算指数级的变量遗忘才能判断公式是不是最简的，计算量是很大的。能不能改进这个算法呢？通过仔细观察，如果我们考察了对于$\{a,b\}$，有$ForgetVar(\varphi,\{a,b\}) \not\models \varphi$，那么对于任何包含$\{a,b\}$最为子集的集合$V$，同样有$ForgetVar(\varphi,V) \not\models \varphi$。这是因为$ForgetVar(\varphi,\{a,b\}) \models ForgetVar(\varphi,V)$。所以我们的改进算法是，依次考虑遗忘每一个原子$p \in Var(\varphi)$，判断$ForgetVar(\varphi,p)$是不是等价于$\varphi$。如果发现存在一个$p \in Var(\varphi)$，使得$ForgetVar(\varphi,p) \equiv \varphi$，那么算法终止，$\varphi$不是最简公式。否则，我们不需要再考虑遗忘其他子集，因为他们肯定包含某个单原子集合作为其子集。这样我们只需要计算$|Var(\varphi)|$步遗忘就可以判断是否为最简公式。

命题 3.2: 对任意公式φ，如果不存在变量$p \in Var(\varphi)$，使得$\varphi \equiv ForgetVar(\varphi,p)$，那么$\varphi$就是变量最简公式。

每一个公式都可以求出其变量最简公式，我们可以依次遗忘掉每一个变量，若与原公式等价，则接着遗忘其他变元直到不可能等价于原公式为止，最终得到最简公式。那么我们可以修改先前的遗忘使其满足语法独立性，得到如下定义。

定义 3.3: 对于任意公式φ, ψ，ψ的变量最简公式是ψ'，遗忘算子$Forget'_V(\varphi, \psi)$定义为

$$Forget'_V(\varphi, \psi) \equiv ForgetVar(\varphi, Atoms(\psi'))$$

我们可以看到，在上述遗忘算子定义中，φ 可以是任意形式，不一定是最简的，但是ψ 必须是最简公式，因为我们的算子是遗忘掉其中出现的变量，等价公式可能会有不同的变量出现，这影响了遗忘的结果。修改后的遗忘满足所有的公设。

命题 3.3: 对于任意公式φ和公式ψ，有$Forget'_V(\varphi, \psi)$满足(F1)-(F5)。

证明: 我们只证明(F4)成立，其他证明类似于$Forget_V(\varphi, \psi)$。如果$\psi_1 \equiv \psi_2$且ψ'_1, ψ'_2分别是ψ_1, ψ_2的变量最简公式，那么显然$\psi'_1 \equiv \psi'_2$，所以$Atoms(\psi'_1) = Atoms(\psi'_2)$。这样得到，如果$\varphi_1 \equiv \varphi_2$，那么$Forget'_V(\varphi_1, \psi_1) \equiv Forget'_V(\varphi_2, \psi_2)$。

另外一种遗忘不是遗忘出现的变量，而是遗忘出现的文字，比如要遗忘$\psi \equiv \neg a \vee b$，转化为遗忘文字集$\{\neg a, b\}$。因为文字遗忘比变量遗忘更精细，所以从这个角度定义也许更好。

定义 3.4: 对于任意公式φ, ψ，我们的第二个遗忘算子表示为$Forget_L(\varphi, \psi)$，如下定义

$$Forget_L(\varphi, \psi) \equiv ForgetLit(\varphi, Lit(\psi))$$

命题 3.4: 算子$Forget_L(\varphi, \psi)$满足(F1)-(F3)，(F5)。

证明: 证明与变量类似，略。

它也不满足公设(F4)，来看例子，$\varphi = a \wedge \neg b \wedge c$，$\psi_1 = a \vee b$，$\psi_2 = (a \vee b) \wedge (c \vee \neg c)$，那么$\psi_1 \equiv \psi_2$，但是显然$Forget_L(\varphi, \psi_1) \equiv ForgetLit(\varphi, \{a, b\}) \equiv \neg b \wedge c$，而$Forget_L(\varphi, \psi_2) = ForgetLit(\varphi, \{a, b, c\}) = \neg b$，因此他们不等价。

为了适当的限定公式的形式达到公式独立性，我们类似定义文字最简公式的概念。

定义 3.5: 对任意公式φ，如果不存在文字集$L \subseteq Lit(\varphi)$，使得$\varphi \equiv ForgetLit(\varphi, L)$，那么$\varphi$就是文字最简公式。

例 3.2: 公式$\varphi_1 = (a \wedge b) \vee (\neg b \wedge c)$是文字最简公式，公式$\varphi_2 = (a \wedge b) \vee (\neg b \wedge c) \vee (b \wedge b)$是文字最简公式，而公式$\varphi_3 = (a \wedge b) \wedge (a \vee \neg b)$不是文字最简的，因为$ForgetLit(\varphi_3, \neg b) \equiv a \wedge b$，而$\varphi_3 \equiv a \wedge b$，所以$ForgetLit(\varphi_3, c) \equiv \varphi_3$，这里文字$\neg b$是多余的。

对上述文字最简公式定义也可作算法改进以提高计算效率，对于$Lit(\varphi)$中的每一个文字l，依次检查$ForgetLit(\varphi, l)$与φ是否等价即可。若存在$l \in Lit(\varphi)$使得$ForgetLit(\varphi, l) \equiv \varphi$，那么算法终止，公式$\varphi$不是文字最简的。否则，算法也停止，是最简的。

类似地，我们也可以得到任意公式的文字最简公式，下面算子是从文字最简公式角度定义的。

定义 3.6: 对于任意公式φ, ψ，ψ的文字最简公式是ψ'，遗忘算子$Forget'_L(\varphi, \psi)$定义为

$$Forget'_L(\varphi, \psi) \equiv ForgetLit(\varphi, Lit(\psi'))$$

同样地，φ 可以是任意形式，不一定是最简的，但是ψ 必须是文字最简的，因为我们的算子是遗忘掉其中出现的文字，等价公式可能会有不同的文字出现。对基于文字的遗忘改进也满足语法独立性，即

命题 3.5: 算子$Forget'_L$满足(F1)-(F5)。

上述两种形式的定义满足下面几个命题，第一个是说他们都是相应变量或文字遗忘的自然推广。

命题 3.6: 任意命题公式φ 和原子p，文字l，有
$Forget_V(\varphi, p) \equiv ForgetVar(\varphi, p)$；$Forget_L(\varphi, l) \equiv ForgetLit(\varphi, l)$。

证明:

$$Forget_V(\varphi, p) \equiv ForgetVar(\varphi, Var(p)) = ForgetVar(\varphi, p)$$

$$Forget_L(\varphi, l) \equiv ForgetLit(\varphi, Lit(l)) = ForgetVar(\varphi, l)$$

下一个称为对称性。

命题 3.7: 对称性 任意公式φ, ψ，我们都有
$Forget_V(\varphi, \psi) = Forget_V(\varphi, \neg\psi)$。

证明: 因为ψ 与$\neg\psi$包含相同的变量，假设此变量集合为V，那么$Forget(\varphi, \psi) = ForgetVar(\varphi, V)$并且$Forget(\varphi, \neg\psi) = ForgetVar(\varphi, V)$，所以两者相等。

根据文字定义的遗忘不满足上述对称性，因为我们不同对待文字l和他的否文字$\neg l$，而变量遗忘同等对待。比如，$Forget_L(a \wedge \neg b \wedge c, \neg b \vee c) \equiv a$，而$Forget_L(a \wedge \neg b \wedge c, b \wedge \neg c) \equiv a \wedge \neg b \wedge c$，显然不相等。

命题 3.8: 前置单调性
如果$\varphi_1 \models \varphi_2$，那么
$Forget_V(\varphi_1, \psi) \models Forget_V(\varphi_2, \psi)$；$Forget_L(\varphi_1, \psi) \models Forget_L(\varphi_2, \psi)$。

证明: 设$V = Atoms(\psi)$。因为$\varphi_1 \models \varphi_2$，所以$Mod(\varphi_1) \subseteq Mod(\varphi_2)$。根据变量遗忘的性质，$Mod(ForgetVar(\varphi, V)) = Mod(\varphi) \cup \{\omega | \exists \omega' \in Mod(\varphi), \omega, \omega'$在$V$的某子集上上指定不同的真值$\}$。所以$Mod(ForgetVar(\varphi_1, V)) \subseteq Mod(ForgetVar(\varphi_2, V))$。因此$Forget_V(\phi_1, \psi) \models Forget_V(\phi_2, \psi)$。

同样假设$L = Lit(\psi)$，根据文字遗忘的性质，$Mod(ForgetLit(\varphi, L)) = Mod(\varphi) \cup \{\omega | \exists \omega' \in Mod(\varphi), \omega$使$L$上某子集上的文字都为假，$\omega'$使其为真$\}$。所以$Mod(ForgetLit(\varphi_1, L)) \subseteq Mod(ForgetLit(\varphi_2, L))$。因此$Forget_L(\phi_1, \psi) \models Forget_L(\phi_2, \psi)$。

下面的单调性是对于遗忘算子第二个变元的单调性，所以叫做后置单调性。遗忘变量越多，公式越弱，所以下面的性质是显然的。

命题 3.9: 后置单调性
如果$Var(\psi_1) \subseteq Var(\psi_2)$，那么$Forget_V(\varphi, \psi_1) \models Forget_V(\varphi, \psi_2)$；
如果$Lit(\psi_1) \subseteq Lit(\psi_2)$，那么$Forget_L(\varphi, \psi_1) \models Forget_L(\varphi, \psi_2)$。

证明: 根据$Forget_V, Forget_L$的定义和相应变量集和文字集性质，很容易证明，略。

下一个性质是平凡性，逻辑常量遗忘任何公式还是本身，任意公式遗忘自身是永真式，下面性质描述了这些方面。

命题 3.10: 平凡性
$Forget_V(\phi,\phi)=\top, Forget_L(\phi,\phi)=\top$
$Forget_V(\top,\psi)=\top, Forget_L(\top,\psi)=\top$
$Forget_V(\bot,\psi)=\bot, Forget_L(\bot,\psi)=\bot$
$Forget_V(\phi,\bot)=\phi, Forget_L(\phi,\bot)=\phi$

证明: 由定义很容易看出，证明略。

由于变量遗忘满足分配律，上面这种简单定义也满足此性质。

命题 3.11: 对于任意公式φ_1,φ_2和ψ，则
$Forget_V(\varphi_1\vee\varphi_2,\psi)\equiv Forget_V(\varphi_1,\psi)\vee Forget_V(\varphi_2,\psi)$;
$Forget_L(\varphi_1\vee\varphi_2,\psi)\equiv Forget_L(\varphi_1,\psi)\vee Forget_L(\varphi_2,\psi)$。

证明: 因为$ForgetVar(\varphi_1\vee\varphi_2,p)=ForgetVar(\varphi_1,p)\vee ForgetVar(\varphi_2,p)$，所以对于任意变量集合$V$，很容易归纳出$ForgetVar(\varphi_1\vee\varphi_2,V)=ForgetVar(\varphi_1,V)\vee ForgetVar(\varphi_2,V)$，从而$Forget_V(\varphi_1\vee\varphi_2,\psi)\equiv Forget_V(\varphi_1,\psi)\vee Forget_V(\varphi_2,\psi)$成立。对于基于文字遗忘的定义也满足，证明类似。

上面的性质对于改进之后的两个定义也满足，不再赘述。尽管这些定义简单直观，有不错的性质，但是它们最大的缺点是不能区分遗忘公式的形式。很容易看出，$Forget_V(\varphi,a\vee b)\equiv Forget_V(\varphi,a\wedge b)$和$Forget_L(\varphi,a\vee b)\equiv Forget_L(\varphi,a\wedge b)$。不管是遗忘$a\vee b$还是$a\wedge b$，我们都是转化为遗忘$a,b$两个变量或者两个文字，所以结果肯定是一样的。还有，$Forget_V(\varphi,\neg a\vee\neg b)\equiv Forget_V(\varphi,a\vee b)\equiv Forget_V(\varphi,a\rightarrow b)$也是成立的。所以此定义对于很多形式不同但是包含相同变量的遗忘公式，结果都是一样的。

3.2 两种语义定义

从模型的角度看公式遗忘，操作非常简单。首先看在变量遗忘和文字遗忘中，他们的模型是怎样变化的，即我们怎样由最初公式的模型变换出新的模型，最终得到结果公式的模型。在图3.1中，我们考虑3个变量构成的命题语言，图中总共有8个模型，每个都用三位的$0,1$序列表示。我们假设构成这个命题语言的变量是a,b,c，按照a,b,c这个顺序表示，每个3位序列

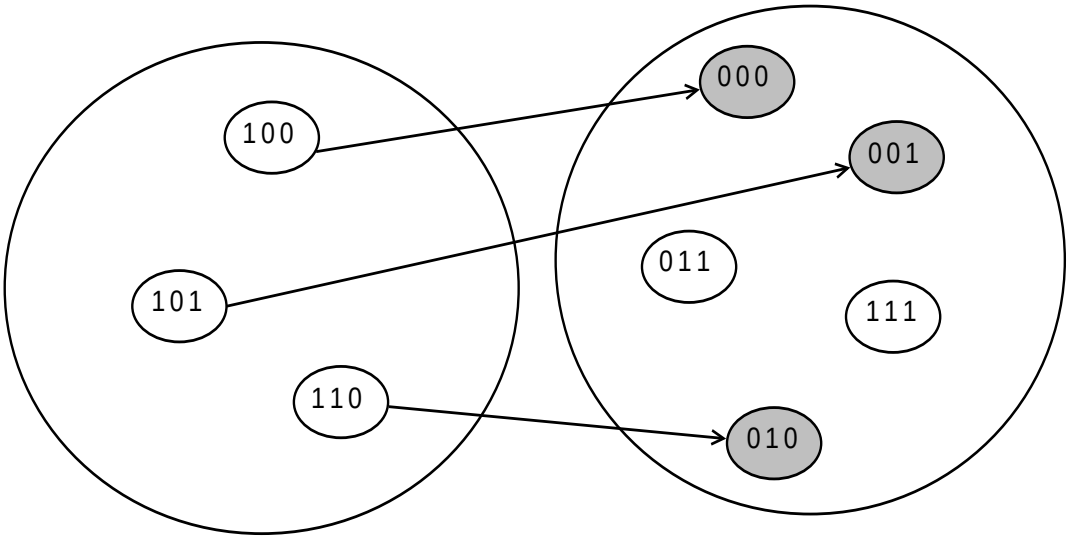

图 3.1　变量遗忘模型变换图

代表一个模型，1表示对应变脸为真，0代表对应变元为假。比如001表示模型$\neg a \wedge \neg b \wedge c$，101表示模型$a \wedge \neg b \wedge c$。左边代表我们要遗忘的公式$\varphi$，现在考虑$ForgetVar(\varphi, a)$。因为我们是遗忘原子$a$，所以结果的模型关于变量$a$是对称的。对于$\varphi$的每个模型，改变$a$变量的真值（原先为真，变为假；原先为假，变为真），其余变量真值不变（因为他们并没有被遗忘，应保持不变）。比如，100变成000，只是对应a的第一位由1变为0，第二三位保持不变，对φ其他两个模型做同样变换。即把现有的3个模型分别按图中箭头指示变化到一个新的模型，那么我们最后把阴影部分模型和φ的模型作为我们的结果公式的模型，即$Mod(ForgetVar(\varphi, a)) = \{100, 101, 110, 001, 000, 010\} =\equiv \neg b \vee \neg c$。对于文字遗忘，我们考虑$ForgetLit(\varphi, \neg b)$。文字遗忘不是要一定要改变对应变量的真值，它只做单向变换，也就是只把每个模型变到使遗忘文字为假，如果原先模型就是遗忘文字为假，那么变换之后还是本身。例如，模型100变成使文字$\neg b$为假，就把第二位变成1得到110，101变成111，110本身使$\neg b$为假，所以还是变成自身。所以我们的结果是$Mod(ForgetLit(\varphi, \neg b)) = \{100, 101, 110, 111\} \equiv a$，参考图3.2。值得注意的是图中有两个模型变换的新模型还在左边圆圈内，有一个模型变换后是其本身。

把公式遗忘看做文字遗忘的推广更自然，因为文字本身就是最简单的公式，首先来看文字遗忘的一个模型定义。假设φ是命题公式，l是一文字（正文字或负文字），那么$Mod(ForgetLit(\varphi, l)) = Mod(\varphi) \vee \{Force(\omega, \neg l) | \omega \models \varphi\}$。

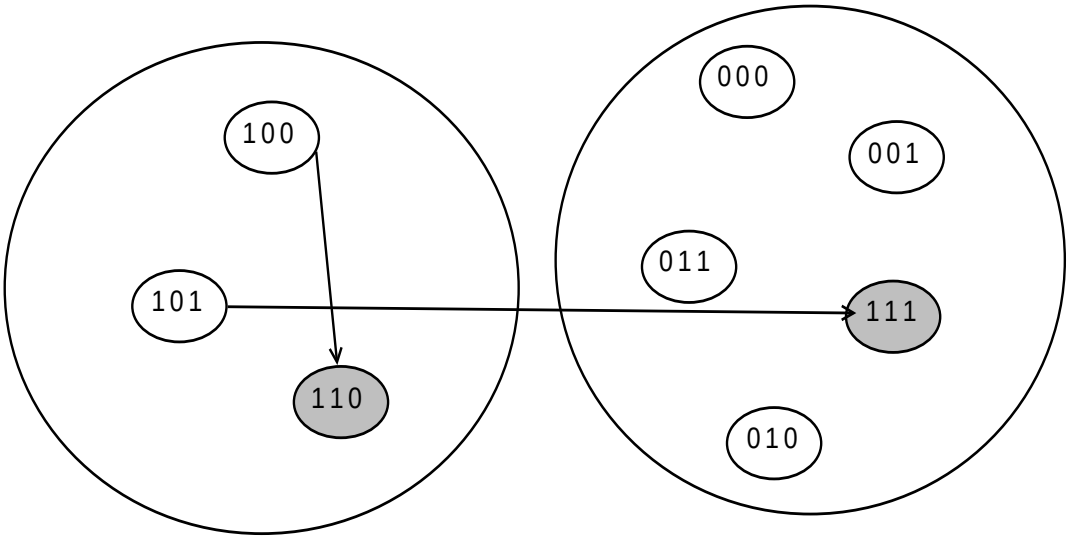

图 3.2　文字遗忘模型变换图

这里$Force(\omega, \neg l)$ 代表一个新的模型ω'使得$\omega' \models \neg l$且ω, ω' 仅可能在l 上赋值不同。$Force(\omega, \neg l)$表示与ω最近邻的模型并且使得$\neg l$ 为真。当然ω, ω'可能相同。比如，$\omega = a \wedge b \wedge c$，如果$l = c$，那么$\omega' = Force(\omega, \neg l) = a \wedge b \wedge \neg c$；如果如果$l = \neg c$，那么$\omega' = Force(\omega, \neg l) = a \wedge b \wedge c = \omega$。

我们的第一种语义定义就是从这个方面考虑的，这里需要推广$Force(\omega, \neg l)$这个概念到公式层面上。首先需要给出两个模型的距离，对于两个模型ω, ω'，$Dis(\omega, \omega')$表示它们之间的距离，定义为这两个模型赋值不同的原子的个数。假设我们考虑命题原子集$\{a, b, c\}$上语言，令$\omega = 101$（这三个数字分别代表a, b, c的取值，1表示对应命题原子为真，0表示为假，所以实际上$\omega = a \wedge \neg b \wedge c$），$\omega' = 111$，因为$\omega, \omega'$只是对原子$b$的赋值不同，所以$Dis(\omega, \omega') = 1$。

在图3.3中，我们考虑的是建立在三个命题原子上的语言，总共有8个模型，从000到111，不妨设这3个原子分别为a, b, c。每一条边连接的两个模型距离为1，也就是它们只有一个赋值不同的原子。而且图中任意两个模型的距离就是它们之间的最短路径，可以看出距离最长的两个模型是$a \wedge b \wedge c$和$\neg a \wedge \neg b \wedge \neg c$，有$Dis(a \wedge b \wedge c, \neg a \wedge \neg b \wedge \neg c) = 3$。我们可得$Force(a \wedge b \wedge \neg c, \neg c) = a \wedge b \wedge \neg c$，$Force(a \wedge b \wedge \neg c, c) = a \wedge b \wedge c$。

定义 3.7: $Force(\omega, \psi)$代表模型的集合，使得任意$\omega' \in Force(\omega, \psi)$，有如下条件成立：

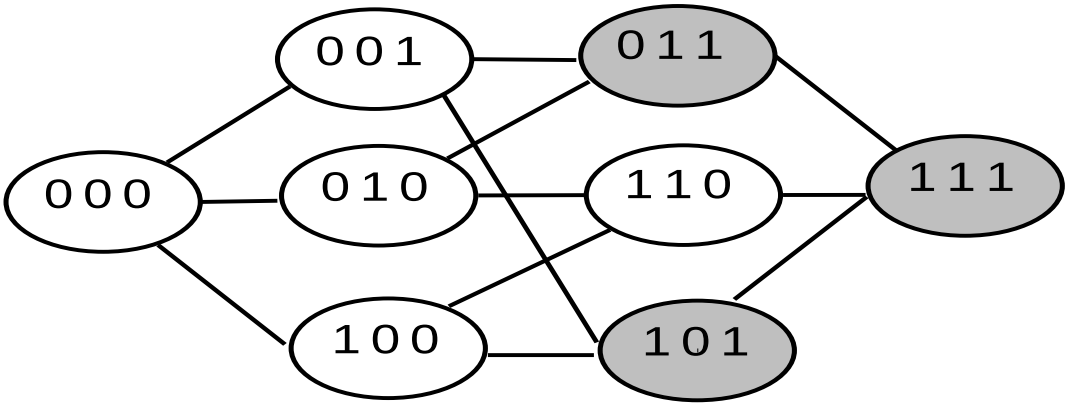

图 3.3　Force模型变换图

- $\omega' \models \psi$。
- ω'使得$Dis(\omega, \omega')$ 最小。

假设$\psi = (b \wedge c) \vee (a \wedge \neg b \wedge c)$，图3.3中阴影部分3个模型表示$\psi$的模型。$Force(100, \neg\psi) = \{100\}$ 因为模型100本身就使得$\neg\psi$为真，不需要变换。而$Force(011, \neg\psi) = \{001, 010\}$，因为011使得$\neg\psi$为假，在所有使得$\neg\psi$为真的5个模型中，只有$001, 010$两个距离011最近，距离为1。我们可以看到，对于公式ψ，$Force(\omega, \psi)$可能包含一个模型，可能有多个模型，这就是我们把其定义为模型集合的原因，而$Force(\omega, l)$只是一个模型，这是他们的不同。相同点是$Force(\omega, \psi)$也可能只有ω一个模型。

根据定义易得，如果$\omega \in Mod(\psi)$，那么$Force(\omega, \psi)$只包含一个模型，就是ω本身，即$Force(\omega, \psi) = \{\omega\}$。对于模型和公式的距离默认采用$Dis(\omega, \psi) = Min_{\omega' \in Mod(\psi)}(\omega, \omega')$，也就是等于与公式模型距离的最小值。因此有下面命题。

命题 3.12: 对于任意公式ψ和模型ω，如果$\omega' \in Mod(Force(\omega, \psi))$，那么$Dis(\omega, \omega') = Dis(\omega, \psi)$。

证明: 如果$\omega' \in Force(\omega, \psi)$，那么$\omega' \in Mod(\psi)$并且满足$Dis(\omega, \omega')$达到最小，所以$Dis(\omega, \omega') = Dis(\omega, \psi)$。

这种扩充之后，面临的第一个问题就是新的$Force$定义能否包含文字作为特殊情况，这就是下面的性质。为了区别定义，我们在下一个命题暂时用$Force_F$表示输入是公式的情况，以后还是用$Force$，因为通过第二个输入变元是公式还是文字，它的意义就很明确。

命题 3.13: 对于文字l和模型ω，有

$$Force_F(\omega, l) = \{Force(\omega, l)\}$$

证明: 假设$\omega' = Force(\omega, l)$，那么$\omega' \models l$并且$Dis(\omega, \omega') \leq 1$。如果$Dis(\omega, \omega') = 0$，则$\omega = \omega'$，显然成立；如果$Dis(\omega, \omega') = 1$，则$\omega \models \neg l$，所以任意$\omega_1 \in Force_F(\omega, l)$，都有$Dis(\omega_1, \omega) \geq 1$，因此$\omega' \in Force_F(\omega, l)$。对于其他使得$l$为真的模型，其与$\omega$的距离都大于1，因此$Force(\omega, l)$只包含$\omega'$一个模型。

命题 3.14: 假设$\omega' \in Force(\omega, \psi)$，那么$\omega, \omega'$仅在$Var(\psi)$上赋值不同。

证明: 对于整个命题原子集合Ω，一般来说，$Var(\psi) \subseteq \Omega$。不妨设$\Omega - Atoms(\psi) \neq \emptyset$，如果$\omega, \omega'$在某个命题原子$p \in \Omega - Var(\psi)$上赋值不同，那么令$\omega''$与$\omega'$仅在$p$上赋值不同，因为$\omega' \models \psi$，所以$\omega'' \models \psi$。那么$\omega''$与$\omega$在$p$上赋值相同，这样$Dis(\omega, \omega'') < Dis(\omega, \omega'')$，从而$\omega \notin Force(\omega, \psi)$，矛盾。

有了上面的概念，公式遗忘就很自然了，下面的遗忘定义形式上类似于文字遗忘。

定义 3.8: 假设φ, ψ是命题公式，定义一个新的公式遗忘算子表示为$Forget_F$，则有
$Mod(Forget_F(\varphi, \psi)) = Mod(\varphi) \vee \{\omega' \in Force(\omega, \neg\psi) | \omega \models \varphi\}$。

例 3.3: 假设$\varphi = Forget_F(a \wedge b \wedge c, b \wedge c)$，首先对于$a \wedge b \wedge c$的每一个模型$\omega$（碰巧只有一个），来求$Force(\omega, \neg b \vee \neg c)$，我们得到$Force(111, \neg b \vee \neg c) = \{110, 101\}$，因此有$Mod(varphi) = Mod(a \wedge b \wedge c) \cup \{110, 101\} = \{111, 110, 101\}$。所以$Forget_F(a \wedge b \wedge c, b \wedge c) = (a \wedge b) \vee (a \wedge \neg b \wedge c)$。

假设$\varphi' = Forget_F(a \vee b \vee c, b \vee c)$，那么$Mod(a \vee b \vee c) = \{001, 010, 100, 011, 101, 110, 111\}$和$Mod(\neg(b \vee c)) = \{000, 100\}$。从图3.1很容易得出，

$Force(001, \neg(b \vee c)) = \{000\}$，$Force(010, \neg(b \vee c)) = \{000\}$，
$Force(100, \neg(b \vee c)) = \{100\}$，$Force(011, \neg(b \vee c)) = \{000\}$，
$Force(101, \neg(b \vee c)) = \{100\}$，$Force(110, \neg(b \vee c)) = \{100\}$，
$Force(111, \neg(b \vee c)) = \{100\}$，

所以$Mod(\varphi') = \{000, 001, 010, 100, 011, 101, 110, 111\}$。
因此有$Forget_F(a \vee b \vee c, b \vee c) \equiv \top$。

命题 3.15: $Forget_F$满足(F1)-(F5)、前置单调性和析取分配律。

证明: 显然$Mod(\phi) \subseteq Mod(Forget_1(\phi, \psi))$，所以$\phi \models Forget_1(\phi, \psi)$，(F1)成立。从$Forget_1(\phi, \psi)$的模型来看，增加的新模型只可能改变$Var(\psi)$中变量的值，剩余变量赋值保持不变，所以对于任意建立在S上的公式ϕ'，都有$\phi \models \phi'$当且仅当$Forget_F(\phi, \psi) \models \phi'$，(F2)成立。假设$\psi$不是重言式，根据定义，$Forget_F(\phi, \psi)$的模型中必定包括使$\psi$为假的模型，所以$Forget_F(\phi, \psi) \not\models \psi$，(F3)成立。假设$\varphi_1 \equiv \varphi_2$，$\psi_1 \equiv \psi_2$，那么$Mod(\varphi_1) = Mod(\varphi_2)$且$Mod(\psi) = Mod(\psi)$。$Forget_F$就是操纵的模型，既然模型集合都一样，那么对于等价公式遗忘等价公式得到的结果也是等价的，(F4)成立。对于$Mod(Forget_F(\phi, \psi))$中任意的新模型$\omega'$，$\omega' \models \neg\psi$，所以在第二次遗忘的时候，这些模型并不会产生新模型；对于其他模型产生的新模型与第一次遗忘时相同，所以两次遗忘与一次遗忘结果相同。

如果$\varphi_1 \models \varphi_2$，那么$Mod(\varphi_1) \subseteq Mod(\varphi_2)$。所以对于同一个遗忘公式$\psi$，有$Mod(Forget_F(\varphi_1, \psi)) \subseteq Mod(Forget_F(\varphi_2, \psi))$，因此$Forget_F(\varphi_1, \psi) \models Forget_F(\varphi_2, \psi)$，前置单调性成立。

令$\varphi' = Forget_F(\varphi_1 \vee \varphi_2, \psi)$，则$Mod(\varphi') = Mod(\varphi_1 \vee \varphi_2) \cup \{\omega \in Force(\omega', \neg\psi) | \omega' \in Mod(\varphi_1 \vee \varphi_2)\} = Mod(\varphi_1) \cup \{\omega \in Force(\omega', \neg\psi) | \omega \in Mod(\varphi_1)\} \cup Mod(\varphi_2) \cup \{\omega \in Force(\omega', \neg\psi) | \omega \in Mod(\varphi_2)\}$，所以$Forget_F(\varphi_1 \vee \varphi_2, \psi) \equiv Forget_F(\varphi_1, \psi) \vee Forget_F(\varphi_2, \psi)$，分配律成立。

例 3.4: 设$\phi = a \wedge b \wedge c, \psi = \neg a \vee \neg b$，则$Forget_F(\phi, \psi) = (a \wedge b \wedge c)$，这是因为$\phi \models \neg\psi$。

上面的例子告诉我们，遗忘之后不能推出被遗忘公式，但是可以推出它的否定。接下来换一种思路，我们希望遗忘ψ之后，对于ψ的认识处于不确定状态，即我们不知道它的真假值。如果我们把对于ψ最不稳定的模型加入进来，那么得到一个新的算子。最不稳定的模型也就是最容易变化的，即我们稍微改变模型某个变元的真值得到新的模型却使得公式为假。先给出公式边缘的概念。

定义 3.9: 任意公式ψ的边缘是一公式，表示为$Margin(\psi)$，那么$Mod(Margin(\phi)) = \{\omega|\omega \in Mod(\phi)$ 且$Dis(\omega, \neg\phi)$ 是最小的$\}$。其中我们把边缘公式的模型称为ψ的边缘模型。

对于任意公式ϕ，它将整个可能世界Ω分为两部分，一部分模型使得ϕ为真，其余的为假。边缘公式描述的是ϕ与$\neg\phi$它们最相似的模型，这些模型处于最不稳定状态（也就是说，做最小的改变就可以使公式的非为真）。同样我们可以再给出$\neg\phi$的边缘公式，即$Mod(Margin(\neg\phi)) = \{\omega|\omega \in Mod(\neg\phi)$ 且$d(\omega, \phi)$是最小的$\}$。我们很容易得到下述命题。

命题 3.16: 任意$\omega \in Mod(Margin(\phi))$，有$Dis(\omega, \neg\phi) = 1$。

证明: 因为$\omega \in Mod(Margin(\phi))$，所以$\omega \in Mod(\phi)$，从而$Dis(\omega, \neg\phi) > 0$。我们只需要说明$Dis(\phi, \neg\phi) = 1$即可，假设$Dis(\phi, \neg\phi) > 1$，那么存在$\omega_1 \in Mod(\phi), \omega_2 \in Mod(\neg\phi)$使得$Dis(\omega_1, \omega_2) = Dis(\phi, \neg\phi) > 1$，那么$\omega_1, \omega_2$最短路径上肯定存在一个模型$\omega'$。若$\omega' \in Mod(\phi)$，则$Dis(\omega', \omega_2) < Dis(\phi, \neg\phi)$；若$\omega' \in Mod(\neg\phi)$，则$Dis(\omega_1, \omega') < Dis(\phi, \neg\phi)$，这两种情况都是不可能的（因为两个公式的距离就是它们各自模型集的最近模型的距离）。

根据上面的命题，某公式的边缘实际上选取的是距离其非为1的模型，这部分模型只要改变一个命题符号的真值就能使其否定公式为真。

如果对边缘公式再求边缘公式，结果不变，即边缘公式具有稳定性。

推论 3.1: 对于任意公式ψ，有$Margin(Margin(\psi)) \equiv Margin(\psi)$。

证明: 任意$\omega \in Mod(Margin(\psi))$，则有$Dis(\omega, \neg\psi) = 1$。另一方面，根据定义，$Margin(\psi) \models \psi$，因此$\neg\psi \models \neg Margin(\psi)$。那么得到$Dis(\omega, \neg Margin(\psi)) \leq Dis(\omega, \neg\psi) = 1$，而且$\omega \notin Mod(\neg Margin(\psi))$，所以$Dis(\omega, \neg Margin(\psi)) = 1$，从而$Margin(Margin(\psi)) \equiv Margin(\psi)$成立。

我们把遗忘公式及其非公式的边缘模型中距离原公式最近的模型放到结果集中就得到下述定义。

定义 3.10: $Forget_M(\phi, \psi)$ 表示我们新的算子，则$Mod(Forget_M(\phi, \psi)) = Mod(\phi) \cup \{\omega|\omega \in Margin(\psi)$ 并且$\nexists\omega' \in Margin(\psi), d(\omega', \phi) < d(\omega, \phi)\} \cup \{\omega|\omega \in Margin(\neg\psi)$ 且$\nexists\omega' \in Margin(\neg\psi), d(\omega', \phi) < d(\omega, \phi)\}$。

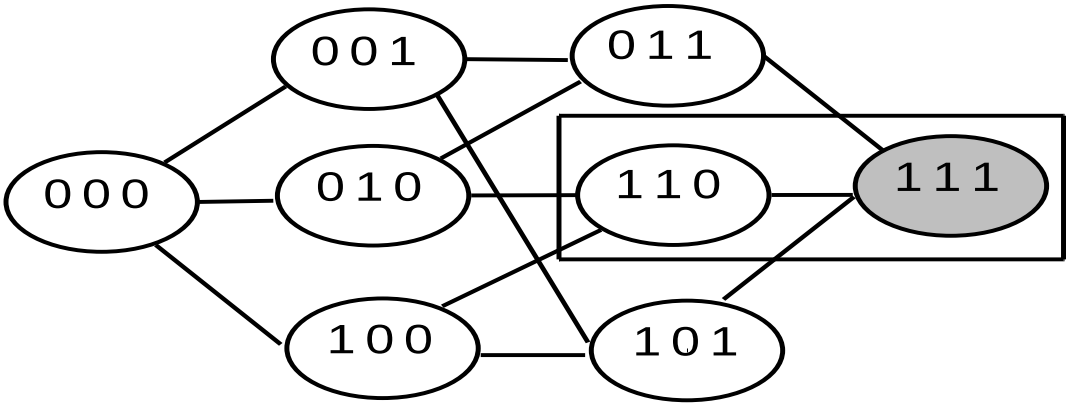

图 3.4　公式边缘模型图

接着讨论上面的例子。

例 3.5: 设$\phi = a \wedge b \wedge c, \psi = a \wedge b$，图3.4中阴影部分的111是$\phi$的模型，方框中两个模型是$\psi$的，显然$d(110, \neg\psi) = d(110, 010) = 1$，$d(111, \neg\psi) = d(111, 011) = d(111, 101) = 1$。所以$Mod(Margin(\psi)) = \{110, 111\}$，$Margin(\psi) = \psi$。类似地，$d(010, \psi) = d(011, \psi) = d(100, \psi) = d(101, \psi) = 1$，所以$Margin(\neg\psi) = (\neg a \wedge b) \vee (a \wedge \neg b)$。我们首先在$Mod(Margin(\psi))$中找到距离$\phi$最近的模型111，然后在$Mod(Margin(\neg\psi))$中找到距离$\phi$最近的模型有两个$011, 101$。所以$Forget_M(\phi, \psi) = (b \wedge c) \vee (a \wedge \neg b \wedge c)$。

命题 3.17: $Forget_M$ 满足(F1)-(F5)，对称性。

证明: 根据定义，$Forget_M(\phi, \psi)$算子结果包含比ϕ更多的模型，所以$\phi \models Forget_M(\phi, \psi)$，(F1)成立。

令$S = Var(\phi) - Var(\psi) \neq \emptyset$，对于由$S$建立的公式$\phi'$，假设有$\phi \models \phi'$，那么$Mod(\phi) \subseteq Mod(\phi')$。因为$Var(\phi') \cap Var(\psi) = \emptyset$，所以对于$\psi$和$\neg\psi$的边缘模型$\omega$，如果$Dis(\omega, \phi)$最小，那么$\omega \in Mod(\phi')$（否则，可以吧$\omega$变换成满足$\omega'$，使得$Dis(\omega', \phi) < Dis(\omega, \phi)$，这与$Dis(\omega, \phi)$最小矛盾）。因此我们得到$Forget_M(\phi, \psi)$定义中右边所有模型都是$\phi'$的模型，所以$Forget_M(\phi, \psi) \models \phi'$。反之，如果$Forget_M(\phi, \psi) \models \phi'$，显然$\phi \models \phi'$。

如果ψ不是重言式，那么我们是把满足$Dis(\omega, \psi) = 1$或者$Dis(\omega', \neg\psi) = 1$所有模型$\omega, \omega'$放到结果公式模型集中，所以显然包含使得$\psi$为假的模型，因此$Forget_M(\phi, \psi) \not\models \psi$，(F3)成立。

如果$\phi_1 \equiv \phi_2$且$\psi_1 \equiv \psi_2$，那么有$Mod(\phi_1) = Mod(\phi_2)$和$Mod(\psi_1) = Mod(\psi_2)$。因此$Forget_M(\phi_1, \psi_1) \equiv Forget_M(\phi_2, \psi_2)$。

因为$Mod(Forget_M(\phi, \psi))$已经包含$\psi$和$\neg\psi$的边缘模型，所以再加入一次$\psi$和$\neg\psi$的边缘模型，模型结果集不变，我们得到$Forget_M(Forget_M(\phi, \psi), \psi) \equiv Forget_M(\phi, \psi)$，(F5)成立。

对于$Forget_M(\phi, \psi)$和$Forget_M(\phi, \neg\psi)$来说，我们都是在ϕ原有的模型中加入ψ和$\neg\psi$的模型，因此结果模型集相同，所以有$Forget_M(\phi, \psi) \equiv sForget_M(\phi, \neg\psi)$。

第四章　基于本原蕴涵式的公式遗忘

上一章我们从语义方面给出了遗忘的几个定义，并证明他们满足我们的遗忘公设。不幸的是，这些定义有些时候表现不太让人满意。像$Forget_V, Forget_L$不区分遗忘公式的形式，只要他们包含相同的文字或者变量，不管析取、合取等，结果都是一样的。这种遗忘过于粗糙，因为好多看起来结果不同的遗忘不能区分。比如，$Forget_V(a \wedge b \wedge c, a \wedge b) = c$，$Forget_V(a \wedge b \wedge c, \neg a \vee \neg b) = c$。对于算子$Forget_F$，我们得到$Forget_F(a \wedge b \wedge c, a \wedge b) = (a \wedge b) vee (a \neg b \wedge c)$。如果我们在$a \wedge b \wedge c$中把$a \wedge b$看做整体，那么公式遗忘可看做文字遗忘，即$ForgetLit(t \wedge c, t) = c$，这里$t = a \wedge b$。而我们的算子$Forget_F$达不到这个效果。同样对于遗忘$Forget_M$，$Forget_M(a \wedge b \wedge c, a \wedge b) = (b \wedge c) \vee (a \wedge \neg b \wedge c)$，结果也不是$c$。因为我们之前的遗忘是基于语义的（通过模型变换得到），不满足这种语法形式的性质就不奇怪了。那么我们能不能基于语法形式给出遗忘定义来满足前面的观察结果，这就是下面基于本原蕴涵式的遗忘。幸运的是这种遗忘满足语法独立性，尽管它是基于语法的。

4.1 本原蕴涵式

正是由于变量或文字遗忘对应着本原蕴涵式（prime implicates）集合上的操作，在把遗忘算子推广到公式（或语句）遗忘时，我们的基本工具就是使用它。求解公式的本原蕴涵式有很多的方法 [66, 67, 68, 69, 70, 71, 72, 73, 74, 75]，并且广泛应用在知识编译、推理和及各种逻辑系统中 [76, 77, 78, 79, 80, 81, 82, 83, 84, 85]。

φ 是一个命题公式, 子句C 是φ 的一个蕴涵式当且仅当$\varphi \models C$。子句是文字的析取，通常可以表示为其组成文字的集合，当然解释为集合中文字的析取。

定义 4.1: 子句C 称为公式φ 的本原蕴涵式当且仅当它满足下面条件：

- C 是φ 的蕴涵式。
- 不存在其他的φ 的蕴涵式C' 满足$C' \models C$。

从这个定义可以看出，本原蕴涵式是公式所能推导出的子句中逻辑上最强的。通常一个公式的本原蕴涵式不止一个，而是一个子句的集合。而且，每一个公式和它的本原蕴涵式有着密切对应的关系，并且在等价条件下是一一对应的。

假设$PI(\varphi)$表示公式φ的本原蕴涵式构成的集合，$\wedge(PI(\varphi))$表示集合$PI(\varphi)$中所有子句的合取。显然，根据上面定义，此集合中任意两个元素相互是不可推出的。下面的性质告诉我们，对于任意的公式，只需要考虑它的本原蕴涵式就可以了，其他的蕴涵子句都是多余的。

命题 4.1: 对于任意命题公式φ，我们有$\varphi \equiv \wedge(PI(\varphi))$。

例 4.1: 假设公式$\varphi = (a \wedge b) \vee (\neg a \wedge c)$，那么

$$PI(\varphi) = \{a \vee c, b \vee c, \neg a \vee b\}$$

下面的质蕴含项是和本原蕴涵式相对应的，本原蕴涵式是公式的必要条件且是子句，质蕴含项是公式的充分条件且是项（即是文字的合取），它们来自于公式推导的两个方面。φ是一个命题公式, 项t称为φ的一个蕴涵项当且仅当$t \models \varphi$。

本原蕴涵式是最强的必要条件，而质蕴含项是最弱的充分条件。

定义 4.2: 项t称为公式φ的质蕴含项当且仅当它满足下面条件：

- 项t是φ的蕴涵项。
- 不存在φ其他的蕴涵项t'满足$t \models t'$。

类似地，公式的质蕴含项一般不是唯一的，是一个项的集合。对于质蕴含项，假设$IP(\varphi)$代表公式φ的所有质蕴含项的集合，$\vee(IP(\varphi))$表示集合$IP(\varphi)$中元素的析取，那么我们有下面类似的性质：

命题 4.2: 对于任意命题公式φ，我们有$\varphi \equiv \vee(IP(\varphi))$。

例 4.2: 还是看前面的例子，公式$\varphi = (a \wedge b) \vee (\neg a \wedge c)$，那么

$$IP(\varphi) = \{a \wedge b, \neg a \wedge c, b \wedge c\}$$

在不引起歧义的情况下，我们用PI代表某公式的本原蕴含式集合，IP表示某公式质蕴含项的集合。

遗忘和本原蕴涵式的关系早已给出，它的操作非常简单。既然遗忘一个原子变量集合的结果公式等价于某个不包含这个集合中的任何原子的公式，那么

在本原蕴涵式集合中去掉包含这些原子变量的子句就能自然得到我们遗忘之后的公式。

命题 4.3: （[24]）假设φ是一个命题公式，V是一原子变量集合，那么$Forget(\varphi, V) \equiv \wedge\{C \in PI(\varphi) \mid C$ 中不含V 中的原子$\}$。

实际上，记$S = \{C \in PI(\varphi) \mid C$ 中不含V 中的原子$\}$，则集合S中的子句就是$Forget(\varphi, V)$的所有本原蕴含式。

命题 4.4: 假设φ是一个命题公式，V是一原子变量集合，那么

$$PI(\wedge S) = S$$

。

此时，称S为关于PI的不动点。

这个结论对于文字遗忘也是成立的，不过我们删除的子句是包含遗忘文字的，不是包含变量的。也就是对于某个变量p，当遗忘文字$\neg p$时，所有包含$\neg p$的本原蕴涵式都要删去，但是包含p的子句可以留下来。就是说，遗忘文字$\neg p$，我们的结果可能包含p的正文字。

命题 4.5: （[25]）假设φ是一个命题公式，L是一文字集合，那么$ForgetLit(\varphi, L) \equiv \wedge\{C \in PI(\varphi) \mid C$ 中不含L 中的文字$\}$。

例 4.3: 假设$S = \{a \vee b, \neg b \vee c, a \vee c\}$，则有$\wedge S \equiv (a \vee b) \wedge (\neg b \vee c)$，并且$PI(\wedge S) = S$，所以$S$是关于$PI$的不动点。

如果令$S' = \{a \vee b, \neg b \vee c\}$，那么$\wedge S' \equiv \wedge S$。因此有$PI(\wedge S') = S$，而$S \neq S'$，所以$S'$不是关于$PI$的不动点。

同样地，记$S' = \{C \in PI(\varphi) \mid C$ 中不含L 中的文字$\}$，则集合S'中的子句就是$Forget(\varphi, L)$的所有本原蕴含式，即S'也是关于PI的不动点。

命题 4.6: 假设φ是一个命题公式，L是一原子变量集合，那么

$$PI(\wedge S') = S'$$

。

通过上面两个结论，遗忘和本原蕴涵式存在着某种必然的联系，这就为我们在推广遗忘概念时利用它做好了准备。一旦我们求出公式的所有本原蕴涵式，那么变量遗忘或文字遗忘都可以在线性时间（关于遗忘变量或文字的个数和本原蕴涵式的个数）内得到结果。

4.2 子句遗忘

由于逻辑公式表示形式的多样性，以及转化为标准范式（比如合取范式或者析取范式）的规模通常会很大，所以直接考虑一般的公式遗忘无疑是很复杂的。既然我们想要通过本原蕴涵式来讨论公式遗忘，而本原蕴涵式本身就是子句的形式，那么先考虑子句遗忘就是顺理成章的事情。直觉告诉我们，定义子句遗忘也不是一件简单的事情。在给出定义之前，我们需要一些准备工作。像变量或者文字遗忘一样，我们的步骤也是找到和遗忘子句相关的本原蕴涵式并且清除他们，不过仅有这一步骤是远远不够的。例如，公式φ表示我们要考虑的知识库，C_1, C_2, C' 在其本原蕴涵式集合中，子句C' 包含遗忘子句C，并且C' 是C_1, C_2 消解的结果，那么C' 必须被删除，而C_1, C_2 一般不包含C但是不能同时存在于结果集中（否则，C' 还在集合中）。因此在考虑删除某基本蕴涵式时，还需要考虑通过消解得到这些子句的基本蕴涵式，接着删除，直到不存在这种消解关系，也就是说，剩余的本原蕴涵式中不存在可以消解出已经删除的本原蕴涵式。而文字（变量）遗忘只需要删除包括文字（变量）的基本蕴涵式即可，剩余的子句不可能消解出这些子句（因为他们不含遗忘的文字（变量））。

另一方面，在本原蕴涵式集合中，若某个本原蕴涵式被遗忘子句所包含也应该被排除。这是由于遗忘某子句之后，结果不能蕴含该子句。在变量或者文字遗忘中，我们不需要考虑这种情况，因为变量或文字都只有一个符号变量，或者他们可以看做最简单的子句，不可能再将他们分解。同样地，为了确保消除某些子句，我们需要继续考虑子句之间的消解关系。

在寻找与遗忘子句相关的子句并删除的过程中，我们经常面对几个子句不能共存的情况，即它们中至少要删除一个。直觉上，选择删除那些与遗忘子句更近的子句，因此必要给出一种合理的子句距离定义从而定义一种偏序关系来帮助我们选择删除哪些子句。既然子句具有相同的范式即都是文字的析取，那么一个很好的量度是考虑两个子句具有的不同文字的数目。假设C_1, C_2 是两个子句，$|C_1 - C_2|$ 表示在C_1 而不在C_2 中文字的个数，$Dis(C_1, C_2)$ 代表C_1 和C_2的

距离。我们将其定义如下：

$$Dis(C_1, C_2) = \frac{|C_1 - C_2| \cdot |C_2 - C_1|}{|C_1| \cdot |C_2|}$$

。

分子表示属于其中一个子句而不属于另一个的文字数目，分母表示出现在两个子句中所有的文字的数量（不考虑重复）。采用这种分数形式是将距离标准化。显然$0 \leq Dis(C_1, C_2) \leq 1$，$Dis(C_1, C_2) = 0$ 当且仅当$C_1 = C_2$，$Dis(C_1, C_2) = 1$ 当且仅当C_1 和C_2 没有任何相同的文字。也就是说，当两个子句完全一样时距离最小，当它们无关时（即它们没有描述共同的对象）距离最大。

假设a, b, c, d 代表四个命题变量，那么

$$Dis(a \vee b, \neg a \vee b \vee c) = \frac{3}{4}$$

$$Dis(a \vee b, \neg a \vee d) = 1$$

$$Dis(a \vee b, c \vee d) = 1$$

$$Dis(a \vee b, a \vee b) = 0$$

有了子句距离，关于子句C 的偏序$\leq_C$ 容易定义为：$C_1 \leq_C C_2$ 当且仅当$Dis(C_1, C) \leq Dis(C_2, C)$。如果$C_1 \leq_C C_2$ 并且$C_2 \not\leq_C C_1$，那么$C_1 < C_2$，即C_1 严格小于C_2。在偏序关系$\leq_C$ 下越小的子句，其与C 的距离越小，显然C 是此偏序关系下的最小元。当多个子句不能共存于结果集中时，根据相应的偏序关系来决定它们的去留。

变量或文字遗忘是在原公式的本源蕴含集上进行的，即删除包含遗忘变量或文字的本原蕴涵式，剩余的本原蕴涵式的合取即作为我们遗忘的结果。但是在公式φ 中遗忘子句C，类似地在$PI(\varphi)$ 上操作容易产生歧义。比如

$$PI(\phi) = \{C_0 = a \vee b \vee c \vee d \vee f, C_1 = a \vee b \vee c \vee d \vee \neg e, C_2 = a \vee b \vee f \vee e, C_3 = a \vee b \vee \neg e \vee \neg h, C_4 = c \vee d \vee h, C_5 = a \vee b \vee f \vee \neg h\}$$

$C = a \vee b \vee c \vee d \vee f$。图4.1 展示了它们之间的消解关系。蕴含遗忘子句$a \vee b \vee$

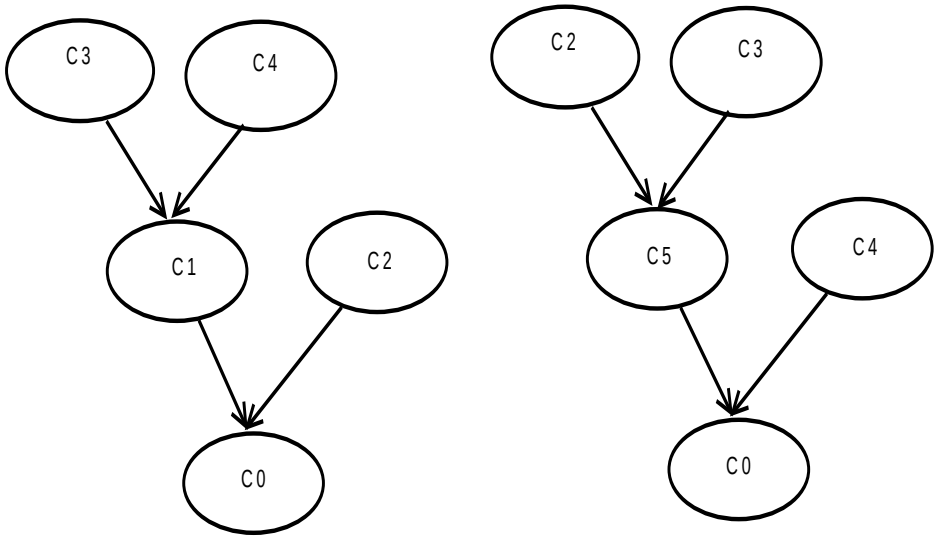

图 4.1　消解森林

$c \vee d \vee f$的子句集总共有四个：

$$S_0 = \{C_0\}, S_1 = \{C_1, C_2\}, S_2 = \{C_2, C_3, C_4\}, S_3 = \{C_4, C_5\}$$

根据子句距离的定义,我们必须删除S_0 中的C_0 ，S_1 中的C_1，S_2 中的C_2，S_3 中的C_5。从而我们得到剩余集合是$\{C_3, C_4\}$。但是C_1 是C_3, C_4 的消解结果，因此C_1 实际上并没有被删除，这当然不是我们所期望的。

图4.1也表明我们只需要关注两个子句消解的情况，三个及以上子句消解可以转化为多次的两个子句消解。首先我们找到子集$S_1 = \{C_1, C_2\}$和$S_3 = \{C_4, C_5\}$。对于S_1，如果我们去掉C_1，那么接下来C_3 或者C_4 必须也被删除因为他们消解出C_1，从而我们破坏掉子集S_2。如果我们删除C_2，S_2 也被破坏。不管消解树有多高，我们总能逐步破坏掉所有能消解出遗忘子句的子句集。上面的步骤顺序是合理的，在消解出与遗忘子句相关的子句时，参与的子句越多，其重要性就越低。最少个数当然只有一个子句，上例中是C_0；接下来是两个子句，即$S_1 = \{C_1, C_2\}$ 和$S_3 = \{C_4, C_5\}$，因为他们消解出C_0；再往下就是考虑能消解出S_1 中的C_1 或者C_2，以及能消解出S_3 中的C_4 或者C_5，这样我们实际上是找到了包含三个子句的子句集$S_2 = \{C_2, C_3, C_4\}$。比较S_1 和S_2，我们得到C_1 和C_2 比C_3 和C_4 更关键。因此我们把图4.1中的结构转变成图4.2。

图4.2描述的步骤也可以由图4.1这样得到：对于一个消解二叉树森林，先考

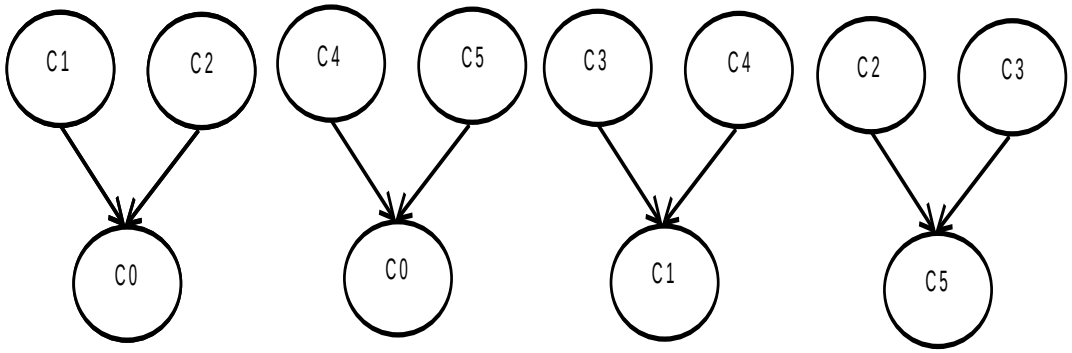

图 4.2　二元消解图

虑每个树的根节点（这种二叉树是倒置的，最下面的那个节点是根），如果它们在遗忘过程中是要删除的，那么接着考虑它们的儿子节点，然后是孙子节点等等，直到叶子节点。

为了只考虑两个子句消解的形式，我们使用本原蕴涵式的消解闭包代替本原蕴涵式集合，下面是它的定义。

定义 4.3: 假设φ是一命题公式，$PI(\varphi)$表示其本原蕴涵式构成的集合，称$CPI(\varphi)$为消解闭包，如果满足$CPI(\varphi) = \{C' \mid C' \in PI(\varphi)$或者$C'$是$PI(\varphi)$中子句消解的结果且$C' \neq \top\}$。

类似地，我们用CPI表示某个公式的本原蕴含式消解闭包。

一般来说，我们有$PI(\varphi) \subseteq CPI(\varphi)$，即$CPI(\varphi)$中的子句不一定是公式$\varphi$的本原蕴涵式。比如，$PI(\varphi) = \{a \vee b, \neg b \vee c \vee d, a \vee d\}$，但是$CPI(\varphi) = \{a \vee b, \neg b \vee c \vee d, a \vee d, a \vee c \vee d\}$，所以$PI(\varphi) \subset CPI(\varphi)$。$a \vee c \vee d$之所以不是本原蕴含式是由于$a \vee d$是本原蕴含式。

根据定义，如果$C' \in CPI(\varphi)$且$C' \notin PI(\varphi)$，那么$C' \neq \top$且C'是$PI(\varphi)$中子句消解的结果，这里$C' \neq \top$是由于排除消解结果为$\top$的情况，从而使得$CPI(\varphi)$中包含我们所关心的子句，达到简化的目的。比如，$PI(\varphi) = \{a \vee b, \neg a \vee \neg b\}$，那么$a \vee b$和$\neg a \vee \neg b$消解出$\top$。对于这个例子，得到$PI(\varphi) = CPI(\varphi)$。

之所以在消解闭包上操作，一方面是我们只需要考虑两个子句消解出另一个子句，不必考虑多个子句消解的情况。另一方面如果我们在$PI(\varphi)$中删除子句，某些新的子句会成为结果公式的本原蕴涵式，而他们之前未出现

在$PI(\varphi)$中。假设$PI(\varphi) = \{a \vee b \vee \neg c, c \vee d \vee e, a \vee b \vee d\}$. 如果我们想在$\varphi$ 中遗忘$a \vee b \vee d$，仅删除$a \vee b \vee d$ 是不行的。注意到$PI(\wedge\{a \vee b \vee \neg c, c \vee d \vee e\}) = \{a \vee b \vee \neg c, c \vee d \vee e, a \vee b \vee d \vee e\}$。$a \vee b \vee d \vee e$ 不应该保留因为它包含$a \vee b \vee d$。$a \vee b \vee \neg c$ 和$c \vee d \vee e$ 不能同时存在因为他们消解出$a \vee b \vee d \vee e$。$a \vee b \vee d \vee e$ 出现在CPI但不在PI 中。所以我们很容易在CPI 中发现与遗忘子句相关的隐含子句。

接下来的性质说明，对于变量或文字遗忘，在PI 和CPI 上操作得到逻辑等价的结果。既然我们要考虑的子句甚至公式遗忘是比变量和文字遗忘更复杂具体的操作，而且文字本身就可以被看做最简单的子句和公式，所以这个定理是必要的，并且说明我们在集合CPI 上定义更一般遗忘是合理性。

定理 4.1: 假设φ 是一命题公式，V 是一变量集合，L 是文字集合。PI_V 和PI_L 分别表示在$PI(\varphi)$中删除包含V 中变量和包含L 中文字子句剩下的集合。CPI_V 和CPI_L 类似定义。那么$\wedge(PI_V) \equiv \wedge(CPI_V)$ 和$\wedge(PI_L) \equiv \wedge(CPI_L)$。

证明: 我们只证明对于文字集L，第二个等价式成立。假设$PI(\phi) = X_1 \cup X_2$，X_1 表示包含L 中文字的子句集而X_2 中不包含。$CPI(\phi) = X_1 \cup X_2 \cup X_3 \cup X_4$，$X_i(i = 1, \cdots, 4)$ 互不相交。$X_3 \cup X_4$ 中的子句是$PI(\phi)$ 中子句消解出的新子句。X_3 包含L 中的文字而X_4 不包含。我们只需证明$\wedge X_2 \equiv \wedge(X_2 \cup X_4)$即可。$\forall C \in X_4$，必定存在本原蕴涵式$C' \in PI(\phi)$ 满足$C' \models C$，因为$\phi \models C$。因为C 不包含L中的文字所以$C' \in X_2$ 。因此$\wedge X_2 \equiv \wedge(X_2 \cup X_4)$ 并且$\wedge(PI_V) \equiv \wedge(CPI_V)$。类似地，$\wedge(PI_L) \equiv \wedge(CPI_L)$ 也成立。

根据变量或文字遗忘的性质，由上述定理我们很容易得到下面推论。

推论 4.1: 对于任意公式φ 和变量集合V 与文字集合L，我们有

$$ForgetVar(\varphi, V) \equiv \wedge\{C \in CPI(\varphi) \mid Var(C) \cap V = \emptyset\}$$

$$ForgetLit(\varphi, L) \equiv \wedge\{C \in CPI(\varphi) \mid Lit(C) \cap L = \emptyset\}$$

无论是在PI还是在CPI上执行操作，遗忘都是要删除掉其中部分子句，我们把这部分子句的集合定义为遗忘集。

定义 4.4: 在集合$CPI(\phi)$ 中，关于C 的遗忘集用$FS(\phi, C)$ 表示，如下递归定义：

$FS_1 = \{B|B \in CPI(\phi), Dis(B,C) = 0\}$
$FS_{i+1} = \cup_{B \in FS_i}\{C'|\exists C_1, C_2 \in (CPI(\phi) - \cup_{k=1}^{i} FS_k)$,$B$ 是C_1, C_2 的消解结果，C' 是$\{C_1, C_2\}$ 中在$\leq_C$下较小的$\}$ $i = 1, 2, \cdots$

如果对于某个整数n，我们有$FS_{n+1} = \emptyset$，那么$FS(\phi, C) = \cup_{i=1}^{n} FS_i$。

既然$CPI(\varphi)$ 是有限的并且递归的每一步至少删除一个子句，所有上述过程必定终止。遗忘集是我们找到的所有与遗忘子句最相关的子句集，它是我们要删除的。有了这个概念，我们就可以定义子句遗忘。

定义 4.5: 对于任意公式ϕ 和子句C，遗忘子句用$ForgetCla(\phi, C)$表示，那么

$$ForgetCla(\phi, C) = \wedge(CPI(\phi) - FS(\phi, C))$$

接下来我们看几个简单的例子。

例 4.4: 假设$\phi = a \wedge b \wedge c$，$C = a \vee b$，那么$CPI(\phi) = PI(\phi) = \{a, b, c\}$。因为$a \vDash C, b \vDash C$，$ForgetCla(\phi, C) = c$。

例 4.5: 假设$\phi = a \vee b \vee c, C = a \vee b$，那么$CPI(\phi) = PI(\phi) = \{a \vee b \vee c\}$，那么$C \vDash a \vee b \vee c$，我们有$ForgetCla(\phi, C) = \top$。

下面我们来看子句遗忘的性质。在$ForgetCla$ 定义中，集合$CPI(\varphi) - FS(\varphi, C)$ 包含$ForgetCla(\varphi, C)$ 的所有本原蕴涵式，当然可能有多余的子句（被此集合中其他子句所蕴含）。

命题 4.7: 对任意公式φ 和子句C，$CPI(\varphi) - FS(\varphi, C)$ 是消解封闭的。

证明: 假设存在一个子句$C' \notin CPI(\varphi) - FS(\varphi, C)$，满足：$CPI(\varphi) - FS(\varphi, C)$ 中有两个子句C_1, C_2，C' 是C_1, C_2 的消解。因为$CPI(\varphi)$ 是消解封闭的，所以$C' \in CPI(\varphi)$，从而有假设可知，$C' \in FS(\varphi, C)$。那么必定存在一个正整数k，使得$C' \in FS_k$。那么按照遗忘集的定义，C_1, C_2 至少有一个在集合FS_{k+1}中，因此属于遗忘集，这与C_1, C_2 都属于集合$CPI(\varphi) - FS(\varphi, C)$ 矛盾。命题得证。

命题 4.8: 对任意公式φ 和子句C，我们有

$$PI(ForgetCla(\varphi, C)) \subseteq CPI(\varphi) - FS(\varphi, C)$$

证明: 假设$X = CPI(\phi) - FS(\phi, C)$。$\forall C' \in PI(ForgetCla(\phi, C))$，我们要证明$C' \in CPI(\phi) - FS(\phi, C)$。由于$C'$ $ForgetCla(\phi, C)$ 的本原蕴涵式，假设存在子句$C'' \in CPI(\phi) - FS(\phi, C)$ 满足$C'' \models C'$，那么$C'' = C'$。所以这种情况下$C' \in X$。如果有一个子句集$X' \subseteq X$ 满足$X' \models C'$，那么C' 必定是X' 的消解。(否则，存在X' 的消解C''，满足$C'' \models C'$ 并且$C' \neq C''$，这与C' 是本原蕴涵式矛盾。) 由上一个命题我们得知X 是消解封闭的，因此$C' \in X$。

下面这个命题说明了遗忘的本质：忘记了的东西不会存在于随后的知识库，也不会对它产生影响。

命题 4.9: 对于任意公式φ 和子句C，我们有

$$ForgetCla(\varphi, C) \not\models C, C \not\models ForgetCla(\varphi, C)$$

证明: 假设$ForgetCla(\varphi, C) \models C$，那么存在$C' \in PI(ForgetCla(\phi, C))$ 使得$C' \models C$。当然$C = C'$ 可能成立。根据上述命题得知$C' \in CPI(\phi) - FS(\phi, C)$。但是因为$Dis(C', C) = 0$ 所以$C' \in FS(\phi, C)$，这是不可能的。因此$ForgetCla(\varphi, C) \not\models C$。另一方面，我们把所有满足$C \models C'$ 子句C' 放入集合$FS(\phi, C)$中，所以$C \not\models ForgetCla(\phi, C)$ 显然成立。

显然命题逻辑中，文字可以看做是最简单的子句，因此文字遗忘应该是子句遗忘的特例。尽管子句遗忘定义在消解闭包上，不过由于文字遗忘在本原蕴涵式集合和其消解闭包上的操作是等价的，这为下面性质奠定了基础。

命题 4.10: 任意公式φ 和文字l，$ForgetCla(\phi, l) \equiv ForgetLit(\phi, l)$ 成立。

证明: 按照之前的结论，$ForgetLit(\phi, l) \equiv \wedge\{C \in CPI(\varphi) \mid l$ 不出现在C 中$\}$。根据遗忘集的定义，$FS_1 = \{B \in CPI(\varphi) \mid B$ 中包含文字$l\}$，那么集合$CPI(\varphi) - FS_1$ 中的任意子句都不包含文字l。而任意不含文字l 的子句不可能消解出含文字l 的子句，所以$FS_2 = \emptyset$，过程终止。因此无论是根据子句遗忘还是文字遗忘的定义，其结果都是一样的。

下一个性质符合我们的直观，遗忘过程是知识减少的过程，所以子句遗忘也是一种弱化。

命题 4.11: 对于任意公式φ 和子句C，我们有

$$\varphi \models ForgetCla(\varphi, C)$$

证明: 因为$\varphi \equiv \wedge CPI(\varphi)$ 和$ForgetCla(\varphi, C) \equiv \wedge(CPI(\varphi) - FS(\varphi, C))$，所以$\varphi \models ForgetCla(\varphi, C)$。

尽管遗忘的定义并不是直接操作公式的模型，但是由于每一个公式与本原蕴涵式集合是一一对应的，这就是把逻辑公式都转换为标准范式（本原蕴涵式集合的合取），所以这种定义也满足语法独立性。

命题 4.12: 任意公式φ_1, φ_2 和子句C，如果$\varphi_1 \equiv \varphi_2$，那么

$$ForgetCla(\varphi_1, C) \equiv ForgetCla(\varphi_2, C)$$

证明: 由本原蕴涵式的性质可知，等价的命题公式有相同的本原蕴涵式，并且命题公式等价于其本原蕴涵式集合的逻辑合取。对于等价公式φ_1 和φ_2，遗忘都是操作同样的本原蕴涵式集合，所以等价公式遗忘的结果在逻辑等价下是唯一的。

遗忘一个子句并不是要遗忘掉子句中的所有文字，遗忘结果通常还会包含有此子句中出现的文字。从这个意义上来说，文字遗忘比子句遗忘要弱。

例 4.6: 假设公式$CPI(\varphi) = \{a \vee b, \neg b \vee c, a \vee c\}$，那么$ForgetCla(\varphi, a \vee b) \equiv (\neg b \vee c) \wedge (a \vee c)$，$ForgetCla(\varphi, a \vee c) \equiv \top$，而$ForgetLit(\varphi, \{a, b\}) \equiv \top$ 和$ForgetLit(\varphi, \{a, c\}) \equiv \top$。这里我们有$ForgetCla(\varphi, a \vee b) \models ForgetLit(\varphi, \{a, b\})$，$ForgetCla(\varphi, a \vee c) \equiv ForgetLit(\varphi, \{a, c\})$。

命题 4.13: 对于任意公式φ 和子句C，有

$$ForgetCla(\varphi, C) \models ForgetLit(\varphi, Lit(C))$$

其中$Lit(C)$ 表示子句C 中出现文字的集合。

证明: 任意$C' \in FS(\varphi, C)$，则C' 一定包含C 中出现的某些文字，这是因为任意两个不含C 中文字的子句不可能消解出含C 中文字的子句。但是并不是所有包

含C 中文字的子句都属于遗忘集，而$ForgetLit(\varphi, Lit(C))$正是排除所有包含$C$中文字的子句，因此$ForgetLit(\varphi, Lit(C))$ 去掉的子句个数不会比遗忘集少，我们得到$ForgetCla(\varphi, C) \models ForgetLit(\varphi, Lit(C))$。

对于子句遗忘来说，$ForgetCla(\varphi_1 \vee \varphi_2, C) = ForgetCl(\varphi_1, C) \vee ForgetCl(\varphi_1, C)$ 一般不成立。比如，$ForgetCla((a \vee b) \vee (b \vee c), a \vee c) \equiv \top$，但是$ForgetCla(a \vee b, a \vee c) \equiv a \vee b$，且$ForgetCla(b \vee c, a \vee c) \equiv b \vee c$，因此$ForgetCla(a \vee b, a \vee c) \vee ForgetCla(b \vee c, a \vee c) \equiv a \vee b \vee c$。

关于推导关系的单调性，子句遗忘并不满足。例如，$\varphi = a \vee b$，$\psi = a \vee b \vee c$，显然$\varphi \models \psi$。但是$ForgetCla(\varphi, a \vee b \vee d) = \top$，而$ForgetCla(\psi, a \vee b \vee d) = a \vee b \vee c$。所以$ForgetCla(\varphi, a \vee b \vee d) \not\models ForgetCla(\psi, a \vee b \vee d)$。

最后我们说明基于本原蕴涵式的子句遗忘也满足遗忘公设。

命题 4.14: 算子$ForgetCla$满足(F1)-(F5)。

证明: 由命题4.15得出(F1)成立。

假设$\varphi \models \varphi'$，且φ'不含C中任何变量符号，那么由文字遗忘的性质得$ForgetCla(\varphi, Lit(C)) \models \varphi'$。再由命题4.17得$ForgetCla(\varphi, C) \models \varphi'$。所以(F2)成立。

由命题4.13得(F3)成立。

若$\varphi_1 \equiv \varphi_2$，那么$CPI(\varphi_1) = CPI(\varphi_2)$。根据定义，

$$ForgetCla(\phi_1, C) = \wedge(CPI(\phi_1) - FS(\phi_1, C))$$

$$ForgetCla(\phi_2, C) = \wedge(CPI(\phi_2) - FS(\phi_2, C))$$

因此有$ForgetCla(\phi_1, C) \equiv ForgetCla(\phi_2, C)$，(F4)成立。

任意$C' \in CPI(ForgetCla(\varphi, C))$，那么有$C' \in CPI(\varphi) - FS(\varphi, C)$，所以$Dis(C', C) > 0$，因此$FS(ForgetCla(\varphi, C), C) = \emptyset$，我们得到$ForgetCla(ForgetCla(\varphi, C), C) = ForgetCla(\varphi, C)$，(F5)成立。

4.3 公式遗忘

有了子句遗忘，下面考虑更一般的公式遗忘。前面说过，一个命题公式等价于其本原蕴涵式的合取，这样我们可以把公式看做是子句集合。但

是对子句集合的遗忘不能等价于递归地遗忘每一个子句，因为结果与子句顺序相关。比如，$CPI(\varphi) = \{a \vee b \vee c, a \vee b \vee \neg d, c \vee d\}$，我们考虑遗忘子句集合$\{a \vee b \vee cc \vee d\}$。那么$ForgetCla(ForgetCla(\varphi, a \vee b \vee c), c \vee d) = \top, ForgetCla(ForgetCla(\varphi, c \vee d), a \vee b \vee c) = a \vee b \vee \neg d$，所以结果与遗忘顺序是相关的，从而类似于变量集合或文字集合的递归定义不适合子句集合的遗忘。既然单子句遗忘就是要消除和此子句相关的所有子句（即先找出距离为零的子句，再依次找出能够消解出它们的子句直到终止），那么对子句集合，我们希望在消解闭包中找到与此集合中每个子句相关的子句并删除，定义如下：

定义 4.6: 假设φ 是一公式，S 是子句集，C 是一子句，那么我们定义**子句集遗忘** 如下：

$$ForgetClas(\varphi, S) \equiv \bigwedge \bigcap_{C \in S} (CPI(\varphi) - FS(\varphi, C))$$

显然文字集也可以看做子句集，子句集遗忘和文字集遗忘是一致的。

命题 4.15: 对于任意公式φ和文字集合L，$ForgetClas(\varphi, L) \equiv ForgetLit(\varphi, L)$。

证明: 记$S = \{C \mid C \in CPI(\varphi), Lit(C) \cap L \neq \emptyset\}$，根据文字遗忘的性质有，$ForgetLit(\varphi, L) = \bigwedge(CPI(\varphi) - S)$。而$ForgetClas(\varphi, L) \equiv \bigwedge \bigcap_{l \in L}(CPI(\varphi) - FS(\varphi, l))$，其中$FS(\varphi, l)$表示$CPI(\varphi)$中包含文字$l$的子句集合。所以$ForgetClas(\varphi, L) = \bigwedge(CPI(\varphi) - S)$。

被遗忘的公式不一定是子句，但是可以等价看做其本原蕴涵式的合取，从上面子句集合上的遗忘，下面我们给出一般公式遗忘的概念。

定义 4.7: 任意两个公式φ, ψ，我们使用$ForgetFor(\varphi, \psi)$ 表示在φ 中遗忘公式ψ，那么

$$ForgetFor(\varphi, \psi) = ForgetCla(\varphi, PI(\psi))$$

例 4.7: 假设$\varphi = a \wedge b \wedge c, \psi = a \wedge b$，则$PI(\psi) = \{a, b\}$ 并且$CPI(\varphi) = PI(\varphi) = \{a, b, c\}$。因此$ForgetFor(\varphi, \psi) \equiv \bigwedge(\{b, c\} \cap \{a, c\}) \equiv c$。

例 4.8: 我们再考虑绪论中的那个例子. $\varphi = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \wedge b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge d)$，那么$CPI(\phi) = \{a \vee b \vee c \vee d, \neg a \vee \neg b, \neg a \vee \neg c, \neg a \vee \neg d, \neg b \vee \neg c, \neg b \vee \neg d, \neg c \vee \neg d\}$。$PI(\psi) = \{\neg a \vee \neg b, \neg a \vee \neg c, \neg a \vee \neg d, \neg b \vee \neg c, \neg b \vee \neg d, \neg c \vee \neg d\}$。因此我们有$ForgetFor(\phi, \psi) = a \vee b \vee c \vee d$。

定义中第二维变量采用的是PI而不是CPI，这是由于PI足矣描述一个公式，CPI中一般会包含多余的子句（即被其他字句所蕴含）。对于遗忘的第一维公式使用CPI的目的仅仅是为了按照比较合理的方式顺序删除相关的子句。一般的，$ForgetClas(\varphi, PI(\psi)) \models ForgetClas(\varphi, CPI(\psi))$。假设$CPI(\varphi) = \{a \vee d, a \vee b, \neg b \vee c, a \vee c\}$，$PI(\psi) = \{a \vee b, \neg b \vee c \vee d, a \vee c\}$，所以$CPI(\psi) = \{a \vee b, \neg b \vee c \vee d, a \vee c, a \vee c \vee d\}$。那么$ForgetClas(\varphi, PI(\psi)) = a \vee d$，但是$ForgetClas(\varphi, CPI(\psi)) = \top$。

子句显然是公式的特例，所以下面的性质是显然的，并且单子句遗忘和单子句集遗忘结果是相同的。

命题 4.16: 任意公式ϕ和子句C，有

$$ForgetFor(\phi, C) = ForgetCla(\phi, C)$$

$$ForgetClas(\phi, \{C\}) = ForgetCla(\phi, C)$$

下面的定理说明，在特殊情况下，我们可以把被遗忘的公式看做整体，从而可以转化为变量遗忘。

定理 4.2: 任意两个公式φ, ψ, $\varphi(t/\psi)$ 表示在φ 中用t 取代所有的ψ，如果φ 可以用ψ 和一些不在$Var(\psi)$ 中的变量表达，那么

$$ForgetFor(\varphi, \psi) = ForgetLit(\varphi(t/\psi), t)$$

证明: 假设$\psi = t$，$PI(\varphi(t/\psi)) = S_1 \cup S_2 \cup S_3 = \{t \vee C_1, \cdots, t \vee C_m, \neg t \vee C'_1, \cdots, \neg t \vee C'_n, C''_1, \cdots, C''_k\}$。$S_1$ 代表包含文字t 的子句集，S_2 表示包含文字$\neg t$的子句集，S_3 代表剩余子句构成集合。假设$PI(\psi) = \{D_1, \cdots, D_s\}$ 并且$PI(\neg\psi) = \{D'_1, \cdots, D'_t\}$。我们用$\wedge PI(\psi)$取代$S_1$ 中的每一个t且用$\wedge PI(\neg\psi)$取代S_2中的每一个$\neg t$ 然后我们展开公式的形式。S_1 假定变成$S'_1 = \{D_1 \vee C_1, \cdots, D_s \vee C_1, \cdots, D_1 \vee C_m, \cdots, D_s \vee C_m\}$，$S_2$ 成为$S'_2 = \{D'_1 \vee C'_1, \cdots, D'_t \vee C'_1, \cdots, D'_1 \vee C'_n, \cdots, D'_t \vee C'_n\}$。现在我们首先证明$PI(\varphi) = S'_1 \cup S'_2 \cup S_3$。一方面对于任何$\varphi$的蕴涵$C$，$\phi(t/\psi) \models C$。则有$\exists C' \in PI(\phi(t/\psi))$ 满足$C' \models C$。如果$C' = t \vee C_i, (1 \leq i \leq m)$，那么$C$ 可以由子句C_1 和C_2得到，使得$t \models C_1, C_i \models C_2$。因此必定有$D_j, (1 \leq j \leq s)$ 满足$D_j \models C_1$ 和$D_j \vee C_i \in S'_1$。

对于$C' \in S_2$证明类似。如果$C' \in S_3$，那么$C' \in PI(\phi)$。另一方面，我们要证明$\forall C_0, C_0' \in S_1' \cup S_2' \cup S_3$，$C_0 \not\models C_0'$和$C_0' \not\models C_0$。如果$C_0 \in S_i', C_0' \in S_i'(i = 1, 2, 3)$或者$C_0 \in S_1', C_0' \in S_3$或者$C_0 \in S_2', C_0' \in S_3$，它成立因为任何$PI(\phi(t/\psi))$中的子句不能包含它中另外一个。来看最后一种情况：$C_0 \in S_1', C_0' \in S_2'$。如果$D_i \in PI(\psi), D_j' \in PI(\neg\psi)$, 那么$D_i$和$D_j'$互相不包含。否则，$D_i \models D_j'$。我们有$\psi \models D_j'$和$\neg\psi \models D_j'$，那么$\psi \vee \neg\psi \models D_j'$，也就是说，$D_j' \equiv \top$。这是不可能的。从而我们证明了$PI(\phi) = S_1' \cup S_2' \cup S_3$。

接下来计算$CPI(\varphi)$。称$\{C_1, \cdots, C_m, C_1', \cdots, C_n', C_1'', \cdots, C_k''\}$中子句为$C-$子句且$\{D_1, \cdots, D_s, D_1', \cdots, D_t'\}$中子句为$D-$子句。那么$C-$子句和$D-$子句没有共同变量。我们通过$C-$子句和$D-$子句消解得到$CPI(\varphi)$。新产生的子句可以分为七类：

(1)由S_1'中子句消解得到的，用$S_{1,1}$代表它们的集合。

(2) 由S_2'消解得到的，用$S_{2,2}$表示。

(3) 由S_3得到的，用$S_{3,3}$表示。

(4)由S_1'和S_2'中子句消解得到的，表示为$S_{1,2}$。

(5)由S_1'和S_3得到的，表示为$S_{1,3}$。

(6)由S_2'和S_3得到的，表示为$S_{2,3}$.

(7)由S_1'，S_2'和S_3消解得到，用$S_{1,2,3}$表示。

在上面七个集合和三个集合S_1', S_2', S_3执行遗忘操作。先计算FS_1。$S_1' \subseteq FS_1$因为S_1'中子句包含$PI(\psi)$中某个子句。很容易看出S_2'和S_3中子句不属于FS_1。我们得到$S_{1,1}$通过$C-$子句消解和$D-$子句消解，它们中每一个都包含$PI(\psi)$中某个子句，因此$S_{1,1} \subseteq FS_1$。$S_{2,2}$和$S_{3,3}$与FS_1不相交。这两个集合中的子句都被$\wedge(S_2' \cup S_3)$蕴含。现在我们证明，对于情形(4)，每一个$C-$子句消解出$\top$。对于任意$1 \leq i \leq l$和$1 \leq j \leq s$，$\neg D_i \models \neg\psi$由于$\psi \models D_i$和$\neg D_j' \models \psi$因为$\neg\psi \models D_j'$。$\neg D_i \wedge \neg D_j' \models \psi \wedge \neg\psi$。因此$\neg D_i \wedge \neg D_j' \equiv \bot$。那么我们证明$D_i \vee D_j' \equiv \top$。因此$D-$子句消解参与产生$S_{1,2}$中的子句。那么$S_{1,2}$中每个子句包含$S_3$中子句因为$S_1$和$S_2$子句的消解是$S_3$中某子句或者包含它，它们中某个子句可能被忽略。对于(5)和(6)，$S_{1,3} \subseteq FS_1$且$S_{2,3} \wedge FS_1 = \emptyset$。(7) 的证明类似于(4)，$S_1'$和$S_2'$中的$D-$子句消解一定出现。这些子句都被$S_3$中子句蕴含。$FS_1$中每个子句或在$S_1'$或由$S_1'$和其他集合子句消解得到。剩余集合$S$不会消解出已经去掉的子句并且它们都被$\{S_2' \cup S_3\}$中子句蕴含。因此$FS_2 = \emptyset$，并且$\wedge S \equiv \wedge(S_2' \cup S_3) \equiv ForgetLit(\varphi(t/\psi), t)$。

有两种特殊情形：一种是$PI(\varphi(t/\psi)) = \{t, \neg t \vee C'_1, \cdots, \neg t \vee C'_n, C''_1, \cdots, C''_k\}$；另一种是$PI(\varphi(t/\psi)) = \{t \vee C_1, \cdots, t \vee C_m, \neg t, C''_1, \cdots, C''_k\}$。这两种很容易证明，从略。

例 4.9: $\phi = (\neg\psi \wedge b) \vee (\psi \wedge c), \psi = d \wedge e$。那么$\phi = [(\neg d \vee \neg e) \wedge b] \vee (c \wedge d \wedge e)$，所以$PI(\phi) = \{c \vee \neg d \vee \neg e, b \vee c, b \vee d, b \vee e\}$。因此$ForgetFor(\phi, \psi) = \wedge\{c \vee \neg d \vee \neg e, b \vee c\}$，并且有$ForgetCla(\phi(t/\psi), t) = \wedge\{\neg t \vee c, b \vee c\} = \wedge\{c \vee \neg d \vee \neg e, b \vee c\}$。

前面子句遗忘的性质对公式来说同样成立，而且在极端情况下，我们有下面的性质。

命题 4.17: 对于任意两个公式φ, ψ，我们有

$$ForgetFor(\varphi, \varphi) = \top$$

$$ForgetFor(\top, \psi) = \top$$

$$ForgetFor(\bot, \psi) = \bot$$

$$ForgetFor(\varphi, \bot) = \phi$$

证明: 因为$ForgetFor(\varphi, \varphi) = ForgetClas(\varphi, PI(\varphi))$，所以在遗忘过程中，$PI(\varphi)$ 中的本原蕴涵式首先被去掉，剩余子句都属于$CPI(\varphi) - PI(\varphi)$。任意$CPI(\varphi) - PI(\varphi)$中的子句$C$，必定存在$C' \in PI(\varphi)$使得$C' \models C$。所以$Dis(C, C') = 0$，那么$C$ 也属于遗忘集。这样$CPI(\varphi)$中所有子句都会被删除，所以$ForgetFor(\varphi, \varphi) = \top$。

$ForgetFor(\top, \psi) = \top$ 显然成立，因为$\top$ 是最弱的逻辑常量。

逻辑常量$\bot$ 没有任何模型，其本原蕴涵式集合为空，所以在遗忘过程中没法删除相关子句，即$ForgetFor(\bot, \psi) = \bot$，并且$ForgetFor(\varphi, \bot) = \phi$。

第一个等式直觉意义是遗忘掉整个知识库就相当于什么也不知道；第二三两个式子是说两个逻辑常量遗忘之后不变；最后一个是遗忘掉错误的知识不会对原知识库有影响。

下面的性质与子句遗忘类似。

命题 4.18: 对于任意命题公式φ和ψ，有

$$ForgetFor(\varphi,\psi) \not\models \psi$$

$$\psi \not\models ForgetFor(\varphi,\psi)$$

证明: 如果$ForgetFor(\varphi,\psi) \models \psi$成立，那么存在$C \in PI(ForgetFor(\varphi,\psi))$和$C' \in PI(\psi)$使得$C \models C'$，那么$C$必定在遗忘过程中被删除，不可能还存在于遗忘结果集$PI(ForgetFor(\varphi,\psi))$中。所以$ForgetFor(\varphi,\psi) \not\models \psi$成立。同理，假设$\psi \models ForgetFor(\varphi,\psi)$，那么存在$C \in PI(ForgetFor(\varphi,\psi))$和$C' \in PI(\psi)$使得$C' \models C$成立，这样$C$必定在遗忘过程中被删除。所以$\psi \not\models ForgetFor(\varphi,\psi)$成立。

命题 4.19: 对于任意命题公式φ和ψ，有$\varphi \models ForgetFor(\varphi,\psi)$。

证明: 因为$\varphi \equiv \wedge CPI(\varphi)$，而公式遗忘无非就是在集合$CPI(\varphi)$中删除部分子句，然后取剩余子句的合取作为遗忘的结果，所以遗忘之后的公式是原公式的弱化。

下一个性质是更一般意义上的语法独立性。

命题 4.20: 对于任意命题公式φ_1,φ_2和ψ_1,ψ_2，如果$\varphi_1 \equiv \varphi_2$且$\psi_1 \equiv \psi_2$，那么

$$ForgetFor(\varphi_1,\psi_1) \equiv ForgetFor(\varphi_2,\psi_2)$$

证明: 逻辑等价的命题公式有相同的本原蕴涵式集合及其消解闭包，公式遗忘定义在被遗忘公式的消解闭包和遗忘公式的本原蕴涵式集合上，所以等价公式遗忘结果是相同的。

多次遗忘与一次遗忘的结果是相同的，也就是通过一次遗忘就能把相关的信息去掉。

命题 4.21: φ,ψ是两个公式，则有

$$ForgetFor(ForgetFor(\varphi,\psi),\psi) = ForgetFor(\varphi,\psi)$$

证明: 根据定义得，$ForgetFor(\varphi,\psi) = \wedge \cap_{C\in PI(\psi)}(CPI(\varphi) - FS(\varphi,C))$。既然$PI(ForgetFor(\varphi,\psi)) \subseteq \cap_{C\in PI(\psi)}(CPI(\varphi) - FS(\varphi,C))$，那么$FS(ForgetFor(\varphi,\psi),\psi) = \emptyset$。所以有$ForgetFor(ForgetFor(\varphi,\psi),\psi) = ForgetFor(\varphi,\psi)$。

如果我们记$ForgetFor^n(\varphi,\psi)$表示在$\varphi$中遗忘$\psi$ n 次，那么$ForgetFor^n(\varphi,\psi) = ForgetFor(\varphi,\psi)$。特别地，$n=0$时，$ForgetFor^0(\varphi,\psi) = \varphi$。

命题 4.22: 对于任意公式φ 和ψ，有

$$ForgetFor(\varphi,\psi) \models ForgetLit(\varphi, Lit(\psi))$$

其中$Lit(\psi)$ 表示公式ψ 中出现文字的集合。

证明: 证明类似，略。

命题 4.23: $ForgetFor$算子满足(F1)-(F5)。

证明: 由命题4.24得(F1)成立。

假设$\varphi \models \varphi'$，且φ'不含ψ中任何变量符号，那么由变量遗忘的性质得$ForgetCla(\varphi, Var(\psi)) \models \varphi'$。再由命题4.27得$ForgetFor(\varphi,\psi) \models \varphi'$。所以(F2)成立。

由命题4.23得(F3)成立。

由命题4.25满足语法无关性，(F4)成立。

由命题4.26得遗忘多次等价于遗忘一次公式的结果，(F5)成立。

如果被遗忘公式是项的形式，那么有下面简单方法求解，即等价于遗忘项中的每个文字。

命题 4.24: 假设φ是一命题公式，$t = l_1 \wedge \cdots \wedge l_n$（$l_1,\cdots,l_n$是$n$个文字），则

$$ForgetFor(\varphi,t) \equiv ForgetLit(\varphi,\{l_1,\cdots,l_n\})$$

证明: 因为$ForgetFor(\varphi,t) = ForgetClas(\varphi,\{l_1,\cdots,l_n\})$，而根据命题4.22得，$ForgetClas(\varphi,\{l_1,\cdots,l_n\}) = ForgetLit(\varphi,\{l_1,\cdots,l_n\})$。

第五章　遗忘的应用

5.1 用遗忘做归并

在这一部分我们把遗忘应用在知识归并中。通常我们的知识库来自于多个信息源，智能地处理这些分布式信息是很重要的。尤其是我们不能简单把他们放在一起，因为它们有可能存在冲突。信念归并就是决定一个群组的整体信念，如果它们本身协调的，归并就是一个平凡的问题；当存在矛盾时，怎样推理出合理的结果是归并考虑的重点。关于归并请参考文献 [21, 86, 87, 22, 88, 89, 29, 23]。

归并的问题从逻辑角度形式化为：假设$\Phi = \{\varphi_1, \cdots, \varphi_n\}$ 是包含来自n 个信息源的知识库集合，每个源的信息用一个命题公式φ_i 表示(因为我们只考虑单个信息源是协调的，而整个知识库集Φ 通常是存在矛盾的)，公式μ 表示在归并过程中必须满足的信息，归并的目标是得到一个新的知识库$\triangle_\mu(\Phi)$。

有两类归并的方法，其一是基于语法的归并，这种方法和知识表达形式有关，因此对于逻辑等价的知识库结果可能不一样。最常用的方法是取极大协调子集，一般来说这样的子集可能不止一个，这是从纯粹公式层面来操作（每一个公式是最基本的操作单位，要么保留，要么删除，只是为了保留一致性。）其二是基于语义的，该种方法是在$Mod(\mu)$ 中选择在某种意义下关于知识集最近的模型，最常用的有三个这种归并算子。假设Φ 是一个知识库并且ω, ω'是两个解释。

- Σ–距离表示一个解释和一个知识库之间的距离，表示为$d_\Sigma(\omega, \Phi) = \Sigma_{\varphi\in\Phi} d(\omega, \varphi)$。则我们有如下偏序：$\omega \leq_\Phi^\Sigma \omega'$ 当且仅当$d_\Sigma(\omega, \Phi) \leq d_\Sigma(\omega', \Phi)$。归并算子$\triangle_\mu^\Sigma$ 定义为：$mod(\triangle_\mu^\Sigma) = min(mod(\mu), \leq_\Phi^\Sigma)$。
- Max–距离表示一个解释和一个知识库的距离，定义为：$d_{Max}(\omega, \Phi) = Max_{\varphi\in\Phi} d(\omega, \varphi)$。则我们得到一个新的偏序：$\omega \leq_\Phi^{Max} \omega'$ 当且仅当$d_{Max}(\omega, \Phi) \leq d_{Max}(\omega', \Phi)$。归并算子$\triangle_\mu^{Max}$定义为：$mod(\triangle_\mu^{Max}) = min(mod(\mu), \leq_\Phi^{Max})$。
- 假设$\Phi = \{\varphi_1, \cdots, \varphi_n\}$，$d_j^\omega = d(\omega, \varphi_j)$。令$L_\omega^\Phi$ 把$(d_1^\omega, \cdots, d_n^\omega)$按降序排列的列表，$\leq_{lex}$ 是整数序列的字典序，那么偏序关系$\leq_\Phi^{GMax}$ 定义为：$\omega \leq_\Phi^{GMax} \omega'$ 当且仅当$L_\omega^\Phi \leq_{lex} L_{\omega'}^\Phi$。归并算子$\triangle_\mu^{GMax}$定义为：$mod(\triangle_\mu^{GMax} =$

$min(mod(\mu), \leq_{\Phi}^{GMax})$。

这种方法由于满足语法无关性被广泛采用，来看一个经典的例子。

例 5.1: 在一个小区业主联合会议上，组织方提出来年可能建立游泳池、网球场和停车场，但是如果建其中两个，租金将会明显增长。我们用S, T, P分别表示建游泳池、建网球场和建停车场，用I表示租金上涨。组织方阐述建任意两个将会是租金上涨，这表示为：$\mu = ((S \wedge T) \vee (S \wedge P) \vee (T \wedge P)) \rightarrow I$。

总共有四个小区业主，他们的意见集合表示为$\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$。其中两个业主这三项都想建但是不关心租金上涨与否：$\varphi_1 = \varphi_2 = S \wedge T \wedge P$。第三个人想让租金最低，因此他反对建任何东西：$\varphi_3 = \neg S \wedge \neg T \wedge \neg P \wedge \neg I$。第三个人认为小区真正需要的是网球场和停车场，但他不希望租金上涨：$\varphi_4 = T \wedge P \wedge \neg I$。

对于前面游泳池的例子，已有的三类归并算子结果如下：
$\triangle_{\mu}^{\Sigma}(\Phi) = S \wedge T \wedge P \wedge I$。
$\triangle_{\mu}^{Max} = (\neg S \wedge \neg T \wedge P) \vee (\neg S \wedge T \neg P) \vee (S \wedge \neg T \wedge P \wedge I)$。
$\triangle_{\mu}^{GMax} = \neg S \wedge \neg I \wedge ((\neg T \wedge P) \vee (T \wedge \neg P))$。

不过从命题符号角度看结果并不直观。对变量I，既然φ_1, φ_2不关心它并且φ_3, φ_4支持$\neg I$，因此文字公式$\neg I$可以被作为整个群组的观点。对变量S, T, P，前两个业主和第三个在它们每一个都有相反的意见，即前连个支持S, T和P而第三个支持它们的反面。因此如果任何一方不让步的话，他们不可能达成一致。从而$\neg I \wedge \mu$可以作为归并的自然结果。不幸的是，$\triangle_{\mu}^{\Sigma}(\Phi)$和$\triangle_{\mu}^{Max}$不能捕获$\neg I$。另一方面，尽管$\triangle_{\mu}^{GMax} \models \neg I$，这个算子太强了，$\triangle_{\mu}^{GMax} \models \neg S$. 两个业主支持$S$，一个支持它的反面且另一个中立。即使我们不用多数原则的话，也不能想当然认为$\neg S$.比较谨慎的处理是既不承认S，也不承认$\neg S$。

矛盾一般只是由于知识的某些方面造成的，在查询之前，我们完全可以过滤无关的信息，从而简化推理甚至消除矛盾。遗忘正是对原公式的弱化，并且是通过遗忘无关信息从而集中到所考虑的论域内。

我们考虑带完整性约束条件IC的归并，Φ是我们要归并的知识库集合，$\triangle_{\mu}(\Phi)$是我们的归并算子。$\triangle$是一个IC第三章介绍的知识归并公设类似于著名的信念修正的A.G.M假设，但是它们是有争议的。特别是**IC5**和**IC6**，它们想表达的意思是两个知识集若共有某些信息，则等价于同时归并这两个知识集

构成的集合。但是在基于逻辑的知识表示中，$\triangle_\mu(\Phi_1) \wedge \triangle_\mu(\Phi_2)$ 并不代表共有的信息，因为这个合取要比$\triangle_\mu(\Phi_1)$和$\triangle_\mu(\Phi_2)$ 逻辑上都强。从这个角度看，这两个公设在合理性上存在商榷，我们提出的基于遗忘的归并恰不满足这两个性质。

还可以考虑以下两个相对的性质。

大数特性 **(Maj)**：$\exists n \triangle_\mu(\Phi_1 \sqcup \Phi_2^n) \vdash \triangle_\mu(\Phi_2)$. 直觉上，如果一个子群组出现足够多次，那么它代表整个群组的观点。

大数无关性 要求我们只考虑不同的知识库，相同知识库只考虑一个，用**(MI)**表示：$\triangle_\mu(\Phi_1 \sqcup \Phi_2^n) \leftrightarrow \triangle_\mu(\Phi_1 \sqcup \Phi_2)$。

很容易看出$\triangle_\mu^\Sigma$满足**(IC0)**-**(IC8)**, **(Maj)**。$\triangle_\mu^{Max}$ 满足**(IC0)**-**(IC5)**，**(IC7)**，**(IC8)** and **(MI)**。尤其是不满足**(IC6)**，**(Maj)**。$\triangle_\mu^{GMax}$ 满足**(IC0)**-**(IC8)**。

之前提到的三类归并算子可以用遗忘来表示，在证明之前首先说明归并可以表示成扩张的形式，其次扩张可以转化为遗忘。先给出扩张的概念 [90]。扩张算子D 是一个从命题公式到命题公式的映射且满足：$mod(D(\varphi)) = \{\omega \in \mathcal{M} \mid d(\omega, \varphi) \leq 1\}$。

我们有$D^1(\varphi) = D(\varphi)$ 且$D^n(\varphi) = D(D^{n-1}(\varphi))$。因此$mod(D^n(\varphi)) = \{\omega \in \mathcal{M} \mid d(\omega, \varphi) \leq n\}$。

假设$\Phi = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$。这三类归并算子可以用扩张表示：

- $\triangle_\mu^\Sigma(\Phi) \equiv \vee_{c_1+\cdots+c_n=k}(D^{c_1}(\varphi_1) \wedge \cdots \wedge D^{c_n}(\varphi_n) \wedge \mu)$，这里$k$ 是使整个析取式协调的最小整数。
- $\triangle_\mu^{Max}(\Phi) \equiv D^k(\varphi_1) \wedge \cdots \wedge D^k(\varphi_n) \wedge \mu$，这里$k$ 是使$D^k(\varphi_1) \wedge \cdots \wedge D^k(\varphi_n) \wedge \mu$ 协调的最小整数。
- $\triangle_\mu^{GMax}(\Phi) \equiv \vee_{<c_1,\cdots,c_n> \in perm(T)}(D^{c_1}(\varphi_1) \wedge \cdots \wedge D^{c_n}(\varphi_n) \wedge \mu)$, 这里$T$ 是n元整数数组，满足使整个析取式协调且在降序排列时是字典序最小的。

在说明扩张和遗忘的关系前，需要下面两个命题。

引理 5.1: 假设φ 是一公式且V 是变量集，我们有$\forall \omega \in mod(\exists V.\varphi)$，$d(\omega, \varphi) \leq |V|$。

证明: $\forall \omega \in mod(\exists V.\varphi)$，必定存在$\omega' \in mod(\varphi)$使得$\omega, \omega'$ 除V中变量之外指定相同的真值。因此$d(\omega, \omega') \leq |V|$，从而$d(\omega, \varphi) \leq |V|$。

命题 5.1: 假设φ 是一协调公式，p是一命题变元，V 是变量集合且n是一正整数。我们有

- $D(\varphi) \equiv \vee_{p\in Var(\varphi)}\exists\{p\}.\varphi$。
- 如果$1 < n \leq |Var(\varphi)|$，那么$D^n(\varphi) \equiv \vee_{V\subseteq Var(\varphi),|V|=n}\exists V.\varphi$。
- 如果$n > |Var(\varphi)|$，那么$D^n(\varphi) \equiv \top$。

证明: 1. 我们将证明他们有相同的模型。$\forall\omega \in mod(D(\varphi))$，则$d(\omega,\varphi) \leq 1$。如果$d(\omega,\varphi) = 0$那么$\omega \in mod(\varphi)$。因此$\omega \in mod(\exists\{p\}.\varphi)$ 并且$\omega \in mod(\vee_{p\in Var(\varphi)}\exists\{p\}.\varphi)$成立。如果$d(\omega,\varphi) = 1$，那么有$\exists\omega' \in mod(\varphi)$并且$d(\omega,\omega') = 1$。所以$\omega,\omega'$ 排除$var(\varphi)$中变元解释相同。这样$\omega \in mod(\exists\{p\}.\varphi)$。否则，如果$\omega \in mod(\vee_{p\in Var(\varphi)}\exists\{p\}.\varphi)$ 那么$\omega \in mod(\exists\{p\}.\varphi)$ 对于某个p。因此根据上述引理，$d(\omega,\varphi) \leq 1$成立。

2. 如果$1 < n \leq |Var(\varphi)|$，那么$D^n(\varphi) = D(D^{n-1}(\varphi))$。假设$D^{n-1}(\varphi) = \vee_{V\subseteq Var(\varphi),|V|=n-1}\exists V.\varphi$，且$D^n(\varphi) = D(\vee_{V\subseteq Var(\varphi),|V|=n-1}\exists V.\varphi)$
$= \vee_{p\in Var(\varphi)}\exists\{p\}.(\vee_{V\subseteq Var(\varphi),|V|=n-1}\exists V.\varphi)$
$= \vee_{V\subseteq Var(\varphi),|V|=n}\exists V.\varphi \vee \vee_{V\subseteq Var(\varphi),|V|=n-1}\exists V.\varphi$
$= \vee_{V\subseteq Var(\varphi),|V|=n}\exists V.\varphi$。

最后一个结论由于遗忘的特性，实际上，$D^n(\varphi) = \vee_{V\subseteq Var(\varphi),|V|\leq n}forget(\varphi,V)$。在不引起歧义的情况下，我们省略下标中的$V \subseteq Var(\varphi)$，即$D^n(\varphi) = \vee_{|V|\leq n}forget(\varphi,V)$。

3.如果$n > |Var(\varphi)|$，那么$mod(D^n(\varphi)) = \{\omega \in \mathcal{M}|d(\omega,\varphi) \leq n\}$。$\exists Var(\varphi).\varphi \equiv \top$，则$\forall\omega \in \mathcal{M}$，那么$\omega \in mod(\exists Var(\varphi).\varphi)$。从而$d(\omega,\varphi) \leq |Var(\varphi)| < n$。因此$\omega \in mod(D^n(\varphi))$ 并且$D^n(\varphi) = \top$成立。

有了上面的命题，下一个命题告诉我们扩张可以用遗忘来表示。

命题 5.2: 假设φ 是一公式并且V是变量结合，p 是一命题变量，我们有

- $D(\varphi) \equiv \vee_{p\in Var(\varphi)}\exists\{p\}.\varphi$。
- 如果$1 < n \leq |Var(\varphi)|$，那么$D^n(\varphi) \equiv \vee_{V\subseteq Var(\varphi),|V|=n}\exists V.\varphi$。
- 如果$n > |Var(\varphi)|$，那么$D^n(\varphi) \equiv \top$。

有了上面的关系，用遗忘表示归并算子是自然而然的事情。

定理 5.1: 假设$\Phi = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$，那么

- $\triangle_{\mu}^{\Sigma}(\Phi) \equiv \vee_{|V_1|+\cdots+|V_n|=k}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu)$。
- $\triangle_{\mu}^{Max}(\Phi) \equiv \vee_{|V_1|=\cdots=|V_n|=k}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu)$。
- $\triangle_{\mu}^{GMax}(\Phi) = \vee_{<|V_1|,\cdots,|V_n|>\in perm(T)}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu)$。

这里k 和T 与上面说明相同。

证明: 假设$\triangle_{\mu}^{\Sigma}(\Phi) \equiv \vee_{c_1+\cdots+c_n=k}(D^{c_1}(\varphi_1) \wedge \cdots \wedge D^{c_n}(\varphi_n) \wedge \mu)$ 并且k 使整个析取式协调的最小整数。则$\triangle_{\mu}^{\Sigma}(\Phi) \equiv \vee_{c_1+\cdots+c_n=k}(\vee_{|V_1|=c_1}(\exists V_1.\varphi_1) \wedge \cdots \wedge \vee_{|V_n|=c_n}(\exists V_n.\varphi_n) \wedge \mu) \equiv \vee_{c_1+\cdots+c_n=k}(\vee_{|V_1|=c_1,\cdots,|V_n|=c_n}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu) \equiv \vee_{|V_1|+\cdots+|V_n|=k}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu)$。

$\triangle_{\mu}^{Max}(\Phi) \equiv D^k(\varphi_1) \wedge \cdots \wedge D^k(\varphi_n) \wedge \mu$，这里$k$ 是使$D^k(\varphi_1) \wedge \cdots \wedge D^k(\varphi_n) \wedge \mu$ 协调的最小整数。在上式中代入$D^k(\varphi_i) = \vee_{|V_i|=k}\exists V_i.\varphi_i$得，$\triangle_{\mu}^{Max}(\Phi) \equiv (\vee_{|V_1|=k}\exists V_1.\varphi_1) \wedge \cdots \wedge (\vee_{|V_n|=k}\exists V_n.\varphi_n) \wedge \mu = \vee_{|V_1|=\cdots=|V_n|=k}(\exists V_1.\varphi_1 \wedge \cdots \wedge \exists V_n.\varphi_n \wedge \mu)$。

$\triangle_{\mu}^{GMax}(\Phi) \equiv \vee_{<c_1,\cdots,c_n>\in perm(T)}(D^{c_1}(\varphi_1) \wedge \cdots \wedge D^{c_n}(\varphi_n) \wedge \mu)$，这里$T$ 是n元整数数组，满足使整个析取式协调且在降序排列时是字典序最小的。同样在上式代入得，$\triangle_{\mu}^{GMax}(\Phi) \equiv \vee_{<c_1,\cdots,c_n>\in perm(T)}(\vee_{|V_1|=c_1}\exists V_1.\varphi_1) \wedge \cdots \wedge \vee_{|V_n|=c_n}\exists V_n.\varphi_n) \wedge \mu) \equiv \vee_{<|V_1|,\cdots,|V_n|>\in perm(T)}(\exists V_1.\varphi_1) \wedge \cdots \wedge \exists V_n.\varphi_n) \wedge \mu$。

用遗忘表示归并算子形式类似于扩张的表示，既然他们有紧密的关系，我们可以直接通过遗忘定义归并。我们的方法是修改已有归并的遗忘表示获得新的算子，上述命题表明，不同知识库的遗忘变量可以不同，条件中只要求遗忘变量的数量在某种意义上是最少的。他们的不足是结果不能聚焦在某个特定的论域，特别是对于部分论域上的查询，整个知识库集尽管是不一致的，但是在这部分论域上可能是协调的。要想达到这个效果，必须限制每个知识库遗忘掉相同的原子，然后再尽量少地遗忘原子以保留一致性。

文字公式是从单变量建立最简单的子语言，对于文字公式查询，其余的变量不需要考虑。从前面游泳池的例子，一方面如果某些信息源支持一个文字公式，其他对此表示中立，那么整个群体应该支持此文字公式，因为这些信息源在此文字上并不矛盾。另一方面，某些知识库支持，某些知识库反对，更谨慎的态度是对其保留意见，既不支持也不反对。已有的归并算子不会同时满足这两方面要求，下面形式化这两条性质。

(A1) 假设知识库$\Phi' \subseteq \Phi$ 和l 是一文字公式并且$\mu \wedge l$ 是协调的。如果$\forall \varphi' \in \Phi', \varphi' \models l$且$Var(l) \notin Var(\Phi - \Phi')$，那么归并结果$\triangle_\mu(\Phi) \models l \wedge \mu$。

(A2) 假设$\varphi_1, \varphi_2 \in \Phi$。如果$\varphi_1 \models l$ 和$\varphi_2 \models \neg l$，那么$\triangle_\mu(\Phi) \not\models l$。

(A2) 隐含着$\triangle_\mu(\Phi) \not\models \neg l$ 也成立。

尽管一个知识库Φ 关于$Var(\Phi)$ 是冲突的，但它可以关于$Var(\Phi)$的某个子集是协调的。我们分别考虑关于集合包含关系和集合元素数的这种最大子集，得到下面两个定义。

定义 5.1: 假设$\Phi = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$ 且μ 是完整性约束。$V \subseteq Var(\Phi)$，最小遗忘集的集合$FS = \{V$ 关于集合元素数是极小的$| \wedge_{i=1}^n \exists V.\varphi_i \wedge \mu$ 协调$\}$。我们定义$\triangle_\mu^{f_1}$ 如下：

$$\triangle_\mu^{f_1}(\Phi) = \vee_{V \in FS}(\wedge_{i=1}^n \exists V.\varphi_i \wedge \mu)$$

在证明这个算子满足我们预期的性质前，先给出一个引理。

引理 5.2: 假设p是V 中的变量，l 是关于p的文字。如果$\forall 1 \leq i \leq n, \varphi_i \models l$，并且$\wedge_{i=1}^n \exists V.\varphi_i$ 是协调的，那么$\wedge_{i=1}^n \exists (V - \{p\}).\varphi_i$ 也是协调的。

证明: $\forall 1 \leq i \leq n, \varphi_i \models l$，因此我们得到$\exists (V - \{p\}).\varphi_i \models l$。假设$\omega \in mod(\wedge_{i=1}^n \exists V.\varphi_i)$，那么$\omega \in mod(\exists V.\varphi_i)$。如果$\omega \models l$ 那么$\omega \in mod(\exists (V - \{p\}).\varphi_i)$，并且有$\omega \in mod(\wedge_{i=1}^n \exists (V - \{p\}).\varphi_i)$。因此它是协调的。如果$\omega \models \neg l$，则存在模型$\omega' \in mod(\wedge_{i=1}^n \exists V.\varphi_i)$ 使得ω' 和ω 除了p以外指定相同的真值。它可以被简化为第一种情形并且$\omega' \in mod(\wedge_{i=1}^n \exists (V - \{p\}).\varphi_i)$。从而$\wedge_{i=1}^n \exists (V - \{p\}).\varphi_i$ 也是协调的。

命题 5.3: $\triangle_\mu^{f_1}$ 满足性质**(A1)**，**(A2)**。

证明: **(A1)** 假设$V' = Var(\Phi) - \{p\}$，如果$\forall \varphi' \in \Phi', \varphi' \models p$，那么$\forall \varphi' \in \Phi', \exists V'.\varphi' \models p$。因此$\exists V'.\varphi' \equiv p$，并且$\forall \varphi \in \Phi - \Phi', \exists V'.\varphi \equiv \top$。所以如果$\mu \not\models \neg p$ 那么$\wedge_{\varphi \in \Phi}(\exists V'.\varphi) \wedge \mu \equiv p \wedge \mu$ 是协调的。在$\triangle_\mu^{f_1}$的定义中，$\forall V \in FS, p \notin V$。否则，$V - \{p\}$ 满足$\wedge_{\varphi \in \Phi}(\exists (V - \{p\}).\varphi) \wedge \mu$ 协调的最小变量集。因此我们有$\triangle_\mu^{f_1}(\Phi) \models p$。

(A2) $\forall V \in FS, p \in V$ 成立，如若不然，假设$p \notin V$，令$\varphi_1 \models p$ 并且$\varphi_2 \models \neg p$，因此$\exists V.\varphi_1 \models p$并且有$\exists V.\varphi_2 \models \neg p$。$\exists V.\varphi_1 \wedge \exists V.\varphi_2$ 是不协调的，$V \notin FS$。证毕。

我们再来考虑游泳池的例子，$\varphi_1(\varphi_2)$ 和φ_3 在变量S,T,P指定相反的真值，因此这些变量应该被遗忘以保留一致性。$V=\{S,T,P\}$ 是FS 中唯一的集合，$\triangle_{\mu}^{f_1}(\Phi)=\wedge_{i=1}^{4}\exists V.\varphi_i\wedge\mu=\neg I\wedge\mu=\neg I\wedge(\neg S\vee\neg T)\wedge(\neg S\vee\neg P)\wedge(\neg P\vee\neg T)$。

在我们证明$\triangle_{\mu}^{f_1}$满足那些性质钱前，下面的引理是必不可少的。

引理 5.3: 设φ,φ' 使两个命题公式且V 是变元集，我们有

$$d(\exists V.\varphi,\varphi')=d(\exists V.\varphi,\exists V.\varphi')$$

证明: 我们首先证明$d(\exists\{p\}.\varphi,\varphi')=d(\exists\{p\}.\varphi,\exists\{p\}.\varphi')$对任何变元$p$都成立。$Mod(\varphi')\subseteq Mod(\exists\{p\}.\varphi')$，所以$d(\exists\{p\}.\varphi,\exists\{p\}.\varphi')\leq d(\exists\{p\}.\varphi,\varphi')$。取$\omega\in Mod(\exists\{p\}.\varphi),\omega'\in Mod(\exists\{p\}.\varphi')$，则$d(\omega,\omega')=d(\exists\{p\}.\varphi,\exists\{p\}.\varphi')$。有四种情形：

1. 如果$\omega\in Mod(\varphi),\omega'\in Mod(\varphi')$，那么等式成立。

2. 如果$\omega\in Mod(\varphi),\omega'\in Mod(\exists\{p\}.\varphi')-Mod(\varphi')$，令$\omega_1,\omega_1'$ 分别与ω,ω'在p 上赋值不同，那么$d(\omega_1,\omega_1')=d(\omega,\omega')$ and $\omega_1\in Mod(\exists\{p\}.\varphi),\omega_1'\in Mod(\varphi')$，因此等式成立。

3. 如果$\omega\in Mod(\exists\{p\}.\varphi)-Mod(\varphi),\omega\in Mod(\varphi')$，那么等式成立。

4. 如果$\omega\in Mod(\exists\{p\}.\varphi)-Mod(\varphi),\omega'\in Mod(\exists\{p\}.\varphi')-Mod(\varphi')$，假设$\omega_1,\omega_1'$ 仅在p 上分别与ω,ω' 赋值不同，那么有$d(\omega_1,\omega_1')=d(\omega,\omega')$ 并且$\omega_1\in Mod(\varphi),\omega_1'\in Mod(\varphi')$。所以等式成立。

下面我们证明一般情况下$V(|V|>1)$也成立。令$V=\{p_1,\cdots,p_n\}$.

$d(\exists V.\varphi,\exists V.\varphi')$

$=d(\exists\{p_1\}.(\exists(V-\{p_1\}).\varphi),\exists\{p_1\}.(\exists(V-\{p_1\}).\varphi'))$

$=d(\exists\{p_1\}.(\exists(V-\{p_1\}).\varphi),\exists(V-\{p_1\}).\varphi')$

$=d(\exists V.\varphi,\exists(V-\{p_1\}).\varphi')$

$=d(\exists V.\varphi,\exists(V-\{p_1,p_2\}).\varphi')$

$=\cdots$

$=d(\exists V.\varphi,\varphi')$。

引理表明$d(\exists V.\varphi,\varphi')=d(\exists V.\varphi',\varphi)$. So $\exists V.\varphi$ 与φ' 是协调的当且仅当$\exists V.\varphi'$与φ是协调的。

很容易得出对于$V'\subseteq V$，有$d(\exists V.\varphi,\varphi')\equiv d(\exists V.\varphi,\exists V'.\varphi')$。但是$\exists V.\varphi\wedge\varphi'\equiv$

$\exists V.\varphi \wedge \exists V'.\varphi'$ 并不成立。比如，$\varphi \equiv p \wedge q, \varphi' \equiv \neg p$。$\exists\{p\}.\varphi \wedge \varphi' \equiv q \wedge \neg p$，但是$\exists\{p\}.\varphi \wedge \exists\{p\}.\varphi' \equiv q$。

有了上面的引理，我们就可证明算子所满足的归并假设。

命题 5.4: $\triangle_{\mu}^{f_1}(\Phi)$ 满足归并假设**(IC0)**-**(IC4)**，**(IC7)**，**(IC8)** 和**(MI)**。

证明: **(IC0)**-**(IC3)**显然对于$\triangle_{\mu}^{f_1}(\Phi)$来说成立。

$\triangle_{\mu}^{f_1}(\varphi \cup \varphi') \wedge \varphi = \vee_{V \in FS}(\exists V.\varphi \wedge \exists V.\varphi' \wedge \mu) \wedge \varphi = \vee_{V \in FS}(\varphi \wedge \exists V.\varphi')$ 和$\triangle_{\mu}^{f_1}(\varphi \cup \varphi') \wedge \varphi' = \vee_{V \in FS}(\varphi' \wedge \exists V.\varphi)$。根据上面引理得，$\exists V.\varphi \wedge \exists V.\varphi'$是协调的当且仅当$\exists V.\varphi \wedge \varphi'$ (或者$\exists V.\varphi' \wedge \varphi$)是协调的。因此**(IC4)** 也成立。

现在来证明**(IC7)**和**(IC8)**。FS, FS' 分别是$\triangle_{\mu_1}(\Phi)$ 和$\triangle_{\mu_1 \wedge \mu_2}(\Phi)$的遗忘集的集合。假设$\triangle_{\mu_1}^{f_1}(\Phi) = \vee_{V \in FS}(\wedge_{i=1}^{n} \exists V.\varphi_i \wedge \mu_1)$，如果$\triangle_{\mu_1}^{f_1}(\Phi) \wedge \mu_2$协调，那么$\exists V_0$ 使得$\wedge_{i=1}^{n} \exists V_0.\varphi_i \wedge \mu_1 \wedge \mu_2$ 是协调的。FS, FS' 两个集合具有相同的基数，所以$V_0 \in FS'$。相反地，有$\forall V' \in FS', V' \in FS$。则我们得到$\triangle_{\mu_1}^{f_1}(\Phi) \wedge \mu_2 \equiv \triangle_{\mu_1 \wedge \mu_2}^{f_1}(\Phi)$。

对于某个集合V，遗忘V之后，$\Phi_1 \sqcup \Phi_2^n$中公式的合取是协调的当且仅当遗忘V之后，$\Phi_1 \sqcup \Phi_2$中公式的合取是协调的。$\triangle_{\mu}^{f_1}(\Phi_1 \sqcup \Phi_2^n)$定义中遗忘集的集合is和$\triangle_{\mu}^{f_1}(\Phi_1 \sqcup \Phi_2)$的是一样的。知识集$\Phi_1 \sqcup \Phi_2^n$ 遗忘$V \in FS$后是
$\vee_{V \in FS}(\wedge_{\varphi \in \Phi_1} \exists V.\varphi \wedge_{i=1}^{n} (\wedge_{\psi \in \Phi_2} \exists V.\psi) \wedge \mu)$
$= \vee_{V \in FS}(\wedge_{\varphi \in \Phi_1} \exists V.\varphi \wedge_{\psi \in \Phi_2} \exists V.\psi \wedge \mu)$。这样遗忘$V$之后，$\Phi_1 \sqcup \Phi_2^n$ 和$\Phi_1 \sqcup \Phi_2$ 两个知识集变成一样，即$\triangle_{\mu}^{f_1}(\Phi_1 \sqcup \Phi_2^n) \equiv \triangle_{\mu}^{f_1}(\Phi_1 \sqcup \Phi_2)$。

The operator $\triangle_{\mu}^{f_1}$虽然并不满足**(IC5)** 和**(IC6)**，但是它满足我们的两个性质**(A1)**和**(A2)**。我们认为$\triangle_{\mu}(\Phi_1) \wedge \triangle_{\mu}(\Phi_2)$ 并不能表示两个群组共同的选择，这实际上应该是$Th(\triangle_{\mu}(\Phi_1)) \cap Th(\triangle_{\mu}(\Phi_2))$ ($Th(\varphi)$ 表示φ的推理演绎闭包)。如果我们关于集合包含关系选择最小集那么得到另一个算子。

定义 5.2: 假设$\Phi = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$且$\mu$ 是一完整性约束，$V \subseteq \cup_{i=1}^{n} Var(\varphi_i)$。极小遗忘集$FS = \{V$ 在集合包含关系下是极小的$|\ \wedge_{i=1}^{n} \exists V.\varphi \wedge \mu$ 协调$\}$。定义如下算子：

$$\triangle_{\mu}^{f_2}(\Phi) = \vee_{V \in FS}(\wedge_{i=1}^{n} \exists V.\varphi \wedge \mu)$$

对于社区游泳池的例子，$\triangle_{\mu}^{f_2}$ 和$\triangle_{\mu}^{f_1}$ 是相同的。

下面的定理类似，不幸的是，它不满足**(IC8)**。

命题 5.5: $\triangle_{\mu}^{f_2}$ 满足**(A1)**，**(A2)**，**(IC0)**-**(IC4)**，**(IC7)** 和**(MI)**。

由于$\triangle_{\mu}^{f_2}$ 不满足**(IC8)**，而$\triangle_{\mu}^{f_1}$ 满足，所以这两个算子一般是不同的。例如，假设$\Phi = \{\varphi_1, \varphi_2\}$，$\mu = \top$ and $\varphi_1 = \neg p \wedge \neg q \wedge \neg r \wedge \neg s$，$\varphi_2 = ((p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge r)) \wedge \neg s$。$\{a\}, \{b, c\}$ 是$\triangle_{\mu}^{f_2}$的遗忘集，但是$\{a\}$是$\triangle_{\mu}^{f_1}$的唯一一个。所以$\triangle_{\mu}^{f_1}(\Phi) \equiv \neg q \wedge \neg r \wedge \neg s$，而$\triangle_{\mu}^{f_2} \equiv \neg p \wedge \neg s$。

5.2 逻辑独立性

日常生活的许多场景中，我们都面临着的一个问题是判定什么是相关的，也就是信息的独立性([91,92,93,94])。尤其是在智能推理中，一种合理的方式是通过忽略无关的东西以提高推理的效率。比如在开始写这篇论文前，我只需要搜集与论题相关的材料，而完全不用看美食之类的书籍，因为他们对写这篇论文一点帮助也没有。侧重于相关的方面被认为是智能的主要特征，众所周知，我们大脑90%的神经元连接是被抑制并且不接受感知输入。无关性也是概率推理的中心主题 [95]，而且相关性也是定义信息过滤策略（information filtering policies） [96]和协作问答技术（cooperative answering techniques） [97]的主要概念，例如，当一个数据库用户不能形式表达他的查询时，一种方法是判定他感兴趣的论题并以结构化的方式返回关于此论题的数据，显然相干关系被用来捕获这种方法。这也揭示了为什么无关性（或相关性）是人工智能许多领域一个重要的概念（ [98]），尽管可能用不同的名字([99])，比如独立性（independence），非冗余的（irredundancy），非影响的（influenceability），分离的（separability），交互性（interactivity）。

相关性推理主要是对于一个知识库Σ，判定Σ能不能推出一个查询ϕ。有很多推理模式被使用，从经典的（经典推理）到复杂的（常识推理）。从计算复杂性角度，相关性被用来简化并使推理更高效。其他的推理问题，目的是明确获得特征集中的信息，比如告诉你所知道的关于tweety的事情，在这类问题，相关也扮演着重要作用，比如通过遗忘其他信息来表征我们关心的信息。

公式文字与公式变量独立性捕捉在变量真值和公式真值之间一些形式的独立性。粗略的讲，Σ依赖于l 因为它告诉某些关于l的信息，即存在上下文，当加入到Σ中时能推出l。比如$\Sigma = a \rightarrow b$依赖于b，因为b可以在假设a为真的情况下由Σ 推出。当然我们不能假定此上下文与Σ不协调或者就是l本身。

接下来先给出公式文字与公式变量的语法无关性定义。

定义 5.3: ([25]) 假设Σ 是一公式，l 是一文字，L是文字集。

- Σ 语法文字依赖于l（语法文字独立于l）当且仅当$l \in Lit(\Sigma)$（$l \notin Lit(\Sigma)$）。
- Σ 语法文字依赖于L当且仅当存在文字$l \in L$ 使得Σ 语法文字依赖于l。否则，Σ 语法文字独立于L。

例 5.2: 假设$\Phi = \neg(a \wedge b)$，Φ的否定范式是$(\neg a \vee \neg b)$；所以Φ语法文字依赖于$\neg a$和$\neg b$，而它语法文字独立于a 和b。这个例子告诉我们，在考虑语法独立性时，不是完全不管公式的形式，必须转化为否定范式。

下面这种公式变量的独立性或相关性是更基本的。

定义 5.4: ([25]) 假设Σ 是一公式，v 是一变量，V是变量集。

- Σ 语法变量依赖于v（语法变量独立于v）当且仅当$v \in Var(\Sigma)$（$v \notin Var(\Sigma)$）。
- Σ 语法变量依赖于V当且仅当存在变量$v \in V$ 使得Σ 语法文字依赖于v。否则，Σ 语法文字独立于V。

例 5.3: 假设$\Sigma = (a \wedge \neg b)$，那么$\Sigma$ 语法变量依赖于a 和b，但是语法变量独立于c。考察语法变量独立性时，只需要看变量符号有没有出现，不用看是正文字还是负文字，而且不用把公式转化为否定范式，从这个角度来说，语法变量独立性比语法文字独立性判断起来更容易。

语法变量独立性可以由语法文字独立性得到：Σ 语法变量独立于V当且仅当它语法文字独立于$V \cup \{\neg x | x \in V\}$，而且这两种独立性都可以在线性时间内判定。

这两种独立性有两个致命的弱点。第一，他们不满足语法无关性，即等价公式不总是语法文字或变量独立于同样的文字或原子。其次，语法独立性不能总是捕捉依赖性的直觉意义。比如，$\Sigma = a \wedge \neg b \wedge (a \vee b)$语法文字依赖于$a, \neg b, b$，而且有$\sigma \models \neg b$，相反地，在加入某种上下文时，$\Sigma \models b$不可能成立（如果不想产生不一致的话）。

所以从语义上给出独立性定义是必要的，而且是更有意义的。

定义 5.5: ([25]) 假设Σ是一公式，l 是一文字，L 是文字集合，那么

- Σ 文字独立于l，表示为$l \not\mapsto \Sigma$，当且仅当存在一个公式Φ使得$\Phi \equiv \Sigma$ 且Φ语法文字独立于l。否则，Σ 文字依赖于l，表示为$l \mapsto \Sigma$。所有满足$l \mapsto \Sigma$ 的文字l 的集合记为$DepLit(\Sigma)$。
- Σ 文字独立于L 记为$L \not\mapsto \Sigma$，当且仅当$L \cap DepLit(\Sigma) = \emptyset$。否则，$\Sigma$ 文字依赖于L 记为$L \mapsto \Sigma$。

来看一个例子。

例 5.4: 假设$\Sigma = a \wedge \neg b \wedge (a \vee b)$。我们有$\Sigma \equiv a \wedge \neg b$，所以$DepLit(\Sigma) = \{a, \neg b\}$，注意$\Sigma$ 语法文字独立于b。

定义 5.6: ([25]) 假设Σ是一公式，v 是一文字，V 是文字集合，那么

- Σ 变量独立于v，表示为$v \not\mapsto^{\pm} \Sigma$，当且仅当存在一个公式$\Phi$使得$\Phi \equiv \Sigma$ 且Φ语法变量独立于v。否则，Σ 变量依赖于v，表示为$l \mapsto^{\pm} \Sigma$。所有满足$v \mapsto^{\pm} \Sigma$ 的文字v 的集合记为$DepVar(\Sigma)$。
- Σ 变量独立于V 记为$V \not\mapsto^{\pm} \Sigma$，当且仅当$V \cap DepVar(\Sigma) = \emptyset$。否则，$\Sigma$ 变量依赖于L 记为$L \mapsto^{\pm} \Sigma$。

例 5.5: 假设$\Sigma = a \wedge \neg b \wedge (a \vee c)$，那么$DepVar(\Sigma) = \{a, b\}$，因为$\Sigma \equiv a \wedge \neg b$，所以$\Sigma$变量独立于$c$ 。

根据前面最简公式的定义，我们有如下命题。

命题 5.6: Σ是变量最简公式当且仅当$Var(\Sigma) = DepVar(\Sigma)$；$\Sigma$是文字最简公式当且仅当$Lit(\Sigma) = DepLit(\Sigma)$。

证明: 我们只证第一条性质，第二条类似。充分性。如果Σ是最简公式，那么对于任意变量$v \in Var(\Sigma)$，有$Forget(\Sigma, v) \not\equiv \Sigma$。假设$v \notin DepVar(\Sigma)$，那么存在一公式$\Phi \equiv \Sigma$使得$\Phi$语法变量独立于$v$，所以有$Forget(\Sigma, v) \equiv Forget(\Phi) \equiv \Sigma$。矛盾。

必要性。反之，假设$Var(\Sigma) = DepVar(\Sigma)$，则显然对于任意变量$v \in Var(\Sigma)$，有$Forget(\Sigma, v) \not\equiv \Sigma$。

语法独立性和语义独立性在最简公式条件下是一致的。

命题 5.7: ([25]) 假设Σ 是一公式，则

- Σ 是文字最简的当且仅当下列等价成立：对于任意文字集L，Σ 语法文字独立于L 当且仅当$L \not\vdash \Sigma$。
- Σ 是变量最简的当且仅当下列等价成立：对于任意变量集V，Σ 语法变量独立于V 当且仅当$L \not\vdash^{\pm} \Sigma$。

接下来的命题展示了独立性和遗忘的关系，也为判定独立性找到了另一种方法。

命题 5.8: ([25]) 假设Σ 是一公式，L是一文字集，那么$ForgetLit(\Sigma, L)$是文字独立于L且是Σ的逻辑最强推论。

推论 5.1: 假设Σ是一命题公式，L是一文字集，则Σ文字独立于L当且仅当$\Sigma \equiv ForgetLit(\Sigma, L)$成立。

下述推论在简化推理方面很有用。

推论 5.2: 如果公式φ 文字独立于L，那么$\Sigma \models \varphi$当且仅当$ForgetLit(\Sigma, L) \models \varphi$。

变量独立性有上述类似地性质。([25])

- 假设Σ 是一公式，V是一变量集，那么$ForgetVar(\Sigma, V)$是变量独立于V且是Σ的逻辑最强推论。
- 假设Σ是一命题公式，V是一变量集，则Σ变量独立于V当且仅当$\Sigma \equiv ForgetVar(\Sigma, V)$成立。
- 如果公式φ 变量独立于V，那么$\Sigma \models \varphi$当且仅当$ForgetVar(\Sigma, V) \models \varphi$。

命题 5.9: ([25]) Σ 是一个公式，L是文字集合，V是变量集合，那么

- $L \not\vdash \Sigma$ 当且仅当$PI(\Sigma) \subseteq \{C | C$是不包含$L$中文字的子句$\}$
- $V \not\vdash \Sigma$ 当且仅当$PI(\Sigma) \subseteq \{C | C$是不包含$V$中变量的子句$\}$

上面这条性质再一次论证了遗忘和独立性的关系，拿文字遗忘来说，它就相当于在本原蕴涵式集合中删除包含遗忘文字的子句，那么既然文字独立势必本原蕴涵式集合中的子句不包含此文字，那么遗忘此文字也就不用删除任何本原蕴涵式，所以遗忘结果不变。

还有一种条件独立性，可以被看做公式变量独立性的一般化。假设有3个变量集X, Y, Z 和一个命题公式Σ，X和Y是条件独立的当且仅当给定公式Σ 和Z，

增加关于X的信息到Σ中并不能推出任何关于Y的新知识。[100]集中展示了利用条件独立性对几种推理形式（包括演绎、诱导和诊断）的可计算价值，实际上，通过条件独立性，一个全局的计算可以被一局部高效的计算所取代。

前面回顾了公式变量独立性或公式文字独立性，并且说明了他们和遗忘有紧密的关系。我们希望把独立性的概念推广到公式之间，并且符合人们的直观的感觉。最自然的是，如果两个公式不包含共同的原子，那么他们显然是无关的。

定义 5.7: φ, ψ是两个公式，φ'是φ的变量最简公式，ψ'是ψ的变量最简公式，如果$Var(\varphi') \cap Var(\psi') = \emptyset$，那么称$\varphi$与$\psi$强独立的。

例 5.6: 假设$\varphi = a \wedge \neg b \wedge (a \vee c)$，$\psi = c \vee d$，因为$\varphi$的最简公式是$\varphi' = a \wedge \neg b$，并且$Var(\varphi') \cap Var(\psi) = \emptyset$，所以$\varphi$与$\psi$强独立的。尽管$c$也出现在公式$\varphi$中，但是它并不起作用。

这种强独立性在知识归并中有重要作用。假设有3个知识库构成的集合$\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$。其中$\varphi_1 = a \wedge \neg b$，$\varphi_2 = b \vee c$，$\varphi_3 = \neg a \vee \neg c$。我们的查询是$\psi = a$是否成立。如果我们同时承认这三个公式都为真，会产生矛盾。因为有$\varphi_1 \models a$与$\varphi_1 \models \neg b$，由此和φ_2消解出c，再和φ_3消解出$\neg a$，这是不可能的，因为已知能推出a为真。但是考虑到我们的查询只包含一个原子变量a，它与φ_2是强独立的，所以这个问题求解完全可以绕开φ_2，只需要在φ_1, φ_3推理，这样我们能得到a 为真。这种方式的合理性在于，3个知识库看做来自于不同的信息源，可以看作是相对独立的，互不影响。第二个信息源只是说了关于b, c的信息，没有关于a的。在考虑只含a的查询时，φ_2可以被忽略。如果同一个信息源既包含与查询有关的知识，也包含与查询无关的，那么我们在推理时不应该采取类似措施，因为同一知识库的公式不能单独来看，它们是相互影响的。

一般来说，对于两个公式φ, ψ，命题变元可以被分为4类：只属于φ的；只属于ψ的；同时属于φ, ψ的；都不属于这两者的。那么在我们遗忘掉其共同的命题原子之后就会得到两个强相关的公式。

命题 5.10: φ, ψ是两个公式，令$V = Var(\varphi) \cap Var(\psi)$，则有$ForgetVar(\varphi, V)$与$ForgetVar(\psi, V)$是强独立的。

证明: 显然$Var(ForgetVar(\varphi, V)) \cap Var(ForgetVar(\psi, V)) = \emptyset$，所以$ForgetVar(\varphi, V)$与$ForgetVar(\psi, V)$是强独立的。

上述性质中可能有$V = \emptyset$，这时φ, ψ本身就是强独立的。

当两个公式有公共原子变量，我们给出一种弱相关性。正像我们定义公式遗忘时的动机，遗忘掉一个公式就是删除与之相关的知识，那么剩余的知识应该是无关的。

定义 5.8: φ, ψ是两个公式，如果$\varphi \equiv ForgetFor(\varphi, \psi)$且$\psi \equiv ForgetFor(\psi, \varphi)$，那么称$\varphi$与$\psi$是弱独立的。

从定义可以看出弱独立性是对称的，即如果φ与ψ是弱独立的，那么ψ与φ是弱独立的，两个公式的地位是对称的。

例 5.7: 假设$\varphi = (a \vee \neg b) \wedge (b \vee c)$，$\psi = \neg a \wedge \neg c$，那么$PI(\varphi) = \{a \vee \neg b, b \vee c, a \vee c\}$，$PI(\psi) = \{\neg a, \neg c\}$。所以$ForgetFor(\varphi, \psi) \equiv \varphi$ 且$ForgetFor(\psi, \varphi) \equiv \psi$，因此它们是弱独立的。

公式遗忘的下述性质，表明弱独立定义可以简化为一个条件。

命题 5.11: φ, ψ是两个公式，如果$\varphi \equiv ForgetFor(\varphi, \psi)$，那么

$$\psi \equiv ForgetFor(\psi, \varphi)$$

证明: 根据定义，$ForgetFor(\varphi, \psi) = ForgetClas(\varphi, PI(\psi))$。因为$\varphi \equiv ForgetFor(\varphi, \psi)$，所以对于任意$C \in PI(\psi)$，$PI(\varphi)$中没有与之相关的子句（即任意$C' \in PI(\varphi)$，有$C \not\models C'$且$C' \not\models C$）。反之，对于任意子句$C' \in PI(\varphi)$，也不存在$PI(\psi)$中子句与之相关。所以$\psi \equiv ForgetFor(\psi, \varphi)$。

根据上述性质，我们可以把定义简化为下面的形式。

定义 5.9: φ, ψ是两个公式，如果$\varphi \equiv ForgetFor(\varphi, \psi)$（或者$\psi \equiv ForgetFor(\psi, \varphi)$），那么称$\varphi$与$\psi$是弱独立的。

命题 5.12: φ, ψ是任意两个公式，则$ForgetFor(\varphi, \psi)$与ψ弱独立，且$ForgetFor(\psi, \varphi)$与φ弱独立。

证明: 因为根据遗忘性质有，$ForgetFor(ForgetFor(\varphi, \psi), \psi) = ForgetFor(\varphi, \psi)$，所以$ForgetFor(\varphi, \psi)$与$\psi$弱独立，同理可证后面的弱独立性。

我们是从本原蕴涵式定义公式遗忘，对于弱独立的两个公式来说，他们的本原蕴涵式没有蕴含关系。

命题 5.13: φ,ψ是两个弱独立的公式，那么任意$C \in PI(\varphi)$，$C' \in PI(\psi)$，都有$C \not\models C'$且$C' \not\models C$。

证明: 因为φ,ψ是两个弱独立的公式，所以$ForgetFor(\varphi,\psi) = \varphi$。这样对于任意子句$C \in PI(\varphi)$，$C' \in PI(\psi)$，都有$Dis(C,C') > 0$。因此有$C \not\models C'$且$C' \not\models C$。

当其中一个公式为变量时，那么公式独立性就退化成公式变量独立性。与语义变量独立性比较，公式的强独立性就是公式变量独立性；公式的弱独立性不是（因为公式遗忘此时等价于文字遗忘，尽管不含此文字，但是可能含有其否定文字）。当其中一个公式看做文字时，就退化成公式文字独立性。公式的强弱独立性就等价于公式文字独立性。对于弱独立性是因为公式遗忘包含文字遗忘（不是变量遗忘）作为其特殊情况，如果遗忘结果不变，那么公式中不含遗忘文字。这就是下面的命题。

强独立性和弱独立性的关系是显然的。

命题 5.14: 如果公式φ和ψ是强独立的，那么他们也是弱独立的。

证明: 假设公式φ和ψ是强独立的，那么存在两等价公式$\varphi' \equiv \varphi$和$\psi' \equiv \psi$使得$Var(\varphi') \cap Var(\psi') = \emptyset$。所以$ForgetFor(\varphi',\psi') \equiv \varphi'$，从而得到$ForgetFor(\varphi,\psi) \equiv \varphi$。因此$\varphi$和$\psi$是弱独立的。

与独立性相对的就是相关性概念，我们类似定义强相关和弱相关。如果两个公式的本原蕴含式都是相关的，即在遗忘过程中都需要被删除，那么称为是强相关的。

定义 5.10: φ,ψ是两个命题公式，φ和ψ称为强相关的当且仅当

$$ForgetFor(\varphi,\psi) = \top \text{且} ForgetFor(\psi,\varphi) = \top$$

假设$PI(\varphi) = \{a \vee b, \neg b \vee c, a \vee c\}$和$PI(\psi) = \{a \vee c\}$，那么有$ForgetFor(\varphi,\psi) = \top$和$ForgetFor(\psi,\varphi) = \top$，所以$\varphi$和$\psi$是强相关的。

定义 5.11: 对于两公式φ和ψ，如果他们不是弱独立的，则称φ和ψ是弱相关的。

强相关和弱相关是遗忘的两个极端情况，一个是遗忘结果保持不变，一个是遗忘结果不包含任何知识，即为$\top$。假设$PI(\varphi) = \{a \vee b, \neg b \vee c \vee d, a \vee c\}$，$PI(\psi) = \{a \vee c \vee d\}$，那么显然$ForgetFor(\varphi, \psi) = a \vee b$且$ForgetFor(\psi, \varphi) = \top$，所以这两个公式并不是强相关的，而是弱相关，因为只有一个遗忘结果是$\top$。如果$PI(\psi') = \{a \vee b, d \vee e\}$，那么$ForgetFor(\varphi, \psi') = (\neg b \vee c \vee d) \wedge (a \vee c)$且有$ForgetFor(\psi', \varphi) = d \vee e$，所以$\varphi$和$\psi'$是弱相关的。

如果两个公式强相关，则他们一定是弱相关的。对于子句来说，有下述性质。

命题 5.15: 设C_1, C_2是两个子句，C_1和C_2是强相关的当且仅当$C_1 \models C_2$或者$C_2 \models C_1$。

我们总结一下公式的独立性和相关性，公式之间的关联性可以分为弱独立和弱相关，他们是不相交的，两个公式不可能既是弱相关的又是弱独立的。弱独立包含强独立，弱相关包含强相关。

第六章　遗忘与相关概念的比较

我们研究的知识库通常不是固定不变的，随着时间的推移，智能体可能获得新的知识，也可能丢失已有的知识，或者某些知识不在为真，需要被删除，这都涉及知识管理的过程。知识管理的形式有很多，最常见的是信念修正或收缩、知识更新、信念归并和遗忘。

这一部分分析遗忘和其他知识管理方式的关系与区别，首先来看遗忘和知识更新 [46, 61]，尽管他们是完全不同的概念，但是已经有很成熟的理论。已经证明了，它们在特殊条件下是互可定义的。知识更新和擦除是一组相对的概念，它们都是从模型角度来定义，用$\oplus$，$\ominus$ 分别表示这两种操作。一种定义如下：对于任意知识库K和公式φ，

- $Mod(K \oplus \varphi) = \cup_{\omega \in Mod(K)} min_{\leq_\omega}(Mod(\varphi))$。
- $Mod(K \ominus \varphi) = Mod(K) \cup \cup_{\omega \in Mod(K)} min_{\leq_\omega}(Mod(\neg\varphi))$。

上面两个算子中偏序关系$\leq_\omega$ 并没有具体给出，最广泛使用的是Dalal距离。假设ω，ω' 是两个模型，它们之间Dalal的距离$Dist_D(\omega, \omega')$ 是两个模型指派不同值的原子的个数：$Dist_D(\omega, \omega') = |\{a$是原子变量$|\ \omega \models a$ 当且仅当$\omega' \models \neg a\}|$。

有了上面的距离，可以定义Dalal偏序$\leq_\omega^D$：

$$\omega' \leq_\omega^D \omega'' \text{ 当且仅当} Dist_D(\omega, \omega') \leq Dist_D(\omega, \omega'')$$

称采用Dalal偏序的更新和擦除为Dalal更新和Dalal擦除，分别用$\oplus^D$ 和$\ominus^D$ 表示，则它们和变量遗忘是互可定义的。

命题 6.1: ([46]) 给定一个知识库K 和命题原子a，那么$ForgetVar(K, a) \equiv (K \ominus^D a) \vee (K \ominus^D \neg a)$；$ForgetVar(K, a) \equiv (K \oplus^D a) \vee (K \oplus^D \neg a)$。

下面这个例子验证了这个性质。

例 6.1: 假设$K = (a \wedge \neg b) \vee c$，那么$Mod(K) = \{001101, 100, 111, 011\}$。001表示$a$ 为$false$，b 为$false$，c 为$true$，其他类似。显然$Forget(K, a) \equiv b \rightarrow$

c。根据更新定义得$Mod(K \oplus^D a) \equiv \{100101, 111\}$并且$Mod(K \ominus^D a) \equiv \{001011, 100, 101, 111, 000\}$。因此$K \oplus^D a \equiv a \wedge (b \rightarrow c)$和$K \ominus^D a \equiv b \rightarrow c$。同样地，我们可以计算$K \oplus^D \neg a \equiv \neg a \wedge (b \rightarrow c)$和$K \ominus^D \neg a \equiv (a \wedge \neg b) \vee c \equiv K$。从而我们很容易验证上述定理成立。

不仅可以由知识更新和擦除来定义遗忘，反过来也是成立的。

命题 6.2: ([46]) 对于任意知识库K和原子a，我们有下列结论：

- $K \oplus^D a \equiv ForgetVar(K, a) \wedge a$
- $K \ominus^D a \equiv ForgetVar(K, a) \wedge (a \rightarrow K)$

上面给出了变量遗忘和知识更新或擦除之间的关系，实际上，文字遗忘和基于Dalal距离的擦除是一致的。

命题 6.3: 假设φ是一公式，a是一变元，那么有

$$ForgetLit(\varphi, a) \equiv \varphi \ominus^D a$$

证明: 由上述命题得，$\varphi \ominus^D a \equiv ForgetVar(\varphi, a) \wedge (a \rightarrow \varphi)$
$\equiv ForgetVar(\varphi, a) \wedge (\neg a \vee \varphi)$
$\equiv \varphi \vee \neg a \wedge ForgetVar(\varphi, a)$
$\equiv \varphi \vee (\neg a \wedge (ForgetLit(\varphi, a) \vee ForgetLit(\varphi, \neg a)))$
$\equiv \varphi \vee (\neg a \wedge ForgetLit(\varphi, a)) \vee (\neg a \wedge \varphi)$
$\equiv \varphi \vee (\neg a \wedge (ForgetLit(\varphi, a))$
$\equiv ForgetLit(\varphi, a)$。

容易得出，$K \ominus^D \neg a \equiv ForgetVar(K, a) \wedge (\neg a \rightarrow K)$。类似证明下述命题。

命题 6.4: 假设φ是一公式，a是一变元，那么$ForgetLit(\varphi, \neg a) \equiv \varphi \ominus^D \neg a$。一般地，对于任意文字$l$，有

$$ForgetLit(\varphi, l) \equiv \varphi \ominus^D l$$

既然知识更新和变量遗忘可以互相定义，并且知识更新一般是用命题公式去更新原先的知识库，那么同样我们可以遗忘一个公式。不幸的是，通过把更新和遗忘的关系直接推广到公式遗忘有时不符合我们的直观。比如，我们要在公式$\varphi = a \wedge b \wedge c$中遗忘公式$\psi = a \wedge b$，直观结果应该是$c$。但是由于$\varphi \oplus^D \psi \equiv a \wedge b \wedge c$，$\varphi \oplus^D \neg\psi \equiv (a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c)$，因此$\varphi \oplus^D \psi \vee \varphi \oplus^D \neg\psi \not\equiv c$。第一种定义，$Forget'_V(\varphi, \psi) = ForgetVar(\varphi, \{a, b\}) = c$，同理$Forget'_L(\varphi, \psi) = c$；第二类定义，$Forget_F(\varphi, \psi) = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge c) = (\varphi \oplus^D \psi) \vee (\varphi \oplus^D \neg\psi)$，而$Forget_M(\varphi, \psi) = (b \wedge c) \vee (a \wedge \neg b \wedge c) = (\varphi \oplus^D \psi) \vee (\varphi \oplus^D \neg\psi)$，因为$Margin(a \wedge b) = a \wedge b$，$Margin(\neg a \neg b) = (a \wedge \neg b) \vee (\neg a \wedge b)$；第三类基于本原蕴涵式的定义，$ForgetFor(\varphi, \psi) = c$。尽管$Forget_F, Forget_M$偶然等价于$(\varphi \oplus^D \psi) \vee (\varphi \oplus^D \neg\psi)$，但是没有迹象表明这个结论一般任意情况下都是成立的。

对于$Forget_F$算子来说，它实际上是Dalal擦除算子。尽管这两个定义用不同的方式定义，遗忘是用$Force$，而擦除是用Dalal偏序$\leq^D_\omega$，但是它们都是对于φ的每一个模型ω，在$\neg\psi$的模型中选择距离ω最近的模型作为新模型，再合并φ原有的模型得到我们的结果，所以这两个等价。

命题 6.5: 假设φ, ψ是两个公式，那么有

$$Forget_F(\varphi, \psi) \equiv \varphi \ominus^D \psi$$

证明: 证明略。

除$Forget_F$外，其余遗忘算子一般和擦除算子是不同的。从第三章给出的公设可以看出，遗忘和擦除的主要区别有：第一，(E2)指的是如果原先知识库蕴含某公式的反面，那么擦除这个公式不会产生任何影响，即擦除结果不变。而对于遗忘来说，我们目前知道，对于两个公式，如果它们不共享公共原子变量，那么遗忘结果不会产生变化。但是也可能两个公式有相同的变元，但是遗忘结果也不变。第二，知识擦除具有恢复性，即如果再加入被擦除的信息，那么我们得到原先的知识库，这种性质对于遗忘是不满足的。例如，$ForgetFor(a \vee b \vee c, a \vee b) \equiv \top$，所以$ForgetFor(a \vee b \vee c, a \vee b) \wedge (a \vee b) \equiv a \vee b$。第三，擦除满足析取分配律，文字遗忘也满足分配律，但是公式遗忘一般不满足。

当然它们也有共同点：擦除和遗忘都是对原先知识库的弱化，即最初知识库肯定逻辑推导出经过算子计算的结果。而且对于非重言式公式，遗忘和擦除的结果都不会蕴含这个公式。还有，它们都满足语法独立性。

下面从公设比较信念收缩和遗忘的异同。相同点与上面类似。不同的是，(C2)条件过强，这条强迫收缩算子对于相当大的一部分情况，收缩不会产生作用，遗忘并不满足。比如，$ForgetFor(a \vee b \vee c, a \vee b) \equiv \top$，尽管$a \vee b \vee c \not\models a \vee b$。收缩同样有恢复性，无论是变量还是文字遗忘都不满足此性质，所以推广后的公式遗忘也应该不满足。

在 [36]中提出用遗忘来解决冲突矛盾，并给出一个框架。公式之间存在矛盾可以被认为过分定义，遗忘是过滤某些原子且结果比原先更弱，这为解决冲突准备了理论基础。首先需要遗忘上下文（forgetting context）的概念规定变量的依赖关系，即类似某些变量被遗忘另外一些也必须被遗忘这样的知识。

定义 6.1: （ [36]）知识库$B = \{\varphi_1, \cdots, \varphi_n\}$的遗忘上下文$\mathcal{C}$是一个三元组$\mathcal{C} =< F, G, H >$，满足下面三条：

- $F =< F_i >_{i=1\cdots n}$，对于每一个i，F_i是在φ_i可以被遗忘的变量集（即不属于F_i的变量不能被遗忘）。
- $G =< G_i >_{i=1\cdots n}$，对于每个i，G_i是F_i上的有向图。任意G_i中的弧(v, v')表示，V'只有在v出现的前提下有意义，即如果在φ中遗忘v，那么v'也应被遗忘。
- $H =\subseteq PS \times \{1 \cdots n\} \times \{1 \cdots n\}$ 是三元组v, i, j的集合，使得对于$v \in F_i \cap F_j$且$i \neq j$，有如果在φ_i中遗忘v，那么也应在φ_j中遗忘它。

遗忘上下文有三种主要形式，一种是各公式遗忘相同的原子，称为同构（homogeneous）上下文，即对于任意i，有$F_i = F_j$且$G_i = G_j$。第二种是标准（standard）上下文，即$F_i = PS$且$G_i = \emptyset$。这种上下文对遗忘变量没有限制，任何变量都可被公式遗忘，不同公式可以遗忘不同变量。最后一种是二元（binary）上下文，即对于某个变量集V，如果V中某个变量被公式φ_i遗忘，那么V中其他变量也应被遗忘。

接下来给出一个变量集构成的向量，维数与知识库公式数目相同，每一个分量对应相应公式要遗忘的变量集。我们把这向量称为修复（recovery），因为在知识库的每个公式中遗忘恢复中相应的变量集达到协调性，下面是它的定义。

定义 6.2: （ [36]）假设$B = \{\varphi_1, \cdots, \varphi_n\}$是一知识库，$\mathcal{C}$是遗忘上下文，$B$的恢复是一向量$\overrightarrow{V} =< V_1, \cdots, V_n >$，每个$V_i$都是一变量集，满足：

- 任意i，有$V_i \subseteq F_i$
- 任意i，$(v, v') \in G_i$，有$v \in V_i$ 推出$v' \in V_i$
- 任意$(v, i, j) \in H$，$v \in V_i$推出$v \in V_j$
- 公式集合$B|\overrightarrow{V} = \{\exists V.\varphi_i | i = 1 \cdots n\}$是协调的

$B|\overrightarrow{V}$称为恢复在B上的投影，$\mathcal{R}_{\mathcal{C}}(B)$代表所有恢复的集合，$B$ 是可恢复的当且仅当$\mathcal{R}_{\mathcal{C}}(B) \neq \emptyset$。

有好多恢复包含过多的变量，为了能更少的改变原先知识库，选择包含变量尽可能少的恢复取得投影，并由此进行推理。我们用遗忘做归并，就是要求所有公式过滤相同的变量并且尽可能少的遗忘，从这个角度，我们的方法属于同构上下文。在大部分归并问题中，变量之间没有必然的因果关系，而且矛盾通常发生在某个变量子集上，这样对其余变量上面的推理不应该受这种矛盾的影响。最后选择极小（关于元素数或者集合包含关系）的恢复作为遗忘的变量集，这样的恢复可能不止一个，取在各个恢复上投影的析取作为归并的结果。在归并问题中，多个信息源的知识本身应被看做独立的知识库，一味的追求整体的协调性未必是正确的。适当地单独处理这些知识库，从而避开了整体的矛盾性，是有帮助的。不管是上述框架还是其他的归并算子，都没有考虑性质(A1)和(A2)，这使得结果并不那么直观，我们提出的两个算子恰能满足这两个性质。

第七章　总结与展望

遗忘是知识管理的一种重要形式，主要用来处理某些知识随着时间变化不再为真、子语言查询、解决矛盾以保留协调性等问题，而且可以通过遗忘无关信息来提高推理的效率。尽管人们给出了命题逻辑、模态逻辑、描述逻辑以及逻辑程序上遗忘的概念，但它们实际上都属于变量遗忘或文字遗忘的范畴，即在遗忘结果中不包含要遗忘的变量或文字。这种遗忘形式还不能模拟人类日常生活中的遗忘，因为我们遗忘的知识用逻辑表示通常是一个公式或者语句，本文正是把已有的遗忘推广到公式上。

类似于其他知识管理的方式，我们先给出表征遗忘的5个公设。首先，遗忘是对原公式的弱化，即遗忘是知识减少的过程。其次，遗忘过程不应对不在遗忘公式中出现的符号产生影响，因为遗忘一个公式顶多就是遗忘这个公式中出现的所有文字，对建立在其它变元上的公式原先能推出，遗忘之后还能推出，原先不能推出，遗忘之后也推不出。第三，对于不是重言式的任意公式ψ，遗忘ψ之后，结果不会再蕴含它，这是人类遗忘的本质，即忘记的东西就不再被记忆。第四，遗忘与公式的形式无关，它是面向语义的，等价的公式遗忘等价的公式结果也是等价的。最后一个公设，遗忘是稳定的过程，遗忘了就不再被拥有，多次遗忘并不会产生额外的作用。

接下来本文给出了几种遗忘的定义。第一种定义是直接把遗忘转化为遗忘公式中出现的变量或文字，这是一种最弱的形式，但是它不能区分公式的形式。比如，遗忘$a \wedge b$和$a \vee b$结果是等价的，因为它们包含相同的文字，但是却有不同的语义。但是对于任意公式，它可能包含一些多余的文字，所以为了满足遗忘公设的语法独立性，我们定义了变量或文字最简公式，使我们只考虑被遗忘公式的最简公式中包含的文字。这样5个遗忘公设都能得到满足。

既然文字遗忘的模型如下所示

$$Mod(ForgetLit(\varphi, l)) = Mod(\varphi) \vee \{Force(\omega, \neg l) | \omega \models \varphi\}$$

那么只需要把$Force$从$Force(\omega, \neg l)$推广到$Force(\omega, \neg\psi)$，我们就可以类似得到公式遗忘的模型为

$$Mod(ForgetLit(\varphi, \psi)) = Mod(\varphi) \cup_{\omega \models \varphi} Force(\omega, \neg\psi)$$

上式中$Force(\omega, \neg\psi)$一般是模型的集合，而$Force(\omega, \neg l)$只是一个模型。第二种通过模型定义公式遗忘$Forget_M$是取被遗忘公式及其否定公式的部分边缘模型，即

$$Mod(Forget_M(\phi, \psi)) = Mod(\phi) \cup \{\omega \mid \omega \in Mod(Margin(\psi))，且Dis(\omega, \phi)最小\} \cup \{\omega \mid \omega \in Mod(Margin(\neg\psi))，且Dis(\omega, \phi)最小\}$$

这两种语义定义都满足5个遗忘公设。

因为变量或文字遗忘都相当于在本原蕴涵式集合中删除包含遗忘变量或文字的子句，剩余本原蕴涵式的合取就是等价于遗忘结果。最后一种公式遗忘就是从本原蕴涵式角度定义的。本原蕴涵式本身是子句形式，所以先给出子句遗忘的定义。首先在本原蕴涵式集合中删除包含遗忘子句的本原蕴涵式，其次再删除被遗忘子句包含的本原蕴涵式（因为这部分子句的存在使得原公式能够蕴含遗忘子句），这还远远不够，因为剩余的本原蕴涵式还可能消解出已经删除的本原蕴涵式，所以还得删除。为了以一种合理的方式逐步删除这些相关子句，我们在消解闭包上执行删除操作。子句集遗忘就可以定义为删除所有与子句集S相关的子句，即

$$ForgetClas(\varphi, S) \equiv \bigwedge \bigcap_{C \in S} (CPI(\varphi) - FS(\varphi, C))$$

最后因为任意公式都等价于其PI的合取，而且在等价条件是一一对应的，那么公式遗忘就可定义为遗忘公式的本原蕴涵式集合。这种公式遗忘尽管可以看做是语法的角度，但是满足语法独立性，而且其他的遗忘公设也都满足。

遗忘的应用很广泛，我们考虑了用遗忘做归并的问题。通过扩张做中间媒介，我们可以把已有的归并算子表示为遗忘的形式。通过研究，已有的归并算子在解决某些问题时不够直观。比如，如果某些信息源支持原子公式l，其他信息源对l没有任何陈述（即不出现l对应的原子），那么归并的结果应该支持公式l。另一方面，如果某些知识库支持l，另外一些反对l，那么对l持谨慎态度，也就是既不支持它也不反对它。知识归并中的各个知识库本来可看做是相互独立的，我们并不能把它等价于一个由所有知识库知识构成的一个大的知识库。如果我们的查询知识包含一部分原子符号，那么在各个知识库中遗忘其他的变量是有意义的。它体现在两个方面：一是所有知识库放到一起可能是矛盾的，但是仅在查询所考虑的论域内是协调的；二是这对于提高查询效率是有帮助

的。这样的改进后的归并算子要求各个知识库遗忘相同的变元集合，并且在保证协调的条件下是最小的。这种最小既可以是关于集合元素数也可以是关于集合包含关系的，所得到归并算子都满足上面两条性质。第二个应用是考虑公式之间的独立性和相关性。已经提出了公式与变量或文字的独立性（相关性），而且他们和变量或文字遗忘有紧密关系。有了公式遗忘我们同样可以定义公式独立性（相关性）。如果公式φ遗忘ψ结果还是φ，也就是ψ对φ没有影响，我们称两者是弱独立的；如果φ遗忘ψ为$\top$且ψ遗忘φ为$\top$，也就是一个遗忘另一个，结果不包含任何知识，那么φ和ψ是强相关的。

最后从公设和算子本身比较了遗忘和其他知识管理形式之间的异同。未来的一个工作是，质蕴含式是和本原蕴含式相对的概念，遗忘和本原蕴含式的关系能不能发展为和质蕴涵式的关系，从而从质蕴涵角度定义新的公式遗忘，当然我们期望这两类定义能够是一致的。等价公式都有唯一的质蕴涵式集合，即$IP(\varphi) = \{t_1, \cdots, t_n\}$，遗忘表现为对这个集合的操作还不明确。我们不能类似地删除与遗忘公式相关的质蕴含项，因为$\varphi \equiv t_1 \vee \cdots \vee t_n$，去掉某些项之后得到的公式会比$\varphi$更强，而本原蕴含式集合被解释为合取，删除某些子句是对原公式的弱化。一个直观的想法是增加新的质蕴含式，增加项比删除子句要更困难一些。

求解本原蕴含式已经有许多改进算法，我们公式遗忘的定义无疑比它更复杂，如何给出一个高效的算法是我们今后考虑的问题。这种构造性的定义关键是要找到与遗忘公式相关的遗忘集，而遗忘集是通过在CPI上递归求得的。首先在CPI中找到和遗忘公式直接相关的子句，这需要$|CPI|$的线性时间，接下来，在剩余子句中快速找到能够消解出已选择的子句的子句是改进算法的一个关键。我们通过本原蕴含式的定义，在语义上还不是很清楚。它符合语法独立性，但是这种本原蕴含式的操作反映在语义模型上是怎样变化的，这是我们下一个要考虑的问题。

参考文献

[1] J. Liebowitz. Knowledge management handbook[M]. CRC, 1999

[2] M. Alavi, D.E. Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues[J]. MIS quarterly. 2001:107–136

[3] A.H. Gold, A. Malhotra, A.H. Segars. Knowledge management: An organizational capabilities perspective[J]. Journal of Management Information Systems. 2001, **18**(1):185–214

[4] R. Ruggles. The state of the notion: knowledge management in practice[J]. The Knowledge Management Yearbook 1999-2000. 1999:295

[5] W.R. Bukowitz, R.L. Williams. The knowledge management fieldbook[M]. Financial Times Prentice Hall, 1999

[6] R. Booth, S. Chopra, T. Meyer, A. Ghose. A unifying semantics for belief change[C]. ECAI. Citeseer, 2004, vol. 16, 793

[7] A. Grove. Two modellings for theory change[J]. Journal of Philosophical logic. 1988, **17**(2):157–170

[8] C.E. Alchourrón, P. Gärdenfors, D. Makinson. On the logic of theory change: Partial meet contraction and revision functions[J]. Journal of symbolic logic. 1985, **50**(2):510–530

[9] M. Dalal. Investigations into a theory of knowledge base revision: Preliminary report[C]. Proceedings of the Seventh National Conference on Artificial Intelligence. Citeseer, 1988, vol. 2, 475–479

[10] O. Katsuno Alberto, et al. Propositional knowledge base revision and minimal change[J]. Artificial Intelligence. 1991, **52**(3):263–294

[11] D. Dubois, H. Prade. Belief change and possibility theory[J]. Belief Revision. 1992:142–182

[12] H. Rott. Preferential belief change using generalized epistemic entrenchment[J]. Journal of Logic, Language and Information. 1992, **1**(1):45–78

[13] J.P. Delgrande, T. Schaub. A consistency-based approach for belief change[J]. Artificial Intelligence. 2003, **151**(1-2):1–41

[14] N. Friedman, J.Y. Halpern. A knowledge-based framework for belief change part I: foundations[C]. Proceedings of the 5th conference on Theoretical aspects of reasoning about knowledge. Morgan Kaufmann Publishers Inc., 1994, 44–64

[15] A.C. Nayak. Foundational belief change[J]. Journal of Philosophical Logic. 1994, **23**(5):495–533

[16] M.S. Winslett. Updating logical databases[M]. Cambridge Univ Pr, 1990

[17] R. Fagin, G.M. Kuper, J.D. Ullman, M.Y. Vardi, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE. Updating logical databases[J]. Advances in computing Research. 1986, **3**:1–18

[18] A. Herzig, O. Rifi. Propositional belief base update and minimal change[J]. Artificial Intelligence. 1999, **115**(1):107–138

[19] H. Katsuno, A.O. Mendelzon. On the Difference between Updating a Knowledge Base and Revising it[J]. Belief revision. 2003:183

[20] M. Lenzerini, A. Poggi, R. Rosati. On Instance-Level Update and Erasure in Description Logic Ontologies[J]. 2009

[21] S. Konieczny, R.P. Pérez. Merging information under constraints: a logical framework[J]. Journal of Logic and Computation. 2002, **12**(5):773

[22] P. Liberatore, M. Schaerf. Arbitration (or how to merge knowledge bases)[J]. Knowledge and Data Engineering, IEEE Transactions on. 2002, **10**(1):76–90

[23] S. Benferhat, C. Sossai. Reasoning with multiple-source information in a possibilistic logic framework[J]. Information Fusion. 2006, **7**(1):80–96

[24] F. Lin, R. Reiter. Forget it[C]. Working Notes of AAAI Fall Symposium on Relevance. 1994, 154–159

[25] J. éertome Lang, P. Liberatore, U.Î. di Roma, P. Marquis. Propositional independence: Formula-variable independence and forgetting[J]. Journal of Artificial Intelligence Research. 2003, **18**:391–443

[26] Y. Zhang, Y. Zhou. Knowledge forgetting: Properties and applications[J]. Artificial Intelligence. 2009, **173**(16-17):1525–1537

[27] T. Eiter, K. Wang. Forgetting and conflict resolving in disjunctive logic programming[C]. Proceedingsof the National Conference on Artificial Intelligence. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, vol. 21, 238

[28] G. Qi, W. Liu, D.A. Bell. Merging stratified knowledge bases under constraints[C]. PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, vol. 21, 281

[29] J. Lin. Integration of weighted knowledge bases[J]. Artificial Intelligence. 1996, **83**(2):363–378

[30] A. Yue, W. Liu, A. Hunter. Approaches to constructing a stratified merged knowledge base[J]. Symbolic and Quantitative Approaches to Reasoning with Uncertainty. 2007:54–65

[31] S. Benferhat, D. Dubois, H. Prade, M.A. Williams. A practical approach to fusing prioritized knowledge bases[J]. Progress in Artificial Intelligence. 1999:851–851

[32] G. Qi, J.Z. Pan, et al. A stratification-based approach for inconsistency handling in description logics[C]. International Workshop on Ontology Dynamics (IWOD-07). Citeseer, 2007, 83

[33] G. Boole, J. Corcoran. The laws of thought[M]. Prometheus Books, 2003

[34] K. Su, G. Lv, Y. Zhang. Reasoning about knowledge by variable forgetting[J]. Proceedings of KR2004. 2004:576–586

[35] G. Qi, Y. Wang, P. Haase, P. Hitzler. A Forgetting-based Approach for Reasoning with Inconsistent Distributed Ontologies[M]. Citeseer, 2008

[36] J. Lang, P. Marquis. Resolving inconsistencies by variable forgetting[C]. Principles of Knowledge Representation and reasoning-International Conference-. Morgan Kaufmann Publishers; 1998, 2002, 239–250

[37] J. Lang, P. Marquis. Reasoning under inconsistency: A forgetting-based approach[J]. Artificial Intelligence. 2010, **174**(12-13):799–823

[38] J. Lang, P. Liberatore, P. Marquis. Conditional independence in propositional logic[J]. Artificial Intelligence. 2002, **141**(1-2):79–121

[39] T. Eiter, K. Wang. Semantic forgetting in answer set programming[J]. Artificial Intelligence. 2008, **172**(14):1644–1672

[40] Y. Zhang, N. Foo, K. Wang. Solving logic program conflict through strong and weak forgettings[C]. International Joint Conference on Artificial Intelligence. Citeseer, 2005, vol. 19, 627

[41] Y. Moinard. Forgetting literals with varying propositional symbols[J]. Journal of Logic and Computation. 2007, **17**(5):955

[42] K. Su, A. Sattar, G. Lv, Y. Zhang. Variable forgetting in reasoning about knowledge[J]. Journal of Artificial Intelligence Research. 2009, **35**(1):677–716

[43] Y. Zhang, N.Y. Foo. Solving logic program conflict through strong and weak forgettings[J]. Artificial Intelligence. 2006, **170**(8-9):739–778

[44] T. Eiter, K. Wang. Semantic Forgetting in Answer Set Programming[J]. 2007

[45] B. Konev, D. Walther, F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies[C]. Proc. IJCAI. 2009, vol. 9

[46] A. Nayak, Y. Chen, F. Lin. Forgetting and knowledge update[J]. AI 2006: Advances in Artificial Intelligence. 2006:131–140

[47] P. Gärdenfors. Belief revision: An introduction[J]. Belief revision. 1992, **29**:1–28

[48] P. Gärdenfors. Knowledge in flux: modeling the dynamics of epistemic states[J]. 1988

[49] V. Lifschitz. Answer set programming and plan generation[J]. Artificial Intelligence. 2002, **138**(1-2):39–54

[50] Michael Gelfond, Vladimir Lifschitz. The Stable Model Semantics for Logic Programming[C]. ICLP/SLP. 1988, 1070–1080

[51] Vladimir Lifschitz, Thomas Y. C. Woo. Answer Sets in General Nonmonotonic Reasoning (Preliminary Report)[C]. KR. 1992, 603–614

[52] Yuliya Babovich, Esra Erdem, Vladimir Lifschitz. Fages' Theorem and Answer Set Programming[J]. CoRR. 2000, **cs.AI/0003042**

[53] M. Gebser, B. Kaufmann, A. Neumann, T. Schaub. Conflict-driven answer set solving[C]. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI' 07). 2007, 386–392

[54] S. Muggleton, L. De Raedt. Inductive logic programming: Theory and methods[J]. The Journal of Logic Programming. 1994, **19**:629–679

[55] J. Jaffar, M.J. Maher. Constraint logic programming: A survey[J]. The Journal of Logic Programming. 1994, **19**:503–581

[56] H. Blair, V. Subrahmanian. Paraconsistent logic programming[C]. Foundations of Software Technology and Theoretical Computer Science. Springer, 1987, 340–360

[57] E. Giunchiglia, Y. Lierler, M. Maratea. Answer set programming based on propositional satisfiability[J]. Journal of Automated Reasoning. 2006, **36**(4):345–377

[58] F. Lin, Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers[J]. Artificial Intelligence. 2004, **157**(1-2):115–137

[59] A. Van Gelder, K.A. Ross, J.S. Schlipf. The well-founded semantics for general logic programs[J]. Journal of the ACM (JACM). 1991, **38**(3):619–649

[60] R. Fagin. Reasoning about knowledge[M]. The MIT Press, 2003

[61] C. Baral, Y. Zhang. Knowledge updates: semantics and complexity issues[J]. Artificial Intelligence. 2005, **164**(1-2):209–243

[62] H. van Ditmarsch, A. Herzig, J. Lang, P. Marquis. Introspective forgetting[J]. Synthese. 2009, **169**(2):405–423

[63] T. French. Bisimulation quantified logics: undecidability[J]. FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science. 2005:396–407

[64] S. Ghilardi, M. Zawadowski. Undefinability of propositional quantifiers in the modal system S4[J]. Studia Logica. 1995, **55**(2):259–271

[65] J. Lee, V. Lifschitz. Loop formulas for disjunctive logic programs[J]. Logic Programming. 2003:451–465

[66] P. Jackson, J. Pais. Computing prime implicants[C]. 10th International Conference on Automated Deduction. Springer, 1990, 543–557

[67] P. Jackson. Computing prime implicates incrementally[J]. Automated Deduction—CADE-11. 1992:253–267

[68] A. Kean, G. Tsiknis. An incremental method for generating prime implicants/implicates[J]. Journal of Symbolic Computation. 1990, **9**(2):185–206

[69] J. De Kleer. An improved incremental algorithm for generating prime implicates[C]. Proceedings of the National Conference on Artificial Intelligence. JOHN WILEY & SONS LTD, 1992, 780–780

[70] A. Singh. Computing prime implicants via transversal clauses[J]. International Journal of Computer Mathematics. 1999, **70**(3):417–427

[71] J.R. Slagle, C.L. Chang, R.C.T. Lee. A new algorithm for generating prime implicants[J]. Computers, IEEE Transactions on. 2006, **100**(4):304–310

[72] Tadeusz Strzemecki. Polynomial-time algorithms for generation of prime implicants[J]. J Complexity. 1992, **8**(1):37–63

[73] J.P.M. Silva. On computing minimum size prime implicants[C]. International Workshop on Logic Synthesis. Citeseer, 1997

[74] A. Ramesh, G. Becker, N.V. Murray. CNF and DNF considered harmful for computing prime implicants/implicates[J]. Journal of Automated Reasoning. 1997, **18**(3):337–356

[75] A. Ramesh, N. Murray. Non-clausal deductive techniques for computing prime implicants and prime implicates[C]. Logic Programming and Automated Reasoning. Springer, 1993, 277–288

[76] R. Schrag, J.M. Crawford. Implicates and prime implicates in random 3-SAT[J]. Artificial Intelligence. 1996, **81**(1-2):199–222

[77] A. Kean, G. Tsiknis. Assumption-based reasoning and clause management systems[J]. Computational Intelligence. 1992, **8**(1):1–24

[78] M. Bienvenu. Prime implicates and prime implicants: from propositional to modal logic[J]. Journal of Artificial Intelligence Research. 2009, **36**(1):71–128

[79] M. Bienvenu, et al. Prime implicate normal form for ALC concepts[C]. Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI’08). 2008, 412–417

[80] G. Bittencourt. Combining syntax and semantics through prime form representation[J]. Journal of Logic and Computation. 2008, **18**(1):13

[81] P. Marquis. Knowledge compilation using theory prime implicates[C]. International Joint Conference on Artificial Intelligence. 1995, vol. 14, 837–845

[82] Z.Q. Zhuang, M. Pagnucco, T. Meyer. Implementing iterated belief change via prime implicates[C]. Proceedings of the 20th Australian joint conference on Advances in artificial intelligence. Springer-Verlag, 2007, 507–518

[83] D. Rajaratnam, M. Pagnucco. Prime implicates for approximate reasoning[C]. Proceedings of the 2nd international conference on Knowledge science, engineering and management. Springer-Verlag, 2007, 61–72

[84] T.C. Przymusinski. An algorithm to compute circumscription[J]. Artificial intelligence. 1989, **38**(1):49–73

[85] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade. Inconsistency management and prioritized syntax-based entailment[C]. International Joint Conference on Artificial Intelligence. Citeseer, 1993, vol. 13, 640–640

[86] P. Everaere, S. Konieczny, P. Marquis. Disjunctive merging: Quota and Gmin merging operators[J]. Artificial Intelligence. 2010

[87] S. Konieczny, R. Pino-Pérez. On the logic of merging[J]. 1998

[88] S. Konieczny, J. Lang, P. Marquis. DA 2 merging operators[J]. Artificial Intelligence. 2004, **157**(1-2):49–79

[89] P. Maynard-Zhang, D. Lehmann. Representing and aggregating conflicting beliefs[J]. Journal of Artificial Intelligence Research. 2003, **19**(1):155–203

[90] N. Gorogiannis, A. Hunter. Merging first-order knowledge using dilation operators[C]. Proceedings of the 5th international conference on Foundations of information and knowledge systems. Springer-Verlag, 2008, 132–150

[91] G. Sandu. On the logic of informational independence and its applications[J]. Journal of Philosophical Logic. 1993, **22**(1):29–60

[92] G. Sandu. The logic of informational independence and finite models[J]. Logic Journal of IGPL. 1997, **5**(1):79

[93] J. Väänänen. On the semantics of informational independence[J]. Logic Journal of IGPL. 2002, **10**(3):339

[94] M. Sevenster. On the computational consequences of independence in propositional logic[J]. Synthese. 2006, **149**(2):257–283

[95] J. Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference[M]. Morgan Kaufmann, 1988

[96] E. Gregoire, P. Marquis. Novelty in deductive databases[J]. Journal of Logic and Computation. 1996, **6**(5):683

[97] F. Cuppens, R. Demolombe. How to recognize interesting topics to provide cooperative answering[J]. Information Systems. 1989, **14**(2):163–173

[98] A.Y. Levy, R.E. Fikes, E.A. Feigenbaum, P.J. Hayes, N.J. Nilsson, Y.C. Sagiv. Irrelevance reasoning in knowledge based systems[J]. 1993

[99] R. Pearl J. Subramanian D., Greiner. Artificial Intelligence Journal: Special Issue on Relevance[J]. Artificial Intelligence Journal. 1997, **97(1-2)**

[100] A. Darwiche. A logical notion of conditional independence: properties and applications[J]. Artificial Intelligence. 1997, **97**(1-2):45–82